

Sun Performance Library Reference



THE NETWORK IS THE COMPUTER™

SunSoft, Inc.

A Sun Microsystems, Inc. Business
2550 Garcia Avenue
Mountain View, CA 94043 USA
415 960-1300 fax 415 969-9131

Part No.: 802-6439-10
Revision A, December 1996

Copyright 1996 Sun Microsystems, Inc., 2550 Garcia Avenue, Mountain View, California 94043-1100 U.S.A. All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any.

Portions of this product may be derived from the UNIX[®] system, licensed from Novell, Inc., and from the Berkeley 4.3 BSD system, licensed from the University of California. UNIX is a registered trademark in the United States and other countries and is exclusively licensed by X/Open Company Ltd. Third-party software, including font technology in this product, is protected by copyright and licensed from Sun's suppliers.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227-14(g)(2)(6/87) and FAR 52.227-19(6/87), or DFAR 252.227-7015(b)(6/95) and DFAR 227.7202-3(a).

Sun, Sun Microsystems, the Sun logo, SunSoft, Solaris, Sun WorkShop, Sun Performance WorkShop, Sun Performance Library, Sun Visual WorkShop, and Sun WorkShop TeamWare, are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the United States and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK[®] and Sun[™] Graphical User Interfaces were developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

THIS PUBLICATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.



Contents

1. LAPACK Subroutines	1
Naming Conventions	3
Error Handling.....	6
Extensions to LAPACK.....	7
Reciprocal Condition Number of an LU-Factored General Matrix in Banded Storage	8
Equilibration Scale Factors for a General Matrix in Banded Storage 12	
Refined Solution to a Linear System in an LU-Factored General Matrix in Banded Storage.....	15
Solution to a Linear System in a General Matrix in Banded Storage (Simple Driver).....	21
Solution to a Linear System in a General Matrix in Banded Storage (Expert Driver).....	25
LU Factorization of a General Matrix in Banded Storage	33
Solution to a Linear System in an LU-Factored General Matrix in Banded Storage	37
Reciprocal Condition Number of an LU-Factored General Matrix	41

Equilibration Scale Factors for a General Matrix	44
Eigenvalues, Schur Form, and Schur Factorization of a General Matrix (Simple Driver)	48
Eigenvalues, Schur Form, and Schur Factorization of a General Matrix (Expert Driver)	53
Eigenvalues and Eigenvectors of a General Matrix (Simple Driver) 59	
Eigenvalues, Eigenvectors, and Reciprocal Condition Numbers of a General Matrix (Expert Driver)	63
Generalized Eigenvalues, Schur Form, and Schur Vectors of General Matrices	69
Generalized Eigenvalues and Eigenvectors for General Matrices	74
LQ Factorization of a General Matrix	79
Solution to a Least Squares Problem for a Full-Rank General Matrix (Simple Driver).	82
Solution to a Least Squares Problem for a Rank-Deficient General Matrix (Simple Driver)	86
Solution to a Least Squares Problem for a Rank-Deficient General Matrix (Expert Driver)	90
QL Factorization of a General Matrix	94
QR Factorization with Column Pivoting of a General Matrix .	97
QR Factorization of a General Matrix	100
Refined Solution to a Linear System in an LU-Factored General Matrix	103
RQ Factorization of a General Matrix	108
Solution to a Linear System in a General Matrix (Simple Driver)	111
Singular Value Decomposition of a General Matrix	114

Solution to a Linear System in a General Matrix (Expert Driver)	119
LU Factorization of a General Matrix	126
Inverse of an LU-Factored General Matrix	129
Solution to a Linear System in an LU-Factored General Matrix	132
Solution to a Generalized Linear Regression Model Problem .	135
Solution to a Linear Equality Constrained Least Squares Problem	139
Generalized Singular Value Decomposition of General Matrices	143
Reciprocal Condition Number of an LU-Factored Tridiagonal Matrix	149
Refined Solution to a Linear System in an LU-Factored General Tridiagonal Matrix	153
Solution to a Linear System in a General Tridiagonal Matrix (Simple Driver).	160
Solution to a Linear System in a General Tridiagonal Matrix (Expert Driver).	163
LU Factorization of a Tridiagonal Matrix	169
Solution to a Linear System in an LU-Factored General Tridiagonal Matrix	173
Eigenvalues and Eigenvectors of a Hermitian Matrix in Banded Storage (Simple Driver).	177
Eigenvalues and Eigenvectors of a Hermitian Matrix in Banded Storage (Expert Driver).	181
Reciprocal Condition Number of a UDU- or LDL-Factored Hermitian Matrix	186
Eigenvalues and Eigenvectors of a Hermitian Matrix (Simple Driver).	190

Eigenvalues and Eigenvectors of a Hermitian Matrix (Expert Driver).....	193
Generalized Eigenvalues and Eigenvectors for Hermitian Matrices	198
Refined Solution to a Linear System in a UDU- or LDL-Factored Hermitian Matrix.....	203
Solution to a Linear System in a Hermitian Matrix (Simple Driver)	209
Solution to a Linear System in a Hermitian Matrix (Expert Driver)	213
UDU or LDL Factorization of a Hermitian Matrix	218
Inverse of a UDU- or LDL-Factored Hermitian Matrix.....	222
Solution to a Linear System in a UDU- or LDL-Factored Hermitian Matrix	225
Reciprocal Condition Number of a UDU- or LDL-Factored Hermitian Matrix in Packed Storage.....	229
Eigenvalues and Eigenvectors of a Hermitian Matrix in Packed Storage (Simple Driver).....	233
Eigenvalues and Eigenvectors of a Hermitian Matrix in Packed Storage (Expert Driver).....	236
Generalized Eigenvalues and Eigenvectors for Hermitian Matrices in Packed Storage.....	240
Refined Solution to a Linear System in a UDU- or LDL-Factored Hermitian Matrix in Packed Storage.....	244
Solution to a Linear System in a Hermitian Matrix in Packed Storage (Simple Driver).....	250
Solution to a Linear System in a Hermitian Matrix in Packed Storage (Expert Driver).....	253

UDU or LDL Factorization of a Hermitian Matrix in Packed Storage.....	258
Inverse of a UDU- or LDL-Factored Hermitian Matrix in Packed Storage.....	262
Solution to a Linear System in a UDU- or LDL-Factored Hermitian Matrix in Packed Storage	265
Generation of Q Defined in LQ Factorization	269
Generation of Q Defined in QL Factorization	272
Generation of Q Defined in QR Factorization	275
Generation of Q Defined in RQ Factorization	278
Reciprocal Condition Number of a Cholesky-Factored Symmetric Positive Definite Matrix in Banded Storage	281
Equilibration Scale Factors for a Symmetric Positive Definite Matrix in Banded Storage	284
Refined Solution to a Linear System in a Cholesky-Factored Symmetric Positive Definite Matrix in Banded Storage ...	288
Solution to a Linear System in a Symmetric Positive Definite Matrix in Banded Storage (Simple Driver)	294
Solution to a Linear System in a Symmetric Positive Definite Matrix in Banded Storage (Expert Driver)	297
Cholesky Factorization of a Symmetric Positive Definite Matrix in Banded Storage	303
Solution to a Linear System in a Cholesky-Factored Symmetric Positive Definite Matrix in Banded Storage	307
Reciprocal Condition Number of a Cholesky-Factored Symmetric Positive Definite Matrix	311
Equilibration Scale Factors for a Symmetric Positive Definite Matrix	314

Refined Solution to a Linear System in a Cholesky-Factored Symmetric Positive Definite Matrix.	318
Solution to a Linear System in a Symmetric Positive Definite Matrix (Simple Driver).	323
Solution to a Linear System in a Symmetric Positive Definite Matrix (Expert Driver).	326
Cholesky Factorization of a Symmetric Positive Definite Matrix	332
Inverse of a Cholesky-Factored Symmetric Positive Definite Matrix	335
Solution to a Linear System in a Cholesky-Factored Symmetric Positive Definite Matrix	338
Reciprocal Condition Number of a Cholesky-Factored Symmetric Positive Definite Matrix in Packed Storage	342
Equilibration Scale Factors for a Symmetric Positive Definite Matrix in Packed Storage.	345
Refined Solution to a Linear System in a Cholesky-Factored Symmetric Positive Definite Matrix in Packed Storage ...	349
Solution to a Linear System in a Symmetric Positive Definite Matrix in Packed Storage (Simple Driver).	354
Solution to a Linear System in a Symmetric Positive Definite Matrix in Packed Storage (Expert Driver).	357
Cholesky Factorization of a Symmetric Positive Definite Matrix in Packed Storage.	363
Inverse of a Cholesky-Factored Symmetric Positive Definite Matrix in Packed Storage.	366
Solution to a Linear System in a Cholesky-Factored Symmetric Positive Definite Matrix in Packed Storage	369
Reciprocal Condition Number of a UDU- or LDL-Factored Symmetric Positive Definite Tridiagonal Matrix	372

Refined Solution to a Linear System in a UDU- or LDL-Factored Symmetric Positive Definite Tridiagonal Matrix	375
Solution to a Linear System in a Symmetric Positive Definite Tridiagonal Matrix (Simple Driver).	381
Solution to a Linear System in a Symmetric Positive Definite Tridiagonal Matrix (Expert Driver)	384
UDU or LDL Factorization of a Symmetric Positive Definite Tridiagonal Matrix.	389
Solution to a Linear System in a UDU- or LDL-Factored Symmetric Positive Definite Tridiagonal Matrix	392
Eigenvalues and Eigenvectors of a Symmetric Matrix in Banded Storage (Simple Driver).	396
Eigenvalues and Eigenvectors of a Symmetric Matrix in Banded Storage (Expert Driver).	400
Reciprocal Condition Number of a UDU- or LDL-Factored Symmetric Matrix in Packed Storage	405
Eigenvalues and Eigenvectors of a Symmetric Matrix in Packed Storage (Simple Driver).	408
Eigenvalues and Eigenvectors of a Symmetric Matrix in Packed Storage (Expert Driver).	411
Generalized Eigenvalues and Eigenvectors for Symmetric Matrices in Packed Storage.	415
Refined Solution to a Linear System in a UDU- or LDL-Factored Symmetric Matrix in Packed Storage	419
Solution to a Linear System in a Symmetric Matrix in Packed Storage (Simple Driver).	425
Solution to a Linear System in a Symmetric Matrix in Packed Storage (Expert Driver).	428
UDU or LDL Factorization of a Symmetric Matrix in Packed Storage.	433

Inverse of a UDU- or LDL-Factored Symmetric Matrix in Packed Storage	436
Solution to a Linear System in a UDU- or LDL-Factored Symmetric Matrix in Packed Storage	439
Eigenvalues and Eigenvectors of a Symmetric Tridiagonal Matrix (Simple Driver)	442
Eigenvalues and Eigenvectors of a Symmetric Tridiagonal Matrix (Expert Driver)	445
Reciprocal Condition Number of a UDU- or LDL-Factored Symmetric Matrix	450
Eigenvalues and Eigenvectors of a Symmetric Matrix (Simple Driver)	453
Eigenvalues and Eigenvectors of a Symmetric Matrix (Expert Driver)	456
Generalized Eigenvalues and Eigenvectors for Symmetric Matrices	461
Refined Solution to a Linear System in a UDU- or LDL-Factored Symmetric Matrix	465
Solution to a Linear System in a Symmetric Matrix (Simple Driver)	471
Solution to a Linear System in a Symmetric Matrix (Expert Driver)	474
UDU or LDL Factorization of a Symmetric Matrix	479
Inverse of a UDU- or LDL-Factored Symmetric Matrix	482
Solution to a Linear System in a UDU- or LDL-Factored Symmetric Matrix	485
Reciprocal Condition Number of a Triangular Matrix in Banded Storage	488

Error Bounds and Estimates for the Solution to a Linear System in a Triangular Matrix in Banded Storage	492
Solution to a Linear System in a Triangular Matrix in Banded Storage.....	497
Reciprocal Condition Number of a Triangular Matrix in Packed Storage.....	501
Error Bounds and Estimates for the Solution of a Linear System in a Triangular Matrix in Packed Storage.....	504
Inverse of a Triangular Matrix in Packed Storage	509
Solution to a Linear System in a Triangular Matrix in Packed Storage.....	512
Reciprocal Condition Number of a Triangular Matrix	515
Error Bounds and Estimates for the Solution of a Linear System in a Triangular Matrix.....	518
Inverse of a Triangular Matrix	523
Solution to a Linear System in a Triangular Matrix.....	526
2. LINPACK Subroutines.....	531
Naming Conventions	532
Argument Conventions.....	533
Error Handling.....	533
Extensions to LINPACK	534
Cholesky Decomposition of a Symmetric Positive Definite Matrix 535	
Downdate an Augmented Cholesky Decomposition	538
Update an Augmented Cholesky Decomposition with Permutations	542
Update an Augmented Cholesky Decomposition	546

LU Factorization and Condition Number of a General Matrix in Banded Storage	549
Determinant of an LU-Factored General Matrix in Banded Storage	553
LU Factorization of a General Matrix in Banded Storage	556
Solution to a Linear System in an LU-Factored Matrix in Banded Storage	560
LU Factorization and Condition Number of a General Matrix	564
Determinant and Inverse of an LU-Factored General Matrix .	567
LU Factorization of a General Matrix	570
Solution to a Linear System in an LU-Factored General Matrix	572
Solution to a Linear System in a Tridiagonal Matrix	575
UDU Factorization and Condition Number of a Hermitian Matrix	578
Determinant, Inertia, and Inverse of a UDU-Factored Hermitian Matrix	581
UDU Factorization of a Hermitian Matrix	584
Solution to a Linear System in a UDU-Factored Hermitian Matrix	587
UDU Factorization and Condition Number of a Hermitian Matrix in Packed Storage	590
Determinant, Inertia, and Inverse of a UDU-Factored Hermitian Matrix in Packed Storage	593
UDU Factorization of a Hermitian Matrix in Packed Storage .	596
Solution to a Linear System in a UDU-Factored Hermitian Matrix in Packed Storage	598
Cholesky Factorization and Condition Number of a Symmetric Positive Definite Matrix in Banded Storage	601

Determinant of a Cholesky-Factored Symmetric Positive Definite Matrix in Banded Storage	604
Cholesky Factorization of a Symmetric Positive Definite Matrix in Banded Storage	607
Solution to a Linear System in a Cholesky-Factored Symmetric Positive Definite Matrix in Banded Storage	610
Cholesky Factorization and Condition Number of a Symmetric Positive Definite Matrix	613
Determinant and Inverse of a Cholesky-Factored Symmetric Positive Definite Matrix	616
Cholesky Factorization of a Symmetric Positive Definite Matrix	619
Solution to a Linear System in a Cholesky-Factored Symmetric Positive Definite Matrix	622
Cholesky Factorization and Condition Number of a Symmetric Positive Definite Matrix in Packed Storage	625
Determinant and Inverse of a Cholesky-Factored Symmetric Positive Definite Matrix in Packed Storage	628
Cholesky Factorization of a Symmetric Positive Definite Matrix in Packed Storage	631
Solution to a Linear System in a Cholesky-Factored Symmetric Positive Definite Matrix in Packed Storage	634
Solution to a Linear System in a Symmetric Positive Definite Tridiagonal Matrix	637
QR Factorization of a General Matrix	639
Solution to a Linear System in a QR-Factored General Matrix	642
UDU Factorization and Condition Number of a Symmetric Matrix	646
Determinant, Inertia, and Inverse of a UDU-Factored Symmetric Matrix	649

UDU Factorization of a Symmetric Matrix	652
Solution to a Linear System in a UDU-Factored Symmetric Matrix 655	
UDU Factorization and Condition Number of a Symmetric Matrix in Packed Storage	658
Determinant, Inertia, and Inverse of a UDU-Factored Symmetric Matrix in Packed Storage	661
UDU Factorization of a Symmetric Matrix in Packed Storage .	664
Solution to a Linear System in a UDU-Factored Symmetric Matrix in Packed Storage	667
Singular Value Decomposition of a General Matrix	669
Condition Number of a Triangular Matrix	672
Determinant and Inverse of a Triangular Matrix	675
Solution to a Linear System in a Triangular Matrix	678
3. Basic Linear Algebra Subprograms, Level 1 (BLAS1)	681
Conventions	682
Extensions to the Basic Linear Algebra Subprograms	683
Sum of the Absolute Values of a Vector	684
Product of a Scalar and Vector Plus a Vector	687
Copy a Vector	690
Dot Product (Inner Product)	693
Givens Plane Rotation	697
Applied Given's Plane Rotation	700
Modified Given's Plane Rotation	703
Applied Modified Given's Transformation	706
Index of Element with Maximum Absolute Value	709

Euclidean Norm of a Vector	712
Scale a Vector	714
Swap Two Vectors	716
4. Basic Linear Algebra Subprograms, Level 2 (BLAS2).....	719
Naming Conventions	719
Argument Conventions.....	721
Error Handling.....	721
Extensions to the Basic Linear Algebra Subprograms.....	722
Product of a Matrix in Banded Storage and a Vector.....	723
Product of a General Matrix and a Vector.....	727
Rank-1 Update to a General Matrix.....	730
Product of a Hermitian Matrix in Banded Storage and a Vector	734
Product of a Hermitian Matrix and a Vector	738
Rank-1 Update to a Hermitian Matrix.....	742
Rank-2 Update to a Hermitian Matrix.....	745
Product of a Hermitian Matrix in Packed Storage and a Vector	748
Rank-1 Update to a Hermitian Matrix in Packed Storage.....	751
Rank-2 Update to a Hermitian Matrix in Packed Storage.....	754
Product of a Symmetric Matrix in Banded Storage and a Vector	757
Product of a Symmetric Matrix in Packed Storage and a Vector	761
Rank-1 Update to a Real Symmetric Matrix in Packed Storage	764
Rank-2 Update to a Real Symmetric Matrix in Packed Storage	767
Product of a Symmetric Matrix and a Vector	770
Rank-1 Update to a Real Symmetric Matrix.....	773

Rank-2 Update to a Real Symmetric Matrix	776
Product of a Triangular Matrix in Banded Storage and a Vector	779
Solution to a Triangular System in Banded Storage of Linear Equations	783
Product of a Triangular Matrix in Packed Storage and a Vector	787
Solution to a Triangular System of Linear Equations in Packed Storage.	791
Product of a Triangular Matrix and a Vector	795
Solution to a Triangular System of Linear Equations	798
5. Basic Linear Algebra Subprograms, Level 3 (BLAS3).	801
Naming Conventions	802
Argument Conventions.	803
Error Handling.	803
Extensions to the Basic Linear Algebra Subprograms.	804
Product of Two General Matrices.	805
Product of a Hermitian Matrix and a General Matrix.	809
Rank-k Update of a Hermitian Matrix	812
Rank-2k Update of a Hermitian Matrix	815
Product of a Symmetric Matrix and a General Matrix	819
Rank-k Update of a Symmetric Matrix	823
Rank-2k Update of a Symmetric Matrix	826
Product of a Triangular Matrix and a General Matrix.	830
Solution for a Triangular System of Equations.	834
6. Fast Fourier Transform Subroutines (FFTPACK and VFFTPACK)	839

Naming Conventions	840
Extensions to FFTPACK and VFFTPACK	841
Transpose Array	842
Cosine Quarter-Wave Synthesis	844
Cosine Quarter-Wave Transform	847
Initialize Cosine Quarter-Wave Transform and Synthesis	850
Cosine Even-Wave Transform	852
Initialize Cosine Even-Wave Transform	855
EZ Fourier Synthesis	857
EZ Fourier Transform	860
Initialize EZ Fourier Transform and Synthesis	863
Fourier Synthesis	865
Fourier Transform	868
Initialize Fourier Transform and Synthesis	871
Sine Quarter-Wave Synthesis	873
Sine Quarter-Wave Transform	876
Initialize Sine Quarter-Wave Transform and Synthesis	879
Sine Odd-Wave Transform	881
Initialize Sine Odd-Wave Transform	884
A. List of Subprograms by Name	887

Preface

Introduction to the Sun Performance Library

This is the reference manual for the Sun Performance Library, which is a library of subroutines and functions to perform useful operations in computational linear algebra. This introductory section provides background information as well as information about storage conventions that you will need to know in order to use the Sun Performance Library. This reference manual contains a chapter for each of the seven standard libraries (FFTPACK and VFFTPACK are combined in one chapter) implemented in the Sun Performance Library; each chapter describes the subprograms for that library. To help in locating subprogram descriptions, the table of contents lists all the subprograms descriptively by function and appendix A lists all the subprograms alphabetically by name.

Sun Performance Library

The Sun Performance Library is based on the standard libraries LAPACK, LINPACK, BLAS1, BLAS2, BLAS3, FFPACK, and VFFPACK with extensions to support a C/C++ language interface. The Sun Performance Library versions of these subprograms perform the same operation and the FORTRAN-callable subroutines have the same interface as the standard versions, but the Sun Performance Library versions are generally faster and sometimes more accurate. Sun Performance Library subprograms achieve their speed by taking

advantage of specific features of the hardware and software with which they are run. Some subprograms achieve additional speedup by using more than one CPU on MP systems.

Using the Sun Performance Library on Solaris 1

The Solaris 1 version was compiled with the Sun SC3.0.1 FORTRAN and SC3.0 CC compilers using `-cg89` and `-dalign` compilation options. When using the C/C++ interfaces with a C compiler, remember to compile so that float arguments that are passed by value are not promoted to double during the call. When the prototype for a C/C++ function shows an argument as `float x`, the function expects to receive a 32-bit floating point value. When using C++, prototypes must be prefaced with the `extern "C"` qualifier. Your program must be compiled and linked with `-dalign` and linked with `-lsunperf`.

Using the Sun Performance Library on Solaris 2

The Solaris 2 versions were compiled with the Sun SC4.2 FORTRAN and C compilers using `-dalign` and `-xarch` set to one of `[v7 | v8 | v8plusa]`. For each `-xarch` option used to compile the libraries, there is a library compiled with `-xparallel` and a library compiled without `-xparallel`. When linking your program, use `-dalign`, `-xlic_lib=sunperf`, and the same `-xarch` option that you used when compiling. If you cannot use `-dalign` in your program then you must supply a trap 6 handler as described in `/opt/SUNWspro/SC4.2/READMEs/performance_library`. Your system administrator may have moved this file from its default location. If you compile with a value of `-xarch` that is not one of `[v7 | v8 | v8plusa]` then the compiler driver will select the closest match.

Sun Performance Library is a licensed library and so it is linked into an application with the `-xlic_lib` switch rather than the `-l` switch that is used to link in other libraries. The `-xlic_lib` switch is equivalent to the `-l` switch except that `-xlic_lib` links with a licensed library and directs the compiler driver to include a licensing step to allow the executable file to run. The `-xlic_lib` and `-l` switches are otherwise identical including the fact that the

`-xlic_lib` switch must be specified on the command line after any of the files that use Sun Performance Library. An example of the use of `-xlic_lib` is shown below:

```
dakota% f77 -dalign my_file.f -xlic_lib=sunperf
```

There are two models of parallelism available, one that is optimized for a dedicated machine and another for a shared machine. The shared multiprocessor model of parallelism has the following features:

- The parallelism model assumes a machine shared among many tasks.
- Parallelization is implemented with threads library synchronization primitives.

The dedicated multiprocessor model of parallelism has the following features:

- The parallelism model assumes a machine dedicated to one task.
- Parallelization is implemented with spin locks.

On a dedicated system, the dedicated model can be somewhat faster than the shared model due to lower synchronization overhead. On a system running many different tasks, the shared model provides better cooperation in the use of available resources. If you use one of `[-xparallel | -xexplicitpar | -xautopar]` on the compile and link lines then you will get the dedicated MP model. If you use `-mt` on the link line without one of the compiler parallelization options then you will get the shared model. If you do not specify any of the compiler parallelization options or `-mt` on the link line then your code will not use multiple processors.

If you compile with one of the compiler parallelization options then use the same parallelization option on the linking command. If you would like to use multiple processors then add `-mt` to the link line and then specify the number of processors at run-time with the `PARALLEL` environment variable. For example, to use 24 processors, you could enter the commands shown below:

```
dakota% f77 -fast -mt my_app.f -xlic_lib=sunperf
dakota% setenv PARALLEL 24
dakota% ./a.out
```

If you use the shared model with one of the compiler parallelization options then the behavior is unpredictable.

Using the Sun Performance Library on Solaris 2 for x86

The Solaris 2 versions were compiled with the Sun SC4.2 FORTRAN and C compilers. Sun Performance Library is a licensed library and so it is linked into an application with the `-xlic_lib` switch rather than the `-l` switch that is used to link in other libraries. The `-xlic_lib` switch is equivalent to the `-l` switch except that `-xlic_lib` links with a licensed library and directs the compiler driver to include a licensing step to allow the executable file to run. The `-xlic_lib` and `-l` switches are otherwise identical including the fact that the `-xlic_lib` switch must be specified on the command line after any of the files that use Sun Performance Library. An example of the use of `-xlic_lib` is shown below:

```
dakota% f77 my_file.f -xlic_lib=sunperf
```

If you would like to use multiple processors then add `-mt` to the link line and then specify the number of processors at run-time with the `PARALLEL` environment variable. For example, to use 12 processors, you could enter the commands shown below:

```
dakota% f77 -fast -mt my_app.f -xlic_lib=sunperf
dakota% setenv PARALLEL 12
dakota% ./a.out
```

Using the Native C/C++ Interface to Sun Performance Library

The C/C++ interfaces to subprograms in LAPACK, BLAS, LINPACK, FFTPACK, and VFFTPACK in Sun Performance Library are completely different from the FORTRAN interface to the standard libraries in Netlib. In addition, the C/C++ interfaces to LAPACK in Sun Performance Library differ from interfaces to the Netlib CLAPACK. There are three major differences between the C/C++ interface to Sun Performance Library and the FORTRAN interfaces to the standard libraries in Netlib. First, scalar arguments that are used only as inputs are passed by value in Sun Performance Library. Complex and double complex arguments are not considered scalars because they are not

implemented as a scalar type by C. Second, arguments relating to workspace are not used in Sun Performance Library. Third, array indices are based at zero in conformance with C conventions rather than being based at one to conform to FORTRAN conventions. As an example of these changes, compare the standard LAPACK FORTRAN interface with the C/C++ interface in Sun Performance Library shown below with the DGBCON subprogram.

```
CALL DGBCON (NORM, N, NSUB, NSUPER, DA, LDA, IPIVOT, DANORM,  
            DRCOND, DWORK, IWORK2, INFO)  
  
void dgbcon (char norm, long int n, long int, long int nsuper,  
            double *da, long int lda, long int *ipivot, double  
            danorm, double *drcond, long int *info)
```

Note that the names of the arguments are the same and that arguments with the same name have the same base type. Scalar arguments that are used only as input values, such as NORM and N, are passed by value in the C/C++ version. Arrays and scalars that will be used to return values are passed by reference.

Compatibility

The Fortran functions and subroutines are used by calling them from within a program, usually, but not necessarily, a FORTRAN 77 or Fortran 90 program. For instance, the calling program can be C or C++. However, the calling program must use the FORTRAN 77 calling sequence.

- Do not prototype the subroutines with Fortran 90's INTERFACE statement. The use of INTERFACE implies that the subroutines will use the Fortran 90 calling sequence, and this is not the case. Call the subroutines in the same way that you would call any FORTRAN 77 subroutine.
- Arrays are stored columnwise.
- All arguments are passed by reference.
- The number of arguments to a routine is fixed.
- Types of arguments must match even after C or C++ does type conversion. For example, care must be exercised when passing a single precision real value because a C or C++ compiler may automatically promote the argument to double precision.

-
- Indices are base at one in keeping with standard Fortran practice.

The C interfaces are used by calling them from within a program, usually, but not necessarily, a C or C++ program. For instance, the calling program can be a Pascal or Ada program. However, the calling sequence must follow these rules:

- Arrays are stored columnwise. This differs from standard C and C++ conventions.
- Indices are based at zero in keeping with standard C and C++ practice. For example, the FORTRAN interface to IDAMAX, which C programs access as "idamax_", would return a 1 to indicate the first element in a vector. The C interface to idamax, which C programs access as "idamax", would return a 0 to indicate the first element of a vector. This convention is observed in function return values, permutation vectors, and anywhere else that vector or array indices are used.

Special Storage Schemes

The Sun Performance Library processes matrices that are in one of four forms: general, triangular, symmetric (or Hermitian), or tridiagonal. The general form is the most common and most operations performed by the Sun Performance Library can be done on general arrays. In many cases, there also are subprograms that will work with the other forms of the arrays. For example, DGEMM will form the product of two general matrices and DTRMM will form the product of a triangular and a general matrix.

Most of these matrices can be stored in ways that save both storage space and computation time. The two types of storage that Sun Performance Library uses are banded storage and packed storage. Some of the subprograms that work with arrays stored normally have corresponding subprograms that take advantage of these special storage forms. For example, DGBMV will form the product of a general matrix in banded storage and a vector, and DTPMV will form the product of a triangular matrix in packed storage and a vector.

The storage forms and the four types of matrices are discussed and shown below.

Banded Storage

A banded matrix is stored so the j th column of the matrix corresponds to the j th column of the FORTRAN array. See the discussion of array forms for examples.

The following code is an example of taking a banded general matrix in a general array and copying it into banded storage mode.

```
C      Copy the matrix A from the array AG to the array AB.  The
C      matrix is stored in general storage mode in AG and it will
C      be stored in banded storage mode in AB.  The code to copy
C      from general to banded storage mode is taken from the
C      comment block in the original DGBFA by Cleve Moler.
C
      NSUB = 1
      NSUPER = 2
      NDIAG = NSUB + 1 + NSUPER
      DO 10, ICOL = 1, N
          I1 = MAX0 (1, ICOL - NSUPER)
          I2 = MIN0 (N, ICOL + NSUB)
          DO 10, IROW = I1, I2
              IROWB = IROW - ICOL + NDIAG
              AB(IROWB,ICOL) = AG(IROW,ICOL)
      10  CONTINUE
      20  CONTINUE
```

Note that this method of storing banded matrices is compatible with the storage method used by BLAS, LINPACK, and LAPACK, but is inconsistent with the method used by EISPACK.

Packed Storage

A packed vector is an alternate representation for triangular, symmetric, or Hermitian matrices. An array is packed into a vector by storing sequentially column by column into the vector. Space for the diagonal elements is always reserved even if the values of the diagonal elements are known, for example in a unit diagonal matrix. See the discussion of array forms for examples.

An upper triangular matrix or a symmetric (or Hermitian) matrix whose upper triangle is stored in general storage in the array A can be transferred to packed storage in the array AP as shown below. This code comes from the comment block of the LAPACK subroutine DTPTRI:

```
      JC = 1
      DO 20, J = 1, N
        DO 10, I = 1, J
          AP(JC+I-1) = A(I,J)
10      CONTINUE
        JC = JC + J
20 CONTINUE
```

Similarly, a lower triangular matrix or a symmetric (or Hermitian) matrix whose lower triangle is stored in general storage in the array A can be transferred to packed storage in the array AP as shown below:

```
      JC = 1
      DO 20, J = 1, N
        DO 10, I = J, N
          AP(JC+I-1) = A(I,J)
10      CONTINUE
        JC = JC + N - J + 1
20 CONTINUE
```

General Matrices

A general matrix is stored so that there is a one-to-one correspondence between the elements of the matrix and the elements of the array. Element A_{ij} of a matrix A is stored in element A(I,J) of the corresponding array A. The general form is the most common form. A general matrix, because it is dense, has no special storage scheme. In a general banded matrix, however, the diagonal of the matrix is stored in the row below the upper diagonals. For example, the banded general matrix below can be represented with banded storage as shown below. Elements shown with the symbol X are never accessed by subprograms that process banded arrays.

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 \\ a_{21} & a_{22} & a_{23} & a_{24} & 0 \\ 0 & a_{32} & a_{33} & a_{34} & a_{35} \\ 0 & 0 & a_{43} & a_{44} & a_{45} \\ 0 & 0 & 0 & a_{54} & a_{55} \end{pmatrix}$$

General banded matrix

$$\begin{bmatrix} \times & \times & a_{13} & a_{24} & a_{35} \\ \times & a_{12} & a_{23} & a_{34} & a_{45} \\ a_{11} & a_{22} & a_{33} & a_{44} & a_{55} \\ a_{21} & a_{32} & a_{43} & a_{54} & \times \end{bmatrix}$$

General banded array in banded storage

Triangular Matrices

A triangular matrix is stored so that there is a one-to-one correspondence between the non-zero elements of the matrix and the elements of the array, but the elements of the array corresponding to the zero elements of the matrix are never accessed by subprograms that process triangular arrays.

A triangular matrix can be stored using packed storage:

$$\begin{pmatrix} a_{11} & 0 & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

Triangular matrix

$$\begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \\ a_{22} \\ a_{32} \\ a_{33} \end{bmatrix}$$

Triangular array in packed storage

A triangular banded matrix can be stored using banded storage as shown below. Elements shown with the symbol $\times$ are never accessed by subprograms that process banded arrays.

$$\begin{pmatrix} a_{11} & 0 & 0 \\ a_{21} & a_{22} & 0 \\ 0 & a_{32} & a_{33} \end{pmatrix}$$

Triangular banded matrix

$$\begin{bmatrix} a_{11} & a_{22} & a_{33} \\ a_{21} & a_{32} & \times \end{bmatrix}$$

Triangular banded array in banded storage

Symmetric (or Hermitian) Matrices

A symmetric (or Hermitian) matrix is similar to a triangular matrix in that the data in either the upper or lower triangle corresponds to the elements of the array. The contents of the other elements in the array are assumed and those array elements are never accessed by subprograms that process symmetric or Hermitian arrays. The imaginary elements on the main diagonal of a Hermitian matrix are assumed to be zero on entry to a subroutine.

A symmetric (or Hermitian) matrix can be stored using packed storage:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \qquad \begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \\ a_{22} \\ a_{32} \\ a_{33} \end{bmatrix}$$

Symmetric (or Hermitian) matrix Symmetric (or Hermitian) array in packed storage

A symmetric (or Hermitian) banded matrix can be stored using banded storage as shown below. Elements shown with the symbol **X** are never accessed by subprograms that process banded arrays.

$$\begin{pmatrix} a_{11} & a_{12} & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ 0 & a_{32} & a_{33} & a_{34} \\ 0 & 0 & a_{43} & a_{44} \end{pmatrix} \qquad \begin{bmatrix} \mathbf{X} & a_{12} & a_{23} & a_{34} \\ a_{11} & a_{22} & a_{33} & a_{44} \\ a_{21} & a_{32} & a_{43} & \mathbf{X} \end{bmatrix}$$

Symmetric (or Hermitian) banded matrix Symmetric (or Hermitian) banded array
in banded storage

Tridiagonal Matrices

A tridiagonal matrix has elements only on the main diagonal, the first superdiagonal, and the first subdiagonal. It is stored using three 1-dimensional arrays:

$$\begin{pmatrix} a_{11} & a_{12} & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ 0 & a_{32} & a_{33} & a_{34} \\ 0 & 0 & a_{43} & a_{44} \end{pmatrix}$$

Tridiagonal matrix

$$\begin{bmatrix} a_{21} \\ a_{32} \\ a_{43} \end{bmatrix} \quad \begin{bmatrix} a_{11} \\ a_{22} \\ a_{33} \\ a_{44} \end{bmatrix} \quad \begin{bmatrix} a_{12} \\ a_{23} \\ a_{34} \end{bmatrix}$$

Tridiagonal array in tridiagonal storage

This chapter describes the subroutines in LAPACK as they are implemented in the Sun Performance Library. LAPACK is a library of subroutines to perform matrix factorizations, solve systems of linear equations, solve least squares problems, compute eigenvalues and eigenvectors, and other common linear algebra computations. LAPACK is a superset of LINPACK and subroutines in LAPACK are generally faster and more accurate than their LINPACK counterparts, so all new development should be done with LAPACK. The version of LAPACK released with the Sun Performance Library and documented in this manual is version 1.1 released in March 1993. For more information about LAPACK, refer to the information in this chapter or see *LAPACK User's Guide* written by Anderson *et al* and available from the Society for Industrial and Applied Mathematics (SIAM) in Philadelphia. The serious numerical analyst will want to get a copy of the *LAPACK User's Guide* in addition to this reference manual because it contains a wealth of information about error bounds, algorithmic details, and the numerical stability of the algorithms used in LAPACK.

The first three sections of this chapter describe:

- conventions followed in naming arguments and subroutines and conventions used for passing arguments into and out of subroutines
- error handling done by the subroutines
- extensions to the standard version of LAPACK

The remaining sections of this chapter describe the operation of each of the major LAPACK subroutines as described below. For each subroutine, the interface is shown, the operation of the subroutine is described, and a short FORTRAN 77 program is given with sample output.

LAPACK contains three types of subroutines: driver, computational, and auxiliary. The driver subroutines are further subdivided into simple and expert drivers. For each task performed by a driver, there is both a simple and expert driver. The simple drivers perform a complete task such as solving a linear system. The expert drivers usually perform the same task as the simple driver plus related tasks such as computing error bounds or condition numbers. The drivers do their work by calling a series of computational routines, each of which does some part of the larger task. For example, a driver to solve a linear system calls a computational subroutine to factor the coefficient matrix and then calls another computational subroutine to solve the factored system. Auxiliary subroutines tend to perform very low level tasks such as copying a matrix or rearranging columns of a matrix.

This manual documents all of the driver subroutines, and all of the computational subroutines related to solving a linear system, but none of the auxiliary subroutines. Most users will want to use the driver subroutines most of the time. In those cases in which it is necessary or helpful to exercise more direct control over a computation, users may instead call one or more computational routines. It is atypical for a user to make a direct call to an auxiliary subroutine, but it is allowed.

Naming Conventions

The types of all arguments follow the notation shown below:

```
IMPLICIT COMPLEX (C)
IMPLICIT DOUBLE PRECISION (D)
IMPLICIT INTEGER (I-N)
IMPLICIT DOUBLE COMPLEX (Z)
CHARACTER BALANC, COMPZ, DIAG, EQUED, FACT, HOWMNY, JOB,
$          JOBSV, JOBU, JOBVL, JOBVR, JOBVT, NORM, SENSE,
$          SORTEV, TRANSA, UPLO
LOGICAL BWORK3, SELECT
```

The first letter of a subroutine name identifies the type of the result. The second and third letters identify the form of the output matrix and its mode of storage in an array. The legal forms are shown below:

BD	bidagonal
GB	general in banded storage
GE	general
GG	general matrix, generalized problem
GT	general tridiagonal
HB	Hermitian in banded storage
HE	Hermitian
HG	upper Hessenberg matrix, generalized problem
HP	Hermitian in packed storage
HS	upper Hessenberg
OP	orthogonal matrix in packed storage
OR	orthogonal matrix
PB	symmetric (or Hermitian) positive definite in banded storage
PO	symmetric (or Hermitian) positive definite
PP	symmetric (or Hermitian) positive definite in packed storage
PT	symmetric (or Hermitian) tridiagonal positive definite
SB	symmetric in banded storage
SP	symmetric in packed storage
ST	symmetric tridiagonal
SY	symmetric
TB	triangular in banded storage
TG	triangular matrix, generalized problem
TP	triangular in packed storage
TR	triangular
TZ	upper trapezoidal

UN unitary
UP unitary in packed storage

The remaining letters identify the operation that the subroutine performs. The legal operations are shown below. Not all operations shown below can be applied to all types of matrices shown above.

BAK back transform eigenvectors of a balanced matrix
BAL balance a matrix to attempt to improve subsequence eigenvalue / eigenvector computation
BRD reduce a matrix to bidiagonal form
CON estimate the condition number
EBZ compute eigenvalues via bisection
EIN compute eigenvectors via inverse iteration
EQR compute eigenvalues and eigenvectors
EQU compute equilibration scale factors
ERF compute eigenvalues via QR
ES compute eigenvalues and Schur factorization
ESX compute eigenvalues, Schur factorization, and various condition numbers
EV compute eigenvalues and eigenvectors
EVC compute eigenvectors of a quasi-triangular matrix
EVX compute eigenvalues, eigenvectors, condition numbers with high accuracy
EXC move a 1×1 or 2×2 block on the diagonal of a matrix in Schur canonical form to another place

GBR generate U and V after reduction of A to bidiagonal form
GHR generate a matrix after reduction to upper Hessenberg form
GLQ generate Q from elementary reflectors computed during an LQ factorization
GQL generate Q from elementary reflectors computed during a QL factorization
GQR generate Q from elementary reflectors computed during a QR factorization
GRQ generate Q from elementary reflectors computed during an RQ factorization
GST reduce a matrix for a generalized eigenproblem to a matrix for a standard eigenproblem
GTR generate Q from elementary reflectors computed during a reduction to tridiagonal form
GV solve generalized eigenproblem
HRD reduce to upper Hessenberg form
LQF LQ factorization
LS solve a linear least squares problem using QR or LQ factorization
LSS solve a linear least squares problem using the singular value decomposition
LSX solve a linear least squares problem using complete orthogonal factorization
MBR multiply a matrix by U and V after reduction of A to bidiagonal form
MHR multiply a matrix by Q using elementary reflectors computed during a Hessenberg reduction

MLQ	multiply a matrix by Q using elementary reflectors computed during an LQ factorization
SQL	multiply a matrix by Q using elementary reflectors computed during a QL factorization
QQR	multiply a matrix by Q using elementary reflectors computed during a QR factorization
MRQ	multiply a matrix by Q using elementary reflectors computed during an RQ factorization
MTR	multiply a matrix by Q using elementary reflectors computed during a tridiagonal reduction
QLF	QL factorization
QPF	QR factorization with column pivoting
QRF	QR factorization
RFS	refine the solution to a linear system
RQF	RQ factorization
SEN	move a set of eigenvalues of a matrix in Schur form and compute various condition numbers
SNA	compute condition numbers for a matrix in Schur canonical form
SQR	compute singular values of a bidiagonal matrix
SV	solve a linear system, non-expert driver
SV	solve a linear system, expert driver
SVD	singular value decomposition
SYL	solve the Sylvester equation $AX \pm XB = C$
TRD	reduce a matrix to real symmetric tridiagonal form
TRF	factor a matrix
TRI	compute inverse of a matrix
TRS	solve a triangular or LU-factored system

To specify the group of subroutines that perform the same operation but are of different types, this manual uses a lowercase x. For example, xGESV refers to the subroutines DGESV, SGESV, ZGESV, and CGESV. This same convention is used for arguments: xA refers to the arguments DA, SA, ZA, and CA.

Error Handling

If the programmer calls a LAPACK subroutine with an illegal argument value then the subroutine will call the error handling subroutine XERBLA. XERBLA will print the name of the subroutine in which the error was detected and the number of the first argument found to be in error. As an example, consider the following call to DGBCON:

```
CALL DGBCON ('*ILLEGAL*', -526, SUB, SUPER, A, LDA, IPIV, ANORM, RC, WRK, IWRK, INFO)
```

In the call above, the values of NORM and N (the first two arguments) are illegal. DGBCON will call XERBLA, which will print the report shown below and then terminate with a STOP statement.

```
** On entry to DGBCON parameter number 1 had an illegal value
STOP
```

The Sun Performance Library will detect all instances in which the calling subprogram specifies one or more illegal parameter values, but there are many types of interface errors that the Sun Performance Library will not detect. Included in the list of interface errors that the Sun Performance Library will not detect is specifying too many or too few arguments, specifying an argument of the wrong type, and aliasing an argument with the result argument.

Extensions to LAPACK

There are two extensions to the standard described in the *LAPACK User's Guide*. First, all CHARACTER arguments are documented as being legal in upper-, lower-, or mixed case. Standard FORTRAN 77 does not allow lowercase letters, so the LAPACK User's Guide always documents arguments as being uppercase. The standard LAPACK source code, however, does process lowercase letters, so using lowercase letters will be portable to other hardware and software environments. CHARACTER arguments that are returned from LAPACK will always be returned in uppercase. Also, this version of Sun Performance Library contains a native C/C++ interface to LAPACK. Conventions for this interface are described in detail in the preface to this manual.

Reciprocal Condition Number of an LU-Factored General Matrix in Banded Storage

The subroutines described in this section estimate the reciprocal condition number, in either the 1-norm or the ∞ -norm, of a general matrix A in banded storage, which has been LU-factored by xGBTRF.

Calling Sequence

```
CALL DGBCON (NORM, N, NSUB, NSUPER, DA, LDA, IPIVOT, DANORM,
             DRCOND, DWORK, IWORK2, INFO)
CALL SGBCON (NORM, N, NSUB, NSUPER, SA, LDA, IPIVOT, SANORM,
             SRCOND, SWORK, IWORK2, INFO)
CALL ZGBCON (NORM, N, NSUB, NSUPER, ZA, LDA, IPIVOT, DANORM,
             DRCOND, ZWORK, DWORK2, INFO)
CALL CGBCON (NORM, N, NSUB, NSUPER, CA, LDA, IPIVOT, SANORM,
             SRCOND, CWORK, SWORK2, INFO)

void dgbcon (char norm, int n, int nsub, int nsuper, double *da,
             int lda, int *ipivot, double danorm, double *drcond,
             int *info)
void sgbcon (char norm, int n, int nsub, int nsuper, float *sa, int
             lda, int *ipivot, float sanorm, float *srcond, int
             *info)
void zgbcon (char norm, int n, int nsub, int nsuper, doublecomplex
             *za, int lda, int *ipivot, double danorm, double
             *drcond, int *info)
void cgbcon (char norm, int n, int nsub, int nsuper, complex *ca,
             int lda, int *ipivot, float sanorm, float *srcond, int
             *info)
```

Arguments

NORM	Indicates whether the 1-norm condition number or the ∞ -norm condition number is required. The legal values for NORM are listed below. Any values not listed below are illegal. 'O', 'o', or '1' 1-norm condition number is required. 'I' or 'i' ∞ -norm condition number is required.
N	Order of the matrix A. $N \geq 0$.
NSUB	Number of subdiagonals of A. $N-1 \geq NSUB \geq 0$ but if $N = 0$ then $NSUB = 0$.
NSUPER	Number of superdiagonals of A. $N-1 \geq NSUPER \geq 0$ but if $N = 0$ then $NSUPER = 0$.

xA	LU factorization of the matrix A, as computed by xGBTRF.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq 2 \times NSUB + NSUPER + 1$.
IPIVOT	Pivot indices as computed by xGBTRF.
xANORM	On entry, if NORM = 'O', 'o', or '1', then ANORM contains the 1-norm of the original matrix A; if NORM = 'I' or 'i', then ANORM contains the ∞ -norm of the original matrix A.
xRCOND	On exit, the estimated reciprocal of the condition number of the matrix A where the reciprocal condition number of A is defined to be $1 / (ANORM \times \ A^{-1}\)$. The reciprocal of the condition number is estimated instead of the condition number itself to avoid overflow or division by zero. If RCOND is less than machine precision (in particular, if RCOND = 0) then A is singular to working precision.
xWORK	Scratch array with a dimension of $3 \times N$ for real subroutines or $2 \times N$ for complex subroutines.
xWORK2	Scratch array with a dimension of N.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The <i>i</i> th argument, where $i = INFO $, had an illegal value

Sample Program

```
PROGRAM TEST
IMPLICIT NONE

C
C      INTEGER          LDA, LDWORK, N, NSUB, NSUPER
C      PARAMETER        (N = 5)
C      PARAMETER        (NSUB = 4)
C      PARAMETER        (NSUPER = 0)
C      PARAMETER        (LDA = 2*NSUB + 1 + NSUPER)
C      PARAMETER        (LDWORK = 3 * N)
C
C      DOUBLE PRECISION A(LDA,N), ANORM, RCOND, WORK(LDWORK)
C      INTEGER          ICOL, INFO, IPIVOT(N), IROW, IWORK(N)
C
C      EXTERNAL          DGBCON, DGBTRF
C      INTRINSIC         ABS
C
C      Initialize the array A to store in banded form the 5x5
C      matrix A with four subdiagonals and no superdiagonals
C      shown below.
C
C      1
C      1 -1
C      A = 1 -2  1
C      1 -3  3 -1
C      1 -4  6 -4  1
```

```

C
DATA A / NSUB*8D8, 5*1.0D0,
$      NSUB*8D8, -1.0D0, -2.0D0, -3.0D0, -4.0D0, 8D8,
$      NSUB*8D8, 1.0D0, 3.0D0, 6.0D0, 2*8D8,
$      NSUB*8D8, -1.0D0, 4.0D0, 3*8D8,
$      NSUB*8D8, 1.0D0, 4*8D8 /
C
C   Print the initial values of the arrays.
C
PRINT 1000
DO 100, IROW = NSUB + 1, LDA
  PRINT 1010,
  $      (A(IROW - ICOL + 1, ICOL), ICOL = 1, IROW - NSUB),
  $      (0.0D0, ICOL = 1, N - IROW)
100 CONTINUE
PRINT 1020
PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, LDA)
C
C   LU factor A.
C
CALL DGBTRF (N, N, NSUB, NSUPER, A, LDA, IPIVOT, INFO)
IF (INFO .NE. 0) THEN
  PRINT 1030, INFO
  STOP 1
END IF
C
C   Compute and print the condition number of A.
C
ANORM = 1.0D1
CALL DGBCON ('ONE NORM', N, NSUB, NSUPER, A, LDA, IPIVOT,
$      ANORM, RCOND, WORK, IWORK, INFO)
IF (INFO .NE. 0) THEN
  PRINT 1040, ABS(INFO)
  STOP 2
END IF
PRINT 1050, 1.0D0 / RCOND
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (5(3X, F4.1))
1020 FORMAT (/1X, 'A in banded form: (* in unused elements)')
1030 FORMAT (1X, 'Error factoring A, INFO = ', I5)
1040 FORMAT (1X, 'Illegal argument to DGBCON, argument #', I2)
1050 FORMAT (/1X, 'Estimated condition number of A = ', F8.4)
C
END

```


Sample Output

```
A in full form:
  1.0
  1.0  -1.0
  1.0  -2.0   1.0
  1.0  -3.0   3.0  -1.0
  1.0  -4.0   6.0   4.0   1.0

A in banded form: (* in unused elements)
****      ****      ****      ****      ****
****      ****      ****      ****      ****
****      ****      ****      ****      ****
****      ****      ****      ****      ****
  1.0  -1.0   1.0  -1.0   1.0
  1.0  -2.0   3.0   4.0  ****
  1.0  -3.0   6.0  ****  ****
  1.0  -4.0  ****  ****  ****
  1.0  ****  ****  ****  ****

Estimated condition number of A = 260.0000
```

Equilibration Scale Factors for a General Matrix in Banded Storage

The subroutines described in this section compute row and column scalings intended to equilibrate a general matrix A in banded storage and reduce its condition number. The scale factors are chosen to try to make largest element of the equilibrated matrix have magnitude equal to 1. Note that these subroutines only compute the scale factors for A. Additional code must be written to actually equilibrate A. Each element of the equilibrated matrix B has the value $B(i,j) = \text{ROWSC}(i) \times B(i,j) \times \text{COLSC}(j)$.

Calling Sequence

```
CALL DGBEQU (M, N, NSUB, NSUPER, DA, LDA, DROWSC, DCOLSC, DROWCN,
             DCOLCN, DAMAX, INFO)
CALL SGBEQU (M, N, NSUB, NSUPER, SA, LDA, SROWSC, SCOLSC, SROWCN,
             SCOLCN, SAMAX, INFO)
CALL ZGBEQU (M, N, NSUB, NSUPER, ZA, LDA, DROWSC, DCOLSC, DROWCN,
             DCOLCN, DAMAX, INFO)
CALL CGBEQU (M, N, NSUB, NSUPER, CA, LDA, SROWSC, SCOLSC, SROWCN,
             SCOLCN, SAMAX, INFO)

void dgbequ (int m, int n, int nsub, int nsuper, double *da, int
             lda, double *drowsc, double *dcolsc, double *drowcn,
             double *dcolcn, double *damax, int *info)
void sgbequ (int m, int n, int nsub, int nsuper, float *sa, int
             lda, float *srowsc, float *scolsc, float *srowcn, float
             *scolcn, float *samax, int *info)
void zgbequ (int m, int n, int nsub, int nsuper, doublecomplex *za,
             int lda, double *drowsc, double *dcolsc, double
             *drowcn, double *dcolcn, double *damax, int *info)
void cgbequ (int m, int n, int nsub, int nsuper, complex *ca, int
             lda, float *srowsc, float *scolsc, float *srowcn, float
             *scolcn, float *samax, int *info)
```

Arguments

M	Number of rows in the matrix A. $M \geq 0$.
N	Number of columns in the matrix A. $N \geq 0$.
NSUB	Number of subdiagonals of A. $N-1 \geq \text{NSUB} \geq 0$ but if $N = 0$ then $\text{NSUB} = 0$.
NSUPER	Number of superdiagonals of A. $N-1 \geq \text{NSUPER} \geq 0$ but if $N = 0$ then $\text{NSUPER} = 0$.

xA	Matrix A.								
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq NSUB + NSUPER + 1$.								
xROWSC	On exit, if INFO = 0 or INFO > M, ROWSC contains the row scale factors for the matrix A.								
xCOLSC	On exit, if INFO = 0, COLSC contains the column scale factors for the matrix A.								
xROWCN	On exit, if INFO = 0 or INFO > M, the ratio of the smallest ROWSC(i) to the largest ROWSC(i). If ROWCN ≥ 0.1 and AMAX is not close to overflow or underflow, then it is not worth scaling by ROWSC.								
xCOLCN	On exit, if INFO = 0, the ratio of the smallest COLSC(i) to the largest COLSC(i). If COLCN ≥ 0.1 and AMAX is not close to overflow or underflow, then it is not worth scaling by COLSC.								
xAMAX	On exit, the absolute value of the largest element of A. If AMAX is very close to overflow or underflow, A should be scaled regardless of the value of ROWCN or COLCN.								
INFO	On exit: <table> <tr> <td>INFO = 0</td><td>Subroutine completed normally.</td></tr> <tr> <td>INFO < 0</td><td>The i-th argument, where $i = INFO$, had an illegal value.</td></tr> <tr> <td>$1 \leq INFO \leq M$</td><td>Row INFO of the array A is exactly zero.</td></tr> <tr> <td>INFO > M</td><td>Column (INFO-M) of array A is exactly zero.</td></tr> </table>	INFO = 0	Subroutine completed normally.	INFO < 0	The i-th argument, where $i = INFO $, had an illegal value.	$1 \leq INFO \leq M$	Row INFO of the array A is exactly zero.	INFO > M	Column (INFO-M) of array A is exactly zero.
INFO = 0	Subroutine completed normally.								
INFO < 0	The i-th argument, where $i = INFO $, had an illegal value.								
$1 \leq INFO \leq M$	Row INFO of the array A is exactly zero.								
INFO > M	Column (INFO-M) of array A is exactly zero.								

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
C      INTEGER          LDA, N, NSUB, NSUPER
C      PARAMETER        (N = 4)
C      PARAMETER        (NSUB = 1)
C      PARAMETER        (NSUPER = 0)
C      PARAMETER        (LDA = NSUB + 1 + NSUPER)

C
C      DOUBLE PRECISION A(LDA,N), AMAX, COLSCA(N), COLCND
C      DOUBLE PRECISION ROWCND, ROWSCA(N)
C      INTEGER          ICOL, INFO, IROW

C
C      EXTERNAL          DGBEQU
C      INTRINSIC         ABS

C
C      Initialize the array A to store in banded storage mode the
C      4x4 matrix A shown below.
C
C      2  0  0  0
C      A = 4  4  0  0

```

```

C      0  4  4  0
C      0  0  2  2
C
C      DATA A / 2.0D0, 4.0D0, 4.0D0, 4.0D0, 4.0D0, 2.0D0, 2.0D0,
$          8D8 /
C
C      PRINT 1000
C      PRINT 1010, A(1,1), 0.0D0, 0.0D0, 0.0D0
C      PRINT 1010, A(2,1), A(1,2), 0.0D0, 0.0D0
C      PRINT 1010, 0.0D0, A(2,2), A(1,3), 0.0D0
C      PRINT 1010, 0.0D0, 0.0D0, A(2,3), A(1,4)
C      PRINT 1020
C      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, LDA)
C
C      CALL DGBEQU (N, N, NSUB, NSUPER, A, LDA, ROWSCA, COLSCA,
$          ROWCND, COLCND, AMAX, INFO)
C      IF (INFO .NE. 0) THEN
C          PRINT 1030, ABS(INFO)
C          STOP 1
C      END IF
C      PRINT 1040
C      PRINT 1010, ROWSCA
C      PRINT 1050
C      PRINT 1010, COLSCA
C
C      1000 FORMAT (1X, 'A in full format:')
C      1010 FORMAT (4(3X, F8.5))
C      1020 FORMAT (/1X, 'A in banded form: (* in unused elements)')
C      1030 FORMAT (1X, 'Illegal argument to DGBEQU, INFO = ', I4)
C      1040 FORMAT (/1X, 'Row scale factors:')
C      1050 FORMAT (/1X, 'Column scale factors:')
C
C      END

```

Sample Output

```

A in full format:
 2.00000  0.00000  0.00000  0.00000
 4.00000  4.00000  0.00000  0.00000
 0.00000  4.00000  4.00000  0.00000
 0.00000  0.00000  2.00000  2.00000

A in banded form: (* in unused elements)
 2.00000  4.00000  4.00000  2.00000
 4.00000  4.00000  2.00000  *****

Row scale factors:
 0.50000  0.25000  0.25000  0.50000

Column scale factors:
 1.00000  1.00000  1.00000  1.00000

```

Refined Solution to a Linear System in an LU-Factored General Matrix in Banded Storage

The subroutines described in this section refine the solution to a linear system $AX = B$, $A^T X = B$, or $A^H X = B$ for a general matrix A in banded storage, which has been LU-factored by xGBTRF, and general matrices B and X . These subroutines also compute forward error bounds and backward error estimates for the refined solution.

Calling Sequence

```
CALL DGBRFS (TRANSA, N, NSUB, NSUPER, NRHS, DA, LDA, DAF, LDAF,
             IPIVOT, DB, LDB, DX, LDX, DFERR, DBERR, DWORK, IWORK2,
             INFO)
CALL SGBRFS (TRANSA, N, NSUB, NSUPER, NRHS, SA, LDA, SAF, LDAF,
             IPIVOT, SB, LDB, SX, LDX, SFERR, SBERR, SWORK, IWORK2,
             INFO)
CALL ZGBRFS (TRANSA, N, NSUB, NSUPER, NRHS, ZA, LDA, ZAF, LDAF,
             IPIVOT, ZB, LDB, ZX, LDX, ZFERR, ZBERR, ZWORK, DWORK2,
             INFO)
CALL CGBRFS (TRANSA, N, NSUB, NSUPER, NRHS, CA, LDA, CAF, LDAF,
             IPIVOT, CB, LDB, CX, LDX, CFERR, CBERR, CWORK, SWORK2,
             INFO)

void dgbfrfs (char trans, int n, int nsub, int nsuper, int nrhs,
             double *da, int lda, double *daf, int ldaf, int
             *ipivot, double *db, int ldb, double *dx, int ldx,
             double *dferr, double *dberr, int *info)
void sgbfrfs (char trans, int n, int nsub, int nsuper, int nrhs,
             float *sa, int lda, float *saf, int ldaf, int *ipivot,
             float *sb, int ldb, float *sx, int ldx, float *sferr,
             float *sberr, int *info)
void zgbfrfs (char trans, int n, int nsub, int nsuper, int nrhs,
             doublecomplex *za, int lda, doublecomplex *zaf, int
             ldaf, int *ipivot, doublecomplex *zb, int ldb,
             doublecomplex *zx, int ldx, double *dferr, double
             *dberr, int *info)
void cgbfrfs (char trans, int n, int nsub, int nsuper, int nrhs,
             complex *ca, int lda, complex *caf, int ldaf, int
             *ipivot, complex *cb, int ldb, complex *cx, int ldx,
             float *sferr, float *sberr, int *info)
```

Arguments

TRANSA	Indicates the form of the linear system whose solution is to be refined. The legal values for TRANSA are listed below. Any values not listed below are illegal. 'N' or 'n' No transpose, refine $AX = B$. 'T' or 't' Transpose, refine $A^T X = B$. 'C' or 'c' Conjugate transpose, refine $A^H X = B$. Note that A^T and A^H are the same for real matrices.
N	Order of the matrix A. $N \geq 0$.
NSUB	Number of subdiagonals of A. $N-1 \geq NSUB \geq 0$ but if $N = 0$ then $NSUB = 0$.
NSUPER	Number of superdiagonals of A. $N-1 \geq NSUPER \geq 0$ but if $N = 0$ then $NSUPER = 0$.
NRHS	Number of right-hand sides, equal to the number of columns in the matrices B and X. $NRHS \geq 0$.
xA	Matrix A.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq NSUB + NSUPER + 1$.
xAF	LU factorization of the matrix A, as computed by xGBTRF.
LDAF	Leading dimension of the array AF as specified in a dimension or type statement. $LDAF \geq 2 \times NSUB + NSUPER + 1$.
IPIVOT	Pivot indices as computed by xGBTRF.
xB	The $N \times NRHS$ right-hand side matrix B.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
xx	On entry, the $N \times NRHS$ solution matrix X. On exit, the refined $N \times NRHS$ solution matrix.
LDX	Leading dimension of the array X as specified in a dimension or type statement. $LDX \geq \max(1, N)$.
xFERR	On exit, the estimated forward error bound for each solution vector $X(*, j)$ for $1 \leq j \leq NRHS$. If X' is the true solution corresponding to $X(*, j)$ then $FERR(j)$ is an upper bound on the magnitude of the largest element in $X(*, j) - X'$ divided by the magnitude of the largest element in $X(*, j)$.
xBERR	On exit, $BERR(j)$ is the smallest relative change in any element of A or $B(*, j)$ that makes $X(*, j)$ an exact solution to $AX(*, j) = B(*, j)$ for $1 \leq j \leq NRHS$.
xWORK	Scratch array with a dimension of $3 \times N$ for real subroutines or $2 \times N$ for complex subroutines.
xWORK2	Scratch array with a dimension of N.

INFO	On exit:	
	INFO = 0	Subroutine completed normally.
	INFO < 0	The ith argument, where i = INFO , had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C      INTEGER          LDA, LDAF, LDB, LDIWRK, LDWORK, LDX, N
C      INTEGER          NRHS, NSUB, NSUPER
C      PARAMETER        (N = 4)
C      PARAMETER        (NSUB = 1)
C      PARAMETER        (NSUPER = 0)
C      PARAMETER        (LDA = 2*NSUB + 1 + NSUPER)
C      PARAMETER        (LDAF = LDA)
C      PARAMETER        (LDB = N)
C      PARAMETER        (LDIWRK = N)
C      PARAMETER        (LDWORK = 3 * N)
C      PARAMETER        (LDX = N)
C      PARAMETER        (NRHS = 1)
C
C      DOUBLE PRECISION A(LDA,N), AF(LDAF,N), B(LDB,NRHS)
C      DOUBLE PRECISION B1(LDB,NRHS), BERR(NRHS), EPSLON
C      DOUBLE PRECISION FERR(NRHS), WORK(LDWORK), X(LDX,NRHS)
C      INTEGER          ICOL, INFO, IPIVOT(N), IROW, IWORK(LDIWRK)
C
C      EXTERNAL          DCOPY, DGBMV, DGBTRF, DGBTRS, DGBRFS
C      INTRINSIC         ABS, DBLE, MAX
C
C      Initialize the array A to store in banded form the matrix A
C      shown below. Initialize the array B to store the right hand
C      side vector b shown below.
C
C      C
C      C      0          4
C      C      A = 9      0          b = 4
C      C              0.25      5          4
C      C              0          70         4
C      C
C      DATA A / NSUB*8D8, 0.0D0, 9.0D0, NSUB*8D8, 0.0D0, 2.5D-1,
C      $        NSUB*8D8, 5.0D0, 0.0D0, NSUB*8D8, 7.0D1, 8D8 /
C      DATA B / N*4.0D0 /
C
C      Add a small value to the elements of A on the diagonal and
C      on the subdiagonal. After this loop, A will contain
C      something similar to:
C
C      C
C      C      e
C      C      A = 9+e      e
C      C              0.25+e      5+e
C

```

```

C                                0+e 70+e
C
      EPSLON = (((2.0D0 / 3.0D0) + 8.0D0) - 8.0D0) -
$      (2.0D0 / 3.0D0))
      DO 110, ICOL = 1, N
        DO 100, IROW = 1, NSUB + 1 + NSUPER
          A(NSUB+IROW,ICOL) = A(NSUB+IROW,ICOL) + EPSLON
100    CONTINUE
110  CONTINUE
      CALL DCPY (LDA*N, A, 1, AF, 1)
C
C    Print the A array.
C
      PRINT 1000
      PRINT 1010, A(2,1), 0.0D0, 0.0D0, 0.0D0
      PRINT 1010, A(3,1), A(2,2), 0.0D0, 0.0D0
      PRINT 1010, 0.0D0, A(3,2), A(2,3), 0.0D0
      PRINT 1010, 0.0D0, 0.0D0, A(3,3), A(2,4)
      PRINT 1020
      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, LDA)
C
C    Make a copy of B and then print B.
C
      CALL DCPY (N, B, 1, X, 1)
      PRINT 1030
      PRINT 1040, B
C
C    LU factor A.
C
      CALL DGBTRF (N, N, NSUB, NSUPER, AF, LDAF, IPIVOT, INFO)
      IF (INFO .NE. 0) THEN
        PRINT 1050, INFO
        STOP 1
      END IF
C
C    Solve Ax=b and print the solution.
C
      CALL DGBTRS ('NO TRANSPOSE A', N, NSUB, NSUPER, NRHS, AF,
$      LDAF, IPIVOT, X, LDX, INFO)
      IF (INFO .NE. 0) THEN
        PRINT 1060, INFO
        STOP 2
      END IF
      PRINT 1070
      PRINT 1080, X
      CALL DGBMV ('NO TRANSPOSE A', N, N, NSUB, NSUPER, 1.0D0,
$      A(NSUB+1,1), LDA, X, 1, 0.0D0, B1, 1)
      PRINT 1090
      PRINT 1040, B1
C
C    Refine the initial solution and print the new solution.
C
      CALL DGBRFS ('NO TRANSPOSE A', N, NSUB, NSUPER, NRHS,
$      A(NSUB+1,1), LDA, AF, LDAF, IPIVOT, B, LDB,
$      X, LDX, FERR, BERR, WORK, IWORK, INFO)
      IF (INFO .NE. 0) THEN

```



```

        PRINT 1100, INFO
        STOP 3
      END IF
      PRINT 1110
      PRINT 1080, X
      CALL DGBMV ('NO TRANSPOSE A', N, N, NSUB, NSUPER, 1.0D0,
$          A(NSUB+1,1), LDA, X, 1, 0.0D0, B1, 1)
      PRINT 1120
      PRINT 1040, B1
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(2X, F18.15))
1020 FORMAT (/1X, 'A in banded form: (* in unused elements)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (1X, F19.15)
1050 FORMAT (1X, 'Error factoring A in DGBTRF, INFO = ', I5)
1060 FORMAT (1X, 'Error solving Ax=b in DGBTRF, INFO = ', I5)
1070 FORMAT (/1X, 'Initial solution for Ax=b:')
1080 FORMAT (1X, E24.16)
1090 FORMAT (/1X, 'Ax with initial x:')
1100 FORMAT (1X, 'Error improving solution in DGBRFS, INFO = ',
$          I5)
1110 FORMAT (/1X, 'Refined solution for Ax=b:')
1120 FORMAT (/1X, 'Ax with improved x:')
C
      END

```

Sample Output

```

A in full form:
-0.0000000000000001  0.0000000000000000  0.0000000000000000  0.0000000000000000
 9.0000000000000000 -0.0000000000000001  0.0000000000000000  0.0000000000000000
 0.0000000000000000  0.2499999999999999  4.9999999999999999  0.0000000000000000
 0.0000000000000000  0.0000000000000000 -0.0000000000000001  70.0000000000000000

A in banded form: (* in unused elements)
*****
-0.0000000000000001 -0.0000000000000001  4.9999999999999999  70.0000000000000000
 9.0000000000000000  0.2499999999999999 -0.0000000000000001 *****

b:
4.0000000000000000
4.0000000000000000
4.0000000000000000
4.0000000000000000

Initial solution for Ax=b:
-0.7205759403792791E+16
-0.1168266793170336E+33
 0.5841333965851668E+31
 0.4632273902438220E+14

```

```
Ax with initial x:
 3.999999999999999
 8.000000000000000
 0.000000000000000
 4.000000000000000

Refined solution for Ax=b:
-0.7205759403792793E+16
-0.1168266793170336E+33
 0.5841333965851669E+31
 0.4632273902438221E+14

Ax with improved x:
 4.000000000000000
 0.000000000000000
 0.000000000000000
 4.500000000000000
```

Solution to a Linear System in a General Matrix in Banded Storage (Simple Driver)

The subroutines described in this section solve a linear system $AX = B$ for a general matrix A in banded storage, and general matrices B and X . Note that the expert driver `xGBSVX` is also available.

Calling Sequence

```
CALL DGBSV (N, NSUB, NSUPER, NRHS, DA, LDA, IPIVOT, DB, LDB, INFO)
CALL SGBSV (N, NSUB, NSUPER, NRHS, SA, LDA, IPIVOT, SB, LDB, INFO)
CALL ZGBSV (N, NSUB, NSUPER, NRHS, ZA, LDA, IPIVOT, ZB, LDB, INFO)
CALL CGBSV (N, NSUB, NSUPER, NRHS, CA, LDA, IPIVOT, CB, LDB, INFO)

void dgbsv (int n, int nsub, int nsuper, int nrhs, double *da, int
           lda, int *ipivot, double *db, int ldb, int *info)
void sgbsv (int n, int nsub, int nsuper, int nrhs, float *sa, int
           lda, int *ipivot, float *sb, int ldb, int *info)
void zgbsv (int n, int nsub, int nsuper, int nrhs, doublecomplex
           *za, int lda, int *ipivot, doublecomplex *zb, int ldb,
           int *info)
void cgbsv (int n, int nsub, int nsuper, int nrhs, complex *ca,
           int lda, int *ipivot, complex *cb, int ldb, int *info)
```

Arguments

N	Order of the matrix A . $N \geq 0$.
NSUB	Number of subdiagonals of A . $N-1 \geq \text{NSUB} \geq 0$ but if $N = 0$ then $\text{NSUB} = 0$.
NSUPER	Number of superdiagonals of A . $N-1 \geq \text{NSUPER} \geq 0$ but if $N = 0$ then $\text{NSUPER} = 0$.
NRHS	Number of right-hand sides, equal to the number of columns in the matrix B . $\text{NRHS} \geq 0$.
xA	On entry, the matrix A stored in rows $\text{NSUB}+1$ through $2 \times \text{NSUB} + \text{NSUPER} + 1$. Rows 1 through NSUB are ignored on entry and do not need to be set. On exit, the LU factorization of the matrix A as computed by <code>xGBTRF</code> .
LDA	Leading dimension of the array A as specified in a dimension or type statement. $\text{LDA} \geq 2 \times \text{NSUB} + \text{NSUPER} + 1$.
IPIVOT	Pivot indices as computed by <code>xGBTRF</code> .
xB	On entry, the $N \times \text{NRHS}$ right-hand side matrix B . On exit, the $N \times \text{NRHS}$ solution matrix X .

LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.	
INFO	On exit:	
	INFO = 0	Subroutine completed normally.
	INFO < 0	The <i>i</i> th argument, where $i = INFO $, had an illegal value.
	INFO > 0	$U(i,i)$, where $i = INFO$, is exactly zero and U is therefore singular. The LU factorization has been completed, but the solution could not be computed.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
  INTEGER          LDAB, LDAG, LDB, N, NRHS, NSUB, NSUPER
  INTEGER          SIZEAB, SIZEB
  PARAMETER        (NSUB = 1)
  PARAMETER        (NSUPER = 1)
  PARAMETER        (N = 4)
  PARAMETER        (NRHS = 1)
  PARAMETER        (LDAB = 2*NSUB + NSUPER + 1)
  PARAMETER        (LDAG = N)
  PARAMETER        (LDB = N)
  PARAMETER        (SIZEAB = LDAB * N)
  PARAMETER        (SIZEB = LDB * NRHS)
C
  DOUBLE PRECISION AB(LDAB,N), AG(LDAG,N), B(LDB,NRHS)
  INTEGER          ICOL, INFO, IPIVOT(N), IROW, IROWB, I1, I2
  INTEGER          M
C
  EXTERNAL          DGBSV
  INTRINSIC         MAX0, MIN0
C
  C Initialize the array AG to store the 4x4 matrix A with one
  C subdiagonal and one superdiagonal shown below. Initialize
  C the array B to store the vector b shown below.
  C
  C
  C      2  -1
  C  AG = -1  2  -1      b = 5
  C          -1  2  -1      5
  C          -1  2      5
  C
  DATA AB / SIZEAB*8.0D8 /
  DATA AG / 2.0D0, -1.0D0, 2*0.0, -1.0D0, 2.0D0, -1.0D0,
$          0.0D0, 0.0D0, -1.0D0, 2.0D0, -1.0D0, 2*0D0,
$          -1.0D0, 2.0D0 /
  DATA B / SIZEB*5.0D0 /
C
  C Copy the matrix A from the array AG to the array AB. The

```

```
C      matrix is stored in general storage mode in AG and it will
C      be stored in banded storage mode in AB. The code to copy
C      from general to banded storage mode was written for LINPACK
C      by Cleve Moler.
C
      M = NSUB + NSUPER + 1
      DO 20, ICOL = 1, N
        I1 = MAX0 (1, ICOL - NSUPER)
        I2 = MIN0 (N, ICOL + NSUB)
        DO 10, IROW = I1, I2
          IROWB = IROW - ICOL + M
          AB(IROWB,ICOL) = AG(IROW,ICOL)
10      CONTINUE
20 CONTINUE
C
C      Print the initial values of the arrays.
C
      PRINT 1000
      PRINT 1010, ((AG(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
      PRINT 1020
      PRINT 1010, ((AB(IROW,ICOL), ICOL = 1, N), IROW = 1, LDAB)
      PRINT 1030
      PRINT 1040, B
C
C      Solve the system Ax=b.
C
      CALL DGBSV (N, NSUB, NSUPER, NRHS, AB, LDAB, IPIVOT,
$      B, LDB, INFO)
      IF (INFO .EQ. 0) THEN
        PRINT 1050
        PRINT 1040, B
      ELSE
        PRINT 1060, INFO
      END IF
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F4.1))
1020 FORMAT (/1X, 'A in banded form: (* in unused elements)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (1X, 2X, F4.1)
1050 FORMAT (/1X, 'x:')
1060 FORMAT (1X, 'Error in solving system, INFO = ', I3)
C
      END
```

Sample Output

```
A in full form:
  2.0  -1.0  0.0  0.0
 -1.0   2.0 -1.0  0.0
  0.0  -1.0  2.0 -1.0
  0.0   0.0 -1.0  2.0

A in banded form: (* in unused elements)
****      ****      ****      ****
****      -1.0     -1.0     -1.0
  2.0       2.0       2.0       2.0
-1.0       -1.0       -1.0      ****

b:
  5.0
  5.0
  5.0
  5.0

x:
 10.0
 15.0
 15.0
 10.0
```

Solution to a Linear System in a General Matrix in Banded Storage (Expert Driver)

The subroutines described in this section solve a linear system $AX = B$, $A^T X = B$, or $A^H X = B$ for a general matrix A in banded storage, and general matrices B and X . These subroutines also compute forward error bounds and backward error estimates for the solution, estimate the condition number of the matrix A and, optionally, equilibrate the system before computing a solution. Note that the simple driver `xGBSV` is also available.

Calling Sequence

```
CALL DGBSVX (FACT, TRANSA, N, NSUB, NSUPER, NRHS, DA, LDA, DAF,
             LDAF, IPIVOT, EQUED, DROWSC, DCOLSC, DB, LDB, DX, LDX,
             DRCOND, DFERR, DBERR, DWORK, IWORK2, INFO)
CALL SGBSVX (FACT, TRANSA, N, NSUB, NSUPER, NRHS, SA, LDA, SAF,
             LDAF, IPIVOT, EQUED, SROWSC, SCOLSC, SB, LDB, SX, LDX,
             SRCOND, SFERR, SBERR, SWORK, IWORK2, INFO)
CALL ZGBSVX (FACT, TRANSA, N, NSUB, NSUPER, NRHS, ZA, LDA, ZAF,
             LDAF, IPIVOT, EQUED, DROWSC, DCOLSC, ZB, LDB, ZX, LDX,
             DRCOND, DFERR, DBERR, ZWORK, DWORK2, INFO)
CALL CGBSVX (FACT, TRANSA, N, NSUB, NSUPER, NRHS, CA, LDA, CAF,
             LDAF, IPIVOT, EQUED, SROWSC, SCOLSC, CB, LDB, CX, LDX,
             SRCOND, SFERR, SBERR, CWORK, SWORK2, INFO)

void dgbsvx (char fact, char trans, int n, int nsub, int nsuper,
             int nrhs, double *da, int lda, double *daf, int ldaf,
             int *ipivot, char *equed, double *drowsc, double
             *dcolsc, double *db, int ldb, double *dx, int ldx,
             double *drcond, double *dferr, double *dberr, int
             *info)
void sgbsvx (char fact, char trans, int n, int nsub, int nsuper,
             int nrhs, float *sa, int lda, float *saf, int ldaf, int
             *ipivot, char *equed, float *srowsc, float *scolsc,
             float *sb, int ldb, float *sx, int ldx, float *srcond,
             float *sferr, float *sberr, int *info)
void zgbsvx (char fact, char trans, int n, int nsub, int nsuper,
             int nrhs, doublecomplex *za, int lda, doublecomplex
             *zaf, int ldaf, int *ipivot, char *equed, double
             *drowsc, double *dcolsc, doublecomplex *zb, int ldb,
             doublecomplex *zx, int ldx, double *drcond, double
             *dferr, double *dberr, int *info)
```

```
void cgbsvx (char fact, char trans, int n, int nsub, int nsuper,
            int nrhs, complex *ca, int lda, complex *caf, int ldaf,
            int *ipivot, char *equed, float *srowsc, float *scolsc,
            complex *cb, int ldb, complex *cx, int ldx, float
            *srcond, float *sferr, float *sberr, int *info)
```

Arguments

FACT Indicates whether or not the factored form of the matrix A is supplied on entry, and if not, whether the matrix A will be equilibrated before it is factored. FACT interacts with EQUED, A, and AF as shown in this table. FACT also affects the contents of IPIVOT; see its description below.

FACT	EQUED entry	EQUED exit	A entry	A exit	AF entry	AF exit
'F', 'f'	'B', 'b'	Unchanged	Row- and col-equilibrated A	Unchanged	LU-factored row- and col-equilibrated A	Unchanged
'F', 'f'	'C', 'c'	Unchanged	Col-equilibrated A	Unchanged	LU-factored col-equilibrated A	Unchanged
'F', 'f'	'N', 'n'	Unchanged	Matrix A	Unchanged	LU-factored A	Unchanged
'F', 'f'	'R', 'r'	Unchanged	Row-equilibrated A	Unchanged	LU-factored row-equilibrated A	Unchanged
'E', 'e'	Not used	'B'	Matrix A	Row- and col-equilibrated A	Not used	LU-factored row- and col-equilibrated A
'E', 'e'	Not used	'C'	Matrix A	Col-equilibrated A	Not used	LU-factored col-equilibrated A
'E', 'e'	Not used	'N'	Matrix A	Unchanged	Not used	LU-factored A
'E', 'e'	Not used	'R'	Matrix A	Row-equilibrated A	Not used	LU-factored row-equilibrated A
'N', 'n'	Not used	'N'	Matrix A	Unchanged	Not used	LU-factored A

The above table uses the following abbreviations:

LU-factored = the factored form of A as computed by xGBTRF

Row-equilibrated A = $\text{diag}(\text{ROWSC}) \times A$

Col-equilibrated A = $A \times \text{diag}(\text{COLSC})$

Row- and col-equilibrated A = $\text{diag}(\text{ROWSC}) \times A \times \text{diag}(\text{COLSC})$

TRANSA	Indicates the form of the linear system to solve. The legal values for TRANSA are listed below. Any values not listed below are illegal. 'N' or 'n' No transpose, solve $AX = B$. 'T' or 't' Transpose, solve $A^T X = B$. 'C' or 'c' Conjugate transpose, solve $A^H X = B$. Note that A^T and A^H are the same for real matrices.
N	Order of the matrix A. $N \geq 0$.
NSUB	Number of subdiagonals of A. $N-1 \geq \text{NSUB} \geq 0$ but if $N = 0$ then $\text{NSUB} = 0$.
NSUPER	Number of superdiagonals of A. $N-1 \geq \text{NSUPER} \geq 0$ but if $N = 0$ then $\text{NSUPER} = 0$.
NRHS	Number of right-hand sides, equal to the number of columns in the matrices B and X. $\text{NRHS} \geq 0$.
xA	The contents of A vary depending on the values of FACT and EQUED. See the table for the argument FACT above.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $\text{LDA} \geq \text{NSUB} + \text{NSUPER} + 1$.
xAF	The contents of AF vary depending on the values of FACT and EQUED. See the table for the argument FACT above.
LDAF	Leading dimension of the array AF as specified in a dimension or type statement. $\text{LDA} \geq 2 \times \text{NSUB} + \text{NSUPER} + 1$.
IPIVOT	Pivot indices; values vary depending on FACT and EQUED as shown below:

FACT	EQUED Entry	EQUED Exit	IPIVOT Entry	IPIVOT Exit
'F' or 'f'	'B', 'b', 'C', 'c', 'R', 'r'	Not used	Pivot indices from LU factorization of equilibrated A as computed by xGBTRF	Unchanged
'F' or 'f'	'N' or 'n'	Not used	Pivot indices from LU factorization of A as computed by xGBTRF	Unchanged
'E' or 'e'	Not used	'B', 'C', 'R'	Not used	Pivot indices from LU factorization of equilibrated A as computed by xGBTRF

'E' or 'e'	Not used	'N'	Not used	Pivot indices from LU factorization of A as computed by xGBTRF
'N' or 'n'	Not used	'B', 'C', 'R'	Not used	Pivot indices from LU factorization of equilibrated A as computed by xGBTRF
'N' or 'n'	Not used	'N'	Not used	Pivot indices from LU factorization of A as computed by xGBTRF

EQUED

EQUED interacts with FACT and IPIVOT as shown in the tables above.

Note that one must be careful about specifying lowercase letters for the entry value of EQUED when FACT = 'F' or 'f' because ordinarily LAPACK will always return a capital letter. If the caller specifies 'b', 'c', 'n', or 'r' on entry then the lowercase letter will still be in EQUED on exit. The caller may later draw incorrect conclusions if the code contains a series of IF statements to determine whether EQUED = 'B', 'C', 'N', or 'R'.

xROWSC

On entry, if FACT = 'F' or 'f' and EQUED = 'R', 'r', 'B', or 'b', then ROWSC contains the N row scale factors that were used to equilibrate A. Each element of ROWSC must be positive. If FACT or EQUED have other values, the entry values of ROWSC are not used.

On exit, if FACT = 'E' or 'e' and EQUED = 'R' or 'B', then ROWSC contains the N row scale factors that were used to equilibrate A. If FACT or EQUED have other values, ROWSC is unchanged.

xCOLSC

On entry, if FACT = 'F' or 'f' and EQUED = 'C', 'c', 'B', or 'b', then COLSC contains the N column scale factors that were used to equilibrate A. Each element of COLSC must be positive. If FACT or EQUED have other values, the entry values of COLSC are not used. On exit, if FACT = 'E' or 'e' and EQUED = 'C' or 'B', then COLSC contains the N column scale factors that were used to equilibrate A. If FACT or EQUED have other values, COLSC is unchanged.

xB

On entry, the N×NRHS right-hand side matrix B.

On exit, the value of B is determined by the input value TRANSA and the output value of EQUED as shown below. For combinations of TRANSA and EQUED not shown below, B is not changed.

TRANSA	EQUED	Output value of B
'N', 'n'	'R' or 'B'	diag(R) × B
'T', 't', 'C', 'c'	'C' or 'B'	diag(C) × B

LDB

Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.

xX	On exit, the N×NRHS solution matrix for the original problem. Note that the solution to the original problem may differ from the solution to the equilibrated system for the cases shown below:		
	TRANSA	EQUED	Solution to equilibrated system
	'N', 'n'	'C' or 'B'	diag(C)−1 × X
	'T', 't', 'C', 'c'	'R' or 'B'	diag(R)−1 × X
LDX	Leading dimension of the array X as specified in a dimension or type statement. LDX ≥ max(1, N).		
xRCOND	On exit, an estimate of the reciprocal condition number of the matrix A after equilibration, if any, where the reciprocal condition number is defined to be 1 / (A × A ⁻¹). The reciprocal of the condition number is estimated instead of the condition number itself to avoid overflow or division by zero. If RCOND is less than machine precision (in particular, if RCOND = 0) then A is singular to working precision. In this case, INFO > 0 is returned and the solution and error bounds are not computed.		
xFERR	On exit, the estimated forward error bound for each solution vector X(*, j) for 1 ≤ j ≤ NRHS. If X' is the true solution corresponding to X(*, j) then FERR(j) is an upper bound on the magnitude of the largest element in X(*, j) − X' divided by the magnitude of the largest element in X(*, j).		
xBERR	On exit, BERR(j) is the smallest relative change in any element of A or B(*, j) that makes X(*, j) an exact solution to AX(*, j) = B(*, j) for 1 ≤ j ≤ NRHS.		
xWORK	Scratch array with a dimension of 3 × N for real subroutines or 2 × N for complex subroutines.		
xWORK2	Scratch array with a dimension of N.		
INFO	On exit:		
	INFO = 0	Subroutine completed normally.	
	INFO < 0	The ith argument, where i = INFO , had an illegal value.	
	1 ≤ INFO ≤ N	U(i,i), where i = INFO, is exactly zero and U is therefore singular. The LU factorization of A has been completed, but the solution and error bounds could not be computed.	
	INFO = N+1	RCOND is less than machine precision. The LU factorization of A has been completed but A is singular to working precision and so the solution and error bounds could not be computed.	

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER          LDAB, LDAFB, LDB, LDX, N, NRHS, NSUB
  INTEGER          NSUPER, SIZEAB, SIZEB, SIZEX
  PARAMETER        (NSUB = 1)
  PARAMETER        (NSUPER = 1)
  PARAMETER        (N = 4)
  PARAMETER        (NRHS = 1)
  PARAMETER        (LDAB = NSUB + NSUPER + 1)
  PARAMETER        (LDAFB = 2*NSUB + NSUPER + 1)
  PARAMETER        (LDB = N)
  PARAMETER        (LDX = N)
  PARAMETER        (SIZEAB = LDAB * N)
  PARAMETER        (SIZEB = LDB * NRHS)
  PARAMETER        (SIZEX = LDX * NRHS)

C
  DOUBLE PRECISION AB(LDAB,N), AFB(LDAFB,N), B(LDB,NRHS)
  DOUBLE PRECISION BERR(NRHS), COLSCA(N), FERR(NRHS), RCOND
  DOUBLE PRECISION ROWSCA(N), WORK(3*N), X(LDX,NRHS)
  INTEGER          ICOL, INFO, IPIVOT(N), IROW, IWORK(N)
  CHARACTER*1      EQUED

C
  EXTERNAL          DGBSVX
  INTRINSIC         MAX0, MIN0

C
C Initialize the array AB to store in banded storage mode the
C 4x4 matrix A with one subdiagonal and one superdiagonal
C shown below. Initialize the array B to store the vector b
C shown below.
C
C      2  -1
C  A = -1  2  -1      b = 5
C      -1  2  -1      5
C      -1  2  -1      5
C      -1  2
C
C
C  DATA AB /  8.8D8, 2.0D0, -1.0D0, -1.0D0, 2.0D0, -1.0D0,
$           -1.0D0, 2.0D0, -1.0D0, -1.0D0, 2.0D0, 8.8D8 /
C  DATA B / SIZEB*5.0D0 /
C  DATA X / SIZEX*9.9D9 /

C
C Print the initial values of the arrays.
C
  PRINT 1000
  PRINT 1010, AB(2,1), AB(1,2), 0.0D0, 0.0D0
  PRINT 1010, AB(3,1), AB(2,2), AB(1,3), 0.0D0
  PRINT 1010, 0.0D0, AB(3,2), AB(2,3), AB(1,4)
  PRINT 1010, 0.0D0, 0.0D0, AB(3,3), AB(2,4)
  PRINT 1020
  PRINT 1010, ((AB(IROW,ICOL), ICOL = 1, N), IROW = 1, LDAB)
  PRINT 1030
  PRINT 1040, B

```

```

C
C      Solve the system Ax=b.
C
      CALL DGBSVX ('EQUILIBRATE A', 'NO TRANSPOSE A', N, NSUB,
$          NSUPER, NRHS, AB, LDAB, AFB, LDAFB, IPIVOT,
$          EQUED, ROWSCA, COLSCA, B, LDB, X, LDX, RCOND,
$          FERR, BERR, WORK, IWORK, INFO)
      IF (INFO .EQ. 0) THEN
          PRINT 1050
          PRINT 1040, X
          PRINT 1060, 1.0D0 / RCOND
          IF (EQUED .EQ. 'N') PRINT 1070
          IF (EQUED .EQ. 'R') PRINT 1080
          IF (EQUED .EQ. 'C') PRINT 1090
          IF (EQUED .EQ. 'B') PRINT 1100
          PRINT 1110, (IROW, BERR(IROW), IROW = 1, NRHS)
          PRINT 1120, (IROW, FERR(IROW), IROW = 1, NRHS)
      ELSE
          PRINT 1130, INFO
          IF (INFO .EQ. (N + 1)) THEN
              PRINT 1140
          END IF
      END IF
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F4.1))
1020 FORMAT (/1X, 'A in banded form: (* in unused elements)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (1X, 2X, F4.1)
1050 FORMAT (/1X, 'x:')
1060 FORMAT (/1X, 'Estimated condition number of A: ', F7.2)
1070 FORMAT (1X, 'No equilibration was required for A.')
1080 FORMAT (1X, 'Row equilibration was done on A.')
1090 FORMAT (1X, 'Column equilibration was done on A.')
1100 FORMAT (1X, 'Row and column equilibration was done on A.')
1110 FORMAT (/1X, 'Backward error for system #', I1, ': ', E12.6)
1120 FORMAT (1X, 'Forward error for system #', I1, ': ', E12.6)
1130 FORMAT (1X, 'Error in solving system, INFO = ', I3)
1140 FORMAT (1X, 'Matrix is singular to working precision.')
C
      END

```

Sample Output

```

A in full form:
  2.0   -1.0    0.0    0.0
 -1.0    2.0   -1.0    0.0
  0.0   -1.0    2.0   -1.0
  0.0    0.0   -1.0    2.0

```

```
A in banded form: (* in unused elements)
****  -1.0  -1.0  -1.0
      2.0   2.0   2.0   2.0
      -1.0  -1.0  -1.0  ****

b:
   5.0
   5.0
   5.0
   5.0

x:
  10.0
  15.0
  15.0
  10.0

Estimated condition number of A:  12.00
No equilibration was required for A.

Backward error for system #1: 0.592119E-16
Forward error for system #1: 0.511591E-14
```

LU Factorization of a General Matrix in Banded Storage

The subroutines described in this section compute an LU factorization of a general matrix A in banded storage. It is typical to follow a call to xGBTRF with a call to xGBSVX or xGBTRS to solve a linear system $AX = B$, or to xGBCON to estimate the condition number of A .

Calling Sequence

```
CALL DGBTRF (M, N, NSUB, NSUPER, DA, LDA, IPIVOT, INFO)
CALL SGBTRF (M, N, NSUB, NSUPER, SA, LDA, IPIVOT, INFO)
CALL ZGBTRF (M, N, NSUB, NSUPER, ZA, LDA, IPIVOT, INFO)
CALL CGBTRF (M, N, NSUB, NSUPER, CA, LDA, IPIVOT, INFO)
```

```
void dgbtrf (int m, int n, int nsub, int nsuper, double *da, int
            lda, int *ipivot, int *info)
void sgbtrf (int m, int n, int nsub, int nsuper, float *sa, int
            lda, int *ipivot, int *info)
void zgbtrf (int m, int n, int nsub, int nsuper, doublecomplex *za,
            int lda, int *ipivot, int *info)
void cgbtrf (int m, int n, int nsub, int nsuper, complex *ca, int
            lda, int *ipivot, int *info)
```

Arguments

M	Number of rows in the matrix A . $M \geq 0$.
N	Number of columns in the matrix A . $N \geq 0$.
NSUB	Number of subdiagonals of A . $N-1 \geq \text{NSUB} \geq 0$ but if $N = 0$ then $\text{NSUB} = 0$.
NSUPER	Number of superdiagonals of A . $N-1 \geq \text{NSUPER} \geq 0$ but if $N = 0$ then $\text{NSUPER} = 0$.
xA	On entry, the matrix A stored in rows $\text{NSUB} + 1$ through $2 \times \text{NSUB} + \text{NSUPER} + 1$. Rows 1 through NSUB are ignored on entry and do not need to be set. On exit, an LU factorization of A .
LDA	Leading dimension of the array A as specified in a dimension or type statement. $\text{LDA} \geq 2 \times \text{NSUB} + \text{NSUPER} + 1$.
IPIVOT	On exit, the pivot indices. During factorization, row i was exchanged with row $\text{IPIVOT}(i)$ for $1 \leq i \leq \min(M, N)$
INFO	On exit: INFO = 0 Subroutine completed normally.

INFO < 0 The *i*th argument, where *i* = |INFO|, had an illegal value.

INFO > 0 U(*i*,*i*), where *i* = INFO, is exactly zero and U is therefore singular. The factorization has been completed, but division by zero will occur if this factorization is used to solve a linear system.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C   INTEGER          LDAB, LDAG, LDB, M, N, NRHS, NSUB, NSUPER
C   PARAMETER        (NSUB = 1)
C   PARAMETER        (NSUPER = 1)
C   PARAMETER        (M = 4)
C   PARAMETER        (N = 4)
C   PARAMETER        (NRHS = 1)
C   PARAMETER        (LDAB = 2*NSUB + NSUPER + 1)
C   PARAMETER        (LDAG = M)
C   PARAMETER        (LDB = N)
C
C   DOUBLE PRECISION AB(LDAB,N), AG(LDAG,N), B(LDB,NRHS)
C   INTEGER          ICOL, INFO, IPIVOT(N), IROW, IROWB, I1, I2
C   INTEGER          M1
C
C   EXTERNAL          DGBTRF, DGBTRS
C   INTRINSIC         MAX0, MIN0
C
C   Initialize the array AG to store the 4x4 matrix A with one
C   subdiagonal and one superdiagonal shown below. Initialize
C   the array B to store the vector b shown below.
C
C   AG =   2  -1           b = 5
C   AG = -1  2  -1           b = 5
C           -1  2  -1           b = 5
C           -1  2           b = 5
C
C   DATA AB / 16 * 8.0D8 /
C   DATA AG / 2.0D0, -1.0D0, 2*0.0, -1.0D0, 2.0D0, -1.0D0,
C   $          0.0D0, 0.0D0, -1.0D0, 2.0D0, -1.0D0, 2*0D0,
C   $          -1.0D0, 2.0D0 /
C   DATA B / N*5.0D0 /
C
C   Copy the matrix A from the array AG to the array AB. The
C   matrix is stored in general storage mode in AG and it will
C   be stored in banded storage mode in AB. The code to copy
C   from general to banded storage mode was written for LINPACK
C   by Cleve Moler.
C
C   M1 = NSUB + NSUPER + 1

```



```

      DO 20, ICOL = 1, N
        I1 = MAX0 (1, ICOL - NSUPER)
        I2 = MIN0 (N, ICOL + NSUB)
        DO 10, IROW = I1, I2
          IROWB = IROW - ICOL + M1
          AB(IROWB,ICOL) = AG(IROW,ICOL)
10      CONTINUE
20 CONTINUE
C
C      Print the initial values of the arrays.
C
      PRINT 1000
      PRINT 1010, ((AG(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
      PRINT 1020
      PRINT 1010, ((AB(IROW,ICOL), ICOL = 1, N), IROW = 1, LDAB)
      PRINT 1030
      PRINT 1040, B
C
C      Factor the matrix in banded form and print the results.
C
      CALL DGBTRF (M, N, NSUB, NSUPER, AB, LDAB, IPIVOT, INFO)
      IF (INFO .EQ. 0) THEN
        CALL DGBTRS ('NO TRANSPOSE A', N, NSUB, NSUPER, NRHS,
$          AB, LDAB, IPIVOT, B, LDB, INFO)
        IF (INFO .EQ. 0) THEN
          PRINT 1050
          PRINT 1040, B
        ELSE
          PRINT 1060, INFO
        END IF
      ELSE
        PRINT 1070, INFO
      END IF
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F4.1))
1020 FORMAT (/1X, 'A in banded form: (* in unused elements)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (1X, 2X, F4.1)
1050 FORMAT (/1X, 'x:')
1060 FORMAT (1X, 'Error in solving system, INFO = ', I4)
1070 FORMAT (1X, 'Error in factoring system, INFO = ', I4)
C
      END

```

Sample Output

```
A in full form:
  2.0  -1.0  0.0  0.0
 -1.0   2.0 -1.0  0.0
  0.0  -1.0  2.0 -1.0
  0.0   0.0 -1.0  2.0

A in banded form: (* in unused elements)
****      ****      ****      ****
****      -1.0     -1.0     -1.0
  2.0       2.0       2.0       2.0
-1.0       -1.0       -1.0      ****

b:
  5.0
  5.0
  5.0
  5.0

x:
 10.0
 15.0
 15.0
 10.0
```

Solution to a Linear System in an LU-Factored General Matrix in Banded Storage

The subroutines described in this section solve a linear system $AX = B$, $A^T X = B$, or $A^H X = B$ for a general matrix A in banded storage, which has been LU-factored by xGBTRF, and general matrices B and X .

Calling Sequence

```
CALL DGBTRS (TRANSA, N, NSUB, NSUPER, NRHS, DA, LDA, IPIVOT, DB,
LDB, INFO)
CALL SGBTRS (TRANSA, N, NSUB, NSUPER, NRHS, SA, LDA, IPIVOT, SB,
LDB, INFO)
CALL ZGBTRS (TRANSA, N, NSUB, NSUPER, NRHS, ZA, LDA, IPIVOT, ZB,
LDB, INFO)
CALL CGBTRS (TRANSA, N, NSUB, NSUPER, NRHS, CA, LDA, IPIVOT, CB,
LDB, INFO)

void dgbtrs (char trans, int n, int nsub, int nsuper, int nrhs,
double *da, int lda, int *ipivot, double *db, int ldb,
int *info)
void sgbtrs (char trans, int n, int nsub, int nsuper, int nrhs,
float *sa, int lda, int *ipivot, float *sb, int ldb,
int *info)
void zgbtrs (char trans, int n, int nsub, int nsuper, int nrhs,
doublecomplex *za, int lda, int *ipivot, doublecomplex
*zb, int ldb, int *info)
void cgbtrs (char trans, int n, int nsub, int nsuper, int nrhs,
complex *ca, int lda, int *ipivot, complex *cb, int
ldb, int *info)
```

Arguments

TRANSA	Indicates the form of the linear system to solve. The legal values for TRANSA are listed below. Any values not listed below are illegal. 'N' or 'n' No transpose, solve $AX = B$. 'T' or 't' Transpose, solve $A^T X = B$. 'C' or 'c' Conjugate transpose, solve $A^H X = B$. Note that $A^T = A^H$ for real matrices.
N	Order of the matrix A . $N \geq 0$.
NSUB	Number of subdiagonals of A . $N-1 \geq \text{NSUB} \geq 0$ but if $N = 0$ then $\text{NSUB} = 0$.

NSUPER	Number of superdiagonals of A. $N-1 \geq \text{NSUPER} \geq 0$ but if $N = 0$ then $\text{NSUPER} = 0$.
NRHS	Number of right-hand sides, equal to the number of columns in the matrix B. $\text{NRHS} \geq 0$.
xA	LU factorization of the matrix A as computed by xGBTRF.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $\text{LDA} \geq 2 \times \text{NSUB} + \text{NSUPER} + 1$.
IPIVOT	Pivot indices as computed by xGBTRF.
xB	On entry, the $N \times \text{NRHS}$ right-hand side matrix B. On exit, the $N \times \text{NRHS}$ solution matrix X.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $\text{LDB} \geq \max(1, N)$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where $i = \text{INFO} $, had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
C      INTEGER          LDAB, LDAG, LDB, M, N, NRHS, NSUB, NSUPER
C      PARAMETER        (NSUB = 1)
C      PARAMETER        (NSUPER = 1)
C      PARAMETER        (M = 4)
C      PARAMETER        (N = 4)
C      PARAMETER        (NRHS = 1)
C      PARAMETER        (LDAB = 2*NSUB + NSUPER + 1)
C      PARAMETER        (LDAG = M)
C      PARAMETER        (LDB = N)
C
C      DOUBLE PRECISION  AB(LDAB,N), AG(LDAG,N), B(LDB,NRHS)
C      INTEGER           ICOL, INFO, IPIVOT(N), IROW, IROWB, I1, I2
C      INTEGER           M1
C
C      EXTERNAL          DGBTRF, DGBTRS
C      INTRINSIC         MAX0, MIN0
C
C      Initialize the array AG to store the 4x4 matrix A with one
C      subdiagonal and one superdiagonal shown below. Initialize
C      the array B to store the vector b shown below.
C
C      2   -1   5
C      AG = -1   2   -1   b = 5
C           -1   2   -1   5
C           -1   2   5
C

```

```

DATA AB / 16 * 8.0D8 /
DATA AG / 2.0D0, -1.0D0, 2*0.0, -1.0D0, 2.0D0, -1.0D0,
$      0.0D0, 0.0D0, -1.0D0, 2.0D0, -1.0D0, 2*0D0,
$      -1.0D0, 2.0D0 /
DATA B / N*5.0D0 /
C
C      Copy the matrix A from the array AG to the array AB.  The
C      matrix is stored in general storage mode in AG and it will
C      be stored in banded storage mode in AB.  The code to copy
C      from general to banded storage mode was written for LINPACK
C      by Cleve Moler.
C
      M1 = NSUB + NSUPER + 1
      DO 20, ICOL = 1, N
        I1 = MAX0 (1, ICOL - NSUPER)
        I2 = MIN0 (N, ICOL + NSUB)
        DO 10, IROW = I1, I2
          IROWB = IROW - ICOL + M1
          AB(IROWB,ICOL) = AG(IROW,ICOL)
10      CONTINUE
20      CONTINUE
C
C      Print the initial values of the arrays.
C
      PRINT 1000
      PRINT 1010, ((AG(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
      PRINT 1020
      PRINT 1010, ((AB(IROW,ICOL), ICOL = 1, N), IROW = 1, LDAB)
      PRINT 1030
      PRINT 1040, B
C
C      Factor the matrix in banded form and print the results.
C
      CALL DGBTRF (M, N, NSUB, NSUPER, AB, LDAB, IPIVOT, INFO)
      IF (INFO .EQ. 0) THEN
        CALL DGBTRS ('NO TRANSPOSE A', N, NSUB, NSUPER, NRHS,
$          AB, LDAB, IPIVOT, B, LDB, INFO)
        IF (INFO .EQ. 0) THEN
          PRINT 1050
          PRINT 1040, B
        ELSE
          PRINT 1060, INFO
        END IF
      ELSE
        PRINT 1070, INFO
      END IF
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F4.1))
1020 FORMAT (/1X, 'A in banded form: (* in unused elements)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (1X, 2X, F4.1)
1050 FORMAT (/1X, 'x:')
1060 FORMAT (1X, 'Error in solving system, INFO = ', I4)
1070 FORMAT (1X, 'Error in factoring system, INFO = ', I4)

```

```
C
    END
```

Sample Output

```
A in full form:
  2.0  -1.0   0.0   0.0
 -1.0   2.0  -1.0   0.0
  0.0  -1.0   2.0  -1.0
  0.0   0.0  -1.0   2.0

A in banded form: (* in unused elements)
****      ****      ****      ****
****  -1.0  -1.0  -1.0
  2.0   2.0   2.0   2.0
-1.0  -1.0  -1.0  ****

b:
  5.0
  5.0
  5.0
  5.0

x:
 10.0
 15.0
 15.0
 10.0
```

Reciprocal Condition Number of an LU-Factored General Matrix

The subroutines described in this section estimate the reciprocal condition number, in either the 1-norm or the ∞ -norm, of a general matrix A, which has been LU-factored by xGETRF.

Calling Sequence

```
CALL DGECON (NORM, N, DA, LDA, DANORM, DRCOND, DWORK, IWORK2, INFO)
CALL SGECON (NORM, N, SA, LDA, SANORM, SRCOND, SWORK, IWORK2, INFO)
CALL ZGECON (NORM, N, ZA, LDA, DANORM, DRCOND, ZWORK, DWORK2, INFO)
CALL CGECON (NORM, N, CA, LDA, SANORM, SRCOND, CWORK, SWORK2, INFO)
```

```
void dgecon (char norm, int n, double *da, int lda, double danorm,
             double *drcond, int *info)
void sgecon (char norm, int n, float *sa, int lda, float sanorm,
             float *srcond, int *info)
void zgecon (char norm, int n, doublecomplex *za, int lda, double
             danorm, double *drcond, int *info)
void cgecon (char norm, int n, complex *ca, int lda, float sanorm,
             float *srcond, int *info)
```

Arguments

NORM	Indicates whether the 1-norm condition number or the ∞ -norm condition number is required. The legal values for NORM are listed below. Any values not listed below are illegal. 'O', 'o', or '1' 1-norm condition number is required. 'T' or 'i' ∞ -norm condition number is required.
N	Order of the matrix A. $N \geq 0$.
xA	LU factorization of the matrix A, as computed by xGETRF.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
xANORM	On entry, if NORM = 'O', 'o', or '1', then ANORM contains the 1-norm of the original matrix A; if NORM = 'T' or 'i' then ANORM contains the ∞ -norm of the original matrix A.
xRCOND	On exit, the estimated reciprocal of the condition number of the matrix A where the reciprocal condition number of A is defined to be $1 / (ANORM \times \ A^{-1}\)$. The reciprocal of the condition number is estimated instead of the condition number itself to avoid overflow or division by zero. If RCOND is less than machine precision (in particular, if RCOND = 0) then A is singular to working precision.

xWORK	Scratch array with a dimension of $4 \times N$ for real subroutines or $2 \times N$ for complex subroutines.
xWORK2	Scratch array with a dimension of N for real subroutines or $2 \times N$ for complex subroutines.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The <i>i</i> th argument, where $i = \text{INFO} $, had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
C   INTEGER          LDA, M, N, NRHS
C   PARAMETER        (M = 3)
C   PARAMETER        (N = 3)
C   PARAMETER        (NRHS = 1)
C   PARAMETER        (LDA = M)
C
C   DOUBLE PRECISION A(LDA,N), ANORM, RCOND, WORK(4*N)
C   INTEGER          ICOL, INFO, IROW, IPIVOT(N), IWORK(N)
C
C   EXTERNAL          DGECON, DGETRF
C
C   Initialize the array A to store the coefficient matrix A
C   shown below.
C
C       1  2  2
C   A = 2  1  2
C       2  2  1
C
C   DATA A / 1.0D0, 3*2.0D0, 1.0D0, 3*2.0D0, 1.0D0 /
C
C   Print the initial value of A.
C
C   PRINT 1000
C   PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, M)
C
C   LU factor A, then use the factorization to compute the
C   condition number.
C
C   CALL DGETRF (M, N, A, LDA, IPIVOT, INFO)
C   IF (INFO .EQ. 0) THEN
C       ANORM = 5.0D0
C       CALL DGECON ('ONE-NORM', N, A, LDA, ANORM, RCOND,
C   $               WORK, IWORK, INFO)
C       IF (INFO .EQ. 0) THEN
C           PRINT 1020, 1.0D0 / RCOND
C       ELSE
C           PRINT 1030, INFO

```



```
        END IF
        ELSE
            PRINT 1040, INFO
        END IF
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (3(3X, F4.1))
1020 FORMAT (/1X, 'Estimated condition number: ', F9.6)
1030 FORMAT (1X, 'Error computing condition number, INFO = ', I6)
1040 FORMAT (1X, 'Factorization failed with INFO = ', I6)
C
        END
```

Sample Output

```
A:
    1.0    2.0    2.0
    2.0    1.0    2.0
    2.0    2.0    1.0

Estimated condition number:  7.000000
```

Equilibration Scale Factors for a General Matrix

The subroutines described in this section compute row and column scalings intended to equilibrate a general matrix A and reduce its condition number. The scale factors are chosen to try to make largest element of the equilibrated matrix have magnitude equal to 1. Note that these subroutines only compute the scale factors for A. Additional code must be written to actually equilibrate A. Each element of the equilibrated matrix B has the value $B(i,j) = \text{ROWSC}(i) \times B(i,j) \times \text{COLSC}(j)$.

Calling Sequence

```
CALL DGEEQU (M, N, DA, LDA, DROWSC, DCOLSC, DROWCN, DCOLCN, DAMAX,
             INFO)
CALL SGEEQU (M, N, SA, LDA, SROWSC, SCOLSC, SROWCN, SCOLCN, SAMAX,
             INFO)
CALL ZGEEQU (M, N, ZA, LDA, DROWSC, DCOLSC, DROWCN, DCOLCN, DAMAX,
             INFO)
CALL CGEEQU (M, N, CA, LDA, SROWSC, SCOLSC, SROWCN, SCOLCN, SAMAX,
             INFO)

void dgeequ (int m, int n, double *da, int lda, double *drowsc,
             double *dcolsc, double *drowcn, double *dcolcn, double
             *damax, int *info)
void sgeequ (int m, int n, float *sa, int lda, float *srowsc, float
             *scolsc, float *srowcn, float *scolcn, float *samax,
             int *info)
void zgeequ (int m, int n, doublecomplex *za, int lda, double
             *drowcn, double *dcolcn, double *drowcn, double
             *dcolcn, double *damax, int *info)
void cgeequ (int m, int n, complex *ca, int lda, float *srowsc,
             float *scolsc, float *srowcn, float *scolcn, float
             *samax, int *info)
```

Arguments

M	Number of rows in the matrix A. $M \geq 0$.
N	Number of columns in the matrix A. $N \geq 0$.
xA	Matrix A.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, M)$.

xROWSC	On exit, if INFO = 0 or INFO > M, ROWSC contains the row scale factors for the matrix A.
xCOLSC	On exit, if INFO = 0, COLSC contains the column scale factors for the matrix A.
xROWCN	On exit, if INFO = 0 or INFO > M, ROWCN contains the ratio of the smallest ROWSC(i) to the largest ROWSC(i). If ROWCN ≥ 0.1 and AMAX is not close to overflow or underflow, then it is not worth scaling by ROWSC.
xCOLCN	On exit, if INFO = 0, COLCN contains the ratio of the smallest COLSC(i) to the largest COLSC(i). If COLCN ≥ 0.1 and AMAX is not close to overflow or underflow, then it is not worth scaling by COLSC.
xAMAX	On exit, the absolute value of the largest element of A. If AMAX is very close to overflow or underflow, A should be scaled regardless of the value of ROWCN or COLCN.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The <i>i</i> th argument, where <i>i</i> = INFO , had an illegal value. 1 $\leq$ INFO $\leq$ M Row INFO of the array A is exactly zero. INFO > M Column (INFO - M) of the array A is exactly zero.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
C   INTEGER          LDA, N
C   PARAMETER       (N = 3)
C   PARAMETER       (LDA = N)
C
C   DOUBLE PRECISION A(LDA,N), AMAX, COLSCA(N), COLCND
C   DOUBLE PRECISION ROWCND, ROWSCA(N)
C   INTEGER          ICOL, INFO, IROW
C
C   EXTERNAL         DGEEQU
C   INTRINSIC        ABS
C
C   Initialize the array A to store the 3x3 matrix A shown
C   below.
C
C       2    4    8
C   A =  2    8   16
C       2   16   32
C
C   DATA A / 2.0D0, 2.0D0, 2.0D0, 4.0D0, 8.0D0, 1.6D1,

```

```

$      8.0D0, 1.6D1, 3.2D1 /
C
C      Print the initial value of A.
C
      PRINT 1000
      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, LDA)
C
C      Compute and print the scale factors that will
C      equilibrate A.
C
      CALL DGEEQU (N, N, A, LDA, ROWSCA, COLSCA, ROWCND, COLCND,
$              AMAX, INFO)
      IF (INFO .LT. 0) THEN
          PRINT 1020, ABS(INFO)
          STOP 1
      ELSE IF (INFO .GT. N) THEN
          PRINT 1030, INFO - N
          STOP 2
      ELSE IF (INFO .GT. 0) THEN
          PRINT 1040, INFO
          STOP 3
      END IF
      PRINT 1050
      PRINT 1010, ROWSCA
      PRINT 1060
      PRINT 1010, COLSCA
C
C      Use the scale factors to equilibrate A and print the
C      equilibrated A.
C
      DO 110, ICOL = 1, N
          DO 100, IROW = 1, N
              A(IROW,ICOL) = A(IROW,ICOL) * ROWSCA(IROW) *
$                  COLSCA(ICOL)
100      CONTINUE
110      CONTINUE
      PRINT 1070
      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, LDA)
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (3(3X, F8.5))
1020 FORMAT (1X, 'Illegal argument to DGEEQU, argument #', I1)
1030 FORMAT (1X, 'Column ', I1, ' of A is exactly zero.')
1040 FORMAT (1X, 'Row ', I1, ' of A is exactly zero.')
1050 FORMAT (/1X, 'Row scale factors:')
1060 FORMAT (/1X, 'Column scale factors:')
1070 FORMAT (/1X, 'Scaled A:')
C
      END

```

Sample Output

```
A:
  2.00000    4.00000    8.00000
  2.00000    8.00000   16.00000
  2.00000   16.00000   32.00000

Row scale factors:
  0.12500    0.06250    0.03125

Column scale factors:
  4.00000    2.00000    1.00000

Scaled A:
  1.00000    1.00000    1.00000
  0.50000    1.00000    1.00000
  0.25000    1.00000    1.00000
```

Eigenvalues, Schur Form, and Schur Factorization of a General Matrix (Simple Driver)

The subroutines described in this section compute the eigenvalues and real Schur form T of a square general matrix A . Optionally they can also compute the matrix Z of Schur vectors, which gives the Schur factorization of $A=ZTZ^T$ for a real A or $A=ZTZ^H$ for a complex A . They can also optionally order the eigenvalues on the diagonal of the real Schur form with selected eigenvalues at the top left so that the leading columns of Z form an orthonormal basis for the invariant subspace corresponding to the selected eigenvalues. Note that the expert driver xGEESS is also available.

Calling Sequence

```
CALL DGEES (JOBZ, SORTEV, SELECT, N, DA, LDA, NOUT, DWR, DWI, DZ,
            LDZ, DWORK, LDWORK, BWORK3, INFO)
CALL SGEES (JOBZ, SORTEV, SELECT, N, SA, LDA, NOUT, SWR, SWI, SZ,
            LDZ, SWORK, LDWORK, BWORK3, INFO)
CALL ZGEES (JOBZ, SORTEV, SELECT, N, ZA, LDA, NOUT, ZW, ZZ, LDZ,
            ZWORK, LDWORK, DWORK2, BWORK3, INFO)
CALL CGEES (JOBZ, SORTEV, SELECT, N, CA, LDA, NOUT, CW, CZ, LDZ,
            CWORK, LDWORK, SWORK2, BWORK3, INFO)

void dgees (char jobvs, char sort, L_fp select, int n, double *da,
            int lda, int *sdim, double *dwr, double *dwi, double
            *dz, int ldz, int *info)
void sgees (char jobvs, char sort, L_fp select, int n, float *sa,
            int lda, int *sdim, float *swr, float *swi, float *sz,
            int ldz, int *info)
void zgees (char jobvs, char sort, L_fp select, int n,
            doublecomplex *za, int lda, int *sdim, doublecomplex
            *zw, doublecomplex *zz, int ldz, int *info)
void cgees (char jobvs, char sort, L_fp select, int n, complex
            *ca, int lda, int *sdim, complex *cw, complex *cz, int
            ldz, int *info)
```

Arguments

JOBZ	Indicates whether or not the matrix Z of Schur vectors will be computed. The legal values for JOBZ are listed below. Any values not listed below are illegal. 'N' or 'n' Z will not be computed. 'V' or 'v' Z will be computed.
SORTEV	Indicates whether or not to order the eigenvalues on the diagonal of the Schur form. The legal values for SORTEV are listed below. Any values not listed below are illegal. 'N' or 'n' Eigenvalues will not be ordered. 'S' or 's' Eigenvalues will be ordered on the diagonal as determined by the argument SELECT.
SELECT	If SORTEV = 'S' or 's' then SELECT is used to select eigenvalues to sort to the top left of the Schur form. If SORTEV = 'N' or 'n' then SELECT is not used. SELECT is a function of logical type that must be declared in an EXTERNAL statement in the caller. SELECT may no longer return a value of true for a given eigenvalue after sorting because the process of ordering may change the value of complex eigenvalues, especially if the eigenvalue is ill-conditioned. If this happens, INFO is set to N+2 as described below. Aside from the description given here, the details of SELECT vary for real and complex subroutines; these two cases are described next. For real subroutines, SELECT has two real arguments which are the real and imaginary parts of an eigenvalue. An eigenvalue is selected to be sorted to the top left of the real Schur form if SELECT returns a value of true. If one member of a complex conjugate pair of eigenvalues is selected, then both complex eigenvalues are selected. For complex subroutines, SELECT has a single complex argument. The eigenvalue is selected to be sorted to the top left of the Schur form if SELECT returns a value of true.
N	Order of the matrix A. $N \geq 0$.
xA	On entry, the matrix A. On exit, A has been overwritten by its Schur form T.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
NOUT	On exit, the number of eigenvalues (after ordering) for which SELECT returns a value of true. For real subroutines, complex conjugate pairs for which SELECT is true for either eigenvalue count as 2. If SORTEV = 'N' or 'n' then NOUT = 0.

xWR, xWI	(For real subroutines) Real and imaginary parts, respectively, of the N computed eigenvalues in the same order that they appear on the diagonal of the output Schur form T. Complex conjugate pairs of eigenvalues will appear consecutively with the eigenvalue having the positive imaginary part first.
xW	(For complex subroutines) N computed eigenvalues in the same order that they appear on the diagonal of the output Schur form T. Complex conjugate pairs of eigenvalues will appear consecutively with the eigenvalue having the positive imaginary part first.
xZ	On exit, if JOBZ = 'V' or 'v' then Z contains the orthogonal or unitary matrix Z of Schur vectors. If JOBZ = 'N' or 'n' then Z is not used.
LDZ	Leading dimension of the array Z as specified in a dimension or type statement. $LDZ \geq 1$. If JOBZ = 'V' or 'v' then $LDZ \geq \max(1, N)$.
xWORK	Scratch array with a dimension of LDWORK.
LDWORK	Leading dimension of the array WORK as specified in a dimension or type statement. $LDWORK \geq \max(1, 3 \times N)$ for real subroutines or $\max(1, 2 \times N)$ for complex subroutines.
xWORK2	Scratch array with a dimension N for complex subroutines.
BWORK3	Scratch array BWORK3(N) of logical type. If SORTEV = 'N' or 'n' then BWORK3 is not used.
INFO	On exit: INFO = 0 INFO < 0 $1 \leq \text{INFO} \leq N$ INFO = N+1 INFO = N+2 Subroutine completed normally. The ith argument, where $i = \text{INFO}$, had an illegal value. Convergence failure. The eigenvalues could not be reordered because some eigenvalues were too close to separate. It is likely that the problem is very ill-conditioned. After reordering, roundoff changed values of some complex eigenvalues so that SELECT no longer returns a value of true for leading eigenvalues in the Schur form. This could also be caused by underflow due to scaling.

Sample Program

```
PROGRAM TEST
IMPLICIT NONE
```



```
C
      INTEGER          LDA, LDVS, LDWORK, N
      PARAMETER        (N = 2)
      PARAMETER        (LDA = N)
      PARAMETER        (LDVS = 2 * N)
      PARAMETER        (LDWORK = 3 * N)
C
      DOUBLE PRECISION A(LDA,N), TEMP, VS(LDVS,N)
      DOUBLE PRECISION WORK(LDWORK), WR(N), WI(N)
      INTEGER          I, ICOL, INFO, IROW, SORT
C
      EXTERNAL          DGEES, SELECT
      INTRINSIC         ABS
C
      Initialize the array A to store the matrix A shown below.
C
C      A = 0  1
C          -6 5
C
      DATA A / 0.0D0, -6.0D0, 1.0D0, 5.0D0 /
C
      Print the initial value of A.
C
      PRINT 1000
      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
C
      Compute the eigenvalues and Schur vectors of A.
C
      CALL DGEES ('VECTORS AND EIGENVALUES', 'NOT SORTED', SELECT,
$              N, A, LDA, SORT, WR, WI, VS, LDVS, WORK, LDWORK,
$              TEMP, INFO)
      IF (INFO .NE. 0) THEN
        IF (INFO .LT. 0) THEN
          PRINT 1020, ABS(INFO)
          STOP 1
        ELSEIF (INFO .LE. N) THEN
          PRINT 1030, INFO
          STOP 2
        ELSEIF (INFO .EQ. N + 1) THEN
          PRINT 1040
        ELSE
          PRINT 1050
        END IF
      END IF
C
      Print the eigenvalues and Schur vectors.
C
      PRINT 1060
      DO 120, IROW = 1, N
        PRINT 1070, WR(IROW), (VS(I,IROW), I = 1, N)
      120 CONTINUE
C
      1000 FORMAT (1X, 'A:')
      1010 FORMAT (2(3X, F4.1))
      1020 FORMAT (/1X, 'Illegal argument to DGEES, argument #', I2)
      1030 FORMAT (/1X, 'QR failed to converge, INFO = ', I2)
```

```

1040 FORMAT (/1X, 'Eigenvalues could not be reordered; problem',
$          /1X, 'is very ill-conditioned.')
1050 FORMAT (/1X, 'Leading eigenvalues in the Schur form no',
$          /1X, 'longer satisfy SELECT = .TRUE.')
1060 FORMAT (/1X, 'Eigenvalue', 4X, 'Schur vector**T')
1070 FORMAT (1X, F5.2, 10X, '[', F4.2, ', ', F4.2, ']')
C
      END

      LOGICAL FUNCTION SELECT (ARG1, ARG2)
      IMPLICIT NONE
C
      DOUBLE PRECISION ARG1, ARG2
      INTRINSIC INT
C
      The value computed is always .TRUE. It is computed in
      the peculiar way below to avoid compiler messages about
      unused function arguments.
C
      PRINT 1000
      SELECT = (ARG1 .EQ. ARG2)
      STOP
C
1000 FORMAT (///1X, '**** ERROR: ',
$          'SELECT FUNCTION CALLED BUT NOT AVAILABLE. ****')
C
      END

```

Sample Output

```

A:
   0.0    1.0
  -6.0    5.0

Eigenvalue  Schur vector**T
   2.00      [0.45, 0.89]
   3.00      [-.89, 0.45]

```

Eigenvalues, Schur Form, and Schur Factorization of a General Matrix (Expert Driver)

The subroutines described in this section compute the eigenvalues and real Schur form T of a square general matrix A . Optionally they can also compute the matrix Z of Schur vectors, which gives the Schur factorization of $A=ZTZ^T$ for a real A or $A=ZTZ^H$ for a complex A . They can also order the eigenvalues on the diagonal of the real Schur form with selected eigenvalues at the top left, compute a reciprocal condition number for the average of the selected eigenvalues, and compute a reciprocal condition number for the right invariant subspace corresponding to the selected eigenvalues. The leading columns of Z form an orthonormal basis for this invariant subspace. Note that the simple driver xGEES is also available.

Calling Sequence

```
CALL DGEESX (JOBZ, SORTEV, SELECT, SENSE, N, DA, LDA, NOUT, DWR,
             DWI, DZ, LDZ, SRCONE, DRCONV, DWORK, LDWORK, IWORK2,
             LDWRK2, BWORK3, INFO)
CALL SGEESX (JOBZ, SORTEV, SELECT, SENSE, N, SA, LDA, NOUT, SWR,
             SWI, SZ, LDZ, SRCONE, SRCONV, SWORK, LDWORK, IWORK2,
             LDWRK2, BWORK3, INFO)
CALL ZGEESX (JOBZ, SORTEV, SELECT, SENSE, N, ZA, LDA, NOUT, ZW, ZZ,
             LDZ, DRONE, DRCONV, ZWORK, LDWORK, DWORK2, BWORK3,
             INFO)
CALL CGEESX (JOBZ, SORTEV, SELECT, SENSE, N, CA, LDA, NOUT, CW, CZ,
             LDZ, SRCONE, SRCONV, CWORK, LDWORK, SWORK2, BWORK3,
             INFO)

void dgeesx (char jobvs, char sort, L_fp select, char sense, int
             n, double *da, int lda, int *sdim, double *dwr, double
             *dwi, double *dz, int ldz, double *drcone, double
             *drconv, int *info)
void sgeesx (char jobvs, char sort, L_fp select, char sense, int
             n, float *sa, int lda, int *sdim, float *swr, float
             *swi, float *sz, int ldz, float *srcone, float *srconv,
             int *info)
void zgeesx (char jobvs, char sort, L_fp select, char sense, int
             n, doublecomplex *za, int lda, int *sdim, doublecomplex
             *zw, doublecomplex *zz, int ldz, double *drcone, double
             *drconv, int *info)
```

```
void cgeesx (char jobvs, char sort, L_fp select, char sense, int
n, complex *ca, int lda, int *sdim, complex *cw,
complex *cz, int ldz, float *srcone, float *srconv, int
*info)
```

Arguments

JOBZ	Indicates whether or not the matrix Z of Schur vectors will be computed. The legal values for JOBZ are listed below. Any values not listed below are illegal. 'N' or 'n' Z will not be computed. 'V' or 'v' Z will be computed.
SORTEV	Indicates whether or not to order the eigenvalues on the diagonal of the Schur form. The legal values for SORTEV are listed below. Any values not listed below are illegal. 'N' or 'n' Eigenvalues are not ordered. 'S' or 's' Eigenvalues are ordered on the diagonal as determined by the argument SELECT.
SELECT	If SORTEV = 'S' or 's' then SELECT is used to select eigenvalues to sort to the top left of the Schur form. If SORTEV = 'N' or 'n' then SELECT is not used. SELECT is a function of logical type that must be declared in an EXTERNAL statement in the caller. SELECT may no longer return a value of true for a given eigenvalue after sorting because the process of ordering may change the value of complex eigenvalues, especially if the eigenvalue is ill-conditioned. If this happens then INFO is set to N+2 as described below. Aside from the description given here, the details of SELECT vary for real and complex subroutines; these two cases are described next. For real subroutines, SELECT has two real arguments which are the real and imaginary parts of an eigenvalue. An eigenvalue is selected to be sorted to the top left of the real Schur form if SELECT returns a value of true. If one member of a complex conjugate pair of eigenvalues is selected, then both complex eigenvalues are selected. For complex subroutines, SELECT has a single complex argument. The eigenvalue is selected to be sorted to the top left of the Schur form if SELECT returns a value of true.
SENSE	Indicates which reciprocal condition numbers will be computed. The legal values for SENSE are listed below. Any values not shown below are illegal. 'B' or 'b' Computed for both the eigenvalues and the selected right invariant subspace.

	'E' or 'e'	Computed for the average of selected eigenvalues only.
	'N' or 'n'	None computed.
	'V' or 'v'	Computed for the selected right invariant subspace only.
	If SENSE = 'B', 'b', 'E', 'e', 'V', or 'v' then SORTEV must be 'S' or 's'.	
N	Order of the matrix A. $N \geq 0$.	
xA	On entry, the matrix A. On exit, A has been overwritten by its Schur form T.	
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.	
NOUT	On exit, the number of eigenvalues (after ordering) for which SELECT returns a value of true. For real subroutines, complex conjugate pairs for which SELECT is true for either eigenvalue count as 2. If SORTEV = 'N' or 'n' then NOUT = 0.	
xWR, xWI	(For real subroutines) On exit, the real and imaginary parts, respectively, of the N computed eigenvalues in the same order that they appear on the diagonal of the output Schur form T. Complex conjugate pairs of eigenvalues will appear consecutively with the eigenvalue having the positive imaginary part first.	
xW	(For complex subroutines) On exit, the N computed eigenvalues in the same order that they appear on the diagonal of the output Schur form T. Complex conjugate pairs of eigenvalues will appear consecutively with the eigenvalue having the positive imaginary part first.	
xZ	On exit, if JOBZ = 'V' or 'v' then Z contains the orthogonal or unitary matrix Z of Schur vectors. If JOBZ = 'N' or 'n' then Z is not used.	
LDZ	Leading dimension of the array Z as specified in a dimension or type statement. $LDZ \geq 1$. If JOBZ = 'V' or 'v' then $LDZ \geq \max(1, N)$.	
xWORK	Scratch array with a dimension of LDWORK.	
LDWORK	Leading dimension of the array WORK as specified in a dimension or type statement. Values for LDWORK vary as shown below: $LDWORK \geq \max(1, 3 \times N)$ for real subroutines, $LDWORK \geq \max(1, 2 \times N)$ for complex subroutines. If SENSE = 'E', 'e', 'V', 'v', 'B', or 'b' then LDWORK must also satisfy the following: $LDWORK \geq N + 2 \times NOUT \times (N - NOUT)$ for real subroutines; $LDWORK \geq 2 \times NOUT \times (N - NOUT)$ for complex subroutines.	
xWORK2	Scratch array with a dimension of LDWRK2 for real subroutines or N for complex subroutines.	

LDWRK2	Leading dimension of the array WORK2 as specified in a dimension or type statement. $LDWRK2 \geq 1$. If SENSE = 'B', 'b', 'V', or 'v' then LDWRK2 must also satisfy: $LDWRK2 \geq NOUT \times (N - NOUT)$.
BWORK3	Scratch array of logical type and dimension of N. If SORTEV = 'N' or 'n' then BWORK3 is not used.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where $i = INFO $, had an illegal value. INFO = N+1 The eigenvalues could not be reordered because some eigenvalues were too close to separate. It is likely that the problem is very ill-conditioned. INFO = N+2 After reordering, roundoff changed values of some complex eigenvalues so that SELECT no longer returns a value of true for leading eigenvalues in the Schur form. This could also be caused by underflow due to scaling.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
C   INTEGER          LDA, LDVS, LDIWRK, LDWORK, N
C   PARAMETER        (N = 2)
C   PARAMETER        (LDA = N)
C   PARAMETER        (LDVS = 2 * N)
C   PARAMETER        (LDIWRK = N)
C   PARAMETER        (LDWORK = 3 * N)
C
C   DOUBLE PRECISION A(LDA,N), RCONDE, RCONDV, SCRATCH
C   DOUBLE PRECISION VS(LDVS,N), WORK(LDWORK), WR(N), WI(N)
C   INTEGER          I, ICOL, INFO, IROW, IWORK(LDIWRK), SORT
C
C   EXTERNAL          DGEESX, SELECT
C   INTRINSIC         ABS
C
C   Initialize the array A to store the matrix A shown below.
C
C   A = 0  1
C       -6 5
C
C   DATA A / 0.0D0, -6.0D0, 1.0D0, 5.0D0 /
C
C   Print the initial value of A.
C
C   PRINT 1000
C   PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, N)

```

```

C
C      Compute the eigenvalues, condition numbers, and Schur
C      vectors of A.
C
      CALL DGEESX ('VECTORS AND EIGENVALUES', 'SORTED', SELECT,
$           'BOTH CONDITION NUMBERS', N, A, LDA, SORT, WR,
$           WI, VS, LDVS, RCONDE, RCONDV, WORK, LDWORK,
$           IWORK, LDIWRK, SCRATCH, INFO)
      IF (INFO .NE. 0) THEN
        IF (INFO .LT. 0) THEN
          PRINT 1020, ABS(INFO)
          STOP 1
        ELSEIF (INFO .LE. N) THEN
          PRINT 1030, INFO
          STOP 2
        ELSEIF (INFO .EQ. N + 1) THEN
          PRINT 1040
        ELSE
          PRINT 1050
        END IF
      END IF

C
C      Print the eigenvalues, condition numbers, and Schur vectors.
C
      PRINT 1060, 1.0D0 / RCONDE
      PRINT 1070, 1.0D0 / RCONDV
      PRINT 1080
      DO 120, IROW = 1, N
        PRINT 1090, WR(IROW), (VS(I,IROW), I = 1, N)
      120 CONTINUE
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (2(3X, F4.1))
1020 FORMAT (/1X, 'Illegal argument to DGEES, argument #', I2)
1030 FORMAT (/1X, 'QR failed to converge, INFO = ', I2)
1040 FORMAT (/1X, 'Eigenvalues could not be reordered; problem',
$           /1X, 'is very ill-conditioned.')
1050 FORMAT (/1X, 'Leading eigenvalues in the Schur form no',
$           /1X, 'longer satisfy SELECT = .TRUE.')
1060 FORMAT (/1X, 'Condition number of the eigenvalues: ', F6.1)
1070 FORMAT (1X, 'Condition number of the subspace: ', F6.1)
1080 FORMAT (/1X, 'Eigenvalue', 4X, 'Schur vector**T')
1090 FORMAT (1X, F5.2, 10X, '[', F4.2, ', ', F4.2, ']')
C
      END

      LOGICAL FUNCTION SELECT (ARG1, ARG2)
      IMPLICIT NONE
C
      DOUBLE PRECISION ARG1, ARG2
      INTRINSIC INT

      SELECT = (ARG1 .EQ. ARG2)
C
      END

```

Sample Output

```
A:
  0.0  1.0
 -6.0  5.0

Condition number of the eigenvalues: 1.0
Condition number of the subspace: 0.1

Eigenvalue  Schur vector**T
2.00        [0.45, 0.89]
3.00        [-.89, 0.45]
```


Eigenvalues and Eigenvectors of a General Matrix (Simple Driver)

The subroutines described in this section compute the eigenvalues of a square general matrix A and, optionally, the normalized left and / or right eigenvectors of A . The left eigenvectors of A are the same as the right eigenvectors of A^T or A^H . The eigenvectors are normalized to have a Euclidian norm of 1 and a largest component that is real. Note that the expert driver xGEEVX is also available.

Calling Sequence

```
CALL DGEEV (JOBVL, JOBVR, N, DA, LDA, DWR, DWI, DVL, LDVL, DVR,
            LDVR, DWORK, LDWORK, INFO)
CALL SGEEV (JOBVL, JOBVR, N, SA, LDA, SWR, SWI, SVL, LDVL, SVR,
            LDVR, SWORK, LDWORK, INFO)
CALL ZGEEV (JOBVL, JOBVR, N, ZA, LDA, ZW, ZVL, LDVL, ZVR, LDVR,
            ZWORK, LDWORK, DWORK2, INFO)
CALL CGEEV (JOBVL, JOBVR, N, CA, LDA, CW, CVL, LDVL, CVR, LDVR,
            CWORK, LDWORK, SWORK2, INFO)

void dgeev (char jobvl, char jobvr, int n, double *da, int lda,
            double *dwr, double *dwi, double *dvl, int ldvl, double
            *dvr, int ldvr, int *info)
void sgeev (char jobvl, char jobvr, int n, float *sa, int lda,
            float *swr, float *swi, float *svl, int ldvl, float
            *svr, int ldvr, int *info)
void zgeev (char jobvl, char jobvr, int n, doublecomplex *za, int
            lda, doublecomplex *zw, doublecomplex *vl, int ldvl,
            doublecomplex *zvr, int ldvr, int *info)
void cgeev (char jobvl, char jobvr, int n, complex *ca, int lda,
            complex *cw, complex *vl, int ldvl, complex *cvr, int
            ldvr, int *info)
```

Arguments

JOBVL	Indicates whether or not the left eigenvectors will be computed. The legal values for JOBVL are listed below. Any values not listed below are illegal.
'N' or 'n'	Left eigenvectors will not be computed.
'V' or 'v'	Left eigenvectors will be computed.

JOBVR	Indicates whether or not the right eigenvectors will be computed. The legal values for JOBVR are listed below. Any values not listed below are illegal. 'N' or 'n' Right eigenvectors will not be computed. 'V' or 'v' Right eigenvectors will be computed.
N	Order of the matrix A. $N \geq 0$.
xA	On entry, the matrix A. On exit, A has been overwritten.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
xWR, xWI	(For real subroutines) On exit, WR and WI contain the real and imaginary parts, respectively, of the computed eigenvalues. Complex conjugate pairs of eigenvalues appear consecutively with the eigenvalue having the positive imaginary part first.
xW	(For complex subroutines) Computed eigenvalues of A.
xVL	On exit, if JOBVL = 'V' or 'v' then the left eigenvectors u(j) are stored one after another in the columns of VL, in the same order as their eigenvalues. If the jth eigenvalue is real, then $u(j) = VL(*, j)$. If the jth and (j+1)th eigenvalues form a complex conjugate pair, then $u(j) = VL(*, j) + i \times VL(*, j+1)$ and $u(j+1) = VL(*, j) + i \times VL(*, j+1)$. If JOBVL = 'N' or 'n' then VL is not used.
LDVL	Leading dimension of the array VL as specified in a dimension or type statement. $LDVL \geq 1$. If JOBVL = 'V' or 'v' then $LDVL \geq \max(1, N)$.
xVR	On exit, if JOBVR = 'V' or 'v' then the right eigenvectors v(j) are stored one after another in the columns of VR, in the same order as their eigenvalues. If the jth eigenvalue is real, then $v(j) = VR(*, j)$. If the jth and (j+1)th eigenvalues form a complex conjugate pair, then $v(j) = VR(*, j) + i \times VR(*, j+1)$ and $v(j+1) = VR(*, j) + i \times VR(*, j+1)$. If JOBVR = 'N' or 'n' then VR is not used.
LDVR	Leading dimension of the array VR as specified in a dimension or type statement. $LDVR \geq 1$. If JOBVR = 'V' or 'v' then $LDVR \geq \max(1, N)$.
xWORK	Scratch array with a dimension of LDWORK.
LDWORK	Leading dimension of the array WORK as specified in a dimension or type statement. For real subroutines: $LDWORK \geq \max(1, 3 \times N)$. If JOBVL or JOBVR = 'V' or 'v' then $LDWORK \geq \max(1, 4 \times N)$. For complex subroutines: $LDWORK \geq \max(1, 2 \times N)$.
xWORK2	Scratch array with a dimension of $2 \times N$ for complex subroutines.

INFO	On exit:	
INFO = 0		Subroutine completed normally.
INFO < 0		The <i>i</i> th argument, where <i>i</i> = INFO , had an illegal value.
INFO > 0		The QR algorithm failed to compute all the eigenvalues, and no eigenvectors have been computed. Elements <i>i</i> +1 through <i>N</i> of <i>WR</i> and <i>WI</i> (or <i>W</i>), where <i>i</i> = INFO, contain those eigenvalues that have converged.

Sample Program

```
PROGRAM TEST
IMPLICIT NONE
C
C   INTEGER          LDA, LDVR, LDWORK, N
C   PARAMETER        (N = 2)
C   PARAMETER        (LDA = N)
C   PARAMETER        (LDVR = 2 * N)
C   PARAMETER        (LDWORK = 4 * N)
C
C   DOUBLE PRECISION A(LDA,N), SCRATCH, VR(LDVR,N)
C   DOUBLE PRECISION WORK(LDWORK), WR(N), WI(N)
C   INTEGER          I, ICOL, INFO, IROW
C
C   EXTERNAL          DGEEV
C   INTRINSIC         ABS
C
C   Initialize the array A to store the matrix A shown below.
C
C   A = 0   1
C       -6   5
C
C   DATA A / 0.0D0, -6.0D0, 1.0D0, 5.0D0 /
C
C   Print the initial value of A.
C
C   PRINT 1000
C   PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
C
C   Compute the eigenvalues and right eigenvectors of A.
C
C   CALL DGEEV ('NO LEFT EIGENVECTORS', 'VRIGHT', N, A, LDA, WR,
C   $           WI, SCRATCH, 1, VR, LDVR, WORK, LDWORK, INFO)
C   IF (INFO .NE. 0) THEN
C     IF (INFO .LT. 0) THEN
C       PRINT 1020, ABS(INFO)
C       STOP 1
C     ELSE
C       PRINT 1030, INFO
C       STOP 2
C     END IF
C   END IF
```

```

      END IF
C
C   Print the eigenvalues and eigenvectors.
C
      PRINT 1040
      DO 120, IROW = 1, N
        PRINT 1050, WR(IROW), (VR(I,IROW), I = 1, N)
      120 CONTINUE
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (2(3X, F4.1))
1020 FORMAT (/1X, 'Illegal argument to DGEEV, argument = ', I2)
1030 FORMAT (/1X, 'QR failed to converge, INFO = ', I2)
1040 FORMAT (/1X, 'Eigenvalue', 4X, 'Eigenvector**T')
1050 FORMAT (1X, F5.2, 10X, '[', F4.2, ', ', F4.2, ']')
C
      END

```

Sample Output

```

A:
    0.0    1.0
   -6.0    5.0

Eigenvalue  Eigenvector**T
 2.00      [0.45, 0.89]
 3.00      [0.32, 0.95]

```

Eigenvalues, Eigenvectors, and Reciprocal Condition Numbers of a General Matrix (Expert Driver)

The subroutines described in this section compute the eigenvalues of a general matrix A . They can also compute normalized left and/or right eigenvectors, reciprocal condition numbers for the eigenvalues, and reciprocal condition numbers for the right eigenvectors. The left eigenvectors of A are the same as the right eigenvectors of A^T or A^H . The eigenvectors are normalized to have a Euclidean norm of 1 and a largest component that is real.

The computations of eigenvalues, eigenvectors, and condition numbers may be done with an optional balancing transformation which often improves the condition and accuracy. Note that the simple driver `xGEEV` is also available.

Calling Sequence

```
CALL DGEEVX (BALANC, JOBV, JOBVR, SENSE, N, DA, LDA, DWR, DWI,
             DVL, LDVL, DVR, LDVR, ILO, IHI, DSCALE, DABNRM, DRCONV,
             DWORK, LDWORK, IWORK2, INFO)
CALL SGEEVX (BALANC, JOBV, JOBVR, SENSE, N, SA, LDA, SWR, SWI,
             SVL, LDVL, SVR, LDVR, ILO, IHI, SSCALE, SABNRM, SRCONV,
             SWORK, LDWORK, IWORK2, INFO)
CALL ZGEEVX (BALANC, JOBV, JOBVR, SENSE, N, ZA, LDA, ZW, ZVL,
             LDVL, ZVR, LDVR, ILO, IHI, DSCALE, DABNRM, DRCONV,
             ZWORK, LDWORK, DWORK2, INFO)
CALL CGEEVX (BALANC, JOBV, JOBVR, SENSE, N, CA, LDA, CW, CVL,
             LDVL, CVR, LDVR, ILO, IHI, SSCALE, SABNRM, SRCONV,
             CWORK, LDWORK, SWORK2, INFO)

void dgeevx (char balanc, char jobvl, char jobvr, char sense, int
             n, double *da, int lda, double *dwr, double *dwi,
             double *dvl, int ldvl, double *dvr, int ldvr, int *ilo,
             int *ihi, double *dscale, double *dabnrm, double
             *drcone, double *drconv, int *info)
void sgeevx (char balanc, char jobvl, char jobvr, char sense, int
             n, float *sa, int lda, float *swr, float *swi, float
             *svl, int ldvl, float *svr, int ldvr, int *ilo, int
             *ihi, float *sscale, float *sabnrm, float *srcone,
             float *srconv, int *info)
```

```
void zggevz (char balanc, char jobvl, char jobvr, char sense, int
n, doublecomplex *za, int lda, doublecomplex *zw,
doublecomplex *vl, int ldvl, doublecomplex *zvr, int
ldvr, int *ilo, int *ihi, double *dscale, double
*dabnrm, double *drcone, double *drconv, int *info)
void cgeevz (char balanc, char jobvl, char jobvr, char sense, int
n, complex *ca, int lda, complex *cw, complex *vl, int
ldvl, complex *cvr, int ldvr, int *ilo, int *ihi, float
*sscale, float *sabnrm, float *srcone, float *srconv,
int *info)
```

Arguments

BALANC	Indicates how the input matrix will be diagonally scaled and/or permuted to try to improve the conditioning of its eigenvalues. The legal values for BALANC are listed below. Any values not listed below are illegal. 'N' or 'n' Do not diagonally scale or permute A. 'P' or 'p' Permute A to make it more nearly upper triangular, but do not diagonally scale A. 'S' or 's' Diagonally scale A, i.e. replace A by DAD⁻¹ for a diagonal matrix D chosen to make the rows and columns of A more equal in norm. 'B' or 'b' Both diagonally scale and permute A. Computed reciprocal condition numbers will be for the matrix after balancing and/or permuting. Permuting does not change the condition number (in exact arithmetic), but balancing does.
JOBVL	Indicates whether or not the left eigenvectors will be computed. The legal values for JOBVL are listed below. Any values not listed below are illegal. 'N' or 'n' Left eigenvectors will not be computed. 'V' or 'v' Left eigenvectors will be computed. If SENSE = 'E', 'e', 'B', or 'b' then JOBVL must be 'V' or 'v'.
JOBVR	Indicates whether or not the right eigenvectors will be computed. The legal values for JOBVR are listed below. Any values not listed below are illegal. 'N' or 'n' Right eigenvectors will not be computed. 'V' or 'v' Right eigenvectors will be computed. If SENSE = 'E', 'e', 'B', or 'b' then JOBVR must be 'V' or 'v'.
SENSE	Indicates which reciprocal condition numbers will be computed, if any. The legal values for SENSE are shown below. Any values not listed below are illegal. 'N' or 'n' None computed.

	'E' or 'e'	Computed for eigenvalues only.
	'V' or 'v'	Computed for right eigenvectors only.
	'B' or 'b'	Computed for eigenvalues and right eigenvectors.
	If SENSE = 'E', 'e', 'B', or 'b' then both left and right eigenvectors must be computed, so both JOBVL and JOBVR must be set to 'V' or 'v'.	
N	Order of the matrix A. $N \geq 0$.	
xA	On entry, the matrix A. On exit, xA has been overwritten. If either JOBVL or JOBVR contain 'V' or 'v' then xA contains the Schur form of the balanced A.	
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.	
xWR, xWI	(For real subroutines) On exit, WR and WI contain the real and imaginary parts, respectively, of the N computed eigenvalues. Complex conjugate pairs of eigenvalues will appear consecutively with the eigenvalue having the positive imaginary part first.	
xW	(For complex subroutines) Computed eigenvalues of A.	
xVL	On exit, if JOBVL = 'V' or 'v' then the left eigenvectors u(j) are stored one after another in the columns of VL, in the same order as their eigenvalues. If the jth eigenvalue is real, then $u(j) = VL(*, j)$. If the jth and (j+1)th eigenvalues form a complex conjugate pair, then $u(j) = VL(*, j) + i \times VL(*, j+1)$ and $u(j+1) = VL(*, j) + i \times VL(*, j+1)$. If JOBVL = 'N' or 'n' then VL is not used.	
LDVL	Leading dimension of the array VL as specified in a dimension or type statement. $LDVL \geq 1$. If JOBVL = 'V' or 'v' then $LDVL \geq \max(1, N)$.	
xVR	On exit, if JOBVR = 'V' or 'v' then the right eigenvectors v(j) are stored one after another in the columns of VR, in the same order as their eigenvalues. If the jth eigenvalue is real, then $v(j) = VR(*, j)$. If the jth and (j+1)th eigenvalues form a complex conjugate pair, then $v(j) = VR(*, j) + i \times VR(*, j+1)$ and $v(j+1) = VR(*, j) + i \times VR(*, j+1)$. If JOBVR = 'N' or 'n' then VR is not used.	
LDVR	Leading dimension of the array VR as specified in a dimension or type statement. $LDVR \geq 1$. If JOBVR = 'V' or 'v' then $LDVR \geq \max(1, N)$.	
ILO, IHI	On exit, ILO and IHI contain integer values determined when A was balanced. The balanced $A(i,j) = 0$ for $i > j$, $1 \leq j \leq ILO-1$, and $IHI+1 \leq i \leq N$.	
xSCALE	On exit, details of the permutations and scaling factors applied when balancing A. If P(j) is the index of the row and column interchanged with row and column j, and D(j) is the scaling factor applied to row and column j, then	

	SCALE(j) = P(j) for $1 \leq j \leq \text{ILO}-1$
	SCALE(j) = D(j) for $\text{ILO} \leq j \leq \text{IHI}$
	SCALE(j) = P(j) for $\text{IHI}+1 \leq j \leq N$
	The order in which the interchanges are made is N through IHI+1, then 1 through ILO-1.
xABNRM	On exit, the 1-norm of the balanced matrix.
xRCONE	On exit, if SENSE = 'E', 'e', 'B', or 'b', then RCONE(j) for $1 \leq j \leq N$ is the reciprocal condition number of the jth eigenvalue. Otherwise RCONE is not used.
xRCONV	On exit, if SENSE = 'E', 'e', 'B', or 'b', then RCONV(j) for $1 \leq j \leq N$ is the reciprocal condition number of the jth right eigenvector. Otherwise RCONV is not used.
xWORK	Scratch array with a dimension of LDWORK.
LDWORK	Leading dimension of the array WORK as specified in a dimension or type statement. If SENSE = 'N', 'n', 'E', or 'e', then $\text{LDWORK} \geq \max(1, 2 \times N)$. Otherwise: For real subroutines: If JOBVL = 'V' or 'v', then $\text{LDWORK} \geq \max(1, 3 \times N)$. If SENSE = 'V', 'v', 'B', or 'b' then $\text{LDWORK} \geq N \times (N + 6)$. For complex subroutines: If SENSE = 'V', 'v', 'B', or 'b' then $\text{LDWORK} \geq N \times N + 2 \times N$.
xWORK2	Scratch array with the following dimension: For real subroutines: $2 \times N - 2$. If SENSE = 'N', 'n', 'E', or 'e' then WORK2 is not used. For complex subroutines: $2 \times N$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where $i = \text{INFO} $, had an illegal value. INFO > 0 The QR algorithm failed to compute all the eigenvalues, and no eigenvectors or condition numbers have been computed. Elements 1 through ILO-1 and i+1 through N of WR and WI (or of W), where $i = \text{INFO}$ contain eigenvalues that have converged.

Sample Program

	PROGRAM TEST
	IMPLICIT NONE
C	INTEGER LDA, LDIWRK, LDVR, LDWORK, N


```

PARAMETER      (N = 2)
PARAMETER      (LDA = N)
PARAMETER      (LDIWRK = 2 * N - 2)
PARAMETER      (LDVR = 3 * N)
PARAMETER      (LDWORK = 3 * N)
C
DOUBLE PRECISION A(LDA,N), BALNRM, SCALE(N), TEMP
DOUBLE PRECISION VR(LDVR,N), WORK(LDWORK), WR(N), WI(N)
INTEGER         I, ICOL, IHI, ILO, INFO, IROW
INTEGER         IWORK(LDIWRK)
C
EXTERNAL        DGEEVX
INTRINSIC       ABS
C
C      Initialize the array A to store the matrix A shown below.
C
C      A = 0  1
C          -6 5
C
DATA A / 0.0D0, -6.0D0, 1.0D0, 5.0D0 /
C
C      Print the initial value of A.
C
PRINT 1000
PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
C
C      Compute the eigenvalues and right eigenvectors of A.
C
CALL DGEEVX ('BOTH PERMUTE AND SCALE A',
$           'NO LEFT EIGENVECTORS', 'VRIGHT',
$           'NO CONDITION NUMBERS', N, A, LDA, WR, WI,
$           TEMP, 1, VR, LDVR, ILO, IHI, SCALE, BALNRM,
$           TEMP, TEMP, WORK, LDWORK, IWORK, INFO)
IF (INFO .NE. 0) THEN
  IF (INFO .LT. 0) THEN
    PRINT 1020, ABS(INFO)
    STOP 1
  ELSE
    PRINT 1030, INFO
    STOP 2
  END IF
END IF
C
C      Print the eigenvalues and eigenvectors.
C
PRINT 1040
DO 120, IROW = 1, N
  PRINT 1050, WR(IROW), (VR(I,IROW), I = 1, N)
120 CONTINUE
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (2(3X, F4.1))
1020 FORMAT (/1X, 'Illegal argument to DGEEVX, argument = ', I2)
1030 FORMAT (/1X, 'QR failed to converge, INFO = ', I2)
1040 FORMAT (/1X, 'Eigenvalue', 4X, 'Eigenvector**T')
1050 FORMAT (1X, F5.2, 10X, '[' , F4.2, ' ', ' ', F4.2, ']' )

```

```
C
      END
```

Sample Output

```
A:
    0.0    1.0
   -6.0    5.0

Eigenvalue  Eigenvector**T
2.00        [0.45, 0.89]
3.00        [0.32, 0.95]
```

Generalized Eigenvalues, Schur Form, and Schur Vectors of General Matrices

The subroutines in this section compute the generalized eigenvalues (α , β) for a pair of square general matrices A and B, and compute the Schur form $(A, B) = (V_L^T A V_R, V_L^T B V_R)$ for real matrices or the Schur form $(A, B) = (V_L^H A V_R, V_L^H B V_R)$ for complex matrices where V_L and V_R are the left and right Schur vectors. These subroutines can optionally return the left and/or right Schur vectors. Note that the driver xGEGV is also available.

A generalized eigenvalue is a ratio $\lambda = \alpha / \beta$ such that $AX = \lambda BX$ or $\det(A - \lambda B) = 0$ for general matrices A, B, and X. λ is typically represented as a ratio rather than as a scalar because there are reasonable interpretations for $\alpha = 0$, $\beta = 0$, and for $\alpha = \beta = 0$. A right generalized eigenvector x corresponding to a generalized eigenvalue λ is defined by $(A - \lambda B)x = 0$. A left generalized eigenvector x corresponding to a generalized eigenvalue λ is defined by $(A - \lambda B)^H x = 0$. A good reference for generalized eigenproblems is the book *Matrix Computations*, 2nd. ed. by Golub and van Loan (1989, The Johns Hopkins University Press).

Calling Sequence

```
CALL DGEYS (JOBVSL, JOBVSR, N, DA, LDA, DB, LDB, DALPHAR, DALPHAI,
            DBETA, DVSL, LDVSL, DVSR, LDVSR, DWORK, LDWORK, INFO)
CALL SGEYS (JOBVSL, JOBVSR, N, SA, LDA, SB, LDB, SALPHAR, SALPHAI,
            SBETA, SVSL, LDVSL, SVSR, LDVSR, SWORK, LDWORK, INFO)
CALL ZGEYS (JOBVSL, JOBVSR, N, ZA, LDA, ZB, LDB, ZALPHA, ZBETA,
            ZVSL, LDVSL, ZVSR, LDVSR, ZWORK, LDWORK, DWORK2, INFO)
CALL CGEYS (JOBVSL, JOBVSR, N, CA, LDA, CB, LDB, CALPHA, CBETA,
            CVSL, LDVSL, CVSR, LDVSR, CWORK, LDWORK, SWORK2, INFO)

void dgeys (char jobvsl, char jobvsr, int n, double *da, int lda,
            double *db, int ldb, double *dalphar, double *dalphai,
            double *dbeta, double *dvsl, int ldvsl, double *dvsl,
            int ldvsr, int *info)
void sgeys (char jobvsl, char jobvsr, int n, float *sa, int lda,
            float *sb, int ldb, float *salphar, float *dalphai,
            float *sbeta, float *svsl, int ldvsl, float *svsl, int
            ldvsr, int *info)
void zgeys (char jobvsl, char jobvsr, int n, doublecomplex *za,
            int lda, doublecomplex *zb, int ldb, doublecomplex
            *zalpha, doublecomplex *zbeta, doublecomplex *zvsl,
            int ldvsl, doublecomplex *zvsl, int ldvsr, int *info)
```

```
void cgegs (char jobvsl, char jobvsr, int n, complex *ca, int lda,
            complex *cb, int ldb, complex *calpha, complex *cbeta,
            complex *cvsl, int ldvsl, complex *cvsr, int ldvsr, int
            *info)
```

Arguments

JOBVSL	Indicates whether or not to compute the left Schur vectors. The legal values of JOBVSL are shown below. Any values not shown below are illegal. 'N' or 'n' Do not compute the left Schur vectors. 'V' or 'v' Compute the left Schur vectors.
JOBVSR	Indicates whether or not to compute the right Schur vectors. The legal values of JOBVSR are shown below. Any values not shown below are illegal. 'N' or 'n' Do not compute the right Schur vectors. 'V' or 'v' Compute the right Schur vectors.
N	Number of rows and columns in the matrices A, B, VSL, and VSR. $N \geq 0$.
xA	On entry, the first of the pair of matrices whose generalized eigenvalues (and Schur vectors) are to be computed. On exit, the generalized Schur form of A. Note: to avoid overflow, the Frobenius norm of A should be less than the overflow threshold.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
xB	On entry, the second of the pair of matrices whose generalized eigenvalues to be computed. On exit, the generalized Schur form of B. Note: to avoid overflow, the Frobenius norm of B should be less than the overflow threshold.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
xALPHAR, xALPHAI, xBETA	(For real subroutines) On exit, $(ALPHAR(j) + i \times ALPHAI(j)) / BETA(j)$ for $1 \leq j \leq N$ contain the generalized eigenvalues. $ALPHAR(j) + i \times ALPHAI(j)$ and $BETA(j)$ for $1 \leq j \leq N$ are the diagonals of the complex Schur form (A,B) that would result if the 2x2 diagonal blocks of the real Schur form of (A,B) were further reduced to triangular form using 2x2 complex unitary transformations. If $ALPHAI(j) = 0$, then the j th eigenvalue is real; if positive, then the j th and $(j+1)$ th eigenvalues are a complex conjugate pair, with $ALPHAI(j+1)$ negative.

Note: The quotients $\text{ALPHAR}(j) / \text{BETA}(j)$ and $\text{ALPHA}(j) / \text{BETA}(j)$ may easily over- or underflow, and any or all of $\text{ALPHAR}(j)$, $\text{ALPHA}(j)$, $\text{BETA}(j)$ may be zero. Thus, the user should avoid naively computing the ratio α / β . However, ALPHAR and ALPHA will always be less than and usually comparable with $\|A\|$ in magnitude, and BETA always less than and usually comparable with $\|B\|$.

xALPHA, xBETA (For complex subroutines)

On exit, $\text{ALPHA}(j) / \text{BETA}(j)$ for $1 \leq j \leq N$ are the generalized eigenvalues. $\text{ALPHA}(j)$ and $\text{BETA}(j)$ for $1 \leq j \leq N$ are the diagonals of the complex Schur form (A,B) computed by xGEGS. $\text{BETA}(j)$ is non-negative and real.

Note: The quotients $\text{ALPHA}(j) / \text{BETA}(j)$ may easily over- or underflow, and either or both $\text{ALPHA}(j)$ or $\text{BETA}(j)$ may be zero. Thus, the user should avoid naively computing the ratio α / β . However, ALPHA will always be less than and usually comparable with $\|A\|$ in magnitude, and BETA always less than and usually comparable with $\|B\|$.

xVSL	On exit, if $\text{JOBVSL} = 'V'$ or $'v'$ then VSL will contain the left Schur vectors. If $\text{JOBVSL} = 'N'$ or $'n'$ then VSL is not used.								
LDVSL	Leading dimension of the array VSL as specified in a dimension or type statement. $\text{LDVSL} \geq 1$. If $\text{JOBVSL} = 'V'$ or $'v'$ then $\text{LDVSL} \geq \max(1, N)$.								
xVSR	On exit, if $\text{JOBVSR} = 'V'$ or $'v'$ then VSR will contain the right Schur vectors. If $\text{JOBVSR} = 'N'$ or $'n'$ then VSR is not used.								
LDVSR	Leading dimension of the array VSR as specified in a dimension or type statement. $\text{LDVSR} \geq 1$. If $\text{JOBVSR} = 'V'$ or $'v'$, $\text{LDVSR} \geq \max(1, N)$.								
xWORK	Scratch array with a dimension of LDWORK.								
LDWORK	Leading dimension of the array WORK as specified in a dimension or type statement. $\text{LDWORK} \geq \max(1, 4 \times N)$ for real subroutines or $\max(1, 2 \times N)$ for complex subroutines.								
xWORK2	Scratch array with a dimension of $3 \times N$ for complex subroutines.								
INFO	On exit: <table> <tr> <td>$\text{INFO} = 0$</td><td>Subroutine completed normally.</td></tr> <tr> <td>$\text{INFO} < 0$</td><td>The ith argument, where $i = \text{INFO}$, had an illegal value.</td></tr> <tr> <td>$1 \leq \text{INFO} \leq N$</td><td>The QZ iteration failed. (A,B) are not in Schur form, but $\text{ALPHAR}(j)$, $\text{ALPHA}(j)$, and $\text{BETA}(j)$ — or $\text{ALPHA}(j)$ and $\text{BETA}(j)$ — will be correct for $\text{INFO}+1 \leq j \leq N$.</td></tr> <tr> <td>$\text{INFO} > N$</td><td>May indicate an internal LAPACK error.</td></tr> </table>	$\text{INFO} = 0$	Subroutine completed normally.	$\text{INFO} < 0$	The i th argument, where $i = \text{INFO} $, had an illegal value.	$1 \leq \text{INFO} \leq N$	The QZ iteration failed. (A,B) are not in Schur form, but $\text{ALPHAR}(j)$, $\text{ALPHA}(j)$, and $\text{BETA}(j)$ — or $\text{ALPHA}(j)$ and $\text{BETA}(j)$ — will be correct for $\text{INFO}+1 \leq j \leq N$.	$\text{INFO} > N$	May indicate an internal LAPACK error.
$\text{INFO} = 0$	Subroutine completed normally.								
$\text{INFO} < 0$	The i th argument, where $i = \text{INFO} $, had an illegal value.								
$1 \leq \text{INFO} \leq N$	The QZ iteration failed. (A,B) are not in Schur form, but $\text{ALPHAR}(j)$, $\text{ALPHA}(j)$, and $\text{BETA}(j)$ — or $\text{ALPHA}(j)$ and $\text{BETA}(j)$ — will be correct for $\text{INFO}+1 \leq j \leq N$.								
$\text{INFO} > N$	May indicate an internal LAPACK error.								

Sample Program

```
C      PROGRAM TEST
        IMPLICIT NONE

C      INTEGER          LDA, LDB, LDWORK, M, N
        PARAMETER      (M = 3)
        PARAMETER      (N = 3)
        PARAMETER      (LDA = M)
        PARAMETER      (LDB = M)
        PARAMETER      (LDWORK = 4 * N)

C      DOUBLE PRECISION A(LDA,N), ALPHAI(N), ALPHAR(N), B(LDB,N)
        DOUBLE PRECISION BETA(N), TEMP, WORK(LDWORK)
        INTEGER         ICOL, INFO, IROW

C      EXTERNAL          DGEKS
        INTRINSIC       ABS, DCMLPX

C      Initialize the arrays A and B to store the matrices A and B
C      shown below.
C
C      A =  $\begin{pmatrix} 1 & 1 & 1 \\ 2 & 1 & 1 \\ 1 & 3 & 1 \end{pmatrix}$    B =  $\begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$ 
C
C      DATA A / 1.0D0, 2.0D0, 1.0D0,
$           1.0D0, 1.0D0, 3.0D0,
$           1.0D0, 1.0D0, 1.0D0 /
        DATA B / 1.0D0, 1.0D0, 0.0D0,
$           1.0D0, 0.0D0, 1.0D0,
$           1.0D0, 0.0D0, 0.0D0 /

C      PRINT 1000
        PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
        PRINT 1020
        PRINT 1010, ((B(IROW,ICOL), ICOL = 1, N), IROW = 1, N)

C      CALL DGEKS ('NO LEFT SCHUR VECTORS',
$              'NO RIGHT SCHUR VECTORS', N, A, LDA, B, LDB,
$              ALPHAR, ALPHAI, BETA, TEMP, 1, TEMP, 1,
$              WORK, LDWORK, INFO)
        IF (INFO .NE. 0) THEN
            IF (INFO .LT. 0) THEN
                PRINT 1030, ABS(INFO)
                STOP 1
            ELSE
                PRINT 1040, INFO
                STOP 2
            END IF
        END IF
        PRINT 1050
        PRINT 1060, (ALPHAR(IROW), ALPHAI(IROW), BETA(IROW),
$      IROW = 1, M)
```

```

      PRINT 1070
      PRINT 1080, (DCMPLX(ALPHAR(IROW), ALPHAI(IROW)) /
$      BETA(IROW), IROW = 1, M)
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (3(3X, F4.1))
1020 FORMAT (/1X, 'B:')
1030 FORMAT (1X, 'Illegal argument to DGEES, argument #', I2)
1040 FORMAT (1X, 'Internal failure in DGEES, INFO = ', I4)
1050 FORMAT (/1X, T4, 'ALPHA', T20, 'BETA')
1060 FORMAT (1X, '(', F5.2, ', ', F5.2, ')', T19, F6.2)
1070 FORMAT (/1X, 'ALPHA / BETA')
1080 FORMAT (1X, '(', F5.2, ', ', F5.2, ')')
C
      END

```

Sample Output

```

A:
  1.0    1.0    1.0
  2.0    1.0    1.0
  1.0    3.0    1.0

B:
  1.0    1.0    1.0
  1.0    0.0    0.0
  0.0    1.0    0.0

ALPHA      BETA
( 0.71, 0.00) 0.71
( 0.82, 0.00) 0.82
( 3.46, 0.00) 1.73

ALPHA / BETA
( 1.00, 0.00)
( 1.00, 0.00)
( 2.00, 0.00)

```

Generalized Eigenvalues and Eigenvectors for General Matrices

The subroutines in this section compute the generalized eigenvalues (α , β) for a pair of square general matrices A and B. Optionally, these subroutines can also compute the generalized eigenvectors. Note that the driver xGEGS is also available.

A generalized eigenvalue is a ratio $\lambda = \alpha / \beta$ such that $AX = \lambda BX$ or $\det(A - \lambda B) = 0$ for general matrices A, B, and X. λ is typically represented as a ratio rather than as a scalar because there are reasonable interpretations for $\alpha = 0$, $\beta = 0$, and for $\alpha = \beta = 0$. A right generalized eigenvector x corresponding to a generalized eigenvalue λ is defined by $(A - \lambda B)x = 0$. A left generalized eigenvector x corresponding to a generalized eigenvalue λ is defined by $(A - \lambda B)Hx = 0$. A good reference for generalized eigenproblems is the book *Matrix Computations*, 2nd. ed. by Golub and van Loan (1989, The Johns Hopkins University Press).

Calling Sequence

CALL DGEGV	(JOBVL, JOBVR, N, DA, LDA, DB, LDB, DALPHAR, DALPHAI, DBETA, DVL, LDVL, DVR, LDVR, DWORK, LDWORK, INFO)
CALL SGEGV	(JOBVL, JOBVR, N, SA, LDA, SB, LDB, SALPHAR, SALPHAI, SBETA, SVL, LDVL, SVR, LDVR, SWORK, LDWORK, INFO)
CALL ZGEGV	(JOBVL, JOBVR, N, ZA, LDA, ZB, LDB, ZALPHA, ZBETA, ZVL, LDVL, ZVR, LDVR, ZWORK, LDWORK, DWORK2, INFO)
CALL CGEGV	(JOBVL, JOBVR, N, CA, LDA, CB, LDB, CALPHA, CBETA, CVL, LDVL, CVR, LDVR, CWORK, LDWORK, SWORK2, INFO)
void dgegv	(char jobvl, char jobvr, int n, double *da, int lda, double *db, int ldb, double *dalphar, double *dalphai, double *dbeta, double *dvl, int ldvl, double *dvr, int ldvr, int *info)
void sgegv	(char jobvl, char jobvr, int n, float *sa, int lda, float *sb, int ldb, float *salphar, float *dalphai, float *sbeta, float *svl, int ldvl, float *svr, int ldvr, int *info)
void zgegv	(char jobvl, char jobvr, int n, doublecomplex *za, int lda, doublecomplex *zb, int ldb, doublecomplex *zalpha, doublecomplex *zbeta, doublecomplex *zvl, int ldvl, doublecomplex *zvr, int ldvr, int *info)


```
void cgegv (char jobvl, char jobvr, int n, complex *ca, int lda,
            complex *cb, int ldb, complex *calpha, complex *cbeta,
            complex *cvl, int ldvl, complex *cvr, int ldvr, int
            *info)
```

Arguments

JOBVL	Indicates whether the left generalized eigenvectors will be computed. Legal values for JOBVL are shown below. Any values not shown below are illegal. 'N' or 'n' Do not compute the left generalized eigenvectors. 'V' or 'v' Compute the left generalized eigenvectors.
JOBVR	Indicates whether the right generalized eigenvectors will be computed. Legal values for JOBVR are shown below. Any values not shown below are illegal. 'N' or 'n' Do not compute the right generalized eigenvectors. 'V' or 'v' Compute the right generalized eigenvectors.
N	Number of rows and columns in the matrices A, B, VL, and VR. $N \geq 0$.
xA	On entry, the first of the pair of matrices whose generalized eigenvalues are to be computed. On exit, if JOBVL or JOBVR = 'V' or 'v', then A contains the real Schur form of the balanced form of A. Otherwise A is overwritten.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
xB	On entry, the second of the pair of matrices whose generalized eigenvalues are to be computed. On exit, if JOBVL or JOBVR = 'V' or 'v', then B contains the real Schur form of the balanced form of B. Otherwise B is overwritten.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
xALPHAR, xALPHAI, xBETA (For real subroutines)	On exit, $(ALPHAR(j) + i \times ALPHAI(j)) / BETA(j)$ for $1 \leq j \leq N$ will contain the generalized eigenvalues. If $ALPHAI(j) = 0$, then the j th eigenvalue is real; if positive, then the j th and $(j+1)$ th eigenvalues are a complex conjugate pair, with $ALPHAI(j+1)$ negative. Note: The quotients $ALPHAR(j) / BETA(j)$ and $ALPHAI(j) / BETA(j)$ may easily over- or underflow, and any or all of $ALPHAR(j)$, $ALPHAI(j)$, $BETA(j)$ may be zero. Thus, the user should avoid naively computing the ratio α / β . However, $ALPHAR$ and $ALPHAI$ will be always less than and usually comparable with $\ A\ $ in magnitude, and $BETA$ always less than and usually comparable with $\ B\ $.

xALPHA, xBETA (For complex subroutines)

On exit, ALPHA(j) / BETA(j) for $1 \leq j \leq N$ are the generalized eigenvalues. ALPHA(j) and BETA(j) for $1 \leq j \leq N$ are the diagonals of the complex Schur form (A,B) as computed by xGEGS. BETA(j) is non-negative and real.

Note: The quotients ALPHA(j) / BETA(j) may easily over- or underflow, and either or both ALPHA(j) or BETA(j) may be zero. Thus, the user should avoid naively computing the ratio α / β . However, ALPHA will always be less than and usually comparable with ||A|| in magnitude, and BETA always less than and usually comparable with ||B||.

xVL On exit, if JOBVL = 'V' or 'v' then VL contains the left generalized eigenvectors. VL is not used if JOBVL = 'N' or 'n'. Real eigenvectors occupy one column. Complex eigenvectors occupy two columns, the first for the real part and the second for the imaginary part. Complex eigenvectors correspond to an eigenvalue with positive imaginary part. With the exception of those eigenvalues with $\alpha = \beta = 0$, each eigenvector will be scaled so the largest component will have $\text{abs}(\text{real part}) + \text{abs}(\text{imag. part}) = 1$. For eigenvalues with $\alpha = \beta = 0$, a zero vector will be returned as the corresponding eigenvector.

LDVL Leading dimension of the array VL as specified in a dimension or type statement. $\text{LDVL} \geq 1$. If JOBVL = 'V' or 'v' then $\text{LDVL} \geq \max(1, N)$.

xVR On exit, if JOBVR = 'V' or 'v' then VR contains the right generalized eigenvectors. VR is not used if JOBVR = 'N' or 'n'. Real eigenvectors occupy one column. Complex eigenvectors occupy two columns, the first for the real part and the second for the imaginary part. Complex eigenvectors correspond to an eigenvalue with positive imaginary part. With the exception of those eigenvalues with $\alpha = \beta = 0$, each eigenvector will be scaled so the largest component will have $\text{abs}(\text{real part}) + \text{abs}(\text{imag. part}) = 1$. For eigenvalues with $\alpha = \beta = 0$, a zero vector will be returned as the corresponding eigenvector.

LDVR Leading dimension of the array VR as specified in a dimension or type statement. $\text{LDVR} \geq 1$. If JOBVR = 'V' or 'v' then $\text{LDVR} \geq \max(1, N)$.

xWORK Scratch array with a dimension of LDWORK.

LDWORK Leading dimension of the array WORK as specified in a dimension or type statement. $\text{LDWORK} \geq \max(1, 8 \times N)$ for real subroutines or $\max(1, 2 \times N)$ for complex subroutines.

xWORK2 Scratch array with a dimension of $8 \times N$ for complex subroutines.

INFO On exit:
INFO = 0

Subroutine completed normally.

INFO < 0	The <i>i</i> th argument, where $i = \text{INFO} $, had an illegal value. The QZ iteration failed. No eigenvectors have been calculated, but $\text{ALPHAR}(j)$, $\text{ALPHAI}(j)$, and $\text{BETA}(j)$ will be correct for $\text{INFO}+1 \leq j \leq N$.
INFO > N	May indicate an internal LAPACK error.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER          LDA, LDB, LDWORK, M, N
  PARAMETER        (M = 3)
  PARAMETER        (N = 3)
  PARAMETER        (LDA = M)
  PARAMETER        (LDB = M)
  PARAMETER        (LDWORK = 8 * N)

C
  DOUBLE PRECISION A(LDA,N), ALPHAI(N), ALPHAR(N), B(LDB,N)
  DOUBLE PRECISION BETA(N), SCRATCH, WORK(LDWORK)
  INTEGER          ICOL, INFO, IROW

C
  EXTERNAL          DGEGV
  INTRINSIC         ABS, DCMLPX

C
  C Initialize the arrays A and B to store the matrices A and B
  C shown below.
  C
  C
  C      1  1  1      1  1  1
  C  A = 2  1  1      B = 1  0  0
  C      1  3  1      0  1  0
  C
  C      DATA A / 1.0D0, 2.0D0, 1.0D0,
  C $              1.0D0, 1.0D0, 3.0D0,
  C $              1.0D0, 1.0D0, 1.0D0 /
  C      DATA B / 1.0D0, 1.0D0, 0.0D0,
  C $              1.0D0, 0.0D0, 1.0D0,
  C $              1.0D0, 0.0D0, 0.0D0 /

C
  C Print the initial value of the arrays.
  C
  C
  C      PRINT 1000
  C      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
  C      PRINT 1020
  C      PRINT 1010, ((B(IROW,ICOL), ICOL = 1, N), IROW = 1, N)

C
  C Compute and print the eigenvalues.
  C
  C      CALL DGEGV ('NO LEFT EIGENVECTORS',
  C $              'NO RIGHT EIGENVECTORS', N, A, LDA, B, LDB,

```

```

$          ALPHAR, ALPHAI, BETA, SCRATCH, 1, SCRATCH, 1,
$          WORK, LDWORK, INFO)
  IF (INFO .NE. 0) THEN
    IF (INFO .LT. 0) THEN
      PRINT 1030, ABS(INFO)
      STOP 1
    ELSE
      PRINT 1040, INFO
      STOP 2
    END IF
  END IF
  PRINT 1050
  PRINT 1060, (ALPHAR(IROW), ALPHAI(IROW), BETA(IROW),
$   IROW = 1, M)
  PRINT 1070
  PRINT 1080, (DCMPLX(ALPHAR(IROW), ALPHAI(IROW)) /
$   BETA(IROW), IROW = 1, M)
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (3(3X, F4.1))
1020 FORMAT (/1X, 'B:')
1030 FORMAT (1X, 'Illegal argument to DGEGV, argument #', I2)
1040 FORMAT (1X, 'Internal failure in DGEGV, INFO = ', I4)
1050 FORMAT (/1X, T6, 'ALPHA', T20, 'BETA')
1060 FORMAT (1X, '(', F5.2, ', ', F5.2, ')', T19, F5.2)
1070 FORMAT (/1X, 'ALPHA / BETA')
1080 FORMAT (1X, '(', F5.2, ', ', F5.2, ')')
C
  END

```

Sample Output

```

A:
  1.0    1.0    1.0
  2.0    1.0    1.0
  1.0    3.0    1.0

B:
  1.0    1.0    1.0
  1.0    0.0    0.0
  0.0    1.0    0.0

      ALPHA      BETA
( 0.71, 0.00)    0.71
( 0.82, 0.00)    0.82
( 3.46, 0.00)    1.73

ALPHA / BETA
( 1.00, 0.00)
( 1.00, 0.00)
( 2.00, 0.00)

```

LQ Factorization of a General Matrix

The subroutines described in this section compute an LQ factorization of a general matrix A.

Calling Sequence

```
CALL DGELQF (M, N, DA, LDA, DTAU, DWORK, LDWORK, INFO)
CALL SGELQF (M, N, SA, LDA, STAU, SWORK, LDWORK, INFO)
CALL ZGELQF (M, N, ZA, LDA, ZTAU, ZWORK, LDWORK, INFO)
CALL CGELQF (M, N, CA, LDA, CTAU, CWORK, LDWORK, INFO)

void dgelqf (int m, int n, double *da, int lda, double *dtau, int
            *info)
void sgelqf (int m, int n, float *sa, int lda, float *stau, int
            *info)
void zgelfq (int m, int n, doublecomplex *za, int lda,
            doublecomplex *ztau, int *info)
void cgelqf (int m, int n, complex *ca, int lda, complex *ctau, int
            *info)
```

Arguments

M	Number of rows of the matrix A. $M \geq 0$.
N	Number of columns of the matrix A. $N \geq 0$.
xA	On entry, the matrix A. On exit, A and TAU contain an LQ factorization of the matrix A.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, M)$.
xTAU	On exit, the $\min(M, N)$ scalar factors of the elementary reflectors. The elementary reflectors in xTAU together with xA contain an LQ factorization of the matrix A that can be used by other LAPACK subroutines that operate on the matrix A.
xWORK	Scratch array with a dimension of LDWORK.
LDWORK	Leading dimension of the array WORK as specified in a dimension or type statement. $LDWORK \geq \max(1, M)$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The <i>i</i> th argument, where $i = \text{INFO} $, had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER          LDA, LDWORK, M, N
  PARAMETER        (M = 4)
  PARAMETER        (N = 4)
  PARAMETER        (LDA = M)
  PARAMETER        (LDWORK = M)

C
  DOUBLE PRECISION A(LDA,N), TAU(N), WORK(LDWORK)
  INTEGER          ICOL, INFO, IROW

C
  EXTERNAL          DGELQF, DORGLQ
  INTRINSIC         ABS, MIN

C
C   Initialize the array A to store the matrix A shown below.
C
C   A =
C   1   1   1   1
C   1  -1   1  -1
C   5  -5  -1   1
C   9  -1  -5  -3
C
  DATA A / 1.0D0, 1.0D0, 5.0D0, 9.0D0,
$          1.0D0, -1.0D0, -5.0D0, -1.0D0,
$          1.0D0, 1.0D0, -1.0D0, -5.0D0,
$          1.0D0, -1.0D0, 1.0D0, -3.0D0 /

C
C   Compute the LQ factorization of A and print L.
C
  PRINT 1000
  PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, M)
  CALL DGELQF (M, N, A, LDA, TAU, WORK, LDWORK, INFO)
  IF (INFO .NE. 0) THEN
    PRINT 1020, ABS(INFO)
    STOP 1
  END IF
  PRINT 1030
  DO 100, IROW = 1, M
    PRINT 1010, (A(IROW,ICOL), ICOL = 1, MIN(N,IROW)),
$              (0.0D0, ICOL = IROW + 1, N)
  100 CONTINUE

C
C   Compute and print the matrix Q used in the LQ factorization
C   of A.
C
  CALL DORGLQ (N, N, N, A, LDA, TAU, WORK, LDWORK, INFO)
  IF (INFO .NE. 0) THEN
    PRINT 1040, ABS(INFO)
    STOP 2
  END IF
  PRINT 1050
  PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, N)

```

```
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (4(3X, F8.4))
1020 FORMAT (1X, 'Illegal value for argument #', I1,
$           ' in DGELQF.')
1030 FORMAT (/1X, 'L:')
1040 FORMAT (1X, 'Illegal value for argument #', I1,
$           ' in DORGLQ.')
1050 FORMAT (/1X, 'Q:')
C
      END
```

Sample Output

```
A:
  1.0000    1.0000    1.0000    1.0000
  1.0000   -1.0000    1.0000   -1.0000
  5.0000   -5.0000   -1.0000    1.0000
  9.0000   -1.0000   -5.0000   -3.0000

L:
 -2.0000    0.0000    0.0000    0.0000
  0.0000    2.0000    0.0000    0.0000
  0.0000    4.0000    6.0000    0.0000
  0.0000    4.0000    6.0000   -8.0000

Q:
 -0.5000   -0.5000   -0.5000   -0.5000
  0.5000   -0.5000    0.5000   -0.5000
  0.5000   -0.5000   -0.5000    0.5000
 -0.5000   -0.5000    0.5000    0.5000
```

Solution to a Least Squares Problem for a Full-Rank General Matrix (Simple Driver)

The subroutines described in this section solve an overdetermined or underdetermined linear system $AX = B$ or $A^T X = B$ by using a QR or LQ factorization of a general matrix A . The matrix A is assumed to have full rank. Note that the simple driver xGELSS and the expert driver xGELSX are also available for problems in which A is rank-deficient.

Calling Sequence

```
CALL DGELS (TRANSA, M, N, NRHS, DA, LDA, DB, LDB, DWORK, LDWORK,
            INFO)
CALL SGELS (TRANSA, M, N, NRHS, SA, LDA, SB, LDB, SWORK, LDWORK,
            INFO)
CALL ZGELS (TRANSA, M, N, NRHS, ZA, LDA, ZB, LDB, ZWORK, LDWORK,
            INFO)
CALL CGELS (TRANSA, M, N, NRHS, CA, LDA, CB, LDB, CWORK, LDWORK,
            INFO)

void dgels (char trans, int m, int n, int nrhs, double *da, int
            lda, double *db, int ldb, int *info)
void sgels (char trans, int m, int n, int nrhs, float *sa, int
            lda, float *sb, int ldb, int *info)
void zgels (char trans, int m, int n, int nrhs, doublecomplex *za,
            int lda, doublecomplex *zb, int ldb, int *info)
void cgels (char trans, int m, int n, int nrhs, complex *ca, int
            lda, complex *cb, int ldb, int *info)
```

Arguments

TRANSA	Determines the problem to solve. Legal values for TRANSA are shown below. Any values not shown below are illegal. 'N' or 'n' Find a minimum norm solution to $AX = B$ for $M < N$ or minimize $\ B - AX\ $ for $M \geq N$. 'T' or 't' Find a minimum norm solution to $A^T X = B$ for $M \geq N$ or minimize $\ B - A^T X\ $ for $M < N$.
M	Number of rows of the matrix A . $M \geq 0$.
N	Number of columns of the matrix A . $N \geq 0$.
NRHS	Number of right-hand sides, equal to the number of columns of the matrices B and X . $NRHS \geq 0$.

xA	On entry, the matrix A. On exit, if $M \geq N$ then A is overwritten by details of its QR factorization as computed by xGEQRF. On exit, if $M < N$ then A is overwritten by details of its LQ factorization as computed by xGELQF.		
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, M)$.		
xB	On entry, the matrix B of right-hand side vectors, stored columnwise. B is $M \times \text{NRHS}$ if $\text{TRANSA} = \text{'N'}$ or 'n' ; B is $N \times \text{NRHS}$ if $\text{TRANSA} = \text{'T'}$ or 't' . On exit, B is overwritten by the solution vectors, stored columnwise as shown below.		
	M, N	TRANSA	Matrix B
	$M \geq N$	'N' or 'n'	Rows 1 to N of B contain the least squares solution vectors. Residual sum of squares for the solution in each column is given by the sum of squares of elements N+1 to M in that column.
	$M < N$	'N' or 'n'	Rows 1 to N of B contain the minimum norm solution vectors.
	$M \geq N$	'T' or 't'	Rows 1 to M of B contain the minimum norm solution vectors.
	$M < N$	'T' or 't'	Rows 1 to M of B contain the least squares solution vectors. Residual sum of squares for the solution in each column is given by the sum of squares of elements M+1 to N in that column.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, M, N)$.		
xWORK	Scratch array with a dimension of LDWORK.		
LDWORK	Leading dimension of the array WORK as specified in a dimension or type statement. $LDWORK \geq \min(M, N) + \max(1, M, N, \text{NRHS})$.		
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where $i = \text{INFO} $, had an illegal value.		

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
INTEGER          LDA, LDB, LDWORK, M, N, NRHS

```

```

PARAMETER      (M = 4)
PARAMETER      (N = 3)
PARAMETER      (LDA = M)
PARAMETER      (LDB = M)
PARAMETER      (LDWORK = M + N)
PARAMETER      (NRHS = 1)
C
DOUBLE PRECISION A(LDA,N), B(LDB,NRHS), WORK(LDWORK)
INTEGER         ICOL, INFO, IROW
C
EXTERNAL        DGELS
INTRINSIC       ABS
C
C      Initialize the array A to store the coefficient matrix A
C      shown below. Initialize the array B to store the right
C      hand side vector b shown below.
C
C      A = 1   6   2           b = 96
C      A = 1  -2  -8           b = 192
C      A = 1  -2   4           b = 192
C      A = 1   6  14           b = -96
C
DATA A / 1.0D0, 1.0D0, 1.0D0, 1.0D0,
$      6.0D0, -2.0D0, -2.0D0, 6.0D0,
$      2.0D0, -8.0D0, 4.0D0, 1.4D1 /
DATA B / 9.6D1, 1.92D2, 1.92D2, -9.6D1 /
C
C      Print the initial value of the arrays.
C
PRINT 1000
PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, M)
PRINT 1020
PRINT 1030, B
C
C      Compute and print the solution.
C
CALL DGELS ('NO TRANSPOSE A', M, N, NRHS, A, LDA, B, LDB,
$          WORK, LDWORK, INFO)
IF (INFO .NE. 0) THEN
    PRINT 1040, ABS(INFO)
    STOP 1
END IF
PRINT 1050
PRINT 1030, B
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (3(3X, F8.4))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (1X, F8.4)
1040 FORMAT (1X, 'Illegal value for argument #', I2,
$          ' in DGELS.')
1050 FORMAT (/1X, 'min(b - Ax):')
C
END

```

Sample Output

```
A:
  1.0000    6.0000    2.0000
  1.0000   -2.0000   -8.0000
  1.0000   -2.0000    4.0000
  1.0000    6.0000   14.0000

b:
 96.0000
192.0000
192.0000
-96.0000

min(b - Ax):
148.0000
-14.0000
-8.0000
-96.0000
```

Solution to a Least Squares Problem for a Rank-Deficient General Matrix (Simple Driver)

The subroutines described in this section compute a minimum-norm solution to a linear least squares problem minimize $\|B - AX\|_2$ by using the singular value decomposition (SVD) of a general matrix A. The matrix A may be rank-deficient. Note that the expert driver xGELSX is also available and the simple driver xGELS is available for problems in which A has full rank.

Calling Sequence

```
CALL DGELSS (M, N, NRHS, DA, LDA, DB, LDB, DSING, DRCOND, IRANK,
             DWORK, LDWORK, INFO)
CALL SGELSS (M, N, NRHS, SA, LDA, SB, LDB, SSING, SRCOND, IRANK,
             SWORK, LDWORK, INFO)
CALL ZGELSS (M, N, NRHS, ZA, LDA, ZB, LDB, DSING, DRCOND, IRANK,
             ZWORK, LDWORK, DWORK2, INFO)
CALL CGELSS (M, N, NRHS, CA, LDA, CB, LDB, SSING, SRCOND, IRANK,
             CWORK, LDWORK, SWORK2, INFO)

void dgelss (int m, int n, int nrhs, double *da, int lda, double
            *db, int ldb, double *dsing, double rcond, int *rank,
            int *info)
void sgelss (int m, int n, int nrhs, float *sa, int lda, float *sb,
            int ldb, float *ssing, float rcond, int *rank, int
            *info)
void zgelss (int m, int n, int nrhs, doublecomplex *za, int lda,
            doublecomplex *zb, int ldb, double *dsing, double
            rcond, int *rank, int *info)
void cgelss (int m, int n, int nrhs, complex *ca, int lda, complex
            *cb, int ldb, float *ssing, float rcond, int *rank, int
            *info)
```

Arguments

M	Number of rows of the matrix A. $M \geq 0$.
N	Number of columns of the matrix A. $N \geq 0$.
NRHS	Number of right-hand sides, equal to the number of columns of the matrices B and X. $NRHS \geq 0$.

xA	On entry, the matrix A. On exit, the first $\min(M, N)$ rows of A are overwritten with its right singular vectors, stored rowwise.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, M)$.
xB	On entry, the $M \times \text{NRHS}$ right-hand side matrix B. On exit, the $N \times \text{NRHS}$ solution matrix X. If $M \geq N$ and $\text{IRANK} = N$ then the residual sum-of-squares for the solution in the i th column is given by the sum of squares of elements $N+1$ through M in that column. The effective rank of A is determined by treating as zero those singular values that are less than RCOND times the largest singular value.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, M, N)$.
xSING	On exit, the $\min(M, N)$ singular values of A in decreasing order. The condition number of A in the 2-norm = $\text{SING}(1) / \text{SING}(\min(M, N))$. Of course, one must be cautious in computing the condition number because of possible division by zero, overflow, and other potential difficulties.
xRCOND	Determines the effective rank of A. Singular values that satisfy $\text{SING}(i) \leq \text{RCOND} \times \text{SING}(1)$ are treated as zero. If $\text{RCOND} < 0$ then machine precision is used instead.
IRANK	On exit, the effective rank of A, equal to the number of singular values which are greater than $\text{RCOND} \times \text{SING}(1)$.
xWORK	Scratch array with a dimension of LDWORK.
LDWORK	Leading dimension of the array WORK as specified in a dimension or type statement. $\text{LDWORK} \geq 1$ and must also satisfy: for $M \geq N$ $\text{LDWORK} \geq 3 \times N + \max(2 \times N, \text{NRHS}, M)$ for real subroutines $\text{LDWORK} \geq 2 \times N + \max(\text{NRHS}, M)$ for complex subroutines for $M < N$ $\text{LDWORK} \geq 3 \times M + \max(2 \times M, \text{NRHS}, N)$ for real subroutines $\text{LDWORK} \geq 2 \times M + \max(\text{NRHS}, N)$ for complex subroutines
xWORK2	Scratch array with dimension that is the larger of 1 and $(5 \times \min(M, N) - 4)$ for complex subroutines.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The i th argument, where $i = \text{INFO} $, had an illegal value. INFO > 0 The algorithm for computing the SVD failed to converge.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
INTEGER          LDA, LDB, LDSING, LDWORK, M, N, NRHS
PARAMETER        (M = 4)
PARAMETER        (N = 4)
PARAMETER        (LDA = M)
PARAMETER        (LDB = M)
PARAMETER        (LDSING = M)
PARAMETER        (LDWORK = 5 * N)
PARAMETER        (NRHS = 1)

C
DOUBLE PRECISION A(LDA,N), B(LDB,NRHS), RCOND, SING(LDSING)
DOUBLE PRECISION SMALL, WORK(LDWORK)
INTEGER          ICOL, INFO, IROW, IRANK

C
EXTERNAL          DGELSS
INTRINSIC        ABS, SQRT

C
C Initialize the array A to store the coefficient matrix A
C shown below. Initialize the array B to store the right
C hand side vector b shown below. The element labeled z in
C the last column of A differs from 2 only in the last few
C bits. This makes columns three and four nearly equal.
C
C
C      1      6      2      z      96
A = 1  -2  -8  -8      b = 192
C      1  -2   4   4      192
C      1   6  14  14      -96
C
DATA A / 1.0D0, 1.0D0, 1.0D0, 1.0D0,
$        6.0D0, -2.0D0, -2.0D0, 6.0D0,
$        2.0D0, -8.0D0, 4.0D0, 1.4D1,
$        2.0D0, -8.0D0, 4.0D0, 1.4D1 /
DATA B / 9.6D1, 1.92D2, 1.92D2, -9.6D1 /

C
C Print the initial value of the arrays.
C
SMALL = (((((2.0D0 / 3.0D0) + 4.0D0) - 4.0D0) -
$        (2.0D0 / 3.0D0)))
A(1,N) = A(1,N-1) + SMALL
PRINT 1000
PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, M)
PRINT 1020
PRINT 1030, B

C
C Compute and print the rank of A and the solution to the
C minimization problem.
C
RCOND = SQRT(2.2D-16)
CALL DGELSS (M, N, NRHS, A, LDA, B, LDB, SING, RCOND, IRANK,
$           WORK, LDWORK, INFO)

```

```
      IF (INFO .NE. 0) THEN
        PRINT 1040, ABS(INFO)
        STOP 1
      END IF
      PRINT 1050, IRANK
      PRINT 1060
      PRINT 1030, B
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (4(3X, F8.4))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (1X, F8.4)
1040 FORMAT (1X, 'Illegal value for argument #', I1,
$          ' in DGELSS.')
1050 FORMAT (/1X, 'Numerical rank of A: ', I1)
1060 FORMAT (/1X, 'min(b - Ax):')
C
      END
```

Sample Output

```
A:
  1.0000    6.0000    2.0000    2.0000
  1.0000   -2.0000   -8.0000   -8.0000
  1.0000   -2.0000    4.0000    4.0000
  1.0000    6.0000   14.0000   14.0000

b:
 96.0000
192.0000
192.0000
-96.0000

Numerical rank of A: 3

min(b - Ax):
148.0000
-14.0000
-4.0000
-4.0000
```

Solution to a Least Squares Problem for a Rank-Deficient General Matrix (Expert Driver)

The subroutines described in this section compute a minimum-norm solution to a linear least squares problem minimize $\|AX - B\|$ by using complete orthogonal factorization of a general matrix A. The matrix A may be rank-deficient. Note that the simple drivers xGELS (for a full-rank A) and xGELSS (for a rank-deficient A) are also available.

Calling Sequence

```
CALL DGELSX (M, N, NRHS, DA, LDA, DB, LDB, JPIVOT, DRCOND, IRANK,
             DWORK, INFO)
CALL SGELSX (M, N, NRHS, SA, LDA, SB, LDB, JPIVOT, SRCOND, IRANK,
             SWORK, INFO)
CALL ZGELSX (M, N, NRHS, ZA, LDA, ZB, LDB, JPIVOT, DRCOND, IRANK,
             ZWORK, DWORK2, INFO)
CALL CGELSX (M, N, NRHS, CA, LDA, CB, LDB, JPIVOT, SRCOND, IRANK,
             CWORK, SWORK2, INFO)

void dgelsx (int m, int n, int nrhs, double *da, int lda, double
            *db, int ldb, int *jpivot, double rcond, int *rank, int
            *info)
void sgelsx (int m, int n, int nrhs, float *sa, int lda, float *sb,
            int ldb, int *jpivot, float rcond, int *rank, int
            *info)
void zgelsx (int m, int n, int nrhs, doublecomplex *za, int lda,
            doublecomplex *zb, int ldb, int *jpivot, double rcond,
            int *rank, int *info)
void cgelsx (int m, int n, int nrhs, complex *ca, int lda, complex
            *cb, int ldb, int *jpivot, float rcond, int *rank, int
            *info)
```

Arguments

M	Number of rows of the matrix A. $M \geq 0$.
N	Number of columns of the matrix A. $N \geq 0$.
NRHS	Number of right-hand sides, equal to the number of columns of matrices B and X. $NRHS \geq 0$.

xA	On entry, the matrix A. On exit, A has been overwritten by details of its complete orthogonal factorization.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, M)$.
xB	On entry, the $M \times NRHS$ right-hand side matrix B. On exit, the $N \times NRHS$ solution matrix X. If $M \geq N$ and $IRANK = N$ then the residual sum-of-squares for the solution in the i th column is given by the sum of squares of elements $N+1$ through M in that column.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, M, N)$.
JPIVOT	On entry, if $JPIVOT(i) \neq 0$ for $1 \leq i \leq N$ then the i th column of A is an initial column, otherwise it is a free column. Before the QR factorization of A, all initial columns are permuted to the leading positions. Only the remaining free columns are moved as a result of column pivoting during the factorization. On exit, if $JPIVOT(i) \neq k$, then the i th column of A was the k th column of A.
xRCOND	Determines the effective rank of A, defined as the order of the largest leading triangular submatrix R11 in the QR factorization with pivoting of A, whose estimated condition number $< 1 / RCOND$.
IRANK	On exit, the effective rank of A, equal to the order of the submatrix R11 to which the description of RCOND refers.
xWORK	Scratch array whose dimension is the larger of: for real subroutines: $(\min(M, N) + 3 \times N)$ and $(2 \times \min(M, N) + NRHS)$. for complex subroutines: $(\min(M, N) + N)$ and $(\min(M, N) + 2 \times \min(M, N) + NRHS)$.
xWORK2	Scratch array with a dimension of $2 \times N$ for complex subroutines.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The i th argument, where $i = INFO $, had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
INTEGER          LDA, LDB, LDSING, LDWORK, M, N, NRHS
PARAMETER        (M = 4)
PARAMETER        (N = 4)
PARAMETER        (LDA = M)

```

```

PARAMETER      (LDB = M)
PARAMETER      (LDSING = M)
PARAMETER      (LDWORK = 4 * N)
PARAMETER      (NRHS = 1)
C
DOUBLE PRECISION A(LDA,N), B(LDB,NRHS), RCOND, SMALL
DOUBLE PRECISION WORK(LDWORK)
INTEGER        ICOL, INFO, IPIVOT(N), IROW, IRANK
C
EXTERNAL       DGELSX
INTRINSIC      ABS, SQRT
C
C Initialize the array A to store the coefficient matrix A
C shown below. Initialize the array B to store the right
C hand side vector b shown below. The element labeled z in
C the last column of A differs from 2 only in the last few
C bits. This makes columns three and four nearly equal.
C
C      1   6   2   z       96
C  A = 1  -2  -8  -8       b = 192
C      1  -2   4   4       192
C      1   6  14  14       -96
C
DATA A / 1.0D0, 1.0D0, 1.0D0, 1.0D0,
$      6.0D0, -2.0D0, -2.0D0, 6.0D0,
$      2.0D0, -8.0D0, 4.0D0, 1.4D1,
$      2.0D0, -8.0D0, 4.0D0, 1.4D1 /
DATA B / 9.6D1, 1.92D2, 1.92D2, -9.6D1 /
DATA IPIVOT / N*1 /
C
C Print the initial value of the arrays.
C
SMALL = (((2.0D0 / 3.0D0) + 4.0D0) - 4.0D0) -
$      (2.0D0 / 3.0D0)
A(1,N) = A(1,N-1) + SMALL
PRINT 1000
PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, M)
PRINT 1020
PRINT 1030, B
C
C Solve the minimization problem. Print the solution and the
C numerical rank of A.
C
RCOND = SQRT(2.2D-16)
CALL DGELSX (M, N, NRHS, A, LDA, B, LDB, IPIVOT, RCOND,
$           IRANK, WORK, INFO)
IF (INFO .NE. 0) THEN
    PRINT 1040, ABS(INFO)
    STOP 1
END IF
PRINT 1050, IRANK
PRINT 1060
PRINT 1030, B
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (4(3X, F8.4))

```

```
1020 FORMAT (/1X, 'b:')
1030 FORMAT (1X, F8.4)
1040 FORMAT (1X, 'Illegal value for argument #', I2,
$         ' in DGELSX.')
1050 FORMAT (/1X, 'Numerical rank of A: ', I1)
1060 FORMAT (/1X, 'min(b - Ax):')
C
      END
```

Sample Output

```
A:
  1.0000    6.0000    2.0000    2.0000
  1.0000   -2.0000   -8.0000   -8.0000
  1.0000   -2.0000    4.0000    4.0000
  1.0000    6.0000   14.0000   14.0000

b:
 96.0000
192.0000
192.0000
-96.0000

Numerical rank of A: 3

min(b - Ax):
148.0000
-14.0000
-4.0000
-4.0000
```

QL Factorization of a General Matrix

The subroutines described in this section compute a QL factorization of a general matrix A.

Calling Sequence

```
CALL DGEQLF (M, N, DA, LDA, DTAU, DWORK, LDWORK, INFO)
CALL SGEQLF (M, N, SA, LDA, STAU, SWORK, LDWORK, INFO)
CALL ZGEQLF (M, N, ZA, LDA, ZTAU, ZWORK, LDWORK, INFO)
CALL CGEQLF (M, N, CA, LDA, CTAU, CWORK, LDWORK, INFO)

void dgeqlf (int m, int n, double *da, int lda, double *dtau, int
            *info)
void sgeqlf (int m, int n, float *sa, int lda, float *stau, int
            *info)
void zgeqlf (int m, int n, doublecomplex *za, int lda,
            doublecomplex *ztau, int *info)
void cgeqlf (int m, int n, complex *ca, int lda, complex *ctau, int
            *info)
```

Arguments

M	Number of rows of the matrix A. $M \geq 0$.
N	Number of columns of the matrix A. $N \geq 0$.
xA	On entry, the matrix A. On exit, A and TAU contain a QL factorization of the matrix A.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, M)$.
xTAU	On exit, the $\min(M, N)$ scalar factors of the elementary reflectors. The elementary reflectors in xTAU together with xA contain a QL factorization of the matrix A that can be used by other LAPACK subroutines that operate on the matrix A.
xWORK	Scratch array with a dimension of LDWORK.
LDWORK	Leading dimension of the array WORK as specified in a dimension or type statement. $LDWORK \geq \max(1, N)$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The <i>i</i> th argument, where $i = \text{INFO} $, had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER          LDA, LDL, LDWORK, M, N
  PARAMETER        (M = 4)
  PARAMETER        (N = 3)
  PARAMETER        (LDA = M)
  PARAMETER        (LDL = N)
  PARAMETER        (LDWORK = N)

C
  DOUBLE PRECISION A(LDA,N), L(LDL,N), TAU(N), WORK(LDWORK)
  INTEGER          ICOL, INFO, IROW

C
  EXTERNAL          DCOPY, DGEQLF, DORGQL
  INTRINSIC         ABS

C
  Initialize the array A to store the matrix A shown below.
C
C
C   1   2   3
C  A = 1   0  -2
C       1   0   2
C       1  -2  -2
C
  DATA A / 1.0D0, 1.0D0, 1.0D0, 1.0D0,
$          2.0D0, 0.0D0, 0.0D0, -2.0D0,
$          2.0D0, -2.0D0, 2.0D0, -2.0D0 /

C
  Print the initial value of A.
C
  PRINT 1000
  PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, M)

C
  Compute the QL factorization of A.
C
  CALL DGEQLF (M, N, A, LDA, TAU, WORK, LDWORK, INFO)
  IF (INFO .NE. 0) THEN
    PRINT 1020, ABS(INFO)
    STOP 1
  END IF
  DO 100, ICOL = 1, N
    CALL DCOPY (N, A(M-N+1,ICOL), 1, L(1,ICOL), 1)
100 CONTINUE

C
  Compute and print the matrix Q used in the QL factorization
  of A.
C
  CALL DORGQL (M, N, N, A, LDA, TAU, WORK, LDWORK, INFO)
  IF (INFO .NE. 0) THEN
    PRINT 1030, ABS(INFO)
    STOP 2
  END IF
  PRINT 1040

```

```

      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, M)
      PRINT 1050
      DO 110, IROW = 1, N
        PRINT 1010, (L(IROW,ICOL), ICOL = 1, IROW),
          $          (0.0D0, ICOL = IROW + 1, N)
      110 CONTINUE
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (3(3X, F8.4))
1020 FORMAT (1X, 'Illegal value for argument #', I1,
  $          ' in DGEQLF.')
1030 FORMAT (1X, 'Illegal value for argument #', I1,
  $          ' in DORGQL.')
1040 FORMAT (/1X, 'Q:')
1050 FORMAT (/1X, 'L:')
C
      END

```

Sample Output

```

A:
  1.0000    2.0000    2.0000
  1.0000    0.0000   -2.0000
  1.0000    0.0000    2.0000
  1.0000   -2.0000   -2.0000

Q:
 -0.5000    0.5000    0.5000
 -0.5000    0.5000   -0.5000
 -0.5000   -0.5000    0.5000
 -0.5000   -0.5000   -0.5000

L:
 -2.0000    0.0000    0.0000
  0.0000    2.0000    0.0000
  0.0000    2.0000    4.0000

```

QR Factorization with Column Pivoting of a General Matrix

The subroutines described in this section compute a QR factorization with column pivoting of a general matrix A.

Calling Sequence

```
CALL DGEQPF (M, N, DA, LDA, JPIVOT, DTAU, DWORK, INFO)
CALL SGEQPF (M, N, SA, LDA, JPIVOT, STAU, SWORK, INFO)
CALL ZGEQPF (M, N, ZA, LDA, JPIVOT, ZTAU, ZWORK, DWORK2, INFO)
CALL CGEQPF (M, N, CA, LDA, JPIVOT, CTAU, CWORK, SWORK2, INFO)

void dgeqpf (int m, int n, double *da, int lda, int *jpivot, double
            *dtau, int *info)
void sgeqpf (int m, int n, float *sa, int lda, int *jpivot, float
            *stau, int *info)
void zgeqpf (int m, int n, doublecomplex *za, int lda, int *jpivot,
            doublecomplex *ztau, int *info)
void cgeqpf (int m, int n, complex *ca, int lda, int *jpivot,
            complex *ctau, int *info)
```

Arguments

M	Number of rows of the matrix A. $M > 0$.
N	Number of columns of the matrix A. $N > 0$.
xA	On entry, the matrix A. On exit, A and TAU contain a QR factorization of the matrix A.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, M)$.
JPIVOT	On entry, if JPIVOT(i) $\neq 0$ for $1 \leq i \leq N$ then the <i>i</i> th column of A is an initial column, otherwise it is a free column. Before the QR factorization of A, all initial columns are permuted to the leading positions. Only the remaining free columns are moved as a result of column pivoting during the factorization. On exit, if JPIVOT(i) = <i>k</i> , then the <i>i</i> th column of A was the <i>k</i> th column of A.
xTAU	On exit, the $\min(M, N)$ scalar factors of the elementary reflectors. The elementary reflectors in xTAU together with xA contain a QR factorization of the matrix A that can be used by other LAPACK subroutines that operate on the matrix A.
xWORK	Scratch array with a dimension of $3 \times N$ for real subroutines or N for complex subroutines.

xWORK2	Scratch array with a dimension of $2 \times N$ for complex subroutines.
INFO	On exit:
INFO = 0	Subroutine completed normally.
INFO < 0	The ith argument, where $i = INFO $, had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C   INTEGER          LDA, LDR, LDWORK, M, N
C   PARAMETER        (M = 4)
C   PARAMETER        (N = 3)
C   PARAMETER        (LDA = M)
C   PARAMETER        (LDR = N)
C   PARAMETER        (LDWORK = 3 * N)
C
C   DOUBLE PRECISION A(LDA,N), R(LDR,N), TAU(N), WORK(LDWORK)
C   INTEGER          ICOL, INFO, IPIVOT(N), IROW
C
C   EXTERNAL          DCOPY, DGEQPF, DORGQR
C   INTRINSIC         ABS
C
C   Initialize the array A to store the matrix A shown below.
C
C   A =
C   1   6   2
C   1  -2  -8
C   1  -2   4
C   1   6  14
C
C   DATA A / 4*1.0D0, 6.0D0, -2.0D0, -2.0D0, 6.0D0,
$          2.0D0, -8.0D0, 4.0D0, 1.4D1 /
C   DATA IPIVOT / N*0 /
C
C   Print the initial value of A.
C
C   PRINT 1000
C   PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, M)
C
C   Compute the QR factorization of A with pivoting.
C
C   CALL DGEQPF (M, N, A, LDA, IPIVOT, TAU, WORK, INFO)
C   IF (INFO .NE. 0) THEN
C       PRINT 1020, ABS(INFO)
C       STOP 1
C   END IF
C   DO 100, ICOL = 1, N
C       CALL DCOPY (N, A(1,ICOL), 1, R(1,ICOL), 1)
C 100 CONTINUE
C

```



```
C      Compute and print the matrix Q used in the QR factorization
C      of A.
C
      CALL DORGQR (M, N, N, A, LDA, TAU, WORK, LDWORK, INFO)
      IF (INFO .NE. 0) THEN
        PRINT 1030, ABS(INFO)
        STOP 2
      END IF
      PRINT 1040
      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, M)
      PRINT 1050
      DO 110, IROW = 1, N
        PRINT 1010, (0.0D0, ICOL = 1, IROW - 1),
          $          (R(IROW,ICOL), ICOL = IROW, N)
      110 CONTINUE
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (3(3X, F8.4))
1020 FORMAT (1X, 'Illegal value for argument #', I1,
  $          ' in DGEQPF.')
1030 FORMAT (1X, 'Illegal value for argument #', I1,
  $          ' in DORGQR.')
1040 FORMAT (/1X, 'Q:')
1050 FORMAT (/1X, 'R:')
C
      END
```

Sample Output

```
A:
  1.0000    6.0000    2.0000
  1.0000   -2.0000   -8.0000
  1.0000   -2.0000    4.0000
  1.0000    6.0000   14.0000

Q:
 -0.1195   -0.8173   -0.2602
  0.4781   -0.1510   -0.7061
 -0.2390    0.5419   -0.6318
 -0.8367   -0.1244   -0.1858

R:
-16.7332   -6.2152   -0.7171
  0.0000   -6.4321   -0.5508
  0.0000    0.0000   -1.7839
```

QR Factorization of a General Matrix

The subroutines described in this section compute a QR factorization of a general matrix A.

Calling Sequence

```
CALL DGEQRF (M, N, DA, LDA, DTAU, DWORK, LDWORK, INFO)
CALL SGEQRF (M, N, SA, LDA, STAU, SWORK, LDWORK, INFO)
CALL ZGEQRF (M, N, ZA, LDA, ZTAU, ZWORK, LDWORK, INFO)
CALL CGEQRF (M, N, CA, LDA, CTAU, CWORK, LDWORK, INFO)

void dgeqrf (int m, int n, double *da, int lda, double *dtau, int
            *info)
void sgeqrf (int m, int n, float *sa, int lda, float *stau, int
            *info)
void zgeqrf (int m, int n, doublecomplex *za, int lda,
            doublecomplex *ztau, int *info)
void cgeqrf (int m, int n, complex *ca, int lda, complex *ctau, int
            *info)
```

Arguments

M	Number of rows of the matrix A. $M \geq 0$.
N	Number of columns of the matrix A. $N \geq 0$.
xA	On entry, the matrix A. On exit, A and TAU contain a QR factorization of the matrix A.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, M)$.
xTAU	On exit, the $\min(M, N)$ scalar factors of the elementary reflectors. The elementary reflectors in xTAU together with xA contain a QR factorization of the matrix A that can be used by other LAPACK subroutines that operate on the matrix A.
xWORK	Scratch array with a dimension of LDWORK.
LDWORK	Leading dimension of the array WORK as specified in a dimension or type statement. $LDWORK \geq \max(1, N)$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where $i = \text{INFO} $, had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
C   INTEGER          LDA, LDR, LDWORK, M, N
C   PARAMETER        (M = 4)
C   PARAMETER        (N = 3)
C   PARAMETER        (LDA = M)
C   PARAMETER        (LDR = N)
C   PARAMETER        (LDWORK = N)
C
C   DOUBLE PRECISION A(LDA,N), R(LDR,N), TAU(N), WORK(LDWORK)
C   INTEGER          ICOL, INFO, IROW
C
C   EXTERNAL          DCOPY, DGEQRF, DORGQR
C   INTRINSIC         ABS
C
C   Initialize the array A to store the matrix A shown below.
C
C   A =
C   1   6   2
C   1  -2  -8
C   1  -2   4
C   1   6  14
C
C   DATA A / 4*1.0D0, 6.0D0, -2.0D0, -2.0D0, 6.0D0,
C   $          2.0D0, -8.0D0, 4.0D0, 1.4D1 /
C
C   Print the initial value of A.
C
C   PRINT 1000
C   PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, M)
C
C   Compute the QR factorization of A.
C
C   CALL DGEQRF (M, N, A, LDA, TAU, WORK, LDWORK, INFO)
C   IF (INFO .NE. 0) THEN
C       PRINT 1020, ABS(INFO)
C       STOP 1
C   END IF
C   DO 100, ICOL = 1, N
C       CALL DCOPY (N, A(1,ICOL), 1, R(1,ICOL), 1)
100 CONTINUE
C
C   Compute and print the matrix Q used in the QR factorization
C   of A.
C
C   CALL DORGQR (M, N, N, A, LDA, TAU, WORK, LDWORK, INFO)
C   IF (INFO .NE. 0) THEN
C       PRINT 1030, ABS(INFO)
C       STOP 2
C   END IF
C   PRINT 1040
C   PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, M)

```

```

        PRINT 1050
        DO 110, IROW = 1, N
            PRINT 1010, (0.0D0, ICOL = 1, IROW - 1),
$          (R(IROW,ICOL), ICOL = IROW, N)
    110 CONTINUE
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (3(3X, F8.4))
1020 FORMAT (1X, 'Illegal value for argument #', I1,
$          ' in DGEQRF.')
1030 FORMAT (1X, 'Illegal value for argument #', I1,
$          ' in DORGQR.')
1040 FORMAT (/1X, 'Q:')
1050 FORMAT (/1X, 'R:')
C
        END

```

Sample Output

```

A:
  1.0000    6.0000    2.0000
  1.0000   -2.0000   -8.0000
  1.0000   -2.0000    4.0000
  1.0000    6.0000   14.0000

Q:
 -0.5000    0.5000    0.5000
 -0.5000   -0.5000    0.5000
 -0.5000   -0.5000   -0.5000
 -0.5000    0.5000   -0.5000

R:
 -2.0000   -4.0000   -6.0000
  0.0000    8.0000   10.0000
  0.0000    0.0000  -12.0000

```

Refined Solution to a Linear System in an LU-Factored General Matrix

The subroutines described in this section refine the solution to a linear system $AX = B$, $A^T X = B$, or $A^H X = B$ for a general matrix A , which has been LU-factored by xGETRF, and general matrices B and X . These subroutines also compute forward error bounds and backward error estimates for the refined solution.

Calling Sequence

```
CALL DGERFS (TRANSA, N, NRHS, DA, LDA, DAF, LDAF, IPIVOT, DB, LDB,
             DX, LDX, DFERR, DBERR, DWORK, IWORK2, INFO)
CALL SGERFS (TRANSA, N, NRHS, SA, LDA, SAF, LDAF, IPIVOT, SB, LDB,
             SX, LDX, SFERR, SBERR, SWORK, IWORK2, INFO)
CALL ZGERFS (TRANSA, N, NRHS, ZA, LDA, ZAF, LDAF, IPIVOT, ZB, LDB,
             ZX, LDX, DFERR, DBERR, ZWORK, DWORK2, INFO)
CALL CGERFS (TRANSA, N, NRHS, CA, LDA, CAF, LDAF, IPIVOT, CB, LDB,
             CX, LDX, SFERR, SBERR, CWORK, SWORK2, INFO)

void dgerfs (char trans, int n, int nrhs, double *da, int lda,
             double *daf, int ldaf, int *ipivot, double *db, int
             ldb, double *dx, int ldx, double *dferr, double *dberr,
             int *info)
void sgerfs (char trans, int n, int nrhs, float *sa, int lda, float
             *saf, int ldaf, int *ipivot, float *sb, int ldb, float
             *sx, int ldx, float *sferr, float *sberr, int *info)
void zgerfs (char trans, int n, int nrhs, doublecomplex *za, int
             lda, doublecomplex *zaf, int ldaf, int *ipivot,
             doublecomplex *zb, int ldb, doublecomplex *zx, int ldx,
             double *dferr, double *dberr, int *info)
void cgerfs (char trans, int n, int nrhs, complex *ca, int lda,
             complex *caf, int ldaf, int *ipivot, complex *cb, int
             ldb, complex *cx, int ldx, float *sferr, float *sberr,
             int *info)
```

Arguments

TRANSA	Indicates the form of the linear system whose solution is to be refined. The legal values for TRANSA are listed below. Any values not listed below are illegal.
'N' or 'n'	No transpose, refine $AX = B$.
'T' or 't'	Transpose, refine $A^T X = B$.

	'C' or 'c'	Conjugate transpose, refine $A^H X = B$. Note that A^T and A^H are the same for real matrices.
N		Order of the matrix A. $N \geq 0$.
NRHS		Number of right-hand sides, equal to the number of columns of the matrices B and X. $NRHS \geq 0$.
xA		Matrix A.
LDA		Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
xAF		LU factorization of the matrix A as computed by xGETRF.
LDAF		Leading dimension of the array AF as specified in a dimension or type statement. $LDAF \geq \max(1, N)$.
IPIVOT		Pivot indices as computed by xGETRF.
xB		The $N \times NRHS$ right-hand side matrix B.
LDB		Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
xx		On entry, the $N \times NRHS$ solution matrix X. On exit, the refined $N \times NRHS$ solution matrix X.
LDX		Leading dimension of the array X as specified in a dimension or type statement. $LDX \geq \max(1, N)$.
xFERR		On exit, the estimated forward error bound for each solution vector $X(*, j)$ for $1 \leq j \leq NRHS$. If X' is the true solution corresponding to $X(*, j)$ then $FERR(j)$ is an upper bound on the magnitude of the largest element in $X(*, j) - X'$ divided by the magnitude of the largest element in $X(*, j)$.
xBERR		On exit, $BERR(j)$ is the smallest relative change in any element of A or $B(*, j)$ that makes $X(*, j)$ an exact solution to $AX(*, j) = B(*, j)$ for $1 \leq j \leq NRHS$.
xWORK		Scratch array with a dimension of $3 \times N$ for real subroutines or $2 \times N$ for complex subroutines.
xWORK2		Scratch array with a dimension of N.
INFO		On exit: INFO = 0 Subroutine completed normally. INFO < 0 The i th argument, where $i = INFO $, had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
INTEGER LDA, LDAF, LDB, LDIWRK, LDWORK, LDX
INTEGER N, NRHS
PARAMETER (N = 3)

```

```

PARAMETER      (LDA = N)
PARAMETER      (LDAF = N)
PARAMETER      (LDB = N)
PARAMETER      (LDIWRK = N)
PARAMETER      (LDWORK = 3 * N)
PARAMETER      (LDX = N)
PARAMETER      (NRHS = 1)
C
DOUBLE PRECISION A(LDA,N), AF(LDAF,N), B(LDB,NRHS)
DOUBLE PRECISION B1(LDB,NRHS), BERR(NRHS), EPSLON
DOUBLE PRECISION FERR(NRHS), WORK(LDWORK), X(LDX,NRHS)
INTEGER         ICOL, INFO, IPIVOT(N), IROW
INTEGER         IWORK(LDIWRK)
C
EXTERNAL        DCOPY, DGEMV, DGERFS, DGETRF, DGETRS
INTRINSIC       ABS, DBLE
C
EPSLON = (((2.0D0 / 3.0D0) + 1.6D1) - 1.6D1) -
$         (2.0D0 / 3.0D0))
C
C      Initialize the array A to store the matrix A shown below.
C      Each of the diagonal elements is equal to the row number
C      plus epsilon.
C
C      1+e   1   1
C      A = 2   2+e 2
C      3   3   3+e
C
DO 110, ICOL = 1, N
DO 100, IROW = 1, N
A(IROW,ICOL) = DBLE(IROW)
100 CONTINUE
A(ICOL,ICOL) = A(ICOL,ICOL) + EPSLON
110 CONTINUE
CALL DCOPY (LDA*N, A, 1, AF, 1)
C
C      Print the initial value of A.
C
PRINT 1000
DO 120, IROW = 1, N
PRINT 1010, (A(IROW,ICOL), ICOL = 1, N)
120 CONTINUE
C
C      Initialize the array B to store the right hand side
C      vector b shown below. Each element is equal to N plus
C      a small epsilon. Print the contents of the array after
C      initialization.
C
C      N+e
C      b = N+e
C      N+e
C
DO 130, IROW = 1, N
B(IROW,1) = DBLE(N) + EPSLON
130 CONTINUE
CALL DCOPY (N, B, 1, X, 1)

```

```

        PRINT 1020
        PRINT 1030, B
C
C      LU factor A.
C
        CALL DGETRF (N, N, AF, LDAF, IPIVOT, INFO)
        IF (INFO .NE. 0) THEN
            PRINT 1040, INFO
            STOP 1
        END IF
C
C      Use the LU factorization to solve the system.
C
        CALL DGETRS ('NO TRANSPOSE A', N, NRHS, AF, LDAF, IPIVOT,
$              X, LDX, INFO)
        IF (INFO .NE. 0) THEN
            PRINT 1050, INFO
            STOP 2
        END IF
C
C      Print the solution and Ax.
C
        PRINT 1060
        PRINT 1070, X
        CALL DGEMV ('N', N, N, 1.0D0, A, LDA, X, 1, 0.0D0, B1, 1)
        PRINT 1080
        PRINT 1030, B1
C
C      Refine the solution, then print the refined solution and Ax
C      with the refined x.
C
        CALL DGERFS ('NO TRANSPOSE A', N, NRHS, A, LDA, AF, LDAF,
$              IPIVOT, B, LDB, X, LDX, FERR, BERR, WORK,
$              IWORK, INFO)
        IF (INFO .NE. 0) THEN
            PRINT 1090, INFO
            STOP 2
        END IF
        PRINT 1100
        PRINT 1070, X
        CALL DGEMV ('N', N, N, 1.0D0, A, LDA, X, 1, 0.0D0, B1, 1)
        PRINT 1110
        PRINT 1030, B1
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (3(3X, F22.17))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (1X, F22.17)
1040 FORMAT (1X, 'Error factoring A in DGETRF, INFO = ', I5)
1050 FORMAT (1X, 'Error solving Ax=b in DGETRF, INFO = ', I5)
1060 FORMAT (/1X, 'Initial solution for Ax=b:')
1070 FORMAT (1X, E16.6)
1080 FORMAT (/1X, 'Ax with initial x:')
1090 FORMAT (1X, 'Error improving solution in DGERFS,
$              INFO = ', I5)
1100 FORMAT (/1X, 'Improved solution for Ax=b:')

```



```
1110 FORMAT (/1X, 'Ax with improved x:')  
C  
END
```

Sample Output

```
A:  
  1.0000000000000000133    1.0000000000000000000    1.0000000000000000000  
  2.0000000000000000000    2.0000000000000000133    2.0000000000000000000  
  3.0000000000000000000    3.0000000000000000000    3.0000000000000000133  
  
b:  
  3.0000000000000000133  
  3.0000000000000000133  
  3.0000000000000000133  
  
Initial solution for Ax=b:  
  0.112590E+16  
  0.333333E+00  
 -0.112590E+16  
  
Ax with initial x:  
  2.7500000000000000000  
  2.5000000000000000000  
  2.5000000000000000000  
  
Improved solution for Ax=b:  
  0.115717E+16  
  0.625500E+14  
 -0.121972E+16  
  
Ax with improved x:  
  3.2500000000000000000  
  3.5000000000000000000  
  3.0000000000000000000
```

RQ Factorization of a General Matrix

The subroutines described in this section compute an RQ factorization for a general matrix A.

Calling Sequence

```
CALL DGERQF (M, N, DA, LDA, DTAU, DWORK, LDWORK, INFO)
CALL SGERQF (M, N, SA, LDA, STAU, SWORK, LDWORK, INFO)
CALL ZGERQF (M, N, ZA, LDA, ZTAU, ZWORK, LDWORK, INFO)
CALL CGERQF (M, N, CA, LDA, CTAU, CWORK, LDWORK, INFO)

void dgerqf (int m, int n, double *da, int lda, double *dtau, int
            *info)
void sgerqf (int m, int n, float *sa, int lda, float *stau, int
            *info)
void zgerqf (int m, int n, doublecomplex *za, int lda,
            doublecomplex *ztau, int *info)
void cgerqf (int m, int n, complex *ca, int lda, complex *ctau, int
            *info)
```

Arguments

M	Number of rows of the matrix A. $M \geq 0$.
N	Number of columns of the matrix A. $N \geq 0$.
xA	On entry, the matrix A. On exit, A and TAU contain an RQ factorization of the matrix A.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, M)$.
xTAU	On exit, the $\min(M, N)$ scalar factors of the elementary reflectors. The elementary reflectors in xTAU together with xA contain an RQ factorization of the matrix A that can be used by other LAPACK subroutines that operate on the matrix A.
xWORK	Scratch array with a dimension of LDWORK.
LDWORK	Leading dimension of the array WORK as specified in a dimension or type statement. $LDWORK \geq \max(1, M)$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The <i>i</i> th argument, where $i = \text{INFO} $, had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER          LDA, LDR, LDTAU, LDWORK, M, N
  PARAMETER        (M = 5)
  PARAMETER        (N = 4)
  PARAMETER        (LDA = M)
  PARAMETER        (LDR = M)
  PARAMETER        (LDTAU = LDA)
  PARAMETER        (LDWORK = M)

C
  DOUBLE PRECISION A(LDA,N), R(LDR,N), TAU(LDTAU)
  DOUBLE PRECISION WORK(LDWORK)
  INTEGER          ICOL, INFO, IROW

C
  EXTERNAL          DCOPY, DGERQF, DORGRQ
  INTRINSIC         ABS, MAX

C
C   Initialize the array A to store the matrix A shown below.
C
C   1   1   0   0
C   1   1   1   1
C   A = 1   1  -1  -1
C       1  -1   1  -1
C       1  -1  -1   1
C
C   DATA A / 1.0D0, 1.0D0, 1.0D0, 1.0D0, 1.0D0,
$             1.0D0, 1.0D0, 1.0D0, -1.0D0, -1.0D0,
$             0.0D0, 1.0D0, -1.0D0, 1.0D0, -1.0D0,
$             0.0D0, 1.0D0, -1.0D0, -1.0D0, 1.0D0 /

C
C   Print the initial value of the array A.
C
C   PRINT 1000
C   DO 100, IROW = 1, M
C     PRINT 1010, (A(IROW,ICOL), ICOL = 1, N)
100 CONTINUE

C
C   Compute the RQ factorization of A.
C
C   CALL DGERQF (M, N, A, LDA, TAU, WORK, LDWORK, INFO)
C   IF (INFO .NE. 0) THEN
C     PRINT 1020, ABS(INFO)
C     STOP 1
C   END IF
C   DO 110, ICOL = 1, N
C     CALL DCOPY (M, A(1,ICOL), 1, R(1,ICOL), 1)
110 CONTINUE

C
C   Compute and print the matrix Q used in the RQ factorization
C   of A.
C

```

```

      CALL DORGRQ (N, N, N, A(M-N+1,1), LDA, TAU, WORK, LDWORK,
$          INFO)
      IF (INFO .NE. 0) THEN
          PRINT 1030, ABS(INFO)
          STOP 2
      END IF
      PRINT 1040
      DO 120, IROW = 1, M
          PRINT 1010, (0.0D0, ICOL = 1, IROW - (M - N) - 1),
$              (R(IROW,ICOL), ICOL = MAX(1,IROW - (M - N)),N)
120  CONTINUE
      PRINT 1050
      DO 130, IROW = M - N + 1, M
          PRINT 1010, (A(IROW,ICOL), ICOL = 1, N)
130  CONTINUE
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (5(3X, F8.4))
1020 FORMAT (1X, 'Illegal value for argument #',
$          I1, ' in DGERQF.')
1030 FORMAT (1X, 'Illegal value for argument #',
$          I1, ' in DORGRQ.')
1040 FORMAT (/1X, 'R:')
1050 FORMAT (/1X, 'Q:')
C
      END

```

Sample Output

```

A:
  1.0000    1.0000    0.0000    0.0000
  1.0000    1.0000    1.0000    1.0000
  1.0000    1.0000   -1.0000   -1.0000
  1.0000   -1.0000    1.0000   -1.0000
  1.0000   -1.0000   -1.0000    1.0000

R:
 -1.0000   -1.0000    0.0000    0.0000
 -2.0000    0.0000    0.0000    0.0000
  0.0000   -2.0000    0.0000    0.0000
  0.0000    0.0000   -2.0000    0.0000
  0.0000    0.0000    0.0000   -2.0000

Q:
 -0.5000   -0.5000   -0.5000   -0.5000
 -0.5000   -0.5000    0.5000    0.5000
 -0.5000    0.5000   -0.5000    0.5000
 -0.5000    0.5000    0.5000   -0.5000

```

Solution to a Linear System in a General Matrix (Simple Driver)

The subroutines described in this section solve a linear system $AX = B$ for a square general matrix A and general matrices B and X . Note that the expert driver `xGESVX` is also available.

Calling Sequence

```
CALL DGESV (N, NRHS, DA, LDA, IPIVOT, DB, LDB, INFO)
CALL SGESV (N, NRHS, SA, LDA, IPIVOT, SB, LDB, INFO)
CALL ZGESV (N, NRHS, ZA, LDA, IPIVOT, ZB, LDB, INFO)
CALL CGESV (N, NRHS, CA, LDA, IPIVOT, CB, LDB, INFO)
```

```
void dgesv (int n, int nrhs, double *da, int lda, int *ipivot,
double *db, int ldb, int *info)
void sgesv (int n, int nrhs, float *sa, int lda, int *ipivot,
float *sb, int ldb, int *info)
void zgesv (int n, int nrhs, doublecomplex *za, int lda, int
*ipivot, doublecomplex *zb, int ldb, int *info)
void cgesv (int n, int nrhs, complex *ca, int lda, int *ipivot,
complex *cb, int ldb, int *info)
```

Arguments

N	Order of the matrix A . $N \geq 0$.
NRHS	Number of right-hand sides, equal to the number of columns of the matrix B . $NRHS \geq 0$.
xA	On entry, the $N \times N$ matrix A . On exit, the LU factorization of A as computed by <code>xGETRF</code> .
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
IPIVOT	On exit, pivot indices as computed by <code>xGETRF</code> .
xB	On entry, the $N \times NRHS$ right-hand side matrix B . On exit, the $N \times NRHS$ solution matrix X .
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where $i = INFO$, had an illegal value. INFO > 0 $U(i,i)$, where $i = INFO$, is exactly zero and U is therefore singular. The LU factorization has been completed, but the solution could not be computed.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER          LDA, LDB, M, N, NRHS
  PARAMETER        (M = 3)
  PARAMETER        (N = 3)
  PARAMETER        (NRHS = 1)
  PARAMETER        (LDA = M)
  PARAMETER        (LDB = LDA)

C
  DOUBLE PRECISION A(LDA,N), B(LDB,NRHS)
  INTEGER          ICOL, INFO, IROW, IPIVOT(N)

C
  EXTERNAL          DGESV

C
  C Initialize the array A to store the coefficient matrix A
  C shown below. Initialize the array B to store the right
  C hand vector b shown below.
  C
  C      1  2  2      15
  C  A = 2  1  2      b = 15
  C      2  2  1      15
  C
  DATA A / 1.0D0, 3*2.0D0, 1.0D0, 3*2.0D0, 1.0D0 /
  DATA B / 1.5D1, 1.5D1, 1.5D1 /

C
  C Print the initial value of the arrays.
  C
  PRINT 1000
  PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, M)
  PRINT 1020
  PRINT 1030, B

C
  C Solve Ax=b and then print the solution.
  C
  CALL DGESV (N, NRHS, A, LDA, IPIVOT, B, LDB, INFO)
  IF (INFO .EQ. 0) THEN
    PRINT 1040
    PRINT 1030, B
  ELSE
    PRINT 1050, INFO
  END IF

C
  1000 FORMAT (1X, 'A:')
  1010 FORMAT (3(3X, F4.1))
  1020 FORMAT (/1X, 'b:')
  1030 FORMAT (1X, 2X, F4.1)
  1040 FORMAT (/1X, 'x:')
  1050 FORMAT (1X, 'Solve failed with INFO = ', I6)

C
  END

```

Sample Output

```
A:
  1.0    2.0    2.0
  2.0    1.0    2.0
  2.0    2.0    1.0

b:
 15.0
 15.0
 15.0

x:
  3.0
  3.0
  3.0
```

Singular Value Decomposition of a General Matrix

The subroutines described in this section compute the singular values and, optionally, the right and/or left singular vectors of a general matrix A. The singular value decomposition of $A = U\Sigma V^T$ (or $A = U\Sigma V^H$) for an orthogonal or unitary matrix of left singular vectors U, an orthogonal or unitary matrix of right singular vectors V, and a diagonal matrix with the singular values on its diagonal Σ .

Calling Sequence

```
CALL DGESVD (JOBV, JOBV, M, N, DA, LDA, DSING, DU, LDU, DVT, LDVT,
             DWORK, LDWORK, INFO)
CALL SGESVD (JOBV, JOBV, M, N, SA, LDA, SSING, SU, LDU, SVT, LDVT,
             SWORK, LDWORK, INFO)
CALL ZGESVD (JOBV, JOBV, M, N, ZA, LDA, DSING, ZU, LDU, ZVT, LDVT,
             ZWORK, LDWORK, DWORK2, INFO)
CALL CGESVD (JOBV, JOBV, M, N, CA, LDA, SSING, CU, LDU, CVT, LDVT,
             CWORK, LDWORK, SWORK2, INFO)

void dgesvd (char jobv, char jobvt, int m, int n, double *da, int
             lda, double *dsing, double *du, int ldu, double *dvt,
             int ldvt, int *info)
void sgesvd (char jobv, char jobvt, int m, int n, float *sa, int
             lda, float *ssing, float *su, int ldu, float *svt, int
             ldvt, int *info)
void zgesvd (char jobv, char jobvt, int m, int n, doublecomplex
             *za, int lda, double *dsing, doublecomplex *u, int ldu,
             doublecomplex *zvt, int ldvt, int *info)
void cgesvd (char jobv, char jobvt, int m, int n, complex *ca, int
             lda, float *ssing, complex *u, int ldu, complex *cvt,
             int ldvt, int *info)
```

Arguments

JOBV	Indicates which columns of U, which are the left singular vectors will be computed and returned. Legal values of JOBV are shown below. Any values not shown below are illegal. 'A' or 'a' All M left singular vectors are returned in U. 'N' or 'n' No left singular vectors are computed. 'O' or 'o' The first min(M, N) left singular vectors are overwritten on A.

	'S' or 's'	Only the first $\min(M, N)$ left singular vectors are returned in U.
		Note that JOBU and JOBVT cannot both direct that their respective results be written to A.
JOBVT		Indicates which rows of VT, which are the right singular vectors, will be computed and returned. Legal values of JOBVT are shown below. Any values not shown below are illegal.
	'A' or 'a'	All N right singular vectors are returned in VT.
	'N' or 'n'	No right singular vectors are computed.
	'O' or 'o'	The first $\min(M, N)$ right singular vectors are overwritten on A.
	'S' or 's'	Only the first $\min(M, N)$ right singular vectors are returned in VT.
		Note that JOBU and JOBVT cannot both direct that their respective results be written to A.
M		Number of rows of the input matrix A. $M \geq 0$.
N		Number of columns of the input matrix A. $N \geq 0$.
xA		On entry, the matrix A.
		On exit, if JOBU = 'O' or 'o' then A is overwritten with the first $\min(M, N)$ columns of the array U, which contain the left singular vectors stored columnwise.
		On exit, if JOBVT = 'O' or 'o' then A is overwritten with the first $\min(M, N)$ rows of VT, which contain the right singular vectors stored rowwise.
		On exit, if neither JOBU nor JOBVT = 'O' or 'o' then the contents of A are overwritten.
LDA		Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, M)$.
xSING		The $\min(M, N)$ singular values of A, in descending order. The singular values are the diagonal elements of the diagonal matrix Σ .
xU		On exit, the contents of U are a function of the entry value of JOBU:
	JOBU	Size of U Contents of U
	'A' or 'a'	$M \times M$ Orthogonal or unitary matrix U
	'N' or 'n'	Not used U is not used
	'O' or 'o'	Not used U is not used
	'S' or 's'	$M \times \min(M, N)$ First $\min(M, N)$ columns of the orthogonal or unitary matrix U
LDU		Leading dimension of the array U as specified in a dimension or type statement. $LDU \geq 1$.
		If JOBU = 'A', 'a', 'S', or 's' then $LDU \geq \max(1, M)$.

xVT	On exit, the contents of VT are a function of the entry value of JOBVT:															
	<table><tr><th>JOBVT</th><th>Size of VT</th><th>Contents of VT</th></tr><tr><td>'A' or 'a'</td><td>N×N</td><td>Orthogonal or unitary matrix VT</td></tr><tr><td>'N' or 'n'</td><td>Not used</td><td>VT is not used</td></tr><tr><td>'O' or 'o'</td><td>Not used</td><td>VT is not used</td></tr><tr><td>'S' or 's'</td><td>M×min(M, N)</td><td>First min(M, N) rows of the orthogonal or unitary matrix VT stored rowwise</td></tr></table>	JOBVT	Size of VT	Contents of VT	'A' or 'a'	N×N	Orthogonal or unitary matrix VT	'N' or 'n'	Not used	VT is not used	'O' or 'o'	Not used	VT is not used	'S' or 's'	M×min(M, N)	First min(M, N) rows of the orthogonal or unitary matrix VT stored rowwise
JOBVT	Size of VT	Contents of VT														
'A' or 'a'	N×N	Orthogonal or unitary matrix VT														
'N' or 'n'	Not used	VT is not used														
'O' or 'o'	Not used	VT is not used														
'S' or 's'	M×min(M, N)	First min(M, N) rows of the orthogonal or unitary matrix VT stored rowwise														
	Note that VT is returned and not V.															
LDVT	Leading dimension of the array VT as specified in a dimension or type statement.															
	If JOBVT = 'A' or 'a' then $LDVT \geq \max(1, N)$.															
	If JOBVT = 'S' or 's' then $LDVT \geq \min(M, N)$ and $LDVT \geq 1$.															
	For other values of JOBVT, $LDVT \geq 1$.															
xWORK	Scratch array with a dimension of LDWORK.															
LDWORK	Leading dimension of the array WORK as specified in a dimension or type statement. LDWORK is at least 1 but also greater than or equal to:															
	for real subroutines: the larger of $(3 \times \min(M, N) + \max(M, N))$ and $(5 \times \min(M, N) - 4)$.															
	for complex subroutines: $(2 \times \min(M, N) + \max(M, N))$.															
xWORK2	Scratch array with a dimension of $5 \times \max(M, N)$ for complex subroutines.															
INFO	On exit:															
	INFO = 0 Subroutine completed normally.															
	INFO < 0 The ith argument, where i = INFO , had an illegal value.															
	INFO > 0 Convergence failure.															

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER          LDA, LDWORK, M, N
  PARAMETER        (M = 4)
  PARAMETER        (N = 4)
  PARAMETER        (LDA = M)
  PARAMETER        (LDWORK = 5 * M)

C
  DOUBLE PRECISION A(LDA,N), TEMP, SING(M), WORK(LDWORK)
  INTEGER          ICOL, INFO, IROW

C
  EXTERNAL         DGESVD
  INTRINSIC        ABS

C

```

```

C      Initialize the array A to store the matrix A shown below.
C
C      1/2   1/2   1/2   1/2
C      A = 1/2   1/2  -1/2  -1/2
C      1/2  -1/2   1/2  -1/2
C      1/2  -1/2  -1/2   1/2
C
C      DATA A / 5.0D-1, 5.0D-1, 5.0D-1, 5.0D-1,
$              5.0D-1, 5.0D-1, -5.0D-1, -5.0D-1,
$              5.0D-1, -5.0D-1, 5.0D-1, -5.0D-1,
$              5.0D-1, -5.0D-1, -5.0D-1, 5.0D-1 /
C
C      Print the initial value of A.
C
C      PRINT 1000
C      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, M)
C
C      Compute and print the singular values of A.
C
C      CALL DGESVD ('NO LEFT SINGULAR VECTORS',
$                  'NO RIGHT SINGULAR VECTORS', M, N, A, LDA,
$                  SING, TEMP, 1, TEMP, 1, WORK, LDWORK, INFO)
C      IF (INFO .NE. 0) THEN
C          IF (INFO .LT. 0) THEN
C              PRINT 1020, ABS (INFO)
C              STOP 1
C          ELSE
C              PRINT 1030
C              STOP 2
C          END IF
C      END IF
C      PRINT 1040
C      PRINT 1050, SING
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (4(3X, F8.4))
1020 FORMAT (1X, 'Illegal value for argument #', I2,
$           ' in DGESVD.')
1030 FORMAT (1X, 'Singular value decomposition failed',
$           1X, 'to converge.')
1040 FORMAT (/1X, 'Singular values of A:')
1050 FORMAT (1X, F8.4)
C
END

```

Sample Output

```

A:
0.5000    0.5000    0.5000    0.5000
0.5000    0.5000   -0.5000   -0.5000
0.5000   -0.5000    0.5000   -0.5000
0.5000   -0.5000   -0.5000    0.5000

```

```
Singular values of A:  
1.0000  
1.0000  
1.0000  
1.0000
```

Solution to a Linear System in a General Matrix (Expert Driver)

The subroutines described in this section solve a linear system $AX = B$, $A^T X = B$, or $A^H X = B$ for general matrices A , B , and X . These subroutines also compute forward error bounds and backward error estimates for the solution, estimate the condition number of the matrix A and, optionally, equilibrate the system before computing a solution. Note that the simple driver xGESV is also available.

Calling Sequence

```
CALL DGESVX (FACT, TRANSA, N, NRHS, DA, LDA, DAF, LDAF, IPIVOT,
             EQUED, DROWSC, DCOLSC, DB, LDB, DX, LDX, DRCOND, DFERR,
             DBERR, DWORK, IWORK2, INFO)

CALL SGESVX (FACT, TRANSA, N, NRHS, SA, LDA, SAF, LDAF, IPIVOT,
             EQUED, SROWSC, SCOLSC, SB, LDB, SX, LDX, SRCOND, SFERR,
             SBERR, SWORK, IWORK2, INFO)

CALL ZGESVX (FACT, TRANSA, N, NRHS, ZA, LDA, ZAF, LDAF, IPIVOT,
             EQUED, ZROWSC, ZCOLSC, ZB, LDB, ZX, LDX, DRCOND, DFERR,
             DBERR, ZWORK, DWORK2, INFO)

CALL CGESVX (FACT, TRANSA, N, NRHS, CA, LDA, CAF, LDAF, IPIVOT,
             EQUED, CROWSC, CCOLSC, CB, LDB, CX, LDX, SRCOND, SFERR,
             SBERR, CWORK, SWORK2, INFO)

void dgesvx (char fact, char trans, int n, int nrhs, double *da,
             int lda, double *daf, int ldaf, int *ipivot, char
             *equed, double *drowsc, double *dcolsc, double *db, int
             ldb, double *dx, int ldx, double *drcond, double
             *dferr, double *dberr, int *info)

void sgesvx (char fact, char trans, int n, int nrhs, float *sa, int
             lda, float *saf, int ldaf, int *ipivot, char *equed,
             float *drowsc, float *scolsc, float *sb, int ldb, float
             *sx, int ldx, float *srcond, float *sferr, float
             *sberr, int *info)

void zgesvx (char fact, char trans, int n, int nrhs, doublecomplex
             *za, int lda, doublecomplex *zaf, int ldaf, int
             *ipivot, char *equed, double *drowsc, double *dcolsc,
             doublecomplex *zb, int ldb, doublecomplex *zx, int ldx,
             double *drcond, double *dferr, double *dberr, int
             *info)
```

```
void cgesvx (char fact, char trans, int n, int nrhs, complex *ca,
            int lda, complex *caf, int ldaf, int *ipivot, char
            *equed, float *srowsc, float *scalsc, complex *cb, int
            ldb, complex *cx, int ldx, float *srcond, float *sferr,
            float *sberr, int *info)
```

Arguments

FACT Indicates whether or not the factored form of the matrix A is supplied on entry, and if not, whether the matrix A will be equilibrated before it is factored. FACT interacts with EQUED, A, and AF as shown in this table. FACT also affects the contents of IPIVOT; see its description below.

FACT	EQUED entry	EQUED exit	A entry	A exit	AF entry	AF exit
'F', 'f'	'B', 'b'	Unchanged	Row- and col-equilibrated A	Unchanged	LU-factored row- and col-equilibrated A	Unchanged
'F', 'f'	'C', 'c'	Unchanged	Col-equilibrated A	Unchanged	LU-factored col-equilibrated A	Unchanged
'F', 'f'	'N', 'n'	Unchanged	Matrix A	Unchanged	LU-factored A	Unchanged
'F', 'f'	'R', 'r'	Unchanged	Row-equilibrated A	Unchanged	LU-factored row-equilibrated A	Unchanged
'E', 'e'	Not used	'B'	Matrix A	Row- and col-equilibrated A	Not used	LU-factored row- and col-equilibrated A
'E', 'e'	Not used	'C'	Matrix A	Col-equilibrated A	Not used	LU-factored col-equilibrated A
'E', 'e'	Not used	'N'	Matrix A	Unchanged	Not used	LU-factored A
'E', 'e'	Not used	'R'	Matrix A	Row-equilibrated A	Not used	LU-factored row-equilibrated A
'N', 'n'	Not used	'N'	Matrix A	Unchanged	Not used	LU-factored A

The above table uses the following abbreviations:

LU-factored = the factored form of A as computed by xGETRF

Row-equilibrated A = $\text{diag}(\text{ROWSC}) \times A$

Col-equilibrated A = $A \times \text{diag}(\text{COLSC})$

Row- and col-equilibrated A = $\text{diag}(\text{ROWSC}) \times A \times \text{diag}(\text{COLSC})$

TRANSA	Indicates the form of the linear system to solve. The legal values for TRANSA are listed below. Any values not listed below are illegal. 'N' or 'n' No transpose, solve $AX = B$. 'T' or 't' Transpose, solve $A^T X = B$. 'C' or 'c' Conjugate transpose, solve $A^H X = B$. Note that A^T and A^H are the same for real matrices.
N	Order of the matrix A. $N \geq 0$.
NRHS	Number of right-hand sides, equal to the number of columns in the matrices B and X. $NRHS \geq 0$.
xA	The contents of A vary depending on the values of FACT and EQUED. See the table for the argument FACT above.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
xAF	The contents of AF vary depending on the values of FACT and EQUED. See the table for the argument FACT above.
LDAF	Leading dimension of the array AF as specified in a dimension or type statement. $LDAF \geq \max(1, N)$.
IPIVOT	Pivot indices; values vary depending on FACT and EQUED as shown below:

FACT	EQUED Entry	EQUED Exit	IPIVOT Entry	IPIVOT Exit
'F' or 'f'	'B', 'b', 'C', 'c', 'R', 'r'	Not used	Pivot indices from LU factorization of equilibrated A as computed by xGETRF	Unchanged
'F' or 'f'	'N' or 'n'	Not used	Pivot indices from LU factorization of A as computed by xGETRF	Unchanged
'E' or 'e'	Not used	'B', 'C', 'R'	Not used	Pivot indices from LU factorization of equilibrated A as computed by xGETRF
'E' or 'e'	Not used	'N'	Not used	Pivot indices from LU factorization of A as computed by xGETRF
'N' or 'n'	Not used	'B', 'C', 'R'	Not used	Pivot indices from LU factorization of equilibrated A as computed by xGETRF
'N' or 'n'	Not used	'N'	Not used	Pivot indices from LU factorization of A as computed by xGETRF

EQUED	EQUED interacts with FACT and IPIVOT as shown in the tables above. Note that one must be careful about specifying lowercase letters for the entry value of EQUED when FACT = 'F' or 'f' because ordinarily LAPACK will always return a capital letter. If the caller specifies 'b', 'c', 'n', or 'r' on entry then the lowercase letter will still be in EQUED on exit. The caller may later draw incorrect conclusions if the code contains a series of IF statements to determine whether EQUED = 'B', 'C', 'N', or 'R'.										
xROWSC	On entry, if FACT = 'F' or 'f' and EQUED = 'R', 'r', 'B', or 'b', then ROWSC contains the N row scale factors that were used to equilibrate A. Each element of ROWSC must be positive. If FACT or EQUED have other values, the entry values of ROWSC are not used. On exit, if FACT = 'E' or 'e' and EQUED = 'R' or 'B', then ROWSC contains the N row scale factors that were used to equilibrate A. If FACT or EQUED have other values, ROWSC is unchanged.										
xCOLSC	On entry, if FACT = 'F' or 'f' and EQUED = 'R', 'r', 'B', or 'b', then COLSC contains the N column scale factors that were used to equilibrate A. Each element of COLSC must be positive. If FACT or EQUED have other values, the entry values of COLSC are not used. On exit, if FACT = 'E' or 'e' and EQUED = 'R' or 'B', then COLSC contains the N column scale factors that were used to equilibrate A. If FACT or EQUED have other values, COLSC is unchanged.										
xB	On entry, the N×NRHS right-hand side matrix B. On exit, the value of B is determined by the input value TRANSA and the output value of EQUED as shown below. For combinations of TRANSA and EQUED not shown below, B is not changed. <table><tr><th>TRANSA</th><th>EQUED</th><th>Output value of B</th></tr><tr><td>'N', 'n'</td><td>'R' or 'B'</td><td>diag(R) × B</td></tr><tr><td>'T', 't', 'C', 'c'</td><td>'C' or 'B'</td><td>diag(C) × B</td></tr></table>		TRANSA	EQUED	Output value of B	'N', 'n'	'R' or 'B'	diag(R) × B	'T', 't', 'C', 'c'	'C' or 'B'	diag(C) × B
TRANSA	EQUED	Output value of B									
'N', 'n'	'R' or 'B'	diag(R) × B									
'T', 't', 'C', 'c'	'C' or 'B'	diag(C) × B									
LDB	Leading dimension of the array B as specified in a dimension or type statement. LDB ≥ max(1, N).										
xX	On exit, the N×NRHS solution matrix for the original problem. Note that the solution to the original problem may differ from the solution to the equilibrated system for the cases shown below: <table><tr><th>TRANSA</th><th>EQUED</th><th>Solution to equilibrated system</th></tr><tr><td>'N', 'n'</td><td>'C' or 'B'</td><td>diag(C)−1 × X</td></tr><tr><td>'T', 't', 'C', 'c'</td><td>'R' or 'B'</td><td>diag(R)−1 × X</td></tr></table>		TRANSA	EQUED	Solution to equilibrated system	'N', 'n'	'C' or 'B'	diag(C)−1 × X	'T', 't', 'C', 'c'	'R' or 'B'	diag(R)−1 × X
TRANSA	EQUED	Solution to equilibrated system									
'N', 'n'	'C' or 'B'	diag(C)−1 × X									
'T', 't', 'C', 'c'	'R' or 'B'	diag(R)−1 × X									
LDX	Leading dimension of the array X as specified in a dimension or type statement. LDX ≥ max(1, N).										

xRCOND	On exit, an estimate of the reciprocal condition number of the matrix A after equilibration, if any, where the reciprocal condition number is defined to be $1 / (\ A\ \times \ A^{-1}\)$. The reciprocal of the condition number is estimated instead of the condition number itself to avoid overflow or division by zero. If RCOND is less than machine precision (in particular, if RCOND = 0) then A is singular to working precision. In this case, INFO > 0 is returned and the solution and error bounds are not computed.
xFERR	On exit, the estimated forward error bound for each solution vector $X(*, j)$ for $1 \leq j \leq \text{NRHS}$. If X' is the true solution corresponding to $X(*, j)$ then FERR(j) is an upper bound on the magnitude of the largest element in $X(*, j) - X'$ divided by the magnitude of the largest element in $X(*, j)$.
xBERR	On exit, BERR(j) is the smallest relative change in any element of A or $B(*, j)$ that makes $X(*, j)$ an exact solution to $AX(*, j) = B(*, j)$ for $1 \leq j \leq \text{NRHS}$.
xWORK	Scratch array with a dimension of $3 \times N$ for real subroutines or $2 \times N$ for complex subroutines.
xWORK2	Scratch array with a dimension of N for real subroutines or $2 \times N$ for complex subroutines.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where $i = \text{INFO} $, had an illegal value. $1 \leq \text{INFO} \leq N$ U(i,i), where $i = \text{INFO}$, is exactly zero and U is therefore singular. The LU factorization of A has been completed, but the solution and error bounds could not be computed. INFO = N+1 RCOND is less than machine precision. The LU factorization of A has been completed, but A is singular to working precision and so the solution and error bounds could not be computed.

Sample Program

```

C      PROGRAM TEST
C      IMPLICIT NONE

C      INTEGER          LDA, LDAF, LDB, LDWORK, LDX, N, NRHS
C      PARAMETER        (N = 3)
C      PARAMETER        (NRHS = 1)
C      PARAMETER        (LDA = N)
C      PARAMETER        (LDAF = LDA)

```

```

PARAMETER      (LDB = LDA)
PARAMETER      (LDWORK = 4 * N)
PARAMETER      (LDX = N)
C
DOUBLE PRECISION A(LDA,N), AF(LDAF,N), B(LDB,NRHS)
DOUBLE PRECISION BERR(NRHS), COLSCA(N), FERR(NRHS)
DOUBLE PRECISION RCOND, ROWSCA(N), WORK(LDWORK)
DOUBLE PRECISION X(LDX,NRHS)
INTEGER         ICOL, INFO, IROW, IPIVOT(N), IWORK(N)
CHARACTER*1     EQUED
C
EXTERNAL        DGESVX
C
C      Initialize the array A to store the coefficient matrix A
C      shown below.  Initialize the array B to store the right
C      hand vector b shown below.
C
C      1  2  2      15
C      A = 2  1  2      b = 15
C      2  2  1      15
C
DATA A / 1.0D0, 3*2.0D0, 1.0D0, 3*2.0D0, 1.0D0 /
DATA B / 1.5D1, 1.5D1, 1.5D1 /
C
C      Print the initial values of the arrays.
C
PRINT 1000
PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
PRINT 1020
PRINT 1030, B
C
C      Solve the system.
C
CALL DGESVX ('EQUILIBRATE A IF NECESSARY', 'NO TRANSPOSE A',
$           N, NRHS, A, LDA, AF, LDAF, IPIVOT, EQUED,
$           ROWSCA, COLSCA, B, LDB, X, LDX, RCOND, FERR,
$           BERR, WORK, IWORK, INFO)
IF (INFO .EQ. 0) THEN
C
C      Print the solution and error bounds.
C
PRINT 1040
PRINT 1030, X
PRINT 1050, (1.0D0 / RCOND)
IF (EQUED .EQ. 'N') PRINT 1060
IF (EQUED .EQ. 'R') PRINT 1070
IF (EQUED .EQ. 'C') PRINT 1080
IF (EQUED .EQ. 'B') PRINT 1090
PRINT 1100, (IROW, BERR(IROW), IROW = 1, NRHS)
PRINT 1110, (IROW, FERR(IROW), IROW = 1, NRHS)
ELSE
PRINT 1120, INFO
END IF
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (3(3X, F4.1))

```

```
1020 FORMAT (/1X, 'b:')
1030 FORMAT (1X, 2X, F4.1)
1040 FORMAT (/1X, 'x:')
1050 FORMAT (/1X, 'Estimated condition number: ', F6.3)
1060 FORMAT (/1X, 'No equilibration was done on A.')
1070 FORMAT (/1X, 'Row equilibration was done on A.')
1080 FORMAT (/1X, 'Column equilibration was done on A.')
1090 FORMAT (/1X, 'Row and column equilibration was done on A.')
1100 FORMAT (/1X, 'Backward error for system #', I2, ': ', E12.5)
1110 FORMAT (1X, 'Forward error for system #', I2, ': ', E12.5)
1120 FORMAT (1X, 'Solve failed with INFO = ', I6)
C
      END
```

Sample Output

```
A:
  1.0    2.0    2.0
  2.0    1.0    2.0
  2.0    2.0    1.0

b:
 15.0
 15.0
 15.0

x:
  3.0
  3.0
  3.0

Estimated condition number:  7.000

No equilibration was done on A.

Backward error for system # 1:  0.000000E+00
Forward error for system # 1:  0.62172E-14
```

LU Factorization of a General Matrix

The subroutines described in this section compute an LU factorization of a general matrix A. It is typical to follow a call to xGETRF with a call to xGESVX or xGETRS to solve a linear system $AX = B$, to xGECON to estimate the condition number of A, or to xGETRI to compute A^{-1} .

Calling Sequence

```
CALL DGETRF (M, N, DA, LDA, IPIVOT, INFO)
CALL SGETRF (M, N, SA, LDA, IPIVOT, INFO)
CALL ZGETRF (M, N, ZA, LDA, IPIVOT, INFO)
CALL CGETRF (M, N, CA, LDA, IPIVOT, INFO)

void dgetrf (int m, int n, double *da, int lda, int *ipivot, int
             *info)
void sgetrf (int m, int n, float *sa, int lda, int *ipivot, int
             *info)
void zgetrf (int m, int n, doublecomplex *za, int lda, int *ipivot,
             int *info)
void cgetrf (int m, int n, complex *ca, int lda, int *ipivot, int
             *info)
```

Arguments

M	Number of rows of the matrix A. $M \geq 0$.
N	Number of columns of the matrix A. $N \geq 0$.
xA	On entry, the matrix A. On exit, an LU factorization of A.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, M)$.
IPIVOT	On exit, the pivot indices. During factorization, row i was exchanged with row IPIVOT(i) for $1 \leq i \leq \min(M, N)$.
INFO	On exit: INFO = 0 INFO < 0 INFO > 0 Subroutine completed normally. The ith argument, where $i = \text{INFO}$, had an illegal value. U(i,i), where $i = \text{INFO}$, is exactly zero and U is therefore singular. The factorization has been completed, but division by zero will occur if this factorization is used to solve a linear system.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER          LDA, LDB, M, N, NRHS, SIZEB
  PARAMETER        (M = 3)
  PARAMETER        (N = 3)
  PARAMETER        (NRHS = 1)
  PARAMETER        (LDA = M)
  PARAMETER        (LDB = LDA)
  PARAMETER        (SIZEB = LDB * NRHS)

C
  DOUBLE PRECISION A(LDA,N), B(LDB,NRHS)
  INTEGER          ICOL, INFO, IROW, IPIVOT(N)

C
  EXTERNAL          DGETRF, DGETRS

C
  C Initialize the array A to store the coefficient matrix A
  C shown below. Initialize the array B to store the right
  C hand side vector b shown below.
  C
  C      1  2  2      15
  C  A = 2  1  2      b = 15
  C      2  2  1      15
  C
  DATA A / 1.0D0, 3*2.0D0, 1.0D0, 3*2.0D0, 1.0D0 /
  DATA B / SIZEB*1.5D1 /

C
  C Print the initial value of the arrays.
  C
  PRINT 1000
  PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, M)
  PRINT 1020
  PRINT 1030, B

C
  C Factor A, solve the system, and print the solution.
  C
  CALL DGETRF (M, N, A, LDA, IPIVOT, INFO)
  IF (INFO .EQ. 0) THEN
    CALL DGETRS ('NO TRANSPOSE A', N, NRHS, A, LDA, IPIVOT,
$      B, LDB, INFO)
    IF (INFO .EQ. 0) THEN
      PRINT 1040
      PRINT 1030, B
    ELSE
      PRINT 1050, INFO
    END IF
  ELSE
    PRINT 1060, INFO
  END IF

C
1000 FORMAT (1X, 'A:')
1010 FORMAT (3(3X, F4.1))

```

```
1020 FORMAT (/1X, 'b:')
1030 FORMAT (1X, 2X, F4.1)
1040 FORMAT (/1X, 'x:')
1050 FORMAT (1X, 'Solve failed with INFO = ', I6)
1060 FORMAT (1X, 'Factorization failed with INFO = ', I6)
C
      END
```

Sample Output

```
A:
  1.0    2.0    2.0
  2.0    1.0    2.0
  2.0    2.0    1.0

b:
 15.0
 15.0
 15.0

x:
 3.0
 3.0
 3.0
```

Inverse of an LU-Factored General Matrix

The subroutines described in this section compute the inverse of a general matrix A, which has been LU-factored by xGETRF.

Calling Sequence

```
CALL DGETRI (N, DA, LDA, IPIVOT, DWORK, LDWORK, INFO)
CALL SGETRI (N, SA, LDA, IPIVOT, SWORK, LDWORK, INFO)
CALL ZGETRI (N, ZA, LDA, IPIVOT, ZWORK, LDWORK, INFO)
CALL CGETRI (N, CA, LDA, IPIVOT, CWORK, LDWORK, INFO)

void dgetri (int n, double *da, int lda, int *ipivot, int *info)
void sgetri (int n, float *sa, int lda, int *ipivot, int *info)
void zgetri (int n, doublecomplex *za, int lda, int *ipivot, int
             *info)
void cgetri (int n, complex *ca, int lda, int *ipivot, int *info)
```

Arguments

N	Order of the matrix A. $N \geq 0$.
xA	On entry, the LU factorization of A as computed by xGETRF. On exit, A-1.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
IPIVOT	Pivot indices as computed by xGETRF.
xWORK	Scratch array with a dimension of LDWORK.
LDWORK	Leading dimension of the array WORK as specified in a dimension or type statement. $LDWORK \geq \max(1, N)$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The <i>i</i> th argument, where $i = INFO $, had an illegal value. INFO > 0 U(<i>i</i> , <i>i</i>), where $i = INFO$, is exactly zero. The matrix is therefore singular and its inverse could not be computed.

Sample Program

```
PROGRAM TEST
IMPLICIT NONE
C
INTEGER LDA, LDB, LDWORK, M, N
```

```

PARAMETER      (M = 3)
PARAMETER      (N = 3)
PARAMETER      (LDA = M)
PARAMETER      (LDB = LDA)
PARAMETER      (LDWORK = N)
C
DOUBLE PRECISION A(LDA,N), WORK(LDWORK)
INTEGER          ICOL, INFO, IROW, IPIVOT(N)
C
EXTERNAL        DGETRF, DGETRI
C
C Initialize the array A to store the matrix A shown below.
C
C      1   4   6
C  A =  2   9  14
C      3  14  23
C
DATA A / 1.0D0, 2.0D0, 3.0D0,
$        4.0D0, 9.0D0, 1.4D1,
$        6.0D0, 1.4D1, 2.3D1 /
C
C Print the initial value of A.
C
PRINT 1000
PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, M)
C
C LU factor A, then use the factorization to compute the
C inverse.
C
CALL DGETRF (M, N, A, LDA, IPIVOT, INFO)
IF (INFO .EQ. 0) THEN
    CALL DGETRI (N, A, LDA, IPIVOT, WORK, LDWORK, INFO)
    IF (INFO .EQ. 0) THEN
        PRINT 1020
        PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, M)
    ELSE
        PRINT 1030, INFO
    END IF
ELSE
    PRINT 1040, INFO
END IF
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (3(3X, F4.1))
1020 FORMAT (/1X, 'A**(-1):')
1030 FORMAT (1X, 'Inverse failed with INFO = ', I6)
1040 FORMAT (1X, 'Factorization failed with INFO = ', I6)
C
END

```


Sample Output

```
A:
  1.0    4.0    6.0
  2.0    9.0   14.0
  3.0   14.0   23.0

A**(-1):
 11.0   -8.0    2.0
 -4.0    5.0   -2.0
  1.0   -2.0    1.0
```

Solution to a Linear System in an LU-Factored General Matrix

The subroutines described in this section solve a linear system $AX = B$, $A^T X = B$, or $A^H X = B$ for a general matrix A , which has been LU-factored by xGETRF, and general matrices B and X .

Calling Sequence

```
CALL DGETRS (TRANSA, N, NRHS, DA, LDA, IPIVOT, DB, LDB, INFO)
CALL SGETRS (TRANSA, N, NRHS, SA, LDA, IPIVOT, SB, LDB, INFO)
CALL ZGETRS (TRANSA, N, NRHS, ZA, LDA, IPIVOT, ZB, LDB, INFO)
CALL CGETRS (TRANSA, N, NRHS, CA, LDA, IPIVOT, CB, LDB, INFO)

void dgetrs (char trans, int n, int nrhs, double *da, int lda, int
             *ipivot, double *db, int ldb, int *info)
void sgetrs (char trans, int n, int nrhs, float *sa, int lda, int
             *ipivot, float *sb, int ldb, int *info)
void zgetrs (char trans, int n, int nrhs, doublecomplex *za, int
             lda, int *ipivot, doublecomplex *zb, int ldb, int
             *info)
void cgetrs (char trans, int n, int nrhs, complex *ca, int lda, int
             *ipivot, complex *cb, int ldb, int *info)
```

Arguments

TRANSA	Indicates the form of the linear system to solve. The legal values for TRANSA are listed below. Any values not listed below are illegal. 'N' or 'n' No transpose, solve $AX = B$. 'T' or 't' Transpose, solve $A^T X = B$. 'C' or 'c' Conjugate transpose, solve $A^H X = B$. Note that $A^T = A^H$ for real matrices.
N	Order of the matrix A . $N \geq 0$.
NRHS	Number of right-hand sides, equal to the number of columns of the matrix B . $NRHS \geq 0$.
xA	LU factorization of matrix A as computed by xGETRF.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
IPIVOT	Pivot indices as computed by xGETRF.
xB	On entry, the $N \times NRHS$ right-hand side matrix B . On exit, the $N \times NRHS$ solution matrix X .

LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The <i>i</i> th argument, where $i = INFO $, had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C      INTEGER          LDA, LDB, M, N, NRHS, SIZEB
C      PARAMETER        (M = 3)
C      PARAMETER        (N = 3)
C      PARAMETER        (NRHS = 1)
C      PARAMETER        (LDA = M)
C      PARAMETER        (LDB = LDA)
C      PARAMETER        (SIZEB = LDB * NRHS)
C
C      DOUBLE PRECISION A(LDA,N), B(LDB,NRHS)
C      INTEGER          ICOL, INFO, IROW, IPIVOT(N)
C
C      EXTERNAL          DGETRF, DGETRS
C
C      Initialize the array A to store the coefficient matrix A
C      shown below. Initialize the array B to store the right
C      hand side vector b shown below.
C
C      1  1  2      15
C      A = 2  1  2      b = 15
C      2  2  1      15
C
C      DATA A / 1.0D0, 3*2.0D0, 1.0D0, 3*2.0D0, 1.0D0 /
C      DATA B / SIZEB*1.5D1 /
C
C      Print the initial value of the arrays.
C
C      PRINT 1000
C      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, M)
C      PRINT 1020
C      PRINT 1030, B
C
C      LU factor A then use the factorization to solve the system.
C
C      CALL DGETRF (M, N, A, LDA, IPIVOT, INFO)
C      IF (INFO .EQ. 0) THEN
C          CALL DGETRS ('NO TRANSPOSE A', N, NRHS, A, LDA, IPIVOT,
C          $          B, LDB, INFO)
C          IF (INFO .EQ. 0) THEN
C              PRINT 1040
C              PRINT 1030, B

```

```

        ELSE
            PRINT 1050, INFO
        END IF
    ELSE
        PRINT 1060, INFO
    END IF
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (3(3X, F4.1))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (1X, 2X, F4.1)
1040 FORMAT (/1X, 'x:')
1050 FORMAT (1X, 'Solve failed with INFO = ', I6)
1060 FORMAT (1X, 'Factorization failed with INFO = ', I6)
C
    END

```

Sample Output

```

A:
  1.0    2.0    2.0
  2.0    1.0    2.0
  2.0    2.0    1.0

b:
 15.0
 15.0
 15.0

x:
  3.0
  3.0
  3.0

```

Solution to a Generalized Linear Regression Model Problem

The subroutines described in this section solve a generalized linear regression model (GLM) problem

$$\underset{x, y}{\text{minimize}} \quad y^T \times y \quad \text{subject to} \quad d = Ax + By$$

using a generalized QR factorization of A and B, where A is an N×M matrix, B is an N×P matrix, and d is a vector of length N. It is also assumed that $M \leq N \leq M+P$ and:

$$\text{rank}(A) = M \quad \text{and} \quad \text{rank}([AB]) = N$$

Under these assumptions, the constrained equation is always consistent, and there is a unique solution x and a minimal 2-norm solution y. In particular, if matrix B is square nonsingular, then the problem GLM is equivalent to the following weighted linear least squares problem:

$$\underset{x}{\text{minimize}} \quad \|B^{-1} \times (b - Ax)\|_2$$

Calling Sequence

```
CALL DGGGLM (N, M, P, DA, LDA, DB, LDB, DD, DX, DY, DWORK, LDWORK,
             INFO)
CALL SGGGLM (N, M, P, SA, LDA, SB, LDB, SD, SX, SY, SWORK, LDWORK,
             INFO)
CALL ZGGGLM (N, M, P, ZA, LDA, ZB, LDB, ZD, ZX, ZY, ZWORK, LDWORK,
             INFO)
CALL CGGGLM (N, M, P, CA, LDA, CB, LDB, CD, CX, CY, CWORK, LDWORK,
             INFO)

void dggglm (int n, int m, int p, double *da, int lda, double *db,
             int ldb, double *dd, double *dx, double *dy, int *info)
void sggglm (int n, int m, int p, float *sa, int lda, float *sb,
             int ldb, float *sd, float *sx, float *dy, int *info)
void zggglm (int n, int m, int p, doublecomplex *za, int lda,
             doublecomplex *zb, int ldb, doublecomplex *zd,
             doublecomplex *zx, doublecomplex *zy, int *info)
```

```
void cggglm (int n, int m, int p, complex *ca, int lda, complex
            *cb, int ldb, complex *cd, complex *cx, complex *cy,
            int *info)
```

Arguments

N	Number of rows of the matrices A and B. $N \geq 0$.
M	Number of columns of the matrix A. $M \geq 0$.
P	Number of columns of the matrix B. $M+P \geq N \geq M$.
xA	On entry, the $N \times M$ matrix A. On exit, A is overwritten.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
xB	On entry, the $N \times P$ matrix B. On exit, B is overwritten.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
xD	On entry, the left-hand side of the GLM equation. On exit, D is overwritten.
xX	On exit, X and Y are the solutions of the GLM problem.
xY	On exit, X and Y are the solutions of the GLM problem.
xWORK	Scratch array with a dimension of LDWORK.
LDWORK	Leading dimension of the array WORK as specified in a dimension or type statement. $LDWORK \geq M + P + \max(N, M, P)$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where $i = \text{INFO} $, had an illegal value.

Sample Program

```

C      PROGRAM TEST
C      IMPLICIT NONE
C
C      INTEGER          LDA, LDB, LDWORK, N
C      PARAMETER        (N = 3)
C      PARAMETER        (LDA = N)
C      PARAMETER        (LDB = N)
C      PARAMETER        (LDWORK = 3 * N)
C
C      DOUBLE PRECISION A(LDA,N), B(LDB,N), DLEFT(N), X(N), Y(N)
C      DOUBLE PRECISION WORK(LDWORK)
C      INTEGER          ICOL, INFO, IROW
C
C      EXTERNAL          DGGGLM
```

```

C      INTRINSIC      ABS
C
C      Initialize the array A to store the matrix A shown below.
C      Initialize the array B to store the matrix B shown below.
C      Initialize the array DLEFT to store the left hand side d
C      of the GLM equation as shown below.
C
C      A =  $\begin{bmatrix} 1 & 4 & 6 \\ 2 & 9 & 14 \\ 3 & 14 & 23 \end{bmatrix}$       B =  $\begin{bmatrix} 11 & -8 & 2 \\ -4 & 5 & -2 \\ 1 & -2 & 1 \end{bmatrix}$       d =  $\begin{bmatrix} 1 \\ 10 \\ 100 \end{bmatrix}$ 
C
C      DATA A / 1.0D0, 2.0D0, 3.0D0,
C      $          4.0D0, 9.0D0, 1.4D1,
C      $          6.0D0, 1.4D1, 2.3D1 /
C      DATA B / 1.1D1, -4.0D0, 1.0D0,
C      $          -8.0D0, 5.0D0, 2.0D0,
C      $          2.0D0, -2.0D0, 1.0D0 /
C      DATA DLEFT / 1.0D0, 1.0D1, 1.0D2 /
C
C      Print the initial values of A and B.
C
C      PRINT 1000
C      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
C      PRINT 1020
C      PRINT 1010, ((B(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
C
C      Compute and print the solution to the linear regression
C      problem.
C
C      CALL DGGGLM (N, N, N, A, LDA, B, LDB, DLEFT, X, Y, WORK,
C      $              LDWORK, INFO)
C      IF (INFO .LT. 0) THEN
C          PRINT 1030, ABS(INFO)
C          STOP 1
C      END IF
C      PRINT 1040
C      DO 100, IROW = 1, N
C          PRINT 1050, X(IROW), Y(IROW)
C      100 CONTINUE
C
C      1000 FORMAT (1X, 'A:')
C      1010 FORMAT (3(3X, F4.1))
C      1020 FORMAT (/1X, 'B:')
C      1030 FORMAT (1X, 'Illegal value for argument #', I1,
C      $           ' in DGGGLM.')
C      1040 FORMAT (/1X, 3X, 'X', 7X, 'Y')
C      1050 FORMAT (1X, F6.1, 2X, F6.1)
C
C      END

```

Sample Output

```
A:
  1.0    4.0    6.0
  2.0    9.0   14.0
  3.0   14.0   23.0

B:
 11.0   -8.0    2.0
  -4.0    5.0   -2.0
   1.0    2.0    1.0

  X      Y
131.0   0.0
-154.0   0.0
 81.0   0.0
```


Solution to a Linear Equality Constrained Least Squares Problem

The subroutines described in this section solve the linear equality constrained least squares (LSE) problem

$$\text{minimize } \|Ax - c\|_2 \quad \text{subject to} \quad Bx = d$$

using a generalized RQ factorization of matrices A and B, where A is M×N, B is P×N, and $P \leq N \leq M+P$. It is assumed that $\text{rank}(B) = p$ and that the null spaces of A and B intersect only trivially; these conditions ensure that the problem has a unique solution.

Calling Sequence

```
CALL DGGLSE (M, N, P, DA, LDA, DB, LDB, DC, DD, DX, DWORK, LDWORK,
             INFO)
CALL SGGLSE (M, N, P, SA, LDA, SB, LDB, SC, SD, SX, SWORK, LDWORK,
             INFO)
CALL ZGGLSE (M, N, P, ZA, LDA, ZB, LDB, ZC, ZD, ZX, ZWORK, LDWORK,
             INFO)
CALL CGGLSE (M, N, P, CA, LDA, CB, LDB, CC, CD, CX, CWORK, LDWORK,
             INFO)

void dgglse (int m, int n, int p, double *da, int lda, double *db,
             int ldb, double *dc, double *dd, double *dx, int *info)
void sgglse (int m, int n, int p, float *sa, int lda, float *sb,
             int ldb, float *sc, float *sd, float *sx, int *info)
void zgglse (int m, int n, int p, doublecomplex *za, int lda,
             doublecomplex *zb, int ldb, doublecomplex *zc,
             doublecomplex *zd, doublecomplex *zx, int *info)
void cgglse (int m, int n, int p, complex *ca, int lda, complex
             *cb, int ldb, complex *cc, complex *cd, complex *cx,
             int *info)
```

Arguments

M	Number of rows of the matrix A. $M \geq 0$.
N	Number of columns of the matrices A and B. $N \geq 0$.
P	Number of rows of the matrix B. $M+P \geq N \geq P \geq 0$.
xA	On entry, the M×N matrix A. On exit, A is overwritten.

LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, M)$.
xB	On entry, the $P \times N$ matrix B. On exit, B is overwritten.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, P)$.
xC	On entry, the right-hand side vector C of length M for the least squares part of the LSE problem. On exit, the residual sum of squares for the solution is given by the sum of squares of elements $N-P+1$ to M of vector C.
xD	On entry, the right-hand side vector D of length P for the constrained equation. On exit, D is overwritten.
xX	On exit, the N-element array X contains the solution of the LSE problem.
xWORK	Scratch array with a dimension of LDWORK.
LDWORK	Leading dimension of the array WORK as specified in a dimension or type statement. $LDWORK \geq N + P + \max(M, N, P)$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The <i>i</i> th argument, where $i = INFO $, had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER          LDA, LDB, LDWORK
  INTEGER          N
  PARAMETER        (N = 3)
  PARAMETER        (LDA = N)
  PARAMETER        (LDB = N)
  PARAMETER        (LDWORK = 3 * N)

C
  DOUBLE PRECISION A(LDA,N), B(LDB,N), C(N), D(N)
  DOUBLE PRECISION WORK(LDWORK), X(N)
  INTEGER          ICOL, INFO, IROW

C
  EXTERNAL          DGGLSE

C
C  Initialize the array A to store the matrix A shown below.
C  Initialize the array B to store the matrix B shown below.
C  Initialize the array C to store the right hand side vector
C  c for the least squares problem shown below. Initialize
C  the array D to store the right hand side vector d for the
C  constrained equation as shown below.

```

```

C
C      1   4   6      11 -8   2      100      1
C      A = 2   9  14      B = -4   5  -2      c = 10      d = 10
C      3  14  23      1 -2   1      1      100
C
C      DATA A / 1.0D0, 2.0D0, 3.0D0,
$           4.0D0, 9.0D0, 1.4D1,
$           6.0D0, 1.4D1, 2.3D1 /
C      DATA B / 1.1D1, -4.0D0, 1.0D0,
$           -8.0D0, 5.0D0, 2.0D0,
$           2.0D0, -2.0D0, 1.0D0 /
C      DATA C / 1.0D2, 1.0D1, 1.0D0 /
C      DATA D / 1.0D0, 1.0D1, 1.0D2 /
C
C      Print the initial values of A and B.
C
C      PRINT 1000
C      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
C      PRINT 1020
C      PRINT 1010, ((B(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
C
C      Compute and print the solution.
C
C      CALL DGGLSE (N, N, N, A, LDA, B, LDB, C, D, X, WORK, LDWORK,
$                INFO)
C      IF (INFO .LT. 0) THEN
C          PRINT 1030, ABS(INFO)
C          STOP 1
C      END IF
C      PRINT 1040
C      PRINT 1050, X
C
C      1000 FORMAT (1X, 'A:')
C      1010 FORMAT (3(3X, F4.1))
C      1020 FORMAT (/1X, 'B:')
C      1030 FORMAT (1X, 'Illegal value for argument #', I1,
$                ' in DGGLSE.')
C      1040 FORMAT (/1X, ' X')
C      1050 FORMAT (1X, F6.1)
C
C      END

```

Sample Output

```

A:
1.0   4.0   6.0
2.0   9.0  14.0
3.0  14.0  23.0

```

B:			
11.0	-8.0	2.0	
-4.0	5.0	-2.0	
1.0	2.0	1.0	
X			
12.8			
26.2			
34.9			

Generalized Singular Value Decomposition of General Matrices

The subroutines described in this section compute the matrices C, S, and R from the generalized singular value decomposition of the M×N general matrix A and the P×N general matrix B. Optionally, these subroutines may also compute the orthogonal transformation matrices U, V, and Q.

If the numerical rank of the matrix $[A^T \ B^T]^T$ is equal to (K+L), and $(M - K - L) \geq 0$ then the generalized SVD is defined to be:

$$U^T \times A \times Q = \begin{bmatrix} I_{K \times K} & O_{K \times L} \\ O_{L \times K} & C_{L \times L} \\ O_{(M-K-L) \times K} & O_{(M-K-L) \times L} \end{bmatrix} \times \begin{bmatrix} O_{(K+L) \times (N-K-L)} & R_{(K+L) \times (K+L)} \end{bmatrix}$$

and

$$V^T \times B \times Q = \begin{bmatrix} O_{L \times K} & S_{L \times L} \\ O_{(P-L) \times K} & O_{(P-L) \times L} \end{bmatrix} \times \begin{bmatrix} O_{(K+L) \times (N-K-L)} & R_{(K+L) \times (K+L)} \end{bmatrix}$$

where U, V, and Q are orthogonal or unitary matrices, R is nonsingular upper triangular, $[C]^2 + [S]^2 = [I]$, and

$$C = \begin{bmatrix} \alpha_{K+1} & & & \\ & \alpha_{K+2} & & \\ & & \dots & \\ & & & \alpha_{K+L} \end{bmatrix} \quad \text{and} \quad S = \begin{bmatrix} \beta_{K+1} & & & \\ & \beta_{K+2} & & \\ & & \dots & \\ & & & \beta_{K+L} \end{bmatrix}$$

On exit, R is stored in A(1:K+L, N-K-L+1:N).

If the numerical rank of the matrix $[A^T \ B^T]^T$ is equal to (K+L), and $(M - K - L) < 0$ then the generalized SVD is defined to be:

$$U^T \times A \times Q = \begin{bmatrix} I_{K \times K} & O_{K \times (M-K)} & O_{K \times (K+L-M)} \\ O_{(M-K) \times K} & C_{(M-K) \times (M-K)} & O_{(M-K) \times (K+L-M)} \end{bmatrix} \times \begin{bmatrix} O_{(K+L) \times (N-K-L)} & R_{(K+L) \times (K+L)} \end{bmatrix}$$

and

$$V^T \times B \times Q = \begin{bmatrix} O_{(M-K) \times K} & S_{(M-K) \times (M-K)} & O_{(M-K) \times (K+L-M)} \\ O_{(K+L-M) \times K} & O_{(K+L-M) \times (M-K)} & I_{(K+L-M) \times (K+L-M)} \\ O_{(P-L) \times K} & O_{(P-L) \times (M-K)} & O_{(P-L) \times (K+L-M)} \end{bmatrix} \times \begin{bmatrix} O_{(K+L) \times (N-K-L)} & R_{(K+L) \times (K+L)} \end{bmatrix}$$

where U, V, and Q are orthogonal or unitary matrices, R is nonsingular upper triangular, $[C]^2 + [S]^2 = [I]$, and

$$C = \begin{bmatrix} \alpha_{K+1} & & & \\ & \alpha_{K+2} & & \\ & & \dots & \\ & & & \alpha_M \end{bmatrix} \quad \text{and} \quad S = \begin{bmatrix} \beta_{K+1} & & & \\ & \beta_{K+2} & & \\ & & \dots & \\ & & & \beta_M \end{bmatrix}$$

On exit, R(1:M, K+L) is stored in A(1:M, N-K-L+1:N) and R(M+1:K+L, M+1:K+L) is stored in B(M-K+1:L, N+M-K-L+1:N).

Calling Sequence

```
CALL DGGSDV (JOBV, JOBV, JOBQ, M, N, P, K, L, DA, LDA, DB, LDB,
             DALPHA, DBETA, DU, LDU, DV, LDV, DQ, LDQ, DWORK,
             IWORK3, INFO)

CALL SGGSDV (JOBV, JOBV, JOBQ, M, N, P, K, L, SA, LDA, SB, LDB,
             SALPHA, SBETA, SU, LDU, SV, LDV, SQ, LDQ, SWORK,
             IWORK3, INFO)

CALL ZGGSDV (JOBV, JOBV, JOBQ, M, N, P, K, L, ZA, LDA, ZB, LDB,
             DALPHA, DBETA, ZU, LDU, ZV, LDV, ZQ, LDQ, ZWORK,
             DWORK2, IWORK3, INFO)

CALL CGGSDV (JOBV, JOBV, JOBQ, M, N, P, K, L, CA, LDA, CB, LDB,
             SALPHA, SBETA, CU, LDU, CV, LDV, CQ, LDQ, CWORK,
             SWORK2, IWORK3, INFO)

void dggsvd (char jobv, char jobv, char jobq, int m, int n, int p,
            int *k, int *l, double *da, int lda, double *db, int
            ldb, double *dalp, double *dbet, double *du, int
            ldu, double *dv, int ldv, double *dq, int ldq, int
            *info)

void sggsvd (char jobv, char jobv, char jobq, int m, int n, int p,
            int *k, int *l, float *sa, int lda, float *sb, int ldb,
            float *salp, float *sbet, float *su, int ldu, float
            *sv, int ldv, float *q, int ldq, int *info)
```

```

void zggsvd (char jobu, char jobv, char jobq, int m, int n, int p,
             int *k, int *l, doublecomplex *za, int lda,
             doublecomplex *zb, int ldb, double *dalpha, double
             *dbeta, doublecomplex *zu, int ldu, doublecomplex *zv,
             int ldv, doublecomplex *zq, int ldq, int *info)
void cggsvd (char jobu, char jobv, char jobq, int m, int n, int p,
             int *k, int *l, complex *ca, int lda, complex *cb, int
             ldb, float *salpha, float *sbeta, complex *cu, int ldu,
             complex *cv, int ldv, complex *cq, int ldq, int *info)

```

Arguments

JOBQ	Indicates whether or not to compute the matrix Q. Legal values of JOBQ are shown below. Any values not shown below are illegal. 'Q' or 'q' Matrix Q is computed. 'N' or 'n' Matrix Q is not computed.
JOBV	Indicates whether or not to compute the matrix V. Legal values of JOBV are shown below. Any values not shown below are illegal. 'V' or 'v' Matrix V is computed. 'N' or 'n' Matrix V is not computed.
JOBQ	Indicates whether or not to compute the matrix Q. Legal values of JOBQ are shown below. Any values not shown below are illegal. 'Q' or 'q' Matrix Q is computed. 'N' or 'n' Matrix Q is not computed.
M	Number of rows of the matrix A. $M \geq 0$.
N	Number of columns of the matrices A and B. $N \geq 0$.
P	Number of rows of the matrix B. $P \geq 0$.
K, L	On exit, K and L specify the dimension of the sub-blocks described above. $K+L$ = effective numerical rank of $[A^T B^T]^T$.
xA	On entry, the $M \times N$ matrix A. On exit, the triangular matrix R, or part of R. If $M-K-L \geq 0$ then R is stored in $A(1:K+L, N-K-L+1:N)$. If $M-K-L < 0$ then $R(1:M, K+L)$ is stored in $A(1:M, N-K-L+1:N)$ and $R(M+1:K+L, M+1:K+L)$ is stored in $B(M-K+1:L, N+M-K-L+1:N)$.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, M)$.
xB	On entry, the $P \times N$ matrix B. On exit, the triangular matrix R if necessary. If $M-K-L < 0$ then $R(1:M, K+L)$ is stored in $A(1:M, N-K-L+1:N)$ and $R(M+1:K+L, M+1:K+L)$ is stored in $B(M-K+1:L, N+M-K-L+1:N)$.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, P)$.

xALPHA, xBETA

On exit, ALPHA and BETA contain the generalized singular value pairs of A and B:

if $M-K-L \geq 0$

ALPHA(1:K) = ONE ALPHA(K+1:K+L) = C

BETA(1:K) = ZERO BETA(K+1:K+L) = S

if $M-K-L < 0$

ALPHA(1:K) = ONE ALPHA(K+1:M) = C ALPHA(M+1:K+L) = ZERO

BETA(1:K) = ZERO BETA(K+1:M) = S BETA(M+1:K+L) = ONE

and

ALPHA(K+L+1:N) = ZERO

BETA(K+L+1:N) = ZERO

xU On exit, if JOBU = 'U' or 'u', then U contains the $M \times M$ matrix U; otherwise U is not used.

LDU Leading dimension of the array U as specified in a dimension or type statement. $LDU \geq \max(1, M)$.

xV On exit, if JOBV = 'V' or 'v', then V contains the $P \times P$ matrix V; otherwise V is not used.

LDV Leading dimension of the array V as specified in a dimension or type statement. $LDV \geq \max(1, P)$.

xQ On exit, if JOBQ = 'Q' or 'q', then Q contains the $N \times N$ matrix Q; otherwise Q is not used.

LDQ Leading dimension of the array Q as specified in a dimension or type statement. $LDQ \geq \max(1, N)$.

xWORK Scratch array with a dimension of $\max(M, 3 \times N, P) + N$.

xWORK2 Scratch array with a dimension of $2 \times N$ for complex subroutines.

IWORK3 Scratch array with a dimension of N.

INFO On exit:

INFO = 0 Subroutine completed normally.

INFO < 0 The i th argument, where $i = |\text{INFO}|$, had an illegal value.

INFO = 1 Convergence failed.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
INTEGER          LDA, LDB, LDIWRK, LDWORK, N
PARAMETER        (N = 3)
PARAMETER        (LDA = N)
PARAMETER        (LDB = N)
PARAMETER        (LDIWRK = N)

```



```

C      PARAMETER      (LDWORK = 5 * N)
C
C      DOUBLE PRECISION  A(LDA,N), ALPHA(N), BETA(N), B(LDB,N)
C      DOUBLE PRECISION  TEMP, WORK(LDWORK)
C      INTEGER           ICOL, INFO, IROW, IWORK(LDIWRK), K, L
C
C      EXTERNAL          DGGSDV
C      INTRINSIC          ABS
C
C      Initialize the array A to store the matrix A shown below.
C      Initialize the array B to store the matrix B shown below.
C
C      2   4   12       2   4   12
C      A = 0  12  20    B = 0   6  10
C      0   0  24       0   0   8
C
C      DATA A / 2.0D0, 0.0D0, 0.0D0, 4.0D0, 1.2D1, 0.0D0,
$             1.2D1, 2.0D1, 2.4D1 /
C      DATA B / 2.0D0, 0.0D0, 0.0D0, 4.0D0, 6.0D0, 0.0D0,
$             1.2D1, 1.0D1, 8.0D0 /
C
C      Print the initial values of A and B.
C
C      PRINT 1000
C      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
C      PRINT 1020
C      PRINT 1010, ((B(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
C
C      Compute the generalized singular values of (A,B)
C
C      CALL DGGSDV ('NO COMPUTE U', 'NO COMPUTE V', 'NO COMPUTE Q',
$                N, N, N, K, L, A, LDA, B, LDB, ALPHA, BETA,
$                TEMP, 1, TEMP, 1, TEMP, 1, WORK, IWORK, INFO)
C      IF (INFO .LT. 0) THEN
C        PRINT 1030, ABS(INFO)
C        STOP 1
C      ELSE IF (INFO .GT. 0) THEN
C        PRINT 1040
C        STOP 2
C      END IF
C      PRINT 1050
C      PRINT 1060, ALPHA
C      PRINT 1070
C      PRINT 1080, BETA
C      PRINT 1090
C      DO 100, ICOL = 1, N
C        PRINT 1100, (ALPHA(ICOL) / BETA(ICOL))
C      100 CONTINUE
C
C      1000 FORMAT (1X, 'A:')
C      1010 FORMAT (3(3X, F4.1))
C      1020 FORMAT (/1X, 'B:')
C      1030 FORMAT (1X, 'Illegal value for argument #', I1,
$                ' in DGGSDV.')
C      1040 FORMAT (1X, 'Singular value decomposition failed',
$                1X, 'to converge.')

```

```

1050 FORMAT (/1X, 'ALPHA:')
1060 FORMAT (1X, 10(F8.4))
1070 FORMAT (/1X, 'BETA:')
1080 FORMAT (1X, 10(F8.4))
1090 FORMAT (/1X, 'ALPHA(i) / BETA(i):')
1100 FORMAT (1X, F8.4)
C
      END

```

Sample Output

```

A:
  2.0    4.0   12.0
  0.0   12.0   20.0
  0.0    0.0   24.0

B:
  2.0    4.0   12.0
  0.0    6.0   10.0
  0.0    0.0    8.0

ALPHA:
  0.8944  0.9487  0.7071

BETA:
  0.4472  0.3162  0.7071

ALPHA(i) / BETA(i):
  2.0000
  3.0000
  1.0000

```

Reciprocal Condition Number of an LU-Factored Tridiagonal Matrix

The subroutines described in this section estimate the reciprocal condition number, in either the 1-norm or the ∞ -norm, of a tridiagonal matrix A, which has been LU-factored by xGTTRF.

Calling Sequence

```
CALL DGTCON (NORM, N, DLOW, DDIAG, DUP1, DUP2, IPIVOT, DANORM,
             DRCOND, DWORK, IWORK2, INFO)
CALL SGTCON (NORM, N, SLOW, SDIAG, SUP1, SUP2, IPIVOT, SANORM,
             SRCOND, SWORK, IWORK2, INFO)
CALL ZGTCON (NORM, N, ZLOW, ZDIAG, ZUP1, ZUP2, IPIVOT, DANORM,
             DRCOND, ZWORK, INFO)
CALL CGTCON (NORM, N, CLOW, CDIAG, CUP1, CUP2, IPIVOT, SANORM,
             SRCOND, CWORK, INFO)

void dgtcon (char norm, int n, double *dlow, double *ddiag, double
             *dup1, double *dup2, int *ipivot, double danorm, double
             *drcond, int *info)
void sgtcon (char norm, int n, float *slow, float *sdiag, float
             *sup1, float *sup2, int *ipivot, float sanorm, float
             *srcond, int *info)
void zgtcon (char norm, int n, doublecomplex *zlow, doublecomplex
             *zdiag, doublecomplex *zup1, doublecomplex *zup2, int
             *ipivot, double danorm, double *drcond, int *info)
void cgtcon (char norm, int n, complex *clow, complex *cdiag,
             complex *cup1, complex *cup2, int *ipivot, float
             sanorm, float *srcond, int *info)
```

Arguments

NORM	Indicates whether the 1-norm condition number or the ∞ -norm condition number is required. The legal values for NORM are listed below. Any values not listed below are illegal. 'O', 'o', or '1' 1-norm condition number is required. 'I' or 'i' ∞ -norm condition number is required.
N	Order of the matrix A. $N \geq 0$.
xLOW	The N-1 multipliers that define the matrix L from the LU factorization of A, as computed by xGTTRF.
xDIAG	The N diagonal elements of the upper triangular matrix U from the LU factorization of A, as computed by xGTTRF.

xUP1	The N-1 elements of the first superdiagonal of U from the LU factorization of A, as computed by xGTTRF.
xUP2	The N-2 elements of the second superdiagonal of U from the LU factorization of A, as computed by xGTTRF.
IPIVOT	Pivot indices as computed by xGTTRF.
xANORM	The 1-norm of matrix A.
xRCOND	On exit, the estimated reciprocal of the condition number of the matrix A where the reciprocal condition number of A is defined to be $1 / (ANORM \times \ A^{-1}\)$. The reciprocal of the condition number is estimated instead of the condition number itself to avoid overflow or division by zero. If RCOND is less than machine precision (in particular, if RCOND = 0) then A is singular to working precision.
xWORK	Scratch array with a dimension of $2 \times N$.
IWORK2	Scratch array with a dimension of N for real subroutines.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where $i = INFO $, had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
C      INTEGER          LDB, LDIWRK, LDWORK, N, NRHS
C      PARAMETER        (N = 5)
C      PARAMETER        (LDB = N)
C      PARAMETER        (LDIWRK = N)
C      PARAMETER        (LDWORK = 2 * N)
C      PARAMETER        (NRHS = 1)

C
C      DOUBLE PRECISION ANORM, B(LDB,NRHS), DIAG(N), DLOWER(N-1)
C      DOUBLE PRECISION DUP1(N-1), DUP2(N-2), RCOND, WORK(LDWORK)
C      INTEGER          ICOL, INFO, IPIVOT(N), IROW, IWORK(LDIWRK)

C
C      EXTERNAL          DGTCON, DGTTRF, DGTTRS
C      INTRINSIC          ABS

C
C      Initialize the arrays DLOWER, DIAG, and DUP1 to store
C      the first subdiagonal, the diagonal, and the first
C      superdiagonal of the coefficient matrix A shown below.
C      Initialize the array B to store the right hand side
C      matrix b shown below.
C
C
C      -3   1           6
C      1  -2   1           12
C      A =   1  -2   1           18
C           1  -2   1           12

```

```

C          1  -1      6
C
DATA DLOWER / 1.0D0, 1.0D0, 1.0D0, 1.0D0 /
DATA DIAG / -3.0D0, -2.0D0, -2.0D0, -2.0D0, -1.0D0 /
DATA DUP1 / 1.0D0, 1.0D0, 1.0D0, 1.0D0 /
DATA B / 6.0D0, 1.2D1, 1.8D1, 1.2D1, 6.0D0 /

C
PRINT 1000
DO 100, IROW = 1, N
  PRINT 1010, (0.0D0, ICOL = 1, IROW - 2),
$          (DLOWER(ICOL + 1), ICOL = ABS(IROW - 2), IROW - 2),
$          DIAG(IROW),
$          (DUP1(IROW), ICOL = 1, MIN(1, N - IROW)),
$          (0.0D0, ICOL = IROW + 2, N)
100 CONTINUE
PRINT 1020
PRINT 1030, B

C
CALL DGTTRF (N, DLOWER, DIAG, DUP1, DUP2, IPIVOT, INFO)
IF (INFO .NE. 0) THEN
  PRINT 1040, INFO
  STOP 1
END IF
ANORM = 4.0D0
CALL DGTCON ('ONE-NORM', N, DLOWER, DIAG, DUP1, DUP2,
$          IPIVOT, ANORM, RCOND, WORK, IWORK, INFO)
IF (INFO .NE. 0) THEN
  PRINT 1050, ABS (INFO)
  STOP 2
END IF
PRINT 1060, 1.0D0 / RCOND
CALL DGTTRS ('NO TRANSPOSE A', N, NRHS, DLOWER, DIAG, DUP1,
$          DUP2, IPIVOT, B, LDB, INFO)
IF (INFO .NE. 0) THEN
  PRINT 1070, INFO
  STOP 3
END IF
PRINT 1080
PRINT 1030, B

C
1000 FORMAT (1X, 'A:')
1010 FORMAT (5(3X, F5.2))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (1X, F8.2)
1040 FORMAT (1X, 'Error factoring A, INFO = ', I5)
1050 FORMAT (1X, 'Illegal argument to DGTCON, argument #', I2)
1060 FORMAT (/1X, 'Estimated condition number of A: ', F6.2)
1070 FORMAT (1X, 'Error solving Ax=b, INFO = ', I5)
1080 FORMAT (/1X, 'x:')
C
END

```

Sample Output

```
A:
-3.00    1.00    0.00    0.00    0.00
 1.00   -2.00    1.00    0.00    0.00
 0.00    1.00   -2.00    1.00    0.00
 0.00    0.00    1.00   -2.00    1.00
 0.00    0.00    0.00    1.00   -1.00

b:
 6.00
12.00
18.00
12.00
 6.00

Estimated condition number of A:  50.00

x:
-27.00
-75.00
-111.00
-129.00
-135.00
```

Refined Solution to a Linear System in an LU-Factored General Tridiagonal Matrix

The subroutines described in this section refine the solution to a tridiagonal linear system $AX = B$, $A^T X = B$, or $A^H X = B$ for a tridiagonal matrix A , which has been LU-factored by xGTTRE, and general matrices B and X . These subroutines also compute forward error bounds and backward error estimates on the solution.

Calling Sequence

```
CALL DGTRFS (TRANSA, N, NRHS, DLOW, DDIAG, DUP, DLOWF, DDIAGF,
             DUPF1, DUPF2, IPIVOT, DB, LDB, DX, LDX, DFERR, DBERR,
             DWORK, IWORK2, INFO)
CALL SGTRFS (TRANSA, N, NRHS, SLOW, SDIAG, SUP, SLOWF, SDIAGF,
             SUPF1, SUPF2, IPIVOT, SB, LDB, SX, LDX, SFERR, SBERR,
             SWORK, IWORK2, INFO)
CALL ZGTRFS (TRANSA, N, NRHS, ZLOW, ZDIAG, ZUP, ZLOWF, ZDIAGF,
             ZUPF1, ZUPF2, IPIVOT, ZB, LDB, ZX, LDX, DFERR, DBERR,
             ZWORK, DWORK2, INFO)
CALL CGTRFS (TRANSA, N, NRHS, CLOW, CDIAG, CUP, CLOWF, CDIAGF,
             CUPF1, CUPF2, IPIVOT, CB, LDB, CX, LDX, SFERR, SBERR,
             CWORK, SWORK2, INFO)

void dgtrfs (char trans, int n, int nrhs, double *dlow, double
             *ddiag, double *dup, double *dlowf, double *ddiagf,
             double *dupf1, double *dupf2, int *ipivot, double *db,
             int ldb, double *dx, int ldx, double *dferr, double
             *dberr, int *info)
void sgtrfs (char trans, int n, int nrhs, float *slow, float
             *sdiag, float *sup, float *slowf, float *sdiagf, float
             *supf1, float *supf2, int *ipivot, float *sb, int ldb,
             float *sx, int ldx, float *sferr, float *sberr, int
             *info)
void zgtrfs (char trans, int n, int nrhs, doublecomplex *zlow,
             doublecomplex *zdiag, doublecomplex *zup,
             doublecomplex *zlowf, doublecomplex *zdiagf,
             doublecomplex *zupf1, doublecomplex *zupf2, int
             *ipivot, doublecomplex *zb, int ldb, doublecomplex
             *zx, int ldx, double *dferr, double *dberr, int *info)
```

```
void cgtrfs (char trans, int n, int nrhs, complex *clow, complex
*cdiag, complex *cup, complex *clowf, complex *cdiagf,
complex *cupf1, complex *cupf2, int *ipivot, complex
*cb, int ldb, complex *cx, int ldx, float *sferr, float
*sberr, int *info)
```

Arguments

TRANSA	Indicates the form of the linear system whose solution is to be refined. The legal values for TRANSA are listed below. Any values not listed below are illegal. 'N' or 'n' No transpose, refine $AX = B$. 'T' or 't' Transpose, refine $A^T X = B$. 'C' or 'c' Conjugate transpose, refine $A^H X = B$. Note that A^T and A^H are the same for real matrices.
N	Order of the matrix A. $N \geq 0$.
NRHS	Number of right-hand sides, equal to the number of columns of the matrices B and X. $NRHS \geq 0$.
xLOW	The N-1 elements of the first subdiagonal of A.
xDIAG	The N diagonal elements of A.
xUP	The N-1 elements of the first superdiagonal A.
xLOWF	The N-1 multipliers that define the matrix L from the LU factorization of A, as computed by xGTTRF.
xDIAGF	The N diagonal elements of the upper triangular matrix U from the LU factorization of A, as computed by xGTTRF.
xUPF1	The N-1 elements of the first superdiagonal of U from the LU factorization of A, as computed by xGTTRF.
xUPF2	The N-2 elements of the second superdiagonal of U from the LU factorization of A, as computed by xGTTRF.
IPIVOT	Pivot indices as computed by xGTTRF.
xB	The $N \times NRHS$ right-hand side matrix B.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
xx	On entry, the $N \times NRHS$ solution matrix X. On exit, the refined $N \times NRHS$ solution matrix X.
LDX	Leading dimension of the array X as specified in a dimension or type statement. $LDX \geq \max(1, N)$.
xFERR	On exit, the estimated forward error bound for each solution vector $X(*, j)$ for $1 \leq j \leq NRHS$. If X' is the true solution corresponding to $X(*, j)$ then $FERR(j)$ is an upper bound on the magnitude of the largest element in $X(*, j) - X'$ divided by the magnitude of the largest element in $X(*, j)$.

xBERR	On exit, BERR(j) is the smallest relative change in any element of A or B(*, j) that makes X(*, j) an exact solution to AX(*, j) = B(*, j) for $1 \leq j \leq \text{NRHS}$.
xWORK	Scratch array with a dimension of $3 \times N$ for real subroutines or $2 \times N$ for complex subroutines.
xWORK2	Scratch array with a dimension of N.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The i-th argument, where $i = \text{INFO} $, had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
C   INTEGER          LDB, LDIWRK, LDWORK, LDX, N, NRHS
C   PARAMETER        (N = 4)
C   PARAMETER        (LDB = N)
C   PARAMETER        (LDIWRK = N)
C   PARAMETER        (LDWORK = 3 * N)
C   PARAMETER        (LDX = N)
C   PARAMETER        (NRHS = 1)

C
C   DOUBLE PRECISION ANORM, B(LDB,NRHS), BERR(NRHS), DIAG(N)
C   DOUBLE PRECISION DIAGF(N), DLOW(N-1), DLOWF(N-1), DUP1(N-1)
C   DOUBLE PRECISION DUP1F(N-1), DUP2F(N-2), EPSLON, FERR(NRHS)
C   DOUBLE PRECISION RCOND, WORK(LDWORK), X(LDX,NRHS)
C   INTEGER          ICOL, INFO, IPIVOT(N), IROW, IWORK(LDIWRK)

C
C   EXTERNAL          DCOPY, DGTCON, DGTRFS, DGTTRF, DGTRRS
C   INTRINSIC         ABS, MAX

C
C   Initialize the arrays DLOW, DIAG, and DUP1 to store
C   the first subdiagonal, the diagonal, and the first
C   superdiagonal of the coefficient matrix A shown below.
C   Initialize the array B to store the right hand side
C   matrix b shown below.
C
C
C   A = 0      3      7      b = 5
C           0      0      5
C           0.1    0      3      5
C           0      0      0      5
C
C   DATA DLOW / 0.0D0, 1.0D-1, 0.0D0 /
C   DATA DIAG / 0.0D0, 0.0D0, 0.0D0, 0.0D0 /
C   DATA DUP1 / 3.0D0, 7.0D0, 3.0D0 /
C   DATA B / 5.0D0, 5.0D0, 5.0D0, 5.0D0 /

C
C   Add a small value to each of the elements on the diagonal
C   and the first sub- and super-diagonal of A. After this

```

```

C      loop, A will resemble the matrix shown below. Print A
C      after adding epsilon.
C
C      e      3+e
C      A = e      e      7+e
C           0.1+e      e      3+e
C           e      e
C
C      EPSLON = (((((2.0D0 / 3.0D0) + 8.0D0) - 8.0D0) -
$      (2.0D0 / 3.0D0))
DO 100, IROW = 1, N - 1
    DLOW(IROW) = DLOW(IROW) + EPSLON
    DIAG(IROW) = DIAG(IROW) + EPSLON
    DUP1(IROW) = DUP1(IROW) - EPSLON
100 CONTINUE
    DIAG(N) = DIAG(N) + EPSLON
    CALL DCOPY (N - 1, DLOW, 1, DLOWF, 1)
    CALL DCOPY (N, DIAG, 1, DIAGF, 1)
    CALL DCOPY (N - 1, DUP1, 1, DUP1F, 1)
C
C      PRINT 1000
DO 110, IROW = 1, N
    PRINT 1010, (0.0D0, ICOL = 1, IROW - 2),
$      (DLOW(ICOL + 1), ICOL = ABS(IROW - 2), IROW - 2),
$      DIAG(IROW),
$      (DUP1(IROW), ICOL = 1, MIN(1, N - IROW)),
$      (0.0D0, ICOL = IROW + 2, N)
110 CONTINUE
C
C      Add a small value to each element of B. After this loop, B
C      will resemble the matrix shown below. Print B after adding
C      epsilon.
C
C      5+e
C      B = 5+e
C      5+e
C      5+e
C
C      DO 130, ICOL = 1, NRHS
C      DO 120, IROW = 1, N
C      B(IROW,ICOL) = B(IROW,ICOL) + EPSLON
120 CONTINUE
130 CONTINUE
    CALL DCOPY (N, B, 1, X, 1)
    PRINT 1020
    PRINT 1030, B
C
C      LU factor A.
C
C      CALL DGTTRF (N, DLOWF, DIAGF, DUP1F, DUP2F, IPIVOT, INFO)
C      IF (INFO .NE. 0) THEN
C      PRINT 1040, INFO
C      STOP 1
C      END IF
C
C      Estimate the condition number of A.

```

```

C
  ANORM = 9.25D0 + (3.0D0 * EPSLON)
  CALL DGTCON ('ONE-NORM', N, DLOWF, DIAGF, DUP1F, DUP2F,
$             IPIVOT, ANORM, RCOND, WORK, IWORK, INFO)
  IF (INFO .NE. 0) THEN
    PRINT 1050, ABS (INFO)
    STOP 2
  END IF
  PRINT 1060, 1.0D0 / RCOND
C
C   Solve Ax=b and print the solution.
C
  CALL DGTTRS ('NO TRANSPOSE A', N, NRHS, DLOWF, DIAGF,
$             DUP1F, DUP2F, IPIVOT, X, LDX, INFO)
  IF (INFO .NE. 0) THEN
    PRINT 1070, INFO
    STOP 3
  END IF
  PRINT 1080
  PRINT 1090, X
  PRINT 1100
  PRINT 1030,                                DIAG(1) * X(1,1) + DUP1(1) *
$   X(2,1)
  PRINT 1030, DLOW(1) * X(1,1) + DIAG(2) * X(2,1) + DUP1(2) *
$   X(3,1)
  PRINT 1030, DLOW(2) * X(2,1) + DIAG(3) * X(3,1) + DUP1(3) *
$   X(4,1)
  PRINT 1030, DLOW(3) * X(3,1) + DIAG(4) * X(4,1)
C
C   Refine the solution to Ax=b and print the refined solution.
C
  CALL DGTFRS ('NO TRANSPOSE A', N, NRHS, DLOW, DIAG, DUP1,
$             DLOWF, DIAGF, DUP1F, DUP2F, IPIVOT, B, LDB,
$             X, LDX, FERR, BERR, WORK, IWORK, INFO)
  IF (INFO .NE. 0) THEN
    PRINT 1110, ABS(INFO)
    STOP 4
  END IF
  PRINT 1120
  PRINT 1090, X
  PRINT 1130
  PRINT 1030,                                DIAG(1) * X(1,1) + DUP1(1) *
$   X(2,1)
  PRINT 1030, DLOW(1) * X(1,1) + DIAG(2) * X(2,1) + DUP1(2) *
$   X(3,1)
  PRINT 1030, DLOW(2) * X(2,1) + DIAG(3) * X(3,1) + DUP1(3) *
$   X(4,1)
  PRINT 1030, DLOW(3) * X(3,1) + DIAG(4) * X(4,1)
  PRINT 1140, BERR(1)
  PRINT 1150, FERR(1)
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (4(2X, F18.16))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (1X, F21.17)
1040 FORMAT (1X, 'Error factoring A, INFO = ', I5)

```

```

1050 FORMAT (1X, 'Illegal argument to DGTCON, argument #', I2)
1060 FORMAT (/1X, 'Estimated condition number of A: ', E12.6)
1070 FORMAT (1X, 'Error solving Ax=b, INFO = ', I5)
1080 FORMAT (/1X, 'Initial solution to Ax=b:')
1090 FORMAT (1X, E25.17)
1100 FORMAT (/1X, 'Ax with the initial x:')
1110 FORMAT (1X, 'Illegal argument to DGTRFS, INFO = ', I2)
1120 FORMAT (/1X, 'Refined solution to Ax=b:')
1130 FORMAT (/1X, 'Ax with refined x:')
1140 FORMAT (/1X, 'Forward error: ', E12.4)
1150 FORMAT (1X, 'Backward error: ', E12.4)
C
      END

```

Sample Output

```

A:
-.000000000000000006  3.000000000000000004  0.000000000000000000  0.000000000000000000
-.000000000000000006  -.000000000000000006  7.000000000000000009  0.000000000000000000
0.000000000000000000  0.099999999999999995  -.000000000000000006  3.000000000000000004
0.000000000000000000  0.000000000000000000  -.000000000000000006  -.000000000000000006

b:
4.999999999999999911
4.999999999999999911
4.999999999999999911
4.999999999999999911

Estimated condition number of A: 0.227847E+33

Initial solution to Ax=b:
-0.12316065590651126E+33
-0.22789299319224172E+17
-0.97668425653817900E+16
0.75964331064080138E+15

Ax with the initial x:
16.000000000000000000
24.000000000000000000
5.500000000000000000
4.99999999999999822

Refined solution to Ax=b:
-0.12316065590651124E+33
-0.22789299319224172E+17
-0.97668425653817920E+16
0.75964331064080125E+15

```

Ax with refined x:

```
0.000000000000000000
-8.000000000000000000
5.000000000000000000
4.999999999999999911
```

Forward error: 0.1170E-15
Backward error: 0.2384E-14

Solution to a Linear System in a General Tridiagonal Matrix (Simple Driver)

The subroutines described in this section solve a linear system $AX = B$ for a general tridiagonal matrix A and general matrices B and X. Note that the expert driver xGTSVX is also available.

Calling Sequence

```
CALL DGTSV (N, NRHS, DLOW, DDIA, DUP, DB, LDB, INFO)
CALL SGTSV (N, NRHS, SLOW, SDIA, SUP, SB, LDB, INFO)
CALL ZGTSV (N, NRHS, ZLOW, ZDIA, ZUP, ZB, LDB, INFO)
CALL CGTSV (N, NRHS, CLOW, CDIA, CUP, CB, LDB, INFO)

void dgtsv (int n, int nrhs, double *dlow, double *ddiag, double
            *dup, double *db, int ldb, int *info)
void sgtsv (int n, int nrhs, float *slow, float *sdiag, float
            *sup, float *sb, int ldb, int *info)
void zgtsv (int n, int nrhs, doublecomplex *zlow, doublecomplex
            *zdiag, doublecomplex *zup, doublecomplex *zb, int
            ldb, int *info)
void cgtsv (int n, int nrhs, complex *clow, complex *cdiag,
            complex *cup, complex *cb, int ldb, int *info)
```

Arguments

N	Order of the matrix A. $N \geq 0$.
NRHS	Number of right-hand sides, equal to the number of columns of the matrix B. $NRHS \geq 0$.
xLOW	On entry, the N-1 subdiagonal elements of A. On exit, LOW is overwritten by the N-2 elements of the second superdiagonal of the upper triangular matrix U from the LU factorization of A, stored in elements 1 through N-2.
xDIA	On entry, the N diagonal elements of A. On exit, DIA is overwritten by the N diagonal elements of U.
xUP	On entry, the N-1 superdiagonal elements of A. On exit, UP is overwritten by the N-1 elements of the first superdiagonal of U.
xB	On entry, the N×NRHS right-hand side matrix B. On exit, the N×NRHS solution matrix X.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.

INFO	On exit:	
	INFO = 0	Subroutine completed normally.
	INFO < 0	The ith argument, where i = INFO , had an illegal value.
	INFO > 0	U(i,i), where i = INFO, is exactly zero, and U is therefore singular. The factorization has not been completed, unless i = N, and the solution could not be computed.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
C   INTEGER          LDB, LDIWRK, LDWORK, N, NRHS
C   PARAMETER        (N = 4)
C   PARAMETER        (LDB = N)
C   PARAMETER        (LDIWRK = N)
C   PARAMETER        (LDWORK = N)
C   PARAMETER        (NRHS = 1)

C
C   DOUBLE PRECISION B(LDB,NRHS), DIAG(N), DLOWER(N-1)
C   DOUBLE PRECISION DUPPER(N-1)
C   INTEGER          ICOL, INFO, IROW

C
C   EXTERNAL          DGTSV
C   INTRINSIC         ABS, MAX

C
C   Initialize the arrays DLOWER, DIAG, and DUPPER to store the
C   first subdiagonal, the diagonal, and the first superdiagonal
C   of the coefficient matrix A shown below. Initialize the
C   array B to store the right hand side matrix b shown below.
C
C
C   2   -1
C   A = 1   2   -1   b =  0
C           1   2   -1   2
C           1   2   11
C
C   DATA DLOWER / 1.0D0, 1.0D0, 1.0D0 /
C   DATA DIAG / 2.0D0, 2.0D0, 2.0D0, 2.0D0 /
C   DATA DUPPER / -1.0D0, -1.0D0, -1.0D0 /
C   DATA B / 0.0D0, 2.0D0, 4.0D0, 1.1D1 /

C
C   PRINT 1000
C   DO 100, IROW = 1, N
C     PRINT 1010, (0.0D0, ICOL = 1, IROW - 2),
$     (DLOWER(ICOL + 1), ICOL = ABS(IROW - 2), IROW - 2),
$     DIAG(IROW),
$     (DUPPER(IROW), ICOL = 1, MIN(1, N - IROW)),
$     (0.0D0, ICOL = IROW + 2, N)

```

```

100 CONTINUE
   PRINT 1020
   PRINT 1030, B
C
   CALL DGTSV (N, NRHS, DLOWER, DIAG, DUPPER, B, LDB, INFO)
   IF (INFO .LT. 0) THEN
     PRINT 1040, INFO
     STOP 1
   ELSE IF (INFO .GT. 0) THEN
     PRINT 1050
     STOP 2
   END IF
   PRINT 1060
   PRINT 1030, B
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (5(3X, F5.2))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (1X, F8.2)
1040 FORMAT (1X, 'Illegal argument to DGTSV, argument #', I2)
1050 FORMAT (1X, 'A is singular to working precision.')
1060 FORMAT (/1X, 'x:')
C
   END

```

Sample Output

```

A:
  2.00   -1.00    0.00    0.00
  1.00    2.00   -1.00    0.00
  0.00    1.00    2.00   -1.00
  0.00    0.00    1.00    2.00

b:
  0.00
  2.00
  4.00
 11.00

x:
  1.00
  2.00
  3.00
  4.00

```


Solution to a Linear System in a General Tridiagonal Matrix (Expert Driver)

The subroutines described in this section solve a linear system $AX = B$, $A^T X = B$, or $A^H X = B$ for a general tridiagonal matrix A and general matrices B and X . These subroutines also compute forward error bounds and backward error estimates for the solution, and estimate the condition number of the matrix A . Note that the simple driver `xGTSV` is also available.

Calling Sequence

```
CALL DGTSVX (FACT, TRANSA, N, NRHS, DLOW, DDIAG, DUP, DLOWF,
             DDIAGF, DUPF1, DUPF2, IPIVOT, DB, LDB, DX, LDX, DRCOND,
             DFERR, DBERR, DWORK, IWORK2, INFO)
CALL SGTSVX (FACT, TRANSA, N, NRHS, SLOW, SDIAG, SUP, SLOWF,
             SDIAGF, SUPF1, SUPF2, IPIVOT, SB, LDB, SX, LDX, SRCOND,
             SFERR, SBERR, SWORK, IWORK2, INFO)
CALL ZGTSVX (FACT, TRANSA, N, NRHS, ZLOW, ZDIAG, ZUP, ZLOWF,
             ZDIAGF, ZUPF1, ZUPF2, IPIVOT, ZB, LDB, ZX, LDX, DRCOND,
             DFERR, DBERR, ZWORK, DWORK2, INFO)
CALL CGTSVX (FACT, TRANSA, N, NRHS, CLOW, CDIAG, CUP, CLOWF,
             CDIAGF, CUPF1, CUPF2, IPIVOT, CB, LDB, CX, LDX, SRCOND,
             SFERR, SBERR, CWORK, SWORK2, INFO)

void dgtsvx (char fact, char trans, int n, int nrhs, double *dlow,
             double *ddiag, double *dup, double *dlowf, double
             *ddiagf, double *dupf1, double *dupf2, int *ipivot,
             double *db, int ldb, double *dx, int ldx, double
             *drcond, double *dferr, double *dberr, int *info)
void sgtsvx (char fact, char trans, int n, int nrhs, float *slow,
             float *sdiag, float *sup, float *slowf, float *sdiagf,
             float *supf1, float *supf2, int *ipivot, float *sb, int
             ldb, float *sx, int ldx, float *srcond, float *sferr,
             float *sberr, int *info)
void zgtsvx (char fact, char trans, int n, int nrhs, doublecomplex
             *zlow, doublecomplex *zdiag, doublecomplex *zup,
             doublecomplex *zlowf, doublecomplex *zdiagf,
             doublecomplex *zupf1, doublecomplex *zupf2, int
             *ipivot, doublecomplex *zb, int ldb, doublecomplex
             *zx, int ldx, double *drcond, double *dferr, double
             *dberr, int *info)
```

```
void cgtsvx (char fact, char trans, int n, int nrhs, complex *clow,
             complex *cdiag, complex *cup, complex *clowf, complex
             *cdiag, complex *cupf1, complex *cupf2, int *ipivot,
             complex *cb, int ldb, complex *cx, int ldx, float
             *srcond, float *sferr, float *sberr, int *info)
```

Arguments

FACT	Indicates whether or not the factored form of A has been supplied on entry. 'F' or 'f' On entry, LOWF, DIAGF, UPF, UPF2, and IPIVOT contain the factored form of A. These arguments, as well as LOW, DIAG, and UP, will not be modified. 'N' or 'n' The matrix A will be copied to LOWF, DIAGF, and UPF and factored.
TRANS	Indicates the form of the linear system to solve. The legal values for TRANS are listed below. Any values not listed below are illegal. 'N' or 'n' No transpose, solve $AX = B$. 'T' or 't' Transpose, solve $A^T X = B$. 'C' or 'c' Conjugate transpose, solve $A^H X = B$. Note that A^T and A^H are the same for real matrices.
N	Order of the matrix A. $N \geq 0$.
NRHS	Number of right-hand sides, equal to the number of columns of the matrix B. $NRHS \geq 0$.
xLOW	The N-1 subdiagonal elements of A.
xDIAG	The N diagonal elements of A.
xUP	The N-1 superdiagonal elements of A.
xLOWF	On entry, if FACT = 'F' or 'f', then LOWF contains the N-1 multipliers that define the matrix L from the LU factorization of A, as computed by xGTTRE. On exit, if FACT = 'N', then LOWF contains the N-1 multipliers that define the matrix L from the LU factorization of A, as computed by xGTTRE.
xDIAGF	On entry, if FACT = 'F' or 'f', then DIAGF contains the N diagonal elements of the upper triangular matrix U from the LU factorization of A as computed by xGTTRE. On exit, if FACT = 'N', then DIAGF contains the N diagonal elements of the upper triangular matrix U from the LU factorization of A, as computed by xGTTRE.
xUPF1	On entry, if FACT = 'F' or 'f', then UPF1 contains the N-1 elements of the first superdiagonal of U. On exit, if FACT = 'N', then UPF1 contains the N-1 elements of the first superdiagonal of U.

xUPF2	On entry, if FACT = 'F' or 'f', then UPF2 contains the N-2 elements of the second superdiagonal of U. On exit, if FACT = 'N', then UPF2 contains the N-2 elements of the second superdiagonal of U.
IPIVOT	On entry, if FACT = 'F' or 'f', then IPIVOT contains the pivot indices from the LU factorization of A, as computed by xGTTRF. On exit, if FACT = 'N', then IPIVOT contains the pivot indices from the LU factorization of A, as computed by xGTTRF.
xB	The N×NRHS right-hand side matrix B.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
xX	On exit, the N×NRHS solution matrix X.
LDX	Leading dimension of the array X as specified in a dimension or type statement. $LDX \geq \max(1, N)$.
xRCOND	On exit, an estimate of the reciprocal condition number of the matrix A, where the reciprocal condition number is defined to be $1 / (\ A\ \times \ A^{-1}\)$. The reciprocal of the condition number is estimated instead of the condition number itself to avoid overflow or division by zero. If RCOND is less than machine precision (in particular, if RCOND = 0) then A is singular to working precision. In this case, INFO > 0 is returned and the solution and error bounds are not computed.
xFERR	On exit, the estimated forward error bound for each solution vector X(*, j) for $1 \leq j \leq \text{NRHS}$. If X' is the true solution corresponding to X(*, j) then FERR(j) is an upper bound on the magnitude of the largest element in X(*, j) - X' divided by the magnitude of the largest element in X(*, j).
xBERR	On exit, BERR(j) is the smallest relative change in any element of A or B(*, j) that makes X(*, j) an exact solution to AX(*, j) = B(*, j) for $1 \leq j \leq \text{NRHS}$.
xWORK	Scratch array with a dimension of $3 \times N$ for real subroutines or $2 \times N$ for complex subroutines.
xWORK2	Scratch array with a dimension of N for complex subroutines.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The i th argument, where $i = \text{INFO} $, had an illegal value. $1 \leq \text{INFO} \leq N$ U(i,i), where $i = \text{INFO}$, is exactly zero, U is therefore singular. The factorization has not been completed, unless $i = N$, and the solution and error bounds could not be computed.

INFO = N + 1 RCOND is less than machine precision. The factorization has been completed, but the matrix is singular to working precision, and the solution and error bounds could not be computed.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER          LDB, LDIWRK, LDWORK, LDX, N, NRHS
  PARAMETER        (N = 4)
  PARAMETER        (LDB = N)
  PARAMETER        (LDIWRK = N)
  PARAMETER        (LDWORK = 3 * N)
  PARAMETER        (LDX = N)
  PARAMETER        (NRHS = 1)

C
  DOUBLE PRECISION B(LDB,NRHS), BERR(NRHS), DIAG(N), DIAGF(N)
  DOUBLE PRECISION DLOW(N-1), DLOWF(N-1), DUP(N-1), DUPF(N-1)
  DOUBLE PRECISION DUP2F(N-1), FERR(NRHS), RCOND
  DOUBLE PRECISION WORK(LDWORK), X(LDX,NRHS)
  INTEGER          ICOL, INFO, IPIVOT(N), IROW, IWORK(LDIWRK)

C
  EXTERNAL          DGTSVX
  INTRINSIC         ABS, MAX

C
C   Initialize the arrays DLOWER, DIAG, and DUPPER to store
C   the first subdiagonal, the diagonal, and the first
C   superdiagonal of the coefficient matrix A shown below.
C   Initialize the array B to store the right hand side
C   matrix b shown below.
C
C
C   A = 1   2   -1   0
C         1   2   -1   b = 2
C           1   2   -1   4
C            1   2    11
C
C   DATA DLOW / 1.0D0, 1.0D0, 1.0D0 /
C   DATA DIAG / 2.0D0, 2.0D0, 2.0D0, 2.0D0 /
C   DATA DUP  / -1.0D0, -1.0D0, -1.0D0 /
C   DATA B    / 0.0D0, 2.0D0, 4.0D0, 1.1D1 /

C
C   Print the initial value of the arrays.
C
C
  PRINT 1000
  DO 100, IROW = 1, N
    PRINT 1010, (0.0D0, ICOL = 1, IROW - 2),
$      (DLOW(ICOL + 1), ICOL = ABS(IROW - 2), IROW - 2),
$      DIAG(IROW),
$      (DUP(IROW), ICOL = 1, MIN(1, N - IROW)),

```

```

$          (0.0D0, ICOL = IROW + 2, N)
100 CONTINUE
   PRINT 1020
   PRINT 1030, B
C
C      Solve the system Ax=b.  Print the solution and error
C      information.
C
      CALL DGTSVX ('NOT FACTORED ON ENTRY', 'NO TRANSPOSE A', N,
$              NRHS, DLOW, DIAG, DUP, DLOWF, DIAGF, DUPF,
$              DUP2F, IPIVOT, B, LDB, X, LDX, RCOND, FERR,
$              BERR, WORK, IWORK, INFO)
      IF (INFO .LT. 0) THEN
         PRINT 1040, INFO
         STOP 1
      ELSE IF (INFO .GT. 0) THEN
         PRINT 1050
         STOP 2
      END IF
      PRINT 1060
      PRINT 1030, X
      PRINT 1070, BERR(1)
      PRINT 1080, FERR(1)
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (5(3X, F5.2))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (1X, F8.2)
1040 FORMAT (1X, 'Illegal argument to DGTSVX, argument #', I2)
1050 FORMAT (1X, 'A is singular to working precision.')
1060 FORMAT (/1X, 'x:')
1070 FORMAT (/1X, 'Backward error: ', E14.8)
1080 FORMAT (1X, 'Forward error: ', E14.8)
C
      END
```

Sample Output

```
A:
  2.00   -1.00    0.00    0.00
  1.00    2.00   -1.00    0.00
  0.00    1.00    2.00   -1.00
  0.00    0.00    1.00    2.00

b:
  0.00
  2.00
  4.00
 11.00

x:
  1.00
```

2.00
3.00
4.00

Backward error: 0.00000000E+00
Forward error: 0.12174170E-14

LU Factorization of a Tridiagonal Matrix

The subroutines described in this section compute an LU factorization of a tridiagonal matrix A. It is typical to follow a call to xGTTRF with a call to xGTSVX or xGTTTS to solve a linear system $AX = B$, or to xGTCON to estimate the condition number of A.

Calling Sequence

```
CALL DGTTRF (N, DLOW, DDIAG, DUP1, DUP2, IPIVOT, INFO)
CALL SGTTRF (N, SLOW, SDIAG, SUP1, SUP2, IPIVOT, INFO)
CALL ZGTTRF (N, ZLOW, ZDIAG, ZUP1, ZUP2, IPIVOT, INFO)
CALL CGTTRF (N, CLOW, CDIAG, CUP1, CUP2, IPIVOT, INFO)

void dgttrf (int n, double *dlow, double *ddiag, double *dup1,
            double *dup2, int *ipivot, int *info)
void sgttrf (int n, float *slow, float *sdiag, float *sup1, float
            *sup2, int *ipivot, int *info)
void zgttrf (int n, doublecomplex *zlow, doublecomplex *zdiag,
            doublecomplex *zup1, doublecomplex *zup2, int *ipivot,
            int *info)
void cgttrf (int n, complex *clow, complex *cdiag, complex *cup1,
            complex *cup2, int *ipivot, int *info)
```

Arguments

N	Order of the matrix A. $N \geq 0$.
xLOW	On entry, the $N-1$ subdiagonal elements of A. On exit, the $N-1$ multipliers that define the matrix L from the LU factorization of A.
xDIAG	On entry, the N diagonal elements of A. On exit, the N diagonal elements of the upper triangular matrix U from the LU factorization of A.
xUP1	On entry, the $N-1$ superdiagonal elements of A. On exit, the $N-1$ elements of the first superdiagonal of U.
xUP2	On exit, the $N-2$ elements of the second superdiagonal of U.
IPIVOT	On exit, the pivot indices. During factorization, row i was exchanged with row IPIVOT(i) for $1 \leq i \leq N$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The i th argument, where $i = INFO $, had an illegal value.

INFO > 0 U(i,i), where i = INFO, is exactly zero and U is therefore singular. The factorization has been completed, but division by zero will occur if this factorization is used to solve a linear system.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
  INTEGER          LDB, LDIWRK, LDWORK, N, NRHS
  PARAMETER        (N = 5)
  PARAMETER        (LDB = N)
  PARAMETER        (LDIWRK = N)
  PARAMETER        (LDWORK = 2 * N)
  PARAMETER        (NRHS = 1)
C
  DOUBLE PRECISION ANORM, B(LDB,NRHS), DIAG(N), DLOWER(N-1)
  DOUBLE PRECISION DUP1(N-1), DUP2(N-2), RCOND, WORK(LDWORK)
  INTEGER          ICOL, INFO, IPIVOT(N), IROW, IWORK(LDIWRK)
C
  EXTERNAL          DGTCN, DGTTRF, DGTTRS
  INTRINSIC         ABS, MAX
C
  C Initialize the arrays DLOWER, DIAG, and DUP1 to store
  C the first subdiagonal, the diagonal, and the first
  C superdiagonal of the coefficient matrix A shown below.
  C Initialize the array B to store the right hand side
  C matrix b shown below.
C
C
C      -3   1               6
C      1  -2   1               12
C  A =      1  -2   1      b = 18
C              1  -2   1      12
C              1  -1      6
C
C
C      DATA DLOWER / 1.0D0, 1.0D0, 1.0D0, 1.0D0 /
C      DATA DIAG / -3.0D0, -2.0D0, -2.0D0, -2.0D0, -1.0D0 /
C      DATA DUP1 / 1.0D0, 1.0D0, 1.0D0, 1.0D0 /
C      DATA B / 6.0D0, 1.2D1, 1.8D1, 1.2D1, 6.0D0 /
C
C
C      Print the initial value of A.
C
C
C      PRINT 1000
C      DO 100, IROW = 1, N
C        PRINT 1010, (0.0D0, ICOL = 1, IROW - 2),
C          $      (DLOWER(ICOL + 1), ICOL = ABS(IROW - 2), IROW - 2),
C          $      DIAG(IROW),
C          $      (DUP1(IROW), ICOL = 1, MIN(1, N - IROW)),
C          $      (0.0D0, ICOL = IROW + 2, N)
C      100 CONTINUE

```



```
      PRINT 1020
      PRINT 1030, B
C
C      LU factor A.
C
      CALL DGTTRF (N, DLOWER, DIAG, DUP1, DUP2, IPIVOT, INFO)
      IF (INFO .NE. 0) THEN
        PRINT 1040, INFO
        STOP 1
      END IF
      ANORM = 4.0D0
C
C      Estimate and print the condition number of A.
C
      CALL DGTCN ('ONE-NORM', N, DLOWER, DIAG, DUP1, DUP2,
$      IPIVOT, ANORM, RCOND, WORK, IWORK, INFO)
      IF (INFO .NE. 0) THEN
        PRINT 1050, ABS (INFO)
        STOP 2
      END IF
      PRINT 1060, 1.0D0 / RCOND
C
C      Solve the system and print the solution.
C
      CALL DGTTRS ('NO TRANSPOSE A', N, NRHS, DLOWER, DIAG, DUP1,
$      DUP2, IPIVOT, B, LDB, INFO)
      IF (INFO .NE. 0) THEN
        PRINT 1070, INFO
        STOP 3
      END IF
      PRINT 1080
      PRINT 1030, B
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (5(3X, F5.2))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (1X, F8.2)
1040 FORMAT (1X, 'Error factoring A, INFO = ', I5)
1050 FORMAT (1X, 'Illegal argument to DGTCN, argument #', I2)
1060 FORMAT (/1X, 'Estimated condition number of A: ', F6.2)
1070 FORMAT (1X, 'Error solving Ax=b, INFO = ', I5)
1080 FORMAT (/1X, 'x:')
C
      END
```

Sample Output

A:				
-3.00	1.00	0.00	0.00	0.00
1.00	-2.00	1.00	0.00	0.00
0.00	1.00	-2.00	1.00	0.00
0.00	0.00	1.00	-2.00	1.00

```
0.00  0.00  0.00  1.00 -1.00
b:
  6.00
 12.00
 18.00
 12.00
  6.00
Estimated condition number of A: 50.00
x:
-27.00
-75.00
-111.00
-129.00
-135.00
```

Solution to a Linear System in an LU-Factored General Tridiagonal Matrix

The subroutines described in this section solve a linear system $AX = B$, $A^T X = B$, or $A^H X = B$ for a tridiagonal matrix A , which has been LU-factored by xGTTRE, and general matrices B and X .

Calling Sequence

```
CALL DGTTRS (TRANSA, N, NRHS, DLOW, DDIAG, DUP1, DUP2, IPIVOT, DB,
            LDB, INFO)
CALL SGTTRS (TRANSA, N, NRHS, SLOW, SDIAG, SUP1, SUP2, IPIVOT, SB,
            LDB, INFO)
CALL ZGTTRS (TRANSA, N, NRHS, ZLOW, ZDIAG, ZUP1, ZUP2, IPIVOT, ZB,
            LDB, INFO)
CALL CGTTRS (TRANSA, N, NRHS, CLOW, CDIAG, CUP1, CUP2, IPIVOT, CB,
            LDB, INFO)

void dgttrs (char trans, int n, int nrhs, double *dlow, double
            *ddiag, double *dup1, double *dup2, int *ipivot, double
            *db, int ldb, int *info)
void sgttrs (char trans, int n, int nrhs, float *dlow, float
            *ddiag, float *dup1, float *dup2, int *ipivot, float
            *sb, int ldb, int *info)
void zgttrs (char trans, int n, int nrhs, doublecomplex *zlow,
            doublecomplex *zdiag, doublecomplex *zup1,
            doublecomplex *zup2, int *ipivot, doublecomplex *zb,
            int ldb, int *info)
void cgttrs (char trans, int n, int nrhs, complex *clow, complex
            *cdiag, complex *cup1, complex *cup2, int *ipivot,
            complex *cb, int ldb, int *info)
```

Arguments

TRANSA	Indicates the form of the linear system to solve. The legal values for TRANSA are listed below. Any values not listed below are illegal. 'N' or 'n' No transpose, solve $AX = B$. 'T' or 't' Transpose, solve $A^T X = B$. 'C' or 'c' Conjugate transpose, solve $A^H X = B$. Note that $A^T = A^H$ for real matrices.
N	Order of the matrix A . $N \geq 0$.
NRHS	Number of right-hand sides, equal to the number of columns of the matrix B . $NRHS \geq 0$.

xLOW	The N-1 multipliers that define the matrix L from the LU factorization of A, as computed by xGTTRF.
xDIAG	The N diagonal elements of the upper triangular matrix U from the LU factorization of A, as computed by xGTTRF.
xUP1	The N-1 elements of the first superdiagonal of U from the LU factorization of A, as computed by xGTTRF.
xUP2	The N-2 elements of the second superdiagonal of U from the LU factorization of A, as computed by xGTTRF.
IPIVOT	Pivot indices as computed by xGTTRF.
xB	On entry, the N×NRHS right-hand side matrix B. On exit, the N×NRHS solution matrix X.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where i = INFO , had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
C      INTEGER          LDB, LDIWRK, LDWORK, N, NRHS
C      PARAMETER        (N = 5)
C      PARAMETER        (LDB = N)
C      PARAMETER        (LDIWRK = N)
C      PARAMETER        (LDWORK = 2 * N)
C      PARAMETER        (NRHS = 1)

C
C      DOUBLE PRECISION  ANORM, B(LDB,NRHS), DIAG(N), DLOWER(N-1)
C      DOUBLE PRECISION  DUP1(N-1), DUP2(N-2), RCOND, WORK(LDWORK)
C      INTEGER           ICOL, INFO, IPIVOT(N), IROW, IWORK(LDIWRK)

C
C      EXTERNAL          DGTCON, DGTTRF, DGTTRS
C      INTRINSIC          ABS, MAX

C
C      Initialize the arrays DLOWER, DIAG, and DUP1 to store
C      the first subdiagonal, the diagonal, and the first
C      superdiagonal of the coefficient matrix A shown below.
C      Initialize the array B to store the right hand side
C      matrix b shown below.
C
C
C      -3   1   6
C      1  -2   12
C      A =   1  -2   18
C           1  -2   12
C           1  -1   6

```

```

DATA DLOWER / 1.0D0, 1.0D0, 1.0D0, 1.0D0 /
DATA DIAG / -3.0D0, -2.0D0, -2.0D0, -2.0D0, -1.0D0 /
DATA DUP1 / 1.0D0, 1.0D0, 1.0D0, 1.0D0 /
DATA B / 6.0D0, 1.2D1, 1.8D1, 1.2D1, 6.0D0 /
C
C      Print the initial value of A.
C
      PRINT 1000
      DO 100, IROW = 1, N
        PRINT 1010, (0.0D0, ICOL = 1, IROW - 2),
          $          (DLOWER(ICOL + 1), ICOL = ABS(IROW - 2),
          $          IROW - 2), DIAG(IROW),
          $          (DUP1(IROW), ICOL = 1, MIN(1, N - IROW)),
          $          (0.0D0, ICOL = IROW + 2, N)
100 CONTINUE
      PRINT 1020
      PRINT 1030, B
C
C      LU factor A.
C
      CALL DGTTRF (N, DLOWER, DIAG, DUP1, DUP2, IPIVOT, INFO)
      IF (INFO .NE. 0) THEN
        PRINT 1040, INFO
        STOP 1
      END IF
      ANORM = 4.0D0
C
C      Estimate and print the condition number of A.
C
      CALL DGTCON ('ONE-NORM', N, DLOWER, DIAG, DUP1, DUP2,
        $          IPIVOT, ANORM, RCOND, WORK, IWORK, INFO)
      IF (INFO .NE. 0) THEN
        PRINT 1050, ABS (INFO)
        STOP 2
      END IF
      PRINT 1060, 1.0D0 / RCOND
C
C      Solve the system and print the solution.
C
      CALL DGTTRS ('NO TRANSPOSE A', N, NRHS, DLOWER, DIAG, DUP1,
        $          DUP2, IPIVOT, B, LDB, INFO)
      IF (INFO .NE. 0) THEN
        PRINT 1070, INFO
        STOP 3
      END IF
      PRINT 1080
      PRINT 1030, B
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (5(3X, F5.2))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (1X, F8.2)
1040 FORMAT (1X, 'Error factoring A, INFO = ', I5)
1050 FORMAT (1X, 'Illegal argument to DGTCON, argument #', I2)
1060 FORMAT (/1X, 'Estimated condition number of A: ', F6.2)
1070 FORMAT (1X, 'Error solving Ax=b, INFO = ', I5)

```

```
1080 FORMAT (/1X, 'x:')
C
      END
```

Sample Output

```
A:
-3.00    1.00    0.00    0.00    0.00
 1.00   -2.00    1.00    0.00    0.00
 0.00    1.00   -2.00    1.00    0.00
 0.00    0.00    1.00   -2.00    1.00
 0.00    0.00    0.00    1.00   -1.00

b:
 6.00
12.00
18.00
12.00
 6.00

Estimated condition number of A:  50.00

x:
-27.00
-75.00
-111.00
-129.00
-135.00
```

Eigenvalues and Eigenvectors of a Hermitian Matrix in Banded Storage (Simple Driver)

The subroutines described in this section compute the eigenvalues and, optionally, the eigenvectors of a Hermitian matrix A in banded storage. The eigenvectors are normalized to have a Euclidean norm of 1. Note that the expert driver xHBEVX is also available.

Calling Sequence

```
CALL ZHBEV (JOBZ, UPLO, N, NDIAG, ZA, LDA, DW, ZZ, LDZ, ZWORK,
            DWORK2, INFO)
CALL CHBEV (JOBZ, UPLO, N, NDIAG, CA, LDA, SW, CZ, LDZ, CWORK,
            SWORK2, INFO)

void zhbev (char jobz, char uplo, int n, int ndiag, doublecomplex
            *za, int lda, double *dw, doublecomplex *zz, int ldz,
            int *info)
void chbev (char jobz, char uplo, int n, int ndiag, complex *ca,
            int lda, float *sw, complex *cz, int ldz, int *info)
```

Arguments

JOBZ	Indicates whether to compute eigenvalues only or to compute both eigenvalues and eigenvectors. The legal values for JOBZ are listed below. Any values not listed below are illegal. 'N' or 'n' Compute eigenvalues only. 'V' or 'v' Compute both eigenvalues and eigenvectors.
UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
NDIAG	Number of superdiagonals or subdiagonals of the matrix A. $N-1 \geq \text{NDIAG} \geq 0$ but if $N = 0$ then $\text{NDIAG} = 0$. If UPLO = 'U' or 'u', NDIAG is the number of superdiagonals. If UPLO = 'L' or 'l', NDIAG is the number of subdiagonals.
xA	On entry, the upper or lower triangle of the matrix A. On exit, A is overwritten.

LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq NDIAG + 1$.
xW	On exit, the N eigenvalues in ascending order.
xZ	On exit, if JOBZ = 'V' or 'v' then Z contains the orthonormal eigenvectors. If JOBZ = 'N' or 'n' then Z is not used.
LDZ	Leading dimension of the array Z as specified in a dimension or type statement. $LDZ \geq 1$. If JOBZ = 'V' or 'v' then $LDZ \geq \max(1, N)$.
xWORK	Scratch array with a dimension of N.
xWORK2	Scratch array with a dimension of $\max(1, 3 \times N - 2)$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where i = INFO , had an illegal value. INFO > 0 Convergence failure.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
DOUBLE PRECISION ZERO
INTEGER LDA, LDEVAL, LDEVEC, LDWORK, LDWRK2
INTEGER N, NDIAG
PARAMETER (N = 3)
PARAMETER (NDIAG = 1)
PARAMETER (LDA = NDIAG + 1)
PARAMETER (LDEVAL = N)
PARAMETER (LDEVEC = N)
PARAMETER (LDWORK = N)
PARAMETER (LDWRK2 = 3 * N - 2)
PARAMETER (ZERO = 0.0D0)

C
DOUBLE PRECISION EVALS(LDEVAL), WORK2(LDWRK2)
COMPLEX*16 A(LDA,N), EVECS(LDEVEC,N), WORK(LDWORK)
INTEGER ICOL, INFO, IROW

C
EXTERNAL ZHBEV
INTRINSIC ABS, CMPLX, CONJG, DBLE

C
C Initialize the array A to store the coefficient array A
C shown below.
C
C      4      4+4i      0
C  A = 4-4i      4      4+4i
C      0      4-4i      4
C
C
DATA A / (8D8,8D8), (4.0D0, 8D8), (4.0D0, 4.0D0),
$      (4.0D0, 8D8), (4.0D0, 4.0D0), (4.0D0, 8D8) /

```



```

C
C      Print the initial value of A.
C
      PRINT 1000
      PRINT 1010, CMPLX(DBLE(A(2,1)), ZERO), A(1,2),
$      CMPLX(ZERO, ZERO)
      PRINT 1010, CONJG(A(1,2)), CMPLX(DBLE(A(2,2)), ZERO), A(1,3)
      PRINT 1010, (ZERO, ZERO), CONJG(A(1,3)), CMPLX(DBLE(A(2,3)),
$      ZERO)
      PRINT 1020
      DO 100, IROW = 1, LDA
        PRINT 1010, (A(IROW,ICOL), ICOL = 1, N)
100 CONTINUE
C
C      Compute the eigenvalues of A.
C
      CALL ZHBEV ('VALUES AND VECTORS',
$              'UPPER TRIANGLE OF A STORED', N, NDIAG, A, LDA,
$              EVALS, EVECS, LDEVEC, WORK, WORK2, INFO)
      IF (INFO .LT. 0) THEN
        PRINT 1030, ABS(INFO)
        STOP 1
      ELSE IF (INFO .GT. 0) THEN
        PRINT 1040
        STOP 2
      END IF
C
C      Print the eigenvalues and eigenvectors.
C
      PRINT 1050
      DO 200, IROW = 1, N
        PRINT 1060, EVALS(IROW), (EVECS(IROW,ICOL), ICOL = 1, N)
200 CONTINUE
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (1X, 10(: 2X, '(', F5.1, ', ', F5.1, ')'))
1020 FORMAT (/1X, 'A in Hermitian banded form: ',
$          '(* in unused entries)')
1030 FORMAT (1X, 'Illegal argument to ZHEEV, argument #', I2)
1040 FORMAT (1X, 'Convergence failure, INFO = ', I2)
1050 FORMAT (/1X, 'Eigenvalue', 16X, 'Eigenvector**T')
1060 FORMAT (1X, F8.3, 6X, '[', 3('(', F4.1, ', ', F4.1, ') '),
$          ']')
C
      END

```

Sample Output

```
A in full form:
( 4.0, 0.0) ( 4.0, 4.0) ( 0.0, 0.0)
( 4.0, -4.0) ( 4.0, 0.0) ( 4.0, 4.0)
( 0.0, 0.0) ( 4.0, -4.0) ( 4.0, 0.0)

A in Hermitian banded form: (* in unused entries)
(*****,*****) ( 4.0, 4.0) ( 4.0, 4.0)
( 4.0,*****) ( 4.0,*****) ( 4.0,*****)

Eigenvalue      Eigenvector**T
-4.000          [( 0.5, 0.0) (-0.7, 0.0) ( 0.5, 0.0) ]
 4.000          [(-0.5, 0.5) ( 0.0, 0.0) ( 0.5, -0.5) ]
12.000          [( 0.0, -0.5) ( 0.0, -0.7) ( 0.0, -0.5) ]
```

Eigenvalues and Eigenvectors of a Hermitian Matrix in Banded Storage (Expert Driver)

The subroutines described in this section compute the eigenvalues and, optionally, the eigenvectors of a Hermitian matrix *A* in banded storage. The eigenvectors are normalized to have a Euclidean norm of 1. Eigenvalues and eigenvectors can be selected by specifying either a range of values or a range of indices for the desired eigenvalues. Note that the simple driver *xHBEV* is also available.

Calling Sequence

```
CALL ZHBEVX (JOBZ, RANGE, UPLO, N, NDIAG, ZA, LDA, ZQ, LDQ, DVL,
            DVU, IL, IU, DABTOL, NFOUND, DW, ZZ, LDZ, ZWORK,
            DWORK2, IWORK3, IFAIL, INFO)
CALL CHBEVX (JOBZ, RANGE, UPLO, N, NDIAG, CA, LDA, CQ, LDQ, SVL,
            SVU, IL, IU, SABTOL, NFOUND, SW, CZ, LDZ, CWORK,
            SWORK2, IWORK3, IFAIL, INFO)

void zhbevz (char jobz, char range, char uplo, int n, int ndiag,
            doublecomplex *za, int lda, doublecomplex *zq, int ldq,
            double dvl, double dvu, int il, int iu, double dabtol,
            int *nfound, double *dw, doublecomplex *zz, int ldz,
            int *ifail, int *info)
void chbevz (char jobz, char range, char uplo, int n, int ndiag,
            complex *ca, int lda, complex *q, int ldq, float svl,
            float svu, int il, int iu, float sabtol, int *nfound,
            float *sw, complex *cz, int ldz, int *ifail, int *info)
```

Arguments

JOBZ	Indicates whether to compute eigenvalues only or to compute both eigenvalues and eigenvectors. The legal values for JOBZ are listed below. Any values not listed below are illegal. 'N' or 'n' Compute eigenvalues only. 'V' or 'v' Compute both eigenvalues and eigenvectors.
RANGE	Indicates whether to compute all eigenvalues or some subset. Legal values for RANGE are shown below. Any values not shown below are illegal. 'A' or 'a' All eigenvalues will be computed.

	'T' or 'i'	The ILth through IUth eigenvalues will be computed; see the descriptions of the IL and IU arguments below.
	'V' or 'v'	All eigenvalues in the half-open interval (VL,VU] will be computed; see the descriptions of the VL and VU arguments below.
UPLO		Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal.
	'U' or 'u'	xA contains the upper triangle.
	'L' or 'l'	xA contains the lower triangle.
N		Order of the matrix A. $N \geq 0$.
NDIAG		Number of superdiagonals or subdiagonals of the matrix A. $N-1 \geq \text{NDIAG} \geq 0$ but if $N = 0$ then $\text{NDIAG} = 0$. If UPLO = 'U' or 'u', NDIAG is the number of superdiagonals. If UPLO = 'L' or 'l', NDIAG is the number of subdiagonals.
xA		On entry, the upper or lower triangle of the matrix A. On exit, A has been overwritten.
LDA		Leading dimension of the array A as specified in a dimension or type statement. $\text{LDA} \geq \text{NDIAG} + 1$.
xQ		On exit, if JOBZ = 'V' or 'v' then Q contains the $N \times N$ unitary matrix used in the reduction to tridiagonal form. If JOBZ = 'N' or 'n' then Q is not used.
LDQ		Leading dimension of the array Q as specified in a dimension or type statement. $\text{LDQ} \geq \max(1, N)$.
xVL, xVU		If RANGE = 'V' or 'v' then only eigenvalues in the half-open interval (VL,VU] will be computed. Otherwise not used.
IL, IU		If RANGE = 'T' or 'i' then only the ILth through IUth eigenvalues will be computed. Otherwise not used. $1 \leq \text{IL} \leq \text{IU} \leq N$.
xABTOL		The absolute error tolerance for the eigenvalues. An approximate eigenvalue is considered to have converged when it lies in an interval [a,b] of width less than or equal to $\text{ABTOL} + \epsilon \times \max(a , b)$ where ϵ is machine precision. If $\text{ABTOL} \leq 0$ then $\epsilon \times \ T\ _1$ will be used in its place, where $\ T\ _1$ is the 1-norm of the tridiagonal matrix obtained by reducing A to tridiagonal form.
NFOUND		On exit, the total number of eigenvalues found. $0 \leq \text{NFOUND} \leq N$. If RANGE = 'A' or 'a' then $\text{NFOUND} = N$. If RANGE = 'T' or 'i' then $\text{NFOUND} = \text{IU} - \text{IL} + 1$.
xW		On exit, W(1:NFOUND) contains the computed eigenvalues in ascending order.

xZ	On exit, if JOBZ = 'V' or 'v' then the first NFOUND columns of Z contain the orthonormal eigenvectors corresponding to the selected eigenvalues. If an eigenvector fails to converge then the corresponding column of Z contains an approximation to that eigenvector and its index is returned in IFAIL. If JOBZ = 'N' or 'n' then Z is not used. Note that Z must have at least NFOUND columns. If RANGE = 'V' or 'v' then NFOUND may not be known in advance and an upper bound must be used. For A(LDZ,K) where NFOUND > K, the eigenvectors will be stored into memory with unpredictable results.
LDZ	Leading dimension of the array Z as specified in a dimension or type statement. $LDZ \geq 1$. If JOBZ = 'V' or 'v' then $LDZ \geq \max(1, N)$.
xWORK	Scratch array with a dimension of N.
xWORK2	Scratch array with a dimension of $7 \times N$.
IWORK3	Scratch array with a dimension of $5 \times N$.
IFAIL	On exit, an N-element array that indicates which eigenvectors that failed to converge. If JOBZ = 'V' or 'v' and INFO = 0 then all N elements contain zero. If INFO > 0 then IFAIL contains the indices of the eigenvectors that failed to converge. If JOBZ = 'N' or 'n' then IFAIL is not used.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The i-th argument, where $i = \text{INFO} $, had an illegal value. INFO > 0 There were i eigenvectors, where $i = \text{INFO}$, that failed to converge. Their indices are stored in IFAIL.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
DOUBLE PRECISION ZERO
INTEGER LDA, LDEVAL, LDEVEC, LDIWRK, LDQ, LDWORK
INTEGER LDWRK2, N, NDIAG
PARAMETER (N = 3)
PARAMETER (NDIAG = 1)
PARAMETER (LDA = NDIAG + 1)
PARAMETER (LDEVAL = N)
PARAMETER (LDEVEC = N)
PARAMETER (LDIWRK = 5 * N)
PARAMETER (LDQ = N)
PARAMETER (LDWORK = N)
PARAMETER (LDWRK2 = 7 * N)
PARAMETER (ZERO = 0.0D0)

```

```

C      DOUBLE PRECISION  EVALS(LDEVAL), TEMP, WORK2(LDWRK2)
C      COMPLEX*16        A(LDA,N), EVECS(LDEVEC,N), Q(LDQ,N)
C      COMPLEX*16        WORK(LDWORK)
C      INTEGER           ICOL, IFAIL(N), INFO, IROW, ITEMP
C      INTEGER           IWORK(LDIWRK), NFOUND
C
C      EXTERNAL           ZHBEVX
C      INTRINSIC          ABS, CMPLX, CONJG, DBLE
C
C      Initialize the array A to store the coefficient array A
C      shown below.
C
C      DATA A / (8D8,8D8), (4.0D0, 8D8), (4.0D0, 4.0D0),
$              (4.0D0, 8D8), (4.0D0, 4.0D0), (4.0D0, 8D8) /
C
C      Print the initial value of A.
C
C      PRINT 1000
C      PRINT 1010, CMPLX(DBLE(A(2,1)), ZERO), A(1,2),
$      CMPLX(ZERO, ZERO)
C      PRINT 1010, CONJG(A(1,2)), CMPLX(DBLE(A(2,2)), ZERO), A(1,3)
C      PRINT 1010, (ZERO, ZERO), CONJG(A(1,3)), CMPLX(DBLE(A(2,3)),
$      ZERO)
C      PRINT 1020
C      DO 100, IROW = 1, LDA
C          PRINT 1010, (A(IROW,ICOL), ICOL = 1, N)
100  CONTINUE
C
C      Compute the eigenvalues and eigenvectors of A.
C
C      CALL ZHBEVX ('VALUES AND VECTORS', 'ALL EIGENVALUES',
$              'UPPER TRIANGLE OF A STORED', N, NDIAG,
$              A, LDA, Q, LDQ, TEMP, TEMP, ITEMP, ITEMP,
$              0.0D0, NFOUND, EVALS, EVECS, LDEVEC, WORK,
$              WORK2, IWORK, IFAIL, INFO)
C      IF (INFO .LT. 0) THEN
C          PRINT 1030, ABS(INFO)
C          STOP 1
C      ELSE IF (INFO .GT. 0) THEN
C          PRINT 1040
C          STOP 2
C      END IF
C
C      Print the eigenvalues and eigenvectors.
C
C      PRINT 1050
C      DO 200, IROW = 1, N
C          PRINT 1060, EVALS(IROW), (EVECS(IROW,ICOL), ICOL = 1, N)
200  CONTINUE
C
1000  FORMAT (1X, 'A in full form:')
1010  FORMAT (1X, 10(: 2X, '(', F5.1, ', ', F5.1, ')'))
1020  FORMAT (/1X, 'A in Hermitian banded form: ',
$      ' (* in unused entries)')
1030  FORMAT (1X, 'Illegal argument to ZHEEVX, argument #', I2)

```

```
1040 FORMAT (1X, 'Convergence failure, INFO = ', I2)
1050 FORMAT (/1X, 'Eigenvalue', 20X, 'Eigenvector**T')
1060 FORMAT (1X, F8.3, 4X, '[', 3('(', F6.3, ', ', F6.3, ') '),
$      ']'')
C
      END
```

Sample Output

```
A in full form:
(  4.0,  0.0) (  4.0,  4.0) (  0.0,  0.0)
(  4.0, -4.0) (  4.0,  0.0) (  4.0,  4.0)
(  0.0,  0.0) (  4.0, -4.0) (  4.0,  0.0)

A in Hermitian banded form: (* in unused entries)
(*****, *****) (  4.0,  4.0) (  4.0,  4.0)
(  4.0, *****) (  4.0, *****) (  4.0, *****)

Eigenvalue      Eigenvector**T
-4.000  [(  0.500,  0.000) (-0.707,  0.000) (  0.500,  0.000) ]
 4.000  [(-0.500,  0.500) (  0.000,  0.000) (  0.500, -0.500) ]
12.000  [(  0.000, -0.500) (  0.000, -0.707) (  0.000, -0.500) ]
```

Reciprocal Condition Number of a UDU- or LDL-Factored Hermitian Matrix

The subroutines described in this section estimate the reciprocal of the condition number of a Hermitian matrix A, which has been UDU-factored or LDL-factored by xHETRF.

Calling Sequence

```
CALL ZHECON (UPLO, N, ZA, LDA, IPIVOT, DANORM, DRCOND, ZWORK, INFO)
CALL CHECON (UPLO, N, CA, LDA, IPIVOT, SANORM, SRCOND, CWORK, INFO)

void zhecon (char uplo, int n, doublecomplex *za, int lda, int
             *ipivot, double danorm, double *drcond, int *info)
void checon (char uplo, int n, complex *ca, int lda, int *ipivot,
             float sanorm, float *srcond, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
xA	UDU or LDL factorization of the matrix A, as computed by xHETRF.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
IPIVOT	Pivot indices as computed by xHETRF.
xANORM	The 1-norm or ∞ -norm of matrix A.
xRCOND	On exit, the estimated reciprocal of the condition number of the matrix A where the reciprocal condition number of A is defined to be $1 / (ANORM \times \ A^{-1}\)$. The reciprocal of the condition number is estimated instead of the condition number itself to avoid overflow or division by zero. If RCOND is less than machine precision (in particular, if RCOND = 0) then A is singular to working precision.
xWORK	Scratch array with a dimension of $2 \times N$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where $i = INFO $, had an illegal value.


```

DO 160, IROW = 1, N
  ANORM1 = 0.0D0
  DO 140, ICOL = 1, IROW - 1
    ANORM1 = ANORM1 + ABS(A(ICOL,IROW))
140  CONTINUE
  ANORM1 = ANORM1 + ABS(DBLE(A(IROW,IROW)))
  DO 150, ICOL = IROW + 1, N
    ANORM1 = ANORM1 + ABS(A(IROW,ICOL))
150  CONTINUE
  IF (ANORM .LT. ANORM1) THEN
    ANORM = ANORM1
  END IF
160 CONTINUE
C
C   Factor the matrix.
C
  CALL ZHETRF ('UPPER TRIANGLE OF A STORED', N, A, LDA,
$    IPIVOT, WORK, LDWORK, INFO)
  IF (INFO .LT. 0) THEN
    PRINT 1040, ABS(INFO)
    STOP 1
  ELSE IF (INFO .GT. 0) THEN
    PRINT 1050
    STOP 2
  END IF
C
C   Estimate the condition number.  Print a warning if it is
C   unreasonably high.
C
  CALL ZHECON ('UPPER TRIANGULAR FACTOR OF A', N, A, LDA,
$    IPIVOT, ANORM, RCOND, WORK, INFO)
  IF (INFO .LT. 0) THEN
    PRINT 1060, ABS(INFO)
    STOP 3
  END IF
  PRINT 1070, RCOND
  IF ((1.0D0 + RCOND) .EQ. 1.0D0) THEN
    PRINT 1080
  END IF
C
C   Compute and print the solution.
C
  CALL ZHETRS ('UPPER TRIANGULAR FACTOR OF A', N, NRHS,
$    A, LDA, IPIVOT, B, LDB, INFO)
  IF (INFO .LT. 0) THEN
    PRINT 1090, ABS(INFO)
    STOP 4
  END IF
  PRINT 1100
  DO 210, ICOL = 1, NRHS
    DO 200, IROW = 1, N
      PRINT 1010, B(IROW,ICOL)
200  CONTINUE
210 CONTINUE
C
1000 FORMAT (1X, 'A in full form:')

```

```
1010 FORMAT (1X, 10(: 2X, '(', F4.1, ', ', F4.1, ')'))
1020 FORMAT (/1X, 'A in Hermitian form: (* in unused entries)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (1X, 'Illegal argument to ZHETRF, argument #', I2)
1050 FORMAT (1X, 'A is singular to working precision.')
1060 FORMAT (1X, 'Illegal argument to ZHECON, argument #', I2)
1070 FORMAT (/1X,
      $      'Estimated reciprocal condition number of A: ',
      $      F5.3)
1080 FORMAT (1X, 'A may be singular to working precision.')
1090 FORMAT (1X, 'Illegal argument to ZHETRS, argument #', I2)
1100 FORMAT (/1X, 'x:')
C
      END
```

Sample Output

```
A in full form:
( 1.0,****) ( 1.0,-1.0) ( 1.0,-1.0)
( 1.0, 1.0) ( 3.0,****) ( 3.0,-1.0)
( 1.0, 1.0) ( 3.0, 1.0) ( 5.0,****)

A in Hermitian form: (* in unused entries)
( 1.0,****) ( 1.0,-1.0) ( 1.0,-1.0)
(****,****) ( 3.0,****) ( 3.0,-1.0)
(****,****) (****,****) ( 5.0,****)

b:
( 4.0, 1.0)
( 4.0, 6.0)
( 4.0, 7.0)

Estimated reciprocal condition number of A: 0.011

x:
( 1.0, 0.0)
( 0.0, 2.0)
( 1.0, 0.0)
```

Eigenvalues and Eigenvectors of a Hermitian Matrix (Simple Driver)

The subroutines described in this section compute all eigenvalues and, optionally, eigenvectors of a Hermitian matrix A. The eigenvectors are normalized to have a Euclidean norm of 1. Note that the expert driver xHEEVX is also available.

Calling Sequence

```
CALL ZHEEV  (JOBZ, UPLO, N, ZA, LDA, DW, ZWORK, LDWORK, DWORK2,
             INFO)
CALL CHEEV  (JOBZ, UPLO, N, CA, LDA, SW, CWORK, LDWORK, SWORK2,
             INFO)

void zheev  (char jobz, char uplo, int n, doublecomplex *za, int
            lda, double *dw, int *info)
void cheev  (char jobz, char uplo, int n, complex *ca, int lda,
            float *sw, int *info)
```

Arguments

JOBZ	Indicates whether to compute eigenvalues only or to compute both eigenvalues and eigenvectors. The legal values for JOBZ are listed below. Any values not listed below are illegal. 'N' or 'n' Compute eigenvalues only. 'V' or 'v' Compute both eigenvalues and eigenvectors.
UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
xA	On entry, the upper or lower triangle of the matrix A. On exit, if JOBZ = 'V' or 'v' then A contains the orthonormal eigenvectors. If JOBZ = 'N' or 'n' then the upper or lower triangle of A, including the diagonal, is overwritten.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
xW	On exit, the N eigenvalues in ascending order.
xWORK	Scratch array with a dimension of LDWORK.
LDWORK	Leading dimension of the array WORK as specified in a dimension or type statement. $LDWORK \geq \max(1, 2 \times N - 1)$.

xWORK2 Scratch array with a dimension of $\max(1, 3 \times N - 2)$.
 INFO On exit:
 INFO = 0 Subroutine completed normally.
 INFO < 0 The i th argument, where $i = |\text{INFO}|$, had an illegal value.
 INFO > 0 Convergence failure.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  DOUBLE PRECISION  ZERO
  INTEGER            LDA, LDEVAL, LDWORK, LDWRK2, N
  PARAMETER          (N = 3)
  PARAMETER          (LDA = N)
  PARAMETER          (LDEVAL = N)
  PARAMETER          (LDWORK = 2 * N - 1)
  PARAMETER          (LDWRK2 = 3 * N - 2)
  PARAMETER          (ZERO = 0.0D0)

C
  DOUBLE PRECISION  EVALS(LDEVAL), WORK2(LDWRK2)
  COMPLEX*16        A(LDA,N), WORK(LDWORK)
  INTEGER            ICOL, INFO, IROW

C
  EXTERNAL           ZHEEV
  INTRINSIC          ABS, CMPLX, CONJG, DBLE

C
  Initialize the array A to store the coefficient array A
  shown below.

C
C      4      4-4i      0
C      A = 4+4i      4      4-4i
C      0      4+4i      4
C
C      DATA A / (4.0D0,8D8),      (8D8,8D8),      (8D8,8D8),
$              (4.0D0,-4.0D0), (4.0D0,8D8),      (8D8,8D8),
$              (0.0D0,0.0D0), (4.0D0,-4.0D0), (4.0D0,8D8) /

C
C      Print the initial value of A.
C
  PRINT 1000
  DO 100, IROW = 1, N
    PRINT 1010, (CONJG(A(ICOL,IROW)), ICOL = 1, IROW - 1),
$              CMPLX(DBLE(A(IROW,IROW)), ZERO),
$              (A(IROW,ICOL), ICOL = IROW + 1, N)
100 CONTINUE
  PRINT 1020
  DO 110, IROW = 1, N
    PRINT 1010, (A(IROW,ICOL), ICOL = 1, N)
110 CONTINUE

```

```

      CALL ZHEEV ('VALUES AND VECTORS',
$         'UPPER TRIANGLE OF A STORED', N, A, LDA,
$         EVALS, WORK, LDWORK, WORK2, INFO)
      IF (INFO .LT. 0) THEN
        PRINT 1040, ABS(INFO)
        STOP 1
      ELSE IF (INFO .GT. 0) THEN
        PRINT 1050
        STOP 2
      END IF
C
C   Print the eigenvalues and eigenvectors.
C
      PRINT 1060
      DO 200, IROW = 1, N
        PRINT 1070, EVALS(IROW), (A(IROW,ICOL), ICOL = 1, N)
      200 CONTINUE
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (1X, 10(: 2X, '(', F5.1, ', ', F5.1, ')'))
1020 FORMAT (/1X, 'A in Hermitian form: (* in unused entries)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (1X, 'Illegal argument to ZHEEV, argument #', I2)
1050 FORMAT (1X, 'Convergence failure, INFO = ', I2)
1060 FORMAT (/1X, 'Eigenvalue', 16X, 'Eigenvector**T')
1070 FORMAT (1X, F8.3, 6X, '[', 3('(', F4.1, ', ', F4.1, ') '),
$         ']' )
C
      END

```

Sample Output

```

A in full form:
(  4.0,  0.0) (  4.0, -4.0) (  0.0,  0.0)
(  4.0,  4.0) (  4.0,  0.0) (  4.0, -4.0)
(  0.0,  0.0) (  4.0,  4.0) (  4.0,  0.0)

A in Hermitian form: (* in unused entries)
(  4.0, ***** ) (  4.0, -4.0) (  0.0,  0.0)
( *****, ***** ) (  4.0, ***** ) (  4.0, -4.0)
( *****, ***** ) ( *****, ***** ) (  4.0, ***** )

Eigenvalue      Eigenvector**T
-4.000          [(  0.0,  0.5) (  0.0,  0.7) (  0.0,  0.5) ]
 4.000          [(  0.5, -0.5) (  0.0,  0.0) (-0.5,  0.5) ]
12.000          [(-0.5,  0.0) (  0.7,  0.0) (-0.5,  0.0) ]

```

Eigenvalues and Eigenvectors of a Hermitian Matrix (Expert Driver)

The subroutines described in this section compute all eigenvalues and, optionally, eigenvectors of a Hermitian matrix A . The eigenvectors are normalized to have a Euclidean norm of 1. Eigenvalues and eigenvectors can be selected by specifying either a range of values or a range of indices for the desired eigenvalues. Note that the simple driver xHEEV is also available.

Calling Sequence

```
CALL ZHEEVX (JOBZ, RANGE, UPLO, N, ZA, LDA, DVL, DVU, IL, IU,
             DABTOL, NFOUND, DW, ZZ, LDZ, ZWORK, LDWORK, DWORK2,
             IWORK3, IFAIL, INFO)

CALL CHEEVX (JOBZ, RANGE, UPLO, N, CA, LDA, SVL, SVU, IL, IU,
             SABTOL, NFOUND, SW, CZ, LDZ, CWORK, LDWORK, SWORK2,
             IWORK3, IFAIL, INFO)

void zheevx (char jobz, char range, char uplo, int n, doublecomplex
            *za, int lda, double dvl, double dvu, int il, int iu,
            double dabtol, int *nfound, double *dw, doublecomplex
            *zz, int ldz, int *ifail, int *info)

void cheevx (char jobz, char range, char uplo, int n, complex *ca,
            int lda, float svl, float svu, int il, int iu, float
            sabtol, int *nfound, float *sw, complex *cz, int ldz,
            int *ifail, int *info)
```

Arguments

JOBZ	Indicates whether to compute eigenvalues only or to compute both eigenvalues and eigenvectors. The legal values for JOBZ are listed below. Any values not listed below are illegal. 'N' or 'n' Compute eigenvalues only. 'V' or 'v' Compute both eigenvalues and eigenvectors.
RANGE	Indicates whether to compute all eigenvalues or some subset. Legal values for RANGE are shown below. Any values not shown below are illegal. 'A' or 'a' All eigenvalues will be computed. 'I' or 'i' The ILth through IUth eigenvalues will be computed; see the descriptions of the IL and IU arguments below. 'V' or 'v' All eigenvalues in the half-open interval (VL,VU] will be computed; see the descriptions of the VL and VU arguments below.

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
xA	On entry, the upper or lower triangle of the matrix A. On exit, the upper or lower triangle of A, including the diagonal, is overwritten.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
xVL, xVU	If RANGE = 'V' or 'v' then only eigenvalues in the half-open interval (VL,VU] will be computed. Otherwise not used.
IL, IU	If RANGE = 'I' or 'i' then only the ILth through IUth eigenvalues will be computed. Otherwise not used. $1 \leq IL \leq IU \leq N$.
xABTOL	The absolute error tolerance for the eigenvalues. An approximate eigenvalue is considered to have converged when it lies in an interval [a,b] of width less than or equal to $ABTOL + \epsilon \times \max(a , b)$ where ϵ is machine precision. If $ABTOL \leq 0$ then $\epsilon \times \ T\ _1$ will be used in its place, where $\ T\ _1$ is the 1-norm of the tridiagonal matrix obtained by reducing A to tridiagonal form.
NFOUND	On exit, the total number of eigenvalues found. $0 \leq NFOUND \leq N$. If RANGE = 'A' or 'a' then $NFOUND = N$. If RANGE = 'I' or 'i' then $NFOUND = IU - IL + 1$.
xW	On exit, W(1:NFOUND) contains the computed eigenvalues in ascending order.
xZ	On exit, if JOBZ = 'V' or 'v' then the first NFOUND columns of Z contain the orthonormal eigenvectors corresponding to the selected eigenvalues. If an eigenvector fails to converge then the corresponding column of Z contains an approximation to that eigenvector and its index is returned in IFAIL. If JOBZ = 'N' or 'n' then Z is not used. Note that Z must have at least NFOUND columns. If RANGE = 'V' or 'v' then NFOUND may not be known in advance and an upper bound must be used. For A(LDZ,K) where $NFOUND > K$, the eigenvectors will be stored into memory with unpredictable results.
LDZ	Leading dimension of the array Z as specified in a dimension or type statement. $LDZ \geq 1$. If JOBZ = 'V' or 'v' then $LDZ \geq \max(1, N)$.
xWORK	Scratch array with a dimension of LDWORK.
LDWORK	Leading dimension of the array WORK as specified in a dimension or type statement. $LDWORK \geq 2 \times N - 1$.
xWORK2	Scratch array with a dimension of $7 \times N$.
IWORK3	Scratch array with a dimension of $5 \times N$.

IFAIL	On exit, an N-element array that indicates which eigenvectors that failed to converge. If JOBZ = 'V' or 'v' and INFO = 0 then all N elements contain zero. If INFO > 0 then IFAIL contains the indices of the eigenvectors that failed to converge. If JOBZ = 'N' or 'n' then IFAIL is not used.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where i = INFO , had an illegal value. INFO > 0 There were i eigenvectors, where i = INFO, that failed to converge. Their indices are stored in IFAIL.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
DOUBLE PRECISION ZERO
INTEGER LDA, LDEVAL, LDEVEC, LDIWRK, LDWORK
INTEGER LDWRK2, N
PARAMETER (N = 3)
PARAMETER (LDA = N)
PARAMETER (LDEVAL = N)
PARAMETER (LDEVEC = N)
PARAMETER (LDIWRK = 5 * N)
PARAMETER (LDWORK = 2 * N - 1)
PARAMETER (LDWRK2 = 7 * N)
PARAMETER (ZERO = 0.0D0)
C
DOUBLE PRECISION EVALS(LDEVAL), TEMP, WORK2(LDWRK2)
COMPLEX*16 A(LDA,N), EVECS(LDEVEC,N), WORK(LDWORK)
INTEGER ICOL, IFAIL(N), INFO, IROW, ITEMP
INTEGER IWORK(LDIWRK), NFOUND
C
EXTERNAL ZHEEVX
INTRINSIC ABS, CMPLX, CONJG, DBLE
C
C Initialize the array A to store the coefficient array A
C shown below.
C
C      4      4-4i      0
C      A = 4+4i      0      4-4i
C      0      4+4i      0
C
C      DATA A / (4.0D0,8D8),      (8D8,8D8),      (8D8,8D8),
C $              (4.0D0,-4.0D0), (4.0D0,8D8),      (8D8,8D8),
C $              (0.0D0,0.0D0), (4.0D0,-4.0D0), (4.0D0,8D8) /
C
C Print the initial value of A.
C
PRINT 1000

```

```

DO 100, IROW = 1, N
  PRINT 1010, (CONJG(A(ICOL,IROW)), ICOL = 1, IROW - 1),
$          CMPLX(DBLE(A(IROW,IROW)),ZERO),
$          (A(IROW,ICOL), ICOL = IROW + 1, N)
100 CONTINUE
  PRINT 1020
  DO 110, IROW = 1, N
    PRINT 1010, (A(IROW,ICOL), ICOL = 1, N)
110 CONTINUE
  CALL ZHEEVX ('VALUES AND VECTORS', 'ALL VALUES',
$            'UPPER TRIANGLE OF A STORED', N, A, LDA, TEMP,
$            TEMP, ITEMP, ITEMP, ZERO, NFOUND, EVALS,
$            EVECS, LDEVEC, WORK, LDWORK, WORK2, IWORK,
$            IFAIL, INFO)
  IF (INFO .LT. 0) THEN
    PRINT 1040, ABS(INFO)
    STOP 1
  ELSE IF (INFO .GT. 0) THEN
    PRINT 1050
    STOP 2
  END IF
C
C   Print the eigenvalues and eigenvectors.
C
  PRINT 1060
  DO 200, IROW = 1, N
    PRINT 1070, EVALS(IROW), (EVECS(IROW,ICOL), ICOL = 1, N)
200 CONTINUE
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (1X, 10(: 2X, '(', F5.1, ', ', F5.1, ')'))
1020 FORMAT (/1X, 'A in Hermitian form: (* in unused entries)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (1X, 'Illegal argument to ZHEEVX, argument #', I2)
1050 FORMAT (1X, 'Convergence failure, INFO = ', I2)
1060 FORMAT (/1X, 'Eigenvalue', 16X, 'Eigenvector**T')
1070 FORMAT (1X, F8.3, 6X, '[', 3('(', F4.1, ', ', F4.1, ') '),
$          ']'')
C
  END

```

Sample Output

```

A in full form:
( 4.0,  0.0) ( 4.0, -4.0) ( 0.0,  0.0)
( 4.0,  4.0) ( 4.0,  0.0) ( 4.0, -4.0)
( 0.0,  0.0) ( 4.0,  4.0) ( 4.0,  0.0)

A in Hermitian form: (* in unused entries)
( 4.0, *****) ( 4.0, -4.0) ( 0.0,  0.0)
(*****', *****') ( 4.0, *****') ( 4.0, -4.0)
(*****', *****') (*****', *****') ( 4.0, *****')

```

Eigenvalue	Eigenvector**T		
-4.000	[(0.0, 0.5)	(0.0, 0.7)	(0.0, 0.5)]
4.000	[(0.5, -0.5)	(0.0, 0.0)	(-0.5, 0.5)]
12.000	[(-0.5, 0.0)	(0.7, 0.0)	(-0.5, 0.0)]

Generalized Eigenvalues and Eigenvectors for Hermitian Matrices

The subroutines in this section compute the generalized eigenvalues (α , β) for a pair of square general matrices A and B. Optionally, these subroutines can also compute the generalized eigenvectors.

A generalized eigenvalue is a ratio $\lambda = \alpha / \beta$ such that $AX = \lambda BX$ or $\det(A - \lambda B) = 0$ for general matrices A, B, and X. λ is typically represented as a ratio rather than as a scalar because there are reasonable interpretations for $\alpha = 0$, $\beta = 0$, and for $\alpha = \beta = 0$. A right generalized eigenvector x corresponding to a generalized eigenvalue λ is defined by $(A - \lambda B)x = 0$. A left generalized eigenvector x corresponding to a generalized eigenvalue λ is defined by $(A - \lambda B)Hx = 0$. A good reference for generalized eigenproblems is the book *Matrix Computations*, 2nd. ed. by Golub and van Loan (1989, The Johns Hopkins University Press).

Calling Sequence

```
CALL ZHEGV (ITYPE, JOBZ, UPLO, N, ZA, LDA, ZB, LDB, DW, ZWORK,
            LDWORK, DWORK2, INFO)
CALL CHEGV (ITYPE, JOBZ, UPLO, N, CA, LDA, CB, LDB, SW, CWORK,
            LDWORK, SWORK2, INFO)

void zhegv (int itype, char jobz, char uplo, int n, doublecomplex
            *za, int lda, doublecomplex *zb, int ldb, double *dw,
            int *info)
void chegv (int itype, char jobz, char uplo, int n, complex *ca,
            int lda, complex *cb, int ldb, float *sw, int *info)
```

Arguments

ITYPE	Indicates the problem to be solved. Legal values for ITYPE are shown below. Any values not shown below are illegal.
1	$Ax = \lambda Bx$
2	$ABx = \lambda x$
3	$BAx = \lambda x$
JOBZ	Indicates whether to compute eigenvalues only or to compute both eigenvalues and eigenvectors. The legal values for JOBZ are listed below. Any values not listed below are illegal.
'N' or 'n'	Compute eigenvalues only.
'V' or 'v'	Compute both eigenvalues and eigenvectors.

UPLO	Indicates whether xA and xB contain the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA and xB both contain the upper triangle. 'L' or 'l' xA and xB both contain the lower triangle.
N	Order of the matrix A. $N \geq 0$.
xA	On entry, the upper or lower triangle of the matrix A. On exit, if JOBZ = 'V' or 'v' then A contains the matrix Z of eigenvectors normalized as shown below: ITYPE Eigenvectors stored in A 1 or 2 $ZH \times B \times Z = I$ 3 $ZH \times B-1 \times Z = I$ If JOBZ = 'N' or 'n' and UPLO = 'U' or 'u' then the upper triangle of A including the diagonal is overwritten. If JOBZ = 'N' or 'n' and UPLO = 'L' or 'l' then the lower triangle of A including the diagonal is overwritten.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
xB	On entry, the upper or lower triangle of the matrix B. On exit, if UPLO = 'U' or 'u' then B contains the matrix U from the Cholesky factorization UHU of B. If UPLO = 'L' or 'l' then B contains the matrix L from the Cholesky factorization LLH of B.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
xW	On exit, W(1:NFOUND) contains the computed eigenvalues in ascending order.
xWORK	Scratch array with a dimension of LDWORK.
LDWORK	Leading dimension of the array WORK as specified in a dimension or type statement. $LDWORK \geq \max(1, 2 \times N - 1)$.
xWORK2	Scratch array with a dimension of $\max(1, 3 \times N - 2)$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The <i>i</i> th argument, where $i = INFO $, had an illegal value. $0 < INFO \leq N$ Convergence failure INFO > N The leading minor of order <i>i</i> of B, where $i = INFO - N$, is not positive definite.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
DOUBLE PRECISION ZERO
INTEGER          ITYPE, LDA, LDB, LDWORK, LDWRK2, N
PARAMETER        (ITYPE = 1)
PARAMETER        (N = 3)
PARAMETER        (LDA = N)
PARAMETER        (LDB = N)
PARAMETER        (LDWORK = 2 * N - 1)
PARAMETER        (LDWRK2 = 3 * N - 2)
PARAMETER        (ZERO = 0.0D0)

C
DOUBLE PRECISION EVALS(N), WORK2(LDWRK2)
COMPLEX*16        A(LDA,N), B(LDB,N), WORK(LDWORK)
INTEGER           ICOL, INFO, IROW

C
EXTERNAL          ZHEGV
INTRINSIC         ABS, CONJG

C
C Initialize the arrays A and B to store the matrices A and B
C shown below.
C
C      8      2-2i  2-2i      4      1-i   1-i
C      A = 2+2i   16   4-4i      B = 1+i    8    2-2i
C      2+2i   4+4i   32      1+i  2+2i   16
C
C      DATA A / (8.0D0,8D8), (8D8,8D8), (8D8,8D8),
C $              (2.0D0,-2.0D0), (1.6D1,8D8), (8D8,8D8),
C $              (2.0D0,-2.0D0), (4.0D0,-4.0D0), (3.2D1,8D8) /
C      DATA B / (4.0D0,8D8), (8D8,8D8), (8D8,8D8),
C $              (1.0D0,-1.0D0), (8.0D0,8D8), (8D8,8D8),
C $              (1.0D0,-1.0D0), (2.0D0,-2.0D0), (1.6D1,8D8) /

C
PRINT 1000
DO 100, IROW = 1, N
PRINT 1010, (CONJG(A(ICOL,IROW)), ICOL = 1, IROW - 1),
$          CMPLX(DBLE(A(IROW,IROW)), ZERO),
$          (A(IROW,ICOL), ICOL = IROW + 1, N)
100 CONTINUE
PRINT 1020
PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
PRINT 1030
DO 110, IROW = 1, N
PRINT 1010, (CONJG(B(ICOL,IROW)), ICOL = 1, IROW - 1),
$          CMPLX(DBLE(A(IROW,IROW)), ZERO),
$          (B(IROW,ICOL), ICOL = IROW + 1, N)
110 CONTINUE
PRINT 1040
PRINT 1010, ((B(IROW,ICOL), ICOL = 1, N), IROW = 1, N)

C
CALL ZHEGV (ITYPE, 'VALUES AND EIGENVECTORS',

```

```

$          'UPPER TRIANGLE OF A STORED', N, A, LDA, B, LDB,
$          EVALS, WORK, LDWORK, WORK2, INFO)
  IF (INFO .LT. 0) THEN
    PRINT 1050, ABS(INFO)
    STOP 1
  ELSE IF (INFO .GT. N) THEN
    PRINT 1060, INFO
    STOP 2
  ELSE IF (INFO .GT. 0) THEN
    PRINT 1070, INFO
    STOP 3
  END IF
  PRINT 1080
  DO 120, IROW = 1, N
    PRINT 1090, EVALS(IROW), (A(IROW,ICOL), ICOL = 1, N)
120 CONTINUE
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (3(3X, '(', F5.2, ',', F5.2, ')'))
1020 FORMAT (/1X, 'A in Hermitian form: (* in unused elements)')
1030 FORMAT (/1X, 'B in full form:')
1040 FORMAT (/1X, 'B in Hermitian form: (* in unused elements)')
1050 FORMAT (1X, 'Illegal argument to ZHEGV, argument #', I2)
1060 FORMAT (1X, 'B is not positive definite, INFO = ', I4)
1070 FORMAT (1X, 'Convergence failure, INFO = ', I4)
1080 FORMAT (/1X, 'Eigenvalue', 20X, 'Eigenvector**T')
1090 FORMAT (1X, F8.3, 5X, '[', 3('(', F5.3, ',', F5.3, ')', 2X),
$          ']'')
C
      END

```

Sample Output

```

A in full form:
( 8.00, 0.00) ( 2.00,-2.00) ( 2.00,-2.00)
( 2.00, 2.00) (16.00, 0.00) ( 4.00,-4.00)
( 2.00, 2.00) ( 4.00, 4.00) (32.00, 0.00)

A in Hermitian form: (* in unused elements)
( 8.00, *****) ( 2.00,-2.00) ( 2.00,-2.00)
( *****, *****) (16.00, *****) ( 4.00,-4.00)
( *****, *****) ( *****, *****) (32.00, *****)

B in full form:
( 8.00, 0.00) ( 1.00,-1.00) ( 1.00,-1.00)
( 1.00, 1.00) (16.00, 0.00) ( 2.00,-2.00)
( 1.00, 1.00) ( 2.00, 2.00) (32.00, 0.00)

B in Hermitian form: (* in unused elements)
( 4.00, *****) ( 1.00,-1.00) ( 1.00,-1.00)
( *****, *****) ( 8.00, *****) ( 2.00,-2.00)
( *****, *****) ( *****, *****) (16.00, *****)

```

Eigenvalue	Eigenvector**T		
2.000	[(0.500,0.000)	(-.091,0.091)	(-.070,0.035)]
2.000	[(0.000,0.000)	(0.365,0.000)	(-.052,0.070)]
2.000	[(0.000,0.000)	(0.000,0.000)	(0.261,0.000)]

Refined Solution to a Linear System in a UDU- or LDL-Factored Hermitian Matrix

The subroutines described in this section refine the solution to a linear system $AX = B$ for a Hermitian matrix A , which has been UDU-factored or LDL-factored by xHETRF, and general matrices B and X . These subroutines also compute forward error bounds and backward error estimates for the refined solution.

Calling Sequence

```
CALL ZHERFS (UPLO, N, NRHS, ZA, LDA, ZAF, LDAF, IPIVOT, ZB, LDB,
             ZX, LDX, DFERR, DBERR, ZWORK, DWORK2, INFO)
CALL CHERFS (UPLO, N, NRHS, CA, LDA, CAF, LDAF, IPIVOT, CB, LDB,
             CX, LDX, SFERR, SBERR, CWORK, SWORK2, INFO)

void zherfs (char uplo, int n, int nrhs, doublecomplex *za, int
             lda, doublecomplex *zaf, int ldaf, int *ipivot,
             doublecomplex *zb, int ldb, doublecomplex *zx, int ldx,
             double *dferr, double *dberr, int *info)
void cherfs (char uplo, int n, int nrhs, complex *ca, int lda,
             complex *caf, int ldaf, int *ipivot, complex *cb, int
             ldb, complex *cx, int ldx, float *sferr, float *sberr,
             int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
NRHS	Number of right-hand sides, equal to the number of columns of the matrices B and X. $NRHS \geq 0$.
xA	Upper or lower triangle of the matrix A.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
xAF	UDU or LDL factorization of the matrix A as computed by xHETRF.
LDAF	Leading dimension of the array AF as specified in a dimension or type statement. $LDAF \geq \max(1, N)$.
IPIVOT	Pivot indices as computed by xHETRF.
xB	The $N \times NRHS$ right-hand side matrix B.

LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
xX	On entry, the $N \times NRHS$ solution matrix X. On exit, the refined $N \times NRHS$ solution matrix X.
LDX	Leading dimension of the array X as specified in a dimension or type statement. $LDX \geq \max(1, N)$.
xFERR	On exit, the estimated forward error bound for each solution vector $X(*, j)$ for $1 \leq j \leq NRHS$. If X' is the true solution corresponding to $X(*, j)$ then $FERR(j)$ is an upper bound on the magnitude of the largest element in $X(*, j) - X'$ divided by the magnitude of the largest element in $X(*, j)$.
xBERR	On exit, $BERR(j)$ is the smallest relative change in any element of A or $B(*, j)$ that makes $X(*, j)$ an exact solution to $AX(*, j) = B(*, j)$ for $1 \leq j \leq NRHS$.
xWORK	Scratch array with a dimension of $2 \times N$.
xWORK2	Scratch array with a dimension of N.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The <i>i</i> th argument, where $i = INFO $, had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
DOUBLE PRECISION ZERO
INTEGER LDA, LDAF, LDB, LDB1, LDWORK, LDWRK2, LDX
INTEGER N, NRHS
PARAMETER (N = 3)
PARAMETER (LDA = N)
PARAMETER (LDAF = LDA)
PARAMETER (LDB = N)
PARAMETER (LDB1 = N)
PARAMETER (LDWORK = 2 * N)
PARAMETER (LDWRK2 = N)
PARAMETER (LDX = LDB)
PARAMETER (NRHS = 1)
PARAMETER (ZERO = 0.0D0)

C
DOUBLE PRECISION BERR(N), FERR(N), WORK2(LDWRK2)
COMPLEX*16 A(LDA,N), AF(LDA,N), B(LDB,NRHS)
COMPLEX*16 B1(LDB,NRHS), INFNTY, WORK(LDWORK)
COMPLEX*16 X(LDX,NRHS)
INTEGER ICOL, INFO, IPIVOT(N), IROW

C
EXTERNAL ZCOPY, ZHEMM, ZHERFS, ZHETRF, ZHETRS

```

```

C      INTRINSIC          ABS, DCMPLX, DCONJG, DBLE, SQRT, MAX
C
C      Initialize the array A to store the coefficient array A
C      shown below.
C
C      A =      (1e+16, 0)      (1e+16, 1)      (3e+16, 3e-16)
C      (1e+16, -1)      (1e+16, 0)      (3e+26, 3e-16)
C      (3e+16, 3e-16)      (3e+26, -3e-16)      (7e-16, 0)
C
C      Initialize the array B to store the right-hand side vector b
C      shown below.
C
C      b = (2e-26, 2e+16)
C      (2e+26, 2e-16)
C      (2, 2)
C
C      DATA A / (1.0D16,0.0D0), (-8D300,-8D300), (-8D300,-8D300),
$      (1.0D16,1.0D0), (1.0D16,0.0D0), (-8D300,-8D300),
$      (3.0D16,3.0D-16), (3.0D26,3.0D-16),
$      (7.0D-16,0.0D0) /
C      DATA B / (2.0D-26,2.0D16), (2.0D26,2.0D-16), (2.0D0,2.0D0) /
C
C      Print the initial value of the arrays.
C
C      INFNTY = DCMPLX(SQRT(-1.0D0), SQRT(-1.0D0))
C      DO 110, IROW = 1, N
C          DO 100, ICOL = 1, IROW - 1
C              A(IROW,ICOL) = INFNTY
100      CONTINUE
110      CONTINUE
C      PRINT 1000
C      DO 120, IROW = 1, N
C          PRINT 1010, (DCONJG(A(ICOL,IROW)), ICOL = 1, IROW - 1),
$      DCMPLX(DBLE(A(IROW,IROW)), ZERO),
$      (A(IROW,ICOL), ICOL = IROW + 1, N)
120      CONTINUE
C      PRINT 1020
C      DO 130, IROW = 1, N
C          PRINT 1010, (A(IROW,ICOL), ICOL = 1, N)
130      CONTINUE
C      DO 150, ICOL = 1, NRHS
C          PRINT 1030, ICOL
C          DO 140, IROW = 1, N
C              PRINT 1080, B(IROW,ICOL)
140      CONTINUE
150      CONTINUE
C
C      Factor the matrix.
C
C      CALL ZCOPY (LDA * N, A, 1, AF, 1)
C      CALL ZHETRF ('UPPER TRIANGLE OF A STORED', N, AF, LDAF,
$      IPIVOT, WORK, LDWORK, INFO)
C      IF (INFO .LT. 0) THEN
C          PRINT 1040, ABS(INFO)
C          STOP 1
C      ELSE IF (INFO .GT. 0) THEN

```

```

        PRINT 1050
        STOP 2
    END IF
C
C    Compute and print the solution.
C
    CALL ZCOPY (LDB * NRHS, B, 1, X, 1)
    CALL ZHETRS ('UPPER TRIANGULAR FACTOR OF A', N, NRHS, AF,
$           LDAF, IPIVOT, X, LDX, INFO)
    IF (INFO .LT. 0) THEN
        PRINT 1060, ABS(INFO)
        STOP 3
    END IF
    DO 210, ICOL = 1, NRHS
        PRINT 1070, ICOL
        DO 200, IROW = 1, N
            PRINT 1080, X(IROW,ICOL)
200    CONTINUE
210    CONTINUE
    CALL ZHEMM ('LEFT SIDE: AX', 'UPPER TRIANGLE OF A STORED',
$           N, NRHS, DCMLPX(1.0D0,ZERO), A, LDA, X, LDX,
$           DCMLPX(ZERO,ZERO), B1, LDB1)
    DO 230, ICOL = 1, NRHS
        PRINT 1090, ICOL
        DO 220, IROW = 1, N
            PRINT 1080, B1(IROW,ICOL)
220    CONTINUE
230    CONTINUE
C
C    Refine the solution and print the refined solution with the
C    error bounds.
C
    CALL ZHERFS ('UPPER TRIANGLE OF A STORED', N, NRHS, A, LDA,
$           AF, LDAF, IPIVOT, B, LDB, X, LDX, FERR, BERR,
$           WORK, WORK2, INFO)
    DO 310, ICOL = 1, NRHS
        PRINT 1100, ICOL
        DO 300, IROW = 1, N
            PRINT 1080, X(IROW,ICOL)
300    CONTINUE
310    CONTINUE
    CALL ZHEMM ('LEFT SIDE: AX', 'UPPER TRIANGLE OF A STORED',
$           N, NRHS, DCMLPX(1.0D0,ZERO), A, LDA, X, LDX,
$           DCMLPX(ZERO,ZERO), B1, LDB1)
    DO 330, ICOL = 1, NRHS
        PRINT 1110, ICOL
        DO 320, IROW = 1, N
            PRINT 1080, B1(IROW,ICOL)
320    CONTINUE
330    CONTINUE
    DO 340, IROW = 1, NRHS
        PRINT 1120, IROW, BERR(IROW)
        PRINT 1130, IROW, FERR(IROW)
340    CONTINUE
C
1000 FORMAT (1X, 'A in full form:')

```

```

1010 FORMAT (1X, 10(: 1X, '(', E8.2, ', ', E8.2, ')'))
1020 FORMAT (/1X, 'A in Hermitian form: ',
$      ' ("NaN" in unused entries)')
1030 FORMAT (/1X, 'b for system #', I1)
1040 FORMAT (1X, 'Illegal argument to ZHETRF, argument #', I2)
1050 FORMAT (1X, 'A is singular to working precision.')
1060 FORMAT (1X, 'Illegal argument to ZHETRS, argument #', I2)
1070 FORMAT (/1X, 'Initial x for system #', I1)
1080 FORMAT (3X, '(', E20.14, ', ', E20.14, ')')
1090 FORMAT (/1X, 'Ax with initial x for system #', I1)
1100 FORMAT (/1X, 'Refined x for system #', I1)
1110 FORMAT (/1X, 'Ax with refined x for system #', I1)
1120 FORMAT (/1X, 'Backward error for system #', I1, ': ', E12.6)
1130 FORMAT (1X, 'Forward error for system #', I1, ': ', E12.6)
C
      END

```

Sample Output

```

A in full form:
(0.10E+17, 0.00E+00) (0.10E+17, 0.10E+01) (0.30E+17, 0.30E-15)
(0.10E+17, -.10E+01) (0.10E+17, 0.00E+00) (0.30E+27, 0.30E-15)
(0.30E+17, -.30E-15) (0.30E+27, -.30E-15) (0.70E-15, 0.00E+00)

A in Hermitian form: ("NaN" in unused entries)
(0.10E+17, 0.00E+00) (0.10E+17, 0.10E+01) (0.30E+17, 0.30E-15)
(NaN      , NaN      ) (0.10E+17, 0.00E+00) (0.30E+27, 0.30E-15)
(NaN      , NaN      ) (NaN      , NaN      ) (0.70E-15, 0.00E+00)

b for system #1
(0.2000000000000000E-25, 0.2000000000000000E+17)
(0.2000000000000000E+27, 0.2000000000000000E-15)
(0.2000000000000000E+01, 0.2000000000000000E+01)

Initial x for system #1
(-.200000000004000E+01, 0.200000000004000E+01)
(0.200000000004000E-09, -.200000000004000E-09)
(0.66666666673333E+00, -.66666666673333E-10)

Ax with initial x for system #1
(0.4000000000000000E+01, 0.2000000000000000E+17)
(0.2000000000000000E+27, 0.4000000000000000E+01)
(-.8000000000000000E+01, 0.8000000000000000E+01)

Refined x for system #1
(-.200000000004000E+01, 0.200000000004000E+01)
(0.200000000004000E-09, -.200000000004000E-09)
(0.66666666673333E+00, -.66666666673333E-10)

```

```
Ax with refined x for system #1
(0.4000000000000000E+01, 0.2000000000000000E+17)
(0.2000000000000000E+27, 0.4000000000000000E+01)
(-.8000000000000000E+01, 0.8000000000000000E+01)

Backward error for system #1: 0.666667E-16
Forward error for system #1: 0.143228E-14
```

Solution to a Linear System in a Hermitian Matrix (Simple Driver)

The subroutines described in this section solve a linear system $AX = B$ for a Hermitian matrix A and general matrices B and X . Note that the expert driver `xHESVX` is also available.

Calling Sequence

```
CALL ZHESV  (UPLO, N, NRHS, ZA, LDA, IPIVOT, ZB, LDB, DWORK,
             LDWORK, INFO)
CALL CHESV  (UPLO, N, NRHS, CA, LDA, IPIVOT, CB, LDB, SWORK,
             LDWORK, INFO)

void zhesv  (char uplo, int n, int nrhs, doublecomplex *za, int
             lda, int *ipivot, doublecomplex *zb, int ldb, int
             *info)
void chesv  (char uplo, int n, int nrhs, complex *ca, int lda, int
             *ipivot, complex *cb, int ldb, int *info)
```

Arguments

UPLO	Indicates whether <code>xA</code> contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' <code>xA</code> contains the upper triangle. 'L' or 'l' <code>xA</code> contains the lower triangle.
N	Order of the matrix A . $N \geq 0$.
NRHS	Number of right-hand sides, equal to the number of columns of the matrix B . $NRHS \geq 0$.
<code>xA</code>	On entry, the upper or lower triangle of the matrix A . On exit, the UDU or LDL factorization of A as computed by <code>xHETRF</code> .
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
IPIVOT	On exit, pivot indices as computed by <code>xHETRF</code> .
<code>xB</code>	On entry, the $N \times NRHS$ right-hand side matrix B . On exit, the $N \times NRHS$ solution matrix X .
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
<code>xWORK</code>	Scratch array with a dimension of LDWORK.
LDWORK	Leading dimension of the array <code>WORK</code> as specified in a dimension or type statement. $LDWORK \geq 1$.

INFO	On exit:	
INFO = 0		Subroutine completed normally.
INFO < 0		The <i>i</i> th argument, where <i>i</i> = INFO , had an illegal value.
INFO > 0		D(<i>i</i> , <i>i</i>), where <i>i</i> = INFO, is exactly zero and D is therefore singular. The UDU or LDL factorization has been completed, but the solution could not be computed.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
DOUBLE PRECISION  ZERO
INTEGER           LDA, LDB, LDWORK, N, NRHS
PARAMETER         (N = 3)
PARAMETER         (LDA = N)
PARAMETER         (LDB = N)
PARAMETER         (LDWORK = 1)
PARAMETER         (NRHS = 1)
PARAMETER         (ZERO = 0.0D0)

C
COMPLEX*16        A(LDA,N), B(LDB,NRHS), WORK(LDWORK)
INTEGER           ICOL, INFO, IPIVOT(N), IROW

C
EXTERNAL          ZHESV
INTRINSIC         ABS, CMPLX, CONJG, DBLE

C
C Initialize the array A to store the coefficient array A
C shown below. Initialize the array B to store the right
C hand side vector b shown below.
C
C
C      1      1-i    1-i      4+i
C  A = 1+i    3      3-i      b = 4+6i
C      1+i    3+i    5        4+7i
C
C  DATA A / (1.0D0,8D8), (8D8,8D8), (8D8,8D8),
$           (1.0D0,-1.0D0), (3.0D0,8D8), (8D8,8D8),
$           (1.0D0,-1.0D0), (3.0D0,-1.0D0), (5.0D0,8D8) /
C  DATA B / (4.0D0,1.0D0), (4.0D0,6.0D0), (4.0D0,7.0D0) /

C
C Print the initial value of the arrays.
C
C
PRINT 1000
DO 100, IROW = 1, N
    PRINT 1010, (CONJG(A(ICOL,IROW)), ICOL = 1, IROW - 1),
$              CMPLX(DBLE(A(IROW,IROW)), ZERO),
$              (A(IROW,ICOL), ICOL = IROW + 1, N)
100 CONTINUE

```



```
        PRINT 1020
        DO 110, IROW = 1, N
            PRINT 1010, (A(IROW,ICOL), ICOL = 1, N)
110     CONTINUE
        PRINT 1030
        DO 130, ICOL = 1, NRHS
            DO 120, IROW = 1, N
                PRINT 1010, B(IROW,ICOL)
120     CONTINUE
130     CONTINUE
C
C      Compute and print the solution.
C
        CALL ZHESV ('UPPER TRIANGLE OF A STORED', N, NRHS, A, LDA,
$              IPIVOT, B, LDB, WORK, LDWORK, INFO)
        IF (INFO .LT. 0) THEN
            PRINT 1040, ABS(INFO)
            STOP 1
        ELSE IF (INFO .GT. 0) THEN
            PRINT 1050
            STOP 2
        END IF
        PRINT 1060
        DO 210, ICOL = 1, NRHS
            DO 200, IROW = 1, N
                PRINT 1010, B(IROW,ICOL)
200     CONTINUE
210     CONTINUE
C
1000    FORMAT (1X, 'A in full form:')
1010    FORMAT (1X, 10(: 2X, '(', F4.1, ', ', F4.1, ')'))
1020    FORMAT (/1X, 'A in Hermitian form: (* in unused entries)')
1030    FORMAT (/1X, 'b:')
1040    FORMAT (1X, 'Illegal argument to ZHESV, argument #', I2)
1050    FORMAT (1X, 'A is singular to working precision.')
1060    FORMAT (/1X, 'x:')
C
        END
```

Sample Output

```
A in full form:
( 1.0, 0.0) ( 1.0,-1.0) ( 1.0,-1.0)
( 1.0, 1.0) ( 3.0, 0.0) ( 3.0,-1.0)
( 1.0, 1.0) ( 3.0, 1.0) ( 5.0, 0.0)

A in Hermitian form: (* in unused entries)
( 1.0, ****) ( 1.0,-1.0) ( 1.0,-1.0)
(****, ****) ( 3.0, ****) ( 3.0,-1.0)
(****, ****) (****, ****) ( 5.0, ****)
```

```
b:
( 4.0, 1.0)
( 4.0, 6.0)
( 4.0, 7.0)

x:
( 1.0, 0.0)
( 0.0, 2.0)
( 1.0, 0.0)
```

Solution to a Linear System in a Hermitian Matrix (Expert Driver)

The subroutines described in this section solve a linear system $AX = B$ for a Hermitian matrix A and general matrices B and X . These subroutines also compute forward error bounds and backward error estimates for the solution, and estimate the condition number of the matrix A . Note that the simple driver xHESV is also available.

Calling Sequence

```
CALL ZHESVX (FACT, UPLO, N, NRHS, ZA, LDA, ZAF, LDAF, IPIVOT, ZB,
             LDB, ZX, LDX, DRCOND, DFERR, DBERR, ZWORK, LDWORK,
             DWORK2, INFO)
CALL CHESVX (FACT, UPLO, N, NRHS, CA, LDA, CAF, LDAF, IPIVOT, CB,
             LDB, CX, LDX, SRCOND, SFERR, SBERR, CWORK, LDWORK,
             SWORK2, INFO)

void zhesvx (char fact, char uplo, int n, int nrhs, doublecomplex
            *za, int lda, doublecomplex *zaf, int ldaf, int
            *ipivot, doublecomplex *zb, int ldb, doublecomplex
            *zx, int ldx, double *drcond, double *dferr, double
            *dberr, int *info)
void chesvx (char fact, char uplo, int n, int nrhs, complex *ca,
            int lda, complex *caf, int ldaf, int *ipivot, complex
            *cb, int ldb, complex *cx, int ldx, float *srcond,
            float *sferr, float *sberr, int *info)
```

Arguments

FACT	Indicates whether or not the factored form of the matrix A is supplied on entry. 'F' or 'f' On entry, AF and IPIVOT contain the factored form of A . A , AF, and IPIVOT will not be modified. 'N' or 'n' On entry, the matrix A will be copied to AF and factored.
UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A . $N \geq 0$.

NRHS	Number of right-hand sides, equal to the number of columns in the matrices B and X. $NRHS \geq 0$.
xA	Upper or lower triangle of the matrix A.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
xAF	On entry, if FACT = 'F' or 'f' then AF contains the UDU or LDL factorization of A as computed by xHETRF. Otherwise the entry value of AF is not used. On exit, if FACT = 'N' or 'n' then AF contains the UDU or LDL factorization of A as computed by xHETRF. Otherwise AF is unchanged.
LDAF	Leading dimension of the array AF as specified in a dimension or type statement. $LDAF \geq \max(1, N)$.
IPIVOT	On entry, if FACT = 'F' or 'f', IPIVOT contains pivot indices as computed by xHETRF. Otherwise the entry value is not used. On exit, if FACT = 'N' or 'n', IPIVOT contains pivot indices as computed by xHETRF. Otherwise IPIVOT is unchanged.
xB	On entry, the $N \times NRHS$ right-hand side matrix B.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
XX	On exit, the $N \times NRHS$ solution matrix for the original problem.
LDX	Leading dimension of the array X as specified in a dimension or type statement. $LDX \geq \max(1, N)$.
xRCOND	On exit, an estimate of the reciprocal condition number of the matrix A after equilibration, if any, where the reciprocal condition number is defined to be $1 / (\ A\ \times \ A^{-1}\)$. The reciprocal of the condition number is estimated instead of the condition number itself to avoid overflow or division by zero. If RCOND is less than machine precision (in particular, if $RCOND = 0$) then A is singular to working precision. In this case, $INFO > 0$ is returned and the solution and error bounds are not computed.
xFERR	On exit, the estimated forward error bound for each solution vector $X(*, j)$ for $1 \leq j \leq NRHS$. If X' is the true solution corresponding to $X(*, j)$ then $FERR(j)$ is an upper bound on the magnitude of the largest element in $X(*, j) - X'$ divided by the magnitude of the largest element in $X(*, j)$.
xBERR	On exit, $BERR(j)$ is the smallest relative change in any element of A or $B(*, j)$ that makes $X(*, j)$ an exact solution to $AX(*, j) = B(*, j)$ for $1 \leq j \leq NRHS$.
xWORK	Scratch array with a dimension of LDWORK.
LDWORK	Leading dimension of the array WORK as specified in a dimension or type statement. $LDWORK \geq 2 \times N$.

xWORK2	Scratch array with a dimension of N.	
INFO	On exit:	
	INFO = 0	Subroutine completed normally.
	INFO < 0	The <i>i</i> th argument, where <i>i</i> = INFO , had an illegal value.
	$1 \leq \text{INFO} \leq N$	D(<i>i</i> , <i>i</i>), where <i>i</i> = INFO, is exactly zero and D is therefore singular. The UDU or LDL factorization of A has been completed, but the solution and error bounds could not be computed.
	INFO = N+1	D is nonsingular, but RCOND is less than machine precision. The UDU or LDL factorization of A has been completed, but the solution and error bounds could not be computed.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
DOUBLE PRECISION ZERO
INTEGER LDA, LDAF, LDB, LDWORK, LDWRK2, LDX
INTEGER N, NRHS
PARAMETER (N = 3)
PARAMETER (LDA = N)
PARAMETER (LDAF = N)
PARAMETER (LDB = N)
PARAMETER (LDWORK = 2 * N)
PARAMETER (LDWRK2 = N)
PARAMETER (LDX = N)
PARAMETER (NRHS = 1)
PARAMETER (ZERO = 0.0D0)
C
DOUBLE PRECISION BERR(NRHS), FERR(NRHS), RCOND
DOUBLE PRECISION WORK2(LDWRK2)
COMPLEX*16 A(LDA,N), AF(LDAF,N), B(LDB,NRHS)
COMPLEX*16 WORK(LDWORK)
COMPLEX*16 X(LDX,NRHS)
INTEGER ICOL, INFO, IPIVOT(N), IROW
C
EXTERNAL ZHESVX
INTRINSIC ABS, CMPLX, CONJG
C
C Initialize the array A to store the coefficient array A
C shown below. Initialize the array B to store the right
C hand side vector b shown below.
C
DATA A / (1.0D0,8D8), (8D8,8D8), (8D8,8D8),
$ (1.0D0,-1.0D0), (3.0D0,8D8), (8D8,8D8),
$ (1.0D0,-1.0D0), (3.0D0,-1.0D0), (5.0D0,8D8) /

```

```

DATA B / (4.0D0,1.0D0), (4.0D0,6.0D0), (4.0D0,7.0D0) /
C
C   Print the initial value of the arrays.
C
      PRINT 1000
      DO 100, IROW = 1, N
        PRINT 1010, (CONJG(A(ICOL,IROW)), ICOL = 1, IROW - 1),
$          CMPLX(A(IROW,IROW), ZERO),
$          (A(IROW,ICOL), ICOL = IROW, N)
100  CONTINUE
      PRINT 1020
      DO 110, IROW = 1, N
        PRINT 1010, (A(IROW,ICOL), ICOL = 1, N)
110  CONTINUE
      PRINT 1030
      DO 130, ICOL = 1, NRHS
        DO 120, IROW = 1, N
          PRINT 1010, B(IROW,ICOL)
120  CONTINUE
130  CONTINUE
C
C   Compute the solution.
C
      CALL ZHESVX ('NOT FACTORED A', 'UPPER TRIANGLE OF A STORED',
$              N, NRHS, A, LDA, AF, LDAF, IPIVOT, B, LDB, X,
$              LDX, RCOND, FERR, BERR, WORK, LDWORK, WORK2,
$              INFO)
      IF (INFO .LT. 0) THEN
        PRINT 1040, ABS(INFO)
        STOP 1
      ELSE IF (INFO .GT. 0) THEN
        PRINT 1050
        STOP 2
      END IF
C
C   Print the solution and error bounds.
C
      PRINT 1060
      DO 210, ICOL = 1, NRHS
        DO 200, IROW = 1, N
          PRINT 1010, X(IROW,ICOL)
200  CONTINUE
210  CONTINUE
      PRINT 1070, RCOND
      DO 220, IROW = 1, NRHS
        PRINT 1080, IROW, BERR(IROW)
        PRINT 1090, IROW, FERR(IROW)
220  CONTINUE
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (1X, 10(: 2X, '(', F4.1, ', ', F4.1, ')'))
1020 FORMAT (/1X, 'A in Hermitian form: (* in unused entries)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (1X, 'Illegal argument to ZHESVX, argument #', I2)
1050 FORMAT (1X, 'A is singular to working precision.')
1060 FORMAT (/1X, 'x:')

```

```
1070 FORMAT (/1X,  
$          'Estimated reciprocal condition number of A: ',  
$          F5.3)  
1080 FORMAT (/1X, 'Backward error for system #', I2, ': ', E12.5)  
1090 FORMAT (1X, 'Forward error for system #', I2, ': ', E12.5)  
C  
      END
```

Sample Output

```
A in full form:  
 ( 1.0, 0.0) ( 1.0,****) ( 1.0,-1.0) ( 1.0,-1.0)  
 ( 1.0, 1.0) ( 3.0, 0.0) ( 3.0,****) ( 3.0,-1.0)  
 ( 1.0, 1.0) ( 3.0, 1.0) ( 5.0, 0.0) ( 5.0,****)  
  
A in Hermitian form: (* in unused entries)  
 ( 1.0,****) ( 1.0,-1.0) ( 1.0,-1.0)  
 (****,****) ( 3.0,****) ( 3.0,-1.0)  
 (****,****) (****,****) ( 5.0,****)  
  
b:  
 ( 4.0, 1.0)  
 ( 4.0, 6.0)  
 ( 4.0, 7.0)  
  
x:  
 ( 1.0, 0.0)  
 ( 0.0, 2.0)  
 ( 1.0, 0.0)  
  
Estimated reciprocal condition number of A: 0.011  
  
Backward error for system # 1: 0.74015E-16  
Forward error for system # 1: 0.40366E-13
```

UDU or LDL Factorization of a Hermitian Matrix

The subroutines described in this section compute a UDU or LDL factorization of a Hermitian matrix A. It is typical to follow a call to xHETRF with a call to xHESVX or xHETRS to solve a linear system $AX=B$, to xHECON to estimate the condition number of A, or to xHETRI to compute A^{-1} .

Calling Sequence

```
CALL ZHETRF (UPLO, N, ZA, LDA, IPIVOT, ZWORK, LDWORK, INFO)
CALL CHETRF (UPLO, N, CA, LDA, IPIVOT, CWORK, LDWORK, INFO)

void zhetrf (char uplo, int n, doublecomplex *za, int lda, int
            *ipivot, int *info)
void chetrf (char uplo, int n, complex *ca, int lda, int *ipivot,
            int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
xA	On entry, the upper or lower triangle of the matrix A. On exit, a UDU or LDL factorization of A.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
IPIVOT	On exit, an N-element array containing details of the row interchanges and block structure of D.
xWORK	Scratch array with a dimension of LDWORK.
LDWORK	Leading dimension of the array WORK as specified in a dimension or type statement. $LDWORK \geq 1$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where $i = INFO $, had an illegal value. INFO > 0 D(i,i), where $i = INFO$, is exactly zero, and D is therefore singular. The factorization has been completed, but division by zero will occur if this factorization is used to solve a linear system.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
DOUBLE PRECISION ZERO
INTEGER LDA, LDB, LDWORK, N, NRHS
PARAMETER (N = 3)
PARAMETER (LDA = N)
PARAMETER (LDB = N)
PARAMETER (LDWORK = 2 * N)
PARAMETER (NRHS = 1)
PARAMETER (ZERO = 0.0D0)

C
DOUBLE PRECISION ANORM, ANORM1, RCOND
COMPLEX*16 A(LDA,N), B(LDB,NRHS), WORK(LDWORK)
INTEGER ICOL, INFO, IPIVOT(N), IROW

C
EXTERNAL ZHECON, ZHETRF, ZHETRS
INTRINSIC ABS, CONJG, DBLE, MAX

C
C Initialize the array A to store the coefficient array A
C shown below. Initialize the array B to store the right
C hand side vector b shown below.
C
C      1      1-i  1-i      4+i
C  A = 1+i    3    3-i    b = 4+6i
C      1+i  3+i    5      4+7i
C
C      DATA A / (1.0D0,8D8), (8D8,8D8), (8D8,8D8),
C $              (1.0D0,-1.0D0), (3.0D0,8D8), (8D8,8D8),
C $              (1.0D0,-1.0D0), (3.0D0,-1.0D0), (5.0D0,8D8) /
C      DATA B / (4.0D0,1.0D0), (4.0D0,6.0D0), (4.0D0,7.0D0) /

C
C Print the initial value of the arrays.
C
C      PRINT 1000
C      DO 100, IROW = 1, N
C          PRINT 1010, (CONJG(A(ICOL,IROW)), ICOL = 1, IROW - 1),
C $              CMPLX(A(IROW,IROW), ZERO),
C $              (A(IROW,ICOL), ICOL = IROW + 1, N)
100 CONTINUE
C      PRINT 1020
C      DO 110, IROW = 1, N
C          PRINT 1010, (A(IROW,ICOL), ICOL = 1, N)
110 CONTINUE
C      PRINT 1030
C      DO 130, ICOL = 1, NRHS
C          DO 120, IROW = 1, N
C              PRINT 1010, B(IROW,ICOL)
120 CONTINUE
130 CONTINUE

C
C Compute the 1-norm of A. This will be used by ZHECON to

```

```

C      estimate the condition number of A.
C
      ANORM = 0.0D0
      DO 160, IROW = 1, N
        ANORM1 = 0.0D0
        DO 140, ICOL = 1, IROW - 1
          ANORM1 = ANORM1 + ABS(A(ICOL,IROW))
140      CONTINUE
        ANORM1 = ANORM1 + ABS(DBLE(A(IROW,IROW)))
        DO 150, ICOL = IROW + 1, N
          ANORM1 = ANORM1 + ABS(A(IROW,ICOL))
150      CONTINUE
        IF (ANORM .LT. ANORM1) THEN
          ANORM = ANORM1
        END IF
      160 CONTINUE
C
C      Factor the matrix.
C
      CALL ZHETRF ('UPPER TRIANGLE OF A STORED', N, A, LDA,
$             IPIVOT, WORK, LDWORK, INFO)
      IF (INFO .LT. 0) THEN
        PRINT 1040, ABS(INFO)
        STOP 1
      ELSE IF (INFO .GT. 0) THEN
        PRINT 1050
        STOP 2
      END IF
C
C      Estimate the condition number.  Print a warning if it is
C      unreasonably high.
C
      CALL ZHECON ('UPPER TRIANGULAR FACTOR OF A', N, A, LDA,
$             IPIVOT, ANORM, RCOND, WORK, INFO)
      IF (INFO .LT. 0) THEN
        PRINT 1060, ABS(INFO)
        STOP 3
      END IF
      PRINT 1070, RCOND
      IF ((1.0D0 + RCOND) .EQ. 1.0D0) THEN
        PRINT 1080
      END IF
C
C      Compute and print the solution.
C
      CALL ZHETRS ('UPPER TRIANGULAR FACTOR OF A', N, NRHS, A,
$             LDA, IPIVOT, B, LDB, INFO)
      IF (INFO .LT. 0) THEN
        PRINT 1090, ABS(INFO)
        STOP 4
      END IF
      PRINT 1100
      DO 210, ICOL = 1, NRHS
        DO 200, IROW = 1, N
          PRINT 1010, B(IROW,ICOL)
200      CONTINUE

```

```
210 CONTINUE
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (1X, 10(: 2X, '(', F4.1, ', ', F4.1, ')'))
1020 FORMAT (/1X, 'A in Hermitian form: (* in unused entries)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (1X, 'Illegal argument to ZHETRF, argument #', I2)
1050 FORMAT (1X, 'A is singular to working precision.')
1060 FORMAT (1X, 'Illegal argument to ZHECON, argument #', I2)
1070 FORMAT (/1X,
$      'Estimated reciprocal condition number of A: ',
$      F5.3)
1080 FORMAT (1X, 'A may be singular to working precision.')
1090 FORMAT (1X, 'Illegal argument to ZHETRS, argument #', I2)
1100 FORMAT (/1X, 'x:')
C
      END
```

Sample Output

```
A in full form:
( 1.0, 0.0) ( 1.0,-1.0) ( 1.0,-1.0)
( 1.0, 1.0) ( 3.0, 0.0) ( 3.0,-1.0)
( 1.0, 1.0) ( 3.0, 1.0) ( 5.0, 0.0)

A in Hermitian form: (* in unused entries)
( 1.0,****) ( 1.0,-1.0) ( 1.0,-1.0)
(****,****) ( 3.0,****) ( 3.0,-1.0)
(****,****) (****,****) ( 5.0,****)

b:
( 4.0, 1.0)
( 4.0, 6.0)
( 4.0, 7.0)

Estimated reciprocal condition number of A: 0.011

x:
( 1.0, 0.0)
( 0.0, 2.0)
( 1.0, 0.0)
```

Inverse of a UDU- or LDL-Factored Hermitian Matrix

The subroutines described in this section compute the inverse of a Hermitian matrix A, which has been UDU-factored or LDL-factored by xHETRF.

Calling Sequence

```
CALL ZHETRI (UPLO, N, ZA, LDA, IPIVOT, ZWORK, INFO)
CALL CHETRI (UPLO, N, CA, LDA, IPIVOT, CWORK, INFO)

void zhetri (char uplo, int n, doublecomplex *za, int lda, int
            *ipivot, int *info)
void chetri (char uplo, int n, complex *ca, int lda, int *ipivot,
            int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
xA	On entry, the UDU or LDL factorization of the matrix A, as computed by xHETRF. On exit, the Hermitian inverse of the matrix A. If UPLO = 'U' or 'u', the upper triangular part of the inverse is formed and the part of A below the diagonal is not used. If UPLO = 'L' or 'l' the lower triangular part of the inverse is formed and the part of A above the diagonal is not used.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
IPIVOT	Pivot indices as computed by xHETRF.
xWORK	Scratch array with a dimension of N.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where $i = INFO $, had an illegal value. INFO > 0 D(i,i), where $i = INFO$, is zero. The matrix is therefore singular and its inverse could not be computed.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
DOUBLE PRECISION ZERO
INTEGER LDA, LDWORK, N
PARAMETER (N = 3)
PARAMETER (LDA = N)
PARAMETER (LDWORK = 2 * N)
PARAMETER (ZERO = 0.0D0)

C
COMPLEX*16 A(LDA,N), WORK(LDWORK)
INTEGER ICOL, INFO, IPIVOT(N), IROW

C
EXTERNAL ZHETRF, ZHETRI
INTRINSIC ABS, CMPLX, CONJG, DBLE

C
C Initialize the array A to store the coefficient array A
C shown below.
C
C
C      1      1-i      1-i
C  A = 1+i      3      3-i
C      1+i      3+i      5
C
DATA A / (1.0D0,8D8), (8D8,8D8), (8D8,8D8),
$ (1.0D0,-1.0D0), (3.0D0,8D8), (8D8,8D8),
$ (1.0D0,-1.0D0), (3.0D0,-1.0D0), (5.0D0,8D8) /

C
C Print the initial value of A.
C
PRINT 1000
DO 100, IROW = 1, N
  PRINT 1010, (CONJG(A(ICOL,IROW)), ICOL = 1, IROW - 1),
$ CMPLX(DBLE(A(IROW,IROW)), ZERO),
$ (A(IROW,ICOL), ICOL = IROW + 1, N)
100 CONTINUE
PRINT 1020
DO 110, IROW = 1, N
  PRINT 1010, (A(IROW,ICOL), ICOL = 1, N)
110 CONTINUE

C
C Factor the matrix.
C
CALL ZHETRF ('UPPER TRIANGLE OF A STORED', N, A, LDA,
$ IPIVOT, WORK, LDWORK, INFO)
IF (INFO .LT. 0) THEN
  PRINT 1040, ABS(INFO)
  STOP 1
ELSE IF (INFO .GT. 0) THEN
  PRINT 1050
  STOP 2
END IF

C

```

```

C      Compute and print the inverse.
C
      CALL ZHETRI ('UPPER TRIANGULAR FACTOR OF A', N, A, LDA,
$        IPIVOT, WORK, INFO)
      IF (INFO .LT. 0) THEN
        PRINT 1060, ABS(INFO)
        STOP 3
      END IF
      PRINT 1070
      DO 200, IROW = 1, N
        PRINT 1010, (CONJG(A(ICOL,IROW)), ICOL = 1, IROW - 1),
$          (A(IROW,ICOL), ICOL = IROW, N)
      200 CONTINUE
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (1X, 10(: 2X, '(', F5.1, ', ', F5.1, ')'))
1020 FORMAT (/1X, 'A in Hermitian form: (* in unused entries)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (1X, 'Illegal argument to ZHETRF, argument #', I2)
1050 FORMAT (1X, 'A is singular to working precision.')
1060 FORMAT (1X, 'Illegal argument to ZHETRI, argument #', I2)
1070 FORMAT (/1X, 'A**(-1):')
C
      END

```

Sample Output

```

A in full form:
( 1.0, 0.0) ( 1.0, -1.0) ( 1.0, -1.0)
( 1.0, 1.0) ( 3.0, 0.0) ( 3.0, -1.0)
( 1.0, 1.0) ( 3.0, 1.0) ( 5.0, 0.0)

A in Hermitian form: (* in unused entries)
( 1.0, ***** ) ( 1.0, -1.0) ( 1.0, -1.0)
( *****, ***** ) ( 3.0, ***** ) ( 3.0, -1.0)
( *****, ***** ) ( *****, ***** ) ( 5.0, ***** )

A**(-1):
( 5.0, 0.0) ( -1.0, 3.0) ( -1.0, -1.0)
( -1.0, -3.0) ( 3.0, 0.0) ( -1.0, 1.0)
( -1.0, 1.0) ( -1.0, -1.0) ( 1.0, 0.0)

```

Solution to a Linear System in a UDU- or LDL-Factored Hermitian Matrix

The subroutines described in this section solve a linear system $AX = B$ for a Hermitian matrix A , which has been UDU-factored or LDL-factored by xHETRF, and general matrices B and X .

Calling Sequence

```
CALL ZHETRS (UPLO, N, NRHS, ZA, LDA, IPIVOT, ZB, LDB, INFO)
CALL CHETRS (UPLO, N, NRHS, CA, LDA, IPIVOT, CB, LDB, INFO)

void zhetrs (char uplo, int n, int nrhs, doublecomplex *za, int
            lda, int *ipivot, doublecomplex *zb, int ldb, int
            *info)
void chetrs (char uplo, int n, int nrhs, complex *ca, int lda, int
            *ipivot, complex *cb, int ldb, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
NRHS	Number of right-hand sides, equal to the number of columns of the matrix B. $NRHS \geq 0$.
xA	UDU or LDL factorization of the matrix A, as computed by xHETRF.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
IPIVOT	Pivot indices as computed by xHETRF.
xB	On entry, the $N \times NRHS$ right-hand side matrix B. On exit, the $N \times NRHS$ solution matrix X.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The i th argument, where $i = INFO $, had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
DOUBLE PRECISION ZERO
INTEGER LDA, LDB, LDWORK, N, NRHS
PARAMETER (N = 3)
PARAMETER (LDA = N)
PARAMETER (LDB = N)
PARAMETER (LDWORK = 2 * N)
PARAMETER (NRHS = 1)
PARAMETER (ZERO = 0.0D0)

C
DOUBLE PRECISION ANORM, ANORM1, RCOND
COMPLEX*16 A(LDA,N), B(LDB,NRHS), WORK(LDWORK)
INTEGER ICOL, INFO, IPIVOT(N), IROW

C
EXTERNAL ZHECON, ZHETRF, ZHETRS
INTRINSIC ABS, CMPLX, CONJG, DBLE, MAX

C
C Initialize the array A to store the coefficient array A
C shown below. Initialize the array B to store the right
C hand side vector b shown below.
C
C      1      1-i  1-i      4+i
C  A = 1+i    3    3-i    b = 4+6i
C      1+i    3+i  5      4+7i
C
C      DATA A / (1.0D0,8D8), (8D8,8D8), (8D8,8D8),
C $              (1.0D0,-1.0D0), (3.0D0,8D8), (8D8,8D8),
C $              (1.0D0,-1.0D0), (3.0D0,-1.0D0), (5.0D0,8D8) /
C      DATA B / (4.0D0,1.0D0), (4.0D0,6.0D0), (4.0D0,7.0D0) /

C
C Print the initial value of the arrays.
C
C      PRINT 1000
C      DO 100, IROW = 1, N
C          PRINT 1010, (CONJG(A(ICOL,IROW)), ICOL = 1, IROW - 1),
C $              CMPLX(A(IROW,IROW),ZERO),
C $              (A(IROW,ICOL), ICOL = IROW, N)
100 CONTINUE
C      PRINT 1020
C      DO 110, IROW = 1, N
C          PRINT 1010, (A(IROW,ICOL), ICOL = 1, N)
110 CONTINUE
C      PRINT 1030
C      DO 130, ICOL = 1, NRHS
C          DO 120, IROW = 1, N
C              PRINT 1010, B(IROW,ICOL)
120 CONTINUE
130 CONTINUE

C
C Compute the 1-norm of A. This will be used by ZHECON to

```



```
C      estimate the condition number of A.
C
      ANORM = 0.0D0
      DO 160, IROW = 1, N
        ANORM1 = 0.0D0
        DO 140, ICOL = 1, IROW - 1
          ANORM1 = ANORM1 + ABS(A(ICOL,IROW))
140      CONTINUE
        ANORM1 = ANORM1 + ABS(DBLE(A(IROW,IROW)))
        DO 150, ICOL = IROW + 1, N
          ANORM1 = ANORM1 + ABS(A(IROW,ICOL))
150      CONTINUE
        IF (ANORM .LT. ANORM1) THEN
          ANORM = ANORM1
        END IF
      160 CONTINUE
C
C      Factor the matrix.
C
      CALL ZHETRF ('UPPER TRIANGLE OF A STORED', N, A, LDA,
$             IPIVOT, WORK, LDWORK, INFO)
      IF (INFO .LT. 0) THEN
        PRINT 1040, ABS(INFO)
        STOP 1
      ELSE IF (INFO .GT. 0) THEN
        PRINT 1050
        STOP 2
      END IF
C
C      Estimate the condition number.  Print a warning if it is
C      unreasonably high.
C
      CALL ZHECON ('UPPER TRIANGULAR FACTOR OF A', N, A, LDA,
$             IPIVOT, ANORM, RCOND, WORK, INFO)
      IF (INFO .LT. 0) THEN
        PRINT 1060, ABS(INFO)
        STOP 3
      END IF
      PRINT 1070, RCOND
      IF ((1.0D0 + RCOND) .EQ. 1.0D0) THEN
        PRINT 1080
      END IF
C
C      Compute and print the solution.
C
      CALL ZHETRS ('UPPER TRIANGULAR FACTOR OF A', N, NRHS,
$             A, LDA, IPIVOT, B, LDB, INFO)
      IF (INFO .LT. 0) THEN
        PRINT 1090, ABS(INFO)
        STOP 4
      END IF
      PRINT 1100
      DO 210, ICOL = 1, NRHS
        DO 200, IROW = 1, N
          PRINT 1010, B(IROW,ICOL)
200      CONTINUE
```

```

210 CONTINUE
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (1X, 10(: 2X, '(', F4.1, ',', F4.1, ')'))
1020 FORMAT (/1X, 'A in Hermitian form: (* in unused entries)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (1X, 'Illegal argument to ZHETRF, argument #', I2)
1050 FORMAT (1X, 'A is singular to working precision.')
1060 FORMAT (1X, 'Illegal argument to ZHECON, argument #', I2)
1070 FORMAT (/1X,
$      'Estimated reciprocal condition number of A: ',
$      F5.3)
1080 FORMAT (1X, 'A may be singular to working precision.')
1090 FORMAT (1X, 'Illegal argument to ZHETRS, argument #', I2)
1100 FORMAT (/1X, 'x:')
C
      END

```

Sample Output

```

A in full form:
( 1.0, 0.0) ( 1.0, ****) ( 1.0, -1.0) ( 1.0, -1.0)
( 1.0, 1.0) ( 3.0, 0.0) ( 3.0, ****) ( 3.0, -1.0)
( 1.0, 1.0) ( 3.0, 1.0) ( 5.0, 0.0) ( 5.0, ****)

A in Hermitian form: (* in unused entries)
( 1.0, ****) ( 1.0, -1.0) ( 1.0, -1.0)
( ****, ****) ( 3.0, ****) ( 3.0, -1.0)
( ****, ****) ( ****, ****) ( 5.0, ****)

b:
( 4.0, 1.0)
( 4.0, 6.0)
( 4.0, 7.0)

Estimated reciprocal condition number of A: 0.011

x:
( 1.0, 0.0)
( 0.0, 2.0)
( 1.0, 0.0)

```

Reciprocal Condition Number of a UDU- or LDL-Factored Hermitian Matrix in Packed Storage

The subroutines described in this section estimate the reciprocal condition number, in either the 1-norm or the ∞ -norm, of a Hermitian matrix A in packed storage, which has been UDU-factored or LDL-factored by xHPTRF.

Calling Sequence

```
CALL ZHPCON (UPLO, N, ZA, IPIVOT, DANORM, DRCOND, ZWORK, INFO)
CALL CHPCON (UPLO, N, CA, IPIVOT, SANORM, SRCOND, CWORK, INFO)

void zhpcon (char uplo, int n, doublecomplex *za, int *ipivot,
             double danorm, double *drcond, int *info)
void chpcon (char uplo, int n, complex *ca, int *ipivot, float
             sanorm, float *srcond, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
xA	UDU or LDL factorization of the matrix A, as computed by xHPTRF. The dimension of xA is $(N \times N + N) / 2$.
IPIVOT	Pivot indices as computed by xHPTRF.
xANORM	The 1-norm or ∞ -norm of the original matrix A.
xRCOND	On exit, the estimated reciprocal of the condition number of the matrix A where the reciprocal condition number of A is defined to be $1 / (ANORM \times \ A^{-1}\)$. The reciprocal of the condition number is estimated instead of the condition number itself to avoid overflow or division by zero. If RCOND is less than machine precision (in particular, if RCOND = 0) then A is singular to working precision.
xWORK	Scratch array with a dimension of $2 \times N$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The i-th argument, where $i = INFO $, had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
DOUBLE PRECISION ZERO
INTEGER LENGTA, LDB, LDWORK, N, NRHS
PARAMETER (N = 3)
PARAMETER (LENGTA = (N * N + N) / 2)
PARAMETER (LDB = N)
PARAMETER (LDWORK = 2 * N)
PARAMETER (NRHS = 1)
PARAMETER (ZERO = 0.0D0)

C
DOUBLE PRECISION ANORM, RCOND
COMPLEX*16 A(LENGTA), B(LDB,NRHS), WORK(LDWORK)
INTEGER ICOL, INFO, IPIVOT(N), IROW

C
EXTERNAL ZHPCON, ZHPTRF, ZHPTRS
INTRINSIC ABS, CMPLX, CONJG, DBLE

C
C Initialize the array A to store the coefficient array A
C shown below. Initialize the array B to store the right
C hand side vector b shown below.
C
C      1      1-i  1-i      4+i
C  A = 1+i      3      3-i      b = 4+6i
C      1+i  3+i      5      4+7i
C
DATA A / (1.0D0,8D8), (1.0D0,-1.0D0), (3.0D0,8D8),
$      (1.0D0,-1.0D0), (3.0D0,-1.0D0), (5.0D0,8D8) /
DATA B / (4.0D0,1.0D0), (4.0D0,6.0D0), (4.0D0,7.0D0) /

C
C Print the initial value of the arrays.
C
PRINT 1000
PRINT 1010, CMPLX(DBLE(A(1)),ZERO), A(2), A(4)
PRINT 1010, CONJG(A(2)), CMPLX(DBLE(A(3)),ZERO), A(5)
PRINT 1010, CONJG(A(4)), CONJG(A(5)), CMPLX(DBLE(A(6))),
$      ZERO)
PRINT 1020
DO 130, ICOL = 1, NRHS
DO 120, IROW = 1, N
PRINT 1010, B(IROW,ICOL)
120 CONTINUE
130 CONTINUE

C
C Factor the matrix.
C
CALL ZHPTRF ('UPPER TRIANGLE OF A STORED', N, A, IPIVOT,
$      INFO)
IF (INFO .LT. 0) THEN
PRINT 1030, ABS(INFO)
STOP 1

```

```
        ELSE IF (INFO .GT. 0) THEN
            PRINT 1040
            STOP 2
        END IF
C
C      Estimate the condition number.  Print a warning if it is
C      unreasonably high.
C
        ANORM = 1.0D1
        CALL ZHPCON ('UPPER TRIANGULAR FACTOR OF A', N, A, IPIVOT,
$              ANORM, RCOND, WORK, INFO)
        IF (INFO .LT. 0) THEN
            PRINT 1050, ABS(INFO)
            STOP 3
        END IF
        PRINT 1060, RCOND
        IF ((1.0D0 + RCOND) .EQ. 1.0D0) THEN
            PRINT 1070
        END IF
C
C      Compute and print the solution.
C
        CALL ZHPTRS ('UPPER TRIANGULAR FACTOR OF A', N, NRHS, A,
$              IPIVOT, B, LDB, INFO)
        IF (INFO .LT. 0) THEN
            PRINT 1080, ABS(INFO)
            STOP 4
        END IF
        PRINT 1090
        DO 210, ICOL = 1, NRHS
            DO 200, IROW = 1, N
                PRINT 1010, B(IROW,ICOL)
            200    CONTINUE
        210    CONTINUE
C
1000    FORMAT (1X, 'A:')
1010    FORMAT (1X, 10(: 2X, '(', F4.1, ', ', F4.1, ')'))
1020    FORMAT (/1X, 'b:')
1030    FORMAT (1X, 'Illegal argument to ZHPTRF, argument #', I2)
1040    FORMAT (1X, 'A is singular to working precision.')
1050    FORMAT (1X, 'Illegal argument to ZHPCON, argument #', I2)
1060    FORMAT (/1X,
$          'Estimated reciprocal condition number of A: ',
$          F5.3)
1070    FORMAT (1X, 'A may be singular to working precision.')
1080    FORMAT (1X, 'Illegal argument to ZHPTRS, argument #', I2)
1090    FORMAT (/1X, 'x:')
C
        END
```

Sample Output

```
A:
( 1.0, 0.0) ( 1.0, -1.0) ( 1.0, -1.0)
( 1.0, 1.0) ( 3.0, 0.0) ( 3.0, -1.0)
( 1.0, 1.0) ( 3.0, 1.0) ( 5.0, 0.0)

b:
( 4.0, 1.0)
( 4.0, 6.0)
( 4.0, 7.0)

Estimated reciprocal condition number of A: 0.010

x:
( 1.0, 0.0)
( 0.0, 2.0)
( 1.0, 0.0)
```

Eigenvalues and Eigenvectors of a Hermitian Matrix in Packed Storage (Simple Driver)

The subroutines described in this section compute the eigenvalues and, optionally, the eigenvectors of a square Hermitian matrix *A* in packed storage. The eigenvectors are normalized to have a Euclidian norm of 1. Note that the expert driver xHPEVX is also available.

Calling Sequence

```
CALL ZHPEV (JOBZ, UPLO, N, ZA, DW, ZZ, LDZ, ZWORK, DWORK2, INFO)
CALL CHPEV (JOBZ, UPLO, N, CA, SW, CZ, LDZ, CWORK, SWORK2, INFO)

void zhpev (char jobz, char uplo, int n, doublecomplex *za, double
            *dw, doublecomplex *zz, int ldz, int *info)
void chpev (char jobz, char uplo, int n, complex *ca, float *sw,
            complex *cz, int ldz, int *info)
```

Arguments

JOBZ	Indicates whether to compute eigenvalues only or to compute both eigenvalues and eigenvectors. The legal values for JOBZ are listed below. Any values not listed below are illegal. 'N' or 'n' Compute eigenvalues only. 'V' or 'v' Compute both eigenvalues and eigenvectors.
UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
xA	On entry, the upper or lower triangle of the matrix A. The dimension of xA is $(N \times N + N) / 2$. On exit, A is overwritten.
xW	On exit, the N eigenvalues in ascending order.
xZ	On exit, if JOBZ = 'V' or 'v' then Z contains the orthonormal eigenvectors. If JOBZ = 'N' or 'n' then Z is not used.
LDZ	Leading dimension of the array Z as specified in a dimension or type statement. $LDZ \geq 1$. If JOBZ = 'V' or 'v' then $LDZ \geq \max(1, N)$.
xWORK	Scratch array with a dimension of $\max(1, 2 \times N - 1)$.
xWORK2	Scratch array with a dimension of $\max(1, 3 \times N - 2)$.

INFO On exit:
INFO = 0 Subroutine completed normally.
INFO < 0 The *i*th argument, where *i* = |INFO|, had an illegal value.
INFO > 0 Convergence failure.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C   INTEGER          LDEVAL, LDEVEC, LDWORK, LDWRK2, LENGTA, N
C   PARAMETER        (N = 3)
C   PARAMETER        (LDEVAL = N)
C   PARAMETER        (LDEVEC = N)
C   PARAMETER        (LDWORK = 2 * N - 1)
C   PARAMETER        (LDWRK2 = 3 * N - 2)
C   PARAMETER        (LENGTA = (N * N + N) / 2)
C
C   DOUBLE PRECISION EVALS(LDEVAL), WORK2(LDWRK2)
C   COMPLEX*16        A(LENGTA), EVECS(LDEVEC,N), WORK(LDWORK)
C   INTEGER           ICOL, INFO, IROW
C
C   EXTERNAL          ZHPEV
C   INTRINSIC         ABS, CONJG
C
C   Initialize the array A to store the coefficient array A
C   shown below.
C
C   4      4-4i    0
C   A = 4+4i  4      4-4i
C   0      4+4i    4
C
C   DATA A / (4.0D0,0.0D0), (4.0D0,-4.0D0), (4.0D0,0.0D0),
C   $          (0.0D0,0.0D0), (4.0D0,-4.0D0), (4.0D0,0.0D0) /
C
C   Print the initial value of A.
C
C   PRINT 1000
C   PRINT 1010, A(1),          A(2),          A(4)
C   PRINT 1010, CONJG(A(2)), A(3),          A(5)
C   PRINT 1010, CONJG(A(4)), CONJG(A(5)), A(6)
C
C   Compute the eigenvalues and eigenvectors.
C
C   CALL ZHPEV ('VALUES AND VECTORS, EIGEN',
C   $           'UPPER TRIANGLE OF A STORED', N, A, EVALS,
C   $           EVECS, LDEVEC, WORK, WORK2, INFO)
C   IF (INFO .LT. 0) THEN
C       PRINT 1020, ABS(INFO)
C       STOP 1
C   ELSE IF (INFO .GT. 0) THEN

```



```
        PRINT 1030, INFO
        STOP 2
      END IF
C
C      Print the eigenvalues and eigenvectors.
C
      PRINT 1040
      DO 200, IROW = 1, N
        PRINT 1050, EVALS(IROW), (EVECS(IROW,ICOL), ICOL = 1, N)
      200 CONTINUE
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (1X, 10(: 2X, '(', F5.1, ', ', F5.1, ')'))
1020 FORMAT (1X, 'Illegal argument to ZHPEV, argument #', I2)
1030 FORMAT (1X, 'Convergence failure, INFO = ', I2)
1040 FORMAT (/1X, 'Eigenvalue', 16X, 'Eigenvector**T')
1050 FORMAT (1X, F8.3, 6X, '[', 3('(', F4.1, ', ', F4.1, ') '),
$      ']' )
C
      END
```

Sample Output

```
A:
(  4.0,  0.0) (  4.0, -4.0) (  0.0,  0.0)
(  4.0,  4.0) (  4.0,  0.0) (  4.0, -4.0)
(  0.0,  0.0) (  4.0,  4.0) (  4.0,  0.0)

Eigenvalue      Eigenvector**T
-4.000          [(  0.0,  0.5) (  0.0,  0.7) (  0.0,  0.5) ]
 4.000          [(  0.5, -0.5) (  0.0,  0.0) (-0.5,  0.5) ]
12.000          [(-0.5,  0.0) (  0.7,  0.0) (-0.5,  0.0) ]
```

Eigenvalues and Eigenvectors of a Hermitian Matrix in Packed Storage (Expert Driver)

The subroutines described in this section compute the eigenvalues and, optionally, the eigenvectors of a Hermitian matrix A in packed storage. The eigenvectors are normalized to have a Euclidean norm of 1. Eigenvalues and eigenvectors can be selected by specifying either a range of values or a range of indices for the desired eigenvalues. Note that the simple driver xHPEV is also available.

Calling Sequence

```
CALL ZHPEVX (JOBZ, RANGE, UPLO, N, ZA, DVL, DVU, IL, IU, DABTOL,
             NFOUND, DW, ZZ, LDZ, ZWORK, DWORK2, IWORK3, IFAIL,
             INFO)
CALL CHPEVX (JOBZ, RANGE, UPLO, N, CA, SVL, SVU, IL, IU, SABTOL,
             NFOUND, SW, CZ, LDZ, CWORK, SWORK2, IWORK3, IFAIL,
             INFO)

void zhpevx (char jobz, char range, char uplo, int n, doublecomplex
             *za, double dvl, double dvu, int il, int iu, double
             dabtol, int *nfound, double *dw, doublecomplex *zz, int
             ldz, int *ifail, int *info)
void chpevx (char jobz, char range, char uplo, int n, complex *ca,
             float svl, float svu, int il, int iu, float sabtol, int
             *nfound, float *sw, complex *cz, int ldz, int *ifail,
             int *info)
```

Arguments

JOBZ	Indicates whether to compute eigenvalues only or to compute both eigenvalues and eigenvectors. The legal values for JOBZ are listed below. Any values not listed below are illegal. 'N' or 'n' Compute eigenvalues only. 'V' or 'v' Compute both eigenvalues and eigenvectors.
RANGE	Indicates whether to compute all eigenvalues or some subset. Legal values for RANGE are shown below. Any values not shown below are illegal. 'A' or 'a' All eigenvalues will be computed. 'I' or 'i' The ILth through IUth eigenvalues will be computed; see the descriptions of the IL and IU arguments below.

	'V' or 'v'	All eigenvalues in the half-open interval (VL,VU] will be computed; see the descriptions of the VL and VU arguments below.
UPLO		Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal.
	'U' or 'u'	xA contains the upper triangle.
	'L' or 'l'	xA contains the lower triangle.
N		Order of the matrix A. $N \geq 0$.
xA		On entry, the upper or lower triangle of the matrix A. The dimension of xA is $(N \times N + N) / 2$. On exit, A is overwritten.
xVL, xVU		If RANGE = 'V' or 'v' then only eigenvalues in the half-open interval (VL,VU] will be computed. Otherwise not used.
IL, IU		If RANGE = 'I' or 'i' then only the ILth through IUth eigenvalues will be computed. Otherwise not used. $1 \leq IL \leq IU \leq N$.
xABTOL		The absolute error tolerance for the eigenvalues. An approximate eigenvalue is considered to have converged when it lies in an interval [a,b] of width less than or equal to $ABTOL + \epsilon \times \max(a , b)$ where ϵ is machine precision. If $ABTOL \leq 0$ then $\epsilon \times \ T\ _1$ will be used in its place, where $\ T\ _1$ is the 1-norm of the tridiagonal matrix obtained by reducing A to tridiagonal form.
NFOUND		On exit, the total number of eigenvalues found. $0 \leq NFOUND \leq N$. If RANGE = 'A' or 'a' then $NFOUND = N$. If RANGE = 'I' or 'i' then $NFOUND = IU - IL + 1$.
xW		On exit, W(1:NFOUND) contains the computed eigenvalues in ascending order.
xZ		On exit, if JOBZ = 'V' or 'v' then the first NFOUND columns of Z contain the orthonormal eigenvectors corresponding to the selected eigenvalues. If an eigenvector fails to converge then the corresponding column of Z contains an approximation to that eigenvector and its index is returned in IFAIL. If JOBZ = 'N' or 'n' then Z is not used. Note that Z must have at least NFOUND columns. If RANGE = 'V' or 'v' then NFOUND may not be known in advance and an upper bound must be used. For A(LDZ,K) where $NFOUND > K$, the eigenvectors will be stored into memory with unpredictable results.
LDZ		Leading dimension of the array Z as specified in a dimension or type statement. $LDZ \geq 1$. If JOBZ = 'V' or 'v' then $LDZ \geq \max(1, N)$.
xWORK		Scratch array with a dimension of $2 \times N$.
xWORK2		Scratch array with a dimension of $7 \times N$.
IWORK3		Scratch array with a dimension of $5 \times N$.

IFAIL	On exit, an N-element array that indicates which eigenvectors that failed to converge. If JOBZ = 'V' or 'v' and INFO = 0 then all N elements contain zero. If INFO > 0 then IFAIL contains the indices of the eigenvectors that failed to converge. If JOBZ = 'N' or 'n' then IFAIL is not used.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where i = INFO , had an illegal value. INFO > 0 There were i eigenvectors, where i = INFO, that failed to converge. Their indices are stored in IFAIL.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
DOUBLE PRECISION ZERO
INTEGER LDEVAL, LDEVEC, LDIWRK, LDWORK, LDWRK2
INTEGER LENGTA, N
PARAMETER (N = 3)
PARAMETER (LDEVAL = N)
PARAMETER (LDEVEC = N)
PARAMETER (LDIWRK = 5 * N)
PARAMETER (LDWORK = 2 * N)
PARAMETER (LDWRK2 = 7 * N)
PARAMETER (LENGTA = (N * N + N) / 2)
PARAMETER (ZERO = 0.0D0)
C
DOUBLE PRECISION EVALS(LDEVAL), TEMP, WORK2(LDWRK2)
COMPLEX*16 A(LENGTA), EVECS(LDEVEC,N), WORK(LDWORK)
INTEGER ICOL, IFAIL(N), INFO, IROW, ITEMP
INTEGER IWORK(LDIWRK), NFOUND
C
EXTERNAL ZHPEVX
INTRINSIC CONJG
C
C Initialize the array A to store the coefficient array A
C shown below.
C
C      4      4-4i      0
C      A = 4+4i      4      4-4i
C      0      4+4i      4
C
C      DATA A / (4.0D0,0.0D0), (4.0D0,-4.0D0), (4.0D0,0.0D0),
C $              (0.0D0,0.0D0), (4.0D0,-4.0D0), (4.0D0,0.0D0) /
C      DATA A / (4.0D0,8D8), (4.0D0,-4.0D0), (4.0D0,8D8),
C $              (0.0D0,0.0D0), (4.0D0,-4.0D0), (4.0D0,8D8) /
C
C Print the initial value of A.
C
C

```

```

      PRINT 1000
      PRINT 1010, DBLE(A(1)), ZERO, A(2), A(4)
      PRINT 1010, CONJG(A(2)), DBLE(A(3)), ZERO, A(5)
      PRINT 1010, CONJG(A(4)), CONJG(A(5)), DBLE(A(6)), ZERO
C
C      Compute the eigenvalues and eigenvectors.
C
      CALL ZHPEVX ('VALUES AND VECTORS', 'ALL VALUES',
$              'UPPER TRIANGLE OF A STORED', N, A, TEMP, TEMP,
$              ITEMP, ITEMP, 0.0D0, NFOUND, EVALS, EVECS,
$              LDEVEC, WORK, WORK2, IWORK, IFAIL, INFO)
      IF (INFO .LT. 0) THEN
          PRINT 1020, ABS(INFO)
          STOP 1
      ELSE IF (INFO .GT. 0) THEN
          PRINT 1030, INFO
          STOP 2
      END IF
C
C      Print the eigenvalues and eigenvectors.
C
      PRINT 1040
      DO 200, IROW = 1, N
          PRINT 1050, EVALS(IROW), (EVECS(IROW,ICOL), ICOL = 1, N)
      200 CONTINUE
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (1X, 10(: 2X, '(', F5.1, ', ', F5.1, ')'))
1020 FORMAT (1X, 'Illegal argument to ZHPEVX, argument #', I2)
1030 FORMAT (1X, 'Convergence failure, INFO = ', I2)
1040 FORMAT (/1X, 'Eigenvalue', 16X, 'Eigenvector**T')
1050 FORMAT (1X, F8.3, 6X, '[', 3('(', F4.1, ', ', F4.1, ') '),
$          ']'')
C
      END

```

Sample Output

```

A:
(  4.0,  0.0) (  4.0, -4.0) (  0.0,  0.0)
(  4.0,  4.0) (  4.0,  0.0) (  4.0, -4.0)
(  0.0,  0.0) (  4.0,  4.0) (  4.0,  0.0)

Eigenvalue      Eigenvector**T
-4.000          [(  0.0,  0.5) (  0.0,  0.7) (  0.0,  0.5) ]
 4.000          [(  0.5, -0.5) (  0.0,  0.0) (-0.5,  0.5) ]
12.000         [(-0.5,  0.0) (  0.7,  0.0) (-0.5,  0.0) ]

```

Generalized Eigenvalues and Eigenvectors for Hermitian Matrices in Packed Storage

The subroutines in this section compute the generalized eigenvalues (α , β) for a pair of square Hermitian matrices A and B in packed storage. Optionally, these subroutines can also compute the generalized eigenvectors.

A generalized eigenvalue is a ratio $\lambda = \alpha / \beta$ such that $AX = \lambda BX$ or $\det(A - \lambda B) = 0$ for general matrices A, B, and X. λ is typically represented as a ratio rather than as a scalar because there are reasonable interpretations for $\alpha = 0$, $\beta = 0$, and for $\alpha = \beta = 0$. A right generalized eigenvector x corresponding to a generalized eigenvalue λ is defined by $(A - \lambda B)x = 0$. A left generalized eigenvector x corresponding to a generalized eigenvalue λ is defined by $(A - \lambda B)Hx = 0$. A good reference for generalized eigenproblems is the book *Matrix Computations*, 2nd. ed. by Golub and van Loan (1989, The Johns Hopkins University Press).

Calling Sequence

```
CALL ZHPGV (ITYPE, JOBZ, UPLO, N, ZA, ZB, DW, ZZ, LDZ, ZWORK,
            DWORK2, INFO)
CALL CHPGV (ITYPE, JOBZ, UPLO, N, CA, CB, SW, CZ, LDZ, CWORK,
            SWORK2, INFO)

void zhpv (int itype, char jobz, char uplo, int n, doublecomplex
          *za, doublecomplex *zb, double *dw, doublecomplex *zz,
          int ldz, int *info)
void chpv (int itype, char jobz, char uplo, int n, complex *ca,
          complex *cb, float *sw, complex *cz, int ldz, int
          *info)
```

Arguments

ITYPE	Indicates the problem to be solved. Legal values for ITYPE are shown below. Any values not shown below are illegal.
1	$Ax = \lambda Bx$
2	$ABx = \lambda x$
3	$BAx = \lambda x$
JOBZ	Indicates whether to compute eigenvalues only or to compute both eigenvalues and eigenvectors. The legal values for JOBZ are listed below. Any values not listed below are illegal.

	'N' or 'n'	Compute eigenvalues only.
	'V' or 'v'	Compute both eigenvalues and eigenvectors.
UPLO	Indicates whether xA and xB contain the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal.	
	'U' or 'u'	xA and xB both contain the upper triangle.
	'L' or 'l'	xA and xB both contain the lower triangle.
N	Order of the matrix A. $N \geq 0$.	
xA	On entry, the upper or lower triangle of the matrix A. The dimension of xA is $(N \times N + N) / 2$. On exit, A is overwritten.	
xB	On entry, the upper or lower triangle of the matrix B. The dimension of xB is $(N \times N + N) / 2$. On exit, if UPLO = 'U' or 'u' then B contains the matrix U from the Cholesky factorization UHU of B. If UPLO = 'L' or 'l' then B contains the matrix L from the Cholesky factorization LLH of B.	
xW	On exit, W(1:NFOUND) contains the computed eigenvalues in ascending order.	
xZ	On exit, if JOBZ = 'V' or 'v' then xZ contains the matrix Z of eigenvectors normalized as shown below:	
	ITYPE	Eigenvectors stored in Z
	1 or 2	$Z^H \times B \times Z = I$
	3	$Z^H \times B^{-1} \times Z = I$
	If JOBZ = 'N' or 'n' then Z is not used.	
LDZ	Leading dimension of the array Z as specified in a dimension or type statement. $LDZ \geq 1$. If JOBZ = 'V' or 'v' then $LDZ \geq \max(1, N)$.	
xWORK	Scratch array with a dimension of $\max(1, 2 \times N - 1)$.	
xWORK2	Scratch array with a dimension of $\max(1, 3 \times N - 2)$.	
INFO	On exit:	
	INFO = 0	Subroutine completed normally.
	INFO < 0	The <i>i</i> th argument, where $i = \text{INFO} $, had an illegal value.
	$0 < \text{INFO} \leq N$	Convergence failure
	INFO > N	The leading minor of order <i>i</i> of B, where $i = \text{INFO} - N$, is not positive definite.

Sample Program

```
PROGRAM TEST
IMPLICIT NONE
C
DOUBLE PRECISION ZERO
```

```

      INTEGER          ITYPE, LDEVEC, LDWORK, LDWRK2, LENGTA
      INTEGER          LENGTB, N
      PARAMETER        (ITYPE = 1)
      PARAMETER        (N = 3)
      PARAMETER        (LDEVEC = N)
      PARAMETER        (LDWORK = 2 * N - 1)
      PARAMETER        (LDWRK2 = 3 * N - 2)
      PARAMETER        (LENGTA = (N * N + N) / 2)
      PARAMETER        (LENGTB = (N * N + N) / 2)
      PARAMETER        (ZERO = 0.0D0)
C
      DOUBLE PRECISION EVALS(N), WORK2(LDWRK2)
      COMPLEX*16        A(LENGTA), B(LENGTB), EVECS(LDEVEC,N)
      COMPLEX*16        WORK(LDWORK)
      INTEGER          ICOL, INFO, IROW
C
      EXTERNAL          ZHPGV
      INTRINSIC         ABS, CMPLX, CONJG, DBLE
C
C      Initialize the arrays A and B to store the matrices A and B
C      shown below.
C
C      A =      8      2-2i  2-2i      B =      4      1-i  1-i
C      A = 2+2i    16     4-4i      B = 1+i      8      2-2i
C      A = 2+2i    4+4i    32      B = 1+i      2+2i    16
C
      DATA A / (8.0D0,8D8), (2.0D0,-2.0D0), (1.6D1,8D8),
$              (2.0D0,-2.0D0), (4.0D0,-4.0D0), (3.2D1,8D8) /
      DATA B / (4.0D0,8D8), (1.0D0,-1.0D0), (8.0D0,8D8),
$              (1.0D0,-1.0D0), (2.0D0,-2.0D0), (1.6D1,8D8) /
C
      PRINT 1000
      PRINT 1010, CMPLX(DBLE(A(1)),ZERO), A(2), A(4)
      PRINT 1010, CONJG(A(2)), CMPLX(DBLE(A(3)),ZERO), A(5)
      PRINT 1010, CONJG(A(4)), CONJG(A(5)), CMPLX(DBLE(A(6)),ZERO)
      PRINT 1020
      PRINT 1010, CMPLX(DBLE(B(1)),ZERO), B(2), B(4)
      PRINT 1010, CONJG(B(2)), CMPLX(DBLE(B(3)),ZERO), B(5)
      PRINT 1010, CONJG(B(4)), CONJG(B(5)), CMPLX(DBLE(B(6)),ZERO)
C
      CALL ZHPGV (ITYPE, 'VALUES AND EIGENVECTORS',
$              'UPPER TRIANGLES OF A AND B STORED', N, A, B,
$              EVALS, EVECS, LDVEC, WORK, WORK2, INFO)
      IF (INFO .LT. 0) THEN
        PRINT 1030, ABS(INFO)
        STOP 1
      ELSE IF (INFO .GT. N) THEN
        PRINT 1040, INFO
        STOP 2
      ELSE IF (INFO .GT. 0) THEN
        PRINT 1050, INFO
        STOP 3
      END IF
      PRINT 1060
      DO 120, IROW = 1, N
        PRINT 1070, EVALS(IROW), (EVECS(IROW,ICOL), ICOL = 1, N)

```



```
120 CONTINUE
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (3(3X, '(', F5.2, ', ', F5.2, ')'))
1020 FORMAT (/1X, 'B:')
1030 FORMAT (1X, 'Illegal argument to ZHPGV, argument #', I2)
1040 FORMAT (1X, 'B is not positive definite, INFO = ', I4)
1050 FORMAT (1X, 'Convergence failure, INFO = ', I4)
1060 FORMAT (/1X, 'Eigenvalue', 19X, 'Eigenvector**T')
1070 FORMAT (1X, F8.3, 5X, '[', 3('(', F5.3, ', ', F5.3, ')', 2X),
$      ']' )
C
      END
```

Sample Output

```
A:
( 8.00, 0.00)  ( 2.00,-2.00)  ( 2.00,-2.00)
( 2.00, 2.00)  (16.00, 0.00)  ( 4.00,-4.00)
( 2.00, 2.00)  ( 4.00, 4.00)  (32.00, 0.00)

B:
( 4.00, 0.00)  ( 1.00,-1.00)  ( 1.00,-1.00)
( 1.00, 1.00)  ( 8.00, 0.00)  ( 2.00,-2.00)
( 1.00, 1.00)  ( 2.00, 2.00)  (16.00, 0.00)

Eigenvalue      Eigenvector**T
2.000    [(0.500,0.000) (-.091,0.091) (-.070,0.035) ]
2.000    [(0.000,0.000) (0.365,0.000) (-.052,0.070) ]
2.000    [(0.000,0.000) (0.000,0.000) (0.261,0.000) ]
```

Refined Solution to a Linear System in a UDU- or LDL-Factored Hermitian Matrix in Packed Storage

The subroutines described in this section refine the solution to a linear system $AX = B$ for a Hermitian matrix A in packed storage, which has been UDU-factored or LDL-factored by xHPTRF, and general matrices B and X . These subroutines also compute forward error bounds and backward error estimates for the refined solution.

Calling Sequence

```
CALL ZHPRFS (UPLO, N, NRHS, ZA, ZAF, IPIVOT, ZB, LDB, ZX, LDX,
             DFERR, DBERR, ZWORK, DWORK2, INFO)
CALL CHPRFS (UPLO, N, NRHS, CA, CAF, IPIVOT, CB, LDB, CX, LDX,
             SFERR, SBERR, CWORK, SWORK2, INFO)

void zhprfs (char uplo, int n, int nrhs, doublecomplex *za,
             doublecomplex *zaf, int *ipivot, doublecomplex *zb,
             int ldb, doublecomplex *zx, int ldx, double *dferr,
             double *dberr, int *info)
void chprfs (char uplo, int n, int nrhs, complex *ca, complex *caf,
             int *ipivot, complex *cb, int ldb, complex *cx, int
             ldx, float *sferr, float *sberr, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
NRHS	Number of right-hand sides, equal to the number of columns of the matrices B and X. $NRHS \geq 0$.
xA	Upper or lower triangle of the matrix A. The dimension of xA is $(N \times N + N) / 2$.
xAF	UDU or LDL factorization of the matrix A as computed by xHPTRF. The dimension of xAF is $(N \times N + N) / 2$.
IPIVOT	Pivot indices as computed by xHPTRF.
xB	The $N \times NRHS$ right-hand side matrix B.

LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
xX	On entry, the $N \times NRHS$ solution matrix X. On exit, the refined $N \times NRHS$ solution matrix X.
LDX	Leading dimension of the array X as specified in a dimension or type statement. $LDX \geq \max(1, N)$.
xFERR	On exit, the estimated forward error bound for each solution vector $X(*, j)$ for $1 \leq j \leq NRHS$. If X' is the true solution corresponding to $X(*, j)$ then $FERR(j)$ is an upper bound on the magnitude of the largest element in $X(*, j) - X'$ divided by the magnitude of the largest element in $X(*, j)$.
xBERR	On exit, $BERR(j)$ is the smallest relative change in any element of A or $B(*, j)$ that makes $X(*, j)$ an exact solution to $AX(*, j) = B(*, j)$ for $1 \leq j \leq NRHS$.
xWORK	Scratch array with a dimension of $2 \times N$.
xWORK2	Scratch array with a dimension of N.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The <i>i</i> th argument, where $i = INFO $, had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
DOUBLE PRECISION  ZERO
INTEGER           LDB, LDB1, LDWORK, LDWRK2, LDX, LENGTA
INTEGER           N, NRHS
PARAMETER         (N = 3)
PARAMETER         (LDB = N)
PARAMETER         (LDB1 = N)
PARAMETER         (LDWORK = 2 * N)
PARAMETER         (LDWRK2 = N)
PARAMETER         (LDX = LDB)
PARAMETER         (LENGTA = (N * N + N) / 2)
PARAMETER         (NRHS = 1)
PARAMETER         (ZERO = 0.0D0)

C
DOUBLE PRECISION  ANORM, BERR(N), FERR(N), RCOND
DOUBLE PRECISION  WORK2(LDWRK2)
COMPLEX*16        A(LENGTA), AF(LENGTA), B(LDB,NRHS)
COMPLEX*16        B1(LDB,NRHS), WORK(LDWORK), X(LDX,NRHS)
INTEGER           ICOL, INFO, IPIVOT(N), IROW

C
EXTERNAL          ZCOPY, ZHPCON, ZHPMV, ZHPRFS, ZHPTRF
EXTERNAL          ZHPTRS

```

```

C      INTRINSIC      ABS, CMPLX, CONJG, DBLE, MAX
C
C      Initialize the array A to store the coefficient array A
C      shown below. Initialize the array B to store the right
C      hand side vector b shown below.
C
C      A = (1e+16, 0)      (1e+16, 1)      (3e+16, 3e-16)
C      (1e+16, -1)      (1e+16, 0)      (3e+26, 3e-16)
C      (3e+16, 3e-16) (3e+26, -3e-16)      (7e-16, 0)
C
C      b = (2e-26, 2e+16)
C      (2e+26, 2e-16)
C      (2, 2)
C
C      DATA A / (1.0D16,8D8), (1.0D16,1.0D0), (1.0D16,8D8),
$      (3.0D16,3.0D-16), (3.0D26,3.0D-16), (7.0D-16,8D8) /
C      DATA B / (2.0D-26,2.0D16), (2.0D26,2.0D-16), (2.0D0,2.0D0) /
C
C      Print the initial value of the arrays.
C
C      PRINT 1000
C      PRINT 1010, CMPLX(DBLE(A(1)),ZERO), A(2), A(4)
C      PRINT 1010, CONJG(A(2)), CMPLX(DBLE(A(3)),ZERO), A(5)
C      PRINT 1010, CONJG(A(4)), CONJG(A(5)), CMPLX(DBLE(A(6)),ZERO)
C      DO 110, ICOL = 1, NRHS
C      PRINT 1020, ICOL
C      DO 100, IROW = 1, N
C      PRINT 1100, B(IROW,ICOL)
100    CONTINUE
110    CONTINUE
C
C      UDU factor A.
C
C      CALL ZCOPY (LENGTHA, A, 1, AF, 1)
C      CALL ZHPTRF ('UPPER TRIANGLE OF A STORED', N, AF, IPIVOT,
$      INFO)
C      IF (INFO .LT. 0) THEN
C      PRINT 1030, ABS(INFO)
C      STOP 1
C      ELSE IF (INFO .GT. 0) THEN
C      PRINT 1040
C      STOP 2
C      END IF
C
C      Estimate the condition number. Print a warning if it is
C      unreasonably high.
C
C      ANORM = 3.0D26
C      CALL ZHPCON ('UPPER TRIANGULAR FACTOR OF A', N, AF, IPIVOT,
$      ANORM, RCOND, WORK, INFO)
C      IF (INFO .LT. 0) THEN
C      PRINT 1050, ABS(INFO)
C      STOP 3
C      END IF
C      PRINT 1060, RCOND
C      IF ((1.0D0 + RCOND) .EQ. 1.0D0) THEN

```

```

        PRINT 1070
        END IF
C
C      Compute and print the solution.
C
        CALL ZCOPY (LDB * NRHS, B, 1, X, 1)
        CALL ZHPTRS ('UPPER TRIANGULAR FACTOR OF A', N, NRHS, AF,
$          IPIVOT, X, LDX, INFO)
        IF (INFO .LT. 0) THEN
            PRINT 1080, ABS(INFO)
            STOP 4
        END IF
        DO 210, ICOL = 1, NRHS
            PRINT 1090, ICOL
            DO 200, IROW = 1, N
                PRINT 1100, X(IROW,ICOL)
200      CONTINUE
210      CONTINUE
        DO 220, ICOL = 1, NRHS
            CALL ZHPMV ('UPPER TRIANGLE OF A STORED', N,
$              (1.0D0,0.0D0), A, X(1,ICOL), 1, (0.0D0,0.0D0),
$              B1(1,ICOL), 1)
220      CONTINUE
            DO 240, ICOL = 1, NRHS
                PRINT 1110, ICOL
                DO 230, IROW = 1, N
                    PRINT 1100, B1(IROW,ICOL)
230      CONTINUE
240      CONTINUE
C
C      Refine the solution and print the refined solution with the
C      error bounds.
C
        CALL ZHPRFS ('UPPER TRIANGLE OF A STORED', N, NRHS, A, AF,
$          IPIVOT, B, LDB, X, LDX, FERR, BERR, WORK,
$          WORK2, INFO)
        IF (INFO .LT. 0) THEN
            PRINT 1120, ABS(INFO)
            STOP 5
        END IF
        DO 310, ICOL = 1, NRHS
            PRINT 1130, ICOL
            DO 300, IROW = 1, N
                PRINT 1100, X(IROW,ICOL)
300      CONTINUE
310      CONTINUE
        DO 320, ICOL = 1, NRHS
            CALL ZHPMV ('UPPER TRIANGLE OF A STORED', N,
$              (1.0D0,0.0D0), A, X(1,ICOL), 1, (0.0D0,0.0D0),
$              B1(1,ICOL), 1)
320      CONTINUE
            DO 340, ICOL = 1, NRHS
                PRINT 1140, ICOL
                DO 330, IROW = 1, N
                    PRINT 1100, B1(IROW,ICOL)
330      CONTINUE

```

```

340 CONTINUE
    DO 350, IROW = 1, NRHS
        PRINT 1150, IROW, BERR(IROW)
        PRINT 1160, IROW, FERR(IROW)
350 CONTINUE
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (1X, 10(: 1X, '(', E8.2, ', ', ', E8.2, ')'))
1020 FORMAT (/1X, 'b for system #', I1)
1030 FORMAT (1X, 'Illegal argument to ZHPTRF, argument #', I2)
1040 FORMAT (1X, 'A is singular to working precision.')
1050 FORMAT (1X, 'Illegal argument to ZHPCON, argument #', I2)
1060 FORMAT (/1X,
    $      'Estimated reciprocal condition number of A: ',
    $      E9.3)
1070 FORMAT (1X, 'A may be singular to working precision.')
1080 FORMAT (1X, 'Illegal argument to ZHPTRS, argument #', I2)
1090 FORMAT (/1X, 'Initial xb for system #', I1)
1100 FORMAT (3X, '(', E20.14, ', ', ', E20.14, ')')
1110 FORMAT (/1X, 'Ax with initial x for system #', I1)
1120 FORMAT (1X, 'Illegal argument to ZHPRFS, argument #', I2)
1130 FORMAT (/1X, 'Refined xb for system #', I1)
1140 FORMAT (/1X, 'Ax with refined x for system #', I1)
1150 FORMAT (/1X, 'Backward error for system #', I1, ': ', E12.6)
1160 FORMAT (1X, 'Forward error for system #', I1, ': ', E12.6)
C
    END

```

Sample Output

```

A:
(0.10E+17, 0.00E+00) (0.10E+17, 0.10E+01) (0.30E+17, 0.30E-15)
(0.10E+17, -.10E+01) (0.10E+17, 0.00E+00) (0.30E+27, 0.30E-15)
(0.30E+17, -.30E-15) (0.30E+27, -.30E-15) (0.70E-15, 0.00E+00)

b for system #1
(0.2000000000000000E-25, 0.2000000000000000E+17)
(0.2000000000000000E+27, 0.2000000000000000E-15)
(0.2000000000000000E+01, 0.2000000000000000E+01)

Estimated reciprocal condition number of A: 0.333E-10

Initial xb for system #1
(-.200000000004000E+01, 0.200000000004000E+01)
(0.200000000004000E-09, -.200000000004000E-09)
(0.66666666673333E+00, -.66666666673333E-10)

Ax with initial x for system #1
(0.4000000000000000E+01, 0.2000000000000000E+17)
(0.2000000000000000E+27, 0.4000000000000000E+01)
(-.8000000000000000E+01, 0.8000000000000000E+01)

```

```
Refined xb for system #1
(-.200000000004000E+01, 0.200000000004000E+01)
(0.200000000004000E-09, -.200000000004000E-09)
(0.66666666673333E+00, -.66666666673333E-10)

Ax with refined x for system #1
(0.400000000000000E+01, 0.200000000000000E+17)
(0.200000000000000E+27, 0.400000000000000E+01)
(-.800000000000000E+01, 0.800000000000000E+01)

Backward error for system #1: 0.666667E-16
Forward error for system #1: 0.143228E-14
```

Solution to a Linear System in a Hermitian Matrix in Packed Storage (Simple Driver)

The subroutines described in this section solve a linear system $AX = B$ for a Hermitian matrix A in packed storage and general matrices B and X . Note that the expert driver `xHPSVX` is also available.

Calling Sequence

```
CALL ZHPSV  (UPLO, N, NRHS, ZA, IPIVOT, ZB, LDB, INFO)
CALL CHPSV  (UPLO, N, NRHS, CA, IPIVOT, CB, LDB, INFO)

void zhpsv  (char uplo, int n, int nrhs, doublecomplex *za, int
             *ipivot, doublecomplex *zb, int ldb, int *info)
void chpsv  (char uplo, int n, int nrhs, complex *ca, int *ipivot,
             complex *cb, int ldb, int *info)
```

Arguments

UPLO	Indicates whether <code>xA</code> contains the upper or lower triangle of the matrix. The legal values for <code>UPLO</code> are listed below. Any values not listed below are illegal. 'U' or 'u' <code>xA</code> contains the upper triangle. 'L' or 'l' <code>xA</code> contains the lower triangle.
N	Order of the matrix A . $N \geq 0$.
NRHS	Number of right-hand sides, equal to the number of columns of the matrix B . $NRHS \geq 0$.
xA	On entry, the upper or lower triangle of the matrix A . The dimension of <code>xA</code> is $(N \times N + N) / 2$. On exit, the UDU or LDL factorization of A as computed by <code>xHPTRF</code> .
IPIVOT	On exit, pivot indices as computed by <code>xHPTRF</code> .
xB	On entry, the $N \times NRHS$ right-hand side matrix B . On exit, the $N \times NRHS$ solution matrix X .
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The i th argument, where $i = INFO $, had an illegal value.

INFO > 0 D(i,i), where i = INFO, is exactly zero, and D is therefore singular. The UDU or LDL factorization has been completed, but the solution could not be computed.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
  INTEGER      LDB, LDWORK, LENGTA, N, NRHS
  PARAMETER    (N = 3)
  PARAMETER    (LENGTA = (N * N + N) / 2)
  PARAMETER    (LDB = N)
  PARAMETER    (LDWORK = 1)
  PARAMETER    (NRHS = 1)
C
  COMPLEX*16   A(LENGTA), B(LDB,NRHS)
  INTEGER      ICOL, INFO, IPIVOT(N), IROW
C
  EXTERNAL     ZHPSV
  INTRINSIC    ABS, CONJG
C
C   Initialize the array A to store the coefficient array A
C   shown below. Initialize the array B to store the right
C   hand side vector b shown below.
C
C   1      1-i   1-i      4+i
C   A = 1+i   3    3-i     b = 4+6i
C   1+i   3+i   5        4+7i
C
  DATA A / (1.0D0,0.0D0), (1.0D0,-1.0D0), (3.0D0,0.0D0),
$          (1.0D0,-1.0D0), (3.0D0,-1.0D0), (5.0D0,0.0D0) /
  DATA B / (4.0D0,1.0D0), (4.0D0,6.0D0), (4.0D0,7.0D0) /
C
C   Print the initial value of the arrays.
C
  PRINT 1000
  PRINT 1010, A(1),      A(2),      A(4)
  PRINT 1010, CONJG(A(2)), A(3),      A(5)
  PRINT 1010, CONJG(A(4)), CONJG(A(5)), A(6)
  PRINT 1020
  DO 130, ICOL = 1, NRHS
    DO 120, IROW = 1, N
      PRINT 1010, B(IROW,ICOL)
120    CONTINUE
130  CONTINUE
C
C   Compute and print the solution.
C
  CALL ZHPSV ('UPPER TRIANGLE OF A STORED', N, NRHS, A,
```

```

$          IPIVOT, B, LDB, INFO)
  IF (INFO .LT. 0) THEN
    PRINT 1030, ABS(INFO)
    STOP 1
  ELSE IF (INFO .GT. 0) THEN
    PRINT 1040
    STOP 2
  END IF
  PRINT 1050
  DO 210, ICOL = 1, NRHS
    DO 200, IROW = 1, N
      PRINT 1010, B(IROW,ICOL)
    200  CONTINUE
  210  CONTINUE
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (1X, 10(: 2X, '(', F4.1, ', ', F4.1, ')'))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (1X, 'Illegal argument to ZHPSV, argument #', I2)
1040 FORMAT (1X, 'A is singular to working precision.')
1050 FORMAT (/1X, 'x:')
C
      END

```

Sample Output

```

A:
( 1.0, 0.0) ( 1.0, -1.0) ( 1.0, -1.0)
( 1.0, 1.0) ( 3.0, 0.0) ( 3.0, -1.0)
( 1.0, 1.0) ( 3.0, 1.0) ( 5.0, 0.0)

b:
( 4.0, 1.0)
( 4.0, 6.0)
( 4.0, 7.0)

x:
( 1.0, 0.0)
( 0.0, 2.0)
( 1.0, 0.0)

```

Solution to a Linear System in a Hermitian Matrix in Packed Storage (Expert Driver)

The subroutines described in this section solve a linear system $AX = B$ for a Hermitian matrix A in packed storage and general matrices B and X . These subroutines also compute forward error bounds and backward error estimates for the solution, and estimate the condition number of the matrix A . Note that the simple driver `xHPSV` is also available.

Calling Sequence

```
CALL ZHPSVX (FACT, UPLO, N, NRHS, ZA, ZAF, IPIVOT, ZB, LDB, ZX,
            LDX, DRCOND, DFERR, DBERR, ZWORK, DWORK2, INFO)
CALL CHPSVX (FACT, UPLO, N, NRHS, CA, CAF, IPIVOT, CB, LDB, CX,
            LDX, SRCOND, SFERR, SBERR, CWORK, SWORK2, INFO)

void zhpsvx (char fact, char uplo, int n, int nrhs, doublecomplex
            *za, doublecomplex *zaf, int *ipivot, doublecomplex
            *zb, int ldb, doublecomplex *zx, int ldx, double
            *drcond, double *dferr, double *dberr, int *info)
void chpsvx (char fact, char uplo, int n, int nrhs, complex *ca,
            complex *caf, int *ipivot, complex *cb, int ldb,
            complex *cx, int ldx, float *srcond, float *sferr,
            float *sberr, int *info)
```

Arguments

FACT	Indicates whether or not the factored form of the matrix A is supplied on entry. 'F' or 'f' On entry, AF and $IPIVOT$ contain the factored form of A. A, AF, and $IPIVOT$ will not be modified. 'N' or 'n' On entry, the matrix A will be copied to AF and factored.
UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for $UPLO$ are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A . $N \geq 0$.
NRHS	Number of right-hand sides, equal to the number of columns in the matrices B and X . $NRHS \geq 0$.

xA	Upper or lower triangle of the matrix A. The dimension of xA is $(N \times N + N) / 2$.
xAF	The dimension of xAF is $(N \times N + N) / 2$. On entry, if FACT = 'F' or 'f' then AF contains the UDU or LDL factorization of A as computed by xHPTRF. Otherwise the entry value of AF is not used. On exit, if FACT = 'N' or 'n' then AF contains the UDU or LDL factorization of A as computed by xHPTRF. Otherwise AF is unchanged.
IPIVOT	On entry, if FACT = 'F' or 'f', IPIVOT contains pivot indices as computed by xHPTRF. Otherwise the entry value is not used. On exit, if FACT = 'N' or 'n', IPIVOT contains pivot indices as computed by xHPTRF. Otherwise IPIVOT is unchanged.
xB	On entry, the $N \times \text{NRHS}$ right-hand side matrix B.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $\text{LDB} \geq \max(1, N)$.
xX	On exit, the $N \times \text{NRHS}$ solution matrix for the original problem.
LDX	Leading dimension of the array X as specified in a dimension or type statement. $\text{LDX} \geq \max(1, N)$.
xRCOND	On exit, an estimate of the reciprocal condition number of the matrix A after equilibration, if any, where the reciprocal condition number is defined to be $1 / (\ A\ \times \ A^{-1}\)$. The reciprocal of the condition number is estimated instead of the condition number itself to avoid overflow or division by zero. If RCOND is less than machine precision (in particular, if $\text{RCOND} = 0$) then A is singular to working precision. In this case, $\text{INFO} > 0$ is returned and the solution and error bounds are not computed.
xFERR	On exit, the estimated forward error bound for each solution vector $X(*, j)$ for $1 \leq j \leq \text{NRHS}$. If X' is the true solution corresponding to $X(*, j)$ then $\text{FERR}(j)$ is an upper bound on the magnitude of the largest element in $X(*, j) - X'$ divided by the magnitude of the largest element in $X(*, j)$.
xBERR	On exit, $\text{BERR}(j)$ is the smallest relative change in any element of A or $B(*, j)$ that makes $X(*, j)$ an exact solution to $AX(*, j) = B(*, j)$ for $1 \leq j \leq \text{NRHS}$.
xWORK	Scratch array with a dimension of $2 \times N$.
xWORK2	Scratch array with a dimension of N.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where $i = \text{INFO}$, had an illegal value.

$1 \leq \text{INFO} \leq N$ $D(i,i)$, where $i = \text{INFO}$, is exactly zero and D is therefore singular. The UDU or LDL factorization of A has been completed, but the solution and error bounds could not be computed.

$\text{INFO} = N+1$ D is nonsingular, but RCOND is less than machine precision. The UDU or LDL factorization of A has been completed, but the solution and error bounds could not be computed.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
DOUBLE PRECISION ZERO
INTEGER LDB, LDWORK, LDWRK2, LDX, LENGTA, N, NRHS
PARAMETER (N = 3)
PARAMETER (LDB = N)
PARAMETER (LDWORK = 2 * N)
PARAMETER (LDWRK2 = N)
PARAMETER (LDX = N)
PARAMETER (LENGTA = (N * N + N) / 2)
PARAMETER (NRHS = 1)
PARAMETER (ZERO = 0.0D0)

C
DOUBLE PRECISION BERR(NRHS), FERR(NRHS), RCOND
DOUBLE PRECISION WORK2(LDWRK2)
COMPLEX*16 A(LENGTA), AF(LENGTA), B(LDB,NRHS)
COMPLEX*16 WORK(LDWORK), X(LDX,NRHS)
INTEGER ICOL, INFO, IPIVOT(N), IROW

C
EXTERNAL ZHPSVX
INTRINSIC ABS, CMLPX, CONJG, DBLE

C
C Initialize the array A to store the coefficient array A
C shown below. Initialize the array B to store the right
C hand side vector b shown below.
C
C      1      1-i      1-i      4+i
C  A = 1+i      3      3-i      b = 4+6i
C      1+i      3+i      5      4+7i
C
DATA A / (1.0D0,0.0D0), (1.0D0,-1.0D0), (3.0D0,0.0D0),
$        (1.0D0,-1.0D0), (3.0D0,-1.0D0), (5.0D0,0.0D0) /
DATA B / (4.0D0,1.0D0), (4.0D0,6.0D0), (4.0D0,7.0D0) /

C
C Print the initial value of the arrays.
C
PRINT 1000
PRINT 1010, CMLPX(DBLE(A(1)),ZERO), A(2), A(4)

```

```

        PRINT 1010, CONJG(A(2)), CMPLX(DBLE(A(3)),ZERO), A(5)
        PRINT 1010, CONJG(A(4)), CONJG(A(5)), CMPLX(DBLE(A(6)),ZERO)
        PRINT 1020
        DO 130, ICOL = 1, NRHS
            DO 120, IROW = 1, N
                PRINT 1010, B(IROW,ICOL)
            120 CONTINUE
        130 CONTINUE
C
C      Compute the solution.
C
        CALL ZHPSVX ('NOT FACTORED A', 'UPPER TRIANGLE OF A STORED',
$              N, NRHS, A, AF, IPIVOT, B, LDB, X, LDX,
$              RCOND, FERR, BERR, WORK, WORK2, INFO)
        IF (INFO .LT. 0) THEN
            PRINT 1030, ABS(INFO)
            STOP 1
        ELSE IF (INFO .GT. 0) THEN
            PRINT 1040
            STOP 2
        END IF
C
C      Print the solution and error bounds.
C
        PRINT 1050
        DO 210, ICOL = 1, NRHS
            DO 200, IROW = 1, N
                PRINT 1010, X(IROW,ICOL)
            200 CONTINUE
        210 CONTINUE
        PRINT 1060, RCOND
        DO 220, IROW = 1, NRHS
            PRINT 1070, IROW, BERR(IROW)
            PRINT 1080, IROW, FERR(IROW)
        220 CONTINUE
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (1X, 10(: 2X, '(', F4.1, ', ', F4.1, ')'))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (1X, 'Illegal argument to ZHPSVX, argument #', I2)
1040 FORMAT (1X, 'A is singular to working precision.')
1050 FORMAT (/1X, 'x:')
1060 FORMAT (/1X,
$      'Estimated reciprocal condition number of A: ',
$      F5.3)
1070 FORMAT (/1X, 'Backward error for system #', I2, ': ', E12.5)
1080 FORMAT (1X, 'Forward error for system #', I2, ': ', E12.5)
C
        END

```

Sample Output

```
A:
( 1.0, 0.0) ( 1.0, -1.0) ( 1.0, -1.0)
( 1.0, 1.0) ( 3.0, 0.0) ( 3.0, -1.0)
( 1.0, 1.0) ( 3.0, 1.0) ( 5.0, 0.0)

b:
( 4.0, 1.0)
( 4.0, 6.0)
( 4.0, 7.0)

x:
( 1.0, 0.0)
( 0.0, 2.0)
( 1.0, 0.0)

Estimated reciprocal condition number of A: 0.011

Backward error for system # 1: 0.74015E-16
Forward error for system # 1: 0.40366E-13
```

UDU or LDL Factorization of a Hermitian Matrix in Packed Storage

The subroutines described in this section compute a UDU or LDL factorization of a Hermitian matrix A in packed storage. It is typical to follow a call to xHPTRF with a call to xHPSVX or xHPTRS to solve a linear system $AX = B$, to xHPCON to estimate the condition number of A, or to xHPTRI to compute A^{-1} .

Calling Sequence

```
CALL ZHPTRF (UPLO, N, ZA, IPIVOT, INFO)
CALL CHPTRF (UPLO, N, CA, IPIVOT, INFO)

void zhptrf (char uplo, int n, doublecomplex *za, int *ipivot, int
            *info)
void chptrf (char uplo, int n, complex *ca, int *ipivot, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
xA	On entry, the upper or lower triangle of the matrix A. The dimension of xA is $(N \times N + N) / 2$. On exit, a UDU or LDL factorization of A.
IPIVOT	On exit, an N-element array containing details of the row interchanges and block structure of D.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The <i>i</i> th argument, where $i = INFO $, had an illegal value. INFO > 0 D(<i>i</i> , <i>i</i>), where $i = INFO$, is exactly zero and D is therefore singular. The factorization has been completed, but division by zero will occur if this factorization is used to solve a linear system.


```

END IF
C
C   Estimate the condition number.  Print a warning if it is
C   unreasonably high.
C
ANORM = 1.0D1
CALL ZHPCON ('UPPER TRIANGULAR FACTOR OF A', N, A, IPIVOT,
$ ANORM, RCOND, WORK, INFO)
IF (INFO .LT. 0) THEN
    PRINT 1050, ABS(INFO)
    STOP 3
END IF
PRINT 1060, RCOND
IF ((1.0D0 + RCOND) .EQ. 1.0D0) THEN
    PRINT 1070
END IF
C
C   Compute and print the solution.
C
CALL ZHPTRS ('UPPER TRIANGULAR FACTOR OF A', N, NRHS, A,
$ IPIVOT, B, LDB, INFO)
IF (INFO .LT. 0) THEN
    PRINT 1080, ABS(INFO)
    STOP 4
END IF
PRINT 1090
DO 210, ICOL = 1, NRHS
    DO 200, IROW = 1, N
        PRINT 1010, B(IROW,ICOL)
    200 CONTINUE
    210 CONTINUE
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (1X, 10(: 2X, '(', F4.1, ',', F4.1, ')'))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (1X, 'Illegal argument to ZHPTRF, argument #', I2)
1040 FORMAT (1X, 'A is singular to working precision.')
1050 FORMAT (1X, 'Illegal argument to ZHPCON, argument #', I2)
1060 FORMAT (/1X,
$ 'Estimated reciprocal condition number of A: ',
$ F5.3)
1070 FORMAT (1X, 'A may be singular to working precision.')
1080 FORMAT (1X, 'Illegal argument to ZHPTRS, argument #', I2)
1090 FORMAT (/1X, 'x:')
C
END

```

Sample Output

```
A:
( 1.0, 0.0) ( 1.0, -1.0) ( 1.0, -1.0)
( 1.0, 1.0) ( 3.0, 0.0) ( 3.0, -1.0)
( 1.0, 1.0) ( 3.0, 1.0) ( 5.0, 0.0)

b:
( 4.0, 1.0)
( 4.0, 6.0)
( 4.0, 7.0)

Estimated reciprocal condition number of A: 0.010

x:
( 1.0, 0.0)
( 0.0, 2.0)
( 1.0, 0.0)
```

Inverse of a UDU- or LDL-Factored Hermitian Matrix in Packed Storage

The subroutines described in this section compute the inverse of a Hermitian matrix A in packed storage, which has been UDU-factored or LDL-factored by xHPTRF.

Calling Sequence

```
CALL ZHPTRI (UPLO, N, ZA, IPIVOT, ZWORK, INFO)
CALL CHPTRI (UPLO, N, CA, IPIVOT, CWORK, INFO)

void zhptri (char uplo, int n, doublecomplex *za, int *ipivot, int
             *info)
void chptri (char uplo, int n, complex *ca, int *ipivot, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
xA	On entry, the UDU or LDL factorization of the matrix A, as computed by xHPTRF. The dimension of xA is $(N \times N + N) / 2$. On exit, the Hermitian inverse of the matrix A. If UPLO = 'U' or 'u', the upper triangular part of the inverse is formed and the part of A below the diagonal is not used. If UPLO = 'L' or 'l' the lower triangular part of the inverse is formed and the part of A above the diagonal is not used.
IPIVOT	Pivot indices as computed by xHPTRF.
xWORK	Scratch array with a dimension of N.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where $i = INFO $, had an illegal value. INFO > 0 D(i,i), where $i = INFO$, is zero. The matrix is therefore singular and its inverse could not be computed.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER      LDWORK, LENGTA, N
  PARAMETER    (N = 3)
  PARAMETER    (LDWORK = N)
  PARAMETER    (LENGTA = (N * N + N) / 2)

C
  COMPLEX*16   A(LENGTA), WORK(LDWORK)
  INTEGER      INFO, IPIVOT(N)

C
  EXTERNAL     ZHPTRF, ZHPTRI
  INTRINSIC    ABS, CONJG

C
C   Initialize the array A to store the coefficient array A
C   shown below.
C
C
C   A =
C   1   1-i 1-i
C   1+i 3   3-i
C   1+i 3+i 5
C
  DATA A / (1.0D0,0.0D0), (1.0D0,-1.0D0), (3.0D0,0.0D0),
$           (1.0D0,-1.0D0), (3.0D0,-1.0D0), (5.0D0,0.0D0) /

C
C   Print the initial value of A.
C
  PRINT 1000
  PRINT 1010, A(1),      A(2),      A(4)
  PRINT 1010, CONJG(A(2)), A(3),      A(5)
  PRINT 1010, CONJG(A(4)), CONJG(A(5)), A(6)

C
C   Factor the matrix.
C
  CALL ZHPTRF ('UPPER TRIANGLE OF A STORED', N, A, IPIVOT,
$             INFO)
  IF (INFO .LT. 0) THEN
    PRINT 1020, ABS(INFO)
    STOP 1
  ELSE IF (INFO .GT. 0) THEN
    PRINT 1030
    STOP 2
  END IF

C
C   Compute and print the inverse.
C
  CALL ZHPTRI ('UPPER TRIANGULAR FACTOR OF A', N, A, IPIVOT,
$             WORK, INFO)
  IF (INFO .LT. 0) THEN
    PRINT 1040, ABS(INFO)
    STOP 3
  ELSE IF (INFO .GT. 0) THEN
    PRINT 1030

```

```

        STOP 4
      END IF
      PRINT 1050
      PRINT 1010, A(1),      A(2),      A(4)
      PRINT 1010, CONJG(A(2)), A(3),      A(5)
      PRINT 1010, CONJG(A(4)), CONJG(A(5)), A(6)
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (1X, 10(: 2X, '(', F5.1, ', ', F5.1, ')'))
1020 FORMAT (1X, 'Illegal argument to ZHPTRF, argument #', I2)
1030 FORMAT (1X, 'A is singular to working precision.')
1040 FORMAT (1X, 'Illegal argument to ZHPTRI, argument #', I2)
1050 FORMAT (/1X, 'A**(-1):')
C
      END

```

Sample Output

```

A:
(  1.0,  0.0) (  1.0, -1.0) (  1.0, -1.0)
(  1.0,  1.0) (  3.0,  0.0) (  3.0, -1.0)
(  1.0,  1.0) (  3.0,  1.0) (  5.0,  0.0)

A**(-1):
(  5.0,  0.0) ( -1.0,  3.0) ( -1.0, -1.0)
( -1.0, -3.0) (  3.0,  0.0) ( -1.0,  1.0)
( -1.0,  1.0) ( -1.0, -1.0) (  1.0,  0.0)

```

Solution to a Linear System in a UDU- or LDL-Factored Hermitian Matrix in Packed Storage

The subroutines described in this section solve the linear system $AX = B$ for a Hermitian matrix A in packed storage, which has been UDU-factored or LDL-factored by xHPTRF, and general matrices B and X .

Calling Sequence

```
CALL ZHPTRS (UPLO, N, NRHS, ZA, IPIVOT, ZB, LDB, INFO)
CALL CHPTRS (UPLO, N, NRHS, CA, IPIVOT, CB, LDB, INFO)
```

```
void zhptrs (char uplo, int n, int nrhs, doublecomplex *za, int
             *ipivot, doublecomplex *zb, int ldb, int *info)
void chptrs (char uplo, int n, int nrhs, complex *ca, int *ipivot,
             complex *cb, int ldb, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
NRHS	Number of right-hand sides, equal to the number of columns of the matrix B. $NRHS \geq 0$.
xA	UDU or LDL factorization of the matrix A, as computed by xHPTRF. The dimension of xA is $(N \times N + N) / 2$.
IPIVOT	Pivot indices as computed by xHPTRF.
xB	On entry, the $N \times NRHS$ right-hand side matrix B. On exit, the $N \times NRHS$ solution matrix X.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The i th argument, where $i = INFO $, had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
DOUBLE PRECISION ZERO
INTEGER LENGTA, LDB, LDWORK, N, NRHS
PARAMETER (N = 3)
PARAMETER (LENGTA = (N * N + N) / 2)
PARAMETER (LDB = N)
PARAMETER (LDWORK = 2 * N)
PARAMETER (NRHS = 1)
PARAMETER (ZERO = 0.0D0)

C
DOUBLE PRECISION ANORM, RCOND
COMPLEX*16 A(LENGTA), B(LDB,NRHS), WORK(LDWORK)
INTEGER ICOL, INFO, IPIVOT(N), IROW

C
EXTERNAL ZHPCON, ZHPTRF, ZHPTRS
INTRINSIC ABS, CMPLX, CONJG, DBLE

C
C Initialize the array A to store the coefficient array A
C shown below. Initialize the array B to store the right
C hand side vector b shown below.
C
C      1      1-i  1-i      4+i
C  A = 1+i    3    3-i    b = 4+6i
C      1+i  3+i    5      4+7i
C
DATA A / (1.0D0,8D8), (1.0D0,-1.0D0), (3.0D0,8D8),
$      (1.0D0,-1.0D0), (3.0D0,-1.0D0), (5.0D0,8D8) /
DATA B / (4.0D0,1.0D0), (4.0D0,6.0D0), (4.0D0,7.0D0) /

C
C Print the initial value of the arrays.
C
PRINT 1000
PRINT 1010, CMPLX(DBLE(A(1)),ZERO), A(2), A(4)
PRINT 1010, CONJG(A(2)), CMPLX(DBLE(A(3)),ZERO), A(5)
PRINT 1010, CONJG(A(4)), CONJG(A(5)), CMPLX(DBLE(A(6))),
$  ZERO)
PRINT 1020
DO 130, ICOL = 1, NRHS
  DO 120, IROW = 1, N
    PRINT 1010, B(IROW,ICOL)
120  CONTINUE
130  CONTINUE

C
C Factor the matrix.
C
CALL ZHPTRF ('UPPER TRIANGLE OF A STORED', N, A, IPIVOT,
$  INFO)
IF (INFO .LT. 0) THEN
  PRINT 1030, ABS(INFO)
  STOP 1

```



```
        ELSE IF (INFO .GT. 0) THEN
            PRINT 1040
            STOP 2
        END IF
C
C      Estimate the condition number.  Print a warning if it is
C      unreasonably high.
C
        ANORM = 1.0D1
        CALL ZHPCON ('UPPER TRIANGULAR FACTOR OF A', N, A, IPIVOT,
$              ANORM, RCOND, WORK, INFO)
        IF (INFO .LT. 0) THEN
            PRINT 1050, ABS(INFO)
            STOP 3
        END IF
        PRINT 1060, RCOND
        IF ((1.0D0 + RCOND) .EQ. 1.0D0) THEN
            PRINT 1070
        END IF
C
C      Compute and print the solution.
C
        CALL ZHPTRS ('UPPER TRIANGULAR FACTOR OF A', N, NRHS, A,
$              IPIVOT, B, LDB, INFO)
        IF (INFO .LT. 0) THEN
            PRINT 1080, ABS(INFO)
            STOP 4
        END IF
        PRINT 1090
        DO 210, ICOL = 1, NRHS
            DO 200, IROW = 1, N
                PRINT 1010, B(IROW,ICOL)
            200    CONTINUE
        210    CONTINUE
C
1000    FORMAT (1X, 'A:')
1010    FORMAT (1X, 10(: 2X, '(', F4.1, ', ', F4.1, ')'))
1020    FORMAT (/1X, 'b:')
1030    FORMAT (1X, 'Illegal argument to ZHPTRF, argument #', I2)
1040    FORMAT (1X, 'A is singular to working precision.')
1050    FORMAT (1X, 'Illegal argument to ZHPCON, argument #', I2)
1060    FORMAT (/1X,
$          'Estimated reciprocal condition number of A: ',
$          F5.3)
1070    FORMAT (1X, 'A may be singular to working precision.')
1080    FORMAT (1X, 'Illegal argument to ZHPTRS, argument #', I2)
1090    FORMAT (/1X, 'x:')
C
        END
```

Sample Output

```
A:
( 1.0, 0.0) ( 1.0, -1.0) ( 1.0, -1.0)
( 1.0, 1.0) ( 3.0, 0.0) ( 3.0, -1.0)
( 1.0, 1.0) ( 3.0, 1.0) ( 5.0, 0.0)

b:
( 4.0, 1.0)
( 4.0, 6.0)
( 4.0, 7.0)

Estimated reciprocal condition number of A: 0.010

x:
( 1.0, 0.0)
( 0.0, 2.0)
( 1.0, 0.0)
```

Generation of Q Defined in LQ Factorization

The subroutines described in this section generate a matrix Q with orthonormal rows that was defined in an LQ factorization computed by xGELQF.

Calling Sequence

```
CALL DORGLQ (M, N, K, DA, LDA, DTAU, DWORK, LDWORK, INFO)
CALL SORGLQ (M, N, K, SA, LDA, STAU, SWORK, LDWORK, INFO)

void dorglq (int m, int n, int k, double *da, int lda, double *dtau,
            int *info)
void sorglq (int m, int n, int k, float *sa, int lda, float *stau,
            int *info)
```

Arguments

M	Number of rows of the matrix Q. $M \geq 0$.
N	Number of columns of the matrix Q. $N \geq M$.
K	Number of elementary reflectors whose product defines the matrix Q. $0 \leq K \leq M$.
xA	On entry, A and TAU contain the LQ factorization of the matrix A as computed by xGELQF. On exit, the M×N matrix Q.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, M)$.
xTAU	TAU(i) contains the scalar factor of the elementary reflector H(i) for $1 \leq i \leq K$ as computed by xGELQF.
xWORK	Scratch array with a dimension of LDWORK.
LDWORK	Leading dimension of the array WORK as specified in a dimension or type statement. $LDWORK \geq \max(1, M)$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The i th argument, where $i = INFO $, had an illegal value.

Sample Program

```
PROGRAM TEST
IMPLICIT NONE
```

```
C
```

```

      INTEGER          LDA, LDWORK, M, N
      PARAMETER        (M = 4)
      PARAMETER        (N = 4)
      PARAMETER        (LDA = M)
      PARAMETER        (LDWORK = M)
C
      DOUBLE PRECISION A(LDA,N), TAU(N), WORK(LDWORK)
      INTEGER          ICOL, INFO, IROW
C
      EXTERNAL          DCOPY, DGELQF, DORGLQ
      INTRINSIC         ABS, MIN
C
C      Initialize the array A to store the matrix A shown below.
C
C      A =
C      1   1   1   1
C      1  -1   1  -1
C      5  -5  -1   1
C      9  -1  -5  -3
C
      DATA A / 1.0D0,  1.0D0,  5.0D0,  9.0D0,
$             1.0D0, -1.0D0, -5.0D0, -1.0D0,
$             1.0D0,  1.0D0, -1.0D0, -5.0D0,
$             1.0D0, -1.0D0,  1.0D0, -3.0D0 /
C
C      Print the initial value of A.
C
C      PRINT 1000
      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, M)
C
C      Compute the LQ factorization of A.
C
      CALL DGELQF (M, N, A, LDA, TAU, WORK, LDWORK, INFO)
      IF (INFO .NE. 0) THEN
        PRINT 1020, ABS(INFO)
        STOP 1
      END IF
      PRINT 1030
      DO 100, IROW = 1, M
        PRINT 1010, (A(IROW,ICOL), ICOL = 1, MIN(N,IROW)),
$              (0.0D0, ICOL = IROW + 1, N)
      100 CONTINUE
C
C      Compute and print the matrix Q used in the LQ factorization
C      of A.
C
      CALL DORGLQ (N, N, N, A, LDA, TAU, WORK, LDWORK, INFO)
      IF (INFO .NE. 0) THEN
        PRINT 1040, ABS(INFO)
        STOP 2
      END IF
      PRINT 1050
      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
C
      1000 FORMAT (1X, 'A:')
      1010 FORMAT (4(3X, F8.4))
      1020 FORMAT (1X, 'Illegal value for argument #', I1,

```

```
$ ' in DGELQF.')
1030 FORMAT (/1X, 'L:')
1040 FORMAT (1X, 'Illegal value for argument #', I1,
$ ' in DORGLQ.')
1050 FORMAT (/1X, 'Q:')
C
END
```

Sample Output

```
A:
  1.0000    1.0000    1.0000    1.0000
  1.0000   -1.0000    1.0000   -1.0000
  5.0000   -5.0000   -1.0000    1.0000
  9.0000   -1.0000   -5.0000   -3.0000

L:
 -2.0000    0.0000    0.0000    0.0000
  0.0000    2.0000    0.0000    0.0000
  0.0000    4.0000    6.0000    0.0000
  0.0000    4.0000    6.0000   -8.0000

Q:
 -0.5000   -0.5000   -0.5000   -0.5000
  0.5000   -0.5000    0.5000   -0.5000
  0.5000   -0.5000   -0.5000    0.5000
 -0.5000   -0.5000    0.5000    0.5000
```

Generation of Q Defined in QL Factorization

The subroutines described in this section generate a matrix Q with orthonormal columns that was defined in a QL factorization computed by xGEQLF.

Calling Sequence

```
CALL DORGQL (M, N, K, DA, LDA, DTAU, DWORK, LDWORK, INFO)
CALL SORGQL (M, N, K, SA, LDA, STAU, SWORK, LDWORK, INFO)

void dorgql (int m, int n, int k, double *da, int lda, double *dtau,
            int *info)
void sorgql (int m, int n, int k, float *sa, int lda, float *stau,
            int *info)
```

Arguments

M	Number of rows of the matrix Q. $M \geq 0$.
N	Number of columns of the matrix Q. $M \geq N \geq 0$.
K	Number of elementary reflectors whose product defines the matrix Q. $N \geq K \geq 0$.
xA	On entry, A and TAU contain the QL factorization of the matrix A as computed by xGEQLF. On exit, the M×N matrix Q.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, M)$.
xTAU	TAU(i) contains the scalar factor of the elementary reflector H(i) for $1 \leq i \leq K$ as computed by xGEQLF.
xWORK	Scratch array with a dimension of LDWORK.
LDWORK	Leading dimension of the array WORK as specified in a dimension or type statement. $LDWORK \geq \max(1, N)$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where $i = \text{INFO} $, had an illegal value.

Sample Program

```
PROGRAM TEST
IMPLICIT NONE

C
C   INTEGER          LDA, LDL, LDWORK, M, N
C   PARAMETER        (M = 4)
C   PARAMETER        (N = 3)
C   PARAMETER        (LDA = M)
C   PARAMETER        (LDL = N)
C   PARAMETER        (LDWORK = N)
C
C   DOUBLE PRECISION A(LDA,N), L(LDL,N), TAU(N), WORK(LDWORK)
C   INTEGER          ICOL, INFO, IROW
C
C   EXTERNAL          DCOPY, DGEQLF, DORGQL
C   INTRINSIC         ABS
C
C   Initialize the array A to store the matrix A shown below.
C
C   A =
C   1   2   2
C   1   0  -2
C   1   0   2
C   1  -2  -2
C
C   DATA A / 1.0D0, 1.0D0, 1.0D0, 1.0D0,
C   $          2.0D0, 0.0D0, 0.0D0, -2.0D0,
C   $          2.0D0, -2.0D0, 2.0D0, -2.0D0 /
C
C   Print the initial value of A.
C
C   PRINT 1000
C   PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, M)
C
C   Compute the QL factorization of A.
C
C   CALL DGEQLF (M, N, A, LDA, TAU, WORK, LDWORK, INFO)
C   IF (INFO .NE. 0) THEN
C       PRINT 1020, ABS(INFO)
C       STOP 1
C   END IF
C   DO 100, ICOL = 1, N
C       CALL DCOPY (N, A(M-N+1,ICOL), 1, L(1,ICOL), 1)
100 CONTINUE
C
C   Compute and print the matrix Q used in the QL factorization
C   of A.
C
C   CALL DORGQL (M, N, N, A, LDA, TAU, WORK, LDWORK, INFO)
C   IF (INFO .NE. 0) THEN
C       PRINT 1030, ABS(INFO)
C       STOP 2
C   END IF
C   PRINT 1040
```

```

      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, M)
      PRINT 1050
      DO 110, IROW = 1, N
        PRINT 1010, (L(IROW,ICOL), ICOL = 1, IROW),
          $          (0.0D0, ICOL = IROW + 1, N)
      110 CONTINUE
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (3(3X, F8.4))
1020 FORMAT (1X, 'Illegal value for argument #', I1,
  $          ' in DGEQLF.')
1030 FORMAT (1X, 'Illegal value for argument #', I1,
  $          ' in DORGQL.')
1040 FORMAT (/1X, 'Q:')
1050 FORMAT (/1X, 'L:')
C
      END

```

Sample Output

```

A:
  1.0000    2.0000    2.0000
  1.0000    0.0000   -2.0000
  1.0000    0.0000    2.0000
  1.0000   -2.0000   -2.0000

Q:
 -0.5000    0.5000    0.5000
 -0.5000    0.5000   -0.5000
 -0.5000   -0.5000    0.5000
 -0.5000   -0.5000   -0.5000

L:
 -2.0000    0.0000    0.0000
  0.0000    2.0000    0.0000
  0.0000    2.0000    4.0000

```


Generation of Q Defined in QR Factorization

The subroutines described in this section generate a matrix Q with orthonormal columns that was defined in a QR factorization computed by xGEQRF.

Calling Sequence

```
CALL DORGQR (M, N, K, DA, LDA, DTAU, DWORK, LDWORK, INFO)
CALL SORGQR (M, N, K, SA, LDA, STAU, SWORK, LDWORK, INFO)

void dorgqr (int m, int n, int k, double *da, int lda, double *dtau,
            int *info)
void sorgqr (int m, int n, int k, float *sa, int lda, float *stau,
            int *info)
```

Arguments

M	Number of rows of the matrix Q. $M \geq 0$.
N	Number of columns of the matrix Q. $M \geq N \geq 0$.
K	Number of elementary reflectors whose product defines the matrix Q. $N \geq K \geq 0$.
xA	On entry, A and TAU contain the QR factorization of the matrix A as computed by xGEQRF. On exit, the $M \times N$ matrix Q.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, M)$.
xTAU	TAU(i) contains the scalar factor of the elementary reflector H(i) for $1 \leq i \leq K$, as computed by xGEQRF.
xWORK	Scratch array with a dimension of LDWORK.
LDWORK	Leading dimension of the array WORK as specified in a dimension or type statement. $LDWORK \geq \max(1, N)$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where $i = \text{INFO} $, had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER          LDA, LDR, LDWORK, M, N
  PARAMETER        (M = 4)
  PARAMETER        (N = 3)
  PARAMETER        (LDA = M)
  PARAMETER        (LDR = N)
  PARAMETER        (LDWORK = N)

C
  DOUBLE PRECISION A(LDA,N), R(LDR,N), TAU(N), WORK(LDWORK)
  INTEGER          ICOL, INFO, IROW

C
  EXTERNAL          DCOPY, DGEQRF, DORGQR
  INTRINSIC         ABS

C
  Initialize the array A to store the matrix A shown below.
C
C
C   1   6   2
C  A = 1  -2  -8
C       1  -2   4
C       1   6  14
C
  DATA A / 4*1.0D0, 6.0D0, -2.0D0, -2.0D0, 6.0D0,
$          2.0D0, -8.0D0, 4.0D0, 1.4D1 /

C
  Print the initial value of A.
C
C
  PRINT 1000
  PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, M)

C
  Compute the QR factorization of A.
C
  CALL DGEQRF (M, N, A, LDA, TAU, WORK, LDWORK, INFO)
  IF (INFO .NE. 0) THEN
    PRINT 1020, ABS(INFO)
    STOP 1
  END IF
  DO 100, ICOL = 1, N
    CALL DCOPY (N, A(1,ICOL), 1, R(1,ICOL), 1)
100 CONTINUE

C
  Compute and print the matrix Q used in the QR factorization
  of A.
C
C
  CALL DORGQR (M, N, N, A, LDA, TAU, WORK, LDWORK, INFO)
  IF (INFO .NE. 0) THEN
    PRINT 1030, ABS(INFO)
    STOP 2
  END IF
  PRINT 1040
  PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, M)

```

```
        PRINT 1050
        DO 110, IROW = 1, N
            PRINT 1010, (0.0D0, ICOL = 1, IROW - 1),
$          (R(IROW,ICOL), ICOL = IROW, N)
        110 CONTINUE
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (3(3X, F8.4))
1020 FORMAT (1X, 'Illegal value for argument #', I1,
$          ' in DGEQRF.')
1030 FORMAT (1X, 'Illegal value for argument #', I1,
$          ' in DORGQR.')
1040 FORMAT (/1X, 'Q:')
1050 FORMAT (/1X, 'R:')
C
        END
```

Sample Output

```
A:
  1.0000    6.0000    2.0000
  1.0000   -2.0000   -8.0000
  1.0000   -2.0000    4.0000
  1.0000    6.0000   14.0000

Q:
 -0.5000    0.5000    0.5000
 -0.5000   -0.5000    0.5000
 -0.5000   -0.5000   -0.5000
 -0.5000    0.5000   -0.5000

R:
 -2.0000   -4.0000   -6.0000
  0.0000    8.0000   10.0000
  0.0000    0.0000  -12.0000
```

Generation of Q Defined in RQ Factorization

The subroutines described in this section generate a matrix Q with orthonormal rows that was defined in an RQ factorization computed by xGERQF.

Calling Sequence

```
CALL DORGRQ (M, N, K, DA, LDA, DTAU, DWORK, LDWORK, INFO)
CALL SORGRQ (M, N, K, SA, LDA, STAU, SWORK, LDWORK, INFO)

void dorgrq (int m, int n, int k, double *da, int lda, double *dtau,
            int *info)
void sorgrq (int m, int n, int k, float *sa, int lda, float *stau,
            int *info)
```

Arguments

M	Number of rows of the matrix Q. $M \geq 0$.
N	Number of columns of the matrix Q. $N \geq M$.
K	Number of elementary reflectors whose product defines the matrix Q. $M \geq K \geq 0$.
xA	On entry, A and TAU contain the RQ factorization of the matrix A as computed by xGERQF. On exit, the $M \times N$ matrix Q.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, M)$.
xTAU	TAU(i) contains the scalar factor of the elementary reflector H(i) for $1 \leq i \leq K$ as computed by xGERQF.
xWORK	Scratch array with a dimension of LDWORK.
LDWORK	Leading dimension of the array WORK as specified in a dimension or type statement. $LDWORK \geq \max(1, M)$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where $i = \text{INFO} $, had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER          LDA, LDR, LDTAU, LDWORK, M, N
  PARAMETER        (M = 5)
  PARAMETER        (N = 4)
  PARAMETER        (LDA = M)
  PARAMETER        (LDR = M)
  PARAMETER        (LDTAU = LDA)
  PARAMETER        (LDWORK = M)

C
  DOUBLE PRECISION A(LDA,N), R(LDR,N), TAU(LDTAU)
  DOUBLE PRECISION WORK(LDWORK)
  INTEGER          ICOL, INFO, IROW

C
  EXTERNAL          DCOPY, DGERQF, DORGRQ
  INTRINSIC         ABS, MAX

C
C   Initialize the array A to store the matrix A shown below.
C
C   1   1   0   0
C   1   1   1   1
C   A = 1   1  -1  -1
C       1  -1   1  -1
C       1  -1  -1   1
C
C   DATA A / 1.0D0, 1.0D0, 1.0D0, 1.0D0, 1.0D0,
$             1.0D0, 1.0D0, 1.0D0, -1.0D0, -1.0D0,
$             0.0D0, 1.0D0, -1.0D0, 1.0D0, -1.0D0,
$             0.0D0, 1.0D0, -1.0D0, -1.0D0, 1.0D0 /

C
C   Print the initial value of A.
C
C   PRINT 1000
C   DO 100, IROW = 1, M
C     PRINT 1010, (A(IROW,ICOL), ICOL = 1, N)
100 CONTINUE

C
C   Compute the RQ factorization of A.
C
C   CALL DGERQF (M, N, A, LDA, TAU, WORK, LDWORK, INFO)
C   IF (INFO .NE. 0) THEN
C     PRINT 1020, ABS(INFO)
C     STOP 1
C   END IF
C   DO 110, ICOL = 1, N
C     CALL DCOPY (M, A(1,ICOL), 1, R(1,ICOL), 1)
110 CONTINUE

C
C   Compute and print the matrix Q used in the RQ factorization
C   of A.
C

```

```

      CALL DORGRQ (N, N, N, A(M-N+1,1), LDA, TAU, WORK, LDWORK,
$      INFO)
      IF (INFO .NE. 0) THEN
        PRINT 1030, ABS(INFO)
        STOP 2
      END IF
      PRINT 1040
      DO 120, IROW = 1, M
        PRINT 1010, (0.0D0, ICOL = 1, IROW - (M - N) - 1),
$        (R(IROW,ICOL), ICOL = MAX(1,IROW - (M - N)), N)
120  CONTINUE
      PRINT 1050
      DO 130, IROW = M - N + 1, M
        PRINT 1010, (A(IROW,ICOL), ICOL = 1, N)
130  CONTINUE
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (5(3X, F8.4))
1020 FORMAT (1X, 'Illegal value for argument #', I1,
$          ' in DGERQF.')
1030 FORMAT (1X, 'Illegal value for argument #', I1,
$          ' in DORGRQ.')
1040 FORMAT (/1X, 'R:')
1050 FORMAT (/1X, 'Q:')
C
      END

```

Sample Output

```

A:
  1.0000    1.0000    0.0000    0.0000
  1.0000    1.0000    1.0000    1.0000
  1.0000    1.0000   -1.0000   -1.0000
  1.0000   -1.0000    1.0000   -1.0000
  1.0000   -1.0000   -1.0000    1.0000

R:
 -1.0000   -1.0000    0.0000    0.0000
 -2.0000    0.0000    0.0000    0.0000
  0.0000   -2.0000    0.0000    0.0000
  0.0000    0.0000   -2.0000    0.0000
  0.0000    0.0000    0.0000   -2.0000

Q:
 -0.5000   -0.5000   -0.5000   -0.5000
 -0.5000   -0.5000    0.5000    0.5000
 -0.5000    0.5000   -0.5000    0.5000
 -0.5000    0.5000    0.5000   -0.5000

```

Reciprocal Condition Number of a Cholesky-Factored Symmetric Positive Definite Matrix in Banded Storage

The subroutines described in this section estimate the reciprocal condition number of a real symmetric (or Hermitian) positive definite matrix A in banded storage, which has been Cholesky-factored by xPBTRF.

Calling Sequence

```
CALL DPBCON (UPLO, N, NDIAG, DA, LDA, DANORM, DRCOND, DWORK,
             IWORK2, INFO)
CALL SPBCON (UPLO, N, NDIAG, SA, LDA, SANORM, SRCOND, SWORK,
             IWORK2, INFO)
CALL ZPBCON (UPLO, N, NDIAG, ZA, LDA, DANORM, DRCOND, ZWORK,
             DWORK2, INFO)
CALL CPBCON (UPLO, N, NDIAG, CA, LDA, SANORM, SRCOND, CWORK,
             SWORK2, INFO)

void dpbcon (char uplo, int n, int ndiag, double *da, int lda,
            double danorm, double *drcond, int *info)
void spbcon (char uplo, int n, int ndiag, float *sa, int lda, float
            sanorm, float *srcond, int *info)
void zpbcon (char uplo, int n, int ndiag, doublecomplex *za, int
            lda, double danorm, double *drcond, int *info)
void cpbcon (char uplo, int n, int ndiag, complex *ca, int lda,
            float sanorm, float *srcond, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
NDIAG	Number of superdiagonals or subdiagonals of the matrix A. $N-1 \geq \text{NDIAG} \geq 0$ but if $N = 0$ then $\text{NDIAG} = 0$. If UPLO = 'U' or 'u', NDIAG is the number of superdiagonals. If UPLO = 'L' or 'l', NDIAG is the number of subdiagonals.
xA	Cholesky factorization of the matrix A, as computed by xPBTRF.

LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq NDIAG + 1$.	
xANORM	The 1-norm or ∞ -norm of matrix A.	
xRCOND	On exit, the estimated reciprocal of the condition number of the matrix A where the reciprocal condition number of A is defined to be $1 / (ANORM \times \ A^{-1}\)$. The reciprocal of the condition number is estimated instead of the condition number itself to avoid overflow or division by zero. If RCOND is less than machine precision (in particular, if RCOND = 0) then A is singular to working precision.	
xWORK	Scratch array with a dimension of $3 \times N$ for real subroutines or $2 \times N$ for complex subroutines.	
xWORK2	Scratch array with a dimension of N.	
INFO	On exit:	
	INFO = 0	Subroutine completed normally.
	INFO < 0	The ith argument, where $i = INFO $, had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
C   INTEGER          LDA, LDWORK, N, NDIAG
C   PARAMETER        (N = 4)
C   PARAMETER        (NDIAG = 1)
C   PARAMETER        (LDA = NDIAG + 1)
C   PARAMETER        (LDWORK = 3 * N)
C
C   DOUBLE PRECISION A(LDA,N), ANORM, RCOND, WORK(LDWORK)
C   INTEGER          ICOL, INFO, IROW, IWORK(N)
C
C   EXTERNAL          DPBCON, DPBTRF
C   INTRINSIC         ABS
C
C   Initialize the array A to store in symmetric banded form
C   the 4x4 symmetric positive definite matrix A with one
C   subdiagonal and one superdiagonal shown below.
C
C   2   -1   0   0
C   A = -1   2  -1   0
C       0  -1   2  -1
C       0   0  -1   2
C
C   DATA A /      8D8, 2.0D0, -1.0D0, 2.0D0,
C   $        -1.0D0, 2.0D0, -1.0D0, 2.0D0 /
C
C   Print the initial values of the arrays.
C
C

```



```
      PRINT 1000
      PRINT 1010, A(2,1), A(1,2), 0.0D0, 0.0D0
      PRINT 1010, A(1,2), A(2,2), A(1,3), 0.0D0
      PRINT 1010, 0.0D0, A(1,3), A(2,3), A(1,4)
      PRINT 1010, 0.0D0, 0.0D0, A(1,4), A(2,4)
      PRINT 1020
      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, LDA)
C
C      Cholesky factor A.
C
      CALL DPBTRF ('UPPER TRIANGLE OF A STORED', N, NDIAG, A, LDA,
$              INFO)
      IF (INFO .NE. 0) THEN
          PRINT 1030, INFO
          STOP 1
      END IF
C
C      Compute and print the condition number of A.
C
      ANORM = 3.0D0
      CALL DPBCON ('UPPER TRIANGLE FACTOR STORED', N, NDIAG, A,
$              LDA, ANORM, RCOND, WORK, IWORK, INFO)
      IF (INFO .NE. 0) THEN
          PRINT 1040, ABS(INFO)
          STOP 2
      END IF
      PRINT 1050, 1.0D0 / RCOND
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F6.3))
1020 FORMAT (/1X, 'A in banded form: (* in unused elements)')
1030 FORMAT (1X, 'Error factoring A, INFO = ', I5)
1040 FORMAT (1X, 'Illegal argument to DPBCON, argument #', I2)
1050 FORMAT (/1X, 'Estimated condition number of A = ', F8.4)
C
      END
```

Sample Output

```
A in full form:
 2.000  -1.000   0.000   0.000
-1.000   2.000  -1.000   0.000
 0.000  -1.000   2.000  -1.000
 0.000   0.000  -1.000   2.000

A in banded form: (* in unused elements)
*****
 2.000  -1.000  -1.000  -1.000
 2.000   2.000   2.000   2.000

Estimated condition number of A = 9.0000
```

Equilibration Scale Factors for a Symmetric Positive Definite Matrix in Banded Storage

The subroutines described in this section compute equilibration scale factors for a real symmetric (or Hermitian) positive definite matrix A in banded storage. Equilibration is intended to reduce the condition number of A with respect to the 2-norm. SCALE contains scale factors chosen so that the scaled matrix B with elements $B(i,j) = \text{SCALE}(i) \times A(i,j) \times \text{SCALE}(j)$ has ones on the diagonal. This choice of SCALE puts the condition number of B within a factor N of the smallest possible condition number over all possible diagonal scalings. Note that these subroutines only compute the scale factors for A. Additional code must be written to actually equilibrate A.

Calling Sequence

```
CALL DPBEQU (UPLO, N, NDIAG, DA, LDA, DSCALE, DSCOND, DAMAX, INFO)
CALL SPBEQU (UPLO, N, NDIAG, SA, LDA, SSCALE, SSCOND, SAMAX, INFO)
CALL ZPBEQU (UPLO, N, NDIAG, ZA, LDA, DSCALE, DSCOND, DAMAX, INFO)
CALL CPBEQU (UPLO, N, NDIAG, CA, LDA, SSCALE, SSCOND, SAMAX, INFO)
```

```
void dpbequ (char uplo, int n, int ndiag, double *da, int lda,
             double *dscale, double *dscond, double *damax, int
             *info)
void spbequ (char uplo, int n, int ndiag, float *sa, int lda, float
             *sscale, float *sscond, float *samax, int *info)
void zpbequ (char uplo, int n, int ndiag, doublecomplex *za, int
             lda, double *dscale, double *dscond, double *damax, int
             *info)
void cpbequ (char uplo, int n, int ndiag, complex *ca, int lda,
             float *sscale, float *sscond, float *samax, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
NDIAG	Number of superdiagonals or subdiagonals of the matrix A. $N-1 \geq \text{NDIAG} \geq 0$ but if $N = 0$ then $\text{NDIAG} = 0$.

	If UPLO = 'U' or 'u', NDIAG is the number of superdiagonals.
	If UPLO = 'L' or 'l', NDIAG is the number of subdiagonals.
xA	Upper or lower triangle of the matrix A.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq NDIAG + 1$.
xSCALE	On exit, the N scale factors for A.
xSCOND	On exit, the ratio of the smallest SCALE(i) to the largest SCALE(i). If $SCOND \geq 0.1$ and AMAX is not close to overflow or underflow, it is not worth scaling by SCALE.
xAMAX	On exit, the absolute value of the largest element of A. If AMAX is very close to overflow or underflow, A should be scaled regardless of the value of SCOND.
INFO	On exit:
	INFO = 0 Subroutine completed normally.
	INFO < 0 The ith argument, where $i = INFO $, had an illegal value.
	INFO > 0 The ith diagonal entry, where $i = INFO$, is non-positive.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C      INTEGER          LDA, N, NDIAG
C      PARAMETER        (N = 4)
C      PARAMETER        (NDIAG = 1)
C      PARAMETER        (LDA = NDIAG + 1)
C
C      DOUBLE PRECISION A(LDA,N), AMAX, SCALE(N), SCOND
C      INTEGER          ICOL, INFO, IROW
C
C      EXTERNAL          DPBEQU
C      INTRINSIC         ABS
C
C      Initialize the array A to store in symmetric banded form
C      the 4x4 symmetric positive definite matrix A with one
C      subdiagonal and one superdiagonal shown below.
C
C      1024      -2      0      0
C      A =      -2      128      -2      0
C              0      -2      16      -2
C              0      0      -2      1
C
C      DATA A /      8D8, 1.024D3, -2.0D0, 1.28D2,
C      $          -2.0D0, 1.6D1, -2.0D0, 1.0D0 /
C

```

```

C      Print the initial values of the arrays.
C
      PRINT 1000
      PRINT 1010, A(2,1), A(1,2), 0.0D0, 0.0D0
      PRINT 1010, A(1,2), A(2,2), A(1,3), 0.0D0
      PRINT 1010, 0.0D0, A(1,3), A(2,3), A(1,4)
      PRINT 1010, 0.0D0, 0.0D0, A(1,4), A(2,4)
      PRINT 1020
      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, LDA)
C
C      Compute the scale factors to use to equilibrate A.
C
      CALL DPBEQU ('UPPER TRIANGLE OF A STORED', N, NDIAG, A,
$              LDA, SCALE, SCOND, AMAX, INFO)
      IF (INFO .NE. 0) THEN
          PRINT 1030, INFO
          STOP 1
      END IF
C
C      Apply the scale factors to A then print the result.
C
      DO 100, ICOL = 1, N
          A(NDIAG+1,ICOL) = A(NDIAG+1,ICOL) * (SCALE(ICOL) ** 2)
100  CONTINUE
      DO 110, ICOL = 2, N
          A(1,ICOL) = A(1,ICOL) * SCALE(ICOL-1) * SCALE(ICOL)
110  CONTINUE
      PRINT 1040
      PRINT 1050, SCALE
      PRINT 1060
      PRINT 1010, A(2,1), A(1,2), 0.0D0, 0.0D0
      PRINT 1010, A(1,2), A(2,2), A(1,3), 0.0D0
      PRINT 1010, 0.0D0, A(1,3), A(2,3), A(1,4)
      PRINT 1010, 0.0D0, 0.0D0, A(1,4), A(2,4)
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F8.3))
1020 FORMAT (/1X, 'A in banded form: (* in unused elements)')
1030 FORMAT (1X, 'Error computing scale factors of A, INFO =',
$          1X, I5)
1040 FORMAT (/1X, 'Scale factors for A:')
1050 FORMAT (4X, F8.5)
1060 FORMAT (/1X, 'Scaled A:')
C
      END

```

Sample Output

```
A in full form:
1024.000   -2.000    0.000    0.000
-2.000   128.000   -2.000    0.000
 0.000    -2.000   16.000   -2.000
 0.000    0.000   -2.000    1.000

A in banded form: (* in unused elements)
*****   -2.000   -2.000   -2.000
1024.000  128.000   16.000    1.000

Scale factors for A:
 0.03125
 0.08839
 0.25000
 1.00000

Scaled A:
 1.000   -0.006    0.000    0.000
-0.006    1.000   -0.044    0.000
 0.000   -0.044    1.000   -0.500
 0.000    0.000   -0.500    1.000
```

Refined Solution to a Linear System in a Cholesky-Factored Symmetric Positive Definite Matrix in Banded Storage

The subroutines described in this section refine the solution to a real symmetric (or Hermitian) positive definite linear system $AX = B$ for a real symmetric (or Hermitian) positive definite matrix A in banded storage, which has been Cholesky-factored by xPBTRF, and general matrices B and X . These subroutines also compute forward error bounds and backward error estimates for the refined solution.

Calling Sequence

```
CALL DPBRFS (UPLO, N, NDIAG, NRHS, DA, LDA, DAF, LDAF, DB, LDB, DX,
             LDX, DFERR, DBERR, DWORK, IWORK2, INFO)
CALL SPBRFS (UPLO, N, NDIAG, NRHS, SA, LDA, SAF, LDAF, SB, LDB, SX,
             LDX, SFERR, SBERR, SWORK, IWORK2, INFO)
CALL ZPBRFS (UPLO, N, NDIAG, NRHS, ZA, LDA, ZAF, LDAF, ZB, LDB, ZX,
             LDX, DFERR, DBERR, ZWORK, DWORK2, INFO)
CALL CPBRFS (UPLO, N, NDIAG, NRHS, CA, LDA, CAF, LDAF, CB, LDB, CX,
             LDX, SFERR, SBERR, CWORK, SWORK2, INFO)

void dpbrfs (char uplo, int n, int ndiag, int nrhs, double *da, int
             lda, double *daf, int ldaf, double *db, int ldb, double
             *dx, int ldx, double *dferr, double *dberr, int *info)
void spbrfs (char uplo, int n, int ndiag, int nrhs, float *sa, int
             lda, float *saf, int ldaf, float *sb, int ldb, float
             *sx, int ldx, float *sferr, float *sberr, int *info)
void zpbrfs (char uplo, int n, int ndiag, int nrhs, doublecomplex
             *za, int lda, doublecomplex *zaf, int ldaf,
             doublecomplex *zb, int ldb, doublecomplex *zx, int ldx,
             double *dferr, double *dberr, int *info)
void cpbrfs (char uplo, int n, int ndiag, int nrhs, complex *ca,
             int lda, complex *caf, int ldaf, complex *cb, int ldb,
             complex *cx, int ldx, float *sferr, float *sberr, int
             *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
NDIAG	Number of superdiagonals or subdiagonals of the matrix A. $N-1 \geq \text{NDIAG} \geq 0$ but if $N = 0$ then $\text{NDIAG} = 0$. If UPLO = 'U' or 'u', NDIAG is the number of superdiagonals. If UPLO = 'L' or 'l', NDIAG is the number of subdiagonals.
NRHS	Number of right-hand sides, equal to the number of columns of the matrices B and X. $\text{NRHS} \geq 0$.
xA	Upper or lower triangle of the matrix A.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $\text{LDA} \geq \max(1, N)$.
xAF	Cholesky factorization of the matrix A as computed by xPBTRF.
LDAF	Leading dimension of the array AF as specified in a dimension or type statement. $\text{LDAF} \geq \max(1, N)$.
xB	The $N \times \text{NRHS}$ right-hand side matrix B.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $\text{LDB} \geq \max(1, N)$.
XX	On entry, the $N \times \text{NRHS}$ solution matrix X. On exit, the refined $N \times \text{NRHS}$ solution matrix X.
LDX	Leading dimension of the array X as specified in a dimension or type statement. $\text{LDX} \geq \max(1, N)$.
xFERR	On exit, the estimated forward error bound for each solution vector $X(*, j)$ for $1 \leq j \leq \text{NRHS}$. If X' is the true solution corresponding to $X(*, j)$ then $\text{FERR}(j)$ is an upper bound on the magnitude of the largest element in $X(*, j) - X'$ divided by the magnitude of the largest element in $X(*, j)$.
xBERR	On exit, $\text{BERR}(j)$ is the smallest relative change in any element of A or $B(*, j)$ that makes $X(*, j)$ an exact solution to $AX(*, j) = B(*, j)$ for $1 \leq j \leq \text{NRHS}$.
xWORK	Scratch array with a dimension of $3 \times N$ for real subroutines or $2 \times N$ for complex subroutines.
xWORK2	Scratch array with a dimension of N.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The i th argument, where $i = \text{INFO} $, had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER          LDA, LDAF, LDB, LDIWRK, LDWORK, LDX
  INTEGER          N, NRHS, NDIAG
  PARAMETER        (N = 4)
  PARAMETER        (NDIAG = 1)
  PARAMETER        (LDA = NDIAG + 1)
  PARAMETER        (LDAF = LDA)
  PARAMETER        (LDB = N)
  PARAMETER        (LDIWRK = N)
  PARAMETER        (LDWORK = 3 * N)
  PARAMETER        (LDX = N)
  PARAMETER        (NRHS = 1)

C
  DOUBLE PRECISION A(LDA,N), AF(LDAF,N), B(LDB,NRHS)
  DOUBLE PRECISION B1(LDB,NRHS), BERR(NRHS), EPSLON
  DOUBLE PRECISION FERR(NRHS), WORK(LDWORK), X(LDX,NRHS)
  INTEGER          ICOL, INFO, IROW, IWORK(LDIWRK)

C
  EXTERNAL          DCOPY, DSBMV, DPBTRF, DPBTRS, DPBRFS
  INTRINSIC         ABS, DBLE, MAX

C
  C
  C Initialize the array B to store the right hand side
  C vector b shown.
  C
  C
  C      4
  C      b = 4
  C      4
  C      4
  C
  C      DATA B / N*4.0D0 /

C
  C Initialize the array A to contain the matrix A shown below
  C where e=epsilon=small.
  C
  C
  C      e      e*e
  C      A = e*e      e      e*e
  C      e*e      e      e*e
  C      e*e      e
  C
  C
  C      A(1,1) = SQRT(-1.0D0)
  C      EPSLON = ABS((((2.0D0 / 3.0D0) + 8.0D0) - 8.0D0) -
  C      $      (2.0D0 / 3.0D0))
  C      DO 100, ICOL = 1, N
  C        A(NDIAG + 1,ICOL) = EPSLON
100  CONTINUE
  C      DO 110, ICOL = 2, N
  C        A(1,ICOL) = EPSLON * EPSLON
110  CONTINUE
  C      CALL DCOPY (LDA*N, A, 1, AF, 1)
C

```



```
C      Print the A array.
C
      PRINT 1000
      PRINT 1010, A(2,1), A(1,2), 0.0D0, 0.0D0
      PRINT 1010, A(1,2), A(2,2), A(1,3), 0.0D0
      PRINT 1010, 0.0D0, A(1,3), A(2,3), A(1,4)
      PRINT 1010, 0.0D0, 0.0D0, A(1,4), A(2,4)
      PRINT 1020
      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, LDA)
C
C      Make a copy of B and then print B.
C
      CALL DCOPY (N, B, 1, X, 1)
      PRINT 1030
      PRINT 1040, B
C
C      Factor A.
C
      CALL DPBTRF ('UPPER TRIANGLE OF A STORED', N, NDIAG,
$              AF, LDAF, INFO)
      IF (INFO .NE. 0) THEN
          PRINT 1050, INFO
          STOP 1
      END IF
C
C      Solve Ax=b and print the solution.
C
      CALL DPBTRS ('UPPER TRIANGLE OF A STORED', N, NDIAG, NRHS,
$              AF, LDAF, X, LDX, INFO)
      IF (INFO .NE. 0) THEN
          PRINT 1060, INFO
          STOP 2
      END IF
      PRINT 1070
      PRINT 1080, X
      CALL DSBMV ('UPPER TRIANGLE OF A STORED', N, NDIAG, 1.0D0,
$              A, LDA, X, 1, 0.0D0, B1, 1)
      PRINT 1090
      PRINT 1040, B1
C
C      Refine the initial solution and print the new solution.
C
      CALL DPBRFS ('UPPER TRIANGLE OF A STORED', N, NDIAG, NRHS,
$              A, LDA, AF, LDAF, B, LDB, X, LDX, FERR, BERR,
$              WORK, IWORK, INFO)
      IF (INFO .NE. 0) THEN
          PRINT 1100, INFO
          STOP 3
      END IF
      PRINT 1110
      PRINT 1080, X
      CALL DSBMV ('UPPER TRIANGLE OF A STORED', N, NDIAG, 1.0D0,
$              A, LDA, X, 1, 0.0D0, B1, 1)
      PRINT 1120
      PRINT 1040, B1
C
```

```

1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(2X, E15.8))
1020 FORMAT (/1X, 'A in banded form: ("NaN" in unused elements)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (1X, F19.15)
1050 FORMAT (1X, 'Error factoring A in DPBTRF, INFO = ', I5)
1060 FORMAT (1X, 'Error solving Ax=b in DPBTRF, INFO = ', I5)
1070 FORMAT (/1X, 'Initial solution for Ax=b:')
1080 FORMAT (1X, E24.16)
1090 FORMAT (/1X, 'Ax with initial x:')
1100 FORMAT (1X, 'Error improving solution in DPBRFS, INFO = ',
$      I5)
1110 FORMAT (/1X, 'Refined solution for Ax=b:')
1120 FORMAT (/1X, 'Ax with improved x:')
C
      END

```

Sample Output

```

A in full form:
 0.55511151E-15   0.30814879E-30   0.00000000E+00   0.00000000E+00
 0.30814879E-30   0.55511151E-15   0.30814879E-30   0.00000000E+00
 0.00000000E+00   0.30814879E-30   0.55511151E-15   0.30814879E-30
 0.00000000E+00   0.00000000E+00   0.30814879E-30   0.55511151E-15

A in banded form: ("NaN" in unused elements)
NaN              0.30814879E-30   0.30814879E-30   0.30814879E-30
 0.55511151E-15   0.55511151E-15   0.55511151E-15   0.55511151E-15

b:
4.0000000000000000
4.0000000000000000
4.0000000000000000
4.0000000000000000

Initial solution for Ax=b:
 0.7205759403792790E+16
 0.7205759403792784E+16
 0.7205759403792784E+16
 0.7205759403792788E+16

Ax with initial x:
4.0000000000000000
3.9999999999999999
3.9999999999999999
3.9999999999999999

Refined solution for Ax=b:
 0.7205759403792789E+16
 0.7205759403792786E+16
 0.7205759403792786E+16
 0.7205759403792790E+16

```

```
Ax with improved x:  
  4.0000000000000000  
  4.0000000000000000  
  4.0000000000000000  
  4.0000000000000000
```

Solution to a Linear System in a Symmetric Positive Definite Matrix in Banded Storage (Simple Driver)

The subroutines described in this section solve a linear system $AX = B$ for a real symmetric (or Hermitian) positive definite matrix A in banded storage and general matrices B and X. Note that the expert driver xPBSVX is also available.

Calling Sequence

```
CALL DPBSV (UPLO, N, NDIAG, NRHS, DA, LDA, DB, LDB, INFO)
CALL SPBSV (UPLO, N, NDIAG, NRHS, SA, LDA, SB, LDB, INFO)
CALL ZPBSV (UPLO, N, NDIAG, NRHS, ZA, LDA, ZB, LDB, INFO)
CALL CPBSV (UPLO, N, NDIAG, NRHS, CA, LDA, CB, LDB, INFO)

void dpbsv (char uplo, int n, int ndiag, int nrhs, double *da, int
           lda, double *db, int ldb, int *info)
void spbsv (char uplo, int n, int ndiag, int nrhs, float *sa, int
           lda, float *sb, int ldb, int *info)
void zpbsv (char uplo, int n, int ndiag, int nrhs, doublecomplex
           *za, int lda, doublecomplex *zb, int ldb, int *info)
void cpbsv (char uplo, int n, int ndiag, int nrhs, complex *ca,
           int lda, complex *cb, int ldb, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
NDIAG	Number of superdiagonals or subdiagonals of the matrix A. $N-1 \geq \text{NDIAG} \geq 0$ but if $N = 0$ then $\text{NDIAG} = 0$. If UPLO = 'U' or 'u', NDIAG is the number of superdiagonals. If UPLO = 'L' or 'l', NDIAG is the number of subdiagonals.
NRHS	Number of right-hand sides, equal to the number of columns in the matrix B. $\text{NRHS} \geq 0$.
xA	On entry, upper or lower triangle of the matrix A. On exit, the Cholesky factorization of A as computed by xPBTRF.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $\text{LDA} \geq \text{NDIAG} + 1$.

xB	On entry, the $N \times \text{NRHS}$ right-hand side matrix B. On exit, the $N \times \text{NRHS}$ solution matrix X.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $\text{LDB} \geq \max(1, N)$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The i th argument, where $i = \text{INFO} $, had an illegal value. INFO > 0 The leading minor of order i of A, where $i = \text{INFO}$, is not positive definite. The factorization has not been completed and the solution could not be computed.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
C   INTEGER          LDA, LDB, N, NDIAG, NRHS
C   PARAMETER        (N = 4)
C   PARAMETER        (NDIAG = 1)
C   PARAMETER        (NRHS = 1)
C   PARAMETER        (LDA = NDIAG + 1)
C   PARAMETER        (LDB = N)
C
C   DOUBLE PRECISION A(LDA,N), B(LDB,NRHS)
C   INTEGER          ICOL, INFO, IROW
C
C   EXTERNAL          DPBSV
C
C   Initialize the array A to store in symmetric banded form
C   the 4x4 symmetric positive definite coefficient matrix A
C   with one subdiagonal and one superdiagonal shown below.
C   Initialize the array B to store the right hand side vector
C   b shown below.
C
C
C   A =
C   2  -1  0  0
C  -1  2  -1  0
C   0  -1  2  -1
C   0  0  -1  2
C
C   b =
C   6
C  12
C  12
C   6
C
C   DATA A /      8D8, 2.0D0, -1.0D0, 2.0D0,
C   $      -1.0D0, 2.0D0, -1.0D0, 2.0D0 /
C   DATA B / 6.0D0, 1.2D1, 1.2D1, 6.0D0 /
C
C   Print the initial values of the arrays.
C
C   PRINT 1000
C   PRINT 1010, A(2,1), A(1,2), 0.0D0, 0.0D0
C   PRINT 1010, A(1,2), A(2,2), A(1,3), 0.0D0

```

```

PRINT 1010, 0.0D0, A(1,3), A(2,3), A(1,4)
PRINT 1010, 0.0D0, 0.0D0, A(1,4), A(2,4)
PRINT 1020
PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, LDA)
PRINT 1030
PRINT 1040, B
C
C   Solve the system and print the results.
C
CALL DPBSV ('UPPER TRIANGLE OF A STORED', N, NDIAG, NRHS,
$          A, LDA, B, LDB, INFO)
IF (INFO .NE. 0) THEN
    PRINT 1050, INFO
    STOP 1
END IF
PRINT 1060
PRINT 1040, B
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F6.3))
1020 FORMAT (/1X, 'A in banded form: (* in unused elements)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (1X, F6.2)
1050 FORMAT (1X, 'Error solving Ax=b, INFO = ', I5)
1060 FORMAT (/1X, 'x:')
C
END

```

Sample Output

```

A in full form:
  2.000  -1.000   0.000   0.000
 -1.000   2.000  -1.000   0.000
  0.000  -1.000   2.000  -1.000
  0.000   0.000  -1.000   2.000

A in banded form: (* in unused elements)
*****  -1.000  -1.000  -1.000
  2.000   2.000   2.000   2.000

b:
  6.00
 12.00
 12.00
  6.00

x:
 18.00
 30.00
 30.00
 18.00

```

Solution to a Linear System in a Symmetric Positive Definite Matrix in Banded Storage (Expert Driver)

The subroutines described in this section solve a linear system $AX = B$ for a real symmetric (or Hermitian) positive definite matrix A in banded storage and general matrices B and X . These subroutines also compute forward error bounds and backward error estimates for the solution, estimate the condition number of the matrix A and, optionally, equilibrate the system before computing a solution. Note that the simple driver `xPBSV` is also available.

Calling Sequence

```
CALL DPBSVX (FACT, UPLO, N, NDIAG, NRHS, DA, LDA, DAF, LDAF, EQUED,
             DSCALE, DB, LDB, DX, LDX, DRCOND, DFERR, DBERR, DWORK,
             IWORK2, INFO)
CALL SPBSVX (FACT, UPLO, N, NDIAG, NRHS, SA, LDA, SAF, LDAF, EQUED,
             SSCALE, SB, LDB, SX, LDX, SRCOND, SFERR, SBERR, SWORK,
             IWORK2, INFO)
CALL ZPBSVX (FACT, UPLO, N, NDIAG, NRHS, ZA, LDA, ZAF, LDAF, EQUED,
             DSCALE, ZB, LDB, ZX, LDX, DRCOND, DFERR, DBERR, ZWORK,
             DWORK2, INFO)
CALL CPBSVX (FACT, UPLO, N, NDIAG, NRHS, CA, LDA, CAF, LDAF, EQUED,
             SSCALE, CB, LDB, CX, LDX, SRCOND, SFERR, SBERR, CWORK,
             SWORK2, INFO)

void dpbsvx (char fact, char uplo, int n, int ndiag, int nrhs,
             double *da, int lda, double *daf, int ldaf, char
             *equed, double *dscale, double *db, int ldb, double
             *dx, int ldx, double *drcond, double *dferr, double
             *dberr, int *info)
void spbsvx (char fact, char uplo, int n, int ndiag, int nrhs,
             float *sa, int lda, float *saf, int ldaf, char *equed,
             float *sscale, float *sb, int ldb, float *sx, int ldx,
             float *srcond, float *sferr, float *sberr, int *info)
void zpbsvx (char fact, char uplo, int n, int ndiag, int nrhs,
             doublecomplex *za, int lda, doublecomplex *zaf, int
             ldaf, char *equed, double *dscale, doublecomplex *zb,
             int ldb, doublecomplex *zx, int ldx, double *drcond,
             double *dferr, double *dberr, int *info)
```

```
void cpbsvx (char fact, char uplo, int n, int ndiag, int nrhs,
             complex *ca, int lda, complex *caf, int ldaf, char
             *equed, float *sscale, complex *cb, int ldb, complex
             *cx, int ldx, float *srcond, float *sferr, float
             *sberr, int *info)
```

Arguments

FACT Indicates whether or not the factored form of the matrix A is supplied on entry, and if not, whether the matrix A will be equilibrated before it is factored. FACT interacts with EQUED, A, and AF as shown in the table below:

FACT	EQUED entry	EQUED exit	A entry	A exit	AF entry	AF exit
'F', 'f'	'N', 'n'	Unchanged	Matrix A	Unchanged	Cholesky-factored A	Unchanged
'F', 'f'	'Y', 'y'	Unchanged	Equilibrated A	Unchanged	Cholesky-factored equilibrated A	Unchanged
'E', 'e'	Not used	'N'	Matrix A	Unchanged	Not used	Cholesky-factored A
'E', 'e'	Not used	'Y'	Matrix A	Equilibrated A	Not used	Cholesky-factored equilibrated A
'N', 'n'	Not used	'N'	Matrix A	Unchanged	Not used	Cholesky-factored A

The above table uses the following abbreviations:
Cholesky-factored = the UTU or LLT factored form of A as computed by xPBTRF
Equilibrated A = $\text{diag}(\text{SCALE}) \times A \times \text{diag}(\text{SCALE})$

UPLO Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal.
'U' or 'u' xA contains the upper triangle.
'L' or 'l' xA contains the lower triangle.

N Order of the matrix A. $N \geq 0$.

NDIAG Number of superdiagonals or subdiagonals of the matrix A. $N-1 \geq \text{NDIAG} \geq 0$ but if $N = 0$ then $\text{NDIAG} = 0$.
If UPLO = 'U' or 'u', NDIAG is the number of superdiagonals.
If UPLO = 'L' or 'l', NDIAG is the number of subdiagonals.

NRHS Number of right-hand sides, equal to the number of columns of the matrices B and X. $\text{NRHS} \geq 0$.

xA The contents of A vary depending on the values of FACT and EQUED. See the table for the argument FACT above.

LDA Leading dimension of the array A as specified in a dimension or type statement. $\text{LDA} \geq \text{NDIAG} + 1$.

xAF	The contents of AF vary depending on the values of FACT and EQUED. See the table for the argument FACT above.
LDAF	Leading dimension of the array AF as specified in a dimension or type statement. $LDAF \geq NDIAG + 1$.
EQUED	EQUED interacts with FACT as shown in the table above for the argument FACT. Note that one must be careful about specifying lowercase letters for the entry value of EQUED when FACT = 'F' or 'f' because ordinarily LAPACK will always return a capital letter. If the caller specifies 'n', or 'y' on entry then the lowercase letter will still be in EQUED on exit. The caller may later draw incorrect conclusions if the code contains a series of IF statements to determine whether EQUED = 'N' or 'Y'.
xSCALE	On entry, if FACT = 'F' or 'f' and EQUED = 'Y' or 'y' then SCALE contains the N scale factors that were used to equilibrate A. Each element of SCALE must be positive. If FACT or EQUED have other values, the entry values of SCALE are not used. On exit, if FACT = 'E' or 'e' and EQUED = 'Y' then SCALE contains the N scale factors that were used to equilibrate A. If FACT or EQUED have other values, SCALE is unchanged.
xB	On entry, the $N \times NRHS$ right-hand side matrix B. On exit, if EQUED = 'N', then B is not modified; if EQUED = 'Y', then B is overwritten by $diag(SCALE) \times B$.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
xX	On exit, the $N \times NRHS$ solution matrix for the original problem. Note that if the exit value of EQUED = 'Y' then A and B are modified on exit and the solution to the equilibrated system is $diag(SCALE)^{-1} \times X$.
LDX	Leading dimension of the array X as specified in a dimension or type statement. $LDX \geq \max(1, N)$.
xRCOND	On exit, an estimate of the reciprocal condition number of the matrix A after equilibration, if any, where the reciprocal condition number is defined to be $1 / (\ A\ \times \ A^{-1}\)$. The reciprocal of the condition number is estimated instead of the condition number itself to avoid overflow or division by zero. If RCOND is less than machine precision (in particular, if RCOND = 0) then A is singular to working precision. In this case, INFO > 0 is returned and the solution and error bounds are not computed.
xFERR	On exit, the estimated forward error bound for each solution vector $X^*(j)$ for $1 \leq j \leq NRHS$. If X' is the true solution corresponding to $X^*(j)$ then FERR(j) is an upper bound on the magnitude of the largest element in $X^*(j) - X'$ divided by the magnitude of the largest element in $X^*(j)$.

xBERR	On exit, BERR(j) is the smallest relative change in any element of A or B(*, j) that makes X(*, j) an exact solution to AX(*, j) = B(*, j) for $1 \leq j \leq \text{NRHS}$.
xWORK	Scratch array with a dimension of $3 \times N$ for real subroutines or $2 \times N$ for complex subroutines.
xWORK2	Scratch array with a dimension of N.
INFO	On exit:
	INFO = 0 Subroutine completed normally.
	INFO < 0 The ith argument, where $i = \text{INFO} $, had an illegal value.
	$1 \leq \text{INFO} \leq N$ The leading minor of order i of A, where $i = \text{INFO}$, is not positive definite. The factorization has not been completed, and the solution and error bounds could not be computed.
	INFO = N+1 RCOND is less than machine precision. The factorization has been completed, but the matrix is singular to working precision, and the solution and error bounds could not be computed.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER          LDA, LDAF, LDB, LDIWRK, LDWORK, LDX, N
  INTEGER          NDIAG, NRHS
  PARAMETER        (N = 4)
  PARAMETER        (NDIAG = 1)
  PARAMETER        (NRHS = 1)
  PARAMETER        (LDA = NDIAG + 1)
  PARAMETER        (LDAF = LDA)
  PARAMETER        (LDB = N)
  PARAMETER        (LDIWRK = N)
  PARAMETER        (LDX = N)
  PARAMETER        (LDWORK = 3 * N)

C
  DOUBLE PRECISION A(LDA,N), AF(LDAF,N), B(LDB,NRHS)
  DOUBLE PRECISION BERR(NRHS), FERR(NRHS), RCOND, SCALE(N)
  DOUBLE PRECISION WORK(LDWORK), X(LDX,NRHS)
  INTEGER          ICOL, INFO, IROW, IWORK(LDIWRK)
  CHARACTER*1      EQUED

C
  EXTERNAL          DPBSVX

C
C  Initialize the array A to store in symmetric banded form
C  the 4x4 symmetric positive definite coefficient matrix A
C  with one subdiagonal and one superdiagonal shown below.

```

```

C      Initialize the array B to store the right hand side
C      vector b shown below.
C
C      2   -1   0   0       6
C      A = -1   2  -1   0       b = 12
C      0   -1   2  -1       12
C      0   0   -1   2       6
C
C      DATA A /      8D8, 2.0D0, -1.0D0, 2.0D0,
$      -1.0D0, 2.0D0, -1.0D0, 2.0D0 /
C      DATA B / 6.0D0, 1.2D1, 1.2D1, 6.0D0 /
C
C      Print the initial values of the arrays.
C
C      PRINT 1000
C      PRINT 1010, A(2,1), A(1,2), 0.0D0, 0.0D0
C      PRINT 1010, A(1,2), A(2,2), A(1,3), 0.0D0
C      PRINT 1010, 0.0D0, A(1,3), A(2,3), A(1,4)
C      PRINT 1010, 0.0D0, 0.0D0, A(1,4), A(2,4)
C      PRINT 1020
C      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, LDA)
C      PRINT 1030
C      PRINT 1040, B
C
C      Solve the system and print the results.
C
C      CALL DPBSVX ('EQUILIBRATE A', 'UPPER TRIANGLE OF A STORED',
$      N, NDIAG, NRHS, A, LDA, AF, LDAF, EQUED, SCALE,
$      B, LDB, X, LDX, RCOND, FERR, BERR, WORK, IWORK,
$      INFO)
C      IF (INFO .NE. 0) THEN
C          PRINT 1050, INFO
C          IF (INFO .EQ. (N + 1)) THEN
C              PRINT 1060
C              STOP 1
C          END IF
C          STOP 2
C      END IF
C      PRINT 1070
C      PRINT 1040, X
C      PRINT 1080, 1.0D0 / RCOND
C      IF (EQUED .EQ. 'N') THEN
C          PRINT 1090
C      ELSE
C          PRINT 1100
C      END IF
C      PRINT 1110, (IROW, BERR(IROW), IROW = 1, NRHS)
C      PRINT 1120, (IROW, FERR(IROW), IROW = 1, NRHS)
C
C      1000 FORMAT (1X, 'A in full form:')
C      1010 FORMAT (4(3X, F6.3))
C      1020 FORMAT (/1X, 'A in banded form: (* in unused elements)')
C      1030 FORMAT (/1X, 'b:')
C      1040 FORMAT (1X, F6.2)
C      1050 FORMAT (1X, 'Error solving Ax=b, INFO = ', I5)
C      1060 FORMAT (1X, 'Matrix is singular to working precision.')

```

```

1070 FORMAT (/1X, 'x:')
1080 FORMAT (/1X, 'Estimated condition number of A: ', F7.2)
1090 FORMAT (1X, 'No equilibration was required for A.')
1100 FORMAT (1X, 'Diagonal equilibration was done on A.')
1110 FORMAT (/1X, 'Backward error bound for system #', I1, ': ',
$      E12.5)
1120 FORMAT (1X, 'Forward error bound for system #', I1, ': ',
$      E12.5)
C
      END

```

Sample Output

```

A in full form:
  2.000  -1.000   0.000   0.000
 -1.000   2.000  -1.000   0.000
  0.000  -1.000   2.000  -1.000
  0.000   0.000  -1.000   2.000

A in banded form: (* in unused elements)
*****  -1.000  -1.000  -1.000
  2.000   2.000   2.000   2.000

b:
  6.00
 12.00
 12.00
  6.00

x:
 18.00
 30.00
 30.00
 18.00

Estimated condition number of A:  12.00
No equilibration was required for A.

Backward error bound for system #1:  0.00000E+00
Forward error bound for system #1:  0.46185E-14

```

Cholesky Factorization of a Symmetric Positive Definite Matrix in Banded Storage

The subroutines described in this section compute a Cholesky factorization of a real symmetric (or Hermitian) positive definite matrix A in banded storage. It is typical to follow a call to `xPBTRF` with a call to `xPBSVX` or `xPBTRS` to solve a linear system $AX = B$, or to `xPBCON` to estimate the condition number of A .

Calling Sequence

```
CALL DPBTRF (UPLO, N, NDIAG, DA, LDA, INFO)
CALL SPBTRF (UPLO, N, NDIAG, SA, LDA, INFO)
CALL ZPBTRF (UPLO, N, NDIAG, ZA, LDA, INFO)
CALL CPBTRF (UPLO, N, NDIAG, CA, LDA, INFO)
```

```
void dpbtrf (char uplo, int n, int ndiag, double *da, int lda, int
             *info)
void spbtrf (char uplo, int n, int ndiag, float *sa, int lda, int
             *info)
void zpbtrf (char uplo, int n, int ndiag, doublecomplex *za, int
             lda, int *info)
void cpbtrf (char uplo, int n, int ndiag, complex *ca, int lda, int
             *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A . $N \geq 0$.
NDIAG	Number of superdiagonals or subdiagonals of the matrix A . $N-1 \geq \text{NDIAG} \geq 0$ but if $N = 0$ then $\text{NDIAG} = 0$. If UPLO = 'U' or 'u', NDIAG is the number of superdiagonals. If UPLO = 'L' or 'l', NDIAG is the number of subdiagonals.
xA	On entry, the upper or lower triangle of the matrix A . On exit, a Cholesky factorization of the matrix A .
LDA	Leading dimension of the array A as specified in a dimension or type statement. $\text{LDA} \geq \text{NDIAG} + 1$.
INFO	On exit: INFO = 0 Subroutine completed normally.

INFO < 0	The <i>i</i> th argument, where <i>i</i> = INFO , had an illegal value.
1 ≤ INFO ≤ N	The leading minor of order <i>i</i> of A, where <i>i</i> = INFO, is not positive definite. The factorization could not be completed.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C   INTEGER          LDA, LDB, N, NDIAG, NRHS
C   PARAMETER        (N = 4)
C   PARAMETER        (NDIAG = 1)
C   PARAMETER        (NRHS = 1)
C   PARAMETER        (LDA = NDIAG + 1)
C   PARAMETER        (LDB = N)
C
C   DOUBLE PRECISION A(LDA,N), B(LDB,NRHS)
C   INTEGER          ICOL, INFO, IROW
C
C   EXTERNAL          DPBTRF, DPBTRS
C   INTRINSIC         ABS
C
C   Initialize the array A to store in symmetric banded form
C   the 4x4 symmetric positive definite coefficient matrix A
C   with one subdiagonal and one superdiagonal shown below.
C   Initialize the array B to store the right hand side
C   vector b shown below.
C
C
C   A =  2  -1  0  0      b =  6
C   -1  2  -1  0      12
C   0  -1  2  -1      12
C   0  0  -1  2      6
C
C   DATA A /      8D8, 2.0D0, -1.0D0, 2.0D0,
$      -1.0D0, 2.0D0, -1.0D0, 2.0D0 /
C   DATA B / 6.0D0, 1.2D1, 1.2D1, 6.0D0 /
C
C   Print the initial values of the arrays.
C
C   PRINT 1000
C   PRINT 1010, A(2,1), A(1,2), 0.0D0, 0.0D0
C   PRINT 1010, A(1,2), A(2,2), A(1,3), 0.0D0
C   PRINT 1010, 0.0D0, A(1,3), A(2,3), A(1,4)
C   PRINT 1010, 0.0D0, 0.0D0, A(1,4), A(2,4)
C   PRINT 1020
C   PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, LDA)
C   PRINT 1030
C   PRINT 1040, B
C

```

```
C      Cholesky factor A.
C
      CALL DPBTRF ('UPPER TRIANGLE OF A STORED', N, NDIAG, A, LDA,
$          INFO)
      IF (INFO .NE. 0) THEN
          PRINT 1050, INFO
          STOP 1
      END IF
C
C      Use the factored form of A to solve Ax=b then print
C      the result.
C
      CALL DPBTRS ('UPPER TRIANGLE OF A STORED', N, NDIAG, NRHS,
$          A, LDA, B, LDB, INFO)
      IF (INFO .NE. 0) THEN
          PRINT 1060, ABS(INFO)
          STOP 1
      END IF
      PRINT 1070
      PRINT 1040, B
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F6.3))
1020 FORMAT (/1X, 'A in banded form: (* in unused elements)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (1X, F6.2)
1050 FORMAT (1X, 'Error factoring A, INFO = ', I5)
1060 FORMAT (1X, 'Illegal argument to DPBTRS, argument #', I1)
1070 FORMAT (/1X, 'x:')
C
      END
```

Sample Output

```
A in full form:
  2.000  -1.000   0.000   0.000
 -1.000   2.000  -1.000   0.000
  0.000  -1.000   2.000  -1.000
  0.000   0.000  -1.000   2.000

A in banded form: (* in unused elements)
*****
  2.000  -1.000  -1.000  -1.000
  2.000   2.000   2.000   2.000

b:
  6.00
 12.00
 12.00
  6.00
```

x: 18.00 30.00 30.00 18.00

Solution to a Linear System in a Cholesky-Factored Symmetric Positive Definite Matrix in Banded Storage

The subroutines described in this section solve a linear system $AX = B$ for a real symmetric (or Hermitian) positive definite matrix A in banded storage, which has been Cholesky-factored by `xPBTRF`, and general matrices B and X .

Calling Sequence

```
CALL DPBTRS (UPLO, N, NDIAG, NRHS, DA, LDA, DB, LDB, INFO)
CALL SPBTRS (UPLO, N, NDIAG, NRHS, SA, LDA, SB, LDB, INFO)
CALL ZPBTRS (UPLO, N, NDIAG, NRHS, ZA, LDA, ZB, LDB, INFO)
CALL CPBTRS (UPLO, N, NDIAG, NRHS, CA, LDA, CB, LDB, INFO)
```

```
void dpbtrs (char uplo, int n, int ndiag, int nrhs, double *da, int
             lda, double *db, int ldb, int *info)
void spbtrs (char uplo, int n, int ndiag, int nrhs, float *sa, int
             lda, float *sb, int ldb, int *info)
void zpbtrs (char uplo, int n, int ndiag, int nrhs, doublecomplex
             *za, int lda, doublecomplex *zb, int ldb, int *info)
void cpbtrs (char uplo, int n, int ndiag, int nrhs, complex *ca,
             int lda, complex *cb, int ldb, int *info)
```

Arguments

UPLO	Indicates whether <code>xA</code> contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' <code>xA</code> contains the upper triangle. 'L' or 'l' <code>xA</code> contains the lower triangle.
N	Order of the matrix A . $N \geq 0$.
NDIAG	Number of superdiagonals or subdiagonals of the matrix A . $N-1 \geq \text{NDIAG} \geq 0$ but if $N = 0$ then $\text{NDIAG} = 0$. If UPLO = 'U' or 'u', NDIAG is the number of superdiagonals. If UPLO = 'L' or 'l', NDIAG is the number of subdiagonals.
NRHS	Number of right-hand sides, equal to the number of columns in the matrices B and X . $\text{NRHS} \geq 0$.
<code>xA</code>	Cholesky factorization of the matrix A as computed by <code>xPBTRF</code> .
LDA	Leading dimension of the array A as specified in a dimension or type statement. $\text{LDA} \geq \text{NDIAG} + 1$.

xB	On entry, the N×NRHS right-hand side matrix B. On exit, the N×NRHS solution matrix X.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The <i>i</i> th argument, where <i>i</i> = INFO , had an illegal value.

Sample Program

```

C      PROGRAM TEST
C      IMPLICIT NONE
C
C      INTEGER          LDA, LDB, N, NDIAG, NRHS
C      PARAMETER        (N = 4)
C      PARAMETER        (NDIAG = 1)
C      PARAMETER        (NRHS = 1)
C      PARAMETER        (LDA = NDIAG + 1)
C      PARAMETER        (LDB = N)
C
C      DOUBLE PRECISION A(LDA,N), B(LDB,NRHS)
C      INTEGER          ICOL, INFO, IROW
C
C      EXTERNAL          DPBTRF, DPBTRS
C      INTRINSIC         ABS
C
C      Initialize the array A to store in symmetric banded form
C      the 4x4 symmetric positive definite coefficient matrix A
C      with one subdiagonal and one superdiagonal shown below.
C      Initialize the array B to store the right hand side
C      vector b shown below.
C
C      A =
C      2   -1   0   0
C      -1   2   -1  0
C      0   -1   2  -1
C      0   0   -1  2
C
C      b =
C      6
C      12
C      12
C      6
C
C      DATA A /      8D8, 2.0D0, -1.0D0, 2.0D0,
C      $      -1.0D0, 2.0D0, -1.0D0, 2.0D0 /
C      DATA B / 6.0D0, 1.2D1, 1.2D1, 6.0D0 /
C
C      Print the initial values of the arrays.
C
C      PRINT 1000
C      PRINT 1010, A(2,1), A(1,2), 0.0D0, 0.0D0
C      PRINT 1010, A(1,2), A(2,2), A(1,3), 0.0D0
C      PRINT 1010, 0.0D0, A(1,3), A(2,3), A(1,4)
C      PRINT 1010, 0.0D0, 0.0D0, A(1,4), A(2,4)
C      PRINT 1020

```

```
      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, LDA)
      PRINT 1030
      PRINT 1040, B
C
C      Cholesky factor A.
C
      CALL DPBTRF ('UPPER TRIANGLE OF A STORED', N, NDIAG, A, LDA,
$              INFO)
      IF (INFO .NE. 0) THEN
          PRINT 1050, INFO
          STOP 1
      END IF
C
C      Use the factored form of A to solve Ax=b then print result.
C
      CALL DPBTRS ('UPPER TRIANGLE OF A STORED', N, NDIAG, NRHS,
$              A, LDA, B, LDB, INFO)
      IF (INFO .NE. 0) THEN
          PRINT 1060, ABS(INFO)
          STOP 1
      END IF
      PRINT 1070
      PRINT 1040, B
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F6.3))
1020 FORMAT (/1X, 'A in banded form: (* in unused elements)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (1X, F6.2)
1050 FORMAT (1X, 'Error factoring A, INFO = ', I5)
1060 FORMAT (1X, 'Illegal argument to DPBTRS, argument #', I1)
1070 FORMAT (/1X, 'x:')
C
      END
```

Sample Output

```
A in full form:
  2.000  -1.000   0.000   0.000
 -1.000   2.000  -1.000   0.000
  0.000  -1.000   2.000  -1.000
  0.000   0.000  -1.000   2.000

A in banded form: (* in unused elements)
*****  -1.000  -1.000  -1.000
  2.000   2.000   2.000   2.000

b:
  6.00
 12.00
 12.00
  6.00
```

x: 18.00 30.00 30.00 18.00

Reciprocal Condition Number of a Cholesky-Factored Symmetric Positive Definite Matrix

The subroutines described in this section estimate the reciprocal condition number of a real symmetric (or Hermitian) positive definite matrix A, which has been Cholesky-factored by xPOTRF.

Calling Sequence

```
CALL DPOCON (UPLO, N, DA, LDA, DANORM, DRCOND, DWORK, IWORK2, INFO)
CALL SPOCON (UPLO, N, SA, LDA, SANORM, SRCOND, SWORK, IWORK2, INFO)
CALL ZPOCON (UPLO, N, ZA, LDA, DANORM, DRCOND, ZWORK, DWORK2, INFO)
CALL CPOCON (UPLO, N, CA, LDA, SANORM, SRCOND, CWORK, SWORK2, INFO)
```

```
void dpocon (char uplo, int n, double *da, int lda, double danorm,
             double *drcond, int *info)
void spocon (char uplo, int n, float *sa, int lda, float sanorm,
             float *srcond, int *info)
void zpocon (char uplo, int n, doublecomplex *za, int lda, double
             danorm, double *drcond, int *info)
void cpocon (char uplo, int n, complex *ca, int lda, float sanorm,
             float *srcond, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
xA	Cholesky factorization of the matrix A as computed by xPOTRF.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
xANORM	The 1-norm or ∞ -norm of matrix A.
xRCOND	On exit, the estimated reciprocal of the condition number of the matrix A where the reciprocal condition number of A is defined to be $1 / (ANORM \times \ A^{-1}\)$. The reciprocal of the condition number is estimated instead of the condition number itself to avoid overflow or division by zero. If RCOND is less than machine precision (in particular, if RCOND = 0) then A is singular to working precision.

xWORK	Scratch array with a dimension of $3 \times N$ for real subroutines or $2 \times N$ for complex subroutines.
xWORK2	Scratch array with a dimension of N .
INFO	On exit:
	INFO = 0 Subroutine completed normally.
	INFO < 0 The <i>i</i> th argument, where $i = INFO $, had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
C   INTEGER          LDA, LDIWRK, LDWORK, N
C   PARAMETER        (N = 4)
C   PARAMETER        (LDA = N)
C   PARAMETER        (LDIWRK = N)
C   PARAMETER        (LDWORK = 3 * N)
C
C   DOUBLE PRECISION A(LDA,N), ANORM, RCOND, WORK(LDWORK)
C   INTEGER          ICOL, INFO, IROW, IWORK(LDIWRK)
C
C   EXTERNAL          DPOCON, DPOTRF
C   INTRINSIC         ABS
C
C   Initialize the array A to store in symmetric form the 4x4
C   symmetric positive definite matrix A shown below.
C
C       2   -1   0   0
C   A = -1   2   -1   0
C       0   -1   2   -1
C       0   0   -1   2
C
C   DATA A /  2.0D0, 3*8D8, -1.0D0, 2.0D0, 2*8D8, 0.0D0, -1.0D0,
$           2.0D0, 8D8, 0.0D0, 0.0D0, -1.0D0, 2.0D0 /
C
C   Print the initial values of the arrays.
C
C   PRINT 1000
C   DO 100, IROW = 1, N
C       PRINT 1010, (A(ICOL,IROW), ICOL = 1, IROW - 1),
$           (A(IROW,ICOL), ICOL = IROW, N)
100 CONTINUE
C   PRINT 1020
C   PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, LDA)
C
C   Cholesky factor A.
C
C   CALL DPOTRF ('UPPER TRIANGLE OF A STORED', N, A, LDA, INFO)
C   IF (INFO .NE. 0) THEN

```

```
        PRINT 1030, INFO
        STOP 1
      END IF
C
C      Compute and print the condition number of A.
C
      ANORM = 3.0D0
      CALL DPOCON ('UPPER TRIANGLE FACTOR STORED', N, A, LDA,
$                ANORM, RCOND, WORK, IWORK, INFO)
      IF (INFO .NE. 0) THEN
        PRINT 1040, ABS(INFO)
        STOP 2
      END IF
      PRINT 1050, 1.0D0 / RCOND
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F6.3))
1020 FORMAT (/1X, 'A in symmetric form: (* in unused elements)')
1030 FORMAT (1X, 'Error factoring A, INFO = ', I5)
1040 FORMAT (1X, 'Illegal argument to DPOCON, argument #', I2)
1050 FORMAT (/1X, 'Estimated condition number of A = ', F8.4)
C
      END
```

Sample Output

```
A in full form:
  2.000  -1.000   0.000   0.000
 -1.000   2.000  -1.000   0.000
  0.000  -1.000   2.000  -1.000
  0.000   0.000  -1.000   2.000

A in symmetric form: (* in unused elements)
  2.000  -1.000   0.000   0.000
*****   2.000  -1.000   0.000
*****   *****   2.000  -1.000
*****   *****   *****   2.000

Estimated condition number of A =  9.0000
```

Equilibration Scale Factors for a Symmetric Positive Definite Matrix

The subroutines described in this section compute equilibration scale factors for a real symmetric (or Hermitian) positive definite matrix A. Equilibration is intended to reduce the condition number of A with respect to the 2-norm. SCALE contains scale factors chosen so that the scaled matrix B with elements $B(i,j) = \text{SCALE}(i) \times A(i,j) \times \text{SCALE}(j)$ has ones on the diagonal. This choice of SCALE puts the condition number of B within a factor N of the smallest possible condition number over all possible diagonal scalings. Note that these subroutines only compute the scale factors for A. Additional code must be written to actually equilibrate A.

Calling Sequence

```
CALL DPOEQU  (N, DA, LDA, DSCALE, DSCOND, DAMAX, INFO)
CALL SPOEQU  (N, SA, LDA, SSCALE, SSCOND, SAMAX, INFO)
CALL ZPOEQU  (N, ZA, LDA, DSCALE, DSCOND, DAMAX, INFO)
CALL CPOEQU  (N, CA, LDA, SSCALE, SSCOND, SAMAX, INFO)

void dpoequ  (int n, double *da, int lda, double *dscale, double
              *dscond, double *damax, int *info)
void spoequ  (int n, float *sa, int lda, float *sscale, float
              *sscond, float *samax, int *info)
void zpoequ  (int n, doublecomplex *za, int lda, double *dscale,
              double *dscond, double *damax, int *info)
void cpoequ  (int n, complex *ca, int lda, float *sscale, float
              *sscond, float *samax, int *info)
```

Arguments

N	Order of the matrix A. $N \geq 0$.
xA	Matrix A; only the diagonal elements of A are referenced.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
xSCALE	On exit, the N scale factors for A.
xSCOND	On exit, the ratio of the smallest SCALE(i) to the largest SCALE(i). If $\text{SCOND} \geq 0.1$ and AMAX is not close to overflow or underflow, it is not worth scaling by SCALE.
xAMAX	On exit, the absolute value of the largest element of A. If AMAX is very close to overflow or underflow, A should be scaled regardless of the value of SCOND.

INFO	On exit:	
	INFO = 0	Subroutine completed normally.
	INFO < 0	The ith argument, where i = INFO , had an illegal value.
	INFO > 0	The ith diagonal entry, where i = INFO, is non-positive.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C   INTEGER          LDA, LDSCAL, N
C   PARAMETER        (N = 4)
C   PARAMETER        (LDA = N)
C   PARAMETER        (LDSCAL = N)
C
C   DOUBLE PRECISION A(LDA,N), AMAX, SCALE(N), SCOND
C   INTEGER          ICOL, INFO, IROW
C
C   EXTERNAL          DPOEQU
C   INTRINSIC         ABS
C
C   Initialize the array A to store in symmetric form the 4x4
C   symmetric positive definite matrix A shown below.
C
C   4096    -2    0    0
C   A =    -2    256  -2    0
C           0    -2   16   -2
C           0    0   -2    1
C
C   DATA A /  4.096D3, 3*8D8, -2.0D0, 2.56D2, 2*8D8, 0.0D0,
$             -2.0D0, 1.6D1, 8D8, 0.0D0, 0.0D0, -2.0D0, 1.0D0 /
C
C   Print the initial values of the arrays.
C
C   PRINT 1000
C   DO 100, IROW = 1, N
C       PRINT 1010, (A(ICOL,IROW), ICOL = 1, IROW - 1),
$               (A(IROW,ICOL), ICOL = IROW, N)
100 CONTINUE
C   PRINT 1020
C   PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, LDA)
C
C   Compute the scale factors to use to equilibrate A.
C
C   CALL DPOEQU (N, A, LDA, SCALE, SCOND, AMAX, INFO)
C   IF (INFO .NE. 0) THEN
C       PRINT 1030, INFO
C       STOP 1
C   END IF

```

```

C
C      Apply the scale factors to A then print the result.
C
      DO 210, ICOL = 1, N
        DO 200, IROW = 1, ICOL
          A(IROW,ICOL) = A(IROW,ICOL) * SCALE(IROW) * SCALE(ICOL)
200    CONTINUE
210  CONTINUE
      PRINT 1040
      PRINT 1050, SCALE
      PRINT 1060
      DO 220, IROW = 1, N
        PRINT 1010, (A(ICOL,IROW), ICOL = 1, IROW - 1),
          $          (A(IROW,ICOL), ICOL = IROW, N)
220  CONTINUE
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F10.5))
1020 FORMAT (/1X, 'A in symmetric form: (* in unused elements)')
1030 FORMAT (1X, 'Error computing scale factors for A, INFO =',
          $    1X, I5)
1040 FORMAT (/1X, 'Scale factors:')
1050 FORMAT (1X, F8.5)
1060 FORMAT (/1X, 'Equilibrated A:')
C
      END

```

Sample Output

```

A in full form:
 4096.00000   -2.00000    0.00000    0.00000
  -2.00000   256.00000   -2.00000    0.00000
  0.00000   -2.00000   16.00000   -2.00000
  0.00000    0.00000   -2.00000    1.00000

A in symmetric form: (* in unused elements)
 4096.00000   -2.00000    0.00000    0.00000
 *****   256.00000   -2.00000    0.00000
 *****   *****   16.00000   -2.00000
 *****   *****   *****   1.00000

Scale factors:
 0.01562
 0.06250
 0.25000
 1.00000

```

Equilibrated A:

1.00000	-0.00195	0.00000	0.00000
-0.00195	1.00000	-0.03125	0.00000
0.00000	-0.03125	1.00000	-0.50000
0.00000	0.00000	-0.50000	1.00000

Refined Solution to a Linear System in a Cholesky-Factored Symmetric Positive Definite Matrix

The subroutines described in this section refine the solution to a linear system $AX = B$ for a real symmetric (or Hermitian) positive definite matrix A, which has been Cholesky-factored by xPOTRF, and general matrices B and X. These subroutines also compute forward error bounds and backward error estimates for the refined solution.

Calling Sequence

```
CALL DPORFS (UPLO, N, NRHS, DA, LDA, DAF, LDAF, DB, LDB, DX, LDX,
             DFERR, DBERR, DWORK, IWORK2, INFO)
CALL SPORFS (UPLO, N, NRHS, SA, LDA, SAF, LDAF, SB, LDB, SX, LDX,
             SFERR, SBERR, SWORK, IWORK2, INFO)
CALL ZPORFS (UPLO, N, NRHS, ZA, LDA, ZAF, LDAF, ZB, LDB, ZX, LDX,
             DFERR, DBERR, ZWORK, DWORK2, INFO)
CALL CPORFS (UPLO, N, NRHS, CA, LDA, CAF, LDAF, CB, LDB, CX, LDX,
             SFERR, SBERR, CWORK, SWORK2, INFO)

void dporfs (char uplo, int n, int nrhs, double *da, int lda,
             double *daf, int ldaf, double *db, int ldb, double *dx,
             int ldx, double *dferr, double *dberr, int *info)
void sporfs (char uplo, int n, int nrhs, float *sa, int lda, float
             *saf, int ldaf, float *sb, int ldb, float *sx, int ldx,
             float *sferr, float *sberr, int *info)
void zporfs (char uplo, int n, int nrhs, doublecomplex *za, int
             lda, doublecomplex *zaf, int ldaf, doublecomplex *zb,
             int ldb, doublecomplex *zx, int ldx, double *dferr,
             double *dberr, int *info)
void cporfs (char uplo, int n, int nrhs, complex *ca, int lda,
             complex *caf, int ldaf, complex *cb, int ldb, complex
             *cx, int ldx, float *sferr, float *sberr, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.

NRHS	Number of right-hand sides, equal to the number of columns of the matrices B and X. $\text{NRHS} \geq 0$.
xA	Upper or lower triangle of the matrix A.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $\text{LDA} \geq \max(1, N)$.
xAF	Cholesky factorization of the matrix A as computed by xPOTRF.
LDAF	Leading dimension of the array AF as specified in a dimension or type statement. $\text{LDAF} \geq \max(1, N)$.
xB	The $N \times \text{NRHS}$ right-hand side matrix B.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $\text{LDB} \geq \max(1, N)$.
xx	On entry, the $N \times \text{NRHS}$ solution matrix X. On exit, the refined $N \times \text{NRHS}$ solution matrix X.
LDX	Leading dimension of the array X as specified in a dimension or type statement. $\text{LDX} \geq \max(1, N)$.
xFERR	On exit, the estimated forward error bound for each solution vector $X(*, j)$ for $1 \leq j \leq \text{NRHS}$. If X' is the true solution corresponding to $X(*, j)$ then $\text{FERR}(j)$ is an upper bound on the magnitude of the largest element in $X(*, j) - X'$ divided by the magnitude of the largest element in $X(*, j)$.
xBERR	On exit, $\text{BERR}(j)$ is the smallest relative change in any element of A or $B(*, j)$ that makes $X(*, j)$ an exact solution to $AX(*, j) = B(*, j)$ for $1 \leq j \leq \text{NRHS}$.
xWORK	Scratch array with a dimension of $3 \times N$ for real subroutines or $2 \times N$ for complex subroutines.
xWORK2	Scratch array with a dimension of N.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The i th argument, where $i = \text{INFO} $, had an illegal value.

Sample Program

```
PROGRAM TEST
IMPLICIT NONE

C
INTEGER          LDA, LDAF, LDB, LDIWRK, LDWORK, LDX
INTEGER          N, NRHS
PARAMETER        (N = 4)
PARAMETER        (LDA = N)
PARAMETER        (LDAF = N)
PARAMETER        (LDB = N)
PARAMETER        (LDIWRK = N)
PARAMETER        (LDWORK = 3 * N)
```

```

PARAMETER      (LDB = LDB)
PARAMETER      (NRHS = 1)
C
DOUBLE PRECISION A(LDA,N), AF(LDAF,N), B(LDB,NRHS)
DOUBLE PRECISION B1(LDB,NRHS), BERR(NRHS), FERR(NRHS)
DOUBLE PRECISION WORK(LDWORK), X(LDX,NRHS)
INTEGER        ICOL, INFO, IROW, IWORK(LDIWRK)
C
EXTERNAL        DCOPY, DPORFS, DPOTRF, DPOTRS, DSYMV
INTRINSIC       ABS
C
C      Initialize the array A to store in symmetric form the
C      coefficient matrix A shown below. Initialize the array B
C      to store the right hand side vector b shown below.
C
C      10**5   10**-5   10**-6   10**-7           10**4
C      A = 10**-5 10**5   10**-6 10**-7           b = 10**4
C      10**-6 10**-6 10**5   10**-7           10**4
C      10**-7 10**-7 10**-7 10**5           10**4
C
DATA A / 1.0D5, 1.0D-5, 1.0D-6, 1.0D-7,
$      8D180, 1.0D5, 1.0D-6, 1.0D-7,
$      8D180, 8D180, 1.0D5, 1.0D-7,
$      8D180, 8D180, 8D180, 1.0D5 /
DATA B / 1.0D4, 1.0D4, 1.0D4, 1.0D4 /
C
C      Print the initial value of the arrays.
C
PRINT 1000
DO 100, IROW = 1, N
    PRINT 1010, (A(IROW,ICOL), ICOL = 1, IROW),
$      (A(ICOL,IROW), ICOL = IROW + 1, N)
100 CONTINUE
PRINT 1020
DO 110, IROW = 1, LDA
    PRINT 1010, (A(IROW,ICOL), ICOL = 1, N)
110 CONTINUE
CALL DCPY (LDA*N, A, 1, AF, 1)
PRINT 1030
PRINT 1040, B
CALL DCPY (LDB*NRHS, B, 1, X, 1)
C
C      Cholesky factor A.
C
CALL DPOTRF ('LOWER TRIANGLE OF A STORED', N, AF, LDAF,
$      INFO)
IF (INFO .NE. 0) THEN
    PRINT 1050, INFO
    STOP 1
END IF
C
C      Solve Ax=b and print the solution.
C
CALL DPOTRS ('LOWER TRIANGLE OF A STORED', N, NRHS, AF,
$      LDAF, X, LDX, INFO)
IF (INFO .NE. 0) THEN

```

```

        PRINT 1060, ABS(INFO)
        STOP 3
      END IF
      PRINT 1070
      PRINT 1040, X
      CALL DSYMV ('LOWER TRIANGLE OF A STORED', N, 1.0D0, A, LDA,
$             X, 1, 0.0D0, B1, 1)
      PRINT 1080
      PRINT 1040, B1
C
C   Refine the solution to Ax=b and print the refined solution.
C
      CALL DPORFS ('LOWER TRIANGLE OF A STORED', N, NRHS, A, LDA,
$             AF, LDAF, B, LDB, X, LDX, FERR, BERR, WORK,
$             IWORK, INFO)
      IF (INFO .NE. 0) THEN
        PRINT 1090, ABS(INFO)
        STOP 4
      END IF
      PRINT 1100
      PRINT 1040, X
      CALL DSYMV ('LOWER TRIANGLE OF A STORED', N, 1.0D0, A, LDA,
$             X, 1, 0.0D0, B1, 1)
      PRINT 1110
      PRINT 1040, B1
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F14.7))
1020 FORMAT (/1X, 'A in symmetric form: (* in unused elements)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (3X, E25.17)
1050 FORMAT (1X, 'Error factoring A, INFO = ', I5)
1060 FORMAT (1X, 'Illegal argument to DPOTRS, argument #', I2)
1070 FORMAT (/1X, 'Initial solution to Ax=b:')
1080 FORMAT (/1X, 'Ax with the initial x:')
1090 FORMAT (1X, 'Illegal argument to DPORFS, INFO = ', I2)
1100 FORMAT (/1X, 'Refined solution to Ax=b:')
1110 FORMAT (/1X, 'Ax with refined x:')
C
      END

```

Sample Output

```

A in full form:
100000.0000000    0.0000100    0.0000010    0.0000001
  0.0000100    100000.0000000    0.0000010    0.0000001
  0.0000010    0.0000010    100000.0000000    0.0000001
  0.0000001    0.0000001    0.0000001    100000.0000000

```

```

A in symmetric form: (* in unused elements)
100000.0000000 ***** ***** *****
    0.0000100 100000.0000000 ***** *****
    0.0000010    0.0000010 100000.0000000 *****
    0.0000001    0.0000001    0.0000001 100000.0000000

b:
  0.10000000000000000E+05
  0.10000000000000000E+05
  0.10000000000000000E+05
  0.10000000000000000E+05

Initial solution to Ax=b:
  0.99999999988899982E-01
  0.99999999988899982E-01
  0.9999999997899963E-01
  0.999999999699954E-01

Ax with the initial x:
  0.9999999999999982E+04
  0.9999999999999982E+04
  0.9999999999999964E+04
  0.9999999999999964E+04

Refined solution to Ax=b:
  0.9999999998889996E-01
  0.9999999998889996E-01
  0.9999999997899991E-01
  0.999999999699968E-01

Ax with refined x:
  0.10000000000000000E+05
  0.10000000000000000E+05
  0.10000000000000000E+05
  0.9999999999999982E+04

```


Solution to a Linear System in a Symmetric Positive Definite Matrix (Simple Driver)

The subroutines described in this section solve a linear system $AX = B$ for a real symmetric (or Hermitian) positive definite matrix A and general matrices B and X . Note that the expert driver `xPOSVX` is also available.

Calling Sequence

```
CALL DPOSV (UPLO, N, NRHS, DA, LDA, DB, LDB, INFO)
CALL SPOSV (UPLO, N, NRHS, SA, LDA, SB, LDB, INFO)
CALL ZPOSV (UPLO, N, NRHS, ZA, LDA, ZB, LDB, INFO)
CALL CPOSV (UPLO, N, NRHS, CA, LDA, CB, LDB, INFO)

void dposv (char uplo, int n, int nrhs, double *da, int lda,
            double *db, int ldb, int *info)
void sposv (char uplo, int n, int nrhs, float *sa, int lda, float
            *sb, int ldb, int *info)
void zposv (char uplo, int n, int nrhs, doublecomplex *za, int
            lda, doublecomplex *zb, int ldb, int *info)
void cposv (char uplo, int n, int nrhs, complex *ca, int lda,
            complex *cb, int ldb, int *info)
```

Arguments

UPLO	Indicates whether <code>xA</code> contains the upper or lower triangle of the matrix. The legal values for <code>UPLO</code> are listed below. Any values not listed below are illegal. 'U' or 'u' <code>xA</code> contains the upper triangle. 'L' or 'l' <code>xA</code> contains the lower triangle.
N	Order of the matrix A . $N \geq 0$.
NRHS	Number of right-hand sides, equal to the number of columns of the matrix B . $NRHS \geq 0$.
xA	On entry, the upper or lower triangle of the matrix A . On exit, the Cholesky factorization of the matrix A as computed by <code>xPOTRF</code> .
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
xB	On entry, the $N \times NRHS$ right-hand side matrix B . On exit, the $N \times NRHS$ solution matrix X .

LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.	
INFO	On exit:	
	INFO = 0	Subroutine completed normally.
	INFO < 0	The <i>i</i> th argument, where $i = \text{INFO} $, had an illegal value.
	INFO > 0	The leading minor of order <i>i</i> of A, where $i = \text{INFO}$, is not positive definite. The factorization has not been completed, and the solution could not be computed.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C   INTEGER          LDA, LDB, N, NRHS
C   PARAMETER        (N = 4)
C   PARAMETER        (LDA = N)
C   PARAMETER        (LDB = N)
C   PARAMETER        (NRHS = 1)
C
C   DOUBLE PRECISION A(LDA,N), B(LDB,NRHS)
C   INTEGER          ICOL, INFO, IROW
C
C   EXTERNAL          DPOSV
C
C   Initialize the array A to store in symmetric form the 4x4
C   symmetric positive definite matrix A shown below.
C   Initialize the array B to store the right hand side
C   vector b shown below.
C
C
C   2   -1   0   0       6
C   A = -1   2  -1   0       b = 12
C       0  -1   2  -1       12
C       0   0  -1   2       6
C
C   DATA A /  2.0D0, 3*8D8, -1.0D0, 2.0D0, 2*8D8, 0.0D0, -1.0D0,
$           2.0D0, 8D8, 0.0D0, 0.0D0, -1.0D0, 2.0D0 /
C   DATA B / 6.0D0, 1.2D1, 1.2D1, 6.0D0 /
C
C   Print the initial values of the arrays.
C
C
C   PRINT 1000
C   DO 100, IROW = 1, N
C       PRINT 1010, (A(ICOL,IROW), ICOL = 1, IROW - 1),
$           (A(IROW,ICOL), ICOL = IROW, N)
100 CONTINUE
C   PRINT 1020
C   PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, LDA)
C   PRINT 1030

```

```
      PRINT 1040, B
C
C      Solve Ax=b and print the solution.
C
      CALL DPOSV ('UPPER TRIANGLE OF A STORED', N, NRHS, A, LDA,
$             B, LDB, INFO)
      IF (INFO .NE. 0) THEN
          PRINT 1050, INFO
          STOP 1
      END IF
      PRINT 1060
      PRINT 1040, B
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F6.3))
1020 FORMAT (/1X, 'A in symmetric form: (* in unused elements)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (1X, F6.3)
1050 FORMAT (1X, 'Error solving Ax=b, INFO = ', I5)
1060 FORMAT (/1X, 'x:')
C
      END
```

Sample Output

```
A in full form:
  2.000  -1.000   0.000   0.000
 -1.000   2.000  -1.000   0.000
  0.000  -1.000   2.000  -1.000
  0.000   0.000  -1.000   2.000

A in symmetric form: (* in unused elements)
  2.000  -1.000   0.000   0.000
*****   2.000  -1.000   0.000
*****   *****   2.000  -1.000
*****   *****   *****   2.000

b:
 6.000
12.000
12.000
 6.000

x:
18.000
30.000
30.000
18.000
```

Solution to a Linear System in a Symmetric Positive Definite Matrix (Expert Driver)

The subroutines described in this section solve a linear system $AX = B$ for a real symmetric (or Hermitian) positive definite matrix A and general matrices B and X . These subroutines also compute forward error bounds and backward error estimates for the solution, estimate the condition number of the matrix A and, optionally, equilibrate the system before computing a solution. Note that the simple driver `xPOSV` is also available.

Calling Sequence

```
CALL DPOSVX (FACT, UPLO, N, NRHS, DA, LDA, DAF, LDAF, EQUED,
             DSCALE, DB, LDB, DX, LDX, DRCOND, DFERR, DBERR, DWORK,
             IWORK2, INFO)
CALL SPOSVX (FACT, UPLO, N, NRHS, SA, LDA, SAF, LDAF, EQUED,
             SSCALE, SB, LDB, SX, LDX, SRCOND, SFERR, SBERR, SWORK,
             IWORK2, INFO)
CALL ZPOSVX (FACT, UPLO, N, NRHS, ZA, LDA, ZAF, LDAF, EQUED,
             DSCALE, ZB, LDB, ZX, LDX, DRCOND, DFERR, DBERR, ZWORK,
             DWORK2, INFO)
CALL CPOSVX (FACT, UPLO, N, NRHS, CA, LDA, CAF, LDAF, EQUED,
             SSCALE, CB, LDB, CX, LDX, SRCOND, SFERR, SBERR, CWORK,
             SWORK2, INFO)

w
void dposvx (char fact, char uplo, int n, int nrhs, double *da, int
            lda, double *daf, int ldaf, char *equed, double
            *dscale, double *db, int ldb, double *dx, int ldx,
            double *drcond, double *dferr, double *dberr, int
            *info)
void sposvx (char fact, char uplo, int n, int nrhs, float *sa, int
            lda, float *saf, int ldaf, char *equed, float *sscale,
            float *sb, int ldb, float *sx, int ldx, float *srcond,
            float *sferr, float *sberr, int *info)
void zposvx (char fact, char uplo, int n, int nrhs, doublecomplex
            *za, int lda, doublecomplex *zaf, int ldaf, char
            *equed, double *dscale, doublecomplex *zb, int ldb,
            doublecomplex *zx, int ldx, double *drcond, double
            *dferr, double *dberr, int *info)
```

```
void cposvx (char fact, char uplo, int n, int nrhs, complex *ca,
            int lda, complex *caf, int ldaf, char *equed, float
            *sscale, complex *cb, int ldb, complex *cx, int ldx,
            float *srcond, float *sferr, float *sberr, int *info)
```

Arguments

FACT Indicates whether or not the factored form of the matrix A is supplied on entry, and if not, whether the matrix A will be equilibrated before it is factored. FACT interacts with EQUED, A, and AF as shown in the table below:

FACT	EQUED entry	EQUED exit	A entry	A exit	AF entry	AF exit
'F', 'f'	'N', 'n'	Unchanged	Matrix A	Unchanged	Cholesky-factored A	Unchanged
'F', 'f'	'Y', 'y'	Unchanged	Equilibrated A	Unchanged	Cholesky-factored equilibrated A	Unchanged
'E', 'e'	Not used	'N'	Matrix A	Unchanged	Not used	Cholesky-factored A
'E', 'e'	Not used	'Y'	Matrix A	Equilibrated A	Not used	Cholesky-factored equilibrated A
'N', 'n'	Not used	'N'	Matrix A	Unchanged	Not used	Cholesky-factored A

The above table uses the following abbreviations:

Cholesky-factored = the UTU or LLT factored form of A as computed by xPOTRF

Equilibrated A = $\text{diag}(\text{SCALE}) \times A \times \text{diag}(\text{SCALE})$

UPLO

Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal.

'U' or 'u' xA contains the upper triangle.

'L' or 'l' xA contains the lower triangle.

N

Order of the matrix A. $N \geq 0$.

NRHS

Number of right-hand sides, equal to the number of columns of the matrices B and X. $\text{NRHS} \geq 0$.

xA

The contents of A vary depending on the values of FACT and EQUED. See the table for the argument FACT above.

LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
xAF	The contents of AF vary depending on the values of FACT and EQUED. See the table for the argument FACT above.
LDAF	Leading dimension of the array AF as specified in a dimension or type statement. $LDAF \geq \max(1, N)$.
EQUED	EQUED interacts with FACT as shown in the table above for the argument FACT. Note that one must be careful about specifying lowercase letters for the entry value of EQUED when FACT = 'F' or 'f' because ordinarily LAPACK will always return a capital letter. If the caller specifies 'n', or 'y' on entry then the lowercase letter will still be in EQUED on exit. The caller may later draw incorrect conclusions if the code contains a series of IF statements to determine whether EQUED = 'N' or 'Y'.
xSCALE	On entry, if FACT = 'F' or 'f' and EQUED = 'Y' or 'y' then SCALE contains the N scale factors that were used to equilibrate A. Each element of SCALE must be positive. If FACT or EQUED have other values, the entry values of SCALE are not used. On exit, if FACT = 'E' or 'e' and EQUED = 'Y' then SCALE contains the N scale factors that were used to equilibrate A. If FACT or EQUED have other values, SCALE is unchanged.
xB	On entry, the $N \times NRHS$ right-hand side matrix B. On exit, if EQUED = 'N', then B is not modified; if EQUED = 'Y', then B is overwritten by $\text{diag}(\text{SCALE}) \times B$.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
xX	On exit, the $N \times NRHS$ solution matrix for the original problem. Note that if the exit value of EQUED = 'Y' then A and B are modified on exit and the solution to the equilibrated system is $\text{diag}(\text{SCALE})^{-1} \times X$.
LDX	Leading dimension of the array X as specified in a dimension or type statement. $LDX \geq \max(1, N)$.
xRCOND	On exit, an estimate of the reciprocal condition number of the matrix A after equilibration, if any, where the reciprocal condition number is defined to be $1 / (\ A\ \times \ A^{-1}\)$. The reciprocal of the condition number is estimated instead of the condition number itself to avoid overflow or division by zero. If RCOND is less than machine precision (in particular, if RCOND = 0) then A is singular to working precision. In this case, INFO > 0 is returned and the solution and error bounds are not computed.

xFERR	On exit, the estimated forward error bound for each solution vector $X(*, j)$ for $1 \leq j \leq \text{NRHS}$. If X' is the true solution corresponding to $X(*, j)$ then $\text{FERR}(j)$ is an upper bound on the magnitude of the largest element in $X(*, j) - X'$ divided by the magnitude of the largest element in $X(*, j)$.
xBERR	On exit, $\text{BERR}(j)$ is the smallest relative change in any element of A or $B(*, j)$ that makes $X(*, j)$ an exact solution to $AX(*, j) = B(*, j)$ for $1 \leq j \leq \text{NRHS}$.
xWORK	Scratch array with a dimension of $3 \times N$ for real subroutines or $2 \times N$ for complex subroutines.
xWORK2	Scratch array with a dimension of N .
INFO	On exit: INFO = 0 INFO < 0 $1 \leq \text{INFO} \leq N$ INFO = N+1 Subroutine completed normally. The ith argument, where $i = \text{INFO}$, had an illegal value. The leading minor of order i of A, where $i = \text{INFO}$, is not positive definite. The factorization has not been completed, and the solution and error bounds could not be computed. RCOND is less than machine precision. The factorization has been completed, but the matrix is singular to working precision, and the solution and error bounds could not be computed.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
INTEGER          LDA, LDAF, LDB, LDIWRK, LDWORK, LDX
INTEGER          N, NRHS
PARAMETER        (N = 4)
PARAMETER        (LDA = N)
PARAMETER        (LDAF = LDA)
PARAMETER        (LDB = N)
PARAMETER        (LDIWRK = N)
PARAMETER        (LDX = N)
PARAMETER        (LDWORK = 3 * N)
PARAMETER        (NRHS = 1)

C
DOUBLE PRECISION A(LDA,N), AF(LDAF,N), B(LDB,NRHS)
DOUBLE PRECISION BERR(NRHS), FERR(NRHS), RCOND, SCALE(N)
DOUBLE PRECISION WORK(LDWORK), X(LDX,NRHS)

```

```

      INTEGER          ICOL, INFO, IROW, IWORK(LDIWRK)
      CHARACTER*1      EQUED
C
      EXTERNAL          DPOSVX
C
C      Initialize the array A to store in symmetric form the 4x4
C      symmetric positive definite matrix A shown below.
C      Initialize the array B to store the right hand side
C      vector b shown below.
C
C      A =
C      2   -1   0   0      b = 6
C      -1   2   -1   0      12
C      0   -1   2   -1      12
C      0    0   -1   2      6
C
      DATA A /  2.0D0, 3*8D8, -1.0D0, 2.0D0, 2*8D8, 0.0D0,
$             -1.0D0, 2.0D0, 8D8, 0.0D0, 0.0D0, -1.0D0, 2.0D0 /
      DATA B / 6.0D0, 1.2D1, 1.2D1, 6.0D0 /
C
C      Print the initial values of the arrays.
C
      PRINT 1000
      DO 100, IROW = 1, N
        PRINT 1010, (A(ICOL,IROW), ICOL = 1, IROW - 1),
$              (A(IROW,ICOL), ICOL = IROW, N)
100  CONTINUE
      PRINT 1020
      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, LDA)
      PRINT 1030
      PRINT 1040, B
C
C      Solve Ax=b and print the solution.
C
      CALL DPOSVX ('EQUILIBRATE A', 'UPPER TRIANGLE OF A STORED',
$              N, NRHS, A, LDA, AF, LDAF, EQUED, SCALE,
$              B, LDB, X, LDX, RCOND, FERR, BERR, WORK,
$              IWORK, INFO)
      IF (INFO .NE. 0) THEN
        PRINT 1050, INFO
        IF (INFO .EQ. (N + 1)) THEN
          PRINT 1060
          STOP 1
        END IF
        STOP 2
      END IF
      PRINT 1070
      PRINT 1040, X
      PRINT 1080, 1.0D0 / RCOND
      IF (EQUED .EQ. 'N') PRINT 1090
      IF (EQUED .EQ. 'R') PRINT 1100
      IF (EQUED .EQ. 'C') PRINT 1110
      IF (EQUED .EQ. 'B') PRINT 1120
      PRINT 1130, (IROW, BERR(IROW), IROW = 1, NRHS)
      PRINT 1140, (IROW, FERR(IROW), IROW = 1, NRHS)
C
1000  FORMAT (1X, 'A in full form:')

```



```
1010 FORMAT (4(3X, F6.3))
1020 FORMAT (/1X, 'A in symmetric form: (* in unused elements)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (1X, F6.2)
1050 FORMAT (1X, 'Error solving Ax=b, INFO = ', I5)
1060 FORMAT (1X, 'Matrix is singular to working precision.')
1070 FORMAT (/1X, 'x:')
1080 FORMAT (/1X, 'Estimated condition number of A: ', F7.2)
1090 FORMAT (1X, 'No equilibration was required for A.')
1100 FORMAT (1X, 'Row equilibration was done on A.')
1110 FORMAT (1X, 'Column equilibration was done on A.')
1120 FORMAT (1X, 'Row and column equilibration was done on A.')
1130 FORMAT (/1X, 'Backward error for system #', I1, ': ',
$      E12.6)
1140 FORMAT (1X, 'Forward error for system #', I1, ': ', E12.6)
C
      END
```

Sample Output

```
A in full form:
  2.000  -1.000   0.000   0.000
 -1.000   2.000  -1.000   0.000
  0.000  -1.000   2.000  -1.000
  0.000   0.000  -1.000   2.000

A in symmetric form: (* in unused elements)
  2.000  -1.000   0.000   0.000
*****   2.000  -1.000   0.000
*****   *****   2.000  -1.000
*****   *****   *****   2.000

b:
  6.00
 12.00
 12.00
  6.00

x:
 18.00
 30.00
 30.00
 18.00

Estimated condition number of A:  12.00
No equilibration was required for A.

Backward error for system #1:  0.000000E+00
Forward error for system #1:  0.577316E-14
```

Cholesky Factorization of a Symmetric Positive Definite Matrix

The subroutines described in this section compute a Cholesky factorization of a real symmetric (or Hermitian) positive definite matrix A. It is typical to follow a call to xPOTRF with a call to xPOSVX or xPOTRS to solve a linear system $AX = B$, to xPOCON to estimate the condition number of A, or to xPOTRI to compute A^{-1} .

Calling Sequence

```
CALL DPOTRF (UPLO, N, DA, LDA, INFO)
CALL SPOTRF (UPLO, N, SA, LDA, INFO)
CALL ZPOTRF (UPLO, N, ZA, LDA, INFO)
CALL CPOTRF (UPLO, N, CA, LDA, INFO)
```

```
void dpotrf (char uplo, int n, double *da, int lda, int *info)
void spotrf (char uplo, int n, float *sa, int lda, int *info)
void zpotrf (char uplo, int n, doublecomplex *za, int lda, int
            *info)
void cpotrf (char uplo, int n, complex *ca, int lda, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
xA	On entry, the upper or lower triangle of the matrix A. On exit, a Cholesky factorization of the matrix A.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The <i>i</i> th argument, where $i = INFO $, had an illegal value. INFO > 0 The leading minor of order <i>i</i> of A, where $i = INFO$, is not positive definite. The factorization could not be completed.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
C   INTEGER          LDA, LDB, N, NRHS
C   PARAMETER        (N = 4)
C   PARAMETER        (NRHS = 1)
C   PARAMETER        (LDA = N)
C   PARAMETER        (LDB = N)
C
C   DOUBLE PRECISION A(LDA,N), B(LDB,NRHS)
C   INTEGER          ICOL, INFO, IROW
C
C   EXTERNAL          DPOTRF, DPOTRS
C   INTRINSIC         ABS
C
C   Initialize the array A to store in symmetric form the
C   4x4 symmetric positive definite coefficient matrix A
C   shown below. Initialize the array B to store the right
C   hand side vector b shown below.
C
C
C   A =
C   2  -1  0  0
C  -1  2  -1  0
C   0  -1  2  -1
C   0   0  -1  2
C
C   b =
C   6
C  12
C  12
C   6
C
C   DATA A / 2.0D0, 3*8D8, -1.0D0, 2.0D0, 2*8D8, 0.0D0,
$          -1.0D0, 2.0D0, 8D8, 0.0D0, 0.0D0, -1.0D0, 2.0D0 /
C   DATA B / 6.0D0, 1.2D1, 1.2D1, 6.0D0 /
C
C   Print the initial values of the arrays.
C
C   PRINT 1000
C   DO 100, IROW = 1, N
C     PRINT 1010, (A(ICOL,IROW), ICOL = 1, IROW - 1),
$           (A(IROW,ICOL), ICOL = IROW, N)
C   100 CONTINUE
C   PRINT 1020
C   PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, LDA)
C   PRINT 1030
C   PRINT 1040, B
C
C   Cholesky factor A.
C
C   CALL DPOTRF ('UPPER TRIANGLE OF A STORED', N, A, LDA, INFO)
C   IF (INFO .NE. 0) THEN
C     PRINT 1050, INFO
C     STOP 1
C   END IF
C
C   Use the factored form of A to solve Ax=b then print
C   the result.
C

```

```

      CALL DPOTRS ('UPPER TRIANGLE OF A STORED', N, NRHS, A, LDA,
$      B, LDB, INFO)
      IF (INFO .NE. 0) THEN
        PRINT 1060, ABS(INFO)
        STOP 1
      END IF
      PRINT 1070
      PRINT 1040, B
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F6.3))
1020 FORMAT (/1X, 'A in symmetric form: (* in unused elements)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (1X, F6.2)
1050 FORMAT (1X, 'Error factoring A, INFO = ', I5)
1060 FORMAT (1X, 'Illegal argument to DPOTRS, argument #', I1)
1070 FORMAT (/1X, 'x:')
C
      END

```

Sample Output

```

A in full form:
  2.000  -1.000   0.000   0.000
 -1.000   2.000  -1.000   0.000
  0.000  -1.000   2.000  -1.000
  0.000   0.000  -1.000   2.000

A in symmetric form: (* in unused elements)
  2.000  -1.000   0.000   0.000
*****   2.000  -1.000   0.000
*****   *****   2.000  -1.000
*****   *****   *****   2.000

b:
  6.00
 12.00
 12.00
  6.00

x:
 18.00
 30.00
 30.00
 18.00

```

Inverse of a Cholesky-Factored Symmetric Positive Definite Matrix

The subroutines described in this section compute the inverse of a real symmetric (or Hermitian) matrix A, which has been Cholesky-factored by xPOTRF.

Calling Sequence

```
CALL DPOTRI (UPLO, N, DA, LDA, INFO)
CALL SPOTRI (UPLO, N, SA, LDA, INFO)
CALL ZPOTRI (UPLO, N, ZA, LDA, INFO)
CALL CPOTRI (UPLO, N, CA, LDA, INFO)

void dpotri (char uplo, int n, double *da, int lda, int *info)
void spotri (char uplo, int n, float *sa, int lda, int *info)
void zpotri (char uplo, int n, doublecomplex *za, int lda, int
            *info)
void cpotri (char uplo, int n, complex *ca, int lda, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
xA	On entry, the Cholesky factorization of the matrix A as computed by xPOTRF. On exit, the symmetric (or Hermitian) inverse of the matrix A.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The i th argument, where $i = INFO $, had an illegal value. INFO > 0 Element (i,i) of the factor U or L, where $i = INFO$, is zero. The matrix is therefore singular and its inverse could not be computed.


```
      DO 200, IROW = 1, N
        PRINT 1010, (A(ICOL,IROW), ICOL = 1, IROW - 1),
          $          (A(IROW,ICOL), ICOL = IROW, N)
      200 CONTINUE
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F6.3))
1020 FORMAT (/1X, 'A in symmetric form: (* in unused elements)')
1030 FORMAT (1X, 'Error factoring A, INFO = ', I5)
1040 FORMAT (1X, 'Error computing inverse of A, INFO = ', I5)
1050 FORMAT (/1X, 'A**(-1):')
C
      END
```

Sample Output

```
A in full form:
  2.000   1.000   0.000   0.000
  1.000   2.000   1.000   0.000
  0.000   1.000   2.000   1.000
  0.000   0.000   1.000   1.000

A in symmetric form: (* in unused elements)
  2.000   1.000   0.000   0.000
  ***** 2.000   1.000   0.000
  ***** ***** 2.000   1.000
  ***** ***** ***** 1.000

A**(-1):
  1.000  -1.000   1.000  -1.000
 -1.000   2.000  -2.000   2.000
  1.000  -2.000   3.000  -3.000
 -1.000   2.000  -3.000   4.000
```

Solution to a Linear System in a Cholesky-Factored Symmetric Positive Definite Matrix

The subroutines described in this section solve a linear system $AX = B$ for a real symmetric (or Hermitian) positive definite matrix A , which has been Cholesky-factored by xPOTRF, and general matrices B and X .

Calling Sequence

```
CALL DPOTRS (UPLO, N, NRHS, DA, LDA, DB, LDB, INFO)
CALL SPOTRS (UPLO, N, NRHS, SA, LDA, SB, LDB, INFO)
CALL ZPOTRS (UPLO, N, NRHS, ZA, LDA, ZB, LDB, INFO)
CALL CPOTRS (UPLO, N, NRHS, CA, LDA, CB, LDB, INFO)

void dpotrs (char uplo, int n, int nrhs, double *da, int lda,
             double *db, int ldb, int *info)
void spotrs (char uplo, int n, int nrhs, float *sa, int lda, float
             *sb, int ldb, int *info)
void zpotrs (char uplo, int n, int nrhs, doublecomplex *za, int
             lda, doublecomplex *zb, int ldb, int *info)
void cpotrs (char uplo, int n, int nrhs, complex *ca, int lda,
             complex *cb, int ldb, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
NRHS	Number of right-hand sides, equal to the number of columns of the matrix B. $NRHS \geq 0$.
xA	Cholesky factorization of the matrix A as computed by xPOTRF.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
xB	On entry, the $N \times NRHS$ right-hand side matrix B. On exit, the $N \times NRHS$ solution matrix X.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.

INFO	On exit:	
	INFO = 0	Subroutine completed normally.
	INFO < 0	The ith argument, where i = INFO , had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C      INTEGER          LDA, LDB, N, NRHS
C      PARAMETER        (N = 4)
C      PARAMETER        (NRHS = 1)
C      PARAMETER        (LDA = N)
C      PARAMETER        (LDB = N)
C
C      DOUBLE PRECISION A(LDA,N), B(LDB,NRHS)
C      INTEGER          ICOL, INFO, IROW
C
C      EXTERNAL          DPOTRF, DPOTRS
C      INTRINSIC         ABS
C
C      Initialize the array A to store in symmetric form the
C      4x4 symmetric positive definite coefficient matrix A shown
C      below. Initialize the array B to store the right hand side
C      vector b shown below.
C
C      A =
C      2  -1  0  0      b =
C      -1  2  -1  0      6
C      0  -1  2  -1      12
C      0  0  -1  2      12
C                        6
C
C      DATA A / 2.0D0, 3*8D8, -1.0D0, 2.0D0, 2*8D8, 0.0D0, -1.0D0,
C      $          2.0D0, 8D8, 0.0D0, 0.0D0, -1.0D0, 2.0D0 /
C      DATA B / 6.0D0, 1.2D1, 1.2D1, 6.0D0 /
C
C      Print the initial values of the arrays.
C
C      PRINT 1000
C      DO 100, IROW = 1, N
C          PRINT 1010, (A(ICOL,IROW), ICOL = 1, IROW - 1),
C      $              (A(IROW,ICOL), ICOL = IROW, N)
100 CONTINUE
C      PRINT 1020
C      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, LDA)
C      PRINT 1030
C      PRINT 1040, B
C
C      Cholesky factor A.
C
C      CALL DPOTRF ('UPPER TRIANGLE OF A STORED', N, A, LDA, INFO)

```

```

        IF (INFO .NE. 0) THEN
            PRINT 1050, INFO
            STOP 1
        END IF
C
C      Use the factored form of A to solve Ax=b then print
C      the result.
C
        CALL DPOTRS ('UPPER TRIANGLE OF A STORED', N, NRHS, A, LDA,
$              B, LDB, INFO)
        IF (INFO .NE. 0) THEN
            PRINT 1060, ABS(INFO)
            STOP 1
        END IF
        PRINT 1070
        PRINT 1040, B
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F6.3))
1020 FORMAT (/1X, 'A in symmetric form: (* in unused elements)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (1X, F6.2)
1050 FORMAT (1X, 'Error factoring A, INFO = ', I5)
1060 FORMAT (1X, 'Illegal argument to DPOTRS, argument #', I1)
1070 FORMAT (/1X, 'x:')
C
        END

```

Sample Output

```

A in full form:
  2.000  -1.000   0.000   0.000
 -1.000   2.000  -1.000   0.000
  0.000  -1.000   2.000  -1.000
  0.000   0.000  -1.000   2.000

A in symmetric form: (* in unused elements)
  2.000  -1.000   0.000   0.000
*****  2.000  -1.000   0.000
*****  *****  2.000  -1.000
*****  *****  *****  2.000

b:
  6.00
 12.00
 12.00
  6.00

```

```
x:  
18.00  
30.00  
30.00  
18.00
```

Reciprocal Condition Number of a Cholesky-Factored Symmetric Positive Definite Matrix in Packed Storage

The subroutines described in this section estimate the reciprocal condition number of a real symmetric (or Hermitian) positive definite matrix A in packed storage, which has been Cholesky-factored by xPPTRF.

Calling Sequence

```
CALL DPPCON (UPLO, N, DA, DANORM, DRCOND, DWORK, IWORK2, INFO)
CALL SPPCON (UPLO, N, SA, SANORM, SRCOND, SWORK, IWORK2, INFO)
CALL ZPPCON (UPLO, N, ZA, DANORM, DRCOND, ZWORK, DWORK2, INFO)
CALL CPPCON (UPLO, N, CA, SANORM, SRCOND, CWORK, SWORK2, INFO)
```

```
void dppcon (char uplo, int n, double *da, double danorm, double
             *drcond, int *info)
void sppcon (char uplo, int n, float *sa, float sanorm, float
             *srcond, int *info)
void zppcon (char uplo, int n, doublecomplex *za, double danorm,
             double *drcond, int *info)
void cppcon (char uplo, int n, complex *ca, float sanorm, float
             *srcond, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
xA	Cholesky factorization of the matrix A, as computed by xPPTRF. The dimension of xA is $(N \times N + N) / 2$.
xANORM	The 1-norm or ∞ -norm of matrix A.
xRCOND	On exit, the estimated reciprocal of the condition number of the matrix A where the reciprocal condition number of A is defined to be $1 / (ANORM \times \ A^{-1}\)$. The reciprocal of the condition number is estimated instead of the condition number itself to avoid overflow or division by zero. If RCOND is less than machine precision (in particular, if RCOND = 0) then A is singular to working precision.

xWORK	Scratch array with a dimension of $3 \times N$ for real subroutines or $2 \times N$ for complex subroutines.	
xWORK2	Scratch array with a dimension of N .	
INFO	On exit:	
	INFO = 0	Subroutine completed normally.
	INFO < 0	The i th argument, where $i = INFO $, had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C   INTEGER          LDA, LDIWRK, LDWORK, N
C   PARAMETER        (N = 4)
C   PARAMETER        (LDA = (N * (N + 1)) / 2)
C   PARAMETER        (LDIWRK = N)
C   PARAMETER        (LDWORK = 3 * N)
C
C   DOUBLE PRECISION A(LDA), ANORM, RCOND, WORK(LDWORK)
C   INTEGER          INFO, IWORK(LDIWRK)
C
C   EXTERNAL          DPPCON, DPPTRF
C   INTRINSIC         ABS
C
C   Initialize the array A to store in packed symmetric form the
C   4x4 symmetric positive definite matrix A shown below.
C
C       2   -1   0   0
C   A = -1   2   -1   0
C       0   -1   2   -1
C       0   0   -1   2
C
C   DATA A / 2.0D0, -1.0D0, 2.0D0, 0.0D0, -1.0D0, 2.0D0,
$          0.0D0, 0.0D0, -1.0D0, 2.0D0 /
C
C   Print the initial values of the arrays.
C
C   PRINT 1000
C   PRINT 1010, A(1), A(2), A(4), A(7)
C   PRINT 1010, A(2), A(3), A(5), A(8)
C   PRINT 1010, A(4), A(5), A(6), A(9)
C   PRINT 1010, A(7), A(8), A(9), A(10)
C
C   Cholesky factor A.
C
C   CALL DPPTRF ('UPPER TRIANGLE OF A STORED', N, A, INFO)
C   IF (INFO.NE. 0) THEN
C       PRINT 1020, INFO
C       STOP 1

```

```

      END IF
C
C      Compute and print the condition number of A.
C
      ANORM = 4.0D0
      CALL DPPCON ('UPPER TRIANGLE FACTOR STORED', N, A,
$                ANORM, RCOND, WORK, IWORK, INFO)
      IF (INFO .NE. 0) THEN
        PRINT 1030, ABS(INFO)
        STOP 2
      END IF
      PRINT 1040, 1.0D0 / RCOND
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (4(3X, F6.3))
1020 FORMAT (1X, 'Error factoring A, INFO = ', I5)
1030 FORMAT (1X, 'Illegal argument to DPPCON, argument #', I2)
1040 FORMAT (/1X, 'Estimated condition number of A = ', F8.4)
C
      END

```

Sample Output

```

A:
  2.000   -1.000    0.000    0.000
 -1.000    2.000   -1.000    0.000
  0.000   -1.000    2.000   -1.000
  0.000    0.000   -1.000    2.000

Estimated condition number of A = 12.0000

```

Equilibration Scale Factors for a Symmetric Positive Definite Matrix in Packed Storage

The subroutines described in this section compute equilibration scale factors for a real symmetric (or Hermitian) positive definite matrix A in packed storage. Equilibration is intended to reduce the condition number of A with respect to the 2-norm. $SCALE$ contains scale factors chosen so that the scaled matrix B with elements $B(i,j) = SCALE(i) \times A(i,j) \times SCALE(j)$ has ones on the diagonal. This choice of $SCALE$ puts the condition number of B within a factor N of the smallest possible condition number over all possible diagonal scalings. Note that these subroutines only compute the scale factors for A . Additional code must be written to actually equilibrate A .

Calling Sequence

```
CALL DPPEQU (UPLO, N, DA, DSCALE, DSCOND, DAMAX, INFO)
CALL SPPEQU (UPLO, N, SA, SSCALE, SSCOND, SAMAX, INFO)
CALL ZPPEQU (UPLO, N, ZA, DSCALE, DSCOND, DAMAX, INFO)
CALL CPPEQU (UPLO, N, CA, SSCALE, SSCOND, SAMAX, INFO)

void dppequ (char uplo, int n, double *da, double *dscale, double
             *scond, double *damax, int *info)
void sppequ (char uplo, int n, float *sa, float *sscale, float
             *scond, float *samax, int *info)
void zppequ (char uplo, int n, doublecomplex *za, double *dscale,
             double *scond, double *damax, int *info)
void cppequ (char uplo, int n, complex *ca, float *sscale, float
             *scond, float *samax, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A . $N \geq 0$.
xA	Upper or lower triangle of the matrix A . The dimension of xA is $(N \times N + N) / 2$.
$xSCALE$	On exit, the N scale factors for A .

xSCOND	On exit, the ratio of the smallest SCALE(i) to the largest SCALE(i). If $SCOND \geq 0.1$ and AMAX is not close to overflow or underflow, it is not worth scaling by SCALE.						
xAMAX	On exit, the absolute value of the largest element of A. If AMAX is very close to overflow or underflow, A should be scaled regardless of the value of SCOND.						
INFO	On exit: <table> <tr> <td>INFO = 0</td><td>Subroutine completed normally.</td></tr> <tr> <td>INFO < 0</td><td>The ith argument, where i = INFO , had an illegal value.</td></tr> <tr> <td>INFO > 0</td><td>The ith diagonal entry, where i = INFO, is non-positive.</td></tr> </table>	INFO = 0	Subroutine completed normally.	INFO < 0	The ith argument, where i = INFO , had an illegal value.	INFO > 0	The ith diagonal entry, where i = INFO, is non-positive.
INFO = 0	Subroutine completed normally.						
INFO < 0	The ith argument, where i = INFO , had an illegal value.						
INFO > 0	The ith diagonal entry, where i = INFO, is non-positive.						

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C   INTEGER          LDA, LDSCAL, N
C   PARAMETER        (N = 4)
C   PARAMETER        (LDA = (N * (N + 1)) / 2)
C   PARAMETER        (LDSCAL = N)
C
C   DOUBLE PRECISION A(LDA), AMAX, SCALE(N), SCOND
C   INTEGER          ICOL, INFO, IROW
C
C   EXTERNAL          DPPEQU
C   INTRINSIC         ABS
C
C   Initialize the array A to store in symmetric form the upper
C   triangle of the 4x4 symmetric positive definite matrix A
C   shown below.
C
C   4096    -2    0    0
C   A =    -2   256   -2    0
C           0    -2   16   -2
C           0    0   -2    1
C
C   DATA A / 4.096D3, -2.0D0, 2.56D2, 0.0D0, -2.0D0, 1.6D1,
C   $         0.0D0, 0.0D0, -2.0D0, 1.0D0 /
C
C   Print the initial values of the arrays.
C
C   PRINT 1000
C   PRINT 1010, A(1), A(2), A(4), A(7)
C   PRINT 1010, A(2), A(3), A(5), A(8)
C   PRINT 1010, A(4), A(5), A(6), A(9)
C   PRINT 1010, A(7), A(8), A(9), A(10)
C

```



```
C      Compute the scale factors to use to equilibrate A.
C
      CALL DPPEQU ('UPPER TRIANGLE OF A STORED', N, A, SCALE,
$          SCOND, AMAX, INFO)
      IF (INFO .NE. 0) THEN
          PRINT 1030, INFO
          STOP 1
      END IF
C
C      Apply the scale factors to A then print the result.
C
      DO 210, ICOL = 1, N
          DO 200, IROW = 1, ICOL
              A(IROW + (ICOL - 1) * ICOL / 2) =
$          A(IROW + (ICOL - 1) * ICOL / 2) * SCALE(IROW) *
$          SCALE(ICOL)
200      CONTINUE
210      CONTINUE
          PRINT 1040
          PRINT 1050, SCALE
          PRINT 1060
          PRINT 1010, A(1), A(2), A(4), A(7)
          PRINT 1010, A(2), A(3), A(5), A(8)
          PRINT 1010, A(4), A(5), A(6), A(9)
          PRINT 1010, A(7), A(8), A(9), A(10)
C
1000      FORMAT (1X, 'A:')
1010      FORMAT (4(3X, F10.5))
1020      FORMAT (/1X, 'A in symmetric form: (* in unused elements)')
1030      FORMAT (1X, 'Error computing scale factors for A, INFO =',
$          1X, I5)
1040      FORMAT (/1X, 'Scale factors:')
1050      FORMAT (3X, F8.6)
1060      FORMAT (/1X, 'Equilibrated A:')
C
      END
```

Sample Output

```
A:
 4096.00000   -2.00000    0.00000    0.00000
   -2.00000   256.00000   -2.00000    0.00000
    0.00000   -2.00000   16.00000   -2.00000
    0.00000    0.00000   -2.00000    1.00000

Scale factors:
 0.015625
 0.062500
 0.250000
 1.000000
```

Equilibrated A:			
1.00000	-0.00195	0.00000	0.00000
-0.00195	1.00000	-0.03125	0.00000
0.00000	-0.03125	1.00000	-0.50000
0.00000	0.00000	-0.50000	1.00000

Refined Solution to a Linear System in a Cholesky-Factored Symmetric Positive Definite Matrix in Packed Storage

The subroutines described in this section refine the solution to a linear system $AX = B$ for a real symmetric (or Hermitian) positive definite matrix A in packed storage, which has been Cholesky-factored by `xPPTRF`, and general matrices B and X . These subroutines also compute forward error bounds and backward error estimates for the refined solution.

Calling Sequence

```
CALL DPPRFS (UPLO, N, NRHS, DA, DAF, DB, LDB, DX, LDX, DFERR,
             DBERR, DWORK, IWORK2, INFO)
CALL SPPRFS (UPLO, N, NRHS, SA, SAF, SB, LDB, SX, LDX, SFERR,
             SBERR, SWORK, IWORK2, INFO)
CALL ZPPRFS (UPLO, N, NRHS, ZA, ZAF, ZB, LDB, ZX, LDX, DFERR,
             DBERR, ZWORK, DWORK2, INFO)
CALL CPPRFS (UPLO, N, NRHS, CA, CAF, CB, LDB, CX, LDX, SFERR,
             SBERR, CWORK, SWORK2, INFO)

void dpprfs (char uplo, int n, int nrhs, double *da, double *daf,
            double *db, int ldb, double *dx, int ldx, double
            *dferr, double *dberr, int *info)
void spprfs (char uplo, int n, int nrhs, float *sa, float *saf,
            float *sb, int ldb, float *sx, int ldx, float *sferr,
            float *sberr, int *info)
void zpprfs (char uplo, int n, int nrhs, doublecomplex *za,
            doublecomplex *zaf, doublecomplex *zb, int ldb,
            doublecomplex *zx, int ldx, double *dferr, double
            *dberr, int *info)
void cpprfs (char uplo, int n, int nrhs, complex *ca, complex *caf,
            complex *cb, int ldb, complex *cx, int ldx, float
            *sferr, float *sberr, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A . $N \geq 0$.

NRHS	Number of right-hand sides, equal to the number of of the matrices B and X. $NRHS \geq 0$.
xA	Upper or lower triangle of the matrix A. The dimension of xA is $(N \times N + N) / 2$.
xAF	Cholesky factorization of A as computed by xPPTRF. The dimension of xAF is $(N \times N + N) / 2$.
xB	The $N \times NRHS$ right-hand side matrix B.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
xx	On entry, the $N \times NRHS$ solution matrix X. On exit, the refined $N \times NRHS$ solution matrix X.
LDX	Leading dimension of the array X as specified in a dimension or type statement. $LDX \geq \max(1, N)$.
xFERR	On exit, the estimated forward error bound for each solution vector $X(*, j)$ for $1 \leq j \leq NRHS$. If X' is the true solution corresponding to $X(*, j)$ then $FERR(j)$ is an upper bound on the magnitude of the largest element in $X(*, j) - X'$ divided by the magnitude of the largest element in $X(*, j)$.
xBERR	On exit, $BERR(j)$ is the smallest relative change in any element of A or $B(*, j)$ that makes $X(*, j)$ an exact solution to $AX(*, j) = B(*, j)$ for $1 \leq j \leq NRHS$.
xWORK	Scratch array with a dimension of $3 \times N$ for real subroutines or $2 \times N$ for complex subroutines.
xWORK2	Scratch array with a dimension of N.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where $i = INFO $, had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
INTEGER          LDA, LDAF, LDB, LDIWRK, LDWORK, LDX
INTEGER          N, NRHS
PARAMETER        (N = 4)
PARAMETER        (LDA = (N * (N + 1)) / 2)
PARAMETER        (LDAF = LDA)
PARAMETER        (LDB = N)
PARAMETER        (LDIWRK = N)
PARAMETER        (LDWORK = 3 * N)
PARAMETER        (LDX = LDB)
PARAMETER        (NRHS = 1)

```

```

C
C      DOUBLE PRECISION  A(LDA), AF(LDAF), B(LDB,NRHS)
C      DOUBLE PRECISION  B1(LDB,NRHS), BERR(NRHS), FERR(NRHS)
C      DOUBLE PRECISION  WORK(LDWORK), X(LDX,NRHS)
C      INTEGER           INFO, IWORK(LDIWRK)
C
C      EXTERNAL           DCOPY, DPPRFS, DPPTRF, DPPTRS, DSPMV
C      INTRINSIC          ABS
C
C      Initialize the array A to store in symmetric form the
C      coefficient matrix A shown below. Initialize the array
C      B to store the right hand side vector b shown below.
C
C      10**5   10**-5  10**-6  10**-7           3 * 10**4
C      A = 10**-5  10**5   10**-6  10**-7       b = 3 * 10**3
C      10**-6  10**-6  10**5   10**-7           3 * 10**2
C      10**-7  10**-7  10**-7   10**5           3 * 10**1
C
C      DATA A / 1.0D5, 1.0D-5, 1.0D-6, 1.0D-7, 1.0D5, 1.0D-6,
$      1.0D-7, 1.0D5, 1.0D-7, 1.0D5 /
C      DATA B / 3.0D4, 3.0D3, 3.0D2, 3.0D1 /
C
C      Print the initial values of the arrays.
C
C      PRINT 1000
C      PRINT 1010, A(1), A(2), A(3), A(4)
C      PRINT 1010, A(2), A(5), A(6), A(7)
C      PRINT 1010, A(3), A(6), A(8), A(9)
C      PRINT 1010, A(4), A(7), A(9), A(10)
C      CALL DCOPY (LDA, A, 1, AF, 1)
C      PRINT 1020
C      PRINT 1030, B
C      CALL DCOPY (LDB*NRHS, B, 1, X, 1)
C
C      Cholesky factor A.
C
C      CALL DPPTRF ('LOWER TRIANGLE OF A STORED', N, AF, INFO)
C      IF (INFO .NE. 0) THEN
C          PRINT 1040, INFO
C          STOP 1
C      END IF
C
C      Solve Ax=b and print the solution.
C
C      CALL DPPTS ('LOWER TRIANGLE OF A STORED', N, NRHS, AF,
$      X, LDX, INFO)
C      IF (INFO .NE. 0) THEN
C          PRINT 1050, ABS(INFO)
C          STOP 2
C      END IF
C      PRINT 1060
C      PRINT 1030, X
C      CALL DSPMV ('LOWER TRIANGLE OF A STORED', N, 1.0D0, A, X, 1,
$      0.0D0, B1, 1)
C      PRINT 1070
C      PRINT 1030, B1

```

```

C
C      Refine the solution to Ax=b and print the refined solution.
C
      CALL DPPRFS ('LOWER TRIANGLE OF A STORED', N, NRHS, A, AF,
$          B, LDB, X, LDX, FERR, BERR, WORK, IWORK, INFO)
      IF (INFO .NE. 0) THEN
          PRINT 1080, ABS(INFO)
          STOP 3
      END IF
      PRINT 1090
      PRINT 1030, X
      CALL DSPMV ('LOWER TRIANGLE OF A STORED', N, 1.0D0, A, X, 1,
$          0.0D0, B1, 1)
      PRINT 1100
      PRINT 1030, B1
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (4(3X, F14.7))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (3X, E25.17)
1040 FORMAT (1X, 'Error factoring A, INFO = ', I5)
1050 FORMAT (1X, 'Illegal argument to DPPTRS, argument #', I2)
1060 FORMAT (/1X, 'Initial solution to Ax=b:')
1070 FORMAT (/1X, 'Ax with the initial x:')
1080 FORMAT (1X, 'Illegal argument to DPPRFS, INFO = ', I2)
1090 FORMAT (/1X, 'Refined solution to Ax=b:')
1100 FORMAT (/1X, 'Ax with refined x:')
C
      END

```

Sample Output

```

A:
100000.00000000      0.0000100      0.00000010      0.00000001
   0.0000100 100000.00000000      0.00000010      0.00000001
   0.00000010   0.00000010 100000.00000000      0.00000001
   0.00000001   0.00000001   0.00000001 100000.00000000

b:
 0.30000000000000000000E+05
 0.30000000000000000000E+04
 0.30000000000000000000E+03
 0.30000000000000000000E+02

Initial solution to Ax=b:
 0.29999999999999999999E+00
 0.29999999999999999999E-01
 0.29999999999999999999E-02
 0.29999999999999999999E-03

```

```
Ax with the initial x:
0.2999999999999985E+05
0.2999999999999995E+04
0.2999999999999989E+03
0.3000000000000004E+02
```

```
Refined solution to Ax=b:
0.2999999999999969E+00
0.299999999999998E-01
0.299999999999999E-02
0.2999999999999967E-03
```

```
Ax with refined x:
0.2999999999999996E+05
0.3000000000000000E+04
0.3000000000000000E+03
0.3000000000000004E+02
```

Solution to a Linear System in a Symmetric Positive Definite Matrix in Packed Storage (Simple Driver)

The subroutines described in this section solve a linear system $AX = B$ for a real symmetric (or Hermitian) positive definite matrix A in packed storage and general matrices B and X . Note that the expert driver `xPPSVX` is also available.

Calling Sequence

```
CALL DPPSV (UPLO, N, NRHS, DA, DB, LDB, INFO)
CALL SPPSV (UPLO, N, NRHS, SA, SB, LDB, INFO)
CALL ZPPSV (UPLO, N, NRHS, ZA, ZB, LDB, INFO)
CALL CPPSV (UPLO, N, NRHS, CA, CB, LDB, INFO)

void dppsv (char uplo, int n, int nrhs, double *da, double *db,
            int ldb, int *info)
void sppsv (char uplo, int n, int nrhs, float *sa, float *sb, int
            ldb, int *info)
void zppsv (char uplo, int n, int nrhs, doublecomplex *za,
            doublecomplex *zb, int ldb, int *info)
void cppsv (char uplo, int n, int nrhs, complex *ca, complex *cb,
            int ldb, int *info)
```

Arguments

UPLO	Indicates whether <code>xA</code> contains the upper or lower triangle of the matrix. The legal values for <code>UPLO</code> are listed below. Any values not listed below are illegal. 'U' or 'u' <code>xA</code> contains the upper triangle. 'L' or 'l' <code>xA</code> contains the lower triangle.
N	Order of the matrix A . $N \geq 0$.
NRHS	Number of right-hand sides, equal to the number of columns of the matrix B . $NRHS \geq 0$.
xA	On entry, the upper or lower triangle of the matrix A . The dimension of <code>xA</code> is $(N \times N + N) / 2$. On exit, the Cholesky factorization of the matrix A , as computed by <code>xPPTRF</code> .
xB	On entry, the $N \times NRHS$ right-hand side matrix B . On exit, the $N \times NRHS$ solution matrix X .
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.

INFO	On exit:	
	INFO = 0	Subroutine completed normally.
	INFO < 0	The <i>i</i> th argument, where $i = \text{INFO} $, had an illegal value.
	INFO > 0	The leading minor of order <i>i</i> of A, where $i = \text{INFO}$, is not positive definite. The factorization has not been completed, and the solution could not be computed.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
C   INTEGER          LDA, LDB, N, NRHS
C   PARAMETER        (N = 4)
C   PARAMETER        (LDA = (N * (N + 1)) / 2)
C   PARAMETER        (LDB = N)
C   PARAMETER        (NRHS = 1)

C
C   DOUBLE PRECISION A(LDA), B(LDB,NRHS)
C   INTEGER          INFO

C
C   EXTERNAL          DPPSV

C
C   Initialize the array A to store in symmetric form the
C   4x4 symmetric positive definite matrix A shown below.
C   Initialize the array B to store the right hand side
C   vector b shown below.
C
C
C   2   -1   0   0           b = 6
C   A = -1   2  -1   0       b = 12
C       0  -1   2  -1       12
C       0   0  -1   2        6
C
C   DATA A / 2.0D0, -1.0D0, 2.0D0, 0.0D0, -1.0D0, 2.0D0,
$           0.0D0, 0.0D0, -1.0D0, 2.0D0 /
C   DATA B / 6.0D0, 1.2D1, 1.2D1, 6.0D0 /

C
C   Print the initial values of the arrays.
C
C
C   PRINT 1000
C   PRINT 1010, A(1), A(2), A(4), A(7)
C   PRINT 1010, A(2), A(3), A(5), A(8)
C   PRINT 1010, A(4), A(5), A(6), A(9)
C   PRINT 1010, A(7), A(8), A(9), A(10)
C   PRINT 1020
C   PRINT 1030, B

C
C   Solve Ax=b and print the solution.

```

```

C      CALL DPPSV ('UPPER TRIANGLE OF A STORED', N, NRHS, A,
$      B, LDB, INFO)
      IF (INFO .NE. 0) THEN
        PRINT 1040, INFO
        STOP 1
      END IF
      PRINT 1050
      PRINT 1030, B
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (4(3X, F6.3))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (1X, F6.3)
1040 FORMAT (1X, 'Error solving Ax=b, INFO = ', I5)
1050 FORMAT (/1X, 'x:')
C
      END

```

Sample Output

```

A:
  2.000  -1.000   0.000   0.000
 -1.000   2.000  -1.000   0.000
  0.000  -1.000   2.000  -1.000
  0.000   0.000  -1.000   2.000

b:
  6.000
 12.000
 12.000
  6.000

x:
 18.000
 30.000
 30.000
 18.000

```

Solution to a Linear System in a Symmetric Positive Definite Matrix in Packed Storage (Expert Driver)

The subroutines described in this section solve a linear system $AX = B$ for a real symmetric (or Hermitian) positive definite matrix A in packed storage and general matrices B and X . These subroutines also compute forward error bounds and backward error estimates for the solution, estimate the condition number of the matrix A and, optionally, equilibrate the system before computing a solution. Note that the simple driver `xPPSV` is also available.

Calling Sequence

```
CALL DPPSVX (FACT, UPLO, N, NRHS, DA, DAF, EQUED, DSCALE, DB, LDB,
             DX, LDX, DRCOND, DFERR, DBERR, DWORK, IWORK2, INFO)
CALL SPSPVX (FACT, UPLO, N, NRHS, SA, SAF, EQUED, SSCALE, SB, LDB,
             SX, LDX, SRCOND, SFERR, SBERR, SWORK, IWORK2, INFO)
CALL ZPPSVX (FACT, UPLO, N, NRHS, ZA, ZAF, EQUED, DSCALE, ZB, LDB,
             ZX, LDX, DRCOND, DFERR, DBERR, ZWORK, DWORK2, INFO)
CALL CPPSVX (FACT, UPLO, N, NRHS, CA, CAF, EQUED, SSCALE, CB, LDB,
             CX, LDX, SRCOND, SFERR, SBERR, CWORK, SWORK2, INFO)

void dppsvx (char fact, char uplo, int n, int nrhs, double *da,
             double *daf, char *equed, double *dscale, double *db,
             int ldb, double *dx, int ldx, double *drcond, double
             *dferr, double *dberr, int *info)
void sppsvx (char fact, char uplo, int n, int nrhs, float *sa,
             float *saf, char *equed, float *sscale, float *sb, int
             ldb, float *sx, int ldx, float *srcond, float *sferr,
             float *sberr, int *info)
void zppsvx (char fact, char uplo, int n, int nrhs, doublecomplex
             *za, doublecomplex *zaf, char *equed, double *dscale,
             doublecomplex *zb, int ldb, doublecomplex *zx, int ldx,
             double *drcond, double *dferr, double *dberr, int
             *info)
void cppsvx (char fact, char uplo, int n, int nrhs, complex *ca,
             complex *caf, char *equed, float *sscale, complex *cb,
             int ldb, complex *cx, int ldx, float *srcond, float
             *sferr, float *sberr, int *info)
```

Arguments

FACT

Indicates whether or not the factored form of the matrix A is supplied on entry, and if not, whether the matrix A will be equilibrated before it is factored. FACT interacts with EQUED, A, and AF as shown in the table below:

FACT	EQUED entry	EQUED exit	A entry	A exit	AF entry	AF exit
'F', 'f'	'N', 'n'	Unchanged	Matrix A	Unchanged	Cholesky-factored A	Unchanged
'F', 'f'	'Y', 'y'	Unchanged	Equilibrated A	Unchanged	Cholesky-factored equilibrated A	Unchanged
'E', 'e'	Not used	'N'	Matrix A	Unchanged	Not used	Cholesky-factored A
'E', 'e'	Not used	'Y'	Matrix A	Equilibrated A	Not used	Cholesky-factored equilibrated A
'N', 'n'	Not used	'N'	Matrix A	Unchanged	Not used	Cholesky-factored A

The above table uses the following abbreviations:

Cholesky-factored = the UTU or LLT factored form of A as computed by xPPTRF

Equilibrated A = $\text{diag}(\text{SCALE}) \times A \times \text{diag}(\text{SCALE})$

UPLO

Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal.

'U' or 'u' xA contains the upper triangle.

'L' or 'l' xA contains the lower triangle.

N

Order of the matrix A. $N \geq 0$.

NRHS

Number of right-hand sides, equal to the number of columns of the matrices B and X. $\text{NRHS} \geq 0$.

xA

The contents of A vary depending on the values of FACT and EQUED. See the table for the argument FACT above.

The dimension of xA is $(N \times N + N) / 2$.

xAF

The contents of AF vary depending on the values of FACT and EQUED. See the table for the argument FACT above. The dimension of xAF is $(N \times N + N) / 2$.

EQUED

EQUED interacts with FACT as shown in the table above for the argument FACT.

	Note that one must be careful about specifying lowercase letters for the entry value of EQUED when FACT = 'F' or 'f' because ordinarily LAPACK will always return a capital letter. If the caller specifies 'n', or 'y' on entry then the lowercase letter will still be in EQUED on exit. The caller may later draw incorrect conclusions if the code contains a series of IF statements to determine whether EQUED = 'N' or 'Y'.
xSCALE	On entry, if FACT = 'F' or 'f' and EQUED = 'Y' or 'y' then SCALE contains the N scale factors that were used to equilibrate A. Each element of SCALE must be positive. If FACT or EQUED have other values, the entry values of SCALE are not used. On exit, if FACT = 'E' or 'e' and EQUED = 'Y' then SCALE contains the N scale factors that were used to equilibrate A. If FACT or EQUED have other values, SCALE is unchanged.
xB	On entry, the N×NRHS right-hand side matrix B. On exit, if EQUED = 'N', then B is not modified; if EQUED = 'Y', then B is overwritten by $\text{diag}(\text{SCALE}) \times B$.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $\text{LDB} \geq \max(1, N)$.
xX	On exit, the N×NRHS solution matrix for the original problem. Note that if the exit value of EQUED = 'Y' then A and B are modified on exit and the solution to the equilibrated system is $\text{diag}(\text{SCALE})^{-1} \times X$.
LDX	Leading dimension of the array X as specified in a dimension or type statement. $\text{LDX} \geq \max(1, N)$.
xRCOND	On exit, an estimate of the reciprocal condition number of the matrix A after equilibration, if any, where the reciprocal condition number is defined to be $1 / (\ A\ \times \ A^{-1}\)$. The reciprocal of the condition number is estimated instead of the condition number itself to avoid overflow or division by zero. If RCOND is less than machine precision (in particular, if RCOND = 0) then A is singular to working precision. In this case, INFO > 0 is returned and the solution and error bounds are not computed.
xFERR	On exit, the estimated forward error bound for each solution vector $X(*, j)$ for $1 \leq j \leq \text{NRHS}$. If X' is the true solution corresponding to $X(*, j)$ then FERR(j) is an upper bound on the magnitude of the largest element in $X(*, j) - X'$ divided by the magnitude of the largest element in $X(*, j)$.
xBERR	On exit, BERR(j) is the smallest relative change in any element of A or $B(*, j)$ that makes $X(*, j)$ an exact solution to $AX(*, j) = B(*, j)$ for $1 \leq j \leq \text{NRHS}$.
xWORK	Scratch array with a dimension of $3 \times N$ for real subroutines or $2 \times N$ for complex subroutines.
xWORK2	Scratch array with a dimension of N.

INFO	On exit:	
	INFO = 0	Subroutine completed normally.
	INFO < 0	The <i>i</i> th argument, where <i>i</i> = INFO , had an illegal value.
	$1 \leq \text{INFO} \leq N$	The leading minor of order <i>i</i> of A, where <i>i</i> = INFO, is not positive definite. The factorization has not been completed, and the solution and error bounds could not be computed.
	INFO = N+1	RCOND is less than machine precision. The factorization has been completed, but the matrix is singular to working precision, and the solution and error bounds could not be computed.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER          LDA, LDAF, LDB, LDIWRK, LDWORK, LDX
  INTEGER          N, NRHS
  PARAMETER        (N = 4)
  PARAMETER        (LDA = (N * (N + 1)) / 2)
  PARAMETER        (LDAF = LDA)
  PARAMETER        (LDB = N)
  PARAMETER        (LDIWRK = N)
  PARAMETER        (LDX = N)
  PARAMETER        (LDWORK = 3 * N)
  PARAMETER        (NRHS = 1)

C
  DOUBLE PRECISION A(LDA), AF(LDAF), B(LDB,NRHS), BERR(NRHS)
  DOUBLE PRECISION FERR(NRHS), RCOND, SCALE(N), WORK(LDWORK)
  DOUBLE PRECISION X(LDX,NRHS)
  INTEGER          I, INFO, IWORK(LDIWRK)
  CHARACTER*1      EQUED

C
  EXTERNAL          DPPSVX

C
  C Initialize the array A to store in symmetric form the
  C 4x4 symmetric positive definite matrix A shown below.
  C Initialize the array B to store the right hand side
  C vector b shown below.
  C
  C      2   -1   0   0           6
  C  A = -1   2   -1   0           b = 12
  C      0   -1   2   -1           12
  C      0   0   -1   2           6
  C
  DATA A /  2.0D0, -1.0D0, 2.0D0, 0.0D0, -1.0D0, 2.0D0,

```

```

$          0.0D0, 0.0D0, -1.0D0, 2.0D0 /
DATA B / 6.0D0, 1.2D1, 1.2D1, 6.0D0 /
C
C      Print the initial values of the arrays.
C
      PRINT 1000
      PRINT 1010, A(1), A(2), A(4), A(7)
      PRINT 1010, A(2), A(3), A(5), A(8)
      PRINT 1010, A(4), A(5), A(6), A(9)
      PRINT 1010, A(7), A(8), A(9), A(10)
      PRINT 1020
      PRINT 1030, B
C
C      Solve Ax=b and print the solution.
C
      CALL DPPSVX ('EQUILIBRATE A IF NECESSARY',
$              'UPPER TRIANGLE OF A STORED', N, NRHS, A, AF,
$              EQUED, SCALE, B, LDB, X, LDX, RCOND, FERR,
$              BERR, WORK, IWORK, INFO)
      IF (INFO .NE. 0) THEN
          PRINT 1040, INFO
          IF (INFO .EQ. (N + 1)) THEN
              PRINT 1050
              STOP 1
          END IF
          STOP 2
      END IF
      PRINT 1060
      PRINT 1030, X
      PRINT 1070, 1.0D0 / RCOND
      IF (EQUED .EQ. 'N') PRINT 1080
      IF (EQUED .EQ. 'R') PRINT 1090
      IF (EQUED .EQ. 'C') PRINT 1100
      IF (EQUED .EQ. 'B') PRINT 1110
      PRINT 1120, (I, BERR(I), I = 1, NRHS)
      PRINT 1130, (I, FERR(I), I = 1, NRHS)
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (4(3X, F6.3))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (1X, F6.2)
1040 FORMAT (1X, 'Error solving Ax=b, INFO = ', I5)
1050 FORMAT (1X, 'Matrix is singular to working precision.')
1060 FORMAT (/1X, 'x:')
1070 FORMAT (/1X, 'Estimated condition number of A: ', F7.2)
1080 FORMAT (1X, 'No equilibration was required for A.')
1090 FORMAT (1X, 'Row equilibration was done on A.')
1100 FORMAT (1X, 'Column equilibration was done on A.')
1110 FORMAT (1X, 'Row and column equilibration was done on A.')
1120 FORMAT (/1X, 'Backward error for system #', I1, ': ', E12.6)
1130 FORMAT (1X, 'Forward error for system #', I1, ': ', E12.6)
C
      END

```

Sample Output

```
A:
  2.000  -1.000   0.000   0.000
 -1.000   2.000  -1.000   0.000
  0.000  -1.000   2.000  -1.000
  0.000   0.000  -1.000   2.000

b:
  6.00
 12.00
 12.00
  6.00

x:
 18.00
 30.00
 30.00
 18.00

Estimated condition number of A:  12.00
No equilibration was required for A.

Backward error for system #1: 0.000000E+00
Forward error for system #1:  0.577316E-14
```


Cholesky Factorization of a Symmetric Positive Definite Matrix in Packed Storage

The subroutines described in this section compute a Cholesky factorization of a real symmetric (or Hermitian) positive definite matrix A in packed storage. It is typical to follow a call to xPPTRF with a call to xPPSVX or xPPTRS to solve a linear system $AX = B$, to xPPCON to estimate the condition number of A , or to xPPTRI to compute A^{-1} .

Calling Sequence

```
CALL DPPTRF (UPLO, N, DA, INFO)
CALL SPPTRF (UPLO, N, SA, INFO)
CALL ZPPTRF (UPLO, N, ZA, INFO)
CALL CPPTRF (UPLO, N, CA, INFO)
```

```
void dpptrf (char uplo, int n, double *da, int *info)
void spptrf (char uplo, int n, float *sa, int *info)
void zpptrf (char uplo, int n, doublecomplex *za, int *info)
void cpptrf (char uplo, int n, complex *ca, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
xA	On entry, the upper or lower triangle of the matrix A. The dimension of xA is $(N \times N + N) / 2$. On exit, a Cholesky factorization of the matrix A.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The i th argument, where $i = INFO $, had an illegal value. INFO > 0 The leading minor of order i of A, where $i = INFO$, is not positive definite. The factorization could not be completed.


```
      IF (INFO .NE. 0) THEN
        PRINT 1050, ABS(INFO)
        STOP 2
      END IF
      PRINT 1060
      PRINT 1030, B
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F6.3))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (1X, F6.2)
1040 FORMAT (1X, 'Error factoring A, INFO = ', I5)
1050 FORMAT (1X, 'Illegal argument to DPPTRS, argument #', I1)
1060 FORMAT (/1X, 'x:')
C
      END
```

Sample Output

```
A in full form:
 2.000  -1.000   0.000   0.000
-1.000   2.000  -1.000   0.000
 0.000  -1.000   2.000  -1.000
 0.000   0.000  -1.000   2.000

b:
 6.00
12.00
12.00
 6.00

x:
18.00
30.00
30.00
18.00
```

Inverse of a Cholesky-Factored Symmetric Positive Definite Matrix in Packed Storage

The subroutines described in this section compute the inverse of a real symmetric (or Hermitian) positive definite matrix A in packed storage, which has been Cholesky-factored by xPPTRF.

Calling Sequence

```
CALL DPPTRI (UPLO, N, DA, INFO)
CALL SPPTRI (UPLO, N, SA, INFO)
CALL ZPPTRI (UPLO, N, ZA, INFO)
CALL CPPTRI (UPLO, N, CA, INFO)

void dpptri (char uplo, int n, double *da, int *info)
void spptri (char uplo, int n, float *sa, int *info)
void zpptri (char uplo, int n, doublecomplex *za, int *info)
void cpptri (char uplo, int n, complex *ca, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
xA	On entry, the Cholesky factorization of the matrix A as computed by xPPTRF. The dimension of xA is $(N \times N + N) / 2$. On exit, the symmetric (or Hermitian) inverse of A.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where $i = INFO $, had an illegal value. INFO > 0 Element (i,i) of the factor U or L, where $i = INFO$, is zero. The matrix is therefore singular and its inverse could not be computed.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
C      INTEGER          LDA, N
C      PARAMETER        (N = 4)
C      PARAMETER        (LDA = (N * (N + 1)) / 2)
C
C      DOUBLE PRECISION A(LDA)
C      INTEGER          INFO
C
C      EXTERNAL          DPPTRF, DPPTRI
C      INTRINSIC         ABS
C
C      Initialize the array A to store in symmetric form the
C      4x4 symmetric positive definite coefficient matrix A
C      shown below. Initialize the array B to store the right
C      hand side vector b shown below.
C
C      2  1  0  0
C      A = 1  2  1  0
C      0  1  2  1
C      0  0  1  1
C
C      DATA A / 2.0D0, 1.0D0, 2.0D0, 0.0D0, 1.0D0, 2.0D0,
C      $          0.0D0, 0.0D0, 1.0D0, 1.0D0 /
C
C      Print the initial values of the arrays.
C
C      PRINT 1000
C      PRINT 1010, A(1), A(2), A(4), A(7)
C      PRINT 1010, A(2), A(3), A(5), A(8)
C      PRINT 1010, A(4), A(5), A(6), A(9)
C      PRINT 1010, A(7), A(8), A(9), A(10)
C
C      Cholesky factor A.
C
C      CALL DPPTRF ('UPPER TRIANGLE OF A STORED', N, A, INFO)
C      IF (INFO .NE. 0) THEN
C        PRINT 1020, INFO
C        STOP 1
C      END IF
C
C      Use the factored form of A to compute the inverse of A and
C      print the inverse thus computed.
C
C      CALL DPPTRI ('UPPER TRIANGLE OF A STORED', N, A, INFO)
C      IF (INFO .NE. 0) THEN
C        PRINT 1030, ABS(INFO)
C        STOP 2
C      END IF
C      PRINT 1040
C      PRINT 1010, A(1), A(2), A(4), A(7)

```

```

      PRINT 1010, A(2), A(3), A(5), A(8)
      PRINT 1010, A(4), A(5), A(6), A(9)
      PRINT 1010, A(7), A(8), A(9), A(10)
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (4(3X, F6.3))
1020 FORMAT (1X, 'Error factoring A, INFO = ', I5)
1030 FORMAT (1X, 'Error computing inverse of A, INFO = ', I5)
1040 FORMAT (/1X, 'A**(-1):')
C
      END

```

Sample Output

```

A:
  2.000    1.000    0.000    0.000
  1.000    2.000    1.000    0.000
  0.000    1.000    2.000    1.000
  0.000    0.000    1.000    1.000

A**(-1):
  1.000   -1.000    1.000   -1.000
 -1.000    2.000   -2.000    2.000
  1.000   -2.000    3.000   -3.000
 -1.000    2.000   -3.000    4.000

```

Solution to a Linear System in a Cholesky-Factored Symmetric Positive Definite Matrix in Packed Storage

The subroutines described in this section solve a linear system $AX = B$ for a real symmetric (or Hermitian) positive definite matrix A in packed storage, which has been Cholesky-factored by xPPTRF, and general matrices B and X .

Calling Sequence

```
CALL DPPTRS (UPLO, N, NRHS, DA, DB, LDB, INFO)
CALL SPPTS (UPLO, N, NRHS, SA, SB, LDB, INFO)
CALL ZPPTRS (UPLO, N, NRHS, ZA, ZB, LDB, INFO)
CALL CPPTRS (UPLO, N, NRHS, CA, CB, LDB, INFO)

void dpptrs (char uplo, int n, int nrhs, double *da, double *db,
            int ldb, int *info)
void spptrs (char uplo, int n, int nrhs, float *sa, float *sb, int
            ldb, int *info)
void zpptrs (char uplo, int n, int nrhs, doublecomplex *za,
            doublecomplex *zb, int ldb, int *info)
void cpptrs (char uplo, int n, int nrhs, complex *ca, complex *cb,
            int ldb, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
NRHS	Number of right-hand sides, equal to the number of columns of the matrix B. $NRHS \geq 0$.
xA	Cholesky factorization of the matrix A as computed by xPPTRF. The dimension of xA is $(N \times N + N) / 2$.
xB	On entry, the $N \times NRHS$ right-hand side matrix B. On exit, the $N \times NRHS$ solution matrix X.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
INFO	On exit: INFO = 0 Subroutine completed normally.

INFO < 0 The ith argument, where i = |INFO|, had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C   INTEGER          LDA, LDB, N, NRHS
C   PARAMETER        (N = 4)
C   PARAMETER        (NRHS = 1)
C   PARAMETER        (LDA = (N * (N + 1)) / 2)
C   PARAMETER        (LDB = N)
C
C   DOUBLE PRECISION A(LDA), B(LDB,NRHS)
C   INTEGER          INFO
C
C   EXTERNAL          DPPTRF, DPPTRS
C   INTRINSIC         ABS
C
C   Initialize the array A to store in symmetric form the
C   4x4 symmetric positive definite coefficient matrix A
C   shown below. Initialize the array B to store the right
C   hand side vector b shown below.
C
C
C   A =  2  -1  0  0      b =  6
C   -1  2  -1  0      12
C   0  -1  2  -1      12
C   0  0  -1  2       6
C
C   DATA A /  2.0D0, -1.0D0, 2.0D0, 0.0D0, -1.0D0, 2.0D0,
$          0.0D0, 0.0D0, -1.0D0, 2.0D0 /
C   DATA B / 6.0D0, 1.2D1, 1.2D1, 6.0D0 /
C
C   Print the initial values of the arrays.
C
C   PRINT 1000
C   PRINT 1010, A(1), A(2), A(4), A(7)
C   PRINT 1010, A(2), A(3), A(5), A(8)
C   PRINT 1010, A(4), A(5), A(6), A(9)
C   PRINT 1010, A(7), A(8), A(9), A(10)
C   PRINT 1020
C   PRINT 1030, B
C
C   Cholesky factor A.
C
C   CALL DPPTRF ('UPPER TRIANGLE OF A STORED', N, A, INFO)
C   IF (INFO.NE. 0) THEN
C       PRINT 1040, INFO
C       STOP 1
C   END IF
C

```



```
C      Use the factored form of A to solve Ax=b then print
C      the result.
C
C      CALL DPPTRS ('UPPER TRIANGLE OF A STORED', N, NRHS, A,
$      B, LDB, INFO)
C      IF (INFO .NE. 0) THEN
C          PRINT 1050, ABS(INFO)
C          STOP 2
C      END IF
C      PRINT 1060
C      PRINT 1030, B
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (4(3X, F6.3))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (1X, F6.2)
1040 FORMAT (1X, 'Error factoring A, INFO = ', I5)
1050 FORMAT (1X, 'Illegal argument to DPPTRS, argument #', I1)
1060 FORMAT (/1X, 'x:')
C
C      END
```

Sample Output

```
A:
 2.000  -1.000   0.000   0.000
-1.000   2.000  -1.000   0.000
 0.000  -1.000   2.000  -1.000
 0.000   0.000  -1.000   2.000

b:
 6.00
12.00
12.00
 6.00

x:
18.00
30.00
30.00
18.00
```

Reciprocal Condition Number of a UDU- or LDL-Factored Symmetric Positive Definite Tridiagonal Matrix

The subroutines described in this section estimate the reciprocal condition number of a real symmetric (or Hermitian) positive definite tridiagonal matrix A, which has been UDU-factored or LDL-factored by xPTTRF.

Calling Sequence

```
CALL DPTCON (N, DDIAG, DOFFD, DANORM, DRCOND, DWORK, INFO)
CALL SPTCON (N, SDIAG, SOFFD, SANORM, SRCOND, SWORK, INFO)
CALL ZPTCON (N, DDIAG, ZOFFD, DANORM, DRCOND, DWORK, INFO)
CALL CPTCON (N, SDIAG, COFFD, SANORM, SRCOND, SWORK, INFO)

void dptcon (int n, double *ddiag, double *doffd, double danorm,
             double *drcond, int *info)
void sptcon (int n, float *sdiag, float *soffd, float sanorm, float
             *srcond, int *info)
void zptcon (int n, double *ddiag, doublecomplex *zoffd, double
             danorm, double *drcond, int *info)
void cptcon (int n, float *sdiag, complex *zoffd, float sanorm,
             float *srcond, int *info)
```

Arguments

N	Order of the matrix A. $N \geq 0$.
xDIAG	The N diagonal elements of the diagonal matrix D from the factorization of A, as computed by xPTTRF.
xOFFD	The N-1 off-diagonal elements of the unit bidiagonal factor U or L from the UDU or LDL factorization of A, as computed by xPTTRF.
xANORM	The 1-norm or ∞ -norm of matrix A.
xRCOND	On exit, the estimated reciprocal of the condition number of the matrix A where the reciprocal condition number of A is defined to be $1 / (ANORM \times \ A^{-1}\)$. The reciprocal of the condition number is estimated instead of the condition number itself to avoid overflow or division by zero. If RCOND is less than machine precision (in particular, if RCOND = 0) then A is singular to working precision.
xWORK	Scratch array with a dimension of N.
INFO	On exit: INFO = 0 Subroutine completed normally.

INFO < 0

The *i*th argument, where *i* = |INFO|, had an illegal value.*Sample Program*

```

PROGRAM TEST
IMPLICIT NONE

C
C   INTEGER          LDB, LDWORK, N, NRHS
C   PARAMETER        (N = 4)
C   PARAMETER        (LDB = N)
C   PARAMETER        (LDWORK = N)
C   PARAMETER        (NRHS = 1)
C
C   DOUBLE PRECISION ANORM, DIAG(N), DLOWER(N-1), RCOND
C   DOUBLE PRECISION WORK(LDWORK)
C   INTEGER          ICOL, INFO, IROW
C
C   EXTERNAL          DPTCON, DPTTRF
C   INTRINSIC         ABS
C
C   Initialize the arrays DIAG and DLOWER to store the diagonal
C   and first off diagonal of the matrix A shown below.
C
C       2  -1
C   -1  2  -1
C   A =  -1  2  -1
C         1  2  -1
C
C   DATA DIAG / 2.0D0, 2.0D0, 2.0D0, 2.0D0 /
C   DATA DLOWER / -1.0D0, -1.0D0, -1.0D0 /
C
C   Print the initial value of the arrays.
C
C   PRINT 1000
C   DO 100, IROW = 1, N
C       PRINT 1010, (0.0D0, ICOL = 1, IROW - 2),
C   $       (DLOWER(ICOL + 1), ICOL = ABS(IROW - 2), IROW - 2),
C   $       DIAG(IROW),
C   $       (DLOWER(IROW), ICOL = 1, MIN(1, N - IROW)),
C   $       (0.0D0, ICOL = IROW + 2, N)
100 CONTINUE
C
C   LDL factor A.
C
C   CALL DPTTRF (N, DIAG, DLOWER, INFO)
C   IF (INFO .NE. 0) THEN
C       PRINT 1020, INFO
C       STOP 1
C   END IF
C
C   Estimate and print the condition number of A.

```

```

C      ANORM = 4.0D0
      CALL DPTCON (N, DIAG, DLOWER, ANORM, RCOND, WORK, INFO)
      IF (INFO .NE. 0) THEN
        PRINT 1030, ABS (INFO)
        STOP 2
      END IF
      PRINT 1040, 1.0D0 / RCOND
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (5(3X, F5.2))
1020 FORMAT (1X, 'Error factoring A, INFO = ', I5)
1030 FORMAT (1X, 'Illegal argument to DPTCON, argument #', I2)
1040 FORMAT (/1X, 'Estimated condition number of A: ', F6.2)
C
      END

```

Sample Output

```

A:
  2.00   -1.00    0.00    0.00
 -1.00    2.00   -1.00    0.00
  0.00   -1.00    2.00   -1.00
  0.00    0.00   -1.00    2.00

Estimated condition number of A:  12.00

```

Refined Solution to a Linear System in a UDU- or LDL-Factored Symmetric Positive Definite Tridiagonal Matrix

The subroutines described in this section refine the solution to a linear system $AX = B$ for a real symmetric (or Hermitian) positive definite tridiagonal matrix A , which has been UDU-factored or LDL-factored by `xPTTRF`, and general matrices B and X . These subroutines also compute forward error bounds and backward error estimates for the refined solution.

Calling Sequence

```
CALL DPTRFS (N, NRHS, DDIAG, DOFFD, DDIAGF, DOFFDF, DB, LDB, DX,
             LDX, DFERR, DBERR, DWORK, INFO)
CALL SPTRFS (N, NRHS, SDIAG, SOFFD, SDIAGF, SOFFDF, SB, LDB, SX,
             LDX, SFERR, SBERR, SWORK, INFO)
CALL ZPTRFS (UPLO, N, NRHS, DDIAG, ZOFFD, DDIAGF, ZOFFDF, ZB, LDB,
             ZX, LDX, DFERR, DBERR, ZWORK, DWORK2, INFO)
CALL CPTRFS (UPLO, N, NRHS, SDIAG, COFFD, SDIAGF, COFFDF, CB, LDB,
             CX, LDX, SFERR, SBERR, CWORK, SWORK2, INFO)

void dptrfs (int n, int nrhs, double *ddiag, double *doffd, double
             *ddiagf, double *doffdf, double *db, int ldb, double
             *dx, int ldx, double *dferr, double *dberr, int *info)
void sptrfs (int n, int nrhs, float *sdiag, float *soffd, float
             *sdiagf, float *soffdf, float *sb, int ldb, float *sx,
             int ldx, float *sferr, float *sberr, int *info)
void zptrfs (char uplo, int n, int nrhs, double *ddiag,
             doublecomplex *zoffd, double *ddiagf, doublecomplex
             *zoffdf, doublecomplex *zb, int ldb, doublecomplex
             *zx, int ldx, double *dferr, double *dberr, int *info)
void cptrfs (char uplo, int n, int nrhs, float *sdiag, complex
             *coffd, float *sdiagf, complex *coffdf, complex *cb,
             int ldb, complex *cx, int ldx, float *sferr, float
             *sberr, int *info)
```

Arguments

UPLO (For complex matrices)
Indicates whether OFFDF contains the superdiagonal or subdiagonal of the matrix A and the form of the factorization. The legal values for UPLO are listed below. Any values not listed are illegal.

	'U' or 'u'	xOFFDF contains the superdiagonal and the factorization is UDU.
	'L' or 'l'	xOFFDF contains the subdiagonal and the factorization is LDL.
N	Order of the matrix A. $N \geq 0$.	
NRHS	Number of right-hand sides, equal to the number of columns of the matrices B and X. $NRHS \geq 0$.	
xDIAG	The N diagonal elements of A.	
xOFFD	The N-1 off-diagonal elements of A.	
xDIAGF	The N diagonal elements of the matrix D from the UDU or LDL factorization of A, as computed by xPTTRF.	
xOFFDF	The N-1 off-diagonal elements of the matrix U or L from the UDU or LDL factorization of A, as computed by xPTTRF.	
xB	The N×NRHS right-hand side matrix B.	
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.	
xX	On entry, the N×NRHS solution matrix X. On exit, the refined N×NRHS solution matrix X.	
LDX	Leading dimension of the array X as specified in a dimension or type statement. $LDX \geq \max(1, N)$.	
xFERR	On exit, the estimated forward error bound for each solution vector $X(*, j)$ for $1 \leq j \leq NRHS$. If X' is the true solution corresponding to $X(*, j)$ then $FERR(j)$ is an upper bound on the magnitude of the largest element in $X(*, j) - X'$ divided by the magnitude of the largest element in $X(*, j)$.	
xBERR	On exit, $BERR(j)$ is the smallest relative change in any element of A or $B(*, j)$ that makes $X(*, j)$ an exact solution to $AX(*, j) = B(*, j)$ for $1 \leq j \leq NRHS$.	
xWORK	Scratch array with a dimension of $2 \times N$ for real subroutines or N for complex subroutines.	
xWORK2	Scratch array with a dimension of N for complex subroutines.	
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where $i = INFO $, had an illegal value.	

Sample Program

PROGRAM TEST		
IMPLICIT NONE		
C		
INTEGER	LDB, LDWORK, LDX, N, NRHS	

```

PARAMETER      (N = 4)
PARAMETER      (LDB = N)
PARAMETER      (LDWORK = 2 * N)
PARAMETER      (LDX = N)
PARAMETER      (NRHS = 1)
C
DOUBLE PRECISION B(LDB,NRHS), BERR(NRHS), DIAG(N)
DOUBLE PRECISION DIAGF(N), DLOW(N-1), DLOWF(N-1), EPSLON
DOUBLE PRECISION FERR(NRHS), WORK(LDWORK), X(LDX,NRHS)
INTEGER         ICOL, INFO, IROW
C
EXTERNAL        DCOPY, DPTCON, DPTRFS, DPTTRF, DPTTRS
INTRINSIC       ABS, MAX
C
C      Initialize the arrays DLOW, DIAG, and DUP1 to store
C      the first subdiagonal, the diagonal, and the first
C      superdiagonal of the symmetric coefficient matrix A
C      shown below. Initialize the array B to store the right
C      hand side matrix b shown below.
C
C      A =
C      0    0    30
C      0    2    2    b = 50
C      2    2    0    70
C      0    0    110
C
DATA DLOW / 0.0D0, 2.0D0, 0.0D0 /
DATA DIAG / 0.0D0, 2.0D0, 2.0D0, 0.0D0 /
DATA B / 3.0D1, 5.0D1, 7.0D1, 1.1D2 /
C
C      Add a small value to each of the elements on the diagonal
C      and subtract the same value from the first sub- and
C      super-diagonal of A. After this loop, A will resemble the
C      matrix shown below. Print A after adding epsilon.
C
C      A =
C      -e    -e
C      -e    2+e    2-e
C      2-e    2+e    -e
C      -e    -e
C
EPSLON = ABS (((2.0D0 / 3.0D0) + 4.0D0) - 4.0D0) -
$      (2.0D0 / 3.0D0)
DO 100, IROW = 1, N - 1
    DLOW(IROW) = DLOW(IROW) - EPSLON
    DIAG(IROW) = DIAG(IROW) + EPSLON
100 CONTINUE
DIAG(N) = DIAG(N) + EPSLON
CALL DCOPY (N - 1, DLOW, 1, DLOWF, 1)
CALL DCOPY (N, DIAG, 1, DIAGF, 1)
C
PRINT 1000
DO 110, IROW = 1, N
    PRINT 1010, (0.0D0, ICOL = 1, IROW - 2),
$      (DLOW(ICOL + 1), ICOL = ABS(IROW - 2), IROW - 2),
$      DIAG(IROW),
$      (DLOW(IROW), ICOL = 1, MIN(1, N - IROW)),
$      (0.0D0, ICOL = IROW + 2, N)

```

```

110 CONTINUE
C
C   Slightly perturb each element of B. After this loop, B will
C   resemble the matrix shown below. Print B afterwards.
C
C       30+e
C   B = 50+e
C       70+e
C       110+e
C
      DO 130, ICOL = 1, NRHS
        DO 120, IROW = 1, N
          B(IROW,ICOL) = B(IROW,ICOL) * (1.0D0 - EPSLON)
120    CONTINUE
130  CONTINUE
      CALL DCOPY (N, B, 1, X, 1)
      PRINT 1020
      PRINT 1030, B
C
C   LDL factor A.
C
      CALL DPTTRF (N, DIAGF, DLOWF, INFO)
      IF (INFO .LT. 0) THEN
        PRINT 1040, ABS(INFO)
        STOP 1
      ELSE IF (INFO .GT. 0) THEN
        PRINT 1050, INFO
        STOP 2
      END IF
C
C   Solve Ax=b and print the solution.
C
      CALL DPTTRS (N, NRHS, DIAGF, DLOWF, X, LDX, INFO)
      IF (INFO .NE. 0) THEN
        PRINT 1060, INFO
        STOP 3
      END IF
      PRINT 1070
      PRINT 1080, X
      PRINT 1090
      PRINT 1030,                                DIAG(1) * X(1,1) + DLOW(1) *
$      X(2,1)
      PRINT 1030, DLOW(1) * X(1,1) + DIAG(2) * X(2,1) + DLOW(2) *
$      X(3,1)
      PRINT 1030, DLOW(2) * X(2,1) + DIAG(3) * X(3,1) + DLOW(3) *
$      X(4,1)
      PRINT 1030, DLOW(3) * X(3,1) + DIAG(4) * X(4,1)
C
C   Refine the solution to Ax=b and print the refined solution.
C
      CALL DPTRFS (N, NRHS, DIAG, DLOW, DIAGF, DLOWF, B, LDB,
$      X, LDX, FERR, BERR, WORK, INFO)
      IF (INFO .NE. 0) THEN
        PRINT 1100, ABS(INFO)
        STOP 4
      END IF

```



```

      PRINT 1110
      PRINT 1080, X
      PRINT 1120
      PRINT 1030,
      DIAG(1) * X(1,1) + DLOW(1) *
$    X(2,1)
      PRINT 1030, DLOW(1) * X(1,1) + DIAG(2) * X(2,1) + DLOW(2) *
$    X(3,1)
      PRINT 1030, DLOW(2) * X(2,1) + DIAG(3) * X(3,1) + DLOW(3) *
$    X(4,1)
      PRINT 1030, DLOW(3) * X(3,1) + DIAG(4) * X(4,1)
      PRINT 1130, FERR(1)
      PRINT 1140, BERR(1)
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (4(2X, F18.16))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (1X, F21.17)
1040 FORMAT (1X, 'Illegal argument to DPTRFS, argument #', I2)
1050 FORMAT (1X, 'A is not positive definite, INFO = ', I2)
1060 FORMAT (1X, 'Illegal argument to DPTRFS, argument #', I2)
1070 FORMAT (/1X, 'Initial solution to Ax=b:')
1080 FORMAT (1X, E25.17)
1090 FORMAT (/1X, 'Ax with the initial x:')
1100 FORMAT (1X, 'Illegal argument to DPTRFS, INFO = ', I2)
1110 FORMAT (/1X, 'Refined solution to Ax=b:')
1120 FORMAT (/1X, 'Ax with refined x:')
1130 FORMAT (/1X, 'Forward error: ', E14.8)
1140 FORMAT (1X, 'Backward error: ', E14.8)
C
      END

```

Sample Output

```

A:
0.000000000000000003  -0.000000000000000003  0.000000000000000000  0.000000000000000000
-0.000000000000000003  2.000000000000000004  1.9999999999999996  0.000000000000000000
0.000000000000000000  1.9999999999999996  2.000000000000000004  -0.000000000000000003
0.000000000000000000  0.000000000000000000  -0.000000000000000003  0.000000000000000003

b:
29.999999999999998934
49.999999999999998579
69.999999999999997158
109.999999999999995737

Initial solution to Ax=b:
-0.10007999171934384E+17
-0.10007999171934427E+18
0.10007999171934432E+18
0.43034396439318054E+18

```

```
Ax with the initial x:  
29.99999999999999289  
32.00000000000000000  
16.666666666666674246  
109.99999999999994316
```

```
Refined solution to Ax=b:  
-0.57379195252423992E+17  
-0.14745118779983389E+18  
0.14745118779983392E+18  
0.47771516047367027E+18
```

```
Ax with refined x:  
29.99999999999999645  
-64.00000000000000000  
32.888888888888891415  
110.00000000000000000
```

```
Forward error: 0.47796400E+01  
Backward error: 0.17862806E-15
```

Solution to a Linear System in a Symmetric Positive Definite Tridiagonal Matrix (Simple Driver)

The subroutines described in this section solve a linear system $AX = B$ for a real symmetric (or Hermitian) positive definite tridiagonal matrix A and general matrices B and X . Note that the expert driver `xPTSVX` is also available.

Calling Sequence

```
CALL DPTSV (N, NRHS, DDIAG, DSUB, DB, LDB, INFO)
CALL SPTSV (N, NRHS, SDIAG, SSUB, SB, LDB, INFO)
CALL ZPTSV (N, NRHS, DDIAG, ZSUB, ZB, LDB, INFO)
CALL CPTSV (N, NRHS, SDIAG, CSUB, CB, LDB, INFO)

void dptsv (int n, int nrhs, double *ddiag, double *dsub, double
            *db, int ldb, int *info)
void sptsv (int n, int nrhs, float *sdiag, float *ssub, float *sb,
            int ldb, int *info)
void zptsv (int n, int nrhs, double *ddiag, doublecomplex *dsub,
            doublecomplex *zb, int ldb, int *info)
void cptsv (int n, int nrhs, float *sdiag, complex *ssub, complex
            *cb, int ldb, int *info)
```

Arguments

<code>N</code>	Order of the matrix A . $N \geq 0$.
<code>NRHS</code>	Number of right-hand sides, equal to the number of columns of the matrix B . $NRHS \geq 0$.
<code>xDIAG</code>	On entry, the N diagonal elements of the matrix A . On exit, the N diagonal elements of the diagonal matrix D from the LDL factorization of A , as computed by <code>xPTTRF</code> .
<code>xSUB</code>	On entry, the $N-1$ subdiagonal elements of the matrix A . On exit, the $N-1$ subdiagonal elements of the unit bidiagonal factor L from the LDL factorization of A , as computed by <code>xPTTRF</code> .
<code>xB</code>	On entry, the $N \times NRHS$ right-hand side matrix B . On exit, the $N \times NRHS$ solution matrix X .
<code>LDB</code>	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
<code>INFO</code>	On exit: <code>INFO = 0</code> Subroutine completed normally.

INFO < 0 The *i*th argument, where *i* = |INFO|, had an illegal value.

INFO > 0 The leading minor of order *i* of A, where *i* = INFO, is not positive definite. The factorization has not been completed, unless *i* = N, and the solution could not be computed.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C   INTEGER          LDB, N, NRHS
C   PARAMETER        (N = 4)
C   PARAMETER        (NRHS = 1)
C   PARAMETER        (LDB = N)
C
C   DOUBLE PRECISION B(LDB,NRHS), DIAG(N), DLOWER(N-1)
C   INTEGER          ICOL, INFO, IROW
C
C   EXTERNAL          DPTSV
C
C   Initialize the arrays DIAG and DLOWER to store in symmetric
C   tridiagonal form the 4x4 symmetric positive definite
C   coefficient matrix A shown below. Initialize the array B
C   to store the right hand side vector b shown below.
C
C   A =
C   2  -1  0  0      b = 6
C  -1  2  -1  0      12
C   0  -1  2  -1      12
C   0   0  -1  2       6
C
C   DATA DIAG / 2.0D0, 2.0D0, 2.0D0, 2.0D0 /
C   DATA DLOWER / -1.0D0, -1.0D0, -1.0D0 /
C   DATA B / 6.0D0, 1.2D1, 1.2D1, 6.0D0 /
C
C   Print the initial values of the arrays.
C
C   PRINT 1000
C   DO 100, IROW = 1, N
C       PRINT 1010, (0.0D0, ICOL = 1, IROW - 2),
C   $           (DLOWER(ICOL + 1), ICOL = ABS(IROW - 2), IROW - 2),
C   $           DIAG(IROW),
C   $           (DLOWER(IROW), ICOL = 1, MIN(1, N - IROW)),
C   $           (0.0D0, ICOL = IROW + 2, N)
100 CONTINUE
C   PRINT 1020
C   PRINT 1030, B
C
C   Solve the system and print the results.
C

```

```
      CALL DPTSV (N, NRHS, DIAG, DLOWER, B, LDB, INFO)
      IF (INFO .NE. 0) THEN
        PRINT 1040, INFO
        STOP 1
      END IF
      PRINT 1050
      PRINT 1030, B
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (4(3X, F6.3))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (1X, F6.2)
1040 FORMAT (1X, 'Error solving Ax=b, INFO = ', I5)
1050 FORMAT (/1X, 'x:')
C
      END
```

Sample Output

```
A:
  2.000  -1.000   0.000   0.000
 -1.000   2.000  -1.000   0.000
  0.000  -1.000   2.000  -1.000
  0.000   0.000  -1.000   2.000

b:
  6.00
 12.00
 12.00
  6.00

x:
 18.00
 30.00
 30.00
 18.00
```

Solution to a Linear System in a Symmetric Positive Definite Tridiagonal Matrix (Expert Driver)

The subroutines described in this section solve a linear system $AX = B$ for a real symmetric (or Hermitian) positive definite tridiagonal matrix A and general matrices B and X . These subroutines also compute forward error bounds and backward error estimates for the solution, and estimate the condition number of the matrix A . Note that the simple driver `xPTSV` is also available.

Calling Sequence

```
CALL DPTSVX (FACT, N, NRHS, DDIAG, DSUB, DDIAGF, DSUBF, DB, LDB,
             DX, LDX, DRCOND, DFERR, DBERR, DWORK, INFO)
CALL SPTSVX (FACT, N, NRHS, SDIAG, SSUB, SDIAGF, SSUBF, SB, LDB,
             SX, LDX, SRCOND, SFERR, SBERR, SWORK, INFO)
CALL ZPTSVX (FACT, N, NRHS, DDIAG, ZSUB, DDIAGF, ZSUBF, ZB, LDB,
             ZX, LDX, DRCOND, DFERR, DBERR, ZWORK, DWORK2, INFO)
CALL CPTSVX (FACT, N, NRHS, SDIAG, CSUB, SDIAGF, CSUBF, CB, LDB,
             CX, LDX, SRCOND, SFERR, SBERR, CWORK, SWORK2, INFO)

void dptsvx (char fact, int n, int nrhs, double *ddiag, double
             *dsub, double *ddiagf, double *dsubf, double *db, int
             ldb, double *dx, int ldx, double *drcond, double
             *dferr, double *dberr, int *info)
void sptsvx (char fact, int n, int nrhs, float *sdiag, float *ssub,
             float *sdiagf, float *ssubf, float *sb, int ldb, float
             *sx, int ldx, float *srcond, float *sferr, float
             *sberr, int *info)
void zptsvx (char fact, int n, int nrhs, double *ddiag,
             doublecomplex *zsub, double *ddiagf, doublecomplex
             *zsubf, doublecomplex *zb, int ldb, doublecomplex *zx,
             int ldx, double *drcond, double *dferr, double *dberr,
             int *info)
void cptsvx (char fact, int n, int nrhs, float *sdiag, complex
             *csub, float *sdiagf, complex *csubf, complex *cb, int
             ldb, complex *cx, int ldx, float *srcond, float *sferr,
             float *sberr, int *info)
```

Arguments

FACT	Indicates whether or not the factored form of A has been supplied on entry. 'F' or 'f' On entry, DIAGF and SUBF contain the factored form of A. DIAG, SUB, DIAGF, and SUBF will not be modified. 'N' or 'n' The matrix A will be copied to DIAGF and SUBF and factored.
N	Order of the matrix A. $N \geq 0$.
NRHS	Number of right-hand sides, equal to the number of columns of the matrices B and X. $NRHS \geq 0$.
xDIAG	The N diagonal elements of the matrix A.
xSUB	The N-1 subdiagonal elements of the matrix A.
xDIAGF	On entry, if FACT = 'F' or 'f', then DIAGF contains the N diagonal elements of the diagonal matrix D from the LDL factorization of A as computed by xPTTRF. On exit, if FACT = 'N' or 'n', then DIAGF contains the N diagonal elements of the diagonal matrix D from the LDL factorization of A as computed by xPTTRF.
xSUBF	On entry, if FACT = 'F' or 'f', then SUBF contains the N-1 subdiagonal elements of the unit bidiagonal factor L from the LDL factorization of A, as computed by xPTTRF. On exit, if FACT = 'N' or 'n', then SUBF contains the N-1 subdiagonal elements of the unit bidiagonal factor L from the LDL factorization of A, as computed by xPTTRF.
xB	The N×NRHS right-hand side matrix B.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
xX	On exit, the N×NRHS solution matrix X.
LDX	Leading dimension of the array X as specified in a dimension or type statement. $LDX \geq \max(1, N)$.
xRCOND	On exit, an estimate of the reciprocal condition number of the matrix A, where the reciprocal condition number is defined to be $1 / (\ A\ \times \ A^{-1}\)$. The reciprocal of the condition number is estimated instead of the condition number itself to avoid overflow or division by zero. If RCOND is less than machine precision (in particular, if $RCOND = 0$) then A is singular to working precision. In this case, $INFO > 0$ is returned and the solution and error bounds are not computed.

xFERR	On exit, the estimated forward error bound for each solution vector $X(*, j)$ for $1 \leq j \leq \text{NRHS}$. If X' is the true solution corresponding to $X(*, j)$ then $\text{FERR}(j)$ is an upper bound on the magnitude of the largest element in $X(*, j) - X'$ divided by the magnitude of the largest element in $X(*, j)$.
xBERR	On exit, $\text{BERR}(j)$ is the smallest relative change in any element of A or $B(*, j)$ that makes $X(*, j)$ an exact solution to $AX(*, j) = B(*, j)$ for $1 \leq j \leq \text{NRHS}$.
xWORK	Scratch array with a dimension of $3 \times N$ for real subroutines or $2 \times N$ for complex subroutines.
xWORK2	Scratch array with a dimension of N for complex subroutines.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where $i = \text{INFO}$, had an illegal value. $1 \leq \text{INFO} \leq N$ The leading minor of order i of A, where $i = \text{INFO}$, is not positive definite. The factorization has not been completed, unless $i = N$, and the solution and error bounds could not be computed. INFO = N + 1 RCOND is less than machine precision. The factorization has been completed, but the matrix is singular to working precision, and the solution and error bounds could not be computed.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C      INTEGER          LDB, LDWORK, LDX, N, NRHS
C      PARAMETER        (N = 4)
C      PARAMETER        (NRHS = 1)
C      PARAMETER        (LDB = N)
C      PARAMETER        (LDX = N)
C      PARAMETER        (LDWORK = 2 * N)
C
C      DOUBLE PRECISION B(LDB,NRHS), BERR(NRHS), DIAG(N), DIAGF(N)
C      DOUBLE PRECISION DLOW(N-1), DLOWF(N-1), FERR(NRHS), RCOND
C      DOUBLE PRECISION WORK(LDWORK), X(LDX)
C      INTEGER          ICOL, INFO, IROW
C
C      EXTERNAL          DPTSVX
C
C      Initialize the arrays DIAG and DLOWER to store in symmetric
C      tridiagonal form the 4x4 symmetric positive definite
C      coefficient matrix A shown below. Initialize the array B

```



```

C      to store the right hand side vector b shown below.
C
C      2  -1  0  0      6
C      A = -1  2  -1  0      b = 12
C      0  -1  2  -1      12
C      0  0  -1  2      6
C
C      DATA DIAG / 2.0D0, 2.0D0, 2.0D0, 2.0D0 /
C      DATA DLOW / -1.0D0, -1.0D0, -1.0D0 /
C      DATA B / 6.0D0, 1.2D1, 1.2D1, 6.0D0 /
C
C      Print the initial values of the arrays.
C
C      PRINT 1000
C      DO 100, IROW = 1, N
C      PRINT 1010, (0.0D0, ICOL = 1, IROW - 2),
$          (DLOW(ICOL + 1), ICOL = ABS(IROW - 2), IROW - 2),
$          DIAG(IROW),
$          (DLOW(IROW), ICOL = 1, MIN(1, N - IROW)),
$          (0.0D0, ICOL = IROW + 2, N)
100 CONTINUE
C      PRINT 1020
C      PRINT 1030, B
C
C      Solve the system and print the results.
C
C      CALL DPTSVX ('NOT FACTORED A', N, NRHS, DIAG, DLOW, DIAGF,
$          DLOWF, B, LDB, X, LDX, RCOND, FERR, BERR,
$          WORK, INFO)
C      IF (INFO .NE. 0) THEN
C      PRINT 1040, INFO
C      IF (INFO .EQ. (N + 1)) THEN
C      PRINT 1050
C      STOP 1
C      END IF
C      STOP 2
C      END IF
C      PRINT 1060
C      PRINT 1030, X
C      PRINT 1070, 1.0D0 / RCOND
C      PRINT 1080, (IROW, BERR(IROW), IROW = 1, NRHS)
C      PRINT 1090, (IROW, FERR(IROW), IROW = 1, NRHS)
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (4(3X, F6.3))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (1X, F6.2)
1040 FORMAT (1X, 'Error solving Ax=b, INFO = ', I5)
1050 FORMAT (1X, 'Matrix is singular to working precision.')
1060 FORMAT (/1X, 'x:')
1070 FORMAT (/1X, 'Estimated condition number of A: ', F7.2)
1080 FORMAT (/1X, 'Backward error in system #', I1, ': ', E12.6)
1090 FORMAT (1X, 'Forward error in system #', I1, ': ', E12.6)
C
C      END

```

Sample Output

```
A:
  2.000  -1.000   0.000   0.000
 -1.000   2.000  -1.000   0.000
  0.000  -1.000   2.000  -1.000
  0.000   0.000  -1.000   2.000

b:
  6.00
 12.00
 12.00
  6.00

x:
 18.00
 30.00
 30.00
 18.00

Estimated condition number of A:  12.00
Backward error in system #1: 0.296059E-16
Forward error in system #1:  0.568434E-14
```

UDU or LDL Factorization of a Symmetric Positive Definite Tridiagonal Matrix

The subroutines described in this section compute a UHDL or LDLH factorization of a real symmetric (or Hermitian) positive definite tridiagonal matrix A. It is typical to follow a call to xPTTRF with a call to xPTSVX or xPTTRS to solve a linear system $AX = B$, or to xPTCON to estimate the condition number of A.

Calling Sequence

```
CALL DPTTRF (N, DDIAG, DOFFD, INFO)
CALL SPTTRF (N, SDIAG, SOFFD, INFO)
CALL ZPTTRF (N, DDIAG, ZOFFD, INFO)
CALL CPTTRF (N, SDIAG, COFFD, INFO)

void dpttrf (int n, double *ddiag, double *doffd, int *info)
void spttrf (int n, float *sdiag, float *soffd, int *info)
void zpttrf (int n, double *ddiag, doublecomplex *zoffd, int *info)
void cpttrf (int n, float *sdiag, complex *coffd, int *info)
```

Arguments

N	Order of the matrix A. $N \geq 0$.
xDIAG	On entry, the N diagonal elements of the matrix A. On exit, the N diagonal elements of the diagonal matrix D from the UDU or LDL factorization of A.
xOFFD	On entry, the N-1 off-diagonal elements of the matrix A. On exit, the N-1 off-diagonal elements of the unit bidiagonal factor U or L from the factorization of A.
INFO	On exit: INFO = 0 INFO < 0 INFO > 0 Subroutine completed normally. The ith argument, where $i = \text{INFO}$, had an illegal value. The leading minor of order i of A, where $i = \text{INFO}$, is not positive definite. The factorization could not be completed.


```
      CALL DPTTRS (N, NRHS, DIAG, DLOWER, B, LDB, INFO)
      IF (INFO .NE. 0) THEN
        PRINT 1050, ABS(INFO)
        STOP 2
      END IF
      PRINT 1060
      PRINT 1030, B
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (4(3X, F6.3))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (1X, F6.2)
1040 FORMAT (1X, 'Error factoring A, INFO = ', I5)
1050 FORMAT (1X, 'Illegal argument to DPTTRS, argument #', I1)
1060 FORMAT (/1X, 'x:')
C
      END
```

Sample Output

```
A:
  2.000   -1.000    0.000    0.000
 -1.000    2.000   -1.000    0.000
  0.000   -1.000    2.000   -1.000
  0.000    0.000   -1.000    2.000

b:
  6.00
 12.00
 12.00
  6.00

x:
 18.00
 30.00
 30.00
 18.00
```

Solution to a Linear System in a UDU- or LDL-Factored Symmetric Positive Definite Tridiagonal Matrix

The subroutines described in this section solve a linear system $AX = B$ for a real symmetric (or Hermitian) positive definite tridiagonal matrix A, which has been UDU-factored or LDL-factored by xPTTRF, and general matrices B and X.

Calling Sequence

```
CALL DPTTRS (UPLO, N, NRHS, DDIAG, DOFFD, DB, LDB, INFO)
CALL SPTTRS (UPLO, N, NRHS, SDIAG, SOFFD, SB, LDB, INFO)
CALL ZPTTRS (N, NRHS, DDIAG, ZOFFD, ZB, LDB, INFO)
CALL CPTTRS (N, NRHS, SDIAG, COFFD, CB, LDB, INFO)

void dpttrs (int n, int nrhs, double *ddiag, double *doffd, double
            *db, int ldb, int *info)
void spttrs (int n, int nrhs, float *sdiag, float *soffd, float
            *sb, int ldb, int *info)
void zpttrs (char uplo, int n, int nrhs, double *ddiag,
            doublecomplex *zoffd, doublecomplex *zb, int ldb, int
            *info)
void cpttrs (char uplo, int n, int nrhs, float *sdiag, complex
            *coffd, complex *cb, int ldb, int *info)
```

Arguments

UPLO	(For real matrices) Indicates whether OFFD contains the superdiagonal or the subdiagonal of the matrix A and the form of the factorization. The legal values for UPLO are listed below. Any values not listed are illegal. 'U' or 'u' xOFFD contains the superdiagonal and the factorization is UDU. 'L' or 'l' xOFFD contains the subdiagonal and the factorization is LDL. (The two forms are equivalent if A is real.)
N	Order of the matrix A. $N \geq 0$.
NRHS	Number of right-hand sides, equal to the number of columns of the matrix B. $NRHS \geq 0$.
xDIAG	The N diagonal elements of the matrix D from the UDU or LDL factorization of the matrix A, as computed by xPTTRF.

xOFFD	The N-1 off-diagonal elements of the unit bidiagonal factor U or L from the UDU or LDL factorization of the matrix A, as computed by xPTTRF.
xB	On entry, the N×NRHS right-hand side matrix B. On exit, the N×NRHS solution matrix X.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where $i = \text{INFO} $, had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C   INTEGER          LDB, N, NRHS
C   PARAMETER        (N = 4)
C   PARAMETER        (NRHS = 1)
C   PARAMETER        (LDB = N)
C
C   DOUBLE PRECISION B(LDB,NRHS), DIAG(N), DLOWER(N-1)
C   INTEGER          ICOL, INFO, IROW
C
C   EXTERNAL          DPTTRF, DPTTRS
C   INTRINSIC         ABS
C
C   Initialize the arrays DIAG and DLOWER to store in symmetric
C   tridiagonal form the 4x4 symmetric positive definite
C   coefficient matrix A shown below. Initialize the array B
C   to store the right hand side vector b shown below.
C
C   A =
C   2  -1  0  0      b =
C  -1  2  -1  0      6
C   0  -1  2  -1      12
C   0   0  -1  2      6
C
C   DATA DIAG / 2.0D0, 2.0D0, 2.0D0, 2.0D0 /
C   DATA DLOWER / -1.0D0, -1.0D0, -1.0D0 /
C   DATA B / 6.0D0, 1.2D1, 1.2D1, 6.0D0 /
C
C   Print the initial values of the arrays.
C
C   PRINT 1000
C   DO 100, IROW = 1, N
C     PRINT 1010, (0.0D0, ICOL = 1, IROW - 2),
$       (DLOWER(ICOL + 1), ICOL = ABS(IROW - 2), IROW - 2),
$       DIAG(IROW),

```

```

$          (DLOWER(IROW), ICOL = 1, MIN(1, N - IROW)),
$          (0.0D0, ICOL = IROW + 2, N)
100 CONTINUE
   PRINT 1020
   PRINT 1030, B
C
C      LDL factor A.
C
      CALL DPTTRF (N, DIAG, DLOWER, INFO)
      IF (INFO .NE. 0) THEN
         PRINT 1040, INFO
         STOP 1
      END IF
C
C      Use the factored form of A to solve Ax=b then print
C      the result.
C
      CALL DPTTRS (N, NRHS, DIAG, DLOWER, B, LDB, INFO)
      IF (INFO .NE. 0) THEN
         PRINT 1050, ABS(INFO)
         STOP 2
      END IF
      PRINT 1060
      PRINT 1030, B
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (4(3X, F6.3))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (1X, F6.2)
1040 FORMAT (1X, 'Error factoring A, INFO = ', I5)
1050 FORMAT (1X, 'Illegal argument to DPTTRS, argument #', I1)
1060 FORMAT (/1X, 'x:')
C
      END

```

Sample Output

```

A:
  2.000  -1.000   0.000   0.000
 -1.000   2.000  -1.000   0.000
  0.000  -1.000   2.000  -1.000
  0.000   0.000  -1.000   2.000

b:
  6.00
 12.00
 12.00
  6.00

```



```
x:  
18.00  
30.00  
30.00  
18.00
```

Eigenvalues and Eigenvectors of a Symmetric Matrix in Banded Storage (Simple Driver)

The subroutines described in this section compute the eigenvalues and, optionally, the eigenvectors of a symmetric matrix A in banded storage. The eigenvectors are normalized to have a Euclidean norm of 1. Note that the expert driver xSBEVX is also available.

Calling Sequence

```
CALL DSBEV (JOBZ, UPLO, N, NDIAG, DA, LDA, DW, DZ, LDZ, DWORK,
            INFO)
CALL SSBEV (JOBZ, UPLO, N, NDIAG, SA, LDA, SW, SZ, LDZ, SWORK,
            INFO)

void dsbev (char jobz, char uplo, int n, int ndiag, double *da,
            int lda, double *dw, double *dz, int ldz, int *info)
void ssbev (char jobz, char uplo, int n, int ndiag, float *sa, int
            lda, float *sw, float *sz, int ldz, int *info)
```

Arguments

JOBZ	Indicates whether to compute eigenvalues only or to compute both eigenvalues and eigenvectors. The legal values for JOBZ are listed below. Any values not listed below are illegal. 'N' or 'n' Compute eigenvalues only. 'V' or 'v' Compute both eigenvalues and eigenvectors.
UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
NDIAG	Number of superdiagonals or subdiagonals of the matrix A. $N-1 \geq \text{NDIAG} \geq 0$ but if $N = 0$ then $\text{NDIAG} = 0$. If UPLO = 'U' or 'u', NDIAG is the number of superdiagonals. If UPLO = 'L' or 'l', NDIAG is the number of subdiagonals.
xA	On entry, the upper or lower triangle of the matrix A. On exit, A is overwritten.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $\text{LDA} \geq \text{NDIAG} + 1$.

xW	On exit, the N eigenvalues in ascending order.	
xZ	On exit, if JOBZ = 'V' or 'v' then Z contains the orthonormal eigenvectors. If JOBZ = 'N' or 'n' then Z is not used.	
LDZ	Leading dimension of the array Z as specified in a dimension or type statement. LDZ ≥ 1. If JOBZ = 'V' or 'v' then LDZ ≥ max(1, N).	
xWORK	Scratch array with a dimension of max(1, 3 × N − 2).	
INFO	On exit:	
	INFO = 0	Subroutine completed normally.
	INFO < 0	The <i>i</i> th argument, where <i>i</i> = INFO , had an illegal value.
	INFO > 0	Convergence failure.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
C   INTEGER          LDA, LDEVEC, LDWORK, N, NDIAG
C   PARAMETER        (N = 4)
C   PARAMETER        (NDIAG = 1)
C   PARAMETER        (LDA = NDIAG + 1)
C   PARAMETER        (LDEVEC = N)
C   PARAMETER        (LDWORK = 3 * N - 2)

C
C   DOUBLE PRECISION A(LDA,N), EVALS(N), EVECS(LDEVEC,N)
C   DOUBLE PRECISION WORK(LDWORK)
C   INTEGER          ICOL, INFO, IROW

C
C   EXTERNAL          DSBEV
C   INTRINSIC         ABS

C
C   Initialize the array A to store the symmetric tridiagonal
C   matrix A shown below.
C
C   A =
C   1   0   0   0
C   0   2   1   0
C   0   1   2   0
C   0   0   0   1

C
C   DATA A / 8D8, 1.0D0, 0.0D0, 2.0D0, 1.0D0, 2.0D0, 0.0D0,
C   $         1.0D0 /

C
C   Print the initial values of the arrays.
C
C   PRINT 1000
C   PRINT 1010, A(2,1), A(1,2), 0.0D0, 0.0D0
C   PRINT 1010, A(1,2), A(2,2), A(1,3), 0.0D0
C   PRINT 1010, 0.0D0, A(1,3), A(2,3), A(1,4)
C   PRINT 1010, 0.0D0, 0.0D0, A(1,4), A(2,4)

```

```

        PRINT 1020
        DO 100, IROW = 1, LDA
            PRINT 1010, (A(IROW,ICOL), ICOL = 1, N)
100    CONTINUE
C
C        Compute the eigenvalues and right eigenvectors of A.
C
        CALL DSBEV ('VECTORS AND EIGENVALUES',
$          'UPPER TRIANGLE A STORED', N, NDIAG, A, LDA,
$          EVALS, EVECS, LDEVEC, WORK, INFO)
        IF (INFO .NE. 0) THEN
            IF (INFO .LT. 0) THEN
                PRINT 1030, ABS(INFO)
                STOP 1
            ELSE
                PRINT 1040, INFO
                STOP 2
            END IF
        END IF
C
C        Print the eigenvalues and eigenvectors.
C
        PRINT 1050
        DO 120, ICOL = 1, N
            PRINT 1060, EVALS(ICOL), (EVECS(IROW,ICOL), IROW = 1, N)
120    CONTINUE
C
1000    FORMAT (1X, 'A in full form:')
1010    FORMAT (4(3X, F9.6))
1020    FORMAT (/1X,
$          'A in symmetric banded form: ',
$          ' (* in unused elements)')
1030    FORMAT (/1X, 'Illegal argument to DSBEV, argument #', I2)
1040    FORMAT (/1X, 'Convergence failure, INFO = ', I2)
1050    FORMAT (/1X, 'Eigenvalue', 10X, 'Eigenvector**T')
1060    FORMAT (1X, F7.2, 6X, '[', 3(F5.3, ' ', ' '), F5.3, ']')
C
        END

```

Sample Output

```

A in full form:
  1.000000    0.000000    0.000000    0.000000
  0.000000    2.000000    1.000000    0.000000
  0.000000    1.000000    2.000000    0.000000
  0.000000    0.000000    0.000000    1.000000

A in symmetric banded form: (* in unused elements)
*****
  1.000000    2.000000    2.000000    1.000000

```

Eigenvalue	Eigenvector**T
1.00	[1.000, 0.000, 0.000, 0.000]
1.00	[0.000, -.707, 0.707, 0.000]
1.00	[0.000, 0.000, 0.000, 1.000]
3.00	[0.000, 0.707, 0.707, 0.000]

Eigenvalues and Eigenvectors of a Symmetric Matrix in Banded Storage (Expert Driver)

The subroutines described in this section compute the eigenvalues and, optionally, the eigenvectors of a symmetric matrix A in banded storage. The eigenvectors are normalized to have a Euclidean norm of 1. Eigenvalues and eigenvectors can be selected by specifying either a range of values or a range of indices for the desired eigenvalues. Note that the simple driver xSBEV is also available.

Calling Sequence

```
CALL DSBEVX (JOBZ, RANGE, UPLO, N, NDIAG, DA, LDA, DQ, LDQ, DVL,
            DVU, IL, IU, DABTOL, NFOUND, DW, DZ, LDZ, DWORK,
            IWORK2, IFAIL, INFO)
CALL SSBEVX (JOBZ, RANGE, UPLO, N, NDIAG, SA, LDA, SQ, LDQ, SVL,
            SVU, IL, IU, SABTOL, NFOUND, SW, SZ, LDZ, SWORK,
            IWORK2, IFAIL, INFO)

void dsbevz (char jobz, char range, char uplo, int n, int ndiag,
            double *da, int lda, double *dq, int ldq, double dvl,
            double dvu, int il, int iu, double dabtol, int *nfound,
            double *dw, double *dz, int ldz, int *ifail, int *info)
void ssbevz (char jobz, char range, char uplo, int n, int ndiag,
            float *sa, int lda, float *sq, int ldq, float svl,
            float svu, int il, int iu, float sabtol, int *nfound,
            float *sw, float *sz, int ldz, int *ifail, int *info)
```

Arguments

JOBZ	Indicates whether to compute eigenvalues only or to compute both eigenvalues and eigenvectors. The legal values for JOBZ are listed below. Any values not listed below are illegal. 'N' or 'n' Compute eigenvalues only. 'V' or 'v' Compute both eigenvalues and eigenvectors.
RANGE	Indicates whether to compute all eigenvalues or some subset. Legal values for RANGE are shown below. Any values not shown below are illegal. 'A' or 'a' All eigenvalues will be computed. 'I' or 'i' The ILth through IUth eigenvalues will be computed; see the descriptions of the IL and IU arguments below.

	'V' or 'v'	All eigenvalues in the half-open interval (VL,VU] will be computed; see the descriptions of the VL and VU arguments below.
UPLO		Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal.
	'U' or 'u'	xA contains the upper triangle.
	'L' or 'l'	xA contains the lower triangle.
N		Order of the matrix A. $N \geq 0$.
NDIAG		Number of superdiagonals or subdiagonals of the matrix A. $N-1 \geq \text{NDIAG} \geq 0$ but if $N = 0$ then $\text{NDIAG} = 0$. If UPLO = 'U' or 'u', NDIAG is the number of superdiagonals. If UPLO = 'L' or 'l', NDIAG is the number of subdiagonals.
xA		On entry, the upper or lower triangle of the matrix A. On exit, A has been overwritten.
LDA		Leading dimension of the array A as specified in a dimension or type statement. $\text{LDA} \geq \text{NDIAG} + 1$.
xQ		On exit, if JOBZ = 'V' or 'v' then Q contains the $N \times N$ orthogonal matrix used in the reduction to tridiagonal form. If JOBZ = 'N' or 'n' then Q is not used.
LDQ		Leading dimension of the array Q as specified in a dimension or type statement. $\text{LDQ} \geq \max(1, N)$.
xVL, xVU		If RANGE = 'V' or 'v' then only eigenvalues in the half-open interval (VL,VU] will be computed. Otherwise not used.
IL, IU		If RANGE = 'T' or 't' then only the ILth through IUth eigenvalues will be computed. Otherwise not used. $1 \leq \text{IL} \leq \text{IU} \leq N$.
xABTOL		The absolute error tolerance for the eigenvalues. An approximate eigenvalue is considered to have converged when it lies in an interval [a,b] of width less than or equal to $\text{ABTOL} + \epsilon \times \max(a , b)$ where ϵ is machine precision. If $\text{ABTOL} \leq 0$ then $\epsilon \times \ T\ _1$ will be used in its place, where $\ T\ _1$ is the 1-norm of the tridiagonal matrix obtained by reducing A to tridiagonal form.
NFOUND		On exit, the total number of eigenvalues found. $0 \leq \text{NFOUND} \leq N$. If RANGE = 'A' or 'a' then $\text{NFOUND} = N$. If RANGE = 'T' or 't' then $\text{NFOUND} = \text{IU} - \text{IL} + 1$.
xW		On exit, W(1:NFOUND) contains the computed eigenvalues in ascending order.
xZ		On exit, if JOBZ = 'V' or 'v' then the first NFOUND columns of Z contain the orthonormal eigenvectors corresponding to the selected eigenvalues. If an eigenvector fails to converge then the corresponding column of Z contains an approximation to that eigenvector and its index is returned in IFAIL. If JOBZ = 'N' or 'n' then Z is not used.

	Note that Z must have at least NFOUND columns. If RANGE = 'V' or 'v' then NFOUND may not be known in advance and an upper bound must be used. For A(LDZ,K) where NFOUND > K, the eigenvectors will be stored into memory with unpredictable results.						
LDZ	Leading dimension of the array Z as specified in a dimension or type statement. $LDZ \geq 1$. If JOBZ = 'V' or 'v' then $LDZ \geq \max(1, N)$.						
xWORK	Scratch array with a dimension of $7 \times N$.						
IWORK2	Scratch array with a dimension of $5 \times N$.						
IFAIL	On exit, an N-element array that indicates which eigenvectors that failed to converge. If JOBZ = 'V' or 'v' and INFO = 0 then all N elements contain zero. If INFO > 0 then IFAIL contains the indices of the eigenvectors that failed to converge. If JOBZ = 'N' or 'n' then IFAIL is not used.						
INFO	On exit: <table> <tr> <td>INFO = 0</td><td>Subroutine completed normally.</td></tr> <tr> <td>INFO < 0</td><td>The ith argument, where $i = \text{INFO}$, had an illegal value.</td></tr> <tr> <td>INFO > 0</td><td>There were i eigenvectors, where $i = \text{INFO}$, that failed to converge. Their indices are stored in IFAIL.</td></tr> </table>	INFO = 0	Subroutine completed normally.	INFO < 0	The ith argument, where $i = \text{INFO} $, had an illegal value.	INFO > 0	There were i eigenvectors, where $i = \text{INFO}$, that failed to converge. Their indices are stored in IFAIL.
INFO = 0	Subroutine completed normally.						
INFO < 0	The ith argument, where $i = \text{INFO} $, had an illegal value.						
INFO > 0	There were i eigenvectors, where $i = \text{INFO}$, that failed to converge. Their indices are stored in IFAIL.						

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
C   INTEGER          LDA, LDEVEC, LDIWRK, LDQ, LDWORK, N, NDIAG
C   PARAMETER        (N = 4)
C   PARAMETER        (NDIAG = 1)
C   PARAMETER        (LDA = NDIAG + 1)
C   PARAMETER        (LDEVEC = N)
C   PARAMETER        (LDIWRK = 5 * N)
C   PARAMETER        (LDWORK = 7 * N)
C   PARAMETER        (LDQ = N)
C
C   DOUBLE PRECISION A(LDA,N), EVALS(N), EVECS(LDEVEC,N), TEMP
C   DOUBLE PRECISION Q(LDQ,N), WORK(LDWORK)
C   INTEGER          ICOL, IFAIL(N), INFO, IROW, ITEMP
C   INTEGER          IWORK(LDIWRK), NFOUND
C
C   EXTERNAL          DSBEVX
C   INTRINSIC         ABS
C
C   Initialize the array A to store the symmetric tridiagonal
C   matrix A shown below.
C
C       1   0   0   0
C   A =  0   2   1   0

```



```
C      0      1      2      0
C      0      0      0      1
C
C      DATA A / 8D8, 1.0D0, 0.0D0, 2.0D0, 1.0D0, 2.0D0, 0.0D0,
$          1.0D0 /
C
C      PRINT 1000
C      PRINT 1010, A(2,1), A(1,2), 0.0D0, 0.0D0
C      PRINT 1010, A(1,2), A(2,2), A(1,3), 0.0D0
C      PRINT 1010, 0.0D0, A(1,3), A(2,3), A(1,4)
C      PRINT 1010, 0.0D0, 0.0D0, A(1,4), A(2,4)
C      PRINT 1020
C      DO 100, IROW = 1, LDA
C          PRINT 1010, (A(IROW,ICOL), ICOL = 1, N)
100 CONTINUE
C
C      Compute the eigenvalues and right eigenvectors of A.
C
C      CALL DSBEVX ('VECTORS AND EIGENVALUES', 'ALL EIGENVALUES',
$          'UPPER TRIANGLE A STORED', N, NDIAG, A, LDA,
$          Q, LDQ, TEMP, TEMP, ITEMP, ITEMP, 0.0D0,
$          NFOUND, EVALS, EVECS, LDEVEC, WORK, IWORK,
$          IFAIL, INFO)
C      IF (INFO .NE. 0) THEN
C          IF (INFO .LT. 0) THEN
C              PRINT 1030, ABS(INFO)
C              STOP 1
C          ELSE
C              PRINT 1040, INFO
C              STOP 2
C          END IF
C      END IF
C
C      Print the eigenvalues and eigenvectors.
C
C      PRINT 1050
C      DO 120, ICOL = 1, N
C          PRINT 1060, EVALS(ICOL), (EVECS(IROW,ICOL), IROW = 1, N)
120 CONTINUE
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F9.6))
1020 FORMAT (/1X,
$          'A in symmetric banded form: '
$          ' (* in unused elements)')
1030 FORMAT (/1X, 'Illegal argument to DSBEVX, argument #', I2)
1040 FORMAT (/1X, 'Convergence failure, INFO = ', I2)
1050 FORMAT (/1X, 'Eigenvalue', 10X, 'Eigenvector**T')
1060 FORMAT (1X, F7.2, 6X, '[', 3(F5.3, ' '), F5.3, ']')
C
C      END
```

Sample Output

```

A in full form:
  1.000000  0.000000  0.000000  0.000000
  0.000000  2.000000  1.000000  0.000000
  0.000000  1.000000  2.000000  0.000000
  0.000000  0.000000  0.000000  1.000000

A in symmetric banded form: (* in unused elements)
*****
  0.000000  1.000000  0.000000
  1.000000  2.000000  2.000000  1.000000

Eigenvalue      Eigenvector**T
  1.00    [1.000, 0.000, 0.000, 0.000]
  1.00    [0.000, -.707, 0.707, 0.000]
  1.00    [0.000, 0.000, 0.000, 1.000]
  3.00    [0.000, 0.707, 0.707, 0.000]

```

Reciprocal Condition Number of a UDU- or LDL-Factored Symmetric Matrix in Packed Storage

The subroutines described in this section estimate the reciprocal condition number of a symmetric matrix A in packed storage, which has been UDU-factored or LDL-factored by xSPTRF.

Calling Sequence

```
CALL DSPCON (UPLO, N, DA, IPIVOT, DANORM, DRCOND, DWORK, IWORK2,
             INFO)
CALL SSPCON (UPLO, N, SA, IPIVOT, SANORM, SRCOND, SWORK, IWORK2,
             INFO)
CALL ZSPCON (UPLO, N, ZA, IPIVOT, DANORM, DRCOND, ZWORK, INFO)
CALL CSPCON (UPLO, N, CA, IPIVOT, SANORM, SRCOND, CWORK, INFO)

void dspcon (char uplo, int n, double *da, int *ipivot, double
             danorm, double *drcond, int *info)
void sspcon (char uplo, int n, float *sa, int *ipivot, float
             sanorm, float *srcond, int *info)
void zspcon (char uplo, int n, doublecomplex *za, int *ipivot,
             double danorm, double *drcond, int *info)
void cspcon (char uplo, int n, complex *ca, int *ipivot, float
             sanorm, float *srcond, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
xA	UDU or LDL factorization of the matrix A, as computed by xSPTRF. The dimension of xA is $(N \times N + N) / 2$.
IPIVOT	Pivot indices as computed by xSPTRF.
xANORM	The 1-norm or ∞ -norm of matrix A.

xRCOND	On exit, the estimated reciprocal of the condition number of the matrix A where the reciprocal condition number of A is defined to be $1 / (ANORM \times \ A^{-1}\)$. The reciprocal of the condition number is estimated instead of the condition number itself to avoid overflow or division by zero. If RCOND is less than machine precision (in particular, if RCOND = 0) then A is singular to working precision.
xWORK	Scratch array with a dimension of $2 \times N$.
IWORK2	Scratch array with a dimension of N for real subroutines.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where i = INFO , had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C   INTEGER          LDA, LDIWRK, LDWORK, N
C   PARAMETER        (N = 4)
C   PARAMETER        (LDA = (N * (N + 1)) / 2)
C   PARAMETER        (LDIWRK = N)
C   PARAMETER        (LDWORK = 2 * N)
C
C   DOUBLE PRECISION A(LDA), ANORM, RCOND, WORK(LDWORK)
C   INTEGER          INFO, IPIVOT(N), IWORK(LDIWRK)
C
C   EXTERNAL          DSPCON, DSPTRF
C   INTRINSIC         ABS
C
C   Initialize the array A to store in packed symmetric form the
C   4x4 symmetric matrix A shown below.
C
C       2   -1   0   0
C   A = -1   2   -1   0
C       0   -1   2   -1
C       0   0   -1   2
C
C   DATA A / 2.0D0, -1.0D0, 2.0D0, 0.0D0, -1.0D0, 2.0D0,
$           0.0D0, 0.0D0, -1.0D0, 2.0D0 /
C
C   Print the initial values of the arrays.
C
C   PRINT 1000
C   PRINT 1010, A(1), A(2), A(4), A(7)
C   PRINT 1010, A(2), A(3), A(5), A(8)
C   PRINT 1010, A(4), A(5), A(6), A(9)
C   PRINT 1010, A(7), A(8), A(9), A(10)
C

```

```
C      LDL factor A.
C
      CALL DSPTRF ('UPPER TRIANGLE OF A STORED', N, A, IPIVOT,
$          INFO)
      IF (INFO .NE. 0) THEN
          PRINT 1020, INFO
          STOP 1
      END IF
C
C      Compute and print the condition number of A.
C
      ANORM = 4.0D0
      CALL DSPCON ('UPPER TRIANGLE FACTOR STORED', N, A, IPIVOT,
$          ANORM, RCOND, WORK, IWORK, INFO)
      IF (INFO .NE. 0) THEN
          PRINT 1030, ABS(INFO)
          STOP 2
      END IF
      PRINT 1040, 1.0D0 / RCOND
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (4(3X, F6.3))
1020 FORMAT (1X, 'Error factoring A, INFO = ', I5)
1030 FORMAT (1X, 'Illegal argument to DSPCON, argument #', I2)
1040 FORMAT (/1X, 'Estimated condition number of A = ', F8.4)
C
      END
```

Sample Output

```
A:
  2.000  -1.000   0.000   0.000
 -1.000   2.000  -1.000   0.000
  0.000  -1.000   2.000  -1.000
  0.000   0.000  -1.000   2.000

Estimated condition number of A = 12.0000
```

Eigenvalues and Eigenvectors of a Symmetric Matrix in Packed Storage (Simple Driver)

The subroutines described in this section compute the eigenvalues and, optionally, the eigenvectors of a symmetric matrix A in packed storage. The eigenvectors are normalized to have a Euclidean norm of 1. Note that the expert driver xSPEVX is also available.

Calling Sequence

```
CALL DSPEV (JOBZ, UPLO, N, DA, DW, DZ, LDZ, DWORK, INFO)
CALL SSPEV (JOBZ, UPLO, N, SA, SW, SZ, LDZ, SWORK, INFO)

void dspev (char jobz, char uplo, int n, double *da, double *dw,
           double *dz, int ldz, int *info)
void sspev (char jobz, char uplo, int n, float *sa, float *sw,
           float *sz, int ldz, int *info)
```

Arguments

JOBZ	Indicates whether to compute eigenvalues only or to compute both eigenvalues and eigenvectors. The legal values for JOBZ are listed below. Any values not listed below are illegal. 'N' or 'n' Compute eigenvalues only. 'V' or 'v' Compute both eigenvalues and eigenvectors.
UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
xA	On entry, the upper or lower triangle of the matrix A. The dimension of xA is $(N \times N + N) / 2$. On exit, A is overwritten.
xW	On exit, the N eigenvalues in ascending order.
xZ	On exit, if JOBZ = 'V' or 'v' then Z contains the orthonormal eigenvectors. If JOBZ = 'N' or 'n' then Z is not used.
LDZ	Leading dimension of the array Z as specified in a dimension or type statement. $LDZ \geq 1$. If JOBZ = 'V' or 'v' then $LDZ \geq \max(1, N)$.
xWORK	Scratch array with a dimension of $\max(1, 3 \times N)$.

INFO	On exit:	
	INFO = 0	Subroutine completed normally.
	INFO < 0	The ith argument, where i = INFO , had an illegal value.
	INFO > 0	Convergence failure.

Sample Program

```

C      PROGRAM TEST
C      IMPLICIT NONE
C
C      INTEGER          LDEVEC, LDWORK, LENGTA, N
C      PARAMETER        (N = 3)
C      PARAMETER        (LDEVEC = N)
C      PARAMETER        (LDWORK = 3 * N)
C      PARAMETER        (LENGTA = (N * (N + 1)) / 2)
C
C      DOUBLE PRECISION A(LENGTA), EVALS(N), EVECS(LDEVEC,N)
C      DOUBLE PRECISION WORK(LDWORK)
C      INTEGER          ICOL, INFO, IROW
C
C      EXTERNAL          DSPEV
C      INTRINSIC         ABS
C
C      Initialize the array A to store the matrix A shown below.
C
C      9  1  1
C      A = 1  9  1
C      1  1  9
C
C      DATA A / 9.0D0, 1.0D0, 9.0D0, 1.0D0, 1.0D0, 9.0D0 /
C
C      PRINT 1000
C      PRINT 1010, A(1), A(2), A(4)
C      PRINT 1010, A(2), A(3), A(5)
C      PRINT 1010, A(4), A(5), A(6)
C
C      Compute the eigenvalues and eigenvectors of A.
C
C      CALL DSPEV ('VALUES AND VECTORS', 'UPPER TRIANGULAR A', N,
$      A, EVALS, EVECS, LDEVEC, WORK, INFO)
C      IF (INFO .NE. 0) THEN
C        IF (INFO .LT. 0) THEN
C          PRINT 1020, ABS(INFO)
C          STOP 1
C        ELSE
C          PRINT 1030, INFO
C          STOP 2
C        END IF
C      END IF
C
C

```

```

C      Print the eigenvalues and eigenvectors.
C
      PRINT 1040
      DO 120, ICOL = 1, N
        PRINT 1050, EVALS(ICOL), (EVECS(IROW,ICOL), IROW = 1, N)
      120 CONTINUE
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (3(3X, F4.1))
1020 FORMAT (/1X, 'Illegal argument to DSPEV, argument #', I2)
1030 FORMAT (/1X, 'Convergence failure, INFO = ', I2)
1040 FORMAT (/1X, 'Eigenvalue', 4X, 'Eigenvector**T')
1050 FORMAT (1X, F7.2, 5X, '[', F4.2, ', ', F4.2, ', ',
$          F4.2, ']'')
C
      END

```

Sample Output

```

A:
  9.0    1.0    1.0
  1.0    9.0    1.0
  1.0    1.0    9.0

Eigenvalue      Eigenvector**T
  8.00    [0.04, -.73, 0.69]
  8.00    [0.82, -.37, -.44]
 11.00    [0.58, 0.58, 0.58]

```


Eigenvalues and Eigenvectors of a Symmetric Matrix in Packed Storage (Expert Driver)

The subroutines described in this section compute the eigenvalues and, optionally, the eigenvectors of a symmetric matrix A in packed storage. The eigenvectors are normalized to have a Euclidean norm of 1. Eigenvalues and eigenvectors can be selected by specifying either a range of values or a range of indices for the desired eigenvalues. Note that the simple driver xSPEV is also available.

Calling Sequence

```
CALL DSPEVX (JOBZ, RANGE, UPLO, N, DA, DVL, DVU, IL, IU, DABTOL,
             NFOUND, DW, DZ, LDZ, DWORK, IWORK2, IFAIL, INFO )
CALL SSPEVX (JOBZ, RANGE, UPLO, N, SA, SVL, SVU, IL, IU, SABTOL,
             NFOUND, SW, SZ, LDZ, SWORK, IWORK2, IFAIL, INFO )

void dspevx (char jobz, char range, char uplo, int n, double *da,
            double dvl, double dvu, int il, int iu, double dabtol,
            int *nfound, double *dw, double *dz, int ldz, int
            *ifail, int *info)
void sspevx (char jobz, char range, char uplo, int n, float *sa,
            float svl, float svu, int il, int iu, float sabtol, int
            *nfound, float *sw, float *sz, int ldz, int *ifail, int
            *info)
```

Arguments

JOBZ	Indicates whether to compute eigenvalues only or to compute both eigenvalues and eigenvectors. The legal values for JOBZ are listed below. Any values not listed below are illegal. 'N' or 'n' Compute eigenvalues only. 'V' or 'v' Compute both eigenvalues and eigenvectors.
RANGE	Indicates whether to compute all eigenvalues or some subset. Legal values for RANGE are shown below. Any values not shown below are illegal. 'A' or 'a' All eigenvalues will be computed. 'I' or 'i' The ILth through IUth eigenvalues will be computed; see the descriptions of the IL and IU arguments below.

	'V' or 'v'	All eigenvalues in the half-open interval (VL,VU] will be computed; see the descriptions of the VL and VU arguments below.
UPLO		Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal.
	'U' or 'u'	xA contains the upper triangle.
	'L' or 'l'	xA contains the lower triangle.
N		Order of the matrix A. $N \geq 0$.
xA		On entry, the upper or lower triangle of the matrix A. The dimension of xA is $(N \times N + N) / 2$. On exit, A is overwritten.
xVL, xVU		If RANGE = 'V' or 'v' then only eigenvalues in the half-open interval (VL,VU] will be computed. Otherwise not used.
IL, IU		If RANGE = 'I' or 'i' then only the ILth through IUth eigenvalues will be computed. Otherwise not used. $1 \leq IL \leq IU \leq N$.
xABTOL		The absolute error tolerance for the eigenvalues. An approximate eigenvalue is considered to have converged when it lies in an interval [a,b] of width less than or equal to $ABTOL + \epsilon \times \max(a , b)$ where ϵ is machine precision. If $ABTOL \leq 0$ then $\epsilon \times \ T\ _1$ will be used in its place, where $\ T\ _1$ is the 1-norm of the tridiagonal matrix obtained by reducing A to tridiagonal form.
NFOUND		On exit, the total number of eigenvalues found. $0 \leq NFOUND \leq N$. If RANGE = 'A' or 'a' then $NFOUND = N$. If RANGE = 'I' or 'i' then $NFOUND = IU - IL + 1$.
xW		On exit, W(1:NFOUND) contains the computed eigenvalues in ascending order.
xZ		On exit, if JOBZ = 'V' or 'v' then the first NFOUND columns of Z contain the orthonormal eigenvectors corresponding to the selected eigenvalues. If an eigenvector fails to converge then the corresponding column of Z contains an approximation to that eigenvector and its index is returned in IFAIL. If JOBZ = 'N' or 'n' then Z is not used. Note that Z must have at least NFOUND columns. If RANGE = 'V' or 'v' then NFOUND may not be known in advance and an upper bound must be used. For A(LDZ,K) where $NFOUND > K$, the eigenvectors will be stored into memory with unpredictable results.
LDZ		Leading dimension of the array Z as specified in a dimension or type statement. $LDZ \geq 1$. If JOBZ = 'V' or 'v' then $LDZ \geq \max(1, N)$.
xWORK		Scratch array with a dimension of $8 \times N$.
IWORK2		Scratch array with a dimension of $5 \times N$.

IFAIL	On exit, an N-element array that indicates which eigenvectors that failed to converge. If JOBZ = 'V' or 'v' and INFO = 0 then all N elements contain zero. If INFO > 0 then IFAIL contains the indices of the eigenvectors that failed to converge. If JOBZ = 'N' or 'n' then IFAIL is not used.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The <i>i</i> th argument, where <i>i</i> = INFO , had an illegal value. INFO > 0 There were <i>i</i> eigenvectors, where <i>i</i> = INFO, that failed to converge. Their indices are stored in IFAIL.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER          LDEVEC, LDIWRK, LDWORK, LENGTA, N
  PARAMETER        (N = 3)
  PARAMETER        (LDEVEC = N)
  PARAMETER        (LDIWRK = 5 * N)
  PARAMETER        (LDWORK = 8 * N)
  PARAMETER        (LENGTA = (N * N + N) / 2)

C
  DOUBLE PRECISION A(LENGTA), TEMP, EVALS(N), EVECS(LDEVEC,N)
  DOUBLE PRECISION WORK(LDWORK)
  INTEGER          ICOL, IFAIL(N), INFO, IROW, ITEMP
  INTEGER          IWORK(LDIWRK), NFOUND

C
  EXTERNAL          DSYEVX
  INTRINSIC         ABS

C
C   Initialize the array A to store the matrix A shown below.
C
C   9  1  1
C   A = 1  9  1
C   1  1  9
C
  DATA A / 9.0D0, 1.0D0, 9.0D0, 1.0D0, 1.0D0, 9.0D0 /

C
  PRINT 1000
  PRINT 1010, A(1), A(2), A(4)
  PRINT 1010, A(2), A(3), A(5)
  PRINT 1010, A(4), A(5), A(6)

C
C   Compute the eigenvalues and eigenvectors of A.
C
C   Arguments 6 - 9 are not used because RANGE (argument 2)
C   indicates that all eigenvalues should be found. Note

```

```

C      however that the scratch values passed into DSPEVX for
C      arguments 6 - 9 are of the correct type. Arguments 6 and 7
C      take the place of double precision values, so TEMP is
C      declared double precision. Arguments 8 and 9 take the place
C      of integer arguments, so ITEMP is declared integer. Even
C      though these arguments are not referenced, passing arguments
C      of the wrong type may cause run time errors due to the
C      alignment of the arguments in memory.
C
C      CALL DSPEVX ('VALUES AND VECTORS', 'ALL EIGENVALUES',
$      'UPPER TRIANGULAR A', N, A, TEMP, TEMP, ITEMP,
$      ITEMP, 0.0D0, NFOUND, EVALS, EVECS, LDEVEC,
$      WORK, IWORK, IFAIL, INFO)
C      IF (INFO .NE. 0) THEN
C          IF (INFO .LT. 0) THEN
C              PRINT 1020, ABS(INFO)
C              STOP 1
C          ELSE
C              PRINT 1030, INFO
C              STOP 2
C          END IF
C      END IF
C
C      Print the eigenvalues and eigenvectors.
C
C      PRINT 1040
C      DO 120, ICOL = 1, N
C          PRINT 1050, EVALS(ICOL), (EVECS(IROW,ICOL), IROW = 1, N)
120  CONTINUE
C
1000  FORMAT (1X, 'A:')
1010  FORMAT (3(3X, F4.1))
1020  FORMAT (/1X, 'Illegal argument to DSPEVX, argument #', I2)
1030  FORMAT (/1X, 'Convergence failure, INFO = ', I2)
1040  FORMAT (/1X, 'Eigenvalue', 4X, 'Eigenvector**T')
1050  FORMAT (1X, F7.2, 5X, '[', F4.2, ', ', F4.2, ', ', F4.2, ', ',
$      F4.2, ']')
C
C      END

```

Sample Output

```

A:
  9.0    1.0    1.0
  1.0    9.0    1.0
  1.0    1.0    9.0

Eigenvalue    Eigenvector**T
  8.00    [0.04, -.73, 0.69]
  8.00    [0.82, -.37, -.44]
 11.00    [0.58, 0.58, 0.58]

```

Generalized Eigenvalues and Eigenvectors for Symmetric Matrices in Packed Storage

The subroutines in this section compute the generalized eigenvalues (α , β) for a pair of symmetric matrices A and B in packed storage. Optionally, these subroutines can also compute the generalized eigenvectors.

A generalized eigenvalue is a ratio $\lambda = \alpha / \beta$ such that $AX = \lambda BX$ or $\det(A - \lambda B) = 0$ for general matrices A, B, and X. λ is typically represented as a ratio rather than as a scalar because there are reasonable interpretations for $\alpha = 0$, $\beta = 0$, and for $\alpha = \beta = 0$. A right generalized eigenvector x corresponding to a generalized eigenvalue λ is defined by $(A - \lambda B)x = 0$. A left generalized eigenvector x corresponding to a generalized eigenvalue λ is defined by $(A - \lambda B)^T x = 0$. A good reference for generalized eigenproblems is the book *Matrix Computations*, 2nd. ed. by Golub and van Loan (1989, The Johns Hopkins University Press).

Calling Sequence

```
CALL DSPGV (ITYPE, JOBZ, UPLO, N, DA, DB, DW, DZ, LDZ, DWORK,
            INFO)
CALL SSPGV (ITYPE, JOBZ, UPLO, N, SA, SB, SW, SZ, LDZ, SWORK,
            INFO)

void dspgv (int itype, char jobz, char uplo, int n, double *da,
            double *db, double *dw, double *dz, int ldz, int *info)
void sspgv (int itype, char jobz, char uplo, int n, float *sa,
            float *sb, float *sw, float *sz, int ldz, int *info)
```

Arguments

ITYPE	Indicates the problem to be solved. Legal values for ITYPE are shown below. Any values not shown below are illegal.
1	$Ax = \lambda Bx$
2	$ABx = \lambda x$
3	$B Ax = \lambda x$
JOBZ	Indicates whether to compute eigenvalues only or to compute both eigenvalues and eigenvectors. The legal values for JOBZ are listed below. Any values not listed below are illegal.
'N' or 'n'	Compute eigenvalues only.
'V' or 'v'	Compute both eigenvalues and eigenvectors.

UPLO	Indicates whether xA and xB contain the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA and xB both contain the upper triangle. 'L' or 'l' xA and xB both contain the lower triangle.
N	Order of the matrix A. $N \geq 0$.
xA	On entry, the upper or lower triangle of the matrix A. The dimension of xA is $(N \times N + N) / 2$. On exit, A is overwritten.
xB	On entry, the upper or lower triangle of the matrix B. The dimension of xB is $(N \times N + N) / 2$. On exit, if UPLO = 'U' or 'u' then B contains the matrix U from the Cholesky factorization UTU of B. If UPLO = 'L' or 'l' then B contains the matrix L from the Cholesky factorization LLT of B.
xW	On exit, W(1:NFOUND) contains the computed eigenvalues in ascending order.
xZ	On exit, if JOBZ = 'V' or 'v' then xZ contains the matrix Z of eigenvectors normalized as shown below: ITYPE Eigenvectors stored in Z 1 or 2 $Z^T \times B \times Z = I$ 3 $Z^T \times B-1 \times Z = I$ If JOBZ = 'N' or 'n' then Z is not used.
LDZ	Leading dimension of the array Z as specified in a dimension or type statement. $LDZ \geq 1$. If JOBZ = 'V' or 'v' then $LDZ \geq \max(1, N)$.
xWORK	Scratch array with a dimension of $3 \times N$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The <i>i</i> th argument, where $i = INFO $, had an illegal value. 0 < INFO ≤ N Convergence failure INFO > N The leading minor of order <i>i</i> of B, where $i = INFO - N$, is not positive definite.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
INTEGER          ITYPE, LDEVEC, LDWORK, LENGTA, LENGTB, N
PARAMETER        ( ITYPE = 1 )
PARAMETER        ( N = 3 )
PARAMETER        ( LDEVEC = N )
PARAMETER        ( LDWORK = 3 * N )

```

```

PARAMETER      (LENGTA = (N * N + N) / 2)
PARAMETER      (LENGTB = (N * N + N) / 2)
C
DOUBLE PRECISION A(LENGTA), B(LENGTB), EVALS(N)
DOUBLE PRECISION EVECS(LDEVEC,N), WORK(LDWORK)
INTEGER         ICOL, INFO, IROW
C
EXTERNAL        DSPGV
INTRINSIC       ABS
C
C      Initialize the array A to store the matrix A shown below.
C      Initialize the array B to store the matrix B shown below.
C
C      12  0  4      6  0  2
C      A =  0  6  0      B =  0  2  0
C      4  0  4      2  0  2
C
DATA A / 1.2D1, 0.0D0, 6.0D0, 4.0D0, 0.0D0, 4.0D0 /
DATA B / 6.0D0, 0.0D0, 2.0D0, 2.0D0, 0.0D0, 2.0D0 /
C
C      Print the initial value of the arrays.
C
PRINT 1000
PRINT 1010, A(1), A(2), A(4)
PRINT 1010, A(2), A(3), A(5)
PRINT 1010, A(4), A(5), A(6)
PRINT 1020
PRINT 1010, B(1), B(2), B(4)
PRINT 1010, B(2), B(3), B(5)
PRINT 1010, B(4), B(5), B(6)
C
CALL DSPGV (ITYPE, 'VALUES AND EIGENVECTORS',
$          'UPPER TRIANGLES OF A AND B STORED', N, A, B,
$          EVALS, EVECS, LDEVEC, WORK, INFO)
IF (INFO .LT. 0) THEN
  PRINT 1030, ABS(INFO)
  STOP 1
ELSE IF (INFO .GT. N) THEN
  PRINT 1040, INFO
  STOP 2
ELSE IF (INFO .GT. 0) THEN
  PRINT 1050, INFO
  STOP 3
END IF
C
C      Print the eigenvalues and eigenvectors.
C
PRINT 1060
DO 140, IROW = 1, N
  PRINT 1070, EVALS(IROW), (EVECS(IROW,ICOL), ICOL = 1, N)
140 CONTINUE
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (3(3X, F4.1))
1020 FORMAT (/1X, 'B:')
1030 FORMAT (/1X, 'Illegal argument to DSPGV, argument #', I2)

```

```

1040 FORMAT (/1X, 'B is not positive definite, INFO = ', I2)
1050 FORMAT (/1X, 'Convergence failure, INFO = ', I2)
1060 FORMAT (/1X, 'Eigenvalue', 6X, 'Eigenvector**T')
1070 FORMAT (1X, F7.2, 6X, '[', 2(F5.3, ', '), F5.3, ']')
C
      END

```

Sample Output

```

A:
 12.0    0.0    4.0
  0.0    6.0    0.0
  4.0    0.0    4.0

B:
  6.0    0.0    2.0
  0.0    2.0    0.0
  2.0    0.0    2.0

Eigenvalue      Eigenvector**T
 2.00    [0.046, -.498, 0.000]
 2.00    [0.000, 0.000, 0.707]
 3.00    [-.750, 0.433, 0.000]

```


Refined Solution to a Linear System in a UDU- or LDL-Factored Symmetric Matrix in Packed Storage

The subroutines described in this section refine the solution to a linear system $AX = B$ for a symmetric matrix A in packed storage, which has been UDU-factored or LDL-factored by xSPTRF, and general matrices B and X . These subroutines also compute forward error bounds and backward error estimates for the refined solution.

Calling Sequence

```
CALL DSPRFS (UPLO, N, NRHS, DA, DAF, IPIVOT, DB, LDB, DX, LDX,
             DFERR, DBERR, DWORK, IWORK2, INFO)
CALL SSPRFS (UPLO, N, NRHS, SA, SAF, IPIVOT, SB, LDB, SX, LDX,
             SFERR, SBERR, SWORK, IWORK2, INFO)
CALL ZSPRFS (UPLO, N, NRHS, ZA, ZAF, IPIVOT, ZB, LDB, ZX, LDX,
             DFERR, DBERR, ZWORK, DWORK2, INFO)
CALL CSPRFS (UPLO, N, NRHS, CA, CAF, IPIVOT, CB, LDB, CX, LDX,
             SFERR, SBERR, CWORK, SWORK2, INFO)

void dsprfs (char uplo, int n, int nrhs, double *da, double *daf,
             int *ipivot, double *db, int ldb, double *dx, int ldx,
             double *dferr, double *dberr, int *info)
void ssprfs (char uplo, int n, int nrhs, float *sa, float *saf, int
             *ipivot, float *sb, int ldb, float *sx, int ldx, float
             *sferr, float *sberr, int *info)
void zsprfs (char uplo, int n, int nrhs, doublecomplex *za,
             doublecomplex *zaf, int *ipivot, doublecomplex *zb,
             int ldb, doublecomplex *zx, int ldx, double *dferr,
             double *dberr, int *info)
void csprfs (char uplo, int n, int nrhs, complex *ca, complex *caf,
             int *ipivot, complex *cb, int ldb, complex *cx, int
             ldx, float *sferr, float *sberr, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.

NRHS	Number of right-hand sides, equal to the number of columns of the matrices B and X. $\text{NRHS} \geq 0$.
xA	Upper or lower triangle of the matrix A. The dimension of xA is $(N \times N + N) / 2$.
xAF	UDU or LDL factorization of the matrix A as computed by xSPTRF. The dimension of xAF is $(N \times N + N) / 2$.
IPIVOT	Pivot indices as computed by xSPTRF.
xB	The $N \times \text{NRHS}$ right-hand side matrix B.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $\text{LDB} \geq \max(1, N)$.
XX	On entry, the $N \times \text{NRHS}$ solution matrix X. On exit, the refined $N \times \text{NRHS}$ solution matrix X.
LDX	Leading dimension of the array X as specified in a dimension or type statement. $\text{LDX} \geq \max(1, N)$.
xFERR	On exit, the estimated forward error bound for each solution vector $X(*, j)$ for $1 \leq j \leq \text{NRHS}$. If X' is the true solution corresponding to $X(*, j)$ then $\text{FERR}(j)$ is an upper bound on the magnitude of the largest element in $X(*, j) - X'$ divided by the magnitude of the largest element in $X(*, j)$.
xBERR	On exit, $\text{BERR}(j)$ is the smallest relative change in any element of A or $B(*, j)$ that makes $X(*, j)$ an exact solution to $AX(*, j) = B(*, j)$ for $1 \leq j \leq \text{NRHS}$.
xWORK	Scratch array with a dimension of $3 \times N$ for real subroutines or $2 \times N$ for complex subroutines.
xWORK2	Scratch array with a dimension of N.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where $i = \text{INFO}$, had an illegal value.

Sample Program

	PROGRAM TEST
	IMPLICIT NONE
C	INTEGER PARAMETER PARAMETER PARAMETER PARAMETER PARAMETER PARAMETER LDB, LDIWRK, LDWORK, LDX, LENGTA, N, NRHS (N = 4) (LDB = N) (LDIWRK = N) (LDWORK = 3 * N) (LDX = LDB) (LENGTA = (N * N + N) / 2) (NRHS = 1)
C	

```

      DOUBLE PRECISION A(LENGTA), AF(LENGTA), B(LDB,NRHS)
      DOUBLE PRECISION BERR(NRHS), EPSLON, FERR(NRHS)
      DOUBLE PRECISION WORK(LDWORK), X(LDX,NRHS)
      INTEGER          ICOL, INFO, IPIVOT(N), IROW, IWORK(LDIWRK)
C
      EXTERNAL          DCOPY, DSPRFS, DSPTRF, DSPTRS
      INTRINSIC         ABS, MAX
C
C      Initialize the array A to store the coefficient matrix A
C      shown below. Initialize the array B to store the right
C      hand side vector b shown below.
C
C      0      0      30
C      A = 0  2  2      b = 50
C           2  2  0      70
C           0  0      110
C
      DATA A / 0.0D0, 0.0D0, 2.0D0, 0.0D0, 2.0D0, 2.0D0,
$           0.0D0, 0.0D0, 0.0D0, 0.0D0 /
      DATA B / 3.0D1, 5.0D1, 7.0D1, 1.1D2 /
C
C      Slightly perturb some of the entries of A. After this
C      code, A will resemble the matrix shown below. Print A
C      after adding epsilon.
C
C      2e      -e
C      A = -e  2+e  2-e
C           2-e  2+e  -e
C           -e  -e
C
      EPSLON = ABS (((2.0D0 / 3.0D0) + 4.0D0) - 4.0D0) -
$ (2.0D0 / 3.0D0)
      A(1) = A(1) + 2.0D0 * EPSLON
      A(2) = A(2) - EPSLON
      A(3) = A(3) + EPSLON
      A(5) = A(5) - EPSLON
      A(6) = A(6) + EPSLON
      A(9) = A(9) - EPSLON
      A(10) = A(10) - EPSLON
      PRINT 1000
      PRINT 1010, A(1), A(2), A(4), A(7)
      PRINT 1010, A(2), A(3), A(5), A(8)
      PRINT 1010, A(4), A(5), A(6), A(9)
      PRINT 1010, A(7), A(8), A(9), A(10)
      CALL DCOPY (LENGTA, A, 1, AF, 1)
C
C      Slightly perturb each element of B. After this loop, B will
C      resemble the matrix shown below. Print B after scaling.
C
C      30-small
C      B = 50-small
C      70-small
C      110-small
C
      DO 110, ICOL = 1, NRHS
        DO 100, IROW = 1, N

```

```

        B(IROW,ICOL) = B(IROW,ICOL) * (1.0D0 - EPSLON)
100  CONTINUE
110  CONTINUE
    CALL DCOPY (LDB * NRHS, B, 1, X, 1)
    PRINT 1020
    PRINT 1030, B
C
C   LDL factor A.
C
    CALL DSPTRF ('UPPER TRIANGLE OF A STORED', N, AF, IPIVOT,
$             INFO)
    IF (INFO .NE. 0) THEN
        PRINT 1040, INFO
        STOP 1
    END IF
C
C   Solve Ax=b and print the solution.
C
    CALL DSPTRS ('UPPER TRIANGLE OF A STORED', N, NRHS, AF,
$             IPIVOT, X, LDX, INFO)
    IF (INFO .NE. 0) THEN
        PRINT 1050, INFO
        STOP 2
    END IF
    PRINT 1060
    PRINT 1070, X
    PRINT 1080
    PRINT 1030,
        A(1) * X(1,1) + A(2) * X(2,1)
    PRINT 1030, A(2) * X(1,1) + A(3) * X(2,1) + A(5) * X(3,1)
    PRINT 1030, A(5) * X(2,1) + A(6) * X(3,1) + A(9) * X(4,1)
    PRINT 1030, A(9) * X(3,1) + A(10) * X(4,1)
C
C   Refine the solution to Ax=b and print the refined solution.
C
    CALL DSPRFS ('UPPER TRIANGLE OF A STORED', N, NRHS, A, AF,
$             IPIVOT, B, LDB, X, LDX, FERR, BERR, WORK,
$             IWORK, INFO)
    IF (INFO .NE. 0) THEN
        PRINT 1090, ABS(INFO)
        STOP 3
    END IF
    PRINT 1100
    PRINT 1070, X
    PRINT 1110
    PRINT 1030,
        A(1) * X(1,1) + A(2) * X(2,1)
    PRINT 1030, A(2) * X(1,1) + A(3) * X(2,1) + A(5) * X(3,1)
    PRINT 1030, A(5) * X(2,1) + A(6) * X(3,1) + A(9) * X(4,1)
    PRINT 1030, A(9) * X(3,1) + A(10) * X(4,1)
    PRINT 1120, FERR(1)
    PRINT 1130, BERR(1)
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (4(2X, F18.16))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (1X, F21.17)
1040 FORMAT (1X, 'Error factoring A, INFO = ', I5)

```

```
1050 FORMAT (1X, 'Error solving Ax=b, INFO = ', I5)
1060 FORMAT (/1X, 'Initial solution to Ax=b:')
1070 FORMAT (1X, E25.17)
1080 FORMAT (/1X, 'Ax with the initial x:')
1090 FORMAT (1X, 'Illegal argument to DSPRFS, INFO = ', I2)
1100 FORMAT (/1X, 'Refined solution to Ax=b:')
1110 FORMAT (/1X, 'Ax with refined x:')
1120 FORMAT (/1X, 'Forward error: ', E14.8)
1130 FORMAT (1X, 'Backward error: ', E14.8)
C
      END
```

Sample Output

```
A:
 0.00000000000000007  -.00000000000000003  0.0000000000000000
0.00000000000000000
-.00000000000000003  2.00000000000000004  1.9999999999999996
0.00000000000000000
 0.00000000000000000  1.9999999999999996  2.0000000000000004  -
.00000000000000003
 0.00000000000000000  0.00000000000000000  -.00000000000000003  -
.00000000000000003

b:
29.99999999999998934
49.99999999999998579
69.99999999999997158
109.99999999999995737

Initial solution to Ax=b:
 0.70596967131753704E+17
 0.51121941716097496E+17
-0.51121941716097480E+17
-0.27914203095773875E+18

Ax with the initial x:
30.00000000000000000
48.00000000000000000
92.97297297297293994
109.99999999999995737

Refined solution to Ax=b:
 0.77650084444450784E+17
 0.65228176341491688E+17
-0.65228176341491680E+17
-0.26503579633234458E+18
```

```
Ax with refined x:  
29.99999999999998579  
48.00000000000000000  
40.27465303140976971  
109.9999999999995737
```

```
Forward error: 0.57571539E+00  
Backward error: 0.90932126E-16
```

Solution to a Linear System in a Symmetric Matrix in Packed Storage (Simple Driver)

The subroutines described in this section solve a linear system $AX = B$ for a symmetric matrix A in packed storage and general matrices B and X . Note that the expert driver `xSPSVX` is also available.

Calling Sequence

```
CALL DSPSV  (UPLO, N, NRHS, DA, IPIVOT, DB, LDB, INFO)
CALL SSPSV  (UPLO, N, NRHS, SA, IPIVOT, SB, LDB, INFO)
CALL ZSPSV  (UPLO, N, NRHS, ZA, IPIVOT, ZB, LDB, INFO)
CALL CSPSV  (UPLO, N, NRHS, CA, IPIVOT, CB, LDB, INFO)

void dspsv  (char uplo, int n, int nrhs, double *da, int *ipivot,
             double *db, int ldb, int *info)
void sspsv  (char uplo, int n, int nrhs, float *sa, int *ipivot,
             float *sb, int ldb, int *info)
void zspsv  (char uplo, int n, int nrhs, doublecomplex *za, int
             *ipivot, doublecomplex *zb, int ldb, int *info)
void cspsv  (char uplo, int n, int nrhs, complex *ca, int *ipivot,
             complex *cb, int ldb, int *info)
```

Arguments

UPLO	Indicates whether <code>xA</code> contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' <code>xA</code> contains the upper triangle. 'L' or 'l' <code>xA</code> contains the lower triangle.
N	Order of the matrix A . $N \geq 0$.
NRHS	Number of right-hand sides, equal to the number of columns of the matrix B . $NRHS \geq 0$.
<code>xA</code>	On entry, the upper or lower triangle of the matrix A . The dimension of <code>xA</code> is $(N \times N + N) / 2$. On exit, the UDU or LDL factorization of the matrix A as computed by <code>xSPTRF</code> .
IPIVOT	On exit, pivot indices as computed by <code>xSPTRF</code> .
<code>xB</code>	On entry, the $N \times NRHS$ right-hand side matrix B . On exit, the $N \times NRHS$ solution matrix X .

LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.	
INFO	On exit:	
	INFO = 0	Subroutine completed normally.
	INFO < 0	The <i>i</i> th argument, where $i = \text{INFO} $, had an illegal value.
	INFO > 0	D(<i>i</i> , <i>i</i>), where $i = \text{INFO}$, is exactly zero and D is therefore singular. The factorization has been completed, but the solution could not be computed.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C   INTEGER          LDA, LDB, N, NRHS
C   PARAMETER        (N = 4)
C   PARAMETER        (LDA = (N * (N + 1)) / 2)
C   PARAMETER        (LDB = N)
C   PARAMETER        (NRHS = 1)
C
C   DOUBLE PRECISION A(LDA), B(LDB,NRHS)
C   INTEGER          INFO, IPIVOT(N)
C
C   EXTERNAL          DSPSV
C
C   Initialize the array A to store in packed symmetric form
C   the 4x4 symmetric positive definite matrix A shown below.
C   Initialize the array B to store the right hand side
C   vector b shown below.
C
C
C      2   -1   0   0           6
C   A = -1   2   -1   0       b = 12
C      0   -1   2   -1         12
C      0   0   -1   2           6
C
C   DATA A /  2.0D0, -1.0D0, 2.0D0, 0.0D0, -1.0D0, 2.0D0,
$          0.0D0, 0.0D0, -1.0D0, 2.0D0 /
C   DATA B / 6.0D0, 1.2D1, 1.2D1, 6.0D0 /
C
C   Print the initial values of the arrays.
C
C   PRINT 1000
C   PRINT 1010, A(1), A(2), A(4), A(7)
C   PRINT 1010, A(2), A(3), A(5), A(8)
C   PRINT 1010, A(4), A(5), A(6), A(9)
C   PRINT 1010, A(7), A(8), A(9), A(10)
C   PRINT 1020
C   PRINT 1030, B
C

```



```
C      Solve Ax=b and print the solution.
C
      CALL DSPSV ('UPPER TRIANGLE OF A STORED', N, NRHS, A,
$         IPIVOT, B, LDB, INFO)
      IF (INFO .NE. 0) THEN
        PRINT 1040, INFO
        STOP 1
      END IF
      PRINT 1050
      PRINT 1030, B
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (4(3X, F6.3))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (1X, F6.3)
1040 FORMAT (1X, 'Error solving Ax=b, INFO = ', I5)
1050 FORMAT (/1X, 'x:')
C
      END
```

Sample Output

```
A:
  2.000  -1.000   0.000   0.000
 -1.000   2.000  -1.000   0.000
  0.000  -1.000   2.000  -1.000
  0.000   0.000  -1.000   2.000

b:
  6.000
 12.000
 12.000
  6.000

x:
 18.000
 30.000
 30.000
 18.000
```

Solution to a Linear System in a Symmetric Matrix in Packed Storage (Expert Driver)

The subroutines described in this section solve a linear system $AX = B$ for a symmetric matrix A in packed storage and general matrices B and X . These subroutines also compute forward error bounds and backward error estimates for the solution, and estimate the condition number of the matrix A . Note that the simple driver `xSPSV` is also available.

Calling Sequence

```
CALL DSPSVX (FACT, UPLO, N, NRHS, DA, DAF, IPIVOT, DB, LDB, DX,
             LDX, DRCOND, DFERR, DBERR, DWORK, IWORK2, INFO)
CALL SSPSVX (FACT, UPLO, N, NRHS, SA, SAF, IPIVOT, SB, LDB, SX,
             LDX, SRCOND, SFERR, SBERR, SWORK, IWORK2, INFO)
CALL ZSPSVX (FACT, UPLO, N, NRHS, ZA, ZAF, IPIVOT, ZB, LDB, ZX,
             LDX, DRCOND, DFERR, DBERR, ZWORK, DWORK2, INFO)
CALL CSPSVX (FACT, UPLO, N, NRHS, CA, CAF, IPIVOT, CB, LDB, CX,
             LDX, SRCOND, SFERR, SBERR, CWORK, SWORK2, INFO)

void dspsvx (char fact, char uplo, int n, int nrhs, double *da,
             double *daf, int *ipivot, double *db, int ldb, double
             *dx, int ldx, double *drcond, double *dferr, double
             *dberr, int *info)
void sspsvx (char fact, char uplo, int n, int nrhs, float *sa,
             float *saf, int *ipivot, float *sb, int ldb, float *sx,
             int ldx, float *srcond, float *sferr, float *sberr, int
             *info)
void zspsvx (char fact, char uplo, int n, int nrhs, doublecomplex
             *za, doublecomplex *zaf, int *ipivot, doublecomplex
             *zb, int ldb, doublecomplex *zx, int ldx, double
             *drcond, double *dferr, double *dberr, int *info)
void cspsvx (char fact, char uplo, int n, int nrhs, complex *ca,
             complex *caf, int *ipivot, complex *cb, int ldb,
             complex *cx, int ldx, float *srcond, float *sferr,
             float *sberr, int *info)
```

Arguments

FACT	Indicates whether or not the factored form of A has been supplied on entry. 'F' or 'f' On entry, AF and IPIVOT contain the factored form of A. A, AF and IPIVOT will not be modified. 'N' or 'n' On entry, the matrix A will be copied to AF and factored.
UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
NRHS	Number of right-hand sides, equal to the number of columns of the matrices B and X. $NRHS \geq 0$.
xA	Upper or lower triangle of the matrix A. The dimension of xA is $(N \times N + N) / 2$.
xAF	The dimension of xAF is $(N \times N + N) / 2$. On entry, if FACT = 'F' or 'f', then AF contains the UDU or LDL factorization of A, as computed by xSPTRF. Otherwise the entry value of AF is not used. On exit, if FACT = 'N' or 'n', then AF contains the UDU or LDL factorization of A, as computed by xSPTRF. Otherwise AF is unchanged.
IPIVOT	On entry, if FACT = 'F' or 'f', then IPIVOT contains the pivot indices as computed by xSPTRF. Otherwise the entry values of IPIVOT are not used. On exit, if FACT = 'N' or 'n', then IPIVOT contains the pivot indices as computed by xSPTRF. Otherwise IPIVOT is unchanged.
xB	The $N \times NRHS$ right-hand side matrix B.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
xX	On exit, the $N \times NRHS$ solution matrix X.
LDX	Leading dimension of the array X as specified in a dimension or type statement. $LDX \geq \max(1, N)$.

xRCOND	On exit, an estimate of the reciprocal condition number of the matrix A, where the reciprocal condition number is defined to be $1 / (\ A\ \times \ A^{-1}\)$. The reciprocal of the condition number is estimated instead of the condition number itself to avoid overflow or division by zero. If RCOND is less than machine precision (in particular, if RCOND = 0) then A is singular to working precision. In this case, INFO > 0 is returned and the solution and error bounds are not computed.
xFERR	On exit, the estimated forward error bound for each solution vector $X(*, j)$ for $1 \leq j \leq \text{NRHS}$. If X' is the true solution corresponding to $X(*, j)$ then FERR(j) is an upper bound on the magnitude of the largest element in $X(*, j) - X'$ divided by the magnitude of the largest element in $X(*, j)$.
xBERR	On exit, BERR(j) is the smallest relative change in any element of A or $B(*, j)$ that makes $X(*, j)$ an exact solution to $AX(*, j) = B(*, j)$ for $1 \leq j \leq \text{NRHS}$.
xWORK	Scratch array with a dimension of $3 \times N$ for real subroutines or $2 \times N$ for complex subroutines.
xWORK2	Scratch array with a dimension of N.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The <i>i</i> th argument, where $i = \text{INFO} $, had an illegal value. $1 \leq \text{INFO} \leq N$ D(i,i), where $i = \text{INFO}$, is exactly zero and D is therefore singular. The factorization has been completed, but the solution and error bounds could not be computed. INFO = N + 1 The block diagonal matrix D is nonsingular, but RCOND is less than machine precision. The factorization has been completed, but the matrix is singular to working precision, so the solution and error bounds could not be computed.

Sample Program

```

C      PROGRAM TEST
C      IMPLICIT NONE

C      INTEGER          LDA, LDAF, LDB, LDIWRK, LDWORK, LDX
C      INTEGER          N, NRHS
C      PARAMETER        (N = 4)
C      PARAMETER        (LDA = (N * (N + 1)) / 2)
C      PARAMETER        (LDAF = LDA)

```

```

C      PARAMETER      (LDB = N)
C      PARAMETER      (LDIWRK = N)
C      PARAMETER      (LDX = N)
C      PARAMETER      (LDWORK = 3 * N)
C      PARAMETER      (NRHS = 1)

C      DOUBLE PRECISION A(LDA), AF(LDAF), B(LDB,NRHS), BERR(NRHS)
C      DOUBLE PRECISION FERR(NRHS), RCOND, WORK(LDWORK)
C      DOUBLE PRECISION X(LDX,NRHS)
C      INTEGER          I, INFO, IPIVOT(N), IWORK(LDIWRK)

C      EXTERNAL          DSPSVX

C      Initialize the array A to store in packed symmetric form
C      the 4x4 symmetric matrix A shown below. Initialize the
C      array B to store the right hand side vector b shown below.
C
C      A =
C      2  -1  0  0      b = 6
C      -1  2  -1  0      12
C      0  -1  2  -1      12
C      0  0  -1  2       6
C
C      DATA A / 2.0D0, -1.0D0, 2.0D0, 0.0D0, -1.0D0, 2.0D0,
$              0.0D0, 0.0D0, -1.0D0, 2.0D0 /
C      DATA B / 6.0D0, 1.2D1, 1.2D1, 6.0D0 /

C      Print the initial values of the arrays.
C
C      PRINT 1000
C      PRINT 1010, A(1), A(2), A(4), A(7)
C      PRINT 1010, A(2), A(3), A(5), A(8)
C      PRINT 1010, A(4), A(5), A(6), A(9)
C      PRINT 1010, A(7), A(8), A(9), A(10)
C      PRINT 1020
C      PRINT 1030, B

C      Solve Ax=b and print the solution.
C
C      CALL DSPSVX ('NOT FACTORED A', 'UPPER TRIANGLE OF A STORED',
$              N, NRHS, A, AF, IPIVOT, B, LDB, X, LDX, RCOND,
$              FERR, BERR, WORK, IWORK, INFO)
C      IF (INFO .NE. 0) THEN
C          PRINT 1040, INFO
C          IF (INFO .EQ. (N + 1)) THEN
C              PRINT 1050
C              STOP 1
C          END IF
C          STOP 2
C      END IF
C      PRINT 1060
C      PRINT 1030, X
C      PRINT 1070, 1.0D0 / RCOND
C      PRINT 1080, (I, BERR(I), I = 1, NRHS)
C      PRINT 1090, (I, FERR(I), I = 1, NRHS)

C      1000 FORMAT (1X, 'A:')

```

```

1010 FORMAT (4(3X, F6.3))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (1X, F6.2)
1040 FORMAT (1X, 'Error solving Ax=b, INFO = ', I5)
1050 FORMAT (1X, 'Matrix is singular to working precision.')
1060 FORMAT (/1X, 'x:')
1070 FORMAT (/1X, 'Estimated condition number of A: ', F7.2)
1080 FORMAT (/1X, 'Backward error for system #', I1, ': ', E12.6)
1090 FORMAT (1X, 'Forward error for system #', I1, ': ', E12.6)
C
      END

```

Sample Output

```

A:
  2.000  -1.000   0.000   0.000
 -1.000   2.000  -1.000   0.000
  0.000  -1.000   2.000  -1.000
  0.000   0.000  -1.000   2.000

b:
  6.00
 12.00
 12.00
  6.00

x:
 18.00
 30.00
 30.00
 18.00

Estimated condition number of A:  12.00

Backward error for system #1: 0.296059E-16
Forward error for system #1:  0.601001E-14

```

UDU or LDL Factorization of a Symmetric Matrix in Packed Storage

The subroutines described in this section compute a UDU or LDL factorization of a symmetric matrix A in packed storage. It is typical to follow a call to xSPTRF with a call to xSPSVX or xSPTRS to solve a linear system $AX = B$, to xSPCON to estimate the condition number of A , or to xSPTRI to compute A^{-1} .

Calling Sequence

```
CALL DSPTRF (UPLO, N, DA, IPIVOT, INFO)
CALL SSPTRF (UPLO, N, SA, IPIVOT, INFO)
CALL ZSPTRF (UPLO, N, ZA, IPIVOT, INFO)
CALL CSPTRF (UPLO, N, CA, IPIVOT, INFO)
```

```
void dsptrf (char uplo, int n, double *da, int *ipivot, int *info)
void ssptrf (char uplo, int n, float *sa, int *ipivot, int *info)
void zsptrf (char uplo, int n, doublecomplex *za, int *ipivot, int
            *info)
void csptrf (char uplo, int n, complex *ca, int *ipivot, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
xA	On entry, the upper or lower triangle of the matrix A. The dimension of xA is $(N \times N + N) / 2$. On exit, a UDU or LDL factorization of the matrix A.
IPIVOT	On exit, an N-element array containing details of the row interchanges and block structure of D.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The i th argument, where $i = INFO $, had an illegal value. INFO > 0 D(i,i), where $i = INFO$, is exactly zero, and D is therefore singular. The factorization is complete, but division by zero will occur if this factorization is used to solve a linear system.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER          LDA, LDB, N, NRHS
  PARAMETER        (N = 4)
  PARAMETER        (NRHS = 1)
  PARAMETER        (LDA = (N * (N + 1)) / 2)
  PARAMETER        (LDB = N)

C
  DOUBLE PRECISION A(LDA), B(LDB,NRHS)
  INTEGER          INFO, IPIVOT(N)

C
  EXTERNAL          DSPTRF, DSPTRS
  INTRINSIC         ABS

C
C Initialize the array A to store in packed symmetric form the
C 4x4 symmetric coefficient matrix A shown below. Initialize
C array B to store the right hand side vector b shown below.
C
C      2   -1   0   0
C  A = -1   2   -1  0   b = 12
C      0  -1   2  -1   12
C      0   0  -1   2    6
C
C  DATA A /  2.0D0, -1.0D0, 2.0D0, 0.0D0, -1.0D0, 2.0D0,
$           0.0D0, 0.0D0, -1.0D0, 2.0D0 /
C  DATA B / 6.0D0, 1.2D1, 1.2D1, 6.0D0 /

C
C Print the initial values of the arrays.
C
C  PRINT 1000
C  PRINT 1010, A(1), A(2), A(4), A(7)
C  PRINT 1010, A(2), A(3), A(5), A(8)
C  PRINT 1010, A(4), A(5), A(6), A(9)
C  PRINT 1010, A(7), A(8), A(9), A(10)
C  PRINT 1020
C  PRINT 1030, B

C
C LDL factor A.
C
C  CALL DSPTRF ('UPPER TRIANGLE OF A STORED', N, A, IPIVOT,
$             INFO)
C  IF (INFO .NE. 0) THEN
C    PRINT 1040, INFO
C    STOP 1
C  END IF

C
C Use the factored form of A to solve Ax=b then print
C the result.
C
C  CALL DSPTRS ('UPPER TRIANGLE OF A STORED', N, NRHS, A,
$             IPIVOT, B, LDB, INFO)

```



```
        IF (INFO .NE. 0) THEN
          PRINT 1050, ABS(INFO)
          STOP 2
        END IF
        PRINT 1060
        PRINT 1030, B
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (4(3X, F6.3))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (1X, F6.2)
1040 FORMAT (1X, 'Error factoring A, INFO = ', I5)
1050 FORMAT (1X, 'Illegal argument to DSPTRS, argument #', I1)
1060 FORMAT (/1X, 'x:')
C
        END
```

Sample Output

```
A:
 2.000  -1.000   0.000   0.000
-1.000   2.000  -1.000   0.000
 0.000  -1.000   2.000  -1.000
 0.000   0.000  -1.000   2.000

b:
 6.00
12.00
12.00
 6.00

x:
18.00
30.00
30.00
18.00
```

Inverse of a UDU- or LDL-Factored Symmetric Matrix in Packed Storage

The subroutines described in this section compute the inverse of a symmetric matrix A in packed storage, which has been UDU-factored or LDL-factored by xSPTRF.

Calling Sequence

```
CALL DSPTRI (UPLO, N, DA, IPIVOT, DWORK, INFO)
CALL SSPTRI (UPLO, N, SA, IPIVOT, SWORK, INFO)
CALL ZSPTRI (UPLO, N, ZA, IPIVOT, ZWORK, INFO)
CALL CSPTRI (UPLO, N, CA, IPIVOT, CWORK, INFO)
```

```
void dsptri (char uplo, int n, double *da, int *ipivot, int *info)
void ssptri (char uplo, int n, float *sa, int *ipivot, int *info)
void zsptri (char uplo, int n, doublecomplex *za, int *ipivot, int
            *info)
void csptri (char uplo, int n, complex *ca, int *ipivot, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
xA	On entry, the UDU or LDL factorization of the matrix A as computed by xSPTRF. The dimension of xA is $(N \times N + N) / 2$. On exit, the symmetric inverse of the matrix A.
IPIVOT	Pivot indices as computed by xSPTRF.
xWORK	Scratch array with a dimension of N.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The <i>i</i> th argument, where $i = INFO $, had an illegal value. INFO > 0 D(<i>i</i> , <i>i</i>), where $i = INFO$, is zero. The matrix is therefore singular and its inverse could not be computed.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER          LDA, N
  PARAMETER        (N = 4)
  PARAMETER        (LDA = (N * (N + 1)) / 2)
C
  DOUBLE PRECISION A(LDA), WORK(N)
  INTEGER          INFO, IPIVOT(N)
C
  EXTERNAL          DSPTRF, DSPTRI
  INTRINSIC         ABS
C
C Initialize the array A to store in symmetric packed form the
C 4x4 symmetric matrix A shown below.
C
C      2  1  0  0
C  A =  1  2  1  0
C      0  1  2  1
C      0  0  1  1
C
C DATA A / 2.0D0, 1.0D0, 2.0D0, 0.0D0, 1.0D0, 2.0D0,
$          0.0D0, 0.0D0, 1.0D0, 1.0D0 /
C
C Print the initial values of the arrays.
C
C PRINT 1000
C PRINT 1010, A(1), A(2), A(4), A(7)
C PRINT 1010, A(2), A(3), A(5), A(8)
C PRINT 1010, A(4), A(5), A(6), A(9)
C PRINT 1010, A(7), A(8), A(9), A(10)
C
C LDL factor A.
C
C CALL DSPTRF ('UPPER TRIANGLE OF A STORED', N, A, IPIVOT,
$            INFO)
C IF (INFO .NE. 0) THEN
C   PRINT 1020, INFO
C   STOP 1
C END IF
C
C Use the factored form of A to compute the inverse of A and
C print the inverse thus computed.
C
C CALL DSPTRI ('UPPER TRIANGLE OF A STORED', N, A, IPIVOT,
$            WORK, INFO)
C IF (INFO .NE. 0) THEN
C   PRINT 1030, ABS(INFO)
C   STOP 2
C END IF
C PRINT 1040
C PRINT 1010, A(1), A(2), A(4), A(7)

```

```

      PRINT 1010, A(2), A(3), A(5), A(8)
      PRINT 1010, A(4), A(5), A(6), A(9)
      PRINT 1010, A(7), A(8), A(9), A(10)
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (4(3X, F6.3))
1020 FORMAT (1X, 'Error factoring A, INFO = ', I5)
1030 FORMAT (1X, 'Error computing inverse of A, INFO = ', I5)
1040 FORMAT (/1X, 'A**(-1):')
C
      END

```

Sample Output

```

A:
  2.000    1.000    0.000    0.000
  1.000    2.000    1.000    0.000
  0.000    1.000    2.000    1.000
  0.000    0.000    1.000    1.000

A**(-1):
  1.000   -1.000    1.000   -1.000
 -1.000    2.000   -2.000    2.000
  1.000   -2.000    3.000   -3.000
 -1.000    2.000   -3.000    4.000

```

Solution to a Linear System in a UDU- or LDL-Factored Symmetric Matrix in Packed Storage

The subroutines described in this section solve a linear system $AX = B$ for a symmetric matrix A in packed storage, which has been UDU-factored or LDL-factored by xSPTRF, and general matrices B and X .

Calling Sequence

```
CALL DSPTRS (UPLO, N, NRHS, DA, IPIVOT, DB, LDB, INFO)
CALL SSPTRS (UPLO, N, NRHS, SA, IPIVOT, SB, LDB, INFO)
CALL ZSPTRS (UPLO, N, NRHS, ZA, IPIVOT, ZB, LDB, INFO)
CALL CSPTRS (UPLO, N, NRHS, CA, IPIVOT, CB, LDB, INFO)

void dsptrs (char uplo, int n, int nrhs, double *da, int *ipivot,
             double *db, int ldb, int *info)
void sspters (char uplo, int n, int nrhs, float *sa, int *ipivot,
              float *sb, int ldb, int *info)
void zsptrs (char uplo, int n, int nrhs, doublecomplex *za, int
             *ipivot, doublecomplex *zb, int ldb, int *info)
void cspters (char uplo, int n, int nrhs, complex *ca, int *ipivot,
              complex *cb, int ldb, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
NRHS	Number of right-hand sides, equal to the number of columns of the matrix B. $NRHS \geq 0$.
xA	UDU or LDL factorization of the matrix A, as computed by xSPTRF. The dimension of xA is $(N \times N + N) / 2$.
IPIVOT	Pivot indices as computed by xSPTRF.
xB	On entry, the $N \times NRHS$ right-hand side matrix B. On exit, the $N \times NRHS$ solution matrix X.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.

INFO	On exit:	
	INFO = 0	Subroutine completed normally.
	INFO < 0	The ith argument, where i = INFO , had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C   INTEGER          LDA, LDB, N, NRHS
C   PARAMETER        (N = 4)
C   PARAMETER        (NRHS = 1)
C   PARAMETER        (LDA = (N * (N + 1)) / 2)
C   PARAMETER        (LDB = N)
C
C   DOUBLE PRECISION A(LDA), B(LDB,NRHS)
C   INTEGER          INFO, IPIVOT(N)
C
C   EXTERNAL          DSPTRF, DSPTRS
C   INTRINSIC         ABS
C
C   Initialize the array A to store in packed symmetric form the
C   4x4 symmetric coefficient matrix A shown below. Initialize
C   the array B to store the right hand side vector b shown
C   below.
C
C   A =  2  -1  0  0      b =  6
C   -1  2  -1  0      12
C   0  -1  2  -1      12
C   0  0  -1  2      6
C
C   DATA A /  2.0D0, -1.0D0, 2.0D0, 0.0D0, -1.0D0, 2.0D0,
$           0.0D0, 0.0D0, -1.0D0, 2.0D0 /
C   DATA B / 6.0D0, 1.2D1, 1.2D1, 6.0D0 /
C
C   Print the initial values of the arrays.
C
C   PRINT 1000
C   PRINT 1010, A(1), A(2), A(4), A(7)
C   PRINT 1010, A(2), A(3), A(5), A(8)
C   PRINT 1010, A(4), A(5), A(6), A(9)
C   PRINT 1010, A(7), A(8), A(9), A(10)
C   PRINT 1020
C   PRINT 1030, B
C
C   LDL factor A.
C
C   CALL DSPTRF ('UPPER TRIANGLE OF A STORED', N, A, IPIVOT,
$             INFO)
C   IF (INFO .NE. 0) THEN

```

```
        PRINT 1040, INFO
        STOP 2
      END IF
C
C      Use the factored form of A to solve Ax=b then print
C      the result.
C
      CALL DSPTRS ('UPPER TRIANGLE OF A STORED', N, NRHS, A,
$              IPIVOT, B, LDB, INFO)
      IF (INFO .NE. 0) THEN
        PRINT 1050, ABS(INFO)
        STOP 1
      END IF
      PRINT 1060
      PRINT 1030, B
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (4(3X, F6.3))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (1X, F6.2)
1040 FORMAT (1X, 'Error factoring A, INFO = ', I5)
1050 FORMAT (1X, 'Illegal argument to DSPTRS, argument #', I1)
1060 FORMAT (/1X, 'x:')
C
      END
```

Sample Output

```
A:
 2.000  -1.000   0.000   0.000
-1.000   2.000  -1.000   0.000
 0.000  -1.000   2.000  -1.000
 0.000   0.000  -1.000   2.000

b:
 6.00
12.00
12.00
 6.00

x:
18.00
30.00
30.00
18.00
```

Eigenvalues and Eigenvectors of a Symmetric Tridiagonal Matrix (Simple Driver)

The subroutines described in this section compute the eigenvalues and, optionally, the eigenvectors of a symmetric tridiagonal matrix A. The eigenvectors are normalized to have a Euclidean norm of 1. Note that the expert driver xSTEVX is also available.

Calling Sequence

```
CALL DSTEV  (JOBZ, N, DDIAG, DOFFD, DZ, LDZ, DWORK, INFO)
CALL SSTEVE (JOBZ, N, SDIAG, SOFFD, SZ, LDZ, SWORK, INFO)

void dstev  (char jobz, int n, double *ddiag, double *doffd, double
             *dz, int ldz, int *info)
void ssteve (char jobz, int n, float *sdiag, float *soffd, float
             *sz, int ldz, int *info)
```

Arguments

JOBZ	Indicates whether to compute eigenvalues only or to compute both eigenvalues and eigenvectors. The legal values for JOBZ are listed below. Any values not listed below are illegal. 'N' or 'n' Compute eigenvalues only. 'V' or 'v' Compute both eigenvalues and eigenvectors.
N	Order of the matrix A. $N \geq 0$.
xDIAG	On entry, the N diagonal elements of A. On exit, the eigenvalues in ascending order.
xOFFD	On entry, the N-1 off-diagonal elements of A, stored in elements 1 through N-1 of this N-element array. OFFD(N) does not need to be set. On exit, OFFD is overwritten.
xZ	On exit, if JOBZ = 'V' or 'v' then Z contains the orthonormal eigenvectors. If JOBZ = 'N' or 'n' then Z is not used.
LDZ	Leading dimension of the array Z as specified in a dimension or type statement. $LDZ \geq 1$. If JOBZ = 'V' or 'v' then $LDZ \geq \max(1, N)$.
xWORK	Scratch array with a dimension of $\max(1, 2 \times N - 2)$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where $i = \text{INFO} $, had an illegal value. INFO > 0 Convergence failure.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER          LDEVEC, LDWORK, N
  PARAMETER        (N = 4)
  PARAMETER        (LDEVEC = N)
  PARAMETER        (LDWORK = 2 * N - 2)

C
  DOUBLE PRECISION DIAG(N), EVECS(LDEVEC,N), OFF(N), PIDIV4
  DOUBLE PRECISION WORK(LDWORK)
  INTEGER          ICOL, INFO, IROW

C
  EXTERNAL          DSTEVE
  INTRINSIC         ABS, ATAN, COS, MIN, SIN

C
  Initialize the array A to store the symmetric tridiagonal
  matrix A shown below.
C
C
C
C
C
C
C
  A =
  1      0      0      0
  0  cos(pi/4) sin(pi/4)  0
  0  sin(pi/4) -cos(pi/4)  0
  0      0      0      1

C
  PIDIV4 = ATAN(1.0D0)
  DIAG(1) = 1.0D0
  DIAG(2) = COS(PIDIV4)
  DIAG(3) = -COS(PIDIV4)
  DIAG(4) = 1.0D0
  OFF(1) = 0.0D0
  OFF(2) = SIN(PIDIV4)
  OFF(3) = 0.0D0

C
  Print the initial value of A.
C
C
  PRINT 1000
  DO 100, IROW = 1, N
    PRINT 1010, (0.0D0, ICOL = 1, IROW - 2),
$      (OFF(IROW - 1), ICOL = ABS(IROW - 2), IROW - 2),
$      DIAG(IROW),
$      (OFF(IROW), ICOL = 1, MIN(1, N - IROW)),
$      (0.0D0, ICOL = IROW + 2, N)
100 CONTINUE

C
  Compute the eigenvalues and right eigenvectors of A.
C
C
  CALL DSTEVE ('VECTORS AND VALUES', N, DIAG, OFF, EVECS,
$            LDEVEC, WORK, INFO)
  IF (INFO .NE. 0) THEN
    IF (INFO .LT. 0) THEN
      PRINT 1020, ABS(INFO)
      STOP 1
    ELSE

```

```

        PRINT 1030, INFO
        STOP 2
      END IF
    END IF
C
    Print the eigenvalues and eigenvectors.
C
    PRINT 1040
    DO 120, ICOL = 1, N
      PRINT 1050, DIAG(ICOL), (EVECS(IROW,ICOL), IROW = 1, N)
    120 CONTINUE
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (4(3X, F9.6))
1020 FORMAT (/1X, 'Illegal argument to DSTEV, argument #', I2)
1030 FORMAT (/1X, 'QR failed to converge, INFO = ', I2)
1040 FORMAT (/1X, 'Eigenvalue', 4X, 'Eigenvector**T')
1050 FORMAT (1X, F5.2, 10X, '[', 3(F5.3, ', '), F5.3, ']')
C
    END

```

Sample Output

```

A:
  1.000000    0.000000    0.000000    0.000000
  0.000000    0.707107    0.707107    0.000000
  0.000000    0.707107   -0.707107    0.000000
  0.000000    0.000000    0.000000    1.000000

Eigenvalue  Eigenvector**T
-1.00      [0.000, 0.383, -.924, 0.000]
 1.00      [0.000, -.924, -.383, 0.000]
 1.00      [1.000, 0.000, 0.000, 0.000]
 1.00      [0.000, 0.000, 0.000, 1.000]

```

Eigenvalues and Eigenvectors of a Symmetric Tridiagonal Matrix (Expert Driver)

The subroutines described in this section compute the eigenvalues and, optionally, the eigenvectors of a symmetric tridiagonal matrix A . The eigenvectors are normalized to have a Euclidean norm of 1. Eigenvalues and eigenvectors can be selected by specifying either a range of values or a range of indices for the desired eigenvalues. Note that the simple driver `xSTEV` is also available.

Calling Sequence

```
CALL DSTEVX (JOBZ, RANGE, N, DDIAG, DOFFD, DVL, DVU, IL, IU,
             DABTOL, NFOUND, DW, DZ, LDZ, DWORK, IWORK2, IFAIL,
             INFO)
CALL SSTEVD (JOBZ, RANGE, N, SDIAG, SOFFD, SVL, SVU, IL, IU,
             SABTOL, NFOUND, SW, SZ, LDZ, SWORK, IWORK2, IFAIL,
             INFO)

void dstevx (char jobz, char range, int n, double *ddiag, double
             *doffd, double dvl, double dvu, int il, int iu, double
             dabtol, int *nfound, double *dw, double *dz, int ldz,
             int *ifail, int *info)
void sstevd (char jobz, char range, int n, float *sdiag, float
             *soffd, float svl, float svu, int il, int iu, float
             sabtol, int *nfound, float *sw, float *sz, int ldz, int
             *ifail, int *info)
```

Arguments

JOBZ	Indicates whether to compute eigenvalues only or to compute both eigenvalues and eigenvectors. The legal values for JOBZ are listed below. Any values not listed below are illegal. 'N' or 'n' Compute eigenvalues only. 'V' or 'v' Compute both eigenvalues and eigenvectors.
RANGE	Indicates whether to compute all eigenvalues or some subset. Legal values for RANGE are shown below. Any values not shown below are illegal. 'A' or 'a' All eigenvalues will be computed. 'I' or 'i' The ILth through IUth eigenvalues will be computed; see the descriptions of the IL and IU arguments below.

	'V' or 'v'	All eigenvalues in the half-open interval (VL,VU] will be computed; see the descriptions of the VL and VU arguments below.
N		Order of the matrix A. $N \geq 0$.
xDIAG		On entry, the N diagonal elements of A. On exit, DIAG may be multiplied by a constant factor chosen to avoid overflow or underflow in computing the eigenvalues.
xOFFD		On entry, the N-1 off-diagonal elements of A stored in elements 1 through N-1 of this N-element array. OFFD(N) does not need to be set. On exit, OFFD may be multiplied by a constant factor chosen to avoid overflow or underflow in computing the eigenvalues.
xVL, xVU		If RANGE = 'V' or 'v' then only eigenvalues in the half-open interval (VL,VU] will be computed. Otherwise not used.
IL, IU		If RANGE = 'T' or 't' then only the ILth through IUth eigenvalues will be computed. Otherwise not used. $1 \leq IL \leq IU \leq N$.
xABTOL		The absolute error tolerance for the eigenvalues. An approximate eigenvalue is considered to have converged when it lies in an interval [a,b] of width less than or equal to $ABTOL + \epsilon \times \max(a , b)$ where ϵ is machine precision. If $ABTOL \leq 0$ then $\epsilon \times \ T\ _1$ will be used in its place, where $\ T\ _1$ is the 1-norm of the tridiagonal matrix obtained by reducing A to tridiagonal form.
NFOUND		On exit, the total number of eigenvalues found. $0 \leq NFOUND \leq N$. If RANGE = 'A' or 'a' then $NFOUND = N$. If RANGE = 'T' or 't' then $NFOUND = IU - IL + 1$.
xW		On exit, W(1:NFOUND) contains the computed eigenvalues in ascending order.
xZ		On exit, if JOBZ = 'V' or 'v' then the first NFOUND columns of Z contain the orthonormal eigenvectors corresponding to the selected eigenvalues. If an eigenvector fails to converge then the corresponding column of Z contains an approximation to that eigenvector and its index is returned in IFAIL. If JOBZ = 'N' or 'n' then Z is not used. Note that Z must have at least NFOUND columns. If RANGE = 'V' or 'v' then NFOUND may not be known in advance and an upper bound must be used. For A(LDZ,K) where $NFOUND > K$, the eigenvectors will be stored into memory with unpredictable results.
LDZ		Leading dimension of the array Z as specified in a dimension or type statement. $LDZ \geq 1$. If JOBZ = 'V' or 'v' then $LDZ \geq \max(1, N)$.
xWORK		Scratch array with a dimension of $5 \times N$.
IWORK2		Scratch array with a dimension of $5 \times N$.


```

      DO 100, IROW = 1, N
      PRINT 1010, (0.0D0, ICOL = 1, IROW - 2),
$      (OFF(IROW - 1), ICOL = ABS(IROW - 2), IROW - 2),
$      DIAG(IROW),
$      (OFF(IROW), ICOL = 1, MIN(1, N - IROW)),
$      (0.0D0, ICOL = IROW + 2, N)
100 CONTINUE
C
C      Compute the eigenvalues and right eigenvectors of A.
C
      CALL DSTEVX ('VECTORS AND VALUES', 'ALL EIGENVALUES', N,
$      DIAG, OFF, TEMP, TEMP, ITEMP, ITEMP, 0.0D0,
$      NFOUND, EVALS, EVECS, LDEVEC, WORK, IWORK,
$      IFAIL, INFO)
      IF (INFO .NE. 0) THEN
        IF (INFO .LT. 0) THEN
          PRINT 1020, ABS(INFO)
          STOP 1
        ELSE
          PRINT 1030, INFO
          STOP 2
        END IF
      END IF
C
C      Print the eigenvalues and eigenvectors.
C
      PRINT 1040
      DO 120, ICOL = 1, N
        PRINT 1050, DIAG(ICOL), (EVECS(IROW,ICOL), IROW = 1, N)
      120 CONTINUE
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (4(3X, F9.6))
1020 FORMAT (/1X, 'Illegal argument to DSTEVX, argument #', I2)
1030 FORMAT (/1X, 'QR failed to converge, INFO = ', I2)
1040 FORMAT (/1X, 'Eigenvalue', 10X, 'Eigenvector**T')
1050 FORMAT (1X, F7.2, 6X, '[', 3(F5.3, ' ', ' '), F5.3, ' ]')
C
      END

```

Sample Output

```

A:
1.000000    0.000000    0.000000    0.000000
0.000000    0.707107    0.707107    0.000000
0.000000    0.707107   -0.707107    0.000000
0.000000    0.000000    0.000000    1.000000

```

Eigenvalue	Eigenvector**T
1.00	[0.000, 0.383, -.924, 0.000]
0.71	[0.000, -.924, -.383, 0.000]
-0.71	[1.000, 0.000, 0.000, 0.000]
1.00	[0.000, 0.000, 0.000, 1.000]

Reciprocal Condition Number of a UDU- or LDL-Factored Symmetric Matrix

The subroutines described in this section estimate the reciprocal condition number of a symmetric matrix A, which has been UDU-factored or LDL-factored by xSYTRF.

Calling Sequence

```
CALL DSYCON (UPLO, N, DA, LDA, IPIVOT, DANORM, DRCOND, DWORK,
            IWORK2, INFO)
CALL SSYCON (UPLO, N, SA, LDA, IPIVOT, SANORM, SRCOND, SWORK,
            IWORK2, INFO)
CALL ZSYCON (UPLO, N, ZA, LDA, IPIVOT, DANORM, DRCOND, ZWORK, INFO)
CALL CSYCON (UPLO, N, CA, LDA, IPIVOT, SANORM, SRCOND, CWORK, INFO)

void dsycon (char uplo, int n, double *da, int lda, int *ipivot,
            double danorm, double *drcond, int *info)
void ssycon (char uplo, int n, float *sa, int lda, int *ipivot,
            float sanorm, float *srcond, int *info)
void zsycon (char uplo, int n, doublecomplex *za, int lda, int
            *ipivot, double danorm, double *drcond, int *info)
void csycon (char uplo, int n, complex *ca, int lda, int *ipivot,
            float sanorm, float *srcond, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
xA	UDU or LDL factorization of the matrix A as computed by xSYTRF.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
IPIVOT	Pivot indices as computed by xSYTRF.
xANORM	The 1-norm or ∞ -norm of matrix A.

xRCOND	On exit, the estimated reciprocal of the condition number of the matrix A where the reciprocal condition number of A is defined to be $1 / (\text{ANORM} \times \ A^{-1}\)$. The reciprocal of the condition number is estimated instead of the condition number itself to avoid overflow or division by zero. If RCOND is less than machine precision (in particular, if RCOND = 0) then A is singular to working precision.
xWORK	Scratch array with a dimension of $2 \times N$.
IWORK2	Scratch array with a dimension of N for real subroutines.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where i = INFO , had an illegal value.

Sample Program

```

      PROGRAM TEST
      IMPLICIT NONE
C
      INTEGER          LDA, LDIWRK, LDWORK, N
      PARAMETER        (N = 3)
      PARAMETER        (LDA = N)
      PARAMETER        (LDIWRK = N)
      PARAMETER        (LDWORK = 2 * N)
C
      DOUBLE PRECISION A(LDA,N), ANORM, RCOND, WORK(LDWORK)
      INTEGER          ICOL, INFO, IROW, IPIVOT(N), IWORK(LDIWRK)
C
      EXTERNAL          DSYCON, DSYTRF
C
C      Initialize the array A to store the coefficient matrix A
C      shown below.
C
C      1  2  2
C      A = 2  1  2
C      2  2  1
C
      DATA A / 1.0D0, 2*8D8, 2.0D0, 1.0D0, 8D8, 2.0D0, 2.0D0,
$             1.0D0 /
C
      PRINT 1000
      DO 100, IROW = 1, N
        PRINT 1010, (0.0D0, ICOL = 1, IROW - 1),
$              (A(IROW, ICOL), ICOL = IROW, N)
100  CONTINUE
      PRINT 1020
      DO 110, IROW = 1, LDA
        PRINT 1010, (A(IROW,ICOL), ICOL = 1, N)
110  CONTINUE
C

```

```

      CALL DSYTRF ('UPPER TRIANGULAR A', N, A, LDA, IPIVOT,
$          WORK, LDWORK, INFO)
      IF (INFO .EQ. 0) THEN
          ANORM = 5.0D0
          CALL DSYCON ('UPPER TRIANGULAR A', N, A, LDA, IPIVOT,
$          ANORM, RCOND, WORK, IWORK, INFO)
          IF (INFO .EQ. 0) THEN
              PRINT 1030, 1.0D0 / RCOND
          ELSE
              PRINT 1040, INFO
          END IF
      ELSE
          PRINT 1050, INFO
      END IF
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (3(3X, F4.1))
1020 FORMAT (/1X, 'A in symmetric form: (* in unused elements)')
1030 FORMAT (/1X, 'Estimated condition number: ', F6.2)
1040 FORMAT (1X, 'Error computing condition number, INFO = ', I6)
1050 FORMAT (1X, 'Factorization failed with INFO = ', I6)
C
      END

```

Sample Output

```

A in full form:
  1.0    2.0    2.0
  0.0    1.0    2.0
  0.0    0.0    1.0

A in symmetric form: (* in unused elements)
  1.0    2.0    2.0
****    1.0    2.0
****    ****    1.0

Estimated condition number:   7.00

```

Eigenvalues and Eigenvectors of a Symmetric Matrix (Simple Driver)

The subroutines described in this section compute the eigenvalues and, optionally, the eigenvectors of a symmetric matrix A. The eigenvectors are normalized to have a Euclidean norm of 1. Note that the expert driver xSYEVX is also available.

Calling Sequence

```
CALL DSYEV  (JOBZ, UPLO, N, DA, LDA, DW, DWORK, LDWORK, INFO)
CALL SSYEV  (JOBZ, UPLO, N, SA, LDA, SW, SWORK, LDWORK, INFO)

void dsyev  (char jobz, char uplo, int n, double *da, int lda,
             double *dw, int *info)
void ssyev  (char jobz, char uplo, int n, float *sa, int lda, float
             *sw, int *info)
```

Arguments

JOBZ	Indicates whether to compute eigenvalues only or to compute both eigenvalues and eigenvectors. The legal values for JOBZ are listed below. Any values not listed below are illegal. 'N' or 'n' Compute eigenvalues only. 'V' or 'v' Compute both eigenvalues and eigenvectors.
UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
xA	On entry, the upper or lower triangle of the matrix A. On exit, if JOBZ = 'V' or 'v' then A contains the orthonormal eigenvectors. If JOBZ = 'N' or 'n' then all of A is overwritten.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
xW	On exit, the N eigenvalues in ascending order.
xWORK	Scratch array with a dimension of LDWORK.
LDWORK	Leading dimension of the array WORK as specified in a dimension or type statement. $LDWORK \geq \max(1, 3 \times N - 1)$.
INFO	On exit: INFO = 0 Subroutine completed normally.

INFO < 0	The ith argument, where i = INFO , had an illegal value.
INFO > 0	Convergence failure.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C   INTEGER          LDA, LDWORK, N
C   PARAMETER        (N = 3)
C   PARAMETER        (LDA = N)
C   PARAMETER        (LDWORK = 3 * N - 1)
C
C   DOUBLE PRECISION A(LDA,N), EVALS(N), WORK(LDWORK)
C   INTEGER          ICOL, INFO, IROW
C
C   EXTERNAL          DSYEV
C   INTRINSIC         ABS
C
C   Initialize the array A to store the matrix A shown below.
C
C   9  1  1
C   A = 1  9  1
C   1  1  9
C
C   DATA A / 9.0D0, 8D8, 8D8, 1.0D0, 9.0D0, 8D8, 1.0D0, 1.0D0,
C   $          9.0D0 /
C
C   PRINT 1000
C   DO 100, IROW = 1, N
C       PRINT 1010, (A(ICOL,IROW), ICOL = 1, IROW - 1),
C   $           (A(IROW,ICOL), ICOL = IROW, N)
100 CONTINUE
C   PRINT 1020
C   DO 110, IROW = 1, N
C       PRINT 1010, (A(IROW,ICOL), ICOL = 1, N)
110 CONTINUE
C
C   Compute the eigenvalues and eigenvectors of A.
C
C   CALL DSYEV ('VALUES AND VECTORS', 'UPPER TRIANGULAR A', N,
C   $          A, LDA, EVALS, WORK, LDWORK, INFO)
C   IF (INFO .NE. 0) THEN
C       IF (INFO .LT. 0) THEN
C           PRINT 1030, ABS(INFO)
C           STOP 1
C       ELSE
C           PRINT 1040, INFO
C           STOP 2
C       END IF
C   END IF
C   END IF

```

```
C
C      Print the eigenvalues and eigenvectors.
C
      PRINT 1050
      DO 120, ICOL = 1, N
        PRINT 1060, EVALS(ICOL), (A(IROW,ICOL), IROW = 1, N)
      120 CONTINUE
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (3(3X, F4.1))
1020 FORMAT (/1X, 'A in symmetric form: (* in unused elements)')
1030 FORMAT (/1X, 'Illegal argument to DSYEV, argument #', I2)
1040 FORMAT (/1X, 'Convergence failure, INFO = ', I2)
1050 FORMAT (/1X, 'Eigenvalue', 4X, 'Eigenvector**T')
1060 FORMAT (1X, F7.2, 5X, '[', F4.2, ', ', F4.2, ', ', F4.2, ', ', F4.2, ']',
      $      F4.2, ']'')
C
      END
```

Sample Output

```
A in full form:
  9.0    1.0    1.0
  1.0    9.0    1.0
  1.0    1.0    9.0

A in symmetric form: (* in unused elements)
  9.0    1.0    1.0
****    9.0    1.0
****    ****    9.0

Eigenvalue      Eigenvector**T
  8.00      [0.04, -.73, 0.69]
  8.00      [0.82, -.37, -.44]
 11.00      [0.58, 0.58, 0.58]
```

Eigenvalues and Eigenvectors of a Symmetric Matrix (Expert Driver)

The subroutines described in this section compute the eigenvalues and, optionally, the eigenvectors of a symmetric matrix A. The eigenvectors are normalized to have a Euclidean norm of 1. Eigenvalues and eigenvectors can be selected by specifying either a range of values or a range of indices for the desired eigenvalues. Note that the simple driver xSYEV is also available.

Calling Sequence

```
CALL DSYEVX (JOBZ, RANGE, UPLO, N, DA, LDA, DVL, DVU, IL, IU,
             DABTOL, NFOUND, DW, DZ, LDZ, DWORK, LDWORK, IWORK2,
             IFAIL, INFO)
CALL SSYEVX (JOBZ, RANGE, UPLO, N, SA, LDA, SVL, SVU, IL, IU,
             SABTOL, NFOUND, SW, SZ, LDZ, SWORK, LDWORK, IWORK2,
             IFAIL, INFO)

void dsyevx (char jobz, char range, char uplo, int n, double *da,
            int lda, double dvl, double dvu, int il, int iu, double
            dabtol, int *nfound, double *dw, double *dz, int ldz,
            int *ifail, int *info)
void ssyevx (char jobz, char range, char uplo, int n, float *sa,
            int lda, float svl, float svu, int il, int iu, float
            sabtol, int *nfound, float *sw, float *sz, int ldz, int
            *ifail, int *info)
```

Arguments

JOBZ	Indicates whether to compute eigenvalues only or to compute both eigenvalues and eigenvectors. The legal values for JOBZ are listed below. Any values not listed below are illegal. 'N' or 'n' Compute eigenvalues only. 'V' or 'v' Compute both eigenvalues and eigenvectors.
RANGE	Indicates whether to compute all eigenvalues or some subset. Legal values for RANGE are shown below. Any values not shown below are illegal. 'A' or 'a' All eigenvalues will be computed. 'I' or 'i' The ILth through IUth eigenvalues will be computed; see the descriptions of the IL and IU arguments below.

	'V' or 'v'	All eigenvalues in the half-open interval (VL,VU] will be computed; see the descriptions of the VL and VU arguments below.
UPLO		Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal.
	'U' or 'u'	xA contains the upper triangle.
	'L' or 'l'	xA contains the lower triangle.
N		Order of the matrix A. $N \geq 0$.
xA		On entry, the upper or lower triangle of the matrix A. On exit, the upper or lower triangle of A including the diagonal has been overwritten.
LDA		Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
xVL, xVU		If RANGE = 'V' or 'v' then only eigenvalues in the half-open interval (VL,VU] will be computed. Otherwise not used.
IL, IU		If RANGE = 'T' or 't' then only the ILth through IUth eigenvalues will be computed. Otherwise not used. $1 \leq IL \leq IU \leq N$.
xABTOL		The absolute error tolerance for the eigenvalues. An approximate eigenvalue is considered to have converged when it lies in an interval [a,b] of width less than or equal to $ABTOL + \epsilon \times \max(a , b)$ where ϵ is machine precision. If $ABTOL \leq 0$ then $\epsilon \times \ T\ _1$ will be used in its place, where $\ T\ _1$ is the 1-norm of the tridiagonal matrix obtained by reducing A to tridiagonal form.
NFOUND		On exit, the total number of eigenvalues found. $0 \leq NFOUND \leq N$. If RANGE = 'A' or 'a' then $NFOUND = N$. If RANGE = 'T' or 't' then $NFOUND = IU - IL + 1$.
xW		On exit, W(1:NFOUND) contains the computed eigenvalues in ascending order.
xZ		On exit, if JOBZ = 'V' or 'v' then the first NFOUND columns of Z contain the orthonormal eigenvectors corresponding to the selected eigenvalues. If an eigenvector fails to converge then the corresponding column of Z contains an approximation to that eigenvector and its index is returned in IFAIL. If JOBZ = 'N' or 'n' then Z is not used. Note that Z must have at least NFOUND columns. If RANGE = 'V' or 'v' then NFOUND may not be known in advance and an upper bound must be used. For A(LDZ,K) where $NFOUND > K$, the eigenvectors will be stored into memory with unpredictable results.
LDZ		Leading dimension of the array Z as specified in a dimension or type statement. $LDZ \geq 1$. If JOBZ = 'V' or 'v' then $LDZ \geq \max(1, N)$.
xWORK		Scratch array with a dimension of LDWORK.

Sample Output

```
A in full form:
  9.0  1.0  1.0
  1.0  9.0  1.0
  1.0  1.0  9.0

A in symmetric form: (* in unused elements)
  9.0  1.0  1.0
****  9.0  1.0
****  ****  9.0

Eigenvalue      Eigenvector**T
  8.00      [0.04, -.73, 0.69]
  8.00      [0.82, -.37, -.44]
 11.00      [0.58, 0.58, 0.58]
```

Generalized Eigenvalues and Eigenvectors for Symmetric Matrices

The subroutines in this section compute the generalized eigenvalues (α , β) for a pair of symmetric matrices A and B. Optionally, these subroutines can also compute the generalized eigenvectors.

A generalized eigenvalue is a ratio $\lambda = \alpha / \beta$ such that $AX = \lambda BX$ or $\det(A - \lambda B) = 0$ for general matrices A, B, and X. λ is typically represented as a ratio rather than as a scalar because there are reasonable interpretations for $\alpha = 0$, $\beta = 0$, and for $\alpha = \beta = 0$. A right generalized eigenvector x corresponding to a generalized eigenvalue λ is defined by $(A - \lambda B)x = 0$. A left generalized eigenvector x corresponding to a generalized eigenvalue λ is defined by $(A - \lambda B)Hx = 0$. A good reference for generalized eigenproblems is the book *Matrix Computations*, 2nd. ed. by Golub and van Loan (1989, The Johns Hopkins University Press).

Calling Sequence

```
CALL DSYGV (ITYPE, JOBZ, UPLO, N, DA, LDA, DB, LDB, DW, DWORK,
            LDWORK, INFO)
CALL SSYGV (ITYPE, JOBZ, UPLO, N, SA, LDA, SB, LDB, SW, SWORK,
            LDWORK, INFO)

void dsygv (int itype, char jobz, char uplo, int n, double *da,
            int lda, double *db, int ldb, double *dw, int *info)
void ssygv (int itype, char jobz, char uplo, int n, float *sa, int
            lda, float *sb, int ldb, float *sw, int *info)
```

Arguments

ITYPE	Indicates the problem to be solved. Legal values for ITYPE are shown below. Any values not shown below are illegal.
1	$Ax = \lambda Bx$
2	$ABx = \lambda x$
3	$BAX = \lambda x$
JOBZ	Indicates whether to compute eigenvalues only or to compute both eigenvalues and eigenvectors. The legal values for JOBZ are listed below. Any values not listed below are illegal.
'N' or 'n'	Compute eigenvalues only.
'V' or 'v'	Compute both eigenvalues and eigenvectors.

UPLO	Indicates whether xA and xB contain the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA and xB both contain the upper triangle. 'L' or 'l' xA and xB both contain the lower triangle.
N	Order of the matrix A. $N \geq 0$.
xA	On entry, the upper or lower triangle of the matrix A. On exit, if JOBZ = 'V' or 'v' then A contains the matrix Z of eigenvectors normalized as shown below: ITYPE Eigenvectors stored in A 1 or 2 $ZT \times B \times Z = I$ 3 $ZT \times B-1 \times Z = I$ If JOBZ = 'N' or 'n' and UPLO = 'U' or 'u' then the upper triangle of A including the diagonal is overwritten. If JOBZ = 'N' or 'n' and UPLO = 'L' or 'l' then the lower triangle of A including the diagonal is overwritten.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
xB	On entry, the upper or lower triangle of the matrix B. On exit, if UPLO = 'U' or 'u' then B contains the matrix U from the Cholesky factorization UTU of B. If UPLO = 'L' or 'l' then B contains the matrix L from the Cholesky factorization LLT of B.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
xW	On exit, W(1:NFOUND) contains the computed eigenvalues in ascending order.
xWORK	Scratch array with a dimension of LDWORK.
LDWORK	Leading dimension of the array WORK as specified in a dimension or type statement. $LDWORK \geq \max(1, 3 \times N - 1)$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The <i>i</i> th argument, where $i = INFO $, had an illegal value. $0 < INFO \leq N$ Convergence failure. INFO > N The leading minor of order <i>i</i> of B, where $i = INFO - N$, is not positive definite.

Sample Program

```
PROGRAM TEST
IMPLICIT NONE
```

C

```

      INTEGER          ITYPE, LDA, LDB, LDWORK, N
      PARAMETER       (ITYPE = 1)
      PARAMETER       (N = 3)
      PARAMETER       (LDA = N)
      PARAMETER       (LDB = N)
      PARAMETER       (LDWORK = 3 * N - 1)
C
      DOUBLE PRECISION A(LDA,N), B(LDB,N), EVALS(N), WORK(LDWORK)
      INTEGER          ICOL, INFO, IROW
C
      EXTERNAL         DSYGV
      INTRINSIC        ABS
C
      Initialize the array A to store the matrix A shown below.
      Initialize the array B to store the matrix B shown below.
C
C      12  0  4      6  0  2
C      A =  0  6  0      b =  0  2  0
C      4  0  4      2  0  2
C
      DATA A / 1.2D1, 8D8, 8D8, 0.0D0, 6.0D0, 8D8, 4.0D0, 0.0D0,
$              4.0D0 /
      DATA B / 6.0D0, 8D8, 8D8, 0.0D0, 2.0D0, 8D8, 2.0D0, 0.0D0,
$              2.0D0 /
C
      PRINT 1000
      DO 100, IROW = 1, N
        PRINT 1010, (A(ICOL,IROW), ICOL = 1, IROW - 1),
$              (A(IROW,ICOL), ICOL = IROW, N)
100  CONTINUE
      PRINT 1020
      DO 110, IROW = 1, LDA
        PRINT 1010, (A(IROW,ICOL), ICOL = 1, N)
110  CONTINUE
      PRINT 1030
      DO 120, IROW = 1, N
        PRINT 1010, (B(ICOL,IROW), ICOL = 1, IROW - 1),
$              (B(IROW,ICOL), ICOL = IROW, N)
120  CONTINUE
      PRINT 1040
      DO 130, IROW = 1, LDB
        PRINT 1010, (B(IROW,ICOL), ICOL = 1, N)
130  CONTINUE
C
      CALL DSYGV (ITYPE, 'VALUES AND EIGENVECTORS',
$              'UPPER TRIANGLE OF A AND B STORED', N, A, LDA,
$              B, LDB, EVALS, WORK, LDWORK, INFO)
      IF (INFO .LT. 0) THEN
        PRINT 1050, ABS(INFO)
        STOP 1
      ELSE IF (INFO .GT. N) THEN
        PRINT 1060, INFO
        STOP 2
      ELSE IF (INFO .GT. 0) THEN
        PRINT 1070, INFO
        STOP 3

```

```

      END IF
C
C      Print the eigenvalues and eigenvectors.
C
      PRINT 1080
      DO 140, IROW = 1, N
        PRINT 1090, EVALS(IROW), (A(IROW,ICOL), ICOL = 1, N)
      140 CONTINUE
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (3(3X, F4.1))
1020 FORMAT (/1X, 'A in symmetric form: (* in unused elements)')
1030 FORMAT (/1X, 'B in full form:')
1040 FORMAT (/1X, 'B in symmetric form: (* in unused elements)')
1050 FORMAT (/1X, 'Illegal argument to DSYGV, argument #', I2)
1060 FORMAT (/1X, 'B is not positive definite, INFO = ', I2)
1070 FORMAT (/1X, 'Convergence failure, INFO = ', I2)
1080 FORMAT (/1X, 'Eigenvalue', 6X, 'Eigenvector**T')
1090 FORMAT (1X, F7.2, 6X, '[', 2(F5.3, ', '), F5.3, ']')
C
      END

```

Sample Output

```

A in full form:
 12.0   0.0   4.0
   0.0   6.0   0.0
   4.0   0.0   4.0

A in symmetric form: (* in unused elements)
 12.0   0.0   4.0
****   6.0   0.0
****   ****   4.0

B in full form:
 6.0   0.0   2.0
 0.0   2.0   0.0
 2.0   0.0   2.0

B in symmetric form: (* in unused elements)
 6.0   0.0   2.0
****   2.0   0.0
****   ****   2.0

Eigenvalue      Eigenvector**T
 2.00    [-.085, -.493, 0.000]
 2.00    [0.000, 0.000, 0.707]
 3.00    [-.612, 0.612, 0.000]

```

Refined Solution to a Linear System in a UDU- or LDL-Factored Symmetric Matrix

The subroutines described in this section refine the solution to a linear system $AX = B$ for a symmetric matrix A , which has been UDU-factored or LDL-factored by xSYTRF, and general matrices B and X . These subroutines also compute forward error bounds and backward error estimates for the refined solution.

Calling Sequence

```
CALL DSYRFS (UPLO, N, NRHS, DA, LDA, DAF, LDAF, IPIVOT, DB, LDB,
DX, LDX, DFERR, DBERR, DWORK, DWORK2, INFO)
CALL SSYRFS (UPLO, N, NRHS, SA, LDA, SAF, LDAF, IPIVOT, SB, LDB,
SX, LDX, SFERR, SBERR, SWORK, SWORK2, INFO)
CALL ZSYRFS (UPLO, N, NRHS, ZA, LDA, ZAF, LDAF, IPIVOT, ZB, LDB,
ZX, LDX, DFERR, DBERR, ZWORK, DWORK2, INFO)
CALL CSYRFS (UPLO, N, NRHS, CA, LDA, CAF, LDAF, IPIVOT, CB, LDB,
CX, LDX, SFERR, SBERR, CWORK, SWORK2, INFO)

void dsyrfs (char uplo, int n, int nrhs, double *da, int lda,
double *daf, int ldaf, int *ipivot, double *db, int
ldb, double *dx, int ldx, double *dferr, double *dberr,
int *info)
void ssyrfs (char uplo, int n, int nrhs, float *sa, int lda, float
*saf, int ldaf, int *ipivot, float *sb, int ldb, float
*sx, int ldx, float *sferr, float *sberr, int *info)
void zsyrrfs (char uplo, int n, int nrhs, doublecomplex *za, int
lda, doublecomplex *zaf, int ldaf, int *ipivot,
doublecomplex *zb, int ldb, doublecomplex *zx, int ldx,
double *dferr, double *dberr, int *info)
void csyrfs (char uplo, int n, int nrhs, complex *ca, int lda,
complex *caf, int ldaf, int *ipivot, complex *cb, int
ldb, complex *cx, int ldx, float *sferr, float *sberr,
int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal.
'U' or 'u'	xA contains the upper triangle.
'L' or 'l'	xA contains the lower triangle.

N	Order of the matrix A. $N \geq 0$.
NRHS	Number of right-hand sides, equal to the number of columns of the matrices B and X. $NRHS \geq 0$.
xA	Upper or lower triangle of the matrix A.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
xAF	UDU or LDL factorization of the matrix A as computed by xSYTRF.
LDAF	Leading dimension of the array AF as specified in a dimension or type statement. $LDAF \geq \max(1, N)$.
IPIVOT	Pivot indices as computed by xSYTRF.
xB	The $N \times NRHS$ right-hand side matrix B.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
xX	On entry, the $N \times NRHS$ solution matrix X. On exit, the refined $N \times NRHS$ solution matrix X.
LDX	Leading dimension of the array X as specified in a dimension or type statement. $LDX \geq \max(1, N)$.
xFERR	On exit, the estimated forward error bound for each solution vector $X(*, j)$ for $1 \leq j \leq NRHS$. If X' is the true solution corresponding to $X(*, j)$ then $FERR(j)$ is an upper bound on the magnitude of the largest element in $X(*, j) - X'$ divided by the magnitude of the largest element in $X(*, j)$.
xBERR	On exit, $BERR(j)$ is the smallest relative change in any element of A or $B(*, j)$ that makes $X(*, j)$ an exact solution to $AX(*, j) = B(*, j)$ for $1 \leq j \leq NRHS$.
xWORK	Scratch array with a dimension of $3 \times N$.
xWORK2	Scratch array with a dimension of N.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where $i = INFO $, had an illegal value.

Sample Program

	PROGRAM TEST
	IMPLICIT NONE
C	
	INTEGER LDA, LDAF, LDB, LDWORK, LDX, N, NRHS
	PARAMETER (N = 4)
	PARAMETER (LDA = N)
	PARAMETER (LDAF = N)
	PARAMETER (LDB = N)
	PARAMETER (LDWORK = 1)


```

C      PARAMETER      (LDB = LDB)
C      PARAMETER      (NRHS = 1)

C      DOUBLE PRECISION A(LDA,N), AF(LDAF,N), B(LDB,NRHS)
C      DOUBLE PRECISION BERR(NRHS), EPSLON, FERR(NRHS)
C      DOUBLE PRECISION WORK(LDWORK), X(LDX,NRHS)
C      INTEGER          ICOL, INFO, IPIVOT(N), IROW, IWORK(N)

C      EXTERNAL          DCOPY, DSYRFS, DSYTRF, DSYTRS
C      INTRINSIC          ABS, MAX

C      Initialize the array A to store the coefficient matrix A
C      shown below. Initialize the array B to store the right
C      hand side vector b shown below.
C
C      A =
C      0   0   30
C      0   2   2   b = 50
C      2   2   0   70
C      0   0   110

C      DATA A / 0.0D0, 8.0D8, 8.0D8, 8.0D8, 0.0D0, 2.0D0, 8.0D8,
C      $          8.0D8, 0.0D0, 2.0D0, 2.0D0, 8.0D8, 0.0D0, 0.0D0,
C      $          0.0D0, 0.0D0 /
C      DATA B / 3.0D0, 5.0D0, 7.0D0, 1.1D1 /

C      Slightly perturb each of the elements on the diagonal, the
C      first subdiagonal, and the first superdiagonal of A. After
C      this code, A will resemble the matrix shown below.
C
C      A =
C      2e   -e
C      -e   2+e   2-e
C      2-e   2+e   -e
C      -e   2e

C      EPSLON = ABS (((2.0D0 / 3.0D0) + 2.0D0) - 2.0D0) -
C      $ (2.0D0 / 3.0D0)
C      A(1,1) = EPSLON + EPSLON
C      A(N,N) = EPSLON + EPSLON
C      DO 100, IROW = 2, N - 1
C      A(IROW,IROW) = A(IROW,IROW) + EPSLON
100  CONTINUE
C      DO 110, IROW = 1, N - 1
C      A(IROW,IROW + 1) = A(IROW,IROW + 1) - EPSLON
110  CONTINUE

C      Slightly perturb each element of B. After this code, B will
C      resemble the matrix shown below.
C
C      B =
C      3-small
C      5-small
C      7-small
C      11-small

C      DO 130, ICOL = 1, NRHS
C      DO 120, IROW = 1, N
C      B(IROW,ICOL) = B(IROW,ICOL) * (1.0D0 - EPSLON)

```

Sun Performance Library Reference

```

        PRINT 1100, INFO
        STOP 3
      END IF
      PRINT 1110
      PRINT 1080, X
      PRINT 1120
      PRINT 1040, A(1,1) * X(1,1) + A(1,2) * X(2,1)
      PRINT 1040, A(1,2) * X(1,1) + A(2,2) * X(2,1) + A(2,3) *
$      X(3,1)
      PRINT 1040, A(2,3) * X(2,1) + A(3,3) * X(3,1) + A(3,4) *
$      X(4,1)
      PRINT 1040,
$      X(4,1)
        PRINT 1130, FERR(1)
        PRINT 1140, BERR(1)
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(2X, F18.16))
1020 FORMAT (/1X, 'A in symmetric form: (* in unused elements)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (1X, F22.17)
1050 FORMAT (1X, 'Error factoring A, INFO = ', I5)
1060 FORMAT (1X, 'Error solving Ax=b, INFO = ', I5)
1070 FORMAT (/1X, 'Initial solution to Ax=b:')
1080 FORMAT (1X, E25.17)
1090 FORMAT (/1X, 'Ax with the initial x:')
1100 FORMAT (1X, 'Illegal argument to DSYRFS, INFO = ', I3)
1110 FORMAT (/1X, 'Refined solution to Ax=b:')
1120 FORMAT (/1X, 'Ax with refined x:')
1130 FORMAT (/1X, 'Forward error: ', E14.8)
1140 FORMAT (1X, 'Backward error: ', E14.8)
C
      END

```

Sample Output

```

A in full form:
 0.00000000000000002  -0.00000000000000001  0.00000000000000000  0.00000000000000000
-0.00000000000000001  2.00000000000000000  2.00000000000000000  0.00000000000000000
 0.00000000000000000  2.00000000000000000  2.00000000000000000  -0.00000000000000001
 0.00000000000000000  0.00000000000000000  -0.00000000000000001  0.00000000000000002

A in symmetric form: (* in unused elements)
 0.00000000000000002  -0.00000000000000001  0.00000000000000000  0.00000000000000000
*****
 2.00000000000000000  2.00000000000000000  0.00000000000000000
*****
 2.00000000000000000  2.00000000000000000  -0.00000000000000001
*****
 0.00000000000000000  0.00000000000000000
*****

```

```
b:
  2.9999999999999956
  4.9999999999999911
  6.9999999999999911
 10.9999999999999822
```

```
Initial solution to Ax=b:
  0.67553994410557432E+17
  0.10808639105689189E+18
 -0.10808639105689189E+18
 -0.45035996273704960E+16
```

```
Ax with the initial x:
  3.0000000000000000
  0.0000000000000000
  0.5000000000000000
 10.9999999999999822
```

```
Refined solution to Ax=b:
  0.67553994410557432E+17
  0.10808639105689189E+18
 -0.10808639105689189E+18
 -0.45035996273704960E+16
```

```
Ax with refined x:
  3.0000000000000000
  0.0000000000000000
  0.5000000000000000
 10.9999999999999822
```

```
Forward error:  0.73614937E-01
Backward error: 0.11564823E-17
```

Solution to a Linear System in a Symmetric Matrix (Simple Driver)

The subroutines described in this section solve a linear system $AX = B$ for a symmetric matrix A and general matrices B and X . Note that the expert driver `xSYSVX` is also available.

Calling Sequence

```
CALL DSYSV (UPLO, N, NRHS, DA, LDA, IPIVOT, DB, LDB, DWORK,
            LDWORK, INFO)
CALL SSYSV (UPLO, N, NRHS, SA, LDA, IPIVOT, SB, LDB, SWORK,
            LDWORK, INFO)
CALL ZSYSV (UPLO, N, NRHS, ZA, LDA, IPIVOT, ZB, LDB, ZWORK,
            LDWORK, INFO)
CALL CSYSV (UPLO, N, NRHS, CA, LDA, IPIVOT, CB, LDB, CWORK,
            LDWORK, INFO)

void dsysv (char uplo, int n, int nrhs, double *da, int lda, int
            *ipivot, double *db, int ldb, int *info)
void ssysv (char uplo, int n, int nrhs, float *sa, int lda, int
            *ipivot, float *sb, int ldb, int *info)
void zsysv (char uplo, int n, int nrhs, doublecomplex *za, int
            lda, int *ipivot, doublecomplex *zb, int ldb, int
            *info)
void csysv (char uplo, int n, int nrhs, complex *ca, int lda, int
            *ipivot, complex *cb, int ldb, int *info)
```

Arguments

UPLO	Indicates whether <code>xA</code> contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' <code>xA</code> contains the upper triangle. 'L' or 'l' <code>xA</code> contains the lower triangle.
N	Order of the matrix A . $N \geq 0$.
NRHS	Number of right-hand sides, equal to the number of columns of the matrix B . $NRHS \geq 0$.
<code>xA</code>	On entry, the upper or lower triangle of the matrix A . On exit, the UDU or LDL factorization of the matrix A , as computed by <code>xSYTRF</code> .
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.

IPIVOT	On exit, pivot indices as computed by xSYTRF.
xB	On entry, the N×NRHS right-hand side matrix B. On exit, the N×NRHS solution matrix X.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
xWORK	Scratch array with a dimension of LDWORK.
LDWORK	Leading dimension of the array WORK as specified in a dimension or type statement. $LDWORK \geq 1$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where i = INFO , had an illegal value. INFO > 0 D(i,i), where i = INFO, is exactly zero and D is therefore singular. The factorization has been completed, but the solution could not be computed.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C      INTEGER          LDA, LDB, LDWORK, M, N, NRHS
C      PARAMETER        (M = 3)
C      PARAMETER        (N = 3)
C      PARAMETER        (NRHS = 1)
C      PARAMETER        (LDA = M)
C      PARAMETER        (LDB = LDA)
C      PARAMETER        (LDWORK = 1)
C
C      DOUBLE PRECISION A(LDA,N), B(LDB,NRHS), WORK(LDWORK)
C      INTEGER          ICOL, INFO, IROW, IPIVOT(N)
C
C      EXTERNAL          DSYSV
C
C      Initialize the array A to store the coefficient matrix A
C      shown below. Initialize the array B to store the right
C      hand vector b shown below.
C
C      1  2  2      15
C      A = 2  1  2      b = 15
C      2  2  1      15
C
C      DATA A / 1.0D0, 8D8, 8D8, 2.0D0, 1.0D0, 8D8, 2.0D0, 2.0D0,
C      $      1.0D0 /
C      DATA B / 1.5D1, 1.5D1, 1.5D1 /
C
C      PRINT 1000
C      DO 100, IROW = 1, N

```

```
      PRINT 1010, (A(ICOL,IROW), ICOL = 1, IROW - 1),  
$      (A(IROW,ICOL), ICOL = IROW, N)  
100 CONTINUE  
  PRINT 1020  
  DO 110, IROW = 1, LDA  
    PRINT 1010, (A(IROW,ICOL), ICOL = 1, N)  
110 CONTINUE  
  PRINT 1030  
  PRINT 1040, B  
C  
  CALL DSYSV ('UPPER TRIANGULAR A', N, NRHS, A, LDA, IPIVOT,  
$           B, LDB, WORK, LDWORK, INFO)  
  IF (INFO .EQ. 0) THEN  
    PRINT 1050  
    PRINT 1040, B  
  ELSE  
    PRINT 1060, INFO  
  END IF  
C  
1000 FORMAT (1X, 'A in full form:')  
1010 FORMAT (3(3X, F4.1))  
1020 FORMAT (/1X, 'A in symmetric form: (* in unused elements)')  
1030 FORMAT (/1X, 'b:')  
1040 FORMAT (1X, 2X, F4.1)  
1050 FORMAT (/1X, 'x:')  
1060 FORMAT (1X, 'Solve failed with INFO = ', I6)  
C  
  END
```

Sample Output

```
A in full form:  
  1.0    2.0    2.0  
  2.0    1.0    2.0  
  2.0    2.0    1.0  
  
A in symmetric form: (* in unused elements)  
  1.0    2.0    2.0  
****    1.0    2.0  
****    ****    1.0  
  
b:  
 15.0  
 15.0  
 15.0  
  
x:  
  3.0  
  3.0  
  3.0
```

Solution to a Linear System in a Symmetric Matrix (Expert Driver)

The subroutines described in this section solve a linear system $AX = B$ for a symmetric matrix A and general matrices B and X . These subroutines also compute forward error bounds and backward error estimates for the solution, and estimate the condition number of the matrix A . Note that the simple driver `xSYSV` is also available.

Calling Sequence

```
CALL DSYSVX (FACT, UPLO, N, NRHS, DA, LDA, DAF, LDAF, IPIVOT, DB,
             LDB, DX, LDX, DRCOND, DFERR, DBERR, DWORK, LDWORK,
             IWORK2, INFO)
CALL SSYSVX (FACT, UPLO, N, NRHS, SA, LDA, SAF, LDAF, IPIVOT, SB,
             LDB, SX, LDX, SRCOND, SFERR, SBERR, SWORK, LDWORK,
             IWORK2, INFO)
CALL ZSYSVX (FACT, UPLO, N, NRHS, ZA, LDA, ZAF, LDAF, IPIVOT, ZB,
             LDB, ZX, LDX, DRCOND, DFERR, DBERR, ZWORK, LDWORK,
             DWORK2, INFO)
CALL CSYSVX (FACT, UPLO, N, NRHS, CA, LDA, CAF, LDAF, IPIVOT, CB,
             LDB, CX, LDX, SRCOND, SFERR, SBERR, CWORK, LDWORK,
             SWORK2, INFO)

void dsysvx (char fact, char uplo, int n, int nrhs, double *da, int
             lda, double *daf, int ldaf, int *ipivot, double *db,
             int ldb, double *dx, int ldx, double *drcond, double
             *dferr, double *dberr, int *info)
void ssysvx (char fact, char uplo, int n, int nrhs, float *sa, int
             lda, float *saf, int ldaf, int *ipivot, float *sb, int
             ldb, float *sx, int ldx, float *srcond, float *sferr,
             float *sberr, int *info)
void zsysvx (char fact, char uplo, int n, int nrhs, doublecomplex
             *za, int lda, doublecomplex *zaf, int ldaf, int
             *ipivot, doublecomplex *zb, int ldb, doublecomplex
             *zx, int ldx, double *drcond, double *dferr, double
             *dberr, int *info)
void csysvx (char fact, char uplo, int n, int nrhs, complex *ca,
             int lda, complex *caf, int ldaf, int *ipivot, complex
             *cb, int ldb, complex *cx, int ldx, float *srcond,
             float *sferr, float *sberr, int *info)
```


Arguments

FACT	Indicates whether or not the factored form of A has been supplied on entry. 'F' or 'f' On entry, AF and IPIVOT contain the factored form of A. AF and IPIVOT will not be modified. 'N' or 'n' On entry, the matrix A will be copied to AF and factored.
UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
NRHS	Number of right-hand sides, equal to the number of columns of the matrices B and X. $NRHS \geq 0$.
xA	Upper or lower triangle of the matrix A.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
xAF	On entry, if FACT = 'F' or 'f' then AF contains the UDU or LDL factorization of A, as computed by xSYTRF. Otherwise the entry value of AF is not used. On exit, if FACT = 'N' or 'n' then AF contains the UDU or LDL factorization of A, as computed by xSYTRF. Otherwise AF is unchanged.
LDAF	Leading dimension of the array AF as specified in a dimension or type statement. $LDAF \geq \max(1, N)$.
IPIVOT	On entry, if FACT = 'F' or 'f' then IPIVOT contains pivot indices as computed by xSYTRF. Otherwise the entry values of IPIVOT are not used. On exit, if FACT = 'N' or 'n' then IPIVOT contains the pivot indices as computed by xSYTRF. Otherwise IPIVOT is unchanged.
xB	The $N \times NRHS$ right-hand side matrix B.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
xx	On exit, the $N \times NRHS$ solution matrix X.
LDX	Leading dimension of the array X as specified in a dimension or type statement. $LDX \geq \max(1, N)$.

xRCOND	On exit, an estimate of the reciprocal condition number of the matrix A, where the reciprocal condition number is defined to be $1 / (\ A\ \times \ A^{-1}\)$. The reciprocal of the condition number is estimated instead of the condition number itself to avoid overflow or division by zero. If RCOND is less than machine precision (in particular, if RCOND = 0) then A is singular to working precision. In this case, INFO > 0 is returned and the solution and error bounds are not computed.
xFERR	On exit, the estimated forward error bound for each solution vector X(*, j) for $1 \leq j \leq \text{NRHS}$. If X' is the true solution corresponding to X(*, j) then FERR(j) is an upper bound on the magnitude of the largest element in X(*, j) - X' divided by the magnitude of the largest element in X(*, j).
xBERR	On exit, BERR(j) is the smallest relative change in any element of A or B(*, j) that makes X(*, j) an exact solution to AX(*, j) = B(*, j) for $1 \leq j \leq \text{NRHS}$.
xWORK	Scratch array with a dimension of LDWORK.
LDWORK	Leading dimension of the array WORK as specified in a dimension or type statement. LDWORK $\geq 3 \times N$ for real subroutines or LDWORK $\geq 2 \times N$ for complex subroutines.
xWORK2	Scratch array with a dimension of N.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where i = INFO , had an illegal value. $1 \leq \text{INFO} \leq N$ D(i,i), where i = INFO, is exactly zero and D is therefore singular. The factorization has been completed, but the solution and error bounds could not be computed. INFO = N+1 The block diagonal matrix D is nonsingular, but RCOND is less than machine precision. The factorization has been completed, but the matrix is singular to working precision, so the solution and error bounds could not be computed.

Sample Program

C	PROGRAM TEST	
	IMPLICIT NONE	
	INTEGER	LDA, LDAF, LDB, LDIWRK, LDWORK, LDX
	INTEGER	N, NRHS
	PARAMETER	(N = 3)

```

PARAMETER      (NRHS = 1)
PARAMETER      (LDA = N)
PARAMETER      (LDAF = LDA)
PARAMETER      (LDB = LDA)
PARAMETER      (LDIWRK = N)
PARAMETER      (LDWORK = 3 * N)
PARAMETER      (LDX = N)
C
DOUBLE PRECISION A(LDA,N), AF(LDAF,N), B(LDB,NRHS)
DOUBLE PRECISION BERR(NRHS), FERR(NRHS), RCOND
DOUBLE PRECISION WORK(LDWORK), X(LDX,NRHS)
INTEGER         ICOL, INFO, IROW, IPIVOT(N), IWORK(LDIWRK)
C
EXTERNAL        DSYSVX
C
C   Initialize the array A to store the coefficient matrix A
C   shown below. Initialize the array B to store the right
C   hand vector b shown below.
C
C      1  2  2      15
C   A = 2  1  2      b = 15
C      2  2  1      15
C
DATA A / 1.0D0, 8D8, 8D8, 2.0D0, 1.0D0, 8D8, 2.0D0, 2.0D0,
$      1.0D0 /
DATA B / 1.5D1, 1.5D1, 1.5D1 /
C
C   Print the initial values of the arrays.
C
PRINT 1000
DO 100, IROW = 1, N
  PRINT 1010, (0.0D0, ICOL = 1, IROW - 1),
$            (A(IROW, ICOL), ICOL = IROW, N)
100 CONTINUE
PRINT 1020
DO 110, IROW = 1, LDA
  PRINT 1010, (A(IROW,ICOL), ICOL = 1, N)
110 CONTINUE
PRINT 1030
PRINT 1040, B
C
C   Solve the system Ax=b, then print the result.
C
CALL DSYSVX ('NOT FACTORED A', 'UPPER TRIANGLE OF A STORED',
$           N, NRHS, A, LDA, AF, LDAF, IPIVOT, B, LDB,
$           X, LDX, RCOND, FERR, BERR, WORK, LDWORK,
$           IWORK, INFO)
IF (INFO .EQ. 0) THEN
  PRINT 1050
  PRINT 1040, X
  PRINT 1060, (1.0D0 / RCOND)
  PRINT 1070, (IROW, BERR(IROW), IROW = 1, NRHS)
  PRINT 1080, (IROW, FERR(IROW), IROW = 1, NRHS)
ELSE
  PRINT 1090, INFO
END IF

```

```

C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (3(3X, F4.1))
1020 FORMAT (/1X, 'A in symmetric form: (* in unused elements)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (1X, 2X, F4.1)
1050 FORMAT (/1X, 'x:')
1060 FORMAT (/1X, 'Estimated condition number: ', F6.3)
1070 FORMAT (/1X, 'Backward error for system #', I1, ': ', E12.6)
1080 FORMAT (1X, 'Forward error for system #', I1, ': ', E12.6)
1090 FORMAT (1X, 'Solve failed with INFO = ', I6)
C
      END

```

Sample Output

```

A in full form:
  1.0   2.0   2.0
  0.0   1.0   2.0
  0.0   0.0   1.0

A in symmetric form: (* in unused elements)
  1.0   2.0   2.0
****   1.0   2.0
****   ****   1.0

b:
 15.0
 15.0
 15.0

x:
  3.0
  3.0
  3.0

Estimated condition number:  7.000

Backward error for system #1: 0.592119E-16
Forward error for system #1:  0.645410E-14

```

UDU or LDL Factorization of a Symmetric Matrix

The subroutines described in this section compute a UDU or LDL factorization of a symmetric matrix A . It is typical to follow a call to xSYTRF with a call to xSYSVX or xSYTRS to solve a linear system $AX = B$, to xSYCON to estimate the condition number of A , or to xSYTRI to compute A^{-1} .

Calling Sequence

```
CALL DSYTRF (UPLO, N, DA, LDA, IPIVOT, DWORK, LDWORK, INFO)
CALL SSYTRF (UPLO, N, SA, LDA, IPIVOT, SWORK, LDWORK, INFO)
CALL ZSYTRF (UPLO, N, ZA, LDA, IPIVOT, ZWORK, LDWORK, INFO)
CALL CSYTRF (UPLO, N, CA, LDA, IPIVOT, CWORK, LDWORK, INFO)

void dsytrf (char uplo, int n, double *da, int lda, int *ipivot,
            int *info)
void ssytrf (char uplo, int n, float *sa, int lda, int *ipivot, int
            *info)
void zsytrf (char uplo, int n, doublecomplex *za, int lda, int
            *ipivot, int *info)
void csytrf (char uplo, int n, complex *ca, int lda, int *ipivot,
            int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
xA	On entry, the upper or lower triangle of the matrix A. On exit, a UDU or LDL factorization of A.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
IPIVOT	On exit, an N-element array containing details of the row interchanges and block structure of D.
xWORK	Scratch array with a dimension of LDWORK.
LDWORK	Leading dimension of the array WORK as specified in a dimension or type statement. $LDWORK \geq 1$.

INFO	On exit:	
INFO = 0	INFO = 0	Subroutine completed normally.
INFO < 0	INFO < 0	The ith argument, where i = INFO , had an illegal value.
INFO > 0	INFO > 0	D(i,i), where i = INFO, is exactly zero, and D is therefore singular. The factorization is complete, but division by zero will occur if this factorization is used to solve a linear system.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C   INTEGER          LDA, LDB, LDWORK, N, NRHS
C   PARAMETER        (N = 3)
C   PARAMETER        (NRHS = 1)
C   PARAMETER        (LDA = N)
C   PARAMETER        (LDB = LDA)
C   PARAMETER        (LDWORK = 1)
C
C   DOUBLE PRECISION A(LDA,N), B(LDB,NRHS), WORK(LDWORK)
C   INTEGER          ICOL, INFO, IROW, IPIVOT(N)
C
C   EXTERNAL          DSYTRF, DSYTRS
C
C   Initialize the array A to store the coefficient matrix A
C   shown below. Initialize the array B to store the right
C   hand side vector b shown below.
C
C       1  2  2      15
C   A = 2  1  2      b = 15
C       2  2  1      15
C
C   DATA A / 1.0D0, 8D8, 8D8, 2.0D0, 1.0D0, 8D8, 2.0D0, 2.0D0,
$         1.0D0 /
C   DATA B / N*1.5D1 /
C
C   Print the initial values of the arrays.
C
C   PRINT 1000
C   DO 100, IROW = 1, N
C       PRINT 1010, (A(ICOL,IROW), ICOL = 1, IROW - 1),
$         (A(IROW,ICOL), ICOL = IROW, N)
100 CONTINUE
C   PRINT 1020
C   DO 110, IROW = 1, LDA
C       PRINT 1010, (A(IROW,ICOL), ICOL = 1, N)
110 CONTINUE
C   PRINT 1030

```

```
      PRINT 1040, B
C
C      Solve the system Ax=b, then print the result.
C
      CALL DSYTRF ('UPPER TRIANGLE OF A STORED', N, A, LDA,
$             IPIVOT, WORK, LDWORK, INFO)
      IF (INFO .EQ. 0) THEN
      CALL DSYTRS ('UPPER TRIANGULAR FACTOR', N, NRHS, A, LDA,
$             IPIVOT, B, LDB, INFO)
      IF (INFO .EQ. 0) THEN
      PRINT 1050
      PRINT 1040, B
      ELSE
      PRINT 1060, INFO
      END IF
      ELSE
      PRINT 1070, INFO
      END IF
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (3(3X, F4.1))
1020 FORMAT (/1X, 'A in symmetric form: (* in unused elements)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (1X, 2X, F4.1)
1050 FORMAT (/1X, 'x:')
1060 FORMAT (1X, 'Solve failed with INFO = ', I6)
1070 FORMAT (1X, 'Factorization failed with INFO = ', I6)
C
      END
```

Sample Output

```
A in full form:
  1.0   2.0   2.0
  2.0   1.0   2.0
  2.0   2.0   1.0

A in symmetric form: (* in unused elements)
  1.0   2.0   2.0
****   1.0   2.0
****   ****   1.0

b:
 15.0
 15.0
 15.0

x:
  3.0
  3.0
  3.0
```

Inverse of a UDU- or LDL-Factored Symmetric Matrix

The subroutines described in this section compute the inverse of a symmetric matrix A, which has been UDU-factored or LDL-factored by xSYTRF.

Calling Sequence

```
CALL DSYTRI (UPLO, N, DA, LDA, IPIVOT, DWORK, INFO)
CALL SSYTRI (UPLO, N, SA, LDA, IPIVOT, SWORK, INFO)
CALL ZSYTRI (UPLO, N, ZA, LDA, IPIVOT, ZWORK, INFO)
CALL CSYTRI (UPLO, N, CA, LDA, IPIVOT, CWORK, INFO)

void dsytri (char uplo, int n, double *da, int lda, int *ipivot,
            int *info)
void ssytri (char uplo, int n, float *sa, int lda, int *ipivot, int
            *info)
void zsytri (char uplo, int n, doublecomplex *za, int lda, int
            *ipivot, int *info)
void csytri (char uplo, int n, complex *ca, int lda, int *ipivot,
            int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
xA	On entry, the UDU or LDL factorization of the matrix A, as computed by xSYTRF. On exit, the symmetric inverse of the matrix A. If UPLO = 'U' or 'u', the upper triangular part of the inverse is formed and the part of A below the diagonal is not used. If UPLO = 'L' or 'l' the lower triangular part of the inverse is formed and the part of A above the diagonal is not used.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
IPIVOT	Pivot indices as computed by xSYTRF.
xWORK	Scratch array with a dimension of N for real subroutines or $2 \times N$ for complex subroutines.

INFO	On exit:	
	INFO = 0	Subroutine completed normally.
	INFO < 0	The <i>i</i> th argument, where <i>i</i> = INFO , had an illegal value.
	INFO > 0	D(<i>i</i> , <i>i</i>), where <i>i</i> = INFO, is zero. The matrix is therefore singular and its inverse could not be computed.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C   INTEGER          LDA, LDWORK, N
C   PARAMETER        (N = 4)
C   PARAMETER        (LDA = N)
C   PARAMETER        (LDWORK = N)
C
C   DOUBLE PRECISION A(LDA,N), WORK(LDWORK)
C   INTEGER          ICOL, INFO, IPIVOT(N), IROW
C
C   EXTERNAL          DSYTRF, DSYTRI
C   INTRINSIC         ABS
C
C   Initialize the array A to store in symmetric form the 4x4
C   symmetric coefficient matrix A shown below.
C
C       2  1  0  0
C   A = 1  2  1  0
C       0  1  2  1
C       0  0  1  1
C
C   DATA A / 2.0D0, 3*8D8, 1.0D0, 2.0D0, 2*8D8, 0.0D0, 1.0D0,
$           2.0D0, 8D8, 0.0D0, 0.0D0, 1.0D0, 1.0D0 /
C
C   Print the initial values of the arrays.
C
C   PRINT 1000
C   DO 100, IROW = 1, N
C       PRINT 1010, (A(ICOL,IROW), ICOL = 1, IROW - 1),
$           (A(IROW,ICOL), ICOL = IROW, N)
100 CONTINUE
C   PRINT 1020
C   PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, LDA)
C
C   UDU factor A.
C
C   CALL DSYTRF ('UPPER TRIANGLE OF A STORED', N, A, LDA,
$           IPIVOT, WORK, LDWORK, INFO)
C   IF (INFO .NE. 0) THEN

```

```

        PRINT 1030, INFO
        STOP 1
    END IF
C
C    Use the factored form of A to compute the inverse of A and
C    print the inverse thus computed.
C
    CALL DSYTRI ('UPPER TRIANGULAR FACTOR', N, A, LDA, IPIVOT,
$              WORK, INFO)
    IF (INFO .NE. 0) THEN
        PRINT 1040, ABS(INFO)
        STOP 2
    END IF
    PRINT 1050
    DO 200, IROW = 1, N
        PRINT 1010, (A(ICOL,IROW), ICOL = 1, IROW - 1),
$              (A(IROW,ICOL), ICOL = IROW, N)
    200 CONTINUE
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F6.3))
1020 FORMAT (/1X, 'A in symmetric form: (* in unused elements)')
1030 FORMAT (1X, 'Error factoring A, INFO = ', I5)
1040 FORMAT (1X, 'Error computing inverse of A, INFO = ', I5)
1050 FORMAT (/1X, 'A**(-1):')
C
    END

```

Sample Output

```

A in full form:
  2.000   1.000   0.000   0.000
  1.000   2.000   1.000   0.000
  0.000   1.000   2.000   1.000
  0.000   0.000   1.000   1.000

A in symmetric form: (* in unused elements)
  2.000   1.000   0.000   0.000
  ***** 2.000   1.000   0.000
  ***** ***** 2.000   1.000
  ***** ***** ***** 1.000

A**(-1):
  1.000  -1.000   1.000  -1.000
 -1.000   2.000  -2.000   2.000
  1.000  -2.000   3.000  -3.000
 -1.000   2.000  -3.000   4.000

```

Solution to a Linear System in a UDU- or LDL-Factored Symmetric Matrix

The subroutines described in this section solve a linear system $AX = B$ for a symmetric matrix A , which has been UDU-factored or LDL-factored by xSYTRF, and general matrices B and X .

Calling Sequence

```
CALL DSYTRS (UPLO, N, NRHS, DA, LDA, IPIVOT, DB, LDB, INFO)
CALL SSYTRS (UPLO, N, NRHS, SA, LDA, IPIVOT, SB, LDB, INFO)
CALL ZSYTRS (UPLO, N, NRHS, ZA, LDA, IPIVOT, ZB, LDB, INFO)
CALL CSYTRS (UPLO, N, NRHS, CA, LDA, IPIVOT, CB, LDB, INFO)

void dsytrs (char uplo, int n, int nrhs, double *da, int lda, int
             *ipivot, double *db, int ldb, int *info)
void ssytrs (char uplo, int n, int nrhs, float *sa, int lda, int
             *ipivot, float *sb, int ldb, int *info)
void zsytrs (char uplo, int n, int nrhs, doublecomplex *za, int
             lda, int *ipivot, doublecomplex *zb, int ldb, int
             *info)
void csytrs (char uplo, int n, int nrhs, complex *ca, int lda, int
             *ipivot, complex *cb, int ldb, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
N	Order of the matrix A. $N \geq 0$.
NRHS	Number of right-hand sides, equal to the number of columns of the matrix B. $NRHS \geq 0$.
xA	UDU or LDL factorization of the matrix A, as computed by xSYTRF.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
IPIVOT	Pivot indices as computed by xSYTRF.
xB	On entry, the $N \times NRHS$ right-hand side matrix B. On exit, the $N \times NRHS$ solution matrix X.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.

INFO	On exit:	
	INFO = 0	Subroutine completed normally.
	INFO < 0	The <i>i</i> th argument, where <i>i</i> = INFO , had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C      INTEGER          LDA, LDB, LDWORK, N, NRHS
C      PARAMETER        (N = 3)
C      PARAMETER        (NRHS = 1)
C      PARAMETER        (LDA = N)
C      PARAMETER        (LDB = LDA)
C      PARAMETER        (LDWORK = 1)
C
C      DOUBLE PRECISION A(LDA,N), B(LDB,NRHS), WORK(LDWORK)
C      INTEGER          ICOL, INFO, IROW, IPIVOT(N)
C
C      EXTERNAL          DSYTRF, DSYTRS
C
C      Initialize the array A to store the coefficient matrix A
C      shown below. Initialize the array B to store the right
C      hand side vector b shown below.
C
C      1  2  2      15
C      A = 2  1  2      b = 15
C      2  2  1      15
C
C      DATA A / 1.0D0, 8D8, 8D8, 2.0D0, 1.0D0, 8D8, 2.0D0, 2.0D0,
$          1.0D0 /
C      DATA B / N*1.5D1 /
C
C      Print the initial values of the arrays.
C
C      PRINT 1000
C      DO 100, IROW = 1, N
C          PRINT 1010, (A(ICOL,IROW), ICOL = 1, IROW - 1),
$              (A(IROW,ICOL), ICOL = IROW, N)
100  CONTINUE
C      PRINT 1020
C      DO 110, IROW = 1, LDA
C          PRINT 1010, (A(IROW,ICOL), ICOL = 1, N)
110  CONTINUE
C      PRINT 1030
C      PRINT 1040, B
C
C      Solve the system Ax=b, then print the result.
C
C      CALL DSYTRF ('UPPER TRIANGLE OF A STORED', N, A, LDA,

```

```
      $          IPIVOT, WORK, LDWORK, INFO)
      IF (INFO .EQ. 0) THEN
        CALL DSYTRS ('UPPER TRIANGULAR FACTOR', N, NRHS, A, LDA,
      $          IPIVOT, B, LDB, INFO)
        IF (INFO .EQ. 0) THEN
          PRINT 1050
          PRINT 1040, B
        ELSE
          PRINT 1060, INFO
        END IF
      ELSE
        PRINT 1070, INFO
      END IF
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (3(3X, F4.1))
1020 FORMAT (/1X, 'A in symmetric form: (* in unused elements)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (1X, 2X, F4.1)
1050 FORMAT (/1X, 'x:')
1060 FORMAT (1X, 'Solve failed with INFO = ', I6)
1070 FORMAT (1X, 'Factorization failed with INFO = ', I6)
C
      END
```

Sample Output

```
A in full form:
  1.0   2.0   2.0
  2.0   1.0   2.0
  2.0   2.0   1.0

A in symmetric form: (* in unused elements)
  1.0   2.0   2.0
****   1.0   2.0
****   ****   1.0

b:
 15.0
 15.0
 15.0

x:
  3.0
  3.0
  3.0
```

Reciprocal Condition Number of a Triangular Matrix in Banded Storage

The subroutines described in this section estimate the reciprocal condition number, in either the 1-norm or the ∞ -norm, of a triangular matrix A in banded storage.

Calling Sequence

```
CALL DTBCON (NORM, UPLO, DIAG, N, NDIAG, DA, LDA, DRCOND, DWORK,
             IWORK2, INFO)
CALL STBCON (NORM, UPLO, DIAG, N, NDIAG, SA, LDA, SRCOND, SWORK,
             IWORK2, INFO)
CALL ZTBCON (NORM, UPLO, DIAG, N, NDIAG, ZA, LDA, DRCOND, ZWORK,
             DWORK2, INFO)
CALL CTBCON (NORM, UPLO, DIAG, N, NDIAG, CA, LDA, SRCOND, CWORK,
             SWORK2, INFO)

void dtbcon (char norm, char uplo, char diag, int n, int ndiag,
            double *da, int lda, double *drcond, int *info)
void stbcon (char norm, char uplo, char diag, int n, int ndiag,
            float *sa, int lda, float *srcond, int *info)
void ztbcon (char norm, char uplo, char diag, int n, int ndiag,
            doublecomplex *za, int lda, double *drcond, int *info)
void ctbcon (char norm, char uplo, char diag, int n, int ndiag,
            complex *ca, int lda, float *srcond, int *info)
```

Arguments

NORM	Indicates whether the 1-norm condition number or the ∞ -norm condition number is required. The legal values for NORM are listed below. Any values not listed below are illegal. 'O', 'o', or '1' 1-norm condition number is required. 'T' or 'i' ∞ -norm condition number is required.
UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
DIAG	Indicates whether or not A is unit triangular. The legal values for DIAG are listed below. Any values not listed below are illegal. 'N' or 'n' A is not unit triangular. 'U' or 'u' A is unit triangular.

N	Order of the matrix A. $N \geq 0$.
NDIAG	Number of superdiagonals or subdiagonals of the triangular matrix A. $N-1 \geq \text{NDIAG} \geq 0$ but if $N = 0$ then $\text{NDIAG} = 0$.
xA	Upper or lower triangular matrix A. If $\text{DIAG} = 'U'$ or $'u'$, the diagonal elements of A are assumed to be 1 and are not used.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $\text{LDA} \geq \text{NDIAG} + 1$.
xRCOND	On exit, the estimated reciprocal of the condition number of the matrix A where the reciprocal condition number of A is defined to be $1 / (\ A\ \times \ A^{-1}\)$. The reciprocal of the condition number is estimated instead of the condition number itself to avoid overflow or division by zero. If RCOND is less than machine precision (in particular, if $\text{RCOND} = 0$) then A is singular to working precision.
xWORK	Scratch array with a dimension of $3 \times N$ for real subroutines or $2 \times N$ for complex subroutines.
xWORK2	Scratch array with a dimension of N.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The <i>i</i> th argument, where $i = \text{INFO} $, had an illegal value.

Sample Program

```
PROGRAM TEST
IMPLICIT NONE

C
C      INTEGER          LDA, LDIWRK, LDWORK, N, NSUB
C      PARAMETER        (N = 5)
C      PARAMETER        (NSUB = 2)
C      PARAMETER        (LDA = 2*NSUB + 1)
C      PARAMETER        (LDIWRK = N)
C      PARAMETER        (LDWORK = 3 * N)
C
C      DOUBLE PRECISION A(LDA,N), RCOND, WORK(LDWORK)
C      INTEGER          ICOL, INFO, IROW, IWORK(LDIWRK)
C
C      EXTERNAL          DTBCON
C      INTRINSIC          ABS, MAX
C
C      Initialize the array A to store in banded form the 5x5
C      matrix A with two subdiagonals and no superdiagonals
C      shown below.
C
C      1
C      1 -1
C      A = 1 -2 1
C      -3 3 -1
```

```

C          6 -4 1
C
      DATA A / NSUB*8D8, 1.0D0, 1.0D0, 1.0D0,
$          NSUB*8D8, -1.0D0, -2.0D0, -3.0D0,
$          NSUB*8D8, 1.0D0, 3.0D0, 6.0D0,
$          NSUB*8D8, -1.0D0, 4.0D0, 8D8,
$          NSUB*8D8, 1.0D0, 8D8, 8D8 /
C
C      Print the initial values of the arrays.
C
      PRINT 1000
      DO 100, IROW = 1, N
        PRINT 1010, (0.0D0, ICOL = 1, IROW - NSUB - 1),
$          (A(NSUB + 1 + IROW - ICOL, ICOL),
$          ICOL = MAX(1, IROW - NSUB), IROW),
$          (0.0D0, ICOL = IROW + 1, N)
100  CONTINUE
      PRINT 1020
      PRINT 1010, ((A(IROW, ICOL), ICOL = 1, N), IROW = 1, LDA)
C
C      Compute and print the condition number of A.
C
      CALL DTBCON ('ONE-NORM', 'LOWER TRIANGULAR A',
$          'NOT UNIT DIAGONAL A', N, NSUB, A, LDA, RCOND,
$          WORK, IWORK, INFO)
      IF (INFO .NE. 0) THEN
        PRINT 1030, ABS(INFO)
        STOP 1
      END IF
      PRINT 1040, 1.0D0 / RCOND
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (5(3X, F4.1))
1020 FORMAT (/1X, 'A in banded form: (* in unused elements)')
1030 FORMAT (1X, 'Illegal argument to DTBCON, argument #', I2)
1040 FORMAT (/1X, 'Estimated condition number of A = ', F8.4)
C
      END

```

Sample Output

```

A in full form:
  1.0   0.0   0.0   0.0   0.0
  1.0  -1.0   0.0   0.0   0.0
  1.0  -2.0   1.0   0.0   0.0
  0.0  -3.0   3.0  -1.0   0.0
  0.0   0.0   6.0   4.0   1.0

```


A in banded form: (* in unused elements)

```
****  ****  ****  ****  ****
****  ****  ****  ****  ****
  1.0  -1.0   1.0  -1.0   1.0
  1.0  -2.0   3.0   4.0  ****
  1.0  -3.0   6.0  ****  ****
```

Estimated condition number of A = 5.3333

Error Bounds and Estimates for the Solution to a Linear System in a Triangular Matrix in Banded Storage

The subroutines described in this section compute the forward error bounds and backward error estimates for the solution to a linear system $AX = B$, $A^T X = B$, or $A^H X = B$ for a triangular matrix A in banded storage and general matrices B and X . Note that these subroutines do not refine the computed solution as other xxxRFS subroutines do.

Calling Sequence

```
CALL DTBRFS (UPLO, TRANSA, DIAG, N, NDIAG, NRHS, DA, LDA, DB, LDB,
             DX, LDX, DFERR, DBERR, DWORK, IWORK2, INFO)
CALL STBRFS (UPLO, TRANSA, DIAG, N, NDIAG, NRHS, SA, LDA, SB, LDB,
             SX, LDX, SFERR, SBERR, SWORK, IWORK2, INFO)
CALL ZTBRFS (UPLO, TRANSA, DIAG, N, NDIAG, NRHS, ZA, LDA, ZB, LDB,
             ZX, LDX, DFERR, DBERR, ZWORK, DWORK2, INFO)
CALL CTBRFS (UPLO, TRANSA, DIAG, N, NDIAG, NRHS, CA, LDA, CB, LDB,
             CX, LDX, SFERR, SBERR, CWORK, SWORK2, INFO)

void dtbrfs (char uplo, char trans, char diag, int n, int ndiag,
             int nrhs, double *da, int lda, double *db, int ldb,
             double *dx, int ldx, double *dferr, double *dberr, int
             *info)
void stbrfs (char uplo, char trans, char diag, int n, int ndiag,
             int nrhs, float *sa, int lda, float *sb, int ldb, float
             *sx, int ldx, float *sferr, float *sberr, int *info)
void ztbrfs (char uplo, char trans, char diag, int n, int ndiag,
             int nrhs, doublecomplex *za, int lda, doublecomplex
             *zb, int ldb, doublecomplex *zx, int ldx, double
             *dferr, double *dberr, int *info)
void ctbrfs (char uplo, char trans, char diag, int n, int ndiag,
             int nrhs, complex *ca, int lda, complex *cb, int ldb,
             complex *cx, int ldx, float *sferr, float *sberr, int
             *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle.

	'L' or 'l'	xA contains the lower triangle.
TRANSA	Indicates the form of the system of equations. The legal values for TRANSA are listed below. Any values not listed below are illegal.	
	'N' or 'n'	No transpose, use $\text{AX} = \text{B}$.
	'T' or 't'	Transpose, use $\text{A}^T \text{X} = \text{B}$.
	'C' or 'c'	Conjugate transpose, use $\text{A}^H \text{X} = \text{B}$.
	Note that A^T and A^H are the same for real matrices.	
DIAG	Indicates whether or not A is unit triangular. The legal values for DIAG are listed below. Any values not listed below are illegal.	
	'N' or 'n'	A is not unit triangular.
	'U' or 'u'	A is unit triangular.
N	Order of the matrix A. $N \geq 0$.	
NDIAG	Number of superdiagonals or subdiagonals of the triangular matrix A. $N-1 \geq \text{NDIAG} \geq 0$ but if $N = 0$ then $\text{NDIAG} = 0$.	
NRHS	Number of right-hand sides, equal to the number of columns of the matrices B and X. $\text{NRHS} \geq 0$.	
xA	Upper or lower triangular matrix A. If $\text{DIAG} = \text{'U'}$ or 'u' , the diagonal elements of A are assumed to be 1 and are not used.	
LDA	Leading dimension of the array A as specified in a dimension or type statement. $\text{LDA} \geq \text{NDIAG} + 1$.	
xB	The $N \times \text{NRHS}$ right-hand side matrix B.	
LDB	Leading dimension of the array B as specified in a dimension or type statement. $\text{LDB} \geq \max(1, N)$.	
X	The $N \times \text{NRHS}$ solution matrix X.	
LDX	Leading dimension of the array X as specified in a dimension or type statement. $\text{LDX} \geq \max(1, N)$.	
xFERR	On exit, the estimated forward error bound for each solution vector $\text{X}(*, j)$ for $1 \leq j \leq \text{NRHS}$. If X' is the true solution corresponding to $\text{X}(*, j)$ then $\text{FERR}(j)$ is an upper bound on the magnitude of the largest element in $\text{X}(*, j) - \text{X}'$ divided by the magnitude of the largest element in $\text{X}(*, j)$.	
xBERR	On exit, $\text{BERR}(j)$ is the smallest relative change in any element of A or $\text{B}(*, j)$ that makes $\text{X}(*, j)$ an exact solution to $\text{AX}(*, j) = \text{B}(*, j)$ for $1 \leq j \leq \text{NRHS}$.	
xWORK	Scratch array with a dimension of $3 \times N$ for real subroutines or $2 \times N$ for complex subroutines.	
xWORK2	Scratch array with a dimension of N.	
INFO	On exit:	
	$\text{INFO} = 0$	Subroutine completed normally.
	$\text{INFO} < 0$	The i th argument, where $i = \text{INFO} $, had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER          LDA, LDB, LDIWRK, LDWORK, LDX, N, NRHS
  INTEGER          NDIAG
  PARAMETER        (N = 4)
  PARAMETER        (NDIAG = 1)
  PARAMETER        (LDA = NDIAG + 1)
  PARAMETER        (LDB = N)
  PARAMETER        (LDIWRK = N)
  PARAMETER        (LDWORK = 3 * N)
  PARAMETER        (LDX = N)
  PARAMETER        (NRHS = 1)

C
  DOUBLE PRECISION A(LDA,N), B(LDB,NRHS), B1(LDB,NRHS)
  DOUBLE PRECISION BERR(NRHS), EPSLON, FERR(NRHS)
  DOUBLE PRECISION WORK(LDWORK), X(LDX,NRHS)
  INTEGER          ICOL, INFO, IROW, IWORK(LDIWRK)

C
  EXTERNAL          DCOPY, DTBMV, DTBRFS, DTBTRS
  INTRINSIC         MAX, SQRT

C
C  Initialize the array A to store in banded form the matrix A
C  shown below. Initialize the array B to store the right hand
C  side vector b shown below.
C
C
C      0          3
C      A = 9      0          b = 5
C              0.25    5          7
C              7      0          11
C
C      DATA A / 0.0D0, 9.0D0, 0.0D0, 2.5D-1, 5.0D0, 7.0D0, 0.0D0,
C      $          8D8 /
C      DATA B / 3.0D0, 5.0D0, 7.0D0, 1.1D1 /

C
C  Add a small value to the elements of A on the diagonal
C  and on the subdiagonal. After this loop, A will contain
C  something similar to:
C
C
C      e
C      A = 9+e      e      5+e
C              0.25+e      e      7+e
C
C
C      EPSLON = (((((2.0D0 / 3.0D0) + 2.0D0) - 2.0D0) -
C      $          (2.0D0 / 3.0D0))
C      DO 110, ICOL = 1, N
C        DO 100, IROW = 1, NDIAG + 1
C          A(IROW,ICOL) = A(IROW,ICOL) + EPSLON
C          A(1,ICOL) = A(1,ICOL) + EPSLON
100    CONTINUE
110   CONTINUE

```

```

      A(LDA,N) = SQRT(-1.0D0)
C
C      Print the A array.
C
      PRINT 1000
      DO 120, IROW = 1, N
        PRINT 1010, (0.0D0, ICOL = 1, IROW - NDIAG - 1),
$              (A(1 + IROW - ICOL, ICOL),
$              ICOL = MAX(1,IROW - NDIAG), IROW),
$              (0.0D0, ICOL = IROW + 1, N)
120 CONTINUE
      PRINT 1020
      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, LDA)
C
C      Make a copy of B and then print B.
C
      CALL DCOPY (N, B, 1, X, 1)
      PRINT 1030
      PRINT 1040, B
C
C      Solve Ax=b and print the solution.
C
      CALL DTBTRS ('LOWER TRIANGULAR A', 'NO TRANSPOSE A',
$              'NOT UNIT DIAGONAL A', N, NDIAG, NRHS, A,
$              LDA, X, LDX, INFO)
      IF (INFO .NE. 0) THEN
        PRINT 1060, INFO
        STOP 2
      END IF
      PRINT 1070
      PRINT 1080, X
      CALL DCOPY (N, X, 1, B1, 1)
      CALL DTBMV ('LOWER TRIANGULAR A', 'NO TRANSPOSE A',
$              'NOT UNIT DIAGONAL A', N, NDIAG, A, LDA, B1, 1)
      PRINT 1090
      PRINT 1040, B1
C
C      Refine the initial solution and print the new solution.
C
      CALL DTBRFS ('LOWER TRIANGULAR A', 'NO TRANSPOSE A',
$              'NOT UNIT TRIANGULAR A', N, NDIAG, NRHS,
$              A, LDA, B, LDB, X, LDX, FERR, BERR, WORK,
$              IWORK, INFO)
      IF (INFO .EQ. 0) THEN
        PRINT 1100, (IROW, BERR(IROW), IROW = 1, NRHS)
        PRINT 1110, (IROW, FERR(IROW), IROW = 1, NRHS)
      ELSE
        PRINT 1120, INFO
        STOP 3
      END IF
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(2X, E14.8))
1020 FORMAT (/1X, 'A in banded form: ("NaN" in unused elements)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (1X, F20.17)

```

```

1050 FORMAT (1X, 'Error factoring A in DGBTRF, INFO = ', I5)
1060 FORMAT (1X, 'Error solving Ax=b in DGBTRF, INFO = ', I5)
1070 FORMAT (/1X, 'Initial solution for Ax=b:')
1080 FORMAT (1X, E22.16)
1090 FORMAT (/1X, 'Ax with initial x:')
1110 FORMAT (/1X, 'Backward error bound for system #', I2, ' = ',
$          E15.8)
1100 FORMAT (1X, 'Forward error bound for system #', I2, ' = ',
$          E15.8)
1120 FORMAT (1X, 'Error estimating error bounds, INFO = ', I5)
C
      END

```

Sample Output

```

A in full form:
-.33306691E-15  0.00000000E+00  0.00000000E+00  0.00000000E+00
0.90000000E+01 -.33306691E-15  0.00000000E+00  0.00000000E+00
0.00000000E+00  0.25000000E+00  0.50000000E+01  0.00000000E+00
0.00000000E+00  0.00000000E+00  0.70000000E+01 -.33306691E-15

A in banded form: ("NaN" in unused elements)
-.33306691E-15 -.33306691E-15  0.50000000E+01 -.33306691E-15
0.90000000E+01  0.25000000E+00  0.70000000E+01  NaN

b:
3.000000000000000000
5.000000000000000000
7.000000000000000000
11.000000000000000000

Initial solution for Ax=b:
-.9007199254740992E+16
-.2433889152438200E+33
0.1216944576219100E+32
0.2557627865329079E+48

Ax with initial x:
3.000000000000000000
0.000000000000000000
0.000000000000000000
0.000000000000000000

Forward error bound for system # 1 = 0.30839528E-16

Backward error bound for system # 1 = 0.27262143E-14

```

Solution to a Linear System in a Triangular Matrix in Banded Storage

The subroutines described in this section solve a linear system $AX = B$, $A^T X = B$, or $A^H X = B$ for a triangular matrix A in banded storage and general matrices B and X .

Calling Sequence

```
CALL DTBTRS (UPLO, TRANSA, DIAG, N, NDIAG, NRHS, DA, LDA, DB, LDB,
             INFO)
CALL STBTRS (UPLO, TRANSA, DIAG, N, NDIAG, NRHS, SA, LDA, SB, LDB,
             INFO)
CALL ZTBTRS (UPLO, TRANSA, DIAG, N, NDIAG, NRHS, ZA, LDA, ZB, LDB,
             INFO)
CALL CTBTRS (UPLO, TRANSA, DIAG, N, NDIAG, NRHS, CA, LDA, CB, LDB,
             INFO)

void dtbtrs (char uplo, char trans, char diag, int n, int ndiag,
            int nrhs, double *da, int lda, double *db, int ldb, int
            *info)
void stbtrs (char uplo, char trans, char diag, int n, int ndiag,
            int nrhs, float *sa, int lda, float *sb, int ldb, int
            *info)
void ztbtrs (char uplo, char trans, char diag, int n, int ndiag,
            int nrhs, doublecomplex *za, int lda, doublecomplex
            *zb, int ldb, int *info)
void ctbtrs (char uplo, char trans, char diag, int n, int ndiag,
            int nrhs, complex *ca, int lda, complex *cb, int ldb,
            int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
TRANSA	Indicates the form the system of equations. The legal values for TRANSA are listed below. Any values not listed below are illegal. 'N' or 'n' No transpose, solve $AX = B$. 'T' or 't' Transpose, solve $A^T X = B$. 'C' or 'c' Conjugate transpose, solve $A^H X = B$.

	Note that A^T and A^H are the same for real matrices.
DIAG	Indicates whether or not A is unit triangular. The legal values for DIAG are listed below. Any values not listed below are illegal. 'N' or 'n' A is not unit triangular. 'U' or 'u' A is unit triangular.
N	Order of the matrix A. $N \geq 0$.
NDIAG	Number of superdiagonals or subdiagonals of the triangular matrix A. $N-1 \geq \text{NDIAG} \geq 0$ but if $N = 0$ then $\text{NDIAG} = 0$.
NRHS	Number of right-hand sides, equal to the number of columns of the matrix B. $\text{NRHS} \geq 0$.
xA	Upper or lower triangular matrix A. If $\text{DIAG} = 'U'$ or $'u'$, the diagonal elements of A are assumed to be 1 and are not used.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $\text{LDA} \geq \text{NDIAG} + 1$.
xB	On entry, the $N \times \text{NRHS}$ right-hand side matrix B. On exit, the $N \times \text{NRHS}$ solution matrix X.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $\text{LDB} \geq \max(1, N)$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The i th argument, where $i = \text{INFO} $, had an illegal value. INFO > 0 The i th diagonal element, where $i = \text{INFO}$, of A is zero. The matrix is therefore singular and the solution could not be computed.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C      INTEGER          LDA, LDB, N, NRHS, NSUB
C      PARAMETER        (NSUB = 1)
C      PARAMETER        (N = 4)
C      PARAMETER        (NRHS = 1)
C      PARAMETER        (LDA = NSUB + 1)
C      PARAMETER        (LDB = N)
C
C      DOUBLE PRECISION A(LDA,N), B(LDB,NRHS)
C      INTEGER          ICOL, INFO, IROW
C
C      EXTERNAL          DTBTRS
C      INTRINSIC         MAX
C
C      Initialize the array A to store in banded triangular form

```



```

C      the 4x4 coefficient matrix A with one subdiagonal as shown
C      below. Initialize the array B to store the right hand side
C      vector b shown below.
C
C      A =  $\begin{bmatrix} 2 & & & \\ -1 & 2 & & \\ & -1 & 2 & \\ & & -1 & 2 \end{bmatrix}$       b =  $\begin{bmatrix} 8 \\ 8 \\ 6 \\ 4 \end{bmatrix}$ 
C
C      DATA A / 2.0D0, -1.0D0, 2.0D0, -1.0D0, 2.0D0, -1.0D0, 2.0D0,
$          8D8 /
C      DATA B / 8.0D0, 8.0D0, 6.0D0, 4.0D0 /
C
C      Print the initial values of the arrays.
C
C      PRINT 1000
C      DO 100, IROW = 1, N
C          PRINT 1010, (0.0D0, ICOL = 1, IROW - NSUB - 1),
$              (A(1 + IROW - ICOL, ICOL),
$              ICOL = MAX(1,IROW - NSUB), IROW),
$              (0.0D0, ICOL = IROW + 1, N)
100 CONTINUE
C      PRINT 1020
C      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, LDA)
C      PRINT 1030
C      PRINT 1040, B
C
C      Solve the system and print the results.
C
C      CALL DTBTRS ('LOWER TRIANGULAR A', 'NO TRANSPOSE A',
$          'NOT UNIT DIAGONAL A', N, NSUB, NRHS, A, LDA,
$          B, LDB, INFO)
C      IF (INFO .EQ. 0) THEN
C          PRINT 1050
C          PRINT 1040, B
C      ELSE
C          PRINT 1060, INFO
C      END IF
C
C      1000 FORMAT (1X, 'A in full form:')
C      1010 FORMAT (4(3X, F4.1))
C      1020 FORMAT (/1X, 'A in banded form: (* in unused elements)')
C      1030 FORMAT (/1X, 'b:')
C      1040 FORMAT (3X, F4.1)
C      1050 FORMAT (/1X, 'x:')
C      1060 FORMAT (1X, 'Error in solving system, INFO = ', I4)
C
C      END

```

Sample Output

```
A in full form:
  2.0   0.0   0.0   0.0
 -1.0   2.0   0.0   0.0
  0.0  -1.0   2.0   0.0
  0.0   0.0  -1.0   2.0

A in banded form: (* in unused elements)
  2.0   2.0   2.0   2.0
 -1.0  -1.0  -1.0  ****

b:
  8.0
  8.0
  6.0
  4.0

x:
  4.0
  6.0
  6.0
  5.0
```

Reciprocal Condition Number of a Triangular Matrix in Packed Storage

The subroutines described in this section estimate the reciprocal condition number, in either the 1-norm or the ∞ -norm, of a triangular matrix A in packed storage.

Calling Sequence

```
CALL DTPCON (NORM, UPLO, DIAG, N, DA, DRCOND, DWORK, IWORK2, INFO)
CALL STPCON (NORM, UPLO, DIAG, N, SA, SRCOND, SWORK, IWORK2, INFO)
CALL ZTPCON (NORM, UPLO, DIAG, N, ZA, DRCOND, ZWORK, DWORK2, INFO)
CALL CTPCON (NORM, UPLO, DIAG, N, CA, SRCOND, CWORK, SWORK2, INFO)
```

```
void dtpcon (char norm, char uplo, char diag, int n, double *da,
double *drcond, int *info)
void stpcon (char norm, char uplo, char diag, int n, float *sa,
float *srcond, int *info)
void ztpcon (char norm, char uplo, char diag, int n, doublecomplex
*za, double *drcond, int *info)
void ctpcon (char norm, char uplo, char diag, int n, complex *ca,
float *srcond, int *info)
```

Arguments

NORM	Indicates whether the 1-norm condition number or the ∞ -norm condition number is required. The legal values for NORM are listed below. Any values not listed below are illegal. 'O', 'o', or '1' 1-norm condition number is required. 'T' or 't' ∞ -norm condition number is required.
UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
DIAG	Indicates whether or not A is unit triangular. The legal values for DIAG are listed below. Any values not listed below are illegal. 'N' or 'n' A is not unit triangular. 'U' or 'u' A is unit triangular.
N	Order of the matrix A. $N \geq 0$.
xA	Upper or lower triangular matrix A. The dimension of xA is $(N \times N + N) / 2$.

xRCOND	On exit, the estimated reciprocal of the condition number of the matrix A where the reciprocal condition number of A is defined to be $1 / (\ A\ \times \ A^{-1}\)$. The reciprocal of the condition number is estimated instead of the condition number itself to avoid overflow or division by zero. If RCOND is less than machine precision (in particular, if RCOND = 0) then A is singular to working precision.
xWORK	Scratch array with a dimension of $3 \times N$ for real subroutines or $2 \times N$ for complex subroutines.
xWORK2	Scratch array with a dimension of N.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where i = INFO , had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
DOUBLE PRECISION  ZERO
INTEGER            LDA, LDIWRK, LDWORK, N
PARAMETER          (N = 3)
PARAMETER          (LDA = ((N + 1) * N) / 2)
PARAMETER          (LDIWRK = N)
PARAMETER          (LDWORK = 3 * N)
PARAMETER          (ZERO = 0.0D0)
C
DOUBLE PRECISION  A(LDA), RCOND, WORK(LDWORK)
INTEGER           INFO, IWORK(LDIWRK)
C
EXTERNAL          DTPCON
INTRINSIC         ABS
C
C Initialize A to store in packed triangular form the matrix A
C shown below.
C
C      1  2  4
C  A = 0  3  5
C      0  0  6
C
DATA A / 1.0D0, 2.0D0, 3.0D0, 2.0D0, 3.0D0, 3.0D0 /
C
PRINT 1000
PRINT 1010, A(1), A(2), A(4)
PRINT 1010, ZERO, ZERO, A(5)
PRINT 1010, ZERO, A(5), A(6)
C
CALL DTPCON ('ONE-NORM OF A', 'UPPER TRIANGULAR A',
$           'NO UNIT DIAGONAL A', N, A, RCOND, WORK, IWORK,

```

```
      $          INFO)
      IF (INFO .NE. 0) THEN
        PRINT 1020, ABS(INFO)
        STOP 1
      END IF
      PRINT 1030, 1.0D0 / RCOND
C
1000 FORMAT ('A:')
1010 FORMAT (3(3X, F5.2))
1020 FORMAT (1X, 'Illegal argument to DTPCON, argument #', I2)
1030 FORMAT (/1X, 'Estimated condition number of A: ', F7.4)
C
      END
```

Sample Output

```
A:
   1.00    2.00    2.00
   0.00    0.00    3.00
   0.00    3.00    3.00

Estimated condition number of A:  8.00000
```

Error Bounds and Estimates for the Solution of a Linear System in a Triangular Matrix in Packed Storage

The subroutines described in this section compute forward error bounds and backward error estimates for the solution to a linear system $AX = B$, $A^T X = B$, or $A^H X = B$ for a triangular matrix A and general matrices B and X . Note that these subroutines do not refine the computed solution as other xxxRFS subroutines do.

Calling Sequence

```
CALL DTPRFS (UPLO, TRANSA, DIAG, N, NRHS, DA, DB, LDB, DX, LDX,
             DFERR, DBERR, DWORK, IWORK2, INFO)
CALL STPRFS (UPLO, TRANSA, DIAG, N, NRHS, SA, SB, LDB, SX, LDX,
             SFERR, SBERR, SWORK, IWORK2, INFO)
CALL ZTPRFS (UPLO, TRANSA, DIAG, N, NRHS, ZA, ZB, LDB, ZX, LDX,
             DFERR, DBERR, ZWORK, DWORK2, INFO)
CALL CTPRFS (UPLO, TRANSA, DIAG, N, NRHS, CA, CB, LDB, CX, LDX,
             SFERR, SBERR, CWORK, SWORK2, INFO)

void dtprfs (char uplo, char trans, char diag, int n, int nrhs,
             double *da, double *db, int ldb, double *dx, int ldx,
             double *dferr, double *dberr, int *info)
void stprfs (char uplo, char trans, char diag, int n, int nrhs,
             float *sa, float *sb, int ldb, float *sx, int ldx,
             float *sferr, float *sberr, int *info)
void ztprfs (char uplo, char trans, char diag, int n, int nrhs,
             doublecomplex *za, doublecomplex *zb, int ldb,
             doublecomplex *zx, int ldx, double *dferr, double
             *dberr, int *info)
void ctprfs (char uplo, char trans, char diag, int n, int nrhs,
             complex *ca, complex *cb, int ldb, complex *cx, int
             ldx, float *sferr, float *sberr, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal.
'U' or 'u'	xA contains the upper triangle.
'L' or 'l'	xA contains the lower triangle.

TRANSA	Indicates the form of the system of equations. The legal values for TRANSA are listed below. Any values not listed below are illegal. 'N' or 'n' No transpose, use $AX = B$. 'T' or 't' Transpose, use $A^T X = B$. 'C' or 'c' Conjugate transpose, use $A^H X = B$. Note that A^T and A^H are the same for real matrices.
DIAG	Indicates whether or not A is unit triangular. The legal values for DIAG are listed below. Any values not listed below are illegal. 'N' or 'n' A is not unit triangular. 'U' or 'u' A is unit triangular.
N	Order of the matrix A. $N \geq 0$.
NRHS	Number of right-hand sides, equal to the number of columns of the matrices B and X. $NRHS \geq 0$.
xA	Upper or lower triangular matrix A. The dimension of xA is $(N \times N + N) / 2$. If $DIAG = 'U'$ or $'u'$, the diagonal elements of A are assumed to be 1 and are not used.
xB	The $N \times NRHS$ right-hand side matrix B.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
xX	The $N \times NRHS$ solution matrix X.
LDX	Leading dimension of the array X as specified in a dimension or type statement. $LDX \geq \max(1, N)$.
xFERR	On exit, the estimated forward error bound for each solution vector $X(*, j)$ for $1 \leq j \leq NRHS$. If X' is the true solution corresponding to $X(*, j)$ then $FERR(j)$ is an upper bound on the magnitude of the largest element in $X(*, j) - X'$ divided by the magnitude of the largest element in $X(*, j)$.
xBERR	On exit, $BERR(j)$ is the smallest relative change in any element of A or $B(*, j)$ that makes $X(*, j)$ an exact solution to $AX(*, j) = B(*, j)$ for $1 \leq j \leq NRHS$.
xWORK	Scratch array with a dimension of $3 \times N$ for real subroutines or $2 \times N$ for complex subroutines.
xWORK2	Scratch array with a dimension of N.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The i th argument, where $i = INFO $, had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
DOUBLE PRECISION ZERO
INTEGER LDB, LDIWRK, LDWORK, LDX, LENGTA, N, NRHS
PARAMETER (N = 4)
PARAMETER (LDB = N)
PARAMETER (LDIWRK = N)
PARAMETER (LDWORK = 3 * N)
PARAMETER (LDX = LDB)
PARAMETER (LENGTA = (N * N + N) / 2)
PARAMETER (NRHS = 1)
PARAMETER (ZERO = 0.0D0)

C
DOUBLE PRECISION A(LENGTA), B(LDB,NRHS), BERR(NRHS), EPSLON
DOUBLE PRECISION FERR(NRHS), WORK(LDWORK), X(LDX,NRHS)
INTEGER I, INFO, IWORK(LDIWRK)

C
EXTERNAL DCPY, DTPMV, DTPRFS, DTPTRS
INTRINSIC ABS, DBLE

C
C Initialize the array A to store in banded form the matrix A
C shown below. Initialize the array B to store the right hand
C side vector b shown below.
C
C      1
C      A = 1      0      0      0      b = 4
C      0      1      0      0      4
C      0      0      1      0      4
C
C      DATA A / 1.0D0, 1.0D0, 0.0D0, 0.0D0, 0.0D0, 1.0D0, 0.0D0,
$      0.0D0, 1.0D0, 0.0D0 /
C      DATA B / N*4.0D0 /

C
C Add a small value to the elements of A on and below the
C diagonal. After this loop, A will contain something
C similar to:
C
C      1+e
C      A = 1+e      e
C      e      1+e      e
C      e      e      1+e      e
C
C      EPSLON = (((((2.0D0 / 3.0D0) + 1.6D1) - 1.6D1) -
$      (2.0D0 / 3.0D0))
C      DO 100, I = 1, LENGTA
C      A(I) = A(I) + EPSLON
100 CONTINUE

C
C Print the initial value of A.
C
PRINT 1000

```



```

      PRINT 1010, A(1), ZERO, ZERO, ZERO
      PRINT 1010, A(2), A(5), ZERO, ZERO
      PRINT 1010, A(3), A(6), A(8), ZERO
      PRINT 1010, A(4), A(7), A(9), A(10)
C
C      Make a copy of B and then print B.
C
      CALL DCOPY (LDB * NRHS, B, 1, X, 1)
      PRINT 1020
      PRINT 1030, B
C
C      Solve Ax=b and print the solution.
C
      CALL DTPTRS ('LOWER TRIANGULAR A', 'NO TRANSPOSE A',
$              'NO UNIT DIAGONAL A', N, NRHS, A, X, LDX, INFO)
      IF (INFO .NE. 0) THEN
          PRINT 1040, INFO
          STOP 1
      END IF
      PRINT 1050
      PRINT 1060, X
C
C      Estimate the error bounds for the computed solution.
C
      CALL DTPRFS ('LOWER TRIANGULAR A', 'NO TRANSPOSE A',
$              'NO UNIT DIAGONAL A', N, NRHS, A, B, LDB,
$              X, LDX, FERR, BERR, WORK, IWORK, INFO)
      IF (INFO .NE. 0) THEN
          PRINT 1070, ABS(INFO)
          STOP 2
      END IF
      CALL DTPMV ('LOWER TRIANGULAR A', 'NO TRANSPOSE A',
$              'NO UNIT DIAGONAL A', N, A, X, 1)
      PRINT 1080
      PRINT 1030, X
      PRINT 1090, (I, BERR(I), I = 1, NRHS)
      PRINT 1100, (I, FERR(I), I = 1, NRHS)
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (4(2X, F17.15))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (1X, F21.18)
1040 FORMAT (1X, 'Error solving Ax=b, INFO = ', I5)
1050 FORMAT (/1X, 'Initial solution for Ax=b:')
1060 FORMAT (1X, E16.8)
1070 FORMAT (1X, 'Illegal argument to DTPRFS, argument #', I5)
1080 FORMAT (/1X, 'Ax:')
1090 FORMAT (/1X, 'Backward error bound for system #', I2, ' = ',
$          E14.7)
1100 FORMAT (1X, 'Forward error bound for system #', I2, ' = ',
$          E14.7)
C
      END

```

Sample Output

```
A:
1.0000000000000001 0.0000000000000000 0.0000000000000000 0.0000000000000000
1.0000000000000001 0.0000000000000001 0.0000000000000000 0.0000000000000000
0.0000000000000001 1.0000000000000001 0.0000000000000001 0.0000000000000000
0.0000000000000001 0.0000000000000001 1.0000000000000001 0.0000000000000001

b:
4.0000000000000000
4.0000000000000000
4.0000000000000000
4.0000000000000000

Initial solution for Ax=b:
0.40000000E+01
0.00000000E+00
0.32753452E+16
-0.26819715E+31

Ax:
4.0000000000000000
4.0000000000000000
4.0000000000000000
4.000000000000004441

Backward error bound for system # 1 = 0.6779273E-30
Forward error bound for system # 1 = 0.1818182E+01
```

Inverse of a Triangular Matrix in Packed Storage

The subroutines described in this section compute the inverse of a triangular matrix A in packed storage.

Calling Sequence

```
CALL DTPTRI (UPLO, DIAG, N, DA, INFO)
CALL STPTRI (UPLO, DIAG, N, SA, INFO)
CALL ZTPTRI (UPLO, DIAG, N, ZA, INFO)
CALL CTPTRI (UPLO, DIAG, N, CA, INFO)

void dtptri (char uplo, char diag, int n, double *da, int *info)
void stptri (char uplo, char diag, int n, float *sa, int *info)
void ztptri (char uplo, char diag, int n, doublecomplex *za, int
            *info)
void ctptri (char uplo, char diag, int n, complex *ca, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
DIAG	Indicates whether or not A is unit triangular. The legal values for DIAG are listed below. Any values not listed below are illegal. 'N' or 'n' A is not unit triangular. 'U' or 'u' A is unit triangular.
N	Order of the matrix A. $N \geq 0$.
xA	On entry, the upper or lower triangular matrix A. The dimension of xA is $(N \times N + N) / 2$. On exit, the triangular inverse of A.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The <i>i</i> th argument, where $i = INFO $, had an illegal value. INFO > 0 A(<i>i</i> , <i>i</i>), where $i = INFO$, is exactly zero. The matrix is therefore singular and its inverse could not be computed.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
DOUBLE PRECISION  ZERO
INTEGER           LDA, N
PARAMETER        (N = 4)
PARAMETER        (LDA = ((N + 1) * N) / 2)
PARAMETER        (ZERO = 0.0D0)

C
DOUBLE PRECISION  A(LDA)
INTEGER           INFO

C
EXTERNAL  DTPTRI

C
C Initialize the array A to store in packed triangular form
C the matrix A shown below.
C
C
C      1
C      A = 1   1
C           1   1   1
C           1   1   1   1
C
DATA A / LDA*1.0D0 /

C
C Print the A array.
C
PRINT 1000
PRINT 1010, A(1), ZERO, ZERO, ZERO
PRINT 1010, A(2), A(5), ZERO, ZERO
PRINT 1010, A(3), A(6), A(8), ZERO
PRINT 1010, A(4), A(7), A(9), A(10)

C
C Compute and print A**(-1).
C
CALL DTPTRI ('LOWER TRIANGULAR A', 'NO UNIT DIAGONAL A',
$           N, A, INFO)
IF (INFO.NE. 0) THEN
    PRINT 1020, INFO
    STOP 1
END IF
PRINT 1030
PRINT 1010, A(1), ZERO, ZERO, ZERO
PRINT 1010, A(2), A(5), ZERO, ZERO
PRINT 1010, A(3), A(6), A(8), ZERO
PRINT 1010, A(4), A(7), A(9), A(10)

C
1000 FORMAT (1X, 'A:')
1010 FORMAT (4(3X, F5.2))
1020 FORMAT (/1X, 'Error inverting A, INFO = ', I5)
1030 FORMAT (/1X, 'A**(-1):')

```

```
C
      END
```

Sample Output

```
A:
  1.00    0.00    0.00    0.00
  1.00    1.00    0.00    0.00
  1.00    1.00    1.00    0.00
  1.00    1.00    1.00    1.00

A**(-1):
  1.00    0.00    0.00    0.00
 -1.00    1.00    0.00    0.00
  0.00   -1.00    1.00    0.00
  0.00    0.00   -1.00    1.00
```

Solution to a Linear System in a Triangular Matrix in Packed Storage

The subroutines described in this section compute solve a linear system $AX = B$, $A^T X = B$, or $A^H X = B$ for a triangular matrix A in packed storage and general matrices B and X .

Calling Sequence

```
CALL DTPTRS (UPLO, TRANSA, DIAG, N, NRHS, DA, DB, LDB, INFO)
CALL STPTRS (UPLO, TRANSA, DIAG, N, NRHS, SA, SB, LDB, INFO)
CALL ZTPTRS (UPLO, TRANSA, DIAG, N, NRHS, ZA, ZB, LDB, INFO)
CALL CTPTRS (UPLO, TRANSA, DIAG, N, NRHS, CA, CB, LDB, INFO)

void dtptrs (char uplo, char trans, char diag, int n, int nrhs,
             double *da, double *db, int ldb, int *info)
void stptrs (char uplo, char trans, char diag, int n, int nrhs,
             float *sa, float *sb, int ldb, int *info)
void ztptrs (char uplo, char trans, char diag, int n, int nrhs,
             doublecomplex *za, doublecomplex *zb, int ldb, int
             *info)
void ctptrs (char uplo, char trans, char diag, int n, int nrhs,
             complex *ca, complex *cb, int ldb, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
TRANSA	Indicates the form of the system of equations. The legal values for TRANSA are listed below. Any values not listed below are illegal. 'N' or 'n' No transpose, solve $AX = B$. 'T' or 't' Transpose, solve $A^T X = B$. 'C' or 'c' Conjugate transpose, solve $A^H X = B$. Note that A^T and A^H are the same for real matrices.
DIAG	Indicates whether or not A is unit triangular. The legal values for DIAG are listed below. Any values not listed below are illegal. 'N' or 'n' A is not unit triangular. 'U' or 'u' A is unit triangular.
N	Order of the matrix A . $N \geq 0$.

NRHS	Number of right-hand sides, equal to the number of columns of the matrix B. $\text{NRHS} \geq 0$.
xA	Upper or lower triangular matrix A. The dimension of xA is $(N \times N + N) / 2$.
xB	On entry, the $N \times \text{NRHS}$ right-hand side matrix B. On exit, the $N \times \text{NRHS}$ solution matrix X.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $\text{LDB} \geq \max(1, N)$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where $i = \text{INFO}$, had an illegal value. INFO > 0 The ith diagonal element, where $i = \text{INFO}$, of A is zero. The matrix is therefore singular and the solution could not be computed.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C   DOUBLE PRECISION  ZERO
C   INTEGER           LDA, LDB, N, NRHS
C   PARAMETER         (N = 4)
C   PARAMETER         (LDA = ((N + 1) * N) / 2)
C   PARAMETER         (LDB = N)
C   PARAMETER         (NRHS = 1)
C   PARAMETER         (ZERO = 0.0D0)
C
C   DOUBLE PRECISION  A(LDA), B(LDB,NRHS)
C   INTEGER           INFO
C
C   EXTERNAL          DTPTRS
C
C   Initialize the array A to store in packed triangular
C   form the coefficient matrix A shown below. Store A in
C   such a way as to take advantage of the fact that A is
C   unit diagonal. Initialize the array B to store the right
C   hand side vector b shown below.
C
C   A =
C   1      1      1      1
C   1      1      1      1
C   1      1      1      1
C   1      1      1      1
C
C   b =
C   4
C   3
C   2
C   1
C
C   DATA A / 8D8, 1.0D0, 1.0D0, 1.0D0, 8D8, 1.0D0, 1.0D0,
$           8D8, 1.0D0, 8D8 /
C   DATA B / 4.0D0, 3.0D0, 2.0D0, 1.0D0 /

```

```

C
C      Print the A array.
C
      PRINT 1000
      PRINT 1010, A(1)
      PRINT 1010, A(2), A(5)
      PRINT 1010, A(3), A(6), A(8)
      PRINT 1010, A(4), A(7), A(9), A(10)
      PRINT 1020
      PRINT 1030, B
C
C      Solve Ax=b and print the result.
C
      CALL DTPTRS ('LOWER TRIANGULAR A', 'TRANSPPOSE A',
$                'UNIT DIAGONAL A', N, NRHS, A, B, LDB, INFO)
      IF (INFO .NE. 0) THEN
          PRINT 1040, INFO
          STOP 1
      END IF
      PRINT 1050
      PRINT 1030, B
C
1000 FORMAT (1X, 'Unit diagonal A: ',
$          '(* in entries assumed to be 1.0)')
1010 FORMAT (4(3X, F8.4))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (1X, F8.4)
1040 FORMAT (/1X, 'Error solving Ax=b, INFO = ', I5)
1050 FORMAT (/1X, 'x:')
C
      END

```

Sample Output

```

Unit diagonal A: (* in entries assumed to be 1.0)
*****
      1.0000      *****
      1.0000      1.0000      *****
      1.0000      1.0000      1.0000      *****

b:
      4.0000
      3.0000
      2.0000
      1.0000

x:
      1.0000
      1.0000
      1.0000
      1.0000

```


Reciprocal Condition Number of a Triangular Matrix

The subroutines described in this section estimate the reciprocal condition number, in either the 1-norm or the ∞ -norm, of a triangular matrix A.

Calling Sequence

```
CALL DTRCON (NORM, UPLO, DIAG, N, DA, LDA, DRCOND, DWORK, IWORK2,
            INFO)
CALL STRCON (NORM, UPLO, DIAG, N, SA, LDA, SRCOND, SWORK, IWORK2,
            INFO)
CALL ZTRCON (NORM, UPLO, DIAG, N, ZA, LDA, DRCOND, ZWORK, DWORK2,
            INFO)
CALL CTRCON (NORM, UPLO, DIAG, N, CA, LDA, SRCOND, CWORK, SWORK2,
            INFO)

void dtrcon (char norm, char uplo, char diag, int n, double *da,
            int lda, double *drcond, int *info)
void strcon (char norm, char uplo, char diag, int n, float *sa, int
            lda, float *srcond, int *info)
void ztrcon (char norm, char uplo, char diag, int n, doublecomplex
            *za, int lda, double *drcond, int *info)
void ctrcon (char norm, char uplo, char diag, int n, complex *ca,
            int lda, float *srcond, int *info)
```

Arguments

NORM	Indicates whether the 1-norm condition number or the ∞ -norm condition number is required. The legal values for NORM are listed below. Any values not listed below are illegal. 'O', 'o', or '1' 1-norm condition number is required. 'T' or 'i' ∞ -norm condition number is required.
UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
DIAG	Indicates whether or not A is unit triangular. The legal values for DIAG are listed below. Any values not listed below are illegal. 'N' or 'n' A is not unit triangular. 'U' or 'u' A is unit triangular.
N	Order of the matrix A. $N \geq 0$.

xA	Upper or lower triangular matrix A. If DIAG = 'U' or 'u', the diagonal elements of A are assumed to be 1 and are not used.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
xRCOND	On exit, the estimated reciprocal of the condition number of the matrix A where the reciprocal condition number of A is defined to be $1 / (\ A\ \times \ A^{-1}\)$. The reciprocal of the condition number is estimated instead of the condition number itself to avoid overflow or division by zero. If RCOND is less than machine precision (in particular, if $RCOND = 0$) then A is singular to working precision.
xWORK	Scratch array with a dimension of $3 \times N$ for real subroutines or $2 \times N$ for complex subroutines.
xWORK2	Scratch array with a dimension of N.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The ith argument, where $i = INFO $, had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
DOUBLE PRECISION ZERO
INTEGER LDA, LDIWRK, LDWORK, N
PARAMETER (N = 3)
PARAMETER (LDA = N)
PARAMETER (LDIWRK = N)
PARAMETER (LDWORK = 3 * N)
PARAMETER (ZERO = 0.0D0)

C
DOUBLE PRECISION A(LDA,N), RCOND, WORK(LDWORK)
INTEGER ICOL, INFO, IROW, IWORK(LDIWRK)

C
EXTERNAL DTRCON
INTRINSIC ABS

C
C Initialize A to store in triangular form the matrix A
C shown below.
C
C      1
C      A = 2  2
C           3  3  3
C
C
DATA A / 1.0D0, 2.0D0, 3.0D0, 8D8, 2.0D0, 3.0D0, 8D8, 8D8,
$      3.0D0 /

```

```

C      Print the initial value of A.
C
C      PRINT 1000
      DO 100, IROW = 1, N
        PRINT 1010, (A(IROW, ICOL), ICOL = 1, IROW),
          $          (ZERO, ICOL = IROW + 1, N)
100 CONTINUE
      PRINT 1020
      DO 110, IROW = 1, LDA
        PRINT 1010, (A(IROW, ICOL), ICOL = 1, N)
110 CONTINUE
C
      CALL DTRCON ('ONE-NORM OF A', 'LOWER TRIANGULAR A',
        $          'NO UNIT DIAGONAL A', N, A, LDA, RCOND,
        $          WORK, IWORK, INFO)
      IF (INFO .NE. 0) THEN
        PRINT 1030, ABS(INFO)
        STOP 1
      END IF
      PRINT 1040, 1.0D0 / RCOND
C
1000 FORMAT ('A in full form:')
1010 FORMAT (3(3X, F19.16))
1020 FORMAT (/1X, 'A in triangular form: ',
  $          ' (* in unused positions)')
1030 FORMAT (1X, 'Illegal argument to DTRCON, argument #', I2)
1040 FORMAT (/1X, 'Estimated condition number of A: ', F7.4)
C
      END

```

Sample Output

```

A in full form:
  1.00000000000000000000  0.00000000000000000000  0.00000000000000000000
  2.00000000000000000000  2.00000000000000000000  0.00000000000000000000
  3.00000000000000000000  3.00000000000000000000  3.00000000000000000000

A in triangular form: (* in unused positions)
  1.00000000000000000000 *****
  2.00000000000000000000 2.000000000000000000 *****
  3.00000000000000000000 3.000000000000000000 3.000000000000000000

Estimated condition number of A: 12.0000

```

Error Bounds and Estimates for the Solution of a Linear System in a Triangular Matrix

The subroutines described in this section compute forward error bounds and backward error estimates for the solution to a linear system $AX = B$, $A^T X = B$, or $A^H X = B$ for a triangular matrix A and general matrices B and X . Note that these subroutines do not refine the computed solution as other xxxRFS subroutines do.

Calling Sequence

```
CALL DTRRFS (UPLO, TRANSA, DIAG, N, NRHS, DA, LDA, DB, LDB, DX,
             LDX, DFERR, DBERR, DWORK, IWORK2, INFO)
CALL STRRFS (UPLO, TRANSA, DIAG, N, NRHS, SA, LDA, SB, LDB, SX,
             LDX, SFERR, SBERR, SWORK, IWORK2, INFO)
CALL ZTRRFS (UPLO, TRANSA, DIAG, N, NRHS, ZA, LDA, ZB, LDB, ZX,
             LDX, DFERR, DBERR, ZWORK, DWORK2, INFO)
CALL CTRRFS (UPLO, TRANSA, DIAG, N, NRHS, CA, LDA, CB, LDB, CX,
             LDX, SFERR, SBERR, CWORK, SWORK2, INFO)

void dtrrfs (char uplo, char trans, char diag, int n, int nrhs,
             double *da, int lda, double *db, int ldb, double *dx,
             int ldx, double *dferr, double *dberr, int *info)
void strrfs (char uplo, char trans, char diag, int n, int nrhs,
             float *sa, int lda, float *sb, int ldb, float *sx, int
             ldx, float *sferr, float *sberr, int *info)
void ztrrfs (char uplo, char trans, char diag, int n, int nrhs,
             doublecomplex *za, int lda, doublecomplex *zb, int ldb,
             doublecomplex *zx, int ldx, double *dferr, double
             *dberr, int *info)
void ctrrfs (char uplo, char trans, char diag, int n, int nrhs,
             complex *ca, int lda, complex *cb, int ldb, complex
             *cx, int ldx, float *sferr, float *sberr, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal.
'U' or 'u'	xA contains the upper triangle.
'L' or 'l'	xA contains the lower triangle.

TRANSA	Indicates the form of the system of equations. The legal values for TRANSA are listed below. Any values not listed below are illegal. 'N' or 'n' No transpose, use $AX = B$. 'T' or 't' Transpose, use $A^T X = B$. 'C' or 'c' Conjugate transpose, use $A^H X = B$. Note that A^T and A^H are the same for real matrices.
DIAG	Indicates whether or not A is unit triangular. The legal values for DIAG are listed below. Any values not listed below are illegal. 'N' or 'n' A is not unit triangular. 'U' or 'u' A is unit triangular.
N	Order of the matrix A. $N \geq 0$.
NRHS	Number of right-hand sides, equal to the number of columns of the matrices B and X. $NRHS \geq 0$.
xA	Upper or lower triangular matrix A. If $DIAG = 'U'$ or $'u'$, the diagonal elements of A are assumed to be 1 and are not used.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
xB	The $N \times NRHS$ right-hand side matrix B.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $LDB \geq \max(1, N)$.
xX	The $N \times NRHS$ solution matrix X.
LDX	Leading dimension of the array X as specified in a dimension or type statement. $LDX \geq \max(1, N)$.
xFERR	On exit, the estimated forward error bound for each solution vector $X(*, j)$ for $1 \leq j \leq NRHS$. If X' is the true solution corresponding to $X(*, j)$ then $FERR(j)$ is an upper bound on the magnitude of the largest element in $X(*, j) - X'$ divided by the magnitude of the largest element in $X(*, j)$.
xBERR	On exit, $BERR(j)$ is the smallest relative change in any element of A or $B(*, j)$ that makes $X(*, j)$ an exact solution to $AX(*, j) = B(*, j)$ for $1 \leq j \leq NRHS$.
xWORK	Scratch array with a dimension of $3 \times N$ for real subroutines or $2 \times N$ for complex subroutines.
xWORK2	Scratch array with a dimension of N.
INFO	On exit: $INFO = 0$ Subroutine completed normally. $INFO < 0$ The i th argument, where $i = INFO $, had an illegal value.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER          LDA, LDB, LDIWRK, LDWORK, LDX, N, NRHS
  PARAMETER        (N = 4)
  PARAMETER        (LDA = N)
  PARAMETER        (LDB = N)
  PARAMETER        (LDIWRK = N)
  PARAMETER        (LDWORK = 3 * N)
  PARAMETER        (LDX = N)
  PARAMETER        (NRHS = 1)

C
  DOUBLE PRECISION A(LDA,N), B(LDB,NRHS), BERR(NRHS), EPSLON
  DOUBLE PRECISION FERR(NRHS), WORK(LDWORK), X(LDX,NRHS)
  INTEGER          ICOL, INFO, IROW, IWORK(LDIWRK)

C
  EXTERNAL          DCOPY, DTRMV, DTRRFS, DTRTRS
  INTRINSIC         ABS, DBLE

C
C Initialize the array A to store in the matrix A shown below.
C Initialize the array B to store the right hand side vector b
C shown below.
C
C
C      1          4
C      A = 1      0          b = 4
C           0      1          4
C           0      0      0      4
C           0      0      1      0      4
C
C      DATA A / 1.0D0, 1.0D0, 0.0D0, 0.0D0,
C $              8.8D8, 0.0D0, 1.0D0, 0.0D0,
C $              8.8D8, 8.8D8, 0.0D0, 1.0D0,
C $              8.8D8, 8.8D8, 8.8D8, 0.0D0 /
C      DATA B / N*4.0D0 /

C
C Add a small value to the elements of A on and below
C the diagonal. After this loop, A will contain something
C similar to:
C
C      1+e
C      A = 1+e      e
C              e      1+e      e
C              e      e      1+e      e
C
C      EPSLON = (((2.0D0 / 3.0D0) + 1.6D1) - 1.6D1) -
C $              (2.0D0 / 3.0D0))
C      DO 110, ICOL = 1, N
C          DO 100, IROW = ICOL, N
C              A(IROW,ICOL) = A(IROW,ICOL) + EPSLON
C 100      CONTINUE
C 110      CONTINUE

C
C Print the A array.

```

```

C
    PRINT 1000
    DO 120, IROW = 1, N
        PRINT 1010, (A(IROW,ICOL), ICOL = 1, IROW),
$           (0.0D0, ICOL = IROW + 1, N)
120 CONTINUE
    PRINT 1020
    PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, LDA)
C
C    Make a copy of B and then print B.
C
    CALL DCOPY (N, B, 1, X, 1)
    PRINT 1030
    PRINT 1040, B
C
C    Solve Ax=b and print the solution.
C
    CALL DTRTRS ('LOWER TRIANGULAR A', 'NO TRANSPOSE A',
$              'NO UNIT DIAGONAL A', N, NRHS, A, LDA, X, LDX,
$              INFO)
    IF (INFO .NE. 0) THEN
        PRINT 1050, INFO
        STOP 1
    END IF
    PRINT 1060
    PRINT 1070, X
C
C    Estimate the error bounds for the computed solution.
C
    CALL DTRRFS ('LOWER TRIANGULAR A', 'NO TRANSPOSE A',
$              'NO UNIT DIAGONAL A', N, NRHS, A, LDA, B, LDB,
$              X, LDX, FERR, BERR, WORK, IWORK, INFO)
    IF (INFO .NE. 0) THEN
        PRINT 1080, ABS(INFO)
        STOP 1
    END IF
    CALL DTRMV ('LOWER TRIANGULAR A', 'NO TRANSPOSE A',
$              'NO UNIT DIAGONAL A', N, A, LDA, X, 1)
    PRINT 1090
    PRINT 1040, X
    PRINT 1100, (IROW, BERR(IROW), IROW = 1, NRHS)
    PRINT 1110, (IROW, FERR(IROW), IROW = 1, NRHS)
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(1X, F17.15))
1020 FORMAT (/1X, 'A in triangular form: ',
$           ' (* in unused elements)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (1X, F21.18)
1050 FORMAT (1X, 'Error solving Ax=b, INFO = ', I5)
1060 FORMAT (/1X, 'x:')
1070 FORMAT (1X, E16.8)
1080 FORMAT (1X, 'Illegal argument to DTRRFS, argument #', I5)
1090 FORMAT (/1X, 'Ax:')
1110 FORMAT (/1X, 'Backward error bound for system #', I2, ' =',
$           E15.8)

```

```
1100 FORMAT (1X, 'Forward error bound for system #', I2, ' = ',
$          E15.8)
C
      END
```

Sample Output

```
A in full form:
1.0000000000000001 0.0000000000000000 0.0000000000000000 0.0000000000000000
1.0000000000000001 0.0000000000000001 0.0000000000000000 0.0000000000000000
0.0000000000000001 1.0000000000000001 0.0000000000000001 0.0000000000000000
0.0000000000000001 0.0000000000000001 1.0000000000000001 0.0000000000000001

A in triangular form: (* in unused elements)
1.0000000000000001 *****
1.0000000000000001 0.0000000000000001 *****
0.0000000000000001 1.0000000000000001 0.0000000000000001 *****
0.0000000000000001 0.0000000000000001 1.0000000000000001 0.0000000000000001

b:
4.0000000000000000
4.0000000000000000
4.0000000000000000
4.0000000000000000

x:
0.40000000E+01
0.00000000E+00
0.32753452E+16
-0.26819715E+31

Ax:
4.0000000000000000
4.0000000000000000
4.0000000000000000
4.00000000000000004441
Forward error bound for system # 1 = 0.67792734E-30
Backward error bound for system # 1 = 0.18181818E+01
```


Inverse of a Triangular Matrix

The subroutines described in this section compute the inverse of a triangular matrix A.

Calling Sequence

```
CALL DTRTRI (UPLO, DIAG, N, DA, LDA, INFO)
CALL STRTRI (UPLO, DIAG, N, SA, LDA, INFO)
CALL ZTRTRI (UPLO, DIAG, N, ZA, LDA, INFO)
CALL CTRTRI (UPLO, DIAG, N, CA, LDA, INFO)
```

```
void dtrtri (char uplo, char diag, int n, double *da, int lda, int
             *info)
void strtri (char uplo, char diag, int n, float *sa, int lda, int
             *info)
void ztrtri (char uplo, char diag, int n, doublecomplex *za, int
             lda, int *info)
void ctrtri (char uplo, char diag, int n, complex *ca, int lda, int
             *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. 'U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
DIAG	Indicates whether or not A is unit triangular. The legal values for DIAG are listed below. Any values not listed below are illegal. 'N' or 'n' A is not unit triangular. 'U' or 'u' A is unit triangular.
N	Order of the matrix A. $N \geq 0$.
xA	On entry, the upper or lower triangular matrix A. If DIAG = 'U' or 'u', the diagonal elements of A are assumed to be 1 and are not used. On exit, the triangular inverse of the matrix A.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO < 0 The <i>i</i> th argument, where $i = INFO $, had an illegal value.

INFO > 0 A(i,i), where i = INFO, is exactly zero. The matrix is therefore singular and its inverse could not be computed.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C   INTEGER          LDA, N
C   PARAMETER        (N = 4)
C   PARAMETER        (LDA = N)
C
C   DOUBLE PRECISION A(LDA,N)
C   INTEGER          ICOL, INFO, IROW
C
C   EXTERNAL          DTRTRI
C
C   Initialize the array A to store the matrix A shown below.
C
C   A =
C   1
C   1  1
C   1  1  1
C   1  1  1  1
C
C   DATA A / 1.0D0, 1.0D0, 1.0D0, 1.0D0,
$             8.8D8, 1.0D0, 1.0D0, 1.0D0,
$             8.8D8, 8.8D8, 1.0D0, 1.0D0,
$             8.8D8, 8.8D8, 8.8D8, 1.0D0 /
C
C   Print the A array.
C
C   PRINT 1000
C   DO 100, IROW = 1, N
C       PRINT 1010, (A(IROW,ICOL), ICOL = 1, IROW),
$               (0.0D0, ICOL = IROW + 1, N)
100 CONTINUE
C   PRINT 1020
C   PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, LDA)
C
C   Compute and print A**(-1).
C
C   CALL DTRTRI ('LOWER TRIANGULAR A', 'NO UNIT DIAGONAL A',
$             N, A, LDA, INFO)
C   IF (INFO .NE. 0) THEN
C       PRINT 1030, INFO
C       STOP 1
C   END IF
C   PRINT 1040
C   DO 110, IROW = 1, N
C       PRINT 1010, (A(IROW,ICOL), ICOL = 1, IROW),
$               (0.0D0, ICOL = IROW + 1, N)

```

```
110 CONTINUE
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F8.4))
1020 FORMAT (/1X, 'A in triangular form: (* in unused elements)')
1030 FORMAT (/1X, 'Error inverting A, INFO = ', I5)
1040 FORMAT (/1X, 'A**(-1):')
C
      END
```

Sample Output

```
A in full form:
  1.0000    0.0000    0.0000    0.0000
  1.0000    1.0000    0.0000    0.0000
  1.0000    1.0000    1.0000    0.0000
  1.0000    1.0000    1.0000    1.0000

A in triangular form: (* in unused elements)
  1.0000 *****
  1.0000    1.0000 *****
  1.0000    1.0000    1.0000 *****
  1.0000    1.0000    1.0000    1.0000

A**(-1):
  1.0000    0.0000    0.0000    0.0000
 -1.0000    1.0000    0.0000    0.0000
  0.0000   -1.0000    1.0000    0.0000
  0.0000    0.0000   -1.0000    1.0000
```

Solution to a Linear System in a Triangular Matrix

The subroutines described in this section solve a linear system $AX = B$, $A^T X = B$, or $A^H X = B$ for a triangular matrix A and general matrices B and X .

Calling Sequence

```
CALL DTRTRS (UPLO, TRANSA, DIAG, N, NRHS, DA, LDA, DB, LDB, INFO)
CALL STRTRS (UPLO, TRANSA, DIAG, N, NRHS, SA, LDA, SB, LDB, INFO)
CALL ZTRTRS (UPLO, TRANSA, DIAG, N, NRHS, ZA, LDA, ZB, LDB, INFO)
CALL CTRTRS (UPLO, TRANSA, DIAG, N, NRHS, CA, LDA, CB, LDB, INFO)
```

```
void dtrtrs (char uplo, char trans, char diag, int n, int nrhs,
double *da, int lda, double *db, int ldb, int *info)
void strtrs (char uplo, char trans, char diag, int n, int nrhs,
float *sa, int lda, float *sb, int ldb, int *info)
void ztrtrs (char uplo, char trans, char diag, int n, int nrhs,
doublecomplex *za, int lda, doublecomplex *zb, int ldb,
int *info)
void ctrtrs (char uplo, char trans, char diag, int n, int nrhs,
complex *ca, int lda, complex *cb, int ldb, int *info)
```

Arguments

UPLO	Indicates whether xA contains the upper or lower triangle of the matrix. The legal values for UPLO are listed below. Any values not listed below are illegal. U' or 'u' xA contains the upper triangle. 'L' or 'l' xA contains the lower triangle.
TRANSA	Indicates the form of the system of equations. The legal values for TRANSA are listed below. Any values not listed below are illegal. N' or 'n' No transpose, solve $AX = B$. 'T' or 't' Transpose, solve $A^T X = B$. 'C' or 'c' Conjugate transpose, solve $A^H X = B$. Note that A^T and A^H are the same for real matrices.
DIAG	Indicates whether or not A is unit triangular. The legal values for DIAG are listed below. Any values not listed below are illegal. 'N' or 'n' A is not unit triangular. 'U' or 'u' A is unit triangular.
N	Order of the matrix A. $N \geq 0$.

NRHS	Number of right-hand sides, equal to the number of columns of the matrix B. $\text{NRHS} \geq 0$.
xA	Upper or lower triangular matrix A. If $\text{DIAG} = 'U'$ or $'u'$, the diagonal elements of A are assumed to be 1 and are not used.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $\text{LDA} \geq \max(1, N)$.
xB	On entry, the $N \times \text{NRHS}$ right-hand side matrix B. On exit, the $N \times \text{NRHS}$ solution matrix X.
LDB	Leading dimension of the array B as specified in a dimension or type statement. $\text{LDB} \geq \max(1, N)$.
INFO	On exit: INFO = 0 INFO < 0 INFO > 0 Subroutine completed normally. The ith argument, where $i = \text{INFO}$, had an illegal value. The ith diagonal element, where $i = \text{INFO}$, of A is zero. The matrix is therefore singular and the solution could not be computed.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C      DOUBLE PRECISION  ZERO
C      INTEGER           LDA, LDB, N, NRHS
C      PARAMETER         (N = 4)
C      PARAMETER         (LDA = N)
C      PARAMETER         (LDB = N)
C      PARAMETER         (NRHS = 1)
C      PARAMETER         (ZERO = 0.0D0)
C
C      DOUBLE PRECISION  A(LDA,N), B(LDB,NRHS)
C      INTEGER           ICOL, INFO, IROW
C
C      EXTERNAL          DTRTRS
C      INTRINSIC         ABS
C
C      Initialize the array A to store the coefficient matrix A
C      shown below. Initialize the array B to store the right
C      hand side vector b shown below.
C
C      A =
C      1   1   1   1
C      1   1   1   1
C      1   1   1   1
C      1   1   1   1
C
C      b =
C      4
C      3
C      2
C      1

```

```

DATA A / 1.0D0, 1.0D0, 1.0D0, 1.0D0,
$      8.8D8, 1.0D0, 1.0D0, 1.0D0,
$      8.8D8, 8.8D8, 1.0D0, 1.0D0,
$      8.8D8, 8.8D8, 8.8D8, 1.0D0 /
DATA B / 4.0D0, 3.0D0, 2.0D0, 1.0D0 /
C
C      Print the initial value of the arrays.
C
      PRINT 1000
      DO 100, IROW = 1, N
        PRINT 1010, (A(IROW,ICOL), ICOL = 1, IROW),
$              (ZERO, ICOL = IROW + 1, N)
100  CONTINUE
      PRINT 1020
      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, LDA)
      PRINT 1030
      PRINT 1060, B
C
C      Solve Ax=b and print the result.
C
      CALL DTRTRS ('LOWER TRIANGULAR A', 'TRANSPPOSE A',
$              'NO UNIT DIAGONAL A', N, NRHS, A, LDA, B, LDB,
$              INFO)
      IF (INFO .NE. 0) THEN
        PRINT 1040, INFO
        STOP 1
      END IF
      PRINT 1050
      PRINT 1060, B
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F8.4))
1020 FORMAT (/1X, 'A in triangular form: ',
$          ' (* in unused elements)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (/1X, 'Error solving Ax=b, INFO = ', I5)
1050 FORMAT (/1X, 'x:')
1060 FORMAT (1X, F8.4)
C
      END

```

Sample Output

```

A in full form:
  1.0000    0.0000    0.0000    0.0000
  1.0000    1.0000    0.0000    0.0000
  1.0000    1.0000    1.0000    0.0000
  1.0000    1.0000    1.0000    1.0000

A in triangular form: (* in unused elements)
  1.0000 *****
  1.0000    1.0000 *****

```

```
      1.0000      1.0000      1.0000      *****  
      1.0000      1.0000      1.0000      1.0000  
  
      b:  
      4.0000  
      3.0000  
      2.0000  
      1.0000  
  
      x:  
      1.0000  
      1.0000  
      1.0000  
      1.0000
```


This chapter describes the subroutines in LINPACK as they are implemented in the Sun Performance Library. LINPACK is a collection of linear algebra subroutines to perform matrix factorization, solving linear systems of various forms, solutions to least-squares problems, and other common linear algebra operations. For more information about LINPACK, refer to the information in this chapter or see the *LINPACK User's Guide* written by Dongarra *et al* and available from the Society for Industrial and Applied Mathematics (SIAM) in Philadelphia.

The first four sections of this chapter describe:

- conventions followed in naming arguments and subroutines
- conventions used for passing arguments into and out of subroutines
- error handling done by the subroutines
- extensions to the standard version of LINPACK.

The remaining sections of this chapter describe the operation of each of the LINPACK subroutines. For each subroutine, the interface is shown, the operation of the subroutine is described, and a short FORTRAN 77 program is given with sample output.

Naming Conventions

The types of all arguments follow the notation shown below:

```

IMPLICIT COMPLEX (C)
IMPLICIT DOUBLE PRECISION (D)
IMPLICIT INTEGER (I-N)
IMPLICIT DOUBLE COMPLEX (Z)

```

The first letter of a subroutine name identifies the type of the result. The second and third letters identify the form of the output matrix and its mode of storage in an array. The forms are shown below:

CH	Cholesky-factored
GB	general in banded storage
GE	general
GT	general tridiagonal
HI	Hermitian
HP	Hermitian in packed storage
PB	symmetric positive definite in banded storage
PO	symmetric positive definite
PP	symmetric positive definite in packed storage
PT	symmetric positive definite tridiagonal
QR	QR-factored
SI	symmetric
SP	symmetric in packed storage
TR	triangular

When the second and third letters indicate the type of matrix and its storage mode, the fourth and fifth letters indicate the operation that will be performed. The fourth and fifth letters mean different things for different subroutines, so you must read the description of the subroutine in order to find out exactly what it does. For example, DTRCO estimates the condition of a triangular matrix. DGECCO estimates the condition of a general matrix and also determines the LU factorization. The operations are shown below:

CO	Estimate the condition number; sometimes also determine the LU or Cholesky factorization
DC	Decomposition
DD	Downdate a Cholesky factorization
DI	Compute the determinant of a matrix; sometimes also determine the inverse and inertia of a matrix
EX	Permute and update a Cholesky factorization
FA	Factor

SL	Solve a system of linear equations, usually after the matrix has been factored by a subroutine whose name ends in CO or FA
UD	Update a Cholesky factorization

A^{-1} denotes the inverse of the square matrix stored in the array A. Whenever the symbol A^{-1} appears it is assumed that the numerical inverse of the matrix stored in the array A exists. A^T denotes the transpose of an array A and A^H denotes the conjugate transpose of A for a complex matrix A.

Argument Conventions

Although the FORTRAN 77 standard prohibits aliasing arguments, the Sun Performance Library allows the programmer to alias two or more arguments as long as all aliased arguments have the same base type and one of the arguments being aliased is not an argument into which the subroutine will write a result. A real variable and a complex variable have the same base type (real), while a real variable and a double precision variable have different base types. Aliasing arguments having different base types or aliasing any argument with a result argument will produce undefined results at best unless the aliased arguments are not used. For example, the argument DET in the subroutine DGEDI will not be used if the argument JOB has the value 1, so DET could be aliased to any other argument of any other type. If you violate the rules about aliasing then the results are undefined, and it is not even guaranteed that these undefined results will be the same from one run to the next.

Error Handling

Some of the LINPACK subroutines use the argument INFO to return information about whether or not the requested operation completed successfully. The subroutines do not use INFO consistently, so you must read the description of a particular subroutine to determine how it interprets INFO. Many of the subroutines do not have a status argument and simply generate an arithmetic error such as divide by zero if one or more of the operands are not set up satisfactorily. Most of the operating systems and run-time systems on machines in use at the time that LINPACK was developed would abort a program that got an arithmetic exception. IEEE arithmetic in common use today simply records arithmetic exceptions in a status register and continues running. This may mean that errors, especially in programs ported from older operating environments, will go undetected by the program.

Extensions to LINPACK

There are two extensions to the standard described in the *LINPACK User's Guide*. Sun Performance Library contains double complex (COMPLEX*16) although double complex is not a standard FORTRAN 77 data type. This extension is in accordance with the recommendations stated by the LINPACK developers in the *LINPACK User's Guide*. Also, this version of Sun Performance Library contains a native C/C++ interface to LINPACK. Conventions for this interface are described in detail in the preface to this manual.

Cholesky Decomposition of a Symmetric Positive Definite Matrix

The subroutines described in this section compute the Cholesky decomposition of a symmetric positive definite matrix A.

Calling Sequence

```
CALL DCHDC (DA, LDA, N, DWORK, IPIVOT, JOB, INFO)
CALL SCHDC (SA, LDA, N, SWORK, IPIVOT, JOB, INFO)
CALL ZCHDC (ZA, LDA, N, ZWORK, IPIVOT, JOB, INFO)
CALL CCHDC (CA, LDA, N, CWORK, IPIVOT, JOB, INFO)

void dchdc (double *da, int lda, int n, int *ipivot, int job, int
           *info)
void schdc (float *sa, int lda, int n, int *ipivot, int job, int
           *info)
void zchdc (doublecomplex *za, int lda, int n, int *ipivot, int
           job, int *info)
void cchdc (complex *ca, int lda, int n, int *ipivot, int job, int
           *info)
```

Arguments

xA	On entry, the upper triangle of A. On exit, the upper triangle of the Cholesky factor, as permuted by pivoting if pivoting was selected. The strict lower triangle of A is not referenced.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
N	Order of the matrix A. $N \geq 0$.
xWORK	Scratch array with a dimension of N.
IPIVOT	If JOB selected no pivoting then IPIVOT is not referenced. On entry, if JOB selected pivoting, IPIVOT contains integers representing array indices that control the selection of pivot elements from the diagonal of A according to the system below: IPIVOT(k) > 0 A(k,k) is an initial element. IPIVOT(k) = 0 A(k,k) is a free element. IPIVOT(k) < 0 A(k,k) is a final element.

Before the decomposition is computed, symmetric row and column interchanges are used to move initial elements to the beginning of A and final elements to the end of A. During the computation, symmetric row and column interchanges are used to move the largest remaining free diagonal element into the pivot position.

On exit, IPIVOT(k) contains the index of the diagonal element of A that was moved into the kth position.

JOB Determines how decomposition is done:

0 without pivoting
not 0 with pivoting

INFO On exit:

Returns the index of the last positive diagonal element of the Cholesky factor.

The subroutine was successful if INFO = N.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C   INTEGER          LDA, N, NOPIV
C   PARAMETER        (N = 4)
C   PARAMETER        (LDA = N)
C   PARAMETER        (NOPIV = 0)
C
C   DOUBLE PRECISION A(LDA,N), WORK(N)
C   INTEGER          I, INFO, IPIVOT(N), J, JOB
C
C   EXTERNAL          DCHDC
C
C   4  3  2  1
C   A = 3  4  3  2
C   2  3  4  3
C   1  2  3  4
C
C   DATA A / 4.0D0, 3*8D8, 3.0D0, 4.0D0, 2*8D8, 2.0D0, 3.0D0,
$           4.0D0, 8D8, 1.0D0, 2.0D0, 3.0D0, 4.0D0 /
C
C   PRINT 1000
C   DO 100, I = 1, N
C       PRINT 1010, (A(J,I), J = 1, I), (A(I,J), J = I + 1, N)
100 CONTINUE
C   PRINT 1020
C   PRINT 1010, ((A(I,J), J = 1, N), I = 1, N)
C   JOB = NOPIV
C   CALL DCHDC (A, LDA, N, WORK, IPIVOT, JOB, INFO)
C   IF (INFO .EQ. N) THEN
C       PRINT 1030

```

```

        PRINT 1010, A(1,1), A(1,2), A(1,3), A(1,4)
        PRINT 1040,          A(2,2), A(2,3), A(2,4)
        PRINT 1050,          A(3,3), A(3,4)
        PRINT 1060,          A(4,4)
      ELSE
        PRINT 1070
      END IF
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F7.3))
1020 FORMAT (/1X, 'A in symmetric form: (* in unused entries)')
1030 FORMAT (/1X, 'Upper Cholesky factor:')
1040 FORMAT (10X, 3(3X, F7.3))
1050 FORMAT (20X, 2(3X, F7.3))
1060 FORMAT (30X, 1(3X, F7.3))
1070 FORMAT (/1X, 'A is not positive definite.')
C
      END

```

Sample Output

```

A in full form:
  4.000    3.000    2.000    1.000
  3.000    4.000    3.000    2.000
  2.000    3.000    4.000    3.000
  1.000    2.000    3.000    4.000

A in symmetric form: (* in unused entries)
  4.000    3.000    2.000    1.000
  *****    4.000    3.000    2.000
  *****    *****    4.000    3.000
  *****    *****    *****    4.000

Upper Cholesky factor:
  2.000    1.500    1.000    0.500
          1.323    1.134    0.945
                  1.309    1.091
                        1.291

```

Downdate an Augmented Cholesky Decomposition

The subroutines described in this section downdate an augmented Cholesky decomposition of the triangular part of an augmented QR decomposition.

Calling Sequence

CALL DCHDD	(DA, LDA, N, DX, DZ, LDZ, NZ, DY, DRHO, DCOS, DSIN, INFO)
CALL SCHDD	(SA, LDA, N, SX, SZ, LDZ, NZ, SY, SRHO, SCOS, SSIN, INFO)
CALL ZCHDD	(ZA, LDA, N, ZX, ZZ, LDZ, NZ, ZY, DRHO, DCOS, DSIN, INFO)
CALL CCHDD	(CA, LDA, N, CX, CZ, LDZ, NZ, CY, SRHO, SCOS, SSIN, INFO)
void dchdd	(double *da, int lda, int n, double *dx, double *dz, int ldz, int nz, double *dy, double *drho, double *dcos, double *dsin, int *info)
void schdd	(float *sa, int lda, int n, float *sx, float *sz, int ldz, int nz, float *sy, float *srho, float *scos, float *ssin, int *info)
void zchdd	(doublecomplex *za, int lda, int n, doublecomplex *zx, doublecomplex *zz, int ldz, int nz, doublecomplex *zy, double *drho, double *dcos, double *dsin, int *info)
void cchdd	(complex *ca, int lda, int n, complex *cx, complex *cz, int ldz, int nz, complex *cy, double *drho, double *dcos, double *dsin, int *info)

Arguments

xA	On entry, the upper triangular matrix A. On exit, A has been downdated. The strict lower triangle of A is not referenced.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
N	Order of the matrix A. $N \geq 0$.
xX	Row to be added to A.
xZ	Vectors to be downdated with A.
LDZ	Leading dimension on the array Z as specified in a dimension or type statement. $LDZ \geq \max(1, N)$.

NZ	Number of vectors to be downdated with A. $NZ \geq 0$. If $NZ = 0$ then Z, Y, and RHO are not used.
xY	Scalars for downdating the vectors in Z.
xRHO	On entry, the norms of the residual vectors that are to be downdated. On exit, RHO has been downdated. If $RHO(i)$ is negative on entry then it is not changed.
xCOS	Cosines of the transforming rotations.
xSIN	Sines of the transforming rotations.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO = -1 A could not be downdated; all values are left unchanged. INFO = 1 Some RHOs could not be downdated; all RHOs that could not be downdated are changed to -1.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C   INTEGER          LDA, N, NOPIV, NZ
C   PARAMETER        (N = 4)
C   PARAMETER        (LDA = N)
C   PARAMETER        (NOPIV = 0)
C   PARAMETER        (NZ = 0)
C
C   DOUBLE PRECISION A(LDA,N), ANULL, C(N), S(N), WORK(N), X(N)
C   INTEGER          I, INFO, IPIVOT(N), J, JOB, NULL
C
C   EXTERNAL          DCHDC, DCHDD
C
C   Initialize the arrays A and Z to store the matrices A and Z
C   shown below and initialize X and Y to store the vectors x
C   and y shown below.
C
C   4  3  2  1      1
C   A = 3  4  3  2    x = 1
C   2  3  4  3      1
C   1  2  3  4      1
C
C   DATA A / 4.0D0, 3*8D8, 3.0D0, 4.0D0, 2*8D8, 2.0D0, 3.0D0,
C   $         4.0D0, 8D8, 1.0D0, 2.0D0, 3.0D0, 4.0D0 /
C
C   PRINT 1000
C   DO 100, I = 1, N
C     PRINT 1010, (A(J,I), J = 1, I), (A(I,J), J = I + 1, N)
100 CONTINUE
PRINT 1020

```

```

PRINT 1010, ((A(I,J), J = 1, N), I = 1, N)
JOB = NOPIV
CALL DCHDC (A, LDA, N, WORK, IPIVOT, JOB, INFO)
IF (INFO .EQ. N) THEN
  PRINT 1030
  PRINT 1010, A(1,1), A(1,2), A(1,3), A(1,4)
  PRINT 1040,           A(2,2), A(2,3), A(2,4)
  PRINT 1050,           A(3,3), A(3,4)
  PRINT 1060,           A(4,4)
  ANULL = 0.0D0
  NULL = 1
  CALL DCHDD (A, LDA, N, X, ANULL, NULL, NZ, ANULL, ANULL,
$           C, S, INFO)
  IF (INFO .EQ. 0) THEN
    PRINT 1070
    PRINT 1080, (C(I), S(I), I = 1, N)
  ELSE
    PRINT 1090
  END IF
ELSE
  PRINT 1100
END IF
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F7.3))
1020 FORMAT (/1X, 'A in symmetric form (* in unused entries)')
1030 FORMAT (/1X, 'Upper Cholesky factor:')
1040 FORMAT (10X, 3(3X, F7.3))
1050 FORMAT (20X, 2(3X, F7.3))
1060 FORMAT (30X, 1(3X, F7.3))
1070 FORMAT (/1X, 'Cosine', 3X, ' Sine')
1080 FORMAT (1X, F6.3, 3X, F6.3)
1090 FORMAT (/1X, 'A cannot be downdated.')
1100 FORMAT (/1X, 'A is not positive definite.')
C
END

```

Sample Output

```

A in full form:
  4.000    3.000    2.000    1.000
  3.000    4.000    3.000    2.000
  2.000    3.000    4.000    3.000
  1.000    2.000    3.000    4.000

A in symmetric form (* in unused entries)
  4.000    3.000    2.000    1.000
  *****    4.000    3.000    2.000
  *****    *****    4.000    3.000
  *****    *****    *****    4.000

Upper Cholesky factor:

```

	2.000	1.500	1.000	0.500
		1.323	1.134	0.945
			1.309	1.091
				1.291
Cosine		Sine		
1.000		0.000		
1.000		0.000		
1.000		0.000		
1.000		0.000		

Update an Augmented Cholesky Decomposition with Permutations

The subroutines described in this section compute the Cholesky decomposition of a symmetric positive definite matrix A.

Calling Sequence

```
CALL DCHEX (DA, LDA, N, K, L, DZ, LDZ, NZ, DCOS, DSIN, JOB)
CALL SCHEX (SA, LDA, N, K, L, SZ, LDZ, NZ, SCOS, SSIN, JOB)
CALL CCHEX (ZA, LDA, N, K, L, ZZ, LDZ, NZ, DCOS, ZSIN, JOB)
CALL ZCHEX (CA, LDA, N, K, L, CZ, LDZ, NZ, SCOS, CSIN, JOB)

void dchex (double *da, int lda, int n, int k, int l, double *dz,
            int ldz, int nz, double *dcos, double *dsin, int job)
void schex (float *da, int lda, int n, int k, int l, float *sz,
            int ldz, int nz, float *scos, float *ssin, int job)
void zchex (doublecomplex *za, int lda, int n, int k, int l,
            doublecomplex *zz, int ldz, int nz, double *dcos,
            doublecomplex *zsine, int job)
void cchex (complex *r, int lda, int n, int k, int l, complex *cz,
            int ldz, int nz, float *scos, complex *csin, int job)
```

Arguments

xA	On entry, the upper triangle of the matrix A. On exit, the upper triangle of A contains the upper triangle of the updated factor. The strict lower triangle of A is not referenced.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
N	Order of the matrix A. $N \geq 0$.
K	First column to be permuted. $1 \leq K < L$.
L	Last column to be permuted; must be strictly greater than K. $K < L \leq N$.
xZ	Array of vectors into which the transformation U is multiplied. Not used if $NZ = 0$.
LDZ	Leading dimension of the array Z as specified in a dimension or type statement. $LDZ \geq \max(1, N)$.
NZ	Number of columns in the matrix Z. $NZ \geq 0$.
xCOS	Cosines of the transforming rotations.
xDIN	Sines of the transforming rotations.
JOB	Determines the type of permutation:

- 1 right circular shift
- 2 left circular shift

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
  INTEGER          ISHFRT, LDA, N, NULL, NZ
  PARAMETER        (ISHFRT = 1)
  PARAMETER        (N = 4)
  PARAMETER        (LDA = N)
  PARAMETER        (NULL = 1)
  PARAMETER        (NZ = 0)
C
  DOUBLE PRECISION A(LDA,N), ANULL, C(N), S(N), WORK(N)
  INTEGER          ICOL, INFO, IPIVOT(N), IROW, JOB, K
C
  EXTERNAL          DCHDC, DCHEX
C
  Initialize the arrays A and Z to store the matrices A and Z
  shown below and initialize X and Y to store the vectors x
  and y shown below.
C
C      4  3  2  1      1
C      A = 3  4  3  2      x = 1
C           2  3  4  3      1
C           1  2  3  4      1
C
  DATA A / 4.0D0, 3*8D8, 3.0D0, 4.0D0, 2*8D8, 2.0D0, 3.0D0,
$          4.0D0, 8D8, 1.0D0, 2.0D0, 3.0D0, 4.0D0 /
C
  PRINT 1000
  DO 100, IROW = 1, N
    PRINT 1010, (A(ICOL,IROW), ICOL = 1, IROW),
$              (A(IROW,ICOL), ICOL = IROW + 1, N)
  100 CONTINUE
  PRINT 1020
  PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
  CALL DCHDC (A, LDA, N, WORK, IPIVOT, JOB, INFO)
  IF (INFO .EQ. N) THEN
    PRINT 1030
    PRINT 1010, A(1,1), A(1,2), A(1,3), A(1,4)
    PRINT 1040,          A(2,2), A(2,3), A(2,4)
    PRINT 1050,          A(3,3), A(3,4)
    PRINT 1060,          A(4,4)
    K = 1
    ANULL = 0.0D0
    JOB = ISHFRT
    CALL DCHEX (A, LDA, N, K, N, ANULL, NULL, NZ, C, S, JOB)
    PRINT 1070
    PRINT 1010, A(1,1), A(1,2), A(1,3), A(1,4)

```

```

        PRINT 1040,          A(2,2), A(2,3), A(2,4)
        PRINT 1050,          A(3,3), A(3,4)
        PRINT 1060,          A(4,4)
        PRINT 1080
        PRINT 1090, (C(IROW), S(IROW), IROW = 1, N)
    ELSE
        PRINT 1100
    END IF
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F7.3))
1020 FORMAT (/1X, 'A in symmetric form (* in unused entries)')
1030 FORMAT (/1X, 'Upper Cholesky factor:')
1040 FORMAT (10X, 3(3X, F7.3))
1050 FORMAT (20X, 2(3X, F7.3))
1060 FORMAT (30X, 1(3X, F7.3))
1070 FORMAT (1X, 'Updated Cholesky factor:')
1080 FORMAT (/1X, 'Cosine', 3X, ' Sine')
1090 FORMAT (1X, F6.3, 3X, F6.3)
1100 FORMAT (/1X, 'A is not positive definite.')
C
    END

```

Sample Output

```

A in full form:
  4.000    3.000    2.000    1.000
  3.000    4.000    3.000    2.000
  2.000    3.000    4.000    3.000
  1.000    2.000    3.000    4.000

A in symmetric form (* in unused entries)
  4.000    3.000    2.000    1.000
  ***** 4.000    3.000    2.000
  ***** ***** 4.000    3.000
  ***** ***** ***** 4.000

Upper Cholesky factor:
  2.000    1.500    1.000    0.500
           1.323    1.134    0.945
                1.309    1.091
                        1.291

Updated Cholesky factor:
  2.000    0.500    1.000    1.500
           -1.936   -1.291   -0.645
                -1.155   -0.577
                        -1.000

Cosine    Sine
  0.645    0.764
  0.488    0.873

```

0.250	0.968
0.000	2.000

Update an Augmented Cholesky Decomposition

The subroutines described in this section update an augmented Cholesky decomposition of the triangular part of an augmented QR decomposition.

Calling Sequence

```
CALL DCHUD (DA, LDA, N, DX, DZ, LDZ, NZ, DY, DRHO, DCOS, DSIN)
CALL SCHUD (SA, LDA, N, SX, SZ, LDZ, NZ, SY, SRHO, SCOS, SSIN)
CALL ZCHUD (ZA, LDA, N, ZX, ZZ, LDZ, NZ, ZY, DRHO, DCOS, ZSIN)
CALL CCHUD (CA, LDA, N, CX, CZ, LDZ, NZ, CY, SRHO, SCOS, CSIN)

void dchud (double *da, int lda, int n, double *dx, double *dz,
            int ldz, int nz, double *dy, double *drho, double
            *dcos, double *dsin)
void schud (float *sa, int lda, int n, float *sx, float *sz, int
            ldz, int nz, float *sy, float *srho, float *scos, float
            *ssin)
void zchud (doublecomplex *za, int lda, int n, doublecomplex *zx,
            doublecomplex *zz, int ldz, int nz, doublecomplex *zy,
            double *drho, double *dcos, doublecomplex *zsin)
void cchud (complex *ca, int lda, int n, complex *cx, complex *cz,
            int ldz, int nz, complex *cy, float *srho, float *scos,
            complex *csin)
```

Arguments

xA	On entry, the upper triangular matrix A. On exit, A has been updated. The strict lower triangle of A is not referenced.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
N	Order of the matrix A. $N \geq 0$.
xX	Row to be added to A.
xZ	Vectors to be updated with A.
LDZ	Leading dimension on the array Z as specified in a dimension or type statement. $LDZ \geq \max(1, N)$.
NZ	Number of vectors to be updated with A. $NZ \geq 0$. If $NZ = 0$ then Z, Y, and RHO are not used.
xY	Scalars for updating the vectors in Z.

xRHO	On entry, the norms of the residual vectors that are to be updated. On exit, RHO has been updated. If RHO(i) is negative on entry then it is not changed.
xCOS	Cosines of the transforming rotations.
xSIN	Sines of the transforming rotations.

Sample Program

```

C      PROGRAM TEST
C      IMPLICIT NONE
C
C      INTEGER          LDA, N, NOPIV, NZ
C      PARAMETER        (N = 4)
C      PARAMETER        (NOPIV = 0)
C      PARAMETER        (NZ = 0)
C      PARAMETER        (LDA = N)
C
C      DOUBLE PRECISION A(LDA,N), ANULL, C(N), S(N), WORK(N), X(N)
C      INTEGER          I, ICOL, INFO, IPIVOT(N), IROW, JOB, NULL
C
C      Initialize the arrays A and Z to store the matrices A and Z
C      shown below and initialize X and Y to store the vectors x
C      and y shown below.
C
C      4  3  2  1      1
C      A = 3  4  3  2    x = 1
C      2  3  4  3      1
C      1  2  3  4      1
C
C      DATA A / 4.0D0, 3*8D8, 3.0D0, 4.0D0, 2*8D8, 2.0D0, 3.0D0,
C      $          4.0D0, 8D8, 1.0D0, 2.0D0, 3.0D0, 4.0D0 /
C
C      PRINT 1000
C      PRINT 1010, A(1,1), A(1,2), A(1,3), A(1,4)
C      PRINT 1010, A(1,2), A(2,2), A(2,3), A(2,4)
C      PRINT 1010, A(1,3), A(2,3), A(3,3), A(3,4)
C      PRINT 1010, (A(IROW,4), IROW = 1, N)
C      PRINT 1020
C      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
C      JOB = NOPIV
C      CALL DCHDC (A, LDA, N, WORK, IPIVOT, JOB, INFO)
C      IF (INFO .EQ. N) THEN
C          PRINT 1030
C          PRINT 1010, A(1,1), A(1,2), A(1,3), A(1,4)
C          PRINT 1040,          A(2,2), A(2,3), A(2,4)
C          PRINT 1050,          A(3,3), A(3,4)
C          PRINT 1060,          A(4,4)
C          ANULL = 0.0D0
C          NULL = 1
C          CALL DCHUD (A, LDA, N, X, ANULL, NULL, NZ, ANULL, ANULL,
C      $              C, S)

```

```

        PRINT 1070
        PRINT 1080, (C(I), S(I), I = 1, N)
    ELSE
        PRINT 1090
    END IF
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F7.3))
1020 FORMAT (/1X, 'A in symmetric form (* in unused entries)')
1030 FORMAT (/1X, 'Upper Cholesky factor:')
1040 FORMAT (10X, 3(3X, F7.3))
1050 FORMAT (20X, 2(3X, F7.3))
1060 FORMAT (30X, 1(3X, F7.3))
1070 FORMAT (1X, 'Cosine', 3X, ' Sine')
1080 FORMAT (1X, F6.3, 3X, F6.3)
1090 FORMAT (/1X, 'A is not positive definite.')
C
    END

```

Sample Output

```

A in full form:
  4.000    3.000    2.000    1.000
  3.000    4.000    3.000    2.000
  2.000    3.000    4.000    3.000
  1.000    2.000    3.000    4.000

A in symmetric form (* in unused entries)
  4.000    3.000    2.000    1.000
  *****    4.000    3.000    2.000
  *****    *****    4.000    3.000
  *****    *****    *****    4.000

Upper Cholesky factor:
  2.000    1.500    1.000    0.500
           1.323    1.134    0.945
                   1.309    1.091
                          1.291

Cosine    Sine
  1.000    0.000
  1.000    0.000
  1.000    0.000
  1.000    0.000

```

LU Factorization and Condition Number of a General Matrix in Banded Storage

The subroutines described in this section compute the LU factorization and condition number of a general matrix A in banded storage. If the condition number is not needed then xGBFA is slightly faster. It is typical to follow a call to xGBCO with a call to xGBSL to solve $Ax = b$ or to xGBDI to compute the determinant of A.

Calling Sequence

```
CALL DGBCO (DA, LDA, N, NSUB, NSUPER, IPIVOT, DRCOND, DWORK)
CALL SGBCO (SA, LDA, N, NSUB, NSUPER, IPIVOT, SRCOND, SWORK)
CALL ZGBCO (ZA, LDA, N, NSUB, NSUPER, IPIVOT, DRCOND, ZWORK)
CALL CGBCO (CA, LDA, N, NSUB, NSUPER, IPIVOT, SRCOND, CWORK)

void dgbco (double *da, int lda, int n, int nsub, int nsuper, int
            *ipivot, double *drcond)
void sgbco (float *sa, int lda, int n, int nsub, int nsuper, int
            *ipivot, float *srcond)
void zgbco (doublecomplex *za, int lda, int n, int nsub, int
            nsuper, int *ipivot, double *drcond)
void cgbco (complex *ca, int lda, int n, int nsub, int nsuper, int
            *ipivot, float *srcond)
```

Arguments

xA	On entry, the matrix A. On exit, an LU factorization of the matrix A.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq 2 \times NSUB + NSUPER + 1$.
N	Order of the matrix A. $N \geq 0$.
NSUB	Number of subdiagonals of A. $N-1 \geq NSUB \geq 0$ but if $N = 0$ then $NSUB = 0$.
NSUPER	Number of superdiagonals of A. $N-1 \geq NSUPER \geq 0$ but if $N = 0$ then $NSUPER = 0$.
IPIVOT	On exit, a vector of pivot indices.

xRCOND On exit, an estimate of the reciprocal condition number of A.
 $0.0 \leq \text{RCOND} \leq 1.0$. As the value of RCOND gets smaller, operations with A such as solving $Ax = b$ may become less stable. If RCOND satisfies $\text{RCOND} + 1.0 = 1.0$ then A may be singular to working precision.

xWORK Scratch array with a dimension of N.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
  INTEGER          IAXEQB, LDA, LDAB, N, NDIAG, NSUB, NSUPER
  PARAMETER        (IAXEQB = 0)
  PARAMETER        (N = 4)
  PARAMETER        (LDA = N)
  PARAMETER        (NSUB = 1)
  PARAMETER        (NSUPER = 1)
  PARAMETER        (NDIAG = NSUB + 1 + NSUPER)
  PARAMETER        (LDAB = 2 * NSUB + 1 + NSUPER)
C
  DOUBLE PRECISION AB(LDAB,N), AG(LDA,N), B(N), RCOND, WORK(N)
  INTEGER          ICOL, IPIVOT(N), IROW, IROWB, I1, I2, JOB
C
  EXTERNAL          DGBCO, DGBSL
  INTRINSIC         MAX0, MIN0
C
  C Initialize the array AG to store the 4x4 matrix A with one
  C subdiagonal and one superdiagonal shown below. Initialize
  C the array B to store the vector b shown below.
  C
  C      2  -1
  C  AG = -1  2  -1      b = 5
  C          -1  2  -1      5
  C          -1  2      5
  C
  DATA AB / 16*8D8 /
  DATA AG / 2.0D0, -1.0D0, 2*0D0, -1.0D0, 2.0D0, -1.0D0,
  $          2*0D0, -1.0D0, 2.0D0, -1.0D0, 2*0D0, -1.0D0,
  $          2.0D0 /
  DATA B / N*5.0D0 /
C
  C Copy the matrix A from the array AG to the array AB. The
  C matrix is stored in general storage mode in AG and it will
  C be stored in banded storage mode in AB. The code to copy
  C from general to banded storage mode is taken from the
  C comment block in the original DGBFA by Cleve Moler.
  C
  DO 10, ICOL = 1, N
    I1 = MAX0 (1, ICOL - NSUPER)
    I2 = MIN0 (N, ICOL + NSUB)

```

```

      DO 10, IROW = I1, I2
        IROWB = IROW - ICOL + NDIAG
        AB(IROWB,ICOL) = AG(IROW,ICOL)
10    CONTINUE
20  CONTINUE
C
C    Print the initial values of the arrays.
C
      PRINT 1000
      PRINT 1010, ((AG(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
      PRINT 1020
      PRINT 1010, ((AB(IROW,ICOL), ICOL = 1, N),
$      IROW = 2 * NSUB, 2 * NSUB + 1 + NSUPER)
      PRINT 1030
      PRINT 1040, B
C
C    Factor the matrix in banded form.
C
      CALL DGBCO (AB, LDA, N, NSUB, NSUPER, IPIVOT, RCOND, WORK)
      PRINT 1050, RCOND
      IF ((RCOND + 1.0D0) .EQ. 1.0D0) THEN
        PRINT 1070
      END IF
      JOB = IAXEQB
      CALL DGBSL (AB, LDA, N, NSUB, NSUPER, IPIVOT, B, JOB)
      PRINT 1060
      PRINT 1040, B
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F4.1))
1020 FORMAT (/1X, 'A in banded form: (* in unused elements)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (3X, F4.1)
1050 FORMAT (/1X, 'Reciprocal of the condition number: ', F5.2)
1060 FORMAT (/1X, 'A**(-1) * b:')
1070 FORMAT (1X, 'A may be singular to working precision.')
C
      END

```

Sample Output

```

A in full form:
  2.0   -1.0    0.0    0.0
 -1.0    2.0   -1.0    0.0
  0.0   -1.0    2.0   -1.0
  0.0    0.0   -1.0    2.0

A in banded form: (* in unused elements)
****   -1.0   -1.0   -1.0
  2.0    2.0    2.0    2.0
 -1.0   -1.0   -1.0   ****

```

```
b:
  5.0
  5.0
  5.0
  5.0

Reciprocal of the condition number:  0.08

A**(-1) * b:
  10.0
  15.0
  15.0
  10.0
```

Determinant of an LU-Factored General Matrix in Banded Storage

The subroutines described in this section compute the determinant of a general matrix A in banded storage, which has been LU-factored by xGBCO or xGBFA.

Calling Sequence

```
CALL DGBDI (DA, LDA, N, NSUB, NSUPER, IPIVOT, DDET)
CALL SGBDI (SA, LDA, N, NSUB, NSUPER, IPIVOT, SDET)
CALL ZGBDI (ZA, LDA, N, NSUB, NSUPER, IPIVOT, ZDET)
CALL CGBDI (CA, LDA, N, NSUB, NSUPER, IPIVOT, CDET)

void dgbdi (double *da, int lda, int n, int nsub, int nsuper, int
            *ipivot, double *ddet)
void sgbdi (float *sa, int lda, int n, int nsub, int nsuper, int
            *ipivot, float *sdet)
void zgbdi (doublecomplex *za, int lda, int n, int nsub, int
            nsuper, int *ipivot, doublecomplex *zdet)
void cgbdi (complex *ca, int lda, int n, int nsub, int nsuper, int
            *ipivot, complex *cdet)
```

Arguments

xA	LU factorization of the matrix A, as computed by xGBCO or xGBFA.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq 2 \times NSUB + NSUPER + 1$.
N	Order of the original matrix A. $N \geq 0$.
NSUB	Number of subdiagonals of A. $N-1 \geq NSUB \geq 0$ but if $N = 0$ then $NSUB = 0$.
NSUPER	Number of superdiagonals of A. $N-1 \geq NSUPER \geq 0$ but if $N = 0$ then $NSUPER = 0$.
IPIVOT	Pivot vector as computed by xGBCO or xGBFA.
xDET	On exit, the determinant of the matrix A. The determinant is stored as $b \times 10^{\text{expon}}$ where b is stored in DET(1) and expon is stored in DET(2). $1.0 \leq DET(1) < 10.0$ or $DET(1) = 0.0$.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
INTEGER          LDA, LDAB, LENGAB, N, NDIAG, NSUB, NSUPER
PARAMETER        (N = 5)
PARAMETER        (NSUB = 4)
PARAMETER        (NSUPER = 0)
PARAMETER        (NDIAG = NSUB + 1 + NSUPER)
PARAMETER        (LDA = N)
PARAMETER        (LDAB = 2 * NSUB + 1 + NDIAG)
PARAMETER        (LENGAB = LDAB * N)

C
DOUBLE PRECISION A(LDA,N), AB(LDAB, N), DET(2)
INTEGER          ICOL, INFO, IPIVOT(N), IROW, IROWB, I1, I2

C
EXTERNAL          DGBDI, DGBFA

C
C Initialize the array A to store the 5x5 matrix A with four
C subdiagonals and no superdiagonals shown below.
C
C
C      1
C      1 -1
C      A = 1 -2  1
C           1 -3  3 -1
C           1 -4  6 -4  1
C
C      DATA A / 1.0D0, 1.0D0, 1.0D0, 1.0D0, 1.0D0,
C $              0.0D0, -1.0D0, -2.0D0, -3.0D0, -4.0D0,
C $              0.0D0, 0.0D0, 1.0D0, 3.0D0, 6.0D0,
C $              0.0D0, 0.0D0, 0.0D0, -1.0D0, -4.0D0,
C $              0.0D0, 0.0D0, 0.0D0, 0.0D0, 1.0D0 /
C      DATA AB / LENGAB*8D8 /

C      DO 20, ICOL = 1, N
C          I1 = MAX0 (1, ICOL - NSUPER)
C          I2 = MIN0 (N, ICOL + NSUB)
C          DO 10, IROW = I1, I2
C              IROWB = IROW - ICOL + NDIAG
C              AB(IROWB, ICOL) = A(IROW, ICOL)
C 10      CONTINUE
C 20 CONTINUE

C
C Print the initial values of the arrays.
C
C
C      PRINT 1000
C      PRINT 1010, ((A(IROW, ICOL), ICOL = 1, N), IROW = 1, N)
C      PRINT 1020
C      PRINT 1010, ((AB(IROW, ICOL), ICOL = 1, N),
C $              IROW = 1, 2 * NSUB + 1 + NSUPER)

C
C Factor the matrix in banded form.
C

```



```

      CALL DGBFA (AB, LDAB, N, NSUB, NSUPER, IPIVOT, INFO)
      IF (INFO .EQ. 0) THEN
        CALL DGBDI (AB, LDAB, N, NSUB, NSUPER, IPIVOT, DET)
        PRINT 1030, DET(1) * (1.0D0 ** DET(2))
      ELSE
        PRINT 1040, INFO
      END IF
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (5(3X, F4.1))
1020 FORMAT (/1X, 'A in banded form: (* in unused elements)')
1030 FORMAT (/1X, 'det(A) = ', F4.1)
1040 FORMAT (/1X, 'A appears singular at ', I2)
C
      END

```

Sample Output

```

A in full form:
  1.0   0.0   0.0   0.0   0.0
  1.0  -1.0   0.0   0.0   0.0
  1.0  -2.0   1.0   0.0   0.0
  1.0  -3.0   3.0  -1.0   0.0
  1.0  -4.0   6.0  -4.0   1.0

A in banded form: (* in unused elements)
****
****
****
****
  1.0  -1.0   1.0  -1.0   1.0
  1.0  -2.0   3.0  -4.0   ****
  1.0  -3.0   6.0   ****   ****
  1.0  -4.0   ****   ****   ****
  1.0   ****   ****   ****   ****

det(A) =  1.0

```

LU Factorization of a General Matrix in Banded Storage

The subroutines described in this section compute the LU factorization of a matrix A in banded storage. It is typical to follow a call to xGBFA with a call to xGBSL to solve $Ax = b$ or to xGBDI to compute the determinant of A.

Calling Sequence

```
CALL DGBFA (DA, LDA, N, NSUB, NSUPER, IPIVOT, INFO)
CALL SGBFA (SA, LDA, N, NSUB, NSUPER, IPIVOT, INFO)
CALL ZGBFA (ZA, LDA, N, NSUB, NSUPER, IPIVOT, INFO)
CALL CGBFA (CA, LDA, N, NSUB, NSUPER, IPIVOT, INFO)

void dgbfa (double *da, int lda, int n, int nsub, int nsuper, int
            *ipivot, int *info)
void sgbfa (float *sa, int lda, int n, int nsub, int nsuper, int
            *ipivot, int *info)
void zgbfa (doublecomplex *za, int lda, int n, int nsub, int
            nsuper, int *ipivot, int *info)
void cgbfa (complex *ca, int lda, int n, int nsub, int nsuper, int
            *ipivot, int *info)
```

Arguments

xA	On entry, the matrix A. On exit, an LU factorization of the matrix A.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq 2 \times NSUB + NSUPER + 1$.
N	Order of the matrix A. $N \geq 0$.
NSUB	Number of subdiagonals of A. $N-1 \geq NSUB \geq 0$ but if $N = 0$ then $NSUB = 0$.
NSUPER	Number of superdiagonals of A. $N-1 \geq NSUPER \geq 0$ but if $N = 0$ then $NSUPER = 0$.
IPIVOT	On exit, a vector of pivot indices.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO > 0 Returns a value k if $U(k,k) = 0$ to indicate that xGESL will divide by zero if called.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
INTEGER          IAXEQB, LDA, LDAB, N, NDIAG, NSUB, NSUPER
PARAMETER        (IAXEQB = 0)
PARAMETER        (N = 4)
PARAMETER        (LDA = N)
PARAMETER        (NSUB = 1)
PARAMETER        (NSUPER = 1)
PARAMETER        (NDIAG = NSUB + 1 + NSUPER)
PARAMETER        (LDAB = 2 * NSUB + 1 + NSUPER)

C
DOUBLE PRECISION AB(LDAB,N), AG(LDA,N), B(N)
INTEGER          ICOL, INFO, IPIVOT(N), IROW, IROWB, I1
INTEGER          I2, JOB

C
EXTERNAL         DGBFA, DGBSL
INTRINSIC        MAX0, MIN0

C
C Initialize the array AG to store the 4x4 matrix A with one
C subdiagonal and one superdiagonal shown below. Initialize
C the array B to store the vector b shown below.
C
C
C      2  -1
C  AG = -1  2  -1      b = 5
C      -1  2  -1      5
C      -1  2  -1      5
C      -1  2
C
C
C      DATA AB / 16*8D8 /
C      DATA AG /  2.0D0, -1.0D0,  2*0D0, -1.0D0,  2.0D0, -1.0D0,
$              2*0D0, -1.0D0,  2.0D0, -1.0D0,  2*0D0, -1.0D0,
$              2.0D0 /
C      DATA B / N*5.0D0 /

C
C Copy the matrix A from the array AG to the array AB. The
C matrix is stored in general storage mode in AG and it will
C be stored in banded storage mode in AB. The code to copy
C from general to banded storage mode is taken from the
C comment block in the original DGBFA by Cleve Moler.
C
C
C      DO 10, ICOL = 1, N
C          I1 = MAX0 (1, ICOL - NSUPER)
C          I2 = MIN0 (N, ICOL + NSUB)
C          DO 10, IROW = I1, I2
C              IROWB = IROW - ICOL + NDIAG
C              AB(IROWB,ICOL) = AG(IROW,ICOL)
10      CONTINUE
20      CONTINUE

C
C Print the initial values of the arrays.
C
C
PRINT 1000

```

```

        PRINT 1010, ((AG(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
        PRINT 1020
        PRINT 1010, ((AB(IROW,ICOL), ICOL = 1, N),
$       IROW = 2 * NSUB, 2 * NSUB + 1 + NSUPER)
        PRINT 1030
        PRINT 1040, B
C
C       Factor the matrix in banded form.
C
        CALL DGBFA (AB, LDA, N, NSUB, NSUPER, IPIVOT, INFO)
        IF (INFO .EQ. 0) THEN
            JOB = IAXEQB
            CALL DGBSL (AB, LDA, N, NSUB, NSUPER, IPIVOT, B, JOB)
            PRINT 1050
            PRINT 1040, B
        ELSE
            PRINT 1060
        END IF
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F4.1))
1020 FORMAT (/1X, 'A in banded form: (* in unused elements)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (3X, F4.1)
1050 FORMAT (/1X, 'A**(-1) * b:')
1060 FORMAT (1X, 'A is singular to working precision.')
C
        END

```

Sample Output

```

A in full form:
  2.0  -1.0   0.0   0.0
 -1.0   2.0  -1.0   0.0
  0.0  -1.0   2.0  -1.0
  0.0   0.0  -1.0   2.0

A in banded form: (* in unused elements)
****  -1.0  -1.0  -1.0
  2.0   2.0   2.0   2.0
 -1.0  -1.0  -1.0  ****

b:
  5.0
  5.0
  5.0
  5.0

```

```
A**(-1) * b:  
10.0  
15.0  
15.0  
10.0
```

Solution to a Linear System in an LU-Factored Matrix in Banded Storage

The subroutines described in this section solve the linear system $Ax = b$ for a matrix A in banded storage, which has been LU-factored by xGBCO or xGBFA, and vectors b and x .

Calling Sequence

```
CALL DGBSL (DA, LDA, N, NSUB, NSUPER, IPIVOT, DB, JOB)
CALL SGBSL (SA, LDA, N, NSUB, NSUPER, IPIVOT, SB, JOB)
CALL ZGBSL (ZA, LDA, N, NSUB, NSUPER, IPIVOT, ZB, JOB)
CALL CGBSL (CA, LDA, N, NSUB, NSUPER, IPIVOT, CB, JOB)

void dgbsl (double *da, int lda, int n, int nsub, int nsuper, int
            *ipivot, double *db, int job)
void sgbsl (float *sa, int lda, int n, int nsub, int nsuper, int
            *ipivot, float *sb, int job)
void zgbsl (doublecomplex *za, int lda, int n, int nsub, int
            nsuper, int *ipivot, doublecomplex *zb, int job)
void cgbsl (complex *ca, int lda, int n, int nsub, int nsuper, int
            *ipivot, complex *cb, int job)
```

Arguments

xA	LU factorization of the matrix A , as computed by xGBCO or xGBFA.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq 2 \times NSUB + NSUPER + 1$.
N	Order of the matrix A . $N \geq 0$.
NSUB	Number of subdiagonals of A . $N-1 \geq NSUB \geq 0$ but if $N = 0$ then $NSUB = 0$.
NSUPER	Number of superdiagonals of A . $N-1 \geq NSUPER \geq 0$ but if $N = 0$ then $NSUPER = 0$.
IPIVOT	Pivot vector as computed by xGBCO or xGBFA.
xB	On entry, the right-hand side vector b . On exit, the solution vector x .
JOB	Determines which operation this subroutine will perform: 0 solve the system $Ax = b$ not 0 solve the linear system $A^Hx = b$ Note that $A^Tx = A^Hx$ for real matrices.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER          IAXEQB, LDA, LDAB, N, NDIAG, NSUB, NSUPER
  PARAMETER        (IAXEQB = 0)
  PARAMETER        (N = 4)
  PARAMETER        (LDA = N)
  PARAMETER        (NSUB = 1)
  PARAMETER        (NSUPER = 1)
  PARAMETER        (NDIAG = NSUB + 1 + NSUPER)
  PARAMETER        (LDAB = 2 * NSUB + 1 + NSUPER)

C
  DOUBLE PRECISION AB(LDAB,N), AG(LDA,N), B(N), RCOND, WORK(N)
  INTEGER          ICOL, IPIVOT(N), IROW, IROWB, I1, I2, JOB

C
  EXTERNAL          DGBCO, DGBSL
  INTRINSIC         MAX0, MIN0

C
  C Initialize the array AG to store the 4x4 matrix A with one
  C subdiagonal and one superdiagonal shown below. Initialize
  C the array B to store the vector b shown below.
  C
  C
  C      2   -1           5
  C  AG = -1   2   -1      b = 5
  C          -1   2   -1      5
  C          -1   2   -1      5
  C
  C
  C      DATA AB / 16*8D8 /
  C      DATA AG /  2.0D0, -1.0D0,  2*0D0, -1.0D0,  2.0D0, -1.0D0,
  C $              2*0D0, -1.0D0,  2.0D0, -1.0D0,  2*0D0, -1.0D0,
  C $              2.0D0 /
  C      DATA B / N*5.0D0 /

C
  C Copy the matrix A from the array AG to the array AB. The
  C matrix is stored in general storage mode in AG and it will
  C be stored in banded storage mode in AB. The code to copy
  C from general to banded storage mode is taken from the
  C comment block in the original DGBFA by Cleve Moler.
  C
  DO 10, ICOL = 1, N
    I1 = MAX0 (1, ICOL - NSUPER)
    I2 = MIN0 (N, ICOL + NSUB)
    DO 10, IROW = I1, I2
      IROWB = IROW - ICOL + NDIAG
      AB(IROWB,ICOL) = AG(IROW,ICOL)
10    CONTINUE
20  CONTINUE

C
  C Print the initial values of the arrays.
  C
  C
  PRINT 1000
  PRINT 1010, ((AG(IROW,ICOL), ICOL = 1, N), IROW = 1, N)

```

```

        PRINT 1020
        PRINT 1010, ((AB(IROW,ICOL), ICOL = 1, N),
$         IROW = 2 * NSUB, 2 * NSUB + 1 + NSUPER)
        PRINT 1030
        PRINT 1040, B
C
C      Factor the matrix in banded form.
C
        CALL DGBCO (AB, LDA, N, NSUB, NSUPER, IPIVOT, RCOND, WORK)
        PRINT 1050, RCOND
        IF ((RCOND + 1.0D0) .EQ. 1.0D0) THEN
            PRINT 1070
        END IF
        JOB = IAXEQB
        CALL DGBSL (AB, LDA, N, NSUB, NSUPER, IPIVOT, B, JOB)
        PRINT 1060
        PRINT 1040, B
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F4.1))
1020 FORMAT (/1X, 'A in banded form: (* in unused elements)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (3X, F4.1)
1050 FORMAT (/1X, 'Reciprocal of the condition number: ', F5.2)
1060 FORMAT (/1X, 'A**(-1) * b:')
1070 FORMAT (1X, 'A may be singular to working precision.')
C
        END

```

Sample Output

```

A in full form:
  2.0  -1.0   0.0   0.0
 -1.0   2.0  -1.0   0.0
  0.0  -1.0   2.0  -1.0
  0.0   0.0  -1.0   2.0

A in banded form: (* in unused elements)
****  -1.0  -1.0  -1.0
      2.0   2.0   2.0   2.0
 -1.0  -1.0  -1.0  ****

b:
  5.0
  5.0
  5.0
  5.0

Reciprocal of the condition number:  0.08

```



```
A**(-1) * b:  
 10.0  
 15.0  
 15.0  
 10.0
```

LU Factorization and Condition Number of a General Matrix

The subroutines described in this section compute the LU factorization and estimate the condition number of a general matrix A. If the condition number is not needed then xGEFA is slightly faster. It is typical to follow a call to xGECO with a call to xGESL to solve $Ax = b$ or to xGEDI to compute the determinant and inverse of A.

Calling Sequence

```
CALL DGECO (DA, LDA, N, IPIVOT, DRCOND, DWORK)
CALL SGECO (SA, LDA, N, IPIVOT, SRCOND, SWORK)
CALL ZGECO (ZA, LDA, N, IPIVOT, DRCOND, ZWORK)
CALL CGECO (CA, LDA, N, IPIVOT, SRCOND, CWORK)

void dgeco (double *da, int lda, int n, int *ipivot, double
            *drcond)
void sgeco (float *sa, int lda, int n, int *ipivot, float *srcond)
void zgeco (doublecomplex *za, int lda, int n, int *ipivot, double
            *drcond)
void cgeco (complex *ca, int lda, int n, int *ipivot, float
            *srcond)
```

Arguments

xA	On entry, the matrix A. On exit, an LU factorization of A.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
N	Order of the matrix A. $N \geq 0$.
IPIVOT	On exit, a vector of pivot indices.
xRCOND	On exit, an estimate of the reciprocal condition number of A. $0.0 \leq RCOND \leq 1.0$. As the value of RCOND gets smaller, operations with A such as solving $Ax = b$ may become less stable. If RCOND satisfies $RCOND + 1.0 = 1.0$ then A may be singular to working precision.
xWORK	Scratch array with a dimension of N.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
INTEGER          IAXEQB, LDA, LDB, N
PARAMETER        (IAXEQB = 0)
PARAMETER        (N = 3)
PARAMETER        (LDA = N)
PARAMETER        (LDB = LDA)
DOUBLE PRECISION A(LDA,N), B(LDB), RCOND, WORK(N)
INTEGER          ICOL, IPIVOT(N), IROW, JOB

C
EXTERNAL          DGECCO, DGESL

C
C Initialize the array A to store the matrix A shown below.
C Initialize the array B to store the vector b shown below.
C
C      1   2   2           15
C  A =  2   1   2       b = 15
C      2   2   1           15
C
C
DATA A / 1.0D0, 3*2.0D0, 1.0D0, 3*2.0D0, 1.0D0 /
DATA B / 3*1.5D1 /
PRINT 1000
PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
PRINT 1020
PRINT 1030, B
CALL DGECCO (A, LDA, N, IPIVOT, RCOND, WORK)
PRINT 1040, RCOND
IF ((RCOND + 1.0D0) .EQ. 1.0D0) THEN
  PRINT 1060
END IF
JOB = IAXEQB
CALL DGESL (A, LDA, N, IPIVOT, B, JOB)
PRINT 1050
PRINT 1030, B

C
1000 FORMAT (1X, 'A:')
1010 FORMAT (3(3X, F4.1))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (3X, F4.1)
1040 FORMAT (/1X, 'Estimated reciprocal condition number:',
F7.4)
1050 FORMAT (/1X, 'A**(-1) * b:')
1060 FORMAT (1X, 'A may be singular to working precision.')

C
END

```

Sample Output

```
A:
  1.0    2.0    2.0
  2.0    1.0    2.0
  2.0    2.0    1.0

b:
 15.0
 15.0
 15.0

Estimated reciprocal condition number: 0.1429

A**(-1) * b:
  3.0
  3.0
  3.0
```

Determinant and Inverse of an LU-Factored General Matrix

The subroutines described in this section compute the determinant and inverse of a general matrix A, which has been LU-factored by xGECO or xGEFA.

Calling Sequence

```
CALL DGEDI (DA, LDA, N, IPIVOT, DDET, DWORK, JOB)
CALL SGEDI (SA, LDA, N, IPIVOT, SDET, SWORK, JOB)
CALL ZGEDI (ZA, LDA, N, IPIVOT, ZDET, ZWORK, JOB)
CALL CGEDI (CA, LDA, N, IPIVOT, CDET, CWORK, JOB)

void dgedi (double *da, int lda, int n, int *ipivot, double *ddet,
            int job)
void sgedi (float *sa, int lda, int n, int *ipivot, float *sdet,
            int job)
void zgedi (doublecomplex *za, int lda, int n, int *ipivot,
            doublecomplex *zdet, int job)
void cgedi (complex *ca, int lda, int n, int *ipivot, complex
            *cdet, int job)
```

Arguments

xA	On entry, the LU factorization of the matrix A, as computed by xGECO or xGEFA. On exit, the inverse of the original matrix A if the inverse was requested, otherwise it is unchanged.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
N	Order of the original matrix A. $N \geq 0$.
IPIVOT	Pivot vector as computed by xGECO or xGEFA.
xDET	On exit, the determinant of the matrix A. The determinant is stored as $b \times 10^{\text{expon}}$ where b is stored in DET(1) and expon is stored in DET(2). $1.0 \leq \text{DET}(1) < 10.0$ or $\text{DET}(1) = 0.0$.
xWORK	Scratch array with a dimension of N.
JOB	Determines which operation the subroutine will perform: 11 both determinant and inverse 01 inverse only 10 determinant only

Sample Program

```

C      PROGRAM TEST
C      IMPLICIT NONE
C
C      INTEGER          INVDET, LDA, N
C      PARAMETER        (INVDET = 11)
C      PARAMETER        (N = 4)
C      PARAMETER        (LDA = N)
C
C      DOUBLE PRECISION A(LDA,N), DET(2), RCOND, WORK(N)
C      INTEGER          ICOL, IPIVOT(N), IROW, JOB
C
C      EXTERNAL          DGECCO, DGEDII
C
C      Initialize the array A to store the matrix A shown below.
C      Its exact inverse also is shown below.
C
C      A =
C      -3      1
C      1      -2      1
C      1      -2      1
C      1      -1
C
C      A**(-1) =
C      -      1      1      1      1
C      -      1      3      3      3
C      2      1      3      5      5
C      1      3      5      7
C
C      DATA A / -3.0D0, 1.0D0, 2*0.0D0, 1.0D0, -2.0D0, 1.0D0,
C      $      2*0.0D0, 1.0D0, -2.0D0, 1.0D0, 2*0.0D0, 1.0D0,
C      $      -1.0D0 /
C
C      PRINT 1000
C      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
C      CALL DGECCO (A, LDA, N, IPIVOT, RCOND, WORK)
C      JOB = INVDET
C      CALL DGEDII (A, LDA, N, IPIVOT, DET, WORK, JOB)
C      PRINT 1020
C      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
C      PRINT 1030, DET(1) * (1.0D1 ** DET(2))
C
C      1000 FORMAT (1X, 'A:')
C      1010 FORMAT (4(3X, F5.1))
C      1020 FORMAT (/1X, 'A**(-1):')
C      1030 FORMAT (/1X, 'det(A) =', F6.1)
C
C      END

```

Sample Output

A:

-3.0	1.0	0.0	0.0
1.0	-2.0	1.0	0.0
0.0	1.0	-2.0	1.0
0.0	0.0	1.0	-1.0

```
A**(-1):  
  -0.5   -0.5   -0.5   -0.5  
  -0.5   -1.5   -1.5   -1.5  
  -0.5   -1.5   -2.5   -2.5  
  -0.5   -1.5   -2.5   -3.5  
  
det(A) =   2.0
```

LU Factorization of a General Matrix

The subroutines described in this section compute the LU factorization of a general matrix A. It is typical to follow a call to xGEFA with a call to xGESL to solve $Ax = b$ or to xGEDI to compute the determinant of A.

Calling Sequence

```
CALL DGEFA (DA, LDA, N, IPIVOT, INFO)
CALL SGEFA (SA, LDA, N, IPIVOT, INFO)
CALL ZGEFA (ZA, LDA, N, IPIVOT, INFO)
CALL CGEFA (CA, LDA, N, IPIVOT, INFO)

void dgefa (double *da, int lda, int n, int *ipivot, int *info)
void sgefa (float *sa, int lda, int n, int *ipivot, int *info)
void zgefa (doublecomplex *za, int lda, int n, int *ipivot, int *info)
void cgefa (complex *ca, int lda, int n, int *ipivot, int *info)
```

Arguments

xA	On entry, the matrix A. On exit, an LU factorization of A.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
N	Order of the matrix A. $N \geq 0$.
IPIVOT	On exit, a vector of pivot indices.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO > 0 Returns a value k if $U(k, k) = 0$ to indicate that xGESL will divide by zero if called.

Sample Program

```
PROGRAM TEST
IMPLICIT NONE

C      INTEGER          LDA, N
C      PARAMETER        (N = 3)
C      PARAMETER        (LDA = N)
```



```

      DOUBLE PRECISION A(LDA,N)
      INTEGER          ICOL, INFO, IPIVOT(N), IROW
C
      EXTERNAL          DGEFA
C
      Initialize the array A to store the matrix A shown below.
C
C      100    2    3
C      A = 100  202  6
C      100  202  306
C
      DATA A / 1.0D2, 1.0D2, 1.0D2, 2.0D0, 2.02D2, 2.02D2,
$             3.0D0, 6.0D0, 3.06D2 /
      PRINT 1000
      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
      CALL DGEFA (A, LDA, N, IPIVOT, INFO)
      IF (INFO .EQ. 0) THEN
        PRINT 1020
        PRINT 1030, (A(1,ICOL), ICOL = 1, N)
        PRINT 1040, (A(2,ICOL), ICOL = 1, N)
        PRINT 1050, (A(3,ICOL), ICOL = 1, N)
      ELSE
        PRINT 1060, INFO
      END IF
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (3(3X, F5.1))
1020 FORMAT (/8X, 'Multipliers', 19X, 'Upper')
1030 FORMAT (3X, ' 1.0', 17X, 3(2X, F5.1))
1040 FORMAT (3X, F5.1, 2X, ' 1.0', 17X, 2(2X, F5.1))
1050 FORMAT (1X, 2(2X, F5.1), 2X, ' 1.0', 17X, 2X, F5.1)
1060 FORMAT (1X, 'A appears singular at ', I2)
C
      END

```

Sample Output

```

A:
100.0    2.0    3.0
100.0   202.0    6.0
100.0   202.0   306.0

      Multipliers              Upper
      1.0                    100.0    2.0    3.0
      1.0    1.0            200.0    3.0
      1.0    1.0    1.0          300.0

```

Solution to a Linear System in an LU-Factored General Matrix

The subroutines described in this section solve the linear system $Ax = b$ for a general matrix A , which has been LU-factored by xGECO or xGEFA, and vectors b and x .

Calling Sequence

```
CALL DGESL (DA, LDA, N, IPIVOT, DB, JOB)
CALL SGESL (SA, LDA, N, IPIVOT, SB, JOB)
CALL ZGESL (ZA, LDA, N, IPIVOT, ZB, JOB)
CALL CGESL (CA, LDA, N, IPIVOT, CB, JOB)

void dgesl (double *da, int lda, int n, int *ipivot, double *db,
            int job)
void sgesl (float *sa, int lda, int n, int *ipivot, float *sb, int
            job)
void zgesl (doublecomplex *za, int lda, int n, int *ipivot,
            doublecomplex *zb, int job)
void cgesl (complex *ca, int lda, int n, int *ipivot, complex *cb,
            int job)
```

Arguments

xA	LU factorization of the matrix A , as computed by xGECO or xGEFA.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
N	Order of the matrix A . $N \geq 0$.
IPIVOT	Pivot vector as computed by xGECO or xGEFA.
xB	On entry, the right-hand side vector b . On exit, the solution vector x .
JOB	Determines which operation the subroutine will perform: 0 solve the system $Ax = b$. not 0 solve the linear system $A^H x = b$ Note that $A^T x = A^H x$ for real matrices.

Sample Program

```
PROGRAM TEST
IMPLICIT NONE
```

```

C      INTEGER          IAXEQB, LDA, LDB, N
C      PARAMETER        (IAXEQB = 0)
C      PARAMETER        (N = 3)
C      PARAMETER        (LDA = N)
C      PARAMETER        (LDB = LDA)
C
C      DOUBLE PRECISION  A(LDA,N), B(LDB)
C      INTEGER           ICOL, INFO, IPIVOT(N), IROW, JOB
C
C      EXTERNAL          DGEFA, DGESL
C
C      Initialize the array A to store the matrix A shown below.
C      Initialize the array B to store the vector b shown below.
C
C      A =  $\begin{bmatrix} 1 & 2 & 2 \\ 2 & 1 & 2 \\ 2 & 2 & 1 \end{bmatrix}$    b =  $\begin{bmatrix} 15 \\ 15 \\ 15 \end{bmatrix}$ 
C
C      DATA A / 1.0D0, 3*2.0D0, 1.0D0, 3*2.0D0, 1.0D0 /
C      DATA B / 3*1.5D1 /
C
C      PRINT 1000
C      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
C      PRINT 1020
C      PRINT 1030, B
C      CALL DGEFA (A, LDA, N, IPIVOT, INFO)
C      IF (INFO .EQ. 0) THEN
C         JOB = IAXEQB
C         CALL DGESL (A, LDA, N, IPIVOT, B, JOB)
C         PRINT 1040
C         PRINT 1030, B
C      ELSE
C         PRINT 1050, INFO
C      END IF
C
C      1000 FORMAT (1X, 'A:')
C      1010 FORMAT (3(3X, F4.1))
C      1020 FORMAT (/1X, 'b:')
C      1030 FORMAT (1X, 2X, F4.1)
C      1040 FORMAT (/1X, 'A**(-1)*b')
C      1050 FORMAT (1X, 'A appears singular at ', I2)
C
C      END

```

Sample Output

```

A:
  1.0    2.0    2.0
  2.0    1.0    2.0
  2.0    2.0    1.0

```

```
b:
  15.0
  15.0
  15.0
A**(-1)*b
  3.0
  3.0
  3.0
```

Solution to a Linear System in a Tridiagonal Matrix

The subroutines described in this section solve the linear system $Ax = b$ for a tridiagonal matrix A and vectors b and x .

Calling Sequence

```
CALL DGTSL (N, DSUB, DDIAG, DSUPER, DB, INFO)
CALL SGTSL (N, SSUB, SDIAG, SSUPER, SB, INFO)
CALL ZGTSL (N, ZSUB, ZDIAG, ZSUPER, ZB, INFO)
CALL CGTSL (N, CSUB, CDIAG, CSUPER, CB, INFO)

void dgtsl (int n, double *dsub, double *ddiag, double *dsuper,
double *db, int *info)
void sgtsl (int n, float *ssub, float *sdiag, float *ssuper, float
*sb, int *info)
void zgtsl (int n, doublecomplex *zsub, doublecomplex *zdiag,
doublecomplex *zsuper, doublecomplex *zb, int *info)
void cgtsl (int n, complex *csub, complex *cdiag, complex *csuper,
complex *cb, int *info)
```

Arguments

N	Order of the matrix A . $N \geq 0$.
xSUB	Subdiagonal of A . SUB(2) through SUB(N) contain the subdiagonal and SUB(1) is not referenced.
xDIAG	Diagonal of A .
xSUPER	Superdiagonal of A . SUPER(1) through SUPER(N-1) contain the superdiagonal and SUPER(N) is not referenced.
xB	On entry, the right-hand side vector b . On exit, the solution vector x .
INFO	On exit: INFO = 0 Subroutine completed normally. INFO > 0 Returns a value k when the k th element of the diagonal becomes exactly zero.

Sample Program

```
PROGRAM TEST
IMPLICIT NONE
```

```
C
```

```

C      INTEGER          N
C      PARAMETER      (N = 5)

C      DOUBLE PRECISION B(N), SUBD(N), DIAG(N), SUPERD(N)
C      INTEGER          INFO

C      EXTERNAL          DGTSL

C      Initialize the arrays SUBD, DIAG, and SUPERD to store
C      the subdiagonal, diagonal, and superdiagonal of the
C      tridiagonal matrix A shown below. Initialize the array
C      B to store the right hand side of Ax=b.
C
C      -3   1           6
C      1  -2   1           12
C      A =      1  -2   1      b = 18
C              1  -2   1      12
C              1  -1      6
C
C      DATA SUBD / 8D8, 4*1.0D0 /
C      DATA DIAG / -3.0D0, -2.0D0, -2.0D0, -2.0D0, -1.0D0 /
C      DATA SUPERD / 4*1.0D0, 8D8 /
C      DATA B / 6.0D0, 1.2D1, 1.8D1, 1.2D1, 6.0D0 /

C      PRINT 1000
C      PRINT 1010, DIAG(1), SUPERD(1)
C      PRINT 1020, SUBD(2), DIAG(2), SUPERD(2)
C      PRINT 1030, SUBD(3), DIAG(3), SUPERD(3)
C      PRINT 1040, SUBD(4), DIAG(4), SUPERD(4)
C      PRINT 1050, SUBD(5), DIAG(5)
C      PRINT 1060
C      PRINT 1070, B
C      CALL DGTSL (N, SUBD, DIAG, SUPERD, B, INFO)
C      IF (INFO .EQ. 0) THEN
C          PRINT 1080
C          PRINT 1070, B
C      ELSE
C          PRINT 1090, INFO
C      END IF

C      1000 FORMAT (1X, 'A:')
C      1010 FORMAT (1X, 2(2X, F4.1))
C      1020 FORMAT (1X, 3(2X, F4.1))
C      1030 FORMAT (1X, 6X, 3(2X, F4.1))
C      1040 FORMAT (1X, 12X, 3(2X, F4.1))
C      1050 FORMAT (1X, 18X, 2(2X, F4.1))
C      1060 FORMAT (/1X, 'b:')
C      1070 FORMAT (3X, F6.1)
C      1080 FORMAT (/1X, 'A**(-1) * b:')
C      1090 FORMAT (/1X, 'A appears singular at ', I1)

C      END

```

Sample Output

```
A:
  -3.0   1.0
    1.0  -2.0   1.0
          1.0  -2.0   1.0
                1.0  -2.0   1.0
                      1.0  -1.0

b:
    6.0
   12.0
   18.0
   12.0
    6.0

A**(-1) * b:
  -27.0
  -75.0
 -111.0
 -129.0
 -135.0
```

UDU Factorization and Condition Number of a Hermitian Matrix

The subroutines described in this section compute the UDU factorization and condition number of a Hermitian matrix A. If the condition number is not needed then xHIFA is slightly faster. It is typical to follow a call to xHICO with a call to xHISL to solve $Ax = b$ or to xHIDI to compute the determinant, inverse, and inertia of A.

Calling Sequence

```
CALL ZHICO  (ZA, LDA, N, IPIVOT, DRCOND, ZWORK)
CALL CHICO  (CA, LDA, N, IPIVOT, SRCOND, CWORK)

void zhico  (doublecomplex *za, int lda, int n, int *ipivot, double
             *drcond)
void chico  (complex *ca, int lda, int n, int *ipivot, float
             *srcond)
```

Arguments

xA	On entry, the upper triangle of the matrix A. On exit, a UDU factorization of the matrix A. The strict lower triangle of A is not referenced.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
N	Order of the matrix A. $N \geq 0$.
IPIVOT	On exit, a vector of pivot indices.
xRCOND	On exit, an estimate of the reciprocal condition number of A. $0.0 \leq RCOND \leq 1.0$. As the value of RCOND gets smaller, operations with A such as solving $Ax = b$ may become less stable. If RCOND satisfies $RCOND + 1.0 = 1.0$ then A may be singular to working precision.
xWORK	Scratch array with a dimension of N.

Sample Program

```
PROGRAM TEST
IMPLICIT NONE
C
INTEGER    LDA, N
PARAMETER (N = 3)
```



```

C      PARAMETER (LDA = 3)
C      REAL      RCOND
C      COMPLEX    A(LDA,N), B(N), WORK(N)
C      INTEGER    ICOL, IPIVOT(N), IROW
C
C      EXTERNAL   CHICO, CHISL
C      INTRINSIC  CONJG
C
C      Initialize the array A to store the matrix A shown below.
C      Initialize the array B to store the vector b shown below.
C
C      1      1+2i  1+2i      95-180i
C      A = 1+2i   6    -2+6i    b = 545-118i
C      1+2i  -2+6i   11      865+ 62i
C
C      DATA A / (1.0,0.0), (8E8,8E8), (8E8,8E8),
C      $          (1.0,-2.0), (6.0,0.0), (8E8,8E8),
C      $          (1.0,-2.0), (6.0,-2.0), (11.0,0.0) /
C      DATA B / (95.0,-180.0), (545.0,-118.0), (865.0,62.0) /
C
C      PRINT 1000
C      DO 100, IROW = 1, N
C          PRINT 1010, (CONJG(A(ICOL,IROW)), ICOL = 1, IROW),
C      $              (A(IROW,ICOL), ICOL = IROW + 1, N)
100  CONTINUE
C      PRINT 1020
C      DO 110, IROW = 1, N
C          PRINT 1010, (A(IROW,ICOL), ICOL = 1, N)
110  CONTINUE
C      PRINT 1030
C      PRINT 1040, B
C      CALL CHICO (A, LDA, N, IPIVOT, RCOND, WORK)
C      PRINT 1050, RCOND
C      IF ((RCOND + 1.0) .EQ. 1.0) THEN
C          PRINT 1060
C      END IF
C      CALL CHISL (A, LDA, N, IPIVOT, B)
C      PRINT 1070
C      PRINT 1040, B
C
C      1000 FORMAT (1X, 'A in full form:')
C      1010 FORMAT (4(: 3X, '(', F4.1, ', ', F4.1, ')'))
C      1020 FORMAT (/1X, 'A in Hermitian form: (* in unused elements)')
C      1030 FORMAT (/1X, 'b:')
C      1040 FORMAT (3X, '(', F6.1, ', ', F6.1, ')')
C      1050 FORMAT (/1X, 'Reciprocal condition number of A:', F6.3)
C      1060 FORMAT (1X, 'A may be singular to working precision.')
C      1070 FORMAT (/1X, 'A**(-1) * b:')
C
C      END

```

Sample Output

```
A in full form:
( 1.0, 0.0) ( 1.0, -2.0) ( 1.0, -2.0)
( 1.0, 2.0) ( 6.0, 0.0) ( 6.0, -2.0)
( 1.0, 2.0) ( 6.0, 2.0) (11.0, 0.0)

A in Hermitian form: (* in unused elements)
( 1.0, 0.0) ( 1.0, -2.0) ( 1.0, -2.0)
(****, ****) ( 6.0, 0.0) ( 6.0, -2.0)
(****, ****) (****, ****) (11.0, 0.0)

b:
( 95.0, -180.0)
( 545.0, -118.0)
( 865.0, 62.0)

Reciprocal condition number of A: 0.001

A**(-1) * b:
( 5.0, 0.0)
( 26.0, 0.0)
( 64.0, 0.0)
```

Determinant, Inertia, and Inverse of a UDU-Factored Hermitian Matrix

The subroutines described in this section compute the determinant, inertia, and inverse of a Hermitian matrix A, which has been UDU-factored by xHICO or xHIFA.

Calling Sequence

```
CALL ZHIDI (ZA, LDA, N, IPIVOT, DDET, INERT, ZWORK, JOB)
CALL CHIDI (CA, LDA, N, IPIVOT, SDET, INERT, CWORK, JOB)

void zhidi (doublecomplex *za, int lda, int n, int *ipivot, double
            *ddet, int *inert, int job)
void chidi (complex *ca, int lda, int n, int *ipivot, float *sdet,
            int *inert, int job)
```

Arguments

xA	On entry, the UDU factorization of the matrix A, as computed by xHICO or xHIFA. On exit, if the <i>c</i> digit of JOB $\neq 0$, then the upper triangle of A contains the upper triangle of the inverse of the original matrix A if the inverse was requested, otherwise unchanged. The strict lower triangle of A is not referenced.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
N	Order of the original matrix A. $N \geq 0$.
IPIVOT	Pivot vector as computed by xHICO or xHIFA.
xDET	On exit, if the <i>b</i> digit of JOB $\neq 0$, then DET contains the determinant of the matrix A. The determinant is stored as $b \times 10^{\text{expon}}$ where <i>b</i> is stored in DET(1) and expon is stored in DET(2). $1.0 \leq \text{DET}(1) < 10.0$ or $\text{DET}(1) = 0.0$. If the <i>b</i> digit of JOB = 0, DET is not referenced.
INERT	On exit, if the <i>a</i> digit of JOB $\neq 0$, then INERT contains an integer triplet where: INERT(1) = number of positive eigenvalues INERT(2) = number of negative eigenvalues INERT(3) = number of zero eigenvalues If the <i>a</i> digit of JOB = 0 then INERT is not referenced.
xWORK	Scratch array with a dimension of N.
JOB	Integer in the form abc; determines operation the subroutine will perform:

a ≠ 0 Compute the inertia.
b ≠ 0 Compute the determinant.
c ≠ 0 Compute the inverse.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
  INTEGER    IDODET, IDOINR, IDOINV, LDA, N
  PARAMETER (IDODET = 10)
  PARAMETER (IDOINR = 100)
  PARAMETER (IDOINV = 1)
  PARAMETER (N = 3)
  PARAMETER (LDA = 3)
C
  REAL       DET(2), RCOND
  COMPLEX    A(LDA,N), WORK(N)
  INTEGER    ICOL, INERT(3), IPIVOT(N), IROW, JOB
C
  EXTERNAL   CHICO, CHIDI
C
  C
  C   Initialize the array A to store the matrix A shown below.
  C
  C
  C       1    1+2i  1+2i
  C   A = 1+2i    6   -2+6i
  C       1+2i -2+6i   11
  C
  C   DATA A / (1.0,0.0), (8E8,8E8), (8E8,8E8),
  $             (1.0,-2.0), (6.0,0.0), (8E8,8E8),
  $             (1.0,-2.0), (6.0,-2.0), (11.0,0.0) /
C
  PRINT 1000
  DO 100, IROW = 1, N
    PRINT 1010, (CONJG(A(ICOL,IROW)), ICOL = 1, IROW),
  $             (A(IROW,ICOL), ICOL = IROW + 1, N)
100 CONTINUE
  PRINT 1020
  DO 110, IROW = 1, N
    PRINT 1010, (A(IROW,ICOL), ICOL = 1, N)
110 CONTINUE
  CALL CHICO (A, LDA, N, IPIVOT, RCOND, WORK)
  PRINT 1030, RCOND
  IF ((RCOND + 1.0) .EQ. 1.0) THEN
    PRINT 1040
  END IF
  JOB = IDOINR + IDODET + IDOINV
  CALL CHIDI (A, LDA, N, IPIVOT, DET, INERT, WORK, JOB)
  PRINT 1050, DET(1) * (10.0D0 ** DET(2))
  PRINT 1060, INERT
  PRINT 1070
  DO 120, IROW = 1, N

```

```

      PRINT 1010, (CONJG(A(ICOL,IROW)), ICOL = 1, IROW),
$      (A(IROW,ICOL), ICOL = IROW + 1, N)
120 CONTINUE
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(: 3X, '(', F5.1, ',', F5.1, ')'))
1020 FORMAT (/1X, 'A in Hermitian form: (* in unused elements)')
1030 FORMAT (/1X, 'Reciprocal condition number of A:', F6.3)
1040 FORMAT (1X, 'A may be singular to working precision.')
1050 FORMAT (/1X, 'Determinant of A: ', F6.3)
1060 FORMAT (1X, 'Inertia of A: <', I1, ',', I1, ',', I1, '>')
1070 FORMAT (/1X, 'A**(-1):')
C
      END

```

Sample Output

```

A in full form:
( 1.0, 0.0) ( 1.0, -2.0) ( 1.0, -2.0)
( 1.0, 2.0) ( 6.0, 0.0) ( 6.0, -2.0)
( 1.0, 2.0) ( 6.0, 2.0) ( 11.0, 0.0)

A in Hermitian form: (* in unused elements)
( 1.0, 0.0) ( 1.0, -2.0) ( 1.0, -2.0)
(*****,*****) ( 6.0, 0.0) ( 6.0, -2.0)
(*****,*****) (*****,*****) ( 11.0, 0.0)

Reciprocal condition number of A: 0.001

Determinant of A: 1.000
Inertia of A: <3,0,0>

A**(-1):
( 26.0, 0.0) ( -1.0, 12.0) ( -4.0, -2.0)
( -1.0, -12.0) ( 6.0, 0.0) ( -1.0, 2.0)
( -4.0, 2.0) ( -1.0, -2.0) ( 1.0, 0.0)

```

UDU Factorization of a Hermitian Matrix

The subroutines described in this section compute the UDU factorization of a Hermitian matrix A. It is typical to follow a call to xHIFA with a call to xHISL to solve $Ax = b$ or to xHIDI to compute the determinant, inverse, and inertia of A.

Calling Sequence

```
CALL ZHIFA  (ZA, LDA, N, IPIVOT, INFO)
CALL CHIFA  (CA, LDA, N, IPIVOT, INFO)

void zhifa  (doublecomplex *za, int lda, int n, int *ipivot, int
             *info)
void chifa  (complex *ca, int lda, int n, int *ipivot, int *info)
```

Arguments

xA	On entry, the upper triangle of the matrix A. On exit, a UDU factorization of the matrix A. The strict lower triangle of A is not referenced.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
N	Order of the matrix A. $N \geq 0$.
IPIVOT	On exit, a vector of pivot indices.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO > 0 Returns a value k if the kth pivot block is singular to indicate that xHISL or xHIDI will divide by zero if called.

Sample Program

```
PROGRAM TEST
IMPLICIT NONE

C
  INTEGER    LDA, N
  PARAMETER (N = 3)
  PARAMETER (LDA = 3)

C
  COMPLEX    A(LDA,N), B(N)
  INTEGER    ICOL, INFO, IPIVOT(N), IROW
```

```

C      EXTERNAL  CHIFA, CHISL
C      INTRINSIC CONJG
C
C      Initialize the array A to store the matrix A shown below.
C      Initialize the array B to store the vector b shown below.
C
C      1      1+2i  1+2i      95-180i
C      A = 1+2i   6   -2+6i    b = 545-118i
C      1+2i -2+6i   11      865+ 62i
C
C      DATA A / (1.0,0.0), (8E8,8E8), (8E8,8E8),
$             (1.0,-2.0), (6.0,0.0), (8E8,8E8),
$             (1.0,-2.0), (6.0,-2.0), (11.0,0.0) /
C      DATA B / (95.0,-180.0), (545.0,-118.0), (865.0,62.0) /
C
C      PRINT 1000
C      DO 100, IROW = 1, N
C          PRINT 1010, (CONJG(A(ICOL,IROW)), ICOL = 1, IROW),
$              (A(IROW,ICOL), ICOL = IROW + 1, N)
100  CONTINUE
C      PRINT 1020
C      DO 110, IROW = 1, N
C          PRINT 1010, (A(IROW,ICOL), ICOL = 1, N)
110  CONTINUE
C      PRINT 1030
C      PRINT 1040, B
C      CALL CHIFA (A, LDA, N, IPIVOT, INFO)
C      IF (INFO .EQ. 0) THEN
C          CALL CHISL (A, LDA, N, IPIVOT, B)
C          PRINT 1050
C          PRINT 1040, B
C      ELSE
C          PRINT 1060
C      END IF
C
C      1000 FORMAT (1X, 'A in full form:')
C      1010 FORMAT (4(: 3X, '(', F4.1, ', ', F4.1, ')'))
C      1020 FORMAT (/1X, 'A in Hermitian form: (* in unused elements)')
C      1030 FORMAT (/1X, 'b:')
C      1040 FORMAT (3X, '(', F6.1, ', ', F6.1, ')')
C      1050 FORMAT (/1X, 'A**(-1) * b:')
C      1060 FORMAT (1X, 'A is singular to working precision.')
C
C      END

```

Sample Output

```

A in full form:
( 1.0, 0.0) ( 1.0,-2.0) ( 1.0,-2.0)
( 1.0, 2.0) ( 6.0, 0.0) ( 6.0,-2.0)
( 1.0, 2.0) ( 6.0, 2.0) (11.0, 0.0)

```

```
A in Hermitian form: (* in unused elements)
( 1.0, 0.0) ( 1.0, -2.0) ( 1.0, -2.0)
(****, ****) ( 6.0, 0.0) ( 6.0, -2.0)
(****, ****) (****, ****) (11.0, 0.0)

b:
( 95.0, -180.0)
( 545.0, -118.0)
( 865.0, 62.0)

A**(-1) * b:
( 5.0, 0.0)
( 26.0, 0.0)
( 64.0, 0.0)
```


Solution to a Linear System in a UDU-Factored Hermitian Matrix

The subroutines described in this section solve the linear system $Ax = b$ for a Hermitian matrix A , which has been UDU-factored by xHICO or xHIFA, and vectors b and x .

Calling Sequence

```
CALL ZHISL  (ZA, LDA, N, IPIVOT, ZB)
CALL CHISL  (CA, LDA, N, IPIVOT, CB)

void zhisl  (doublecomplex *za, int lda, int n, int *ipivot,
             doublecomplex *zb)
void chisl  (complex *ca, int lda, int n, int *ipivot, complex *cb)
```

Arguments

xA	UDU factorization of the matrix A , as computed by xHICO or xHIFA.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
N	Order of the matrix A . $N \geq 0$.
IPIVOT	Pivot vector as computed by xHICO or xHIFA.
xB	On entry, the right-hand side vector b .
	On exit, the solution vector x .

Sample Program

```
PROGRAM TEST
IMPLICIT NONE
C
C   INTEGER    LDA, N
C   PARAMETER (N = 3)
C   PARAMETER (LDA = 3)
C
C   REAL       RCOND
C   COMPLEX    A(LDA,N), B(N), WORK(N)
C   INTEGER    ICOL, IPIVOT(N), IROW
C
C   EXTERNAL   CHICO, CHISL
C   INTRINSIC  CONJG
C
C   Initialize the array A to store the matrix A shown below.
C   Initialize the array B to store the vector b shown below.
C
```

```

C      1      1+2i  1+2i      95-180i
C      A = 1+2i   6    -2+6i    b = 545-118i
C      1+2i  -2+6i   11      865+ 62i
C
C      DATA A / (1.0,0.0), (8E8,8E8), (8E8,8E8),
$          (1.0,-2.0), (6.0,0.0), (8E8,8E8),
$          (1.0,-2.0), (6.0,-2.0), (11.0,0.0) /
C      DATA B / (95.0,-180.0), (545.0,-118.0), (865.0,62.0) /
C
C      PRINT 1000
C      DO 100, IROW = 1, N
C          PRINT 1010, (CONJG(A(ICOL,IROW)), ICOL = 1, IROW),
$              (A(IROW,ICOL), ICOL = IROW + 1, N)
100  CONTINUE
C      PRINT 1020
C      DO 110, IROW = 1, N
C          PRINT 1010, (A(IROW,ICOL), ICOL = 1, N)
110  CONTINUE
C      PRINT 1030
C      PRINT 1040, B
C      CALL CHICO (A, LDA, N, IPIVOT, RCOND, WORK)
C      PRINT 1050, RCOND
C      IF ((RCOND + 1.0) .EQ. 1.0) THEN
C          PRINT 1060
C      END IF
C      CALL CHISL (A, LDA, N, IPIVOT, B)
C      PRINT 1070
C      PRINT 1040, B
C
C      1000 FORMAT (1X, 'A in full form:')
C      1010 FORMAT (4(: 3X, '(', F4.1, ', ', F4.1, ')'))
C      1020 FORMAT (/1X, 'A in Hermitian form: (* in unused elements)')
C      1030 FORMAT (/1X, 'b:')
C      1040 FORMAT (3X, '(', F6.1, ', ', F6.1, ')')
C      1050 FORMAT (/1X, 'Reciprocal condition number of A:', F6.3)
C      1060 FORMAT (1X, 'A may be singular to working precision.')
C      1070 FORMAT (/1X, 'A**(-1) * b:')
C
C      END

```

Sample Output

```

A in full form:
( 1.0, 0.0) ( 1.0,-2.0) ( 1.0,-2.0)
( 1.0, 2.0) ( 6.0, 0.0) ( 6.0,-2.0)
( 1.0, 2.0) ( 6.0, 2.0) (11.0, 0.0)

A in Hermitian form: (* in unused elements)
( 1.0, 0.0) ( 1.0,-2.0) ( 1.0,-2.0)
(****, ****) ( 6.0, 0.0) ( 6.0,-2.0)
(****, ****) (****, ****) (11.0, 0.0)

```

```
b:
( 95.0, -180.0)
( 545.0, -118.0)
( 865.0, 62.0)

Reciprocal condition number of A: 0.001

A**(-1) * b:
( 5.0, 0.0)
( 26.0, 0.0)
( 64.0, 0.0)
```

UDU Factorization and Condition Number of a Hermitian Matrix in Packed Storage

The subroutines described in this section compute the UDU factorization and condition number of a Hermitian matrix A in packed storage. If the condition number is not needed then xHPFA is slightly faster. It is typical to follow a call to xHPCO with a call to xHPSL to solve $Ax = b$ or to xHPDI to compute the determinant, inverse, and inertia of A.

Calling Sequence

```
CALL ZHPCO  (ZA, N, IPIVOT, DRCOND, ZWORK)
CALL CHPCO  (CA, N, IPIVOT, SRCOND, CWORK)

void zhpco  (doublecomplex *za, int n, int *ipivot, double *drcond)
void chpco  (complex *ca, int n, int *ipivot, float *srcond)
```

Arguments

xA	On entry, the upper triangle of the matrix A. On exit, a UDU factorization of the matrix A.
N	Order of the matrix A. $N \geq 0$.
IPIVOT	On exit, a vector of pivot indices.
xRCOND	On exit, an estimate of the reciprocal condition number of A. $0.0 \leq RCOND \leq 1.0$. As the value of RCOND gets smaller, operations with A such as solving $Ax = b$ may become less stable. If RCOND satisfies $RCOND + 1.0 = 1.0$ then A may be singular to working precision.
xWORK	Scratch array with a dimension of N.

Sample Program

```
PROGRAM TEST
IMPLICIT NONE

C
INTEGER    LENGTA, N
PARAMETER (N = 3)
PARAMETER (LENGTA = (N * N + N) / 2)

C
REAL       RCOND
COMPLEX    A(LENGTA), B(N), WORK(N)
INTEGER    IPIVOT(N)
```

```

C      EXTERNAL  CHPCO, CHPSL
C      INTRINSIC CONJG
C
C      Initialize the array A to store the matrix A shown below.
C      Initialize the array B to store the vector b shown below.
C
C      1      1+2i  1+2i      95-180i
C      A = 1+2i   6   -2+6i    b = 545-118i
C      1+2i -2+6i   11      865+ 62i
C
C      DATA A / (1.0,0.0), (1.0,-2.0), (6.0,0.0),
$          (1.0,-2.0), (6.0,-2.0), (11.0,0.0) /
C      DATA B / (95.0,-180.0), (545.0,-118.0), (865.0,62.0) /
C
C      PRINT 1000
C      PRINT 1010, A(1), A(2), A(4)
C      PRINT 1010, CONJG(A(2)), A(3), A(5)
C      PRINT 1010, CONJG(A(4)), CONJG(A(5)), A(6)
C      PRINT 1020
C      PRINT 1030, B
C      CALL CHPCO (A, N, IPIVOT, RCOND, WORK)
C      PRINT 1040, RCOND
C      IF ((RCOND + 1.0) .EQ. 1.0) THEN
C          PRINT 1050
C      END IF
C      CALL CHPSL (A, N, IPIVOT, B)
C      PRINT 1060
C      PRINT 1030, B
C
C      1000 FORMAT (1X, 'A in full form:')
C      1010 FORMAT (3(3X, '(', F4.1, ', ', F4.1, ')'))
C      1020 FORMAT (/1X, 'b:')
C      1030 FORMAT (3X, '(', F6.1, ', ', F6.1, ')')
C      1040 FORMAT (/1X, 'Reciprocal condition number of A:', F6.3)
C      1050 FORMAT (1X, 'A may be singular to working precision.')
C      1060 FORMAT (/1X, 'A**(-1) * b:')
C
C      END

```

Sample Output

```

A in full form:
( 1.0, 0.0) ( 1.0,-2.0) ( 1.0,-2.0)
( 1.0, 2.0) ( 6.0, 0.0) ( 6.0,-2.0)
( 1.0, 2.0) ( 6.0, 2.0) (11.0, 0.0)

b:
( 95.0,-180.0)
( 545.0,-118.0)
( 865.0, 62.0)

```

```
Reciprocal condition number of A: 0.001
```

```
A**(-1) * b:  
( 5.0, 0.0)  
( 26.0, 0.0)  
( 64.0, 0.0)
```

Determinant, Inertia, and Inverse of a UDU-Factored Hermitian Matrix in Packed Storage

The subroutines described in this section compute the determinant, inertia, and inverse of a Hermitian matrix A in packed storage, which has been UDU-factored by xHPCO or xHPFA.

Calling Sequence

```
CALL ZHPDI  (ZA, N, IPIVOT, DDET, INERT, ZWORK, JOB)
CALL CHPDI  (CA, N, IPIVOT, SDET, INERT, CWORK, JOB)

void zhpdi  (doublecomplex *za, int n, int *ipivot, double *ddet,
             int *inert, int job)
void chpdi  (complex *ca, int n, int *ipivot, float *sdet, int
             *inert, int job)
```

Arguments

xA	On entry, the UDU factorization of the matrix, as computed by xHPCO or xHPFA. On exit, if the c digit of JOB = 0, then A contains the upper triangle of the inverse of the original matrix A; otherwise unchanged.
N	Order of the original matrix A. $N \geq 0$.
IPIVOT	Pivot vector as computed by xHPCO or xHPFA.
xDET	On exit, if the b digit of JOB $\neq 0$, then DET contains the determinant of the matrix A. The determinant is stored as $b \times 10^{\text{expon}}$ where b is stored in DET(1) and expon is stored in DET(2). $1.0 \leq \text{DET}(1) < 10.0$ or $\text{DET}(1) = 0.0$. If the b digit of JOB = 0, DET is not referenced.
INERT	On exit, if the a digit of JOB $\neq 0$, then INERT contains an integer triplet where INERT(1) = number of positive eigenvalues INERT(2) = number of negative eigenvalues INERT(3) = number of zero eigenvalues If the a digit of JOB = 0 then INERT is not referenced.
xWORK	Scratch array with a dimension of N.
JOB	Integer in the form abc; determines operation subroutine will perform: a $\neq 0$ Compute the inertia. b $\neq 0$ Compute the determinant. c $\neq 0$ Compute the inverse.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER    IDODET, IDOINR, IDOINV, LENGTA, N
  PARAMETER (IDODET = 10)
  PARAMETER (IDOINR = 100)
  PARAMETER (IDOINV = 1)
  PARAMETER (N = 3)
  PARAMETER (LENGTA = (N * N + N) / 2)

C
  REAL       DET(2), RCOND
  COMPLEX    A(LENGTA), WORK(N)
  INTEGER    INERT(3), IPIVOT(N), JOB

C
  EXTERNAL   CHPCO, CHPDI

C
  Initialize the array A to store the matrix A shown below.
C
C
C      1      1+2i  1+2i
C  A = 1+2i    6    -2+6i
C      1+2i -2+6i   11
C
  DATA A / (1.0,0.0), (1.0,-2.0), (6.0,0.0),
$          (1.0,-2.0), (6.0,-2.0), (11.0,0.0) /

C
  PRINT 1000
  PRINT 1010, A(1), A(2), A(4)
  PRINT 1010, CONJG(A(2)), A(3), A(5)
  PRINT 1010, CONJG(A(4)), CONJG(A(5)), A(6)
  CALL CHPCO (A, N, IPIVOT, RCOND, WORK)
  PRINT 1020, RCOND
  IF ((RCOND + 1.0) .EQ. 1.0) THEN
    PRINT 1030
  END IF
  JOB = IDOINR + IDODET + IDOINV
  CALL CHPDI (A, N, IPIVOT, DET, INERT, WORK, JOB)
  PRINT 1040, DET(1) * (10.0D0 ** DET(2))
  PRINT 1050, INERT
  PRINT 1060
  PRINT 1010, A(1), A(2), A(4)
  PRINT 1010, CONJG(A(2)), A(3), A(5)
  PRINT 1010, CONJG(A(4)), CONJG(A(5)), A(6)

C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(: 3X, '(', F5.1, ', ', F5.1, ')'))
1020 FORMAT (/1X, 'Reciprocal condition number of A:', F6.3)
1030 FORMAT (1X, 'A may be singular to working precision.')
1040 FORMAT (/1X, 'Determinant of A: ', F6.3)
1050 FORMAT (1X, 'Inertia of A: <', I1, ', ', I1, ', ', I1, '>')
1060 FORMAT (/1X, 'A**(-1):')

```



```
C
      END
```

Sample Output

```
A in full form:
(  1.0,  0.0)  (  1.0, -2.0)  (  1.0, -2.0)
(  1.0,  2.0)  (  6.0,  0.0)  (  6.0, -2.0)
(  1.0,  2.0)  (  6.0,  2.0)  ( 11.0,  0.0)

Reciprocal condition number of A: 0.001

Determinant of A: 1.000
Inertia of A: <3,0,0>

A**(-1):
( 26.0,  0.0)  ( -1.0, 12.0)  ( -4.0, -2.0)
( -1.0, -12.0) (  6.0,  0.0)  ( -1.0,  2.0)
( -4.0,  2.0)  ( -1.0, -2.0)  (  1.0,  0.0)
```

UDU Factorization of a Hermitian Matrix in Packed Storage

The subroutines described in this section compute the UDU factorization of a Hermitian matrix A in packed storage. It is typical to follow a call to xHPFA with a call to xHPSL to solve $Ax = b$ or to xHPDI to compute the determinant, inverse, and inertia of A.

Calling Sequence

```
CALL ZHPFA  (ZA, N, IPIVOT, INFO)
CALL CHPFA  (CA, N, IPIVOT, INFO)
```

```
void zhpfa  (doublecomplex *za, int n, int *ipivot, int *info)
void chpfa  (complex *ca, int n, int *ipivot, int *info)
```

Arguments

xA	On entry, the upper triangle of the matrix A. On exit, a UDU factorization of the matrix A.
N	Order of the matrix A. $N \geq 0$.
IPIVOT	On exit, a vector of pivot indices.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO > 0 Returns a value k if the kth pivot block is singular to indicate that xHPSL or xHPDI will divide by zero if called.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER    LENGTA, N
  PARAMETER (N = 3)
  PARAMETER (LENGTA = (N * N + N) / 2)

C
  COMPLEX    A(LENGTA), B(N)
  INTEGER    INFO, IPIVOT(N)

C
  EXTERNAL   CHPFA, CHPSL
  INTRINSIC  CONJG

C
C  Initialize the array A to store the matrix A shown below.
```

```

C      Initialize the array B to store the vector b shown below.
C
C      1      1+2i  1+2i      95-180i
C      A = 1+2i   6    -2+6i    b = 545-118i
C      1+2i -2+6i   11      865+ 62i
C
C      DATA A / (1.0,0.0), (1.0,-2.0), (6.0,0.0),
$      (1.0,-2.0), (6.0,-2.0), (11.0,0.0) /
C      DATA B / (95.0,-180.0), (545.0,-118.0), (865.0,62.0) /
C
C      PRINT 1000
C      PRINT 1010, A(1), A(2), A(4)
C      PRINT 1010, CONJG(A(2)), A(3), A(5)
C      PRINT 1010, CONJG(A(4)), CONJG(A(5)), A(6)
C      PRINT 1020
C      PRINT 1030, B
C      CALL CHPFA (A, N, IPIVOT, INFO)
C      IF (INFO .EQ. 0) THEN
C          CALL CHPSL (A, N, IPIVOT, B)
C          PRINT 1040
C          PRINT 1030, B
C      ELSE
C          PRINT 1050
C      END IF
C
C      1000 FORMAT (1X, 'A in full form:')
C      1010 FORMAT (4(: 3X, '(', F4.1, ', ', F4.1, ')'))
C      1020 FORMAT (/1X, 'b:')
C      1030 FORMAT (3X, '(', F6.1, ', ', F6.1, ')')
C      1040 FORMAT (/1X, 'A**(-1) * b:')
C      1050 FORMAT (1X, 'A is singular to working precision.')
C
C      END

```

Sample Output

```

A in full form:
( 1.0, 0.0) ( 1.0,-2.0) ( 1.0,-2.0)
( 1.0, 2.0) ( 6.0, 0.0) ( 6.0,-2.0)
( 1.0, 2.0) ( 6.0, 2.0) (11.0, 0.0)

b:
( 95.0,-180.0)
( 545.0,-118.0)
( 865.0, 62.0)

A**(-1) * b:
( 5.0, 0.0)
( 26.0, 0.0)
( 64.0, 0.0)

```

Solution to a Linear System in a UDU-Factored Hermitian Matrix in Packed Storage

The subroutines described in this section solve the linear system $Ax = b$ for a Hermitian matrix A in packed storage, which has been UDU-factored by xHPCO or xHPFA, and vectors b and x .

Calling Sequence

```
CALL ZHPSL  (ZA, N, IPIVOT, ZB)
CALL CHPSL  (CA, N, IPIVOT, CB)

void zhpsl  (doublecomplex *za, int n, int *ipivot, doublecomplex
            *zb)
void chpsl  (complex *ca, int n, int *ipivot, complex *cb)
```

Arguments

xA	On entry, the UDU factorization of the matrix A , as computed xHPCO or xHPFA.
N	Order of the matrix A . $N \geq 0$.
IPIVOT	Pivot vector as computed by xHPCO or xHPFA.
xB	On entry, the right-hand side vector b . On exit, the solution vector x .

Sample Program

```
PROGRAM TEST
IMPLICIT NONE

C
C   INTEGER    LENGTA, N
C   PARAMETER (N = 3)
C   PARAMETER (LENGTA = (N * N + N) / 2)

C
C   REAL       RCOND
C   COMPLEX    A(LENGTA), B(N), WORK(N)
C   INTEGER    IPIVOT(N)

C
C   EXTERNAL   CHPCO, CHPSL
C   INTRINSIC  CONJG

C
C   Initialize the array A to store the matrix A shown below.
C   Initialize the array B to store the vector b shown below.
```

```

C
C
C      1      1+2i  1+2i      95-180i
C      A = 1+2i   6   -2+6i    b = 545-118i
C      1+2i  -2+6i   11      865+ 62i
C
C
C      DATA A / (1.0,0.0), (1.0,-2.0), (6.0,0.0),
$      (1.0,-2.0), (6.0,-2.0), (11.0,0.0) /
C      DATA B / (95.0,-180.0), (545.0,-118.0), (865.0,62.0) /
C
C      PRINT 1000
C      PRINT 1010, A(1), A(2), A(4)
C      PRINT 1010, CONJG(A(2)), A(3), A(5)
C      PRINT 1010, CONJG(A(4)), CONJG(A(5)), A(6)
C      PRINT 1020
C      PRINT 1030, B
C      CALL CHPCO (A, N, IPIVOT, RCOND, WORK)
C      PRINT 1040, RCOND
C      IF ((RCOND + 1.0) .EQ. 1.0) THEN
C        PRINT 1050
C      END IF
C      CALL CHPSL (A, N, IPIVOT, B)
C      PRINT 1060
C      PRINT 1030, B
C
C      1000 FORMAT (1X, 'A in full form:')
C      1010 FORMAT (3(3X, '(', F4.1, ',', F4.1, ')'))
C      1020 FORMAT (/1X, 'b:')
C      1030 FORMAT (3X, '(', F6.1, ',', F6.1, ')')
C      1040 FORMAT (/1X, 'Reciprocal condition number of A:', F6.3)
C      1050 FORMAT (1X, 'A may be singular to working precision.')
C      1060 FORMAT (/1X, 'A**(-1) * b:')
C
C      END

```

Sample Output

```

A in full form:
( 1.0, 0.0) ( 1.0,-2.0) ( 1.0,-2.0)
( 1.0, 2.0) ( 6.0, 0.0) ( 6.0,-2.0)
( 1.0, 2.0) ( 6.0, 2.0) (11.0, 0.0)

b:
( 95.0,-180.0)
( 545.0,-118.0)
( 865.0, 62.0)

Reciprocal condition number of A: 0.001

```

```
A**(-1) * b:  
( 5.0, 0.0)  
( 26.0, 0.0)  
( 64.0, 0.0)
```

Cholesky Factorization and Condition Number of a Symmetric Positive Definite Matrix in Banded Storage

The subroutines described in this section compute a Cholesky factorization and condition number of a symmetric positive definite matrix A in banded storage. If the condition number is not needed then xPBFA is slightly faster. It is typical to follow a call to xPBCO with a call to xPBSL to solve $Ax = b$ or to xPBDI to compute the determinant of A.

Calling Sequence

```
CALL DPBCO (DA, LDA, N, NDIAG, DRCOND, DWORK, INFO)
CALL SPBCO (SA, LDA, N, NDIAG, SRCOND, SWORK, INFO)
CALL ZPBCO (ZA, LDA, N, NDIAG, DRCOND, ZWORK, INFO)
CALL CPBCO (CA, LDA, N, NDIAG, SRCOND, CWORK, INFO)

void dpbco (double *da, int lda, int n, int ndiag, double *drcond,
            int *info)
void spbco (float *sa, int lda, int n, int ndiag, float *srcond,
            int *info)
void zpbco (doublecomplex *za, int lda, int n, int ndiag, double
            *drcond, int *info)
void cpbco (complex *ca, int lda, int n, int ndiag, float *srcond,
            int *info)
```

Arguments

xA	On entry, the upper triangle of the matrix A. On exit, a Cholesky factorization of the matrix A.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq NDIAG + 1$.
N	Order of the matrix A. $N \geq 0$.
NDIAG	Number of diagonals. $N-1 \geq NDIAG \geq 0$ but if $N = 0$ then $NDIAG = 0$.
xRCOND	On exit, an estimate of the reciprocal condition number of A. $0.0 \leq RCOND \leq 1.0$. As the value of RCOND gets smaller, operations with A such as solving $Ax = b$ may become less stable. If RCOND satisfies $RCOND + 1.0 = 1.0$ then A may be singular to working precision.
xWORK	Scratch array with a dimension of N.

INFO On exit:
INFO = 0 Subroutine completed normally.
INFO > 0 Returns a value k if the leading minor of order k is not positive definite.

Sample Program

```

C      PROGRAM TEST
C      IMPLICIT NONE
C
C      INTEGER          LDA, N, NDIAG
C      PARAMETER        (N = 4)
C      PARAMETER        (NDIAG = 1)
C      PARAMETER        (LDA = NDIAG + 1)
C
C      DOUBLE PRECISION A(LDA,N), B(N), RCOND, WORK(N)
C      INTEGER          ICOL, INFO, IROW
C
C      EXTERNAL          DPBCO, DPBSL
C
C      Initialize the array A to store in banded storage mode
C      the matrix A shown below.  Initialize the array B to
C      store the vector B shown below.
C
C      A =
C      2   -1   0   0   60
C      -1   2   -1   0   b = 60
C      0   -1   2   -1   60
C      0   0   -1   2   60
C
C      DATA A / 8D8, 2.0D0, -1.0D0, 2.0D0, -1.0D0, 2.0D0, -1.0D0,
$      2.0D0 /
C      DATA B / 4*6.0D1 /
C
C      PRINT 1000
C      PRINT 1010, A(2,1), A(1,2)
C      PRINT 1010, A(3,1), A(2,2), A(1,3)
C      PRINT 1020,          A(3,2), A(2,3), A(1,4)
C      PRINT 1030,          A(3,3), A(2,4)
C      PRINT 1040
C      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, LDA)
C      PRINT 1050
C      PRINT 1060, B
C      CALL DPBCO (A, LDA, N, NDIAG, RCOND, WORK, INFO)
C      IF (INFO .EQ. 0) THEN
C        IF ((RCOND + 1.0D0) .EQ. 1.0D0) THEN
C          PRINT 1100
C        END IF
C        CALL DPBSL (A, LDA, N, NDIAG, B)
C        PRINT 1070, RCOND
C        PRINT 1080
C        PRINT 1060, B

```



```
        ELSE
          PRINT 1090
        END IF
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F5.1))
1020 FORMAT (8X, 3(3X, F5.1))
1030 FORMAT (16X, 3(3X, F5.1))
1040 FORMAT (/1X, 'A in banded form: (* in unused entries)')
1050 FORMAT (/1X, 'b:')
1060 FORMAT (3X, F5.1)
1070 FORMAT (/1X, 'Reciprocal condition number of A:', F5.1)
1080 FORMAT (/1X, 'A**(-1) * b:')
1090 FORMAT (/1X, 'A is not positive definite.')
1100 FORMAT (1X, 'A may be singular to working precision.')
C
      END
```

Sample Output

```
A in full form:
  2.0   -1.0
 -1.0    2.0   -1.0
          -1.0    2.0   -1.0
                        2.0
A in banded form: (* in unused entries)
*****  -1.0   -1.0   -1.0
         2.0    2.0    2.0    2.0
b:
  60.0
  60.0
  60.0
  60.0
Reciprocal condition number of A:  0.1
A**(-1) * b:
 120.0
 180.0
 180.0
 120.0
```

Determinant of a Cholesky-Factored Symmetric Positive Definite Matrix in Banded Storage

The subroutines described in this section compute the determinant of a symmetric positive definite matrix A in banded storage, which has been Cholesky-factored by xPBCO or xPBFA.

Calling Sequence

```
CALL DPBDI (DA, LDA, N, NDIAG, DDET)
CALL SPBDI (SA, LDA, N, NDIAG, SDET)
CALL ZPBDI (ZA, LDA, N, NDIAG, DDET)
CALL CPBDI (CA, LDA, N, NDIAG, SDET)

void dpbdi (double *da, int lda, int n, int ndiag, double *ddet)
void spbdi (float *sa, int lda, int n, int ndiag, float *sdet)
void zpbdi (doublecomplex *za, int lda, int n, int ndiag, double
            *ddet)
void cpbdi (complex *ca, int lda, int n, int ndiag, float *sdet)
```

Arguments

xA	Cholesky factorization of the matrix A, as computed by xPBCO or xPBFA.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq NDIAG + 1$.
N	Order of the original matrix A. $N \geq 0$.
NDIAG	Number of diagonals. $N-1 \geq NDIAG \geq 0$ but if $N = 0$ then $NDIAG = 0$.
xDET	On exit, the determinant of the matrix A. The determinant is stored as $b \times 10^{\text{expon}}$ where b is stored in DET(1) and expon is stored in DET(2). $1.0 \leq DET(1) < 10.0$ or $DET(1) = 0.0$.

Sample Program

```

C      PROGRAM TEST
C      IMPLICIT NONE
C
C      INTEGER          LDA, N, NDIAG
C      PARAMETER        (N = 4)
C      PARAMETER        (NDIAG = 1)
```

```

C      PARAMETER      (LDA = NDIAG + 1)
C
C      DOUBLE PRECISION A(LDA,N), DET(2)
C      INTEGER          ICOL, INFO, IROW, NDIAG
C      EXTERNAL          DPBDI, DPBFA
C
C      Initialize the array A to store in banded storage mode
C      the matrix A shown below.
C
C      A =
C      2  -1  0  0
C      -1  2  -1  0
C      0  -1  2  -1
C      0  0  -1  2
C
C      DATA A / 8D8, 2.0D0, -1.0D0, 2.0D0, -1.0D0, 2.0D0, -1.0D0,
C      $          2.0D0 /
C
C      PRINT 1000
C      PRINT 1010, A(2,1), A(1,2)
C      PRINT 1010, A(3,1), A(2,2), A(1,3)
C      PRINT 1020,          A(3,2), A(2,3), A(1,4)
C      PRINT 1030,          A(3,3), A(2,4)
C      PRINT 1040
C      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, LDA)
C      CALL DPBFA (A, LDA, N, NDIAG, INFO)
C      IF (INFO .EQ. 0) THEN
C          CALL DPBDI (A, LDA, N, NDIAG, DET)
C          PRINT 1050, DET(1) * (10.0D0 ** DET(2))
C      ELSE
C          PRINT 1060
C      END IF
C
C      1000 FORMAT (1X, 'A in full form:')
C      1010 FORMAT (4(3X, F5.1))
C      1020 FORMAT (8X, 3(3X, F5.1))
C      1030 FORMAT (16X, 3(3X, F5.1))
C      1040 FORMAT (/1X, 'A in banded form: (* in unused entries)')
C      1050 FORMAT (/1X, 'The determinant of A is ', F5.1)
C      1060 FORMAT (/1X, 'A is not positive definite.')
C
C      END

```

Sample Output

```

A in full form:
  2.0  -1.0
 -1.0   2.0  -1.0
        -1.0   2.0  -1.0
                -1.0   2.0

```

```
A in banded form: (* in unused entries)
*****  -1.0  -1.0  -1.0
      2.0   2.0   2.0   2.0

The determinant of A is  5.0
```

Cholesky Factorization of a Symmetric Positive Definite Matrix in Banded Storage

The subroutines described in this section compute a Cholesky factorization of a symmetric positive definite matrix A in banded storage. It is typical to follow a call to xPBFA with a call to xPBSL to solve $Ax = b$ or to xPBDI to compute the determinant of A .

Calling Sequence

```
CALL DPBFA  (DA, LDA, N, NDIAG, INFO)
CALL SPBFA  (SA, LDA, N, NDIAG, INFO)
CALL ZPBFA  (ZA, LDA, N, NDIAG, INFO)
CALL CPBFA  (CA, LDA, N, NDIAG, INFO)
```

```
void dpbfa  (double *da, int lda, int n, int ndiag, int *info)
void spbfa  (float *sa, int lda, int n, int ndiag, int *info)
void zpbfa  (doublecomplex *za, int lda, int n, int ndiag, int
             *info)
void cpbfa  (complex *ca, int lda, int n, int ndiag, int *info)
```

Arguments

xA	On entry, the upper triangle of the matrix A . On exit, a Cholesky factorization of the matrix A .
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq NDIAG + 1$.
N	Order of the matrix A . $N \geq 0$.
NDIAG	Number of diagonals. $N-1 \geq NDIAG \geq 0$ but if $N = 0$ then $NDIAG = 0$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO > 0 Returns a value of k if the leading minor of order k is not positive definite.

Sample Program

```
C      PROGRAM TEST
C      IMPLICIT NONE
C
C      INTEGER          LDA, N, NDIAG
C      PARAMETER        (N = 4)
```

```

PARAMETER          (NDIAG = 1)
PARAMETER          (LDA = NDIAG + 1)
C
DOUBLE PRECISION   A(LDA,N), B(N)
INTEGER            ICOL, INFO, IROW
C
EXTERNAL           DPBFA, DPBSL
C
C   Initialize the array A to store in banded storage mode
C   the matrix A shown below. Initialize the array B to
C   store the vector B shown below.
C
C       2   -1   0   0           60
C   A = -1   2   -1   0           b = 60
C       0   -1   2   -1          60
C       0   0   -1   2           60
C
DATA A / 8D8, 2.0D0, -1.0D0, 2.0D0, -1.0D0, 2.0D0, -1.0D0,
$      2.0D0 /
DATA B / 4*6.0D1 /
C
PRINT 1000
PRINT 1010, A(2,1), A(1,2)
PRINT 1010, A(3,1), A(2,2), A(1,3)
PRINT 1020,           A(3,2), A(2,3), A(1,4)
PRINT 1030,           A(3,3), A(2,4)
PRINT 1040
PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, LDA)
PRINT 1050
PRINT 1060, B
CALL DPBFA (A, LDA, N, NDIAG, INFO)
IF (INFO .EQ. 0) THEN
    CALL DPBSL (A, LDA, N, NDIAG, B)
    PRINT 1070
    PRINT 1060, B
ELSE
    PRINT 1080
END IF
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F5.1))
1020 FORMAT (8X, 3(3X, F5.1))
1030 FORMAT (16X, 3(3X, F5.1))
1040 FORMAT (/1X, 'A in banded form: (* in unused entries)')
1050 FORMAT (/1X, 'b:')
1060 FORMAT (3X, F5.1)
1070 FORMAT (/1X, 'A**(-1) * b:')
1080 FORMAT (/1X, 'A is not positive definite.')
C
END

```

Sample Output

```
A in full form:
  2.0   -1.0
 -1.0    2.0   -1.0   -1.0
           -1.0    2.0    2.0

A in banded form: (* in unused entries)
*****
  2.0   2.0   2.0   2.0

b:
 60.0
 60.0
 60.0
 60.0

A**(-1) * b:
120.0
180.0
180.0
120.0
```

Solution to a Linear System in a Cholesky-Factored Symmetric Positive Definite Matrix in Banded Storage

The subroutines described in this section solve the linear system $Ax = b$ for a symmetric positive definite matrix A in banded storage, which has been Cholesky-factored by xPBCO or xPBFA, and vectors b and x .

Calling Sequence

```
CALL DPBSL (DA, LDA, N, NDIAG, DB)
CALL SPBSL (SA, LDA, N, NDIAG, SB)
CALL ZPBSL (ZA, LDA, N, NDIAG, ZB)
CALL CPBSL (CA, LDA, N, NDIAG, CB)

void dpbsl (double *da, int lda, int n, int ndiag, double *db)
void spbsl (float *sa, int lda, int n, int ndiag, float *sb)
void zpbsl (doublecomplex *za, int lda, int n, int ndiag,
doublecomplex *zb)
void cpbsl (complex *ca, int lda, int n, int ndiag, complex *cb)
```

Arguments

xA	Cholesky factorization of the matrix A , as computed by xPBCO or xPBFA.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq NDIAG + 1$.
N	Order of the matrix A . $N \geq 0$.
NDIAG	Number of diagonals of matrix A . $N-1 \geq NDIAG \geq 0$ but if $N = 0$ then $NDIAG = 0$.
xB	On entry, the right-hand side vector b . On exit, the solution vector x .

Sample Program

```
PROGRAM TEST
IMPLICIT NONE

C
INTEGER          LDA, N, NDIAG
PARAMETER        (N = 4)
PARAMETER        (NDIAG = 1)
PARAMETER        (LDA = NDIAG + 1)
```



```

C      DOUBLE PRECISION  A(LDA,N), B(N)
C      INTEGER          ICOL, INFO, IROW
C
C      EXTERNAL          DPBFA, DPBSL
C
C      Initialize the array A to store in banded storage mode
C      the matrix A shown below. Initialize the array B to
C      store the vector B shown below.
C
C      A =   2   -1   0   0      b =   60
C      -1   2   -1   0      120
C      0   -1   2   -1      120
C      0    0   -1   2      60
C
C      DATA A / 8D8, 2.0D0, -1.0D0, 2.0D0, -1.0D0, 2.0D0, -1.0D0,
$          2.0D0 /
C      DATA B / 6.0D1, 1.2D1, 1.2D1, 6.0D1 /
C
C      PRINT 1000
C      PRINT 1010, A(2,1), A(1,2)
C      PRINT 1010, A(3,1), A(2,2), A(1,3)
C      PRINT 1020,          A(3,2), A(2,3), A(1,4)
C      PRINT 1030,          A(3,3), A(2,4)
C      PRINT 1040
C      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, LDA)
C      PRINT 1050
C      PRINT 1060, B
C      CALL DPBFA (A, LDA, N, NDIAG, INFO)
C      IF (INFO .EQ. 0) THEN
C          CALL DPBSL (A, LDA, N, NDIAG, B)
C          PRINT 1070
C          PRINT 1060, B
C      ELSE
C          PRINT 1080
C      END IF
C
C      1000 FORMAT (1X, 'A in full form:')
C      1010 FORMAT (4(3X, F5.1))
C      1020 FORMAT (8X, 3(3X, F5.1))
C      1030 FORMAT (16X, 3(3X, F5.1))
C      1040 FORMAT (/1X, 'A in banded form: (* in unused entries)')
C      1050 FORMAT (/1X, 'b:')
C      1060 FORMAT (3X, F5.1)
C      1070 FORMAT (/1X, 'A**(-1) * b:')
C      1080 FORMAT (/1X, 'A is not positive definite.')
C
C      END

```

Sample Output

```
A in full form:
  2.0  -1.0
 -1.0   2.0  -1.0  -1.0
          -1.0   2.0   2.0
          -1.0   2.0

A in banded form: (* in unused entries)
*****  -1.0  -1.0  -1.0
        2.0   2.0   2.0   2.0

b:
  60.0
  12.0
  12.0
  60.0

A**(-1) * b:
  72.0
  84.0
  84.0
  72.0
```

Cholesky Factorization and Condition Number of a Symmetric Positive Definite Matrix

The subroutines described in this section compute a Cholesky factorization and condition number of a symmetric positive definite matrix A. If the condition number is not needed then xPOFA is slightly faster. It is typical to follow a call to xPOCO with a call to xPOSL to solve $Ax = b$ or to xPODI to compute the determinant and inverse of A.

Calling Sequence

```
CALL DPOCO (DA, LDA, N, DRCOND, DWORK, INFO)
CALL SPOCO (SA, LDA, N, SRCOND, SWORK, INFO)
CALL ZPOCO (ZA, LDA, N, DRCOND, ZWORK, INFO)
CALL CPOCO (CA, LDA, N, SRCOND, CWORK, INFO)

void dpoco (double *da, int lda, int n, double *drcond, int *info)
void spoco (float *sa, int lda, int n, float *srcond, int *info)
void zpoco (doublecomplex *za, int lda, int n, double *drcond, int
            *info)
void cpoco (complex *ca, int lda, int n, float *srcond, int *info)
```

Arguments

xA	On entry, the upper triangle of the matrix A. On exit, a Cholesky factorization of the matrix A. The strict lower triangle of A is not referenced.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
N	Order of the matrix A. $N \geq 0$.
xRCOND	On exit, an estimate of the reciprocal condition number of A. $0.0 \leq RCOND \leq 1.0$. As the value of RCOND gets smaller, operations with A such as solving $Ax = b$ may become less stable. If RCOND satisfies $RCOND + 1.0 = 1.0$ then A may be singular to working precision.
xWORK	Scratch array with a dimension of N.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO > 0 Returns a value k if the leading minor of order k is not positive definite.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
INTEGER          LDA, N
PARAMETER        (N = 4)
PARAMETER        (LDA = N)

C
DOUBLE PRECISION A(LDA,N), B(N), RCOND, WORK(N)
INTEGER          ICOL, INFO, IROW

C
EXTERNAL          DPOCO, DPOSL

C
C Initialize the array A to store in symmetric storage mode
C the matrix A shown below. Initialize the array B to store
C the vector B shown below.
C
C      2   -1   0   0           40
C  A = -1   2   -1   0           b = 30
C      0   -1   2   -1           20
C      0   0   -1   2           10
C
DATA A / 2.0D0, 3*8D8, -1.0D0, 2.0D0, 2*8D8, 0.0D0, -1.0D0,
$      2.0D0, -1.0D0, 0.0D0, 0.0D0, -1.0D0, 2.0D0 /
DATA B / 4.0D0, 3.0D0, 2.0D0, 1.0D0 /

C
PRINT 1000
DO 100, IROW = 1, N
    PRINT 1010, (A(ICOL,IROW), ICOL = 1, IROW),
$      (A(IROW,ICOL), ICOL = IROW + 1, N)
100 CONTINUE
PRINT 1020
PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
PRINT 1030
PRINT 1040, B
CALL DPOCO (A, LDA, N, RCOND, WORK, INFO)
IF (INFO .EQ. 0) THEN
    IF ((RCOND + 1.0D0) .EQ. 1.0D0) THEN
        PRINT 1070
    END IF
    CALL DPOSL (A, LDA, N, B)
    PRINT 1050
    PRINT 1040, B
    PRINT 1060, RCOND
ELSE
    PRINT 1080
END IF

C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F7.3))
1020 FORMAT (/1X, 'A in symmetric form: (* in unused entries)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (3X, F7.3)

```

```
1050 FORMAT (/1X, 'A**(-1) * b:')
1060 FORMAT (/1X, 'Reciprocal condition number of A:', F5.1)
1070 FORMAT (1X, 'A may be singular to working precision.')
1080 FORMAT (1X, 'A is not positive definite.')
C
      END
```

Sample Output

```
A in full form:
  2.000   -1.000    0.000    0.000
 -1.000    2.000   -1.000    0.000
  0.000   -1.000    2.000   -1.000
  0.000    0.000   -1.000    2.000

A in symmetric form: (* in unused entries)
  2.000   -1.000    0.000    0.000
*****    2.000   -1.000    0.000
*****    *****    2.000   -1.000
*****    *****   -1.000    2.000

b:
  4.000
  3.000
  2.000
  1.000

A**(-1) * b:
  6.000
  8.000
  7.000
  4.000

Reciprocal condition number of A:  0.1
```

Determinant and Inverse of a Cholesky-Factored Symmetric Positive Definite Matrix

The subroutines described in this section compute the determinant and inverse of a symmetric positive definite matrix A, which has been Cholesky-factored by xPOCO, xPOFA, or xQRDC.

Calling Sequence

```
CALL DPODI  (DA, LDA, N, DDET, JOB)
CALL SPODI  (SA, LDA, N, SDET, JOB)
CALL ZPODI  (ZA, LDA, N, DDET, JOB)
CALL CPODI  (CA, LDA, N, SDET, JOB)

void dpodi  (double *da, int lda, int n, double *ddet, int job)
void spodi  (float *sa, int lda, int n, float *sdet, int job)
void zpodi  (doublecomplex *za, int lda, int n, double *ddet, int
job)
void cpodi  (complex *ca, int lda, int n, float *sdet, int job)
```

Arguments

xA	On entry, the Cholesky factorization of the matrix A, as computed by xPOCO, xPOFA, or xQRDC. On exit, the inverse of the original matrix A if the inverse was requested, otherwise A is unchanged. The strict lower triangle of A is not referenced.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
N	Order of the original matrix A. $N \geq 0$.
xDET	On exit, the determinant of the matrix A. The determinant is stored as $b \times 10^{\text{expon}}$ where b is stored in DET(1) and expon is stored in DET(2). $1.0 \leq \text{DET}(1) < 10.0$ or $\text{DET}(1) = 0.0$.
JOB	Determines which operation the subroutine will perform: 11 both determinant and inverse 01 inverse only 10 determinant only

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
INTEGER          INVDET, LDA, N
PARAMETER        (INVDET = 11)
PARAMETER        (N = 4)
PARAMETER        (LDA = N)

C
DOUBLE PRECISION A(LDA,N), DET(2)
INTEGER          ICOL, INFO, IROW, JOB

C
EXTERNAL          DPODI, DPOFA

C
C Initialize the array A to store in symmetric storage mode
C the matrix A shown below.
C
C      4   3   2   1
C  A =  3   4   3   2
C      2   3   4   3
C      1   2   3   4
C
C      DATA A / 4.0D0, 3*8D8, 3.0D0, 4.0D0, 2*8D8, 2.0D0, 3.0D0,
$              4.0D0, 8D8, 1.0D0, 2.0D0, 3.0D0, 4.0D0 /

C
PRINT 1000
DO 100, IROW = 1, N
  PRINT 1010, (A(ICOL,IROW), ICOL = 1, IROW),
$            (A(IROW,ICOL), ICOL = IROW + 1, N)
100 CONTINUE
PRINT 1020
PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
CALL DPOFA (A, LDA, N, INFO)
IF (INFO.EQ. 0) THEN
  JOB = INVDET
  CALL DPODI (A, LDA, N, DET, JOB)
  PRINT 1050
  DO 110, IROW = 1, N
    PRINT 1010, (A(ICOL,IROW), ICOL = 1, IROW),
$              (A(IROW,ICOL), ICOL = IROW + 1, N)
110 CONTINUE
PRINT 1060, DET(1) * (10.0D0 ** DET(2))
ELSE
  PRINT 1070
END IF

C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F7.3))
1020 FORMAT (/1X, 'A in symmetric form: (* in unused entries)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (3X, F7.3)
1050 FORMAT (/1X, 'A**(-1):')
1060 FORMAT (/1X, 'The determinant of A is ', F5.1)

```

```
1070 FORMAT (/1X, 'A is not positive definite.')
C
      END
```

Sample Output

```
A in full form:
  4.000    3.000    2.000    1.000
  3.000    4.000    3.000    2.000
  2.000    3.000    4.000    3.000
  1.000    2.000    3.000    4.000

A in symmetric form: (* in unused entries)
  4.000    3.000    2.000    1.000
  *****    4.000    3.000    2.000
  *****    *****    4.000    3.000
  *****    *****    *****    4.000

A**(-1):
  0.600   -0.500    0.000    0.100
 -0.500    1.000   -0.500    0.000
  0.000   -0.500    1.000   -0.500
  0.100    0.000   -0.500    0.600

The determinant of A is  20.0
```


Cholesky Factorization of a Symmetric Positive Definite Matrix

The subroutines described in this section compute a Cholesky factorization of a symmetric positive definite matrix A . It is typical to follow a call to xPOFA with a call to xPOSL to solve $Ax = b$ or to xPODI to compute the determinant and inverse of A .

Calling Sequence

```
CALL DPOFA  (DA, LDA, N, INFO)
CALL SPOFA  (SA, LDA, N, INFO)
CALL ZPOFA  (ZA, LDA, N, INFO)
CALL CPOFA  (CA, LDA, N, INFO)

void dpofa  (double *da, int lda, int n, int *info)
void spofa  (float *sa, int lda, int n, int *info)
void zpofo  (doublecomplex *za, int lda, int n, int *info)
void cpofo  (complex *ca, int lda, int n, int *info)
```

Arguments

xA	On entry, the upper triangle of the matrix A . On exit, a Cholesky factorization of the matrix A . The strict lower triangle of A is not referenced.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
N	Order of the matrix A . $N \geq 0$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO > 0 Returns a value k if the leading minor of order k is not positive definite.

Sample Program

```
PROGRAM TEST
IMPLICIT NONE

C      INTEGER          LDA, N
C      PARAMETER        (N = 5)
C      PARAMETER        (LDA = N)
```

```

C      DOUBLE PRECISION A(LDA,N), B(N)
C      INTEGER          ICOL, INFO, IROW
C
C      EXTERNAL          DPOFA, DPOSL
C
C      Initialize the array A to store in symmetric storage mode
C      the matrix A shown below. Initialize the array B to store
C      the vector B shown below.
C
C      5  4  3  2  1      120
C      4  5  4  3  2      60
C      A = 3  4  5  4  3      b = 40
C      2  3  4  5  4      60
C      1  2  3  4  5      120
C
C      DATA A / 5.0D0, 4*8D8, 4.0D0, 5.0D0, 3*8D8, 3.0D0, 4.0D0,
$          5.0D0, 2*8D8, 2.0D0, 3.0D0, 4.0D0, 5.0D0, 1*8D8,
$          1.0D0, 2.0D0, 3.0D0, 4.0D0, 5.0D0 /
C      DATA B / 1.2D2, 6.0D1, 4.0D1, 6.0D1, 1.2D2 /
C
C      PRINT 1000
C      DO 100, IROW = 1, N
C          PRINT 1010, (A(ICOL,IROW), ICOL = 1, IROW),
$          (A(IROW,ICOL), ICOL = IROW + 1, N)
100 CONTINUE
C      PRINT 1020
C      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
C      PRINT 1030
C      PRINT 1040, B
C      CALL DPOFA (A, LDA, N, INFO)
C      IF (INFO .EQ. 0) THEN
C          CALL DPOSL (A, LDA, N, B)
C          PRINT 1050
C          PRINT 1040, B
C      ELSE
C          PRINT 1060
C      END IF
C
C      1000 FORMAT (1X, 'A in full form:')
C      1010 FORMAT (5(3X, F7.3))
C      1020 FORMAT (/1X, 'A in symmetric form: (* in unused entries)')
C      1030 FORMAT (/1X, 'b:')
C      1040 FORMAT (3X, F7.3)
C      1050 FORMAT (/1X, 'A*(-1) * b:')
C      1060 FORMAT (/1X, 'A is not positive definite.')
C
C      END

```

Sample Output

```
A in full form:
  5.000  4.000  3.000  2.000  1.000
  4.000  5.000  4.000  3.000  2.000
  3.000  4.000  5.000  4.000  3.000
  2.000  3.000  4.000  5.000  4.000
  1.000  2.000  3.000  4.000  5.000

A in symmetric form: (* in unused entries)
  5.000  4.000  3.000  2.000  1.000
  *****  5.000  4.000  3.000  2.000
  *****  *****  5.000  4.000  3.000
  *****  *****  *****  5.000  4.000
  *****  *****  *****  *****  5.000

b:
 120.000
  60.000
  40.000
  60.000
 120.000

A**(-1) * b:
  50.000
 -20.000
 -20.000
 -20.000
  50.000
```

Solution to a Linear System in a Cholesky-Factored Symmetric Positive Definite Matrix

The subroutines described in this section solve the linear system $Ax = b$ for a symmetric positive definite matrix A , which has been Cholesky-factored by xPOCO or xPOFA, and vectors b and x .

Calling Sequence

```
CALL DPOSL  (DA, LDA, N, DB)
CALL SPOSL  (SA, LDA, N, SB)
CALL ZPOSL  (ZA, LDA, N, ZB)
CALL CPOSL  (CA, LDA, N, CB)

void dposl  (double *da, int lda, int n, double *db)
void sposl  (float *sa, int lda, int n, float *sb)
void zposl  (doublecomplex *za, int lda, int n, doublecomplex *zb)
void cposl  (complex *ca, int lda, int n, complex *cb)
```

Arguments

xA	Cholesky factorization of the matrix A , as computed by xPOCO or xPOFA.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
N	Order of the matrix A . $N \geq 0$.
xB	On entry, the right-hand side vector b . On exit, the solution vector x .

Sample Program

```
PROGRAM TEST
IMPLICIT NONE
C
C   INTEGER          LDA, N
C   PARAMETER        (N = 4)
C   PARAMETER        (LDA = N)
C
C   DOUBLE PRECISION A(LDA,N), B(N), RCOND, WORK(N)
C   INTEGER          ICOL, INFO, IROW
C
C   EXTERNAL          DPOCO, DPOSL
```

```

C
C      Initialize the array A to store in symmetric storage mode
C      the matrix A shown below. Initialize the array B to store
C      the vector B shown below.
C
C      4  3  2  1      20
C      A = 3  4  3  2      b = 20
C          2  3  4  3      20
C          1  2  3  4      20
C
C      DATA A / 4.0D0, 3*8D8, 3.0D0, 4.0D0, 2*8D8, 2.0D0, 3.0D0,
$             4.0D0, 8D8, 1.0D0, 2.0D0, 3.0D0, 4.0D0 /
C      DATA B / 4*2.0D1 /
C
C      PRINT 1000
C      DO 100, IROW = 1, N
C          PRINT 1010, (A(ICOL,IROW), ICOL = 1, IROW),
$             (A(IROW,ICOL), ICOL = IROW + 1, N)
100 CONTINUE
C      PRINT 1020
C      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
C      PRINT 1030
C      PRINT 1040, B
C      CALL DPOCO (A, LDA, N, RCOND, WORK, INFO)
C      IF ((RCOND + 1.0D0) .EQ. 1.0D0) THEN
C          PRINT 1050
C      END IF
C      CALL DPOSL (A, LDA, N, B)
C      PRINT 1060
C      PRINT 1040, B
C
C      1000 FORMAT (1X, 'A in full form:')
C      1010 FORMAT (4(3X, F7.3))
C      1020 FORMAT (/1X, 'A in symmetric form: (* in unused entries)')
C      1030 FORMAT (/1X, 'b:')
C      1040 FORMAT (3X, F7.3)
C      1050 FORMAT (/1X, 'A may be singular to working precision.')
C      1060 FORMAT (/1X, 'A**(-1) * b:')
C
C      END

```

Sample Output

```

A in full form:
  4.000    3.000    2.000    1.000
  3.000    4.000    3.000    2.000
  2.000    3.000    4.000    3.000
  1.000    2.000    3.000    4.000

```

A in symmetric form: (* in unused entries)

4.000	3.000	2.000	1.000
*****	4.000	3.000	2.000
*****	*****	4.000	3.000
*****	*****	*****	4.000

b:

20.000
20.000
20.000
20.000

A**(-1) * b:

4.000
0.000
0.000
4.000

Cholesky Factorization and Condition Number of a Symmetric Positive Definite Matrix in Packed Storage

The subroutines described in this section compute a Cholesky factorization and condition number of a symmetric positive definite matrix A in packed storage. If the condition number is not needed then xPPFA is slightly faster. It is typical to follow a call to xPPCO with a call to xPPSL to solve $Ax = b$ or to xPPDI to compute the determinant and inverse of A .

Calling Sequence

```
CALL DPPCO (DA, N, DRCOND, DWORK, INFO)
CALL SPPCO (SA, N, SRCOND, SWORK, INFO)
CALL ZPPCO (ZA, N, DRCOND, ZWORK, INFO)
CALL CPPCO (CA, N, SRCOND, CWORK, INFO)

void dppco (double *da, int n, double *drcond, int *info)
void sppco (float *sa, int n, float *srcond, int *info)
void zppco (doublecomplex *za, int n, double *drcond, int *info)
void cppco (complex *ca, int n, float *srcond, int *info)
```

Arguments

xA	On entry, the upper triangle of the matrix A . On exit, a Cholesky factorization of the matrix A .
N	Order of the matrix A . $N \geq 0$.
xRCOND	On exit, an estimate of the reciprocal condition number of A . $0.0 \leq \text{RCOND} \leq 1.0$. As the value of RCOND gets smaller, operations with A such as solving $Ax = b$ may become less stable. If RCOND satisfies $\text{RCOND} + 1.0 = 1.0$ then A may be singular to working precision.
xWORK	Scratch array with a dimension of N .
INFO	On exit: INFO = 0 Subroutine completed normally. INFO > 0 Returns a value k if the leading minor of order k is not positive definite.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER          LENGTA, N
  PARAMETER        (N = 4)
  PARAMETER        (LENGTA = (N * N + N) / 2)
C
  DOUBLE PRECISION A(LENGTA), B(N), RCOND, WORK(N)
  INTEGER          INFO
C
  EXTERNAL          DPPCO, DPPSL
C
  C Initialize the array A to store in packed symmetric storage
  C mode the matrix A shown below. Initialize the array B to
  C store the matrix B shown below.
  C
  C
  C      4  3  2  1      60
  C  A = 3  4  3  2      b = 20
  C      2  3  4  3      20
  C      1  2  3  4      60
  C
  DATA A / 4.0D0, 3.0D0, 4.0D0, 2.0D0, 3.0D0, 4.0D0,
$          1.0D0, 2.0D0, 3.0D0, 4.0D0 /
  DATA B / 6.0D1, 2.0D1, 2.0D1, 6.0D1 /
C
  PRINT 1000
  PRINT 1010, A(1), A(2), A(4), A(7)
  PRINT 1010, A(2), A(3), A(5), A(8)
  PRINT 1010, A(4), A(5), A(6), A(9)
  PRINT 1010, A(7), A(8), A(9), A(10)
  PRINT 1020
  PRINT 1030, B
  CALL DPPCO (A, N, RCOND, WORK, INFO)
  IF ((RCOND + 1.0D0) .EQ. RCOND) THEN
    PRINT 1040
  END IF
  CALL DPPSL (A, N, B)
  PRINT 1050, RCOND
  PRINT 1060
  PRINT 1030, B
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (5(3X, F7.3))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (3X, F7.3)
1040 FORMAT (1X, 'A may be singular to working precision.')
1050 FORMAT (/1X, 'Reciprocal condition of A:', F5.2)
1060 FORMAT (/1X, 'A**(-1) * b:')
C
  END

```


Sample Output

```
A in full form:
  4.000    3.000    2.000    1.000
  3.000    4.000    3.000    2.000
  2.000    3.000    4.000    3.000
  1.000    2.000    3.000    4.000

b:
  60.000
  20.000
  20.000
  60.000

Reciprocal condition of A: 0.04

A**(-1) * b:
  32.000
 -20.000
 -20.000
  32.000
```

Determinant and Inverse of a Cholesky-Factored Symmetric Positive Definite Matrix in Packed Storage

The subroutines described in this section compute the determinant and inverse of a symmetric positive definite matrix A in packed storage, which has been Cholesky-factored by xPPCO or xPPFA.

Calling Sequence

```
CALL DPPDI (DA, N, DDET, JOB)
CALL SPPDI (SA, N, SDET, JOB)
CALL ZPPDI (ZA, N, DDET, JOB)
CALL CPPDI (CA, N, SDET, JOB)

void dppdi (double *da, int n, double *ddet, int job)
void sppdi (float *sa, int n, float *sdet, int job)
void zppdi (doublecomplex *za, int n, double *ddet, int job)
void cppdi (complex *ca, int n, float *sdet, int job)
```

Arguments

xA	On entry, the Cholesky factorization of the matrix A, as computed by xPPCO or xPPFA. On exit, the inverse of the original matrix A if the inverse was requested, otherwise A is unchanged. The strict lower triangle of A is not referenced.
N	Order of the original matrix A. $N \geq 0$.
xDET	On exit, the determinant of the matrix A. The determinant is stored as $b \times 10^{\text{expon}}$ where b is stored in DET(1) and expon is stored in DET(2). $1.0 \leq \text{DET}(1) < 10.0$ or $\text{DET}(1) = 0.0$.
JOB	Determines which operation the subroutine will perform: 11 both determinant and inverse 01 inverse only 10 determinant only

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
INTEGER          INVDET, LENGTA, N
PARAMETER        (INVDET = 11)
PARAMETER        (N = 4)
PARAMETER        (LENGTA = (N * N + N) / 2)

C
DOUBLE PRECISION A(LENGTA), DET(2)
INTEGER          INFO, JOB

C
EXTERNAL         DPPFA, DPPDI

C
C Initialize the array A to store in packed symmetric storage
C mode the matrix A shown below. Initialize the array B to
C store the vector B shown below.
C
C      4   3   2   1
C  A =  3   4   3   2
C      2   3   4   3
C      1   2   3   4
C
C      DATA A / 4.0D0, 3.0D0, 4.0D0, 2.0D0, 3.0D0, 4.0D0,
$      1.0D0, 2.0D0, 3.0D0, 4.0D0 /

C
PRINT 1000
PRINT 1010, A(1), A(2), A(4), A(7)
PRINT 1010, A(2), A(3), A(5), A(8)
PRINT 1010, A(4), A(5), A(6), A(9)
PRINT 1010, A(7), A(8), A(9), A(10)
CALL DPPFA (A, N, INFO)
IF (INFO .EQ. 0) THEN
    JOB = INVDET
    CALL DPPDI (A, N, DET, JOB)
    PRINT 1020, DET(1) * (1.0D1 ** DET(2))
    PRINT 1030
    PRINT 1010, A(1), A(2), A(4), A(7)
    PRINT 1010, A(2), A(3), A(5), A(8)
    PRINT 1010, A(4), A(5), A(6), A(9)
    PRINT 1010, A(7), A(8), A(9), A(10)
ELSE
    PRINT 1040
END IF

C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F5.1))
1020 FORMAT (/1X, 'Determinant of A is ', F4.1)
1030 FORMAT (/1X, 'A**(-1):')
1040 FORMAT (/1X, 'A is too ill-conditioned.')

C
END

```

Sample Output

```
A in full form:
  4.0    3.0    2.0    1.0
  3.0    4.0    3.0    2.0
  2.0    3.0    4.0    3.0
  1.0    2.0    3.0    4.0

Determinant of A is 20.0

A**(-1):
  0.6   -0.5    0.0    0.1
 -0.5    1.0   -0.5    0.0
  0.0   -0.5    1.0   -0.5
  0.1    0.0   -0.5    0.6
```

Cholesky Factorization of a Symmetric Positive Definite Matrix in Packed Storage

The subroutines described in this section compute a Cholesky factorization of a symmetric positive definite matrix A in packed storage. It is typical to follow a call to xPPFA with a call to xPPSL to solve $Ax = b$ or to xPPDI to compute the determinant and inverse of A .

Calling Sequence

```
CALL DPPFA  (DA, N, INFO)
CALL SPPFA  (SA, N, INFO)
CALL ZPPFA  (ZA, N, INFO)
CALL CPPFA  (CA, N, INFO)
```

```
void dppfa  (double *da, int n, int *info)
void sppfa  (float *sa, int n, int *info)
void zppfa  (doublecomplex *za, int n, int *info)
void cppfa  (complex *ca, int n, int *info)
```

Arguments

xA	On entry, the upper triangle of the matrix A . On exit, a Cholesky factorization of the matrix A .
N	Order of the matrix A . $N \geq 0$.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO > 0 Returns a value k if $U(k,k) = 0$ to indicate that xGESL will divide by zero if called.

Sample Program

```
PROGRAM TEST
IMPLICIT NONE

C      INTEGER          LENGTA, N
C      PARAMETER        (N = 4)
C      PARAMETER        (LENGTA = (N * N + N) / 2)

C      DOUBLE PRECISION A(LENGTA), B(N)
C      INTEGER          INFO

C      EXTERNAL          DPPFA, DPPSL
```

```

C
C      Initialize the array A to store in packed symmetric storage
C      mode the matrix A shown below. Initialize the array B to
C      store the vector B shown below.
C
C      4  3  2  1      30
C      A = 3  4  3  2      b = 20
C          2  3  4  3      20
C          1  2  3  4      40
C
C      DATA A / 4.0D0, 3.0D0, 4.0D0, 2.0D0, 3.0D0, 4.0D0,
$      1.0D0, 2.0D0, 3.0D0, 4.0D0 /
C      DATA B / 6.0D1, 2.0D1, 2.0D1, 6.0D1 /
C
C      PRINT 1000
C      PRINT 1010, A(1), A(2), A(4), A(7)
C      PRINT 1010, A(2), A(3), A(5), A(8)
C      PRINT 1010, A(4), A(5), A(6), A(9)
C      PRINT 1010, A(7), A(8), A(9), A(10)
C      PRINT 1020
C      PRINT 1030, B
C      CALL DPPFA (A, N, INFO)
C      IF (INFO .EQ. 0) THEN
C          CALL DPPSL (A, N, B)
C          PRINT 1040
C          PRINT 1030, B
C      ELSE
C          PRINT 1050
C      END IF
C
C      1000 FORMAT (1X, 'A in full form:')
C      1010 FORMAT (5(3X, F5.1))
C      1020 FORMAT (/1X, 'b:')
C      1030 FORMAT (3X, F5.1)
C      1040 FORMAT (/1X, 'A**(-1) * b:')
C      1050 FORMAT (/1X, 'A is too ill-conditioned.')
C
C      END

```

Sample Output

```

A in full form:
  4.0    3.0    2.0    1.0
  3.0    4.0    3.0    2.0
  2.0    3.0    4.0    3.0
  1.0    2.0    3.0    4.0

b:
 60.0
 20.0
 20.0
 60.0

```

```
A**(-1) * b:  
 32.0  
-20.0  
-20.0  
 32.0
```

Solution to a Linear System in a Cholesky-Factored Symmetric Positive Definite Matrix in Packed Storage

The subroutines described in this section solve the linear system $Ax = b$ for a symmetric positive definite matrix A in packed storage, which has been Cholesky-factored by xPPCO or xPPFA, and vectors b and x .

Calling Sequence

```
CALL DPPSL (DA, N, DB)
CALL SPPSL (SA, N, SB)
CALL ZPPSL (ZA, N, ZB)
CALL CPPSL (CA, N, CB)

void dppsl (double *da, int n, double *db)
void sppsl (float *sa, int n, float *sb)
void zppsl (doublecomplex *za, int n, doublecomplex *zb)
void cppsl (complex *ca, int n, complex *cb)
```

Arguments

xA	Cholesky factorization of the matrix A , as computed by xPPCO or xPPFA.
N	Order of the matrix A . $N \geq 0$.
xB	On entry, the right-hand side vector b . On exit, the solution vector x .

Sample Program

```
PROGRAM TEST
IMPLICIT NONE
C
C   INTEGER          LENGTA, N
C   PARAMETER       (N = 4)
C   PARAMETER       (LENGTA = (N * N + N) / 2)
C
C   DOUBLE PRECISION A(LENGTA), B(N), RCOND, WORK(N)
C   INTEGER          INFO
C
C   EXTERNAL         DPPCO, DPPSL
C
C   Initialize the array A to store in packed symmetric storage
```



```

C      mode the matrix A shown below.  Initialize the array B to
C      store the matrix B shown below.
C
C      4  3  2  1      20
C      A = 3  4  3  2      b = 20
C      2  3  4  3      20
C      1  2  3  4      20
C
C      DATA A / 4.0D0, 3.0D0, 4.0D0, 2.0D0, 3.0D0, 4.0D0,
$      1.0D0, 2.0D0, 3.0D0, 4.0D0 /
C      DATA B / 4*2.0D1 /
C
C      PRINT 1000
C      PRINT 1010, A(1), A(2), A(4), A(7)
C      PRINT 1010, A(2), A(3), A(5), A(8)
C      PRINT 1010, A(4), A(5), A(6), A(9)
C      PRINT 1010, A(7), A(8), A(9), A(10)
C      PRINT 1020
C      PRINT 1030, B
C      CALL DPPCO (A, N, RCOND, WORK, INFO)
C      IF ((RCOND + 1.0D0) .EQ. RCOND) THEN
C        PRINT 1040
C      END IF
C      CALL DPPSL (A, N, B)
C      PRINT 1050
C      PRINT 1030, B
C
C      1000 FORMAT (1X, 'A in full form:')
C      1010 FORMAT (5(3X, F7.3))
C      1020 FORMAT (/1X, 'b:')
C      1030 FORMAT (3X, F7.3)
C      1040 FORMAT (1X, 'A may be singular to working precision.')
C      1050 FORMAT (/1X, 'A**(-1) * b:')
C
C      END

```

Sample Output

```

A in full form:
  4.000    3.000    2.000    1.000
  3.000    4.000    3.000    2.000
  2.000    3.000    4.000    3.000
  1.000    2.000    3.000    4.000

b:
 20.000
 20.000
 20.000
 20.000

```

```
A**(-1) * b:  
4.000  
0.000  
0.000  
4.000
```

Solution to a Linear System in a Symmetric Positive Definite Tridiagonal Matrix

The subroutines described in this section solve the linear system $Ax = b$ for a symmetric positive definite tridiagonal matrix A and vectors b and x .

Calling Sequence

```
CALL DPTSL  (N, DDIAG, DOFFD, DB)
CALL SPTSL  (N, SDIAG, SOFFD, SB)
CALL ZPTSL  (N, ZDIAG, ZOFFD, ZB)
CALL CPTSL  (N, CDIAG, COFFD, CB)

void dptsl  (int n, double *ddiag, double *doffd, double *db)
void sptsl  (int n, float *sdiag, float *soffd, float *sb)
void zptsl  (int n, doublecomplex *zdiag, doublecomplex *zoffd,
             doublecomplex *zb)
void cptsl  (int n, complex *cdiag, complex *coffd, complex *cb)
```

Arguments

N	Order of the matrix A. $N \geq 0$.
xDIAG	Main diagonal elements of A.
xOFFD	Off diagonal elements of A. OFFD(1) through OFFD(N-1) contains the off diagonal elements; OFFD(N) is not referenced.
xB	On entry, the right-hand side vector b. On exit, the solution vector x.

Sample Program

```
PROGRAM TEST
IMPLICIT NONE
C
C      INTEGER          N
C      PARAMETER      (N = 4)
C
C      DOUBLE PRECISION B(N), SUBD(N), DIAG(N)
C
C      EXTERNAL        DPTSL
C
C      Initialize the arrays SUBD and DIAG to store the subdiagonal
C      and diagonal of the tridiagonal positive definite matrix A
C      shown below. Initialize the array B to store the right hand
C      side vector b shown below.
```

```

C
C      2  -1      6
C      A = -1  2  -1      b = 12
C           -1  2  -1      12
C           -1  2      6
C
C      DATA SUBD / -1.0D0, -1.0D0, -1.0D0, 8D8 /
C      DATA DIAG / 4*2.0D0 /
C      DATA B / 6.0D0, 1.2D1, 1.2D1, 6.0D0 /
C
C      PRINT 1000
C      PRINT 1010, DIAG(1), SUBD(1)
C      PRINT 1020, SUBD(1), DIAG(2), SUBD(2)
C      PRINT 1030,          SUBD(2), DIAG(3), SUBD(3)
C      PRINT 1040,          SUBD(3), DIAG(4)
C      PRINT 1050
C      PRINT 1060, B
C      CALL DPTSL (N, DIAG, SUBD, B)
C      PRINT 1070
C      PRINT 1060, B
C
C      1000 FORMAT (1X, 'A:')
C      1010 FORMAT (1X, 2(2X, F4.1))
C      1020 FORMAT (1X, 3(2X, F4.1))
C      1030 FORMAT (1X, 6X, 3(2X, F4.1))
C      1040 FORMAT (1X, 12X, 2(2X, F4.1))
C      1050 FORMAT (/1X, 'b:')
C      1060 FORMAT (3X, F6.1)
C      1070 FORMAT (/1X, 'A**(-1) * b:')
C
C      END

```

Sample Output

```

A:
  2.0  -1.0
-1.0   2.0  -1.0
      -1.0   2.0  -1.0
          -1.0   2.0

b:
  6.0
 12.0
 12.0
  6.0

A**(-1) * b:
 18.0
 30.0
 30.0
 18.0

```

QR Factorization of a General Matrix

The subroutines described in this section compute the QR factorization of a general matrix A . It is typical to follow a call to xQRDC with a call to xQRSL to solve $Ax = b$ or to xPODI to compute the determinant of A .

Calling Sequence

```
CALL DQRDC (DA, LDA, N, P, DQRAUX, IPIVOT, DWORK, JOB)
CALL SQRDC (SA, LDA, N, P, SQRAUX, IPIVOT, SWORK, JOB)
CALL ZQRDC (ZA, LDA, N, P, ZQRAUX, IPIVOT, ZWORK, JOB)
CALL CQRDC (CA, LDA, N, P, CQRAUX, IPIVOT, CWORK, JOB)

void dqrdc (double *da, int lda, int n, int p, double *dqraux, int
            *ipivot, int job)
void sqrdc (float *sa, int lda, int n, int p, float *sqraux, int
            *ipivot, int job)
void zqrdc (doublecomplex *za, int lda, int n, int p,
            doublecomplex *zqraux, int *ipivot, int job)
void cqrdc (complex *ca, int lda, int n, int p, complex *cqraux,
            int *ipivot, int job)
```

Arguments

xA	On entry, the matrix A . On exit, the upper triangle of A contains the matrix R and the strict lower triangle of A contains information that will allow construction of the matrix Q . If pivoting was requested, A contains the factorization of the original matrix A permuted by the requested pivots.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
N	Number of rows in the matrix A . $N \geq 0$.
P	Number of columns in the matrix A . $P \geq 0$.
xQRAUX	On exit, contains information required to construct the orthogonal matrix Q .
IPIVOT	If JOB selected no pivoting, then IPIVOT is not referenced. If JOB selected pivoting, then on entry to the subroutines, IPIVOT contains integers representing array indices that control the selection of pivot elements from the diagonal of A according to the system below: IPIVOT(k) > 0 $A(k,k)$ is an initial element. IPIVOT(k) = 0 $A(k,k)$ is a free element.

IPIVOT(k) < 0 A(k,k) is a final element.
 Before the decomposition is computed, symmetric row and column interchanges are used to move initial elements to the beginning of A and final elements to the end of A. During the computation, symmetric row and column interchanges are used to move the largest remaining free diagonal element into the pivot position.
 On exit, IPIVOT(k) contains the index of the diagonal element of A that was moved into the kth position.

xWORK Scratch array with a dimension of N. WORK is not referenced if JOB = 0.

JOB Determines how the factorization is done:
 0 with no pivoting
 not 0 with pivoting

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C   INTEGER          IDOB, IDORS, IDOXB, LDA, N, NCOLA, NOPIV
C   INTEGER          NROWA
C   PARAMETER        (IDOB = 100)
C   PARAMETER        (IDORS = 10)
C   PARAMETER        (IDOXB = 1)
C   PARAMETER        (N = 3)
C   PARAMETER        (LDA = N)
C   PARAMETER        (NCOLA = 2)
C   PARAMETER        (NOPIV = 0)
C   PARAMETER        (NROWA = N)
C
C   DOUBLE PRECISION A(LDA,NCOLA), B(NCOLA), NULL(N), QRAUX(N)
C   DOUBLE PRECISION RESID(N), WORK(N), Y(N)
C   INTEGER          ICOL, INFO, IROW, JOB, JPIVOT
C
C   EXTERNAL          DQRDC, DQRSL
C
C   Initialize the array A to store the matrix A shown below.
C   Initialize the array Y to store the vector y shown below.
C
C   A =  1  1      y =  1
C   A =  1  0      y =  0
C   A =  0  1      y = -5
C
C   DATA A / 1.0D0, 1.0D0, 0.0D0, 1.0D0, 0.0D0, 1.0D0 /
C   DATA Y / 1.0D0, 0.0D0, -5.0D0 /
C
C   PRINT 1000
C   PRINT 1010, ((A(IROW,ICOL), ICOL = 1, NCOLA), IROW = 1,
C   $   NROWA)

```

```
      PRINT 1020
      PRINT 1030, Y
      JOB = NOPIV
      CALL DQRDC (A, LDA, NROWA, NCOLA, QRAUX, JPIVOT, WORK, JOB)
      JOB = IDOB + IDORS + IDOXB
      CALL DQSL (A, LDA, NROWA, NCOLA, QRAUX, Y, NULL, NULL, B,
$             RESID, NULL, JOB, INFO)
      IF (INFO .EQ. 0) THEN
        PRINT 1040
        PRINT 1050, B
        PRINT 1060
        PRINT 1050, RESID
      ELSE
        PRINT 1070
      END IF
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (2(3X, F4.1))
1020 FORMAT (/1X, 'y:')
1030 FORMAT (3X, F4.1)
1040 FORMAT (/1X, 'Least squares solution:')
1050 FORMAT (3X, F4.1)
1060 FORMAT (/1X, 'Residual:')
1070 FORMAT (1X, 'A is singular.')
C
      END
```

Sample Output

```
A:
  1.0   1.0
  1.0   0.0
  0.0   1.0

y:
  1.0
  0.0
 -5.0

Least squares solution:
  2.0
 -3.0

Residual:
  2.0
 -2.0
 -2.0
```

Solution to a Linear System in a QR-Factored General Matrix

The subroutines described in this section solve the linear system $Ax = b$ for a general matrix A , which has been QR-factored by xQRDC, and vectors b and x .

Calling Sequence

```
CALL DQRSL  (DA, LDA, N, K, DQRAUX, DY, DQY, DQTY, DB, DRESID, DAB,
             JOB, INFO)
CALL SQRSL  (SA, LDA, N, K, SQRAUX, SY, SQY, SQTY, SB, SRESID, SAB,
             JOB, INFO)
CALL ZQRSL  (ZA, LDA, N, K, ZQRAUX, CY, CQY, ZQTY, ZB, ZRESID, ZAB,
             JOB, INFO)
CALL CQRSL  (CA, LDA, N, K, CQRAUX, ZY, ZQY, CQTY, CB, CRESID, CAB,
             JOB, INFO)

void dqrsl  (double *da, int lda, int n, int k, double *dqraux,
             double *dy, double *dqy, double *dqty, double *db,
             double *dresid, double *dab, int job, int *info)
void sqrsl  (float *sa, int lda, int n, int k, float *sqraux, float
             *sy, float *sqy, float *sqty, float *sb, float *sresid,
             float *sab, int job, int *info)
void zqrsl  (doublecomplex *za, int lda, int n, int k,
             doublecomplex *zqraux, doublecomplex *zy, doublecomplex
             *zqy, doublecomplex *zqty, doublecomplex *zb,
             doublecomplex *zresid, doublecomplex *zab, int job,
             int *info)
void cqrsl  (complex *ca, int lda, int n, int k, complex *cqraux,
             complex *cy, complex *cqy, complex *cqty, complex *cb,
             complex *cresid, complex *cab, int job, int *info)
```

Arguments

xA	Part of the QR factorization of matrix A as computed by xSRDC.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
N	Number of rows in the matrix AK where AK is described below. $N \geq 0$.
K	Number of columns in the matrix AK where AK is described below. $K \geq 0$.
xQRAUX	Auxiliary output from xQRDC.
xY	Vector to be manipulated by xQRSL.

xQY	On exit, QY contains $Q \times Y$ if its computation has been requested in JOB; QY is not referenced if its computation is not requested.
xQTY	On exit, QTY contains $Q^T \times Y$ if its computation has been requested in JOB; QTY is not referenced if its computation is not requested.
xB	On entry, the right-hand side vector b. On exit, the solution vector x. B is not referenced if its computation is not requested.
xRESID	On exit, RESID contains the least squares residual $y - AK \times b$ if its computation has been requested. RESID also is the orthogonal projection of y onto the orthogonal complement of the column space of AK. RESID is not referenced if its computation is not requested.
xAB	On exit, AB contains the least squares approximation $AK \times b$ if its computation has been requested. AB is the orthogonal projection of y onto the column space of x. AB is not referenced if its computation is not requested.
JOB	Integer in the form abcde; determines which operation or operations the subroutine will perform: a $\neq$ 0 compute QY b, c, d, or e $\neq$ 0 compute QTY c $\neq$ 0 compute B d $\neq$ 0 compute RESID e $\neq$ 0 compute AB
INFO	On exit: INFO = 0 Subroutine completed normally. INFO > 0 Returns a value k if the computation of B has been requested and R is singular; the value of k is then the index of the first zero element of R.

The matrix AK is constructed from the factored orthogonal matrix Q and upper triangular matrix R from xQRDC as follows:

$$AK = \begin{bmatrix} R \\ 0 \end{bmatrix}$$

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
INTEGER IDOB, IDORS, IDOXB, LDA, N, NCOLA, NOPIV

```

```

      INTEGER          NROWA
      PARAMETER        (IDOB = 100)
      PARAMETER        (IDORS = 10)
      PARAMETER        (IDOBX = 1)
      PARAMETER        (N = 3)
      PARAMETER        (LDA = N)
      PARAMETER        (NCOLA = 2)
      PARAMETER        (NOPIV = 0)
      PARAMETER        (NROWA = N)
C
      DOUBLE PRECISION A(LDA,NCOLA), B(NCOLA), NULL(N), QRAUX(N)
      DOUBLE PRECISION RESID(N), WORK(N), Y(N)
      INTEGER          ICOL, INFO, IROW, JOB, JPIVOT
C
      EXTERNAL          DQRDC, DQRSL
C
      C      Initialize the array A to store the matrix A shown below.
      C      Initialize the array Y to store the vector y shown below.
C
      C      A =  $\begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$  y =  $\begin{bmatrix} 1 \\ 0 \\ -5 \end{bmatrix}$ 
C
      DATA A / 1.0D0, 1.0D0, 0.0D0, 1.0D0, 0.0D0, 1.0D0 /
      DATA Y / 1.0D0, 0.0D0, -5.0D0 /
C
      PRINT 1000
      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, NCOLA), IROW = 1,
$      NROWA)
      PRINT 1020
      PRINT 1030, Y
      JOB = NOPIV
      CALL DQRDC (A, LDA, NROWA, NCOLA, QRAUX, JPIVOT, WORK, JOB)
      JOB = IDOB + IDORS + IDOBX
      CALL DQRSL (A, LDA, NROWA, NCOLA, QRAUX, Y, NULL, NULL, B,
$      RESID, NULL, JOB, INFO)
      IF (INFO .EQ. 0) THEN
          PRINT 1040
          PRINT 1050, B
          PRINT 1060
          PRINT 1050, RESID
      ELSE
          PRINT 1070
      END IF
C
      1000 FORMAT (1X, 'A:')
      1010 FORMAT (2(3X, F4.1))
      1020 FORMAT (/1X, 'y:')
      1030 FORMAT (3X, F4.1)
      1040 FORMAT (/1X, 'Least squares solution:')
      1050 FORMAT (3X, F4.1)
      1060 FORMAT (/1X, 'Residual:')
      1070 FORMAT (1X, 'A is singular.')
C
      END

```

Sample Output

```
A:
  1.0    1.0
  1.0    0.0
  0.0    1.0

y:
  1.0
  0.0
 -5.0

Least squares solution:
  2.0
 -3.0

Residual:
  2.0
 -2.0
 -2.0
```

UDU Factorization and Condition Number of a Symmetric Matrix

The subroutines described in this section compute the UDU factorization and condition number of a symmetric matrix A. If the condition number is not needed then xSIFA is slightly faster. It is typical to follow a call to xSICO with a call to xSISL to solve $Ax = b$ or to xSIDI to compute the determinant, inverse, and inertia of A.

Calling Sequence

```
CALL DSICO (DA, LDA, N, IPIVOT, DRCOND, DWORK)
CALL SSICO (SA, LDA, N, IPIVOT, SRCOND, SWORK)
CALL ZSICO (ZA, LDA, N, IPIVOT, DRCOND, ZWORK)
CALL CSICO (CA, LDA, N, IPIVOT, SRCOND, CWORK)

void dsico (double *da, int lda, int n, int *ipivot, double
            *drcond)
void ssico (float *sa, int lda, int n, int *ipivot, float *srcond)
void zsico (doublecomplex *za, int lda, int n, int *ipivot, double
            *drcond)
void csico (complex *ca, int lda, int n, int *ipivot, float
            *srcond)
```

Arguments

xA	On entry, the upper triangle of the matrix A. On exit, a UDU factorization of the matrix A. The strict lower triangle of A is not referenced.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
N	Order of the matrix A. $N \geq 0$.
IPIVOT	On exit, a vector of pivot indices.
xRCOND	On exit, an estimate of the reciprocal condition number of A. $0.0 \leq RCOND \leq 1.0$. As the value of RCOND gets smaller, operations with A such as solving $Ax = b$ may become less stable. If RCOND satisfies $RCOND + 1.0 = 1.0$ then A may be singular to working precision.
xWORK	Scratch array with a dimension of N.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
INTEGER          LDA, N
PARAMETER        (N = 4)
PARAMETER        (LDA = 5)

C
DOUBLE PRECISION A(LDA,N), B(N), RCOND, WORK(N)
INTEGER          ICOL, IPIVOT(N), IROW

C
EXTERNAL          DSICO, DSISL

C
C Initialize the array A to store the matrix A shown below.
C Initialize the array B to store the vector b shown below.
C
C      - .5      - .5      - .5      - .5      12
C  A = - .5      -1.5     -1.5     -1.5      b = 6
C      - .5      -1.5     -2.5     -2.5      6
C      - .5      -1.5     -2.5     -3.5      12
C
C      DATA A / -5.0D-1, 4*8D8, -5.0D-1, -1.5D0, 3*8D8, -5.0D-1,
$              -1.5D0, -2.5D0, 2*8D8, -5.0D-1, -1.5D0, -2.5D0,
$              -3.5D0, 8D8 /
C      DATA B / 1.2D1, 6.0D0, 6.0D0, 1.2D1 /

C
PRINT 1000
DO 100, IROW = 1, N
  PRINT 1010, (A(ICOL,IROW), ICOL = 1, IROW),
$            (A(IROW,ICOL), ICOL = IROW + 1, N)
100 CONTINUE
PRINT 1020
PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
PRINT 1030
PRINT 1040, B
CALL DSICO (A, LDA, N, IPIVOT, RCOND, WORK)
PRINT 1050, RCOND
IF ((RCOND + 1.0D0) .EQ. RCOND) THEN
  PRINT 1060
END IF
PRINT 1070, 1.0D0 / RCOND
CALL DSISL (A, LDA, N, IPIVOT, B)
PRINT 1080
PRINT 1040, B

C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F5.1))
1020 FORMAT (/1X, 'A in symmetric form: (* in unused elements)')
1030 FORMAT (/1X, 'b:')
1040 FORMAT (3X, F5.1)
1050 FORMAT (/1X, 'Reciprocal condition number of A:', F6.3)
1060 FORMAT (1X, 'A may be singular to working precision.')
1070 FORMAT (1X, 'Condition number of A: ', F6.3)

```

```
1080 FORMAT (/1X, 'A**(-1) * b:')
C
      END
```

Sample Output

```
A in full form:
-0.5   -0.5   -0.5   -0.5
-0.5   -1.5   -1.5   -1.5
-0.5   -1.5   -2.5   -2.5
-0.5   -1.5   -2.5   -3.5

A in symmetric form: (* in unused elements)
-0.5   -0.5   -0.5   -0.5
*****   -1.5   -1.5   -1.5
*****   *****   -2.5   -2.5
*****   *****   *****   -3.5

b:
12.0
6.0
6.0
12.0

Reciprocal condition number of A: 0.031
Condition number of A: 32.000

A**(-1) * b:
-30.0
6.0
6.0
-6.0
```

Determinant, Inertia, and Inverse of a UDU-Factored Symmetric Matrix

The subroutines described in this section compute the determinant, inertia, and inverse of a symmetric matrix A, which has been UDU-factored by xSICO or xSIFA.

Calling Sequence

```
CALL DSIDI (DA, LDA, N, IPIVOT, DDET, INERT, DWORK, JOB)
CALL SSIDI (SA, LDA, N, IPIVOT, SDET, INERT, SWORK, JOB)
CALL ZSIDI (ZA, LDA, N, IPIVOT, ZDET, ZWORK, JOB)
CALL CSIDI (CA, LDA, N, IPIVOT, CDET, CWORK, JOB)

void dsidi (double *da, int lda, int n, int *ipivot, double *ddet,
            int *inert, int job)
void ssidi (float *sa, int lda, int n, int *ipivot, float *sdet,
            int *inert, int job)
void zsidi (doublecomplex *za, int lda, int n, int *ipivot,
            doublecomplex *zdet, int job)
void csidi (complex *ca, int lda, int n, int *ipivot, complex
            *cdet, int job)
```

Arguments

xA	On entry, the UDU factorization of the matrix A, as computed by xSICO or xSIFA. On exit, if the <i>c</i> digit of JOB $\neq 0$, then the upper triangle of A contains the upper triangle of the inverse of the original matrix A if the inverse was requested, otherwise unchanged. The strict lower triangle of A is not referenced.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
N	Order of the original matrix A. $N \geq 0$.
IPIVOT	Pivot vector as computed by xSICO or xSIFA.
xDET	On exit, if the <i>b</i> digit of JOB $\neq 0$, then DET contains the determinant of the matrix A. The determinant is stored as $b \times 10^{\text{expon}}$ where <i>b</i> is stored in DET(1) and expon is stored in DET(2). $1.0 \leq \text{DET}(1) < 10.0$ or $\text{DET}(1) = 0.0$. If the <i>b</i> digit of JOB = 0, DET is not referenced.
INERT	On exit, if the <i>a</i> digit of JOB $\neq 0$, then INERT contains an integer triplet where: INERT(1) = number of positive eigenvalues

INERT(2) = number of negative eigenvalues
 INERT(3) = number of zero eigenvalues
 If the *a* digit of JOB = 0 then INERT is not referenced.
 xWORK Scratch array with a dimension of N.
 JOB Integer in the form abc for real subroutines or bc for complex subroutines; determines operation the subroutine will perform:
 a ≠ 0 Compute the inertia.
 b ≠ 0 Compute the determinant.
 c ≠ 0 Compute the inverse.
 Note that the inverse should not be computed if xSICO has set RCOND = 0 or if xSIFA has set INFO ≠ 0.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
  INTEGER          IDODET, IDOINR, IDOINV, LDA, N
  PARAMETER        (IDODET = 10)
  PARAMETER        (IDOINR = 100)
  PARAMETER        (IDOINV = 1)
  PARAMETER        (N = 4)
  PARAMETER        (LDA = N)
C
  DOUBLE PRECISION A(LDA,N), DET(2), WORK(N)
  INTEGER          ICOL, INERT(3), INFO, IPIVOT(N), IROW, JOB
C
  EXTERNAL DSIFA, DSIDI
C
  Initialize the array A to store the matrix A shown below.
C
C      -.5   -.5   -.5   -.5
C      A = -.5  -1.5 -1.5 -1.5
C           -.5  -1.5 -2.5 -2.5
C           -.5  -1.5 -2.5 -3.5
C
  DATA A / -5.0D-1, 3*8D8, -5.0D-1, -1.5D0, 2*8D8, -5.0D-1,
$          -1.5D0, -2.5D0, 8D8, -5.0D-1, -1.5D0, -2.5D0,
$          -3.5D0 /
C
  PRINT 1000
  DO 100, IROW = 1, N
    PRINT 1010, (A(ICOL,IROW), ICOL = 1, IROW),
$              (A(IROW,ICOL), ICOL = IROW + 1, N)
  100 CONTINUE
  PRINT 1020
  PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
  CALL DSIFA (A, LDA, N, IPIVOT, INFO)
  IF (INFO .EQ. 0) THEN

```



```

      JOB = IDOINR + IDODET + IDOINV
      CALL DSDI (A, LDA, N, IPIVOT, DET, INERT, WORK, JOB)
      PRINT 1030, DET(1) * (10.0D0 ** DET(2))
      PRINT 1040, INERT
      PRINT 1050
      DO 110, IROW = 1, N
        PRINT 1010, (A(ICOL,IROW), ICOL = 1, IROW),
$      (A(IROW,ICOL), ICOL = IROW + 1, N)
110  CONTINUE
      ELSE
        PRINT 1060
      END IF
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (4(3X, F5.1))
1020 FORMAT (/1X, 'A in symmetric form: (* in unused elements)')
1030 FORMAT (/1X, 'Determinant of A: ', F6.3)
1040 FORMAT (1X, 'Inertia of A: <', I1, ', ', I1, ', ', I1, '>')
1050 FORMAT (/1X, 'A**(-1):')
1060 FORMAT (/1X, 'A is too poorly conditioned.')
C
      END

```

Sample Output

```

A in full form:
-0.5   -0.5   -0.5   -0.5
-0.5   -1.5   -1.5   -1.5
-0.5   -1.5   -2.5   -2.5
-0.5   -1.5   -2.5   -3.5

A in symmetric form: (* in unused elements)
-0.5   -0.5   -0.5   -0.5
*****  -1.5   -1.5   -1.5
*****  *****  -2.5   -2.5
*****  *****  *****  -3.5

Determinant of A:  0.500
Inertia of A: <0,4,0>

A**(-1):
-3.0    1.0    0.0    0.0
 1.0   -2.0    1.0    0.0
 0.0    1.0   -2.0    1.0
 0.0    0.0    1.0   -1.0

```

UDU Factorization of a Symmetric Matrix

The subroutines described in this section compute the UDU factorization of a symmetric matrix A. It is typical to follow a call to xSIFA with a call to xSISL to solve $Ax = b$ or to xSIDI to compute the determinant, inverse, and inertia of A.

Calling Sequence

```
CALL DSIFA (DA, LDA, N, IPIVOT, INFO)
CALL SSIFA (SA, LDA, N, IPIVOT, INFO)
CALL ZSIFA (ZA, LDA, N, IPIVOT, INFO)
CALL CSIFA (CA, LDA, N, IPIVOT, INFO)

void dsifa (double *da, int lda, int n, int *ipivot, int *info)
void ssifa (float *sa, int lda, int n, int *ipivot, int *info)
void zsifa (doublecomplex *za, int lda, int n, int *ipivot, int
            *info)
void csifa (complex *ca, int lda, int n, int *ipivot, int *info)
```

Arguments

xA	On entry, the upper triangle of the matrix A. On exit, a UDU factorization of the matrix A. The strict lower triangle of A is not referenced.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
N	Order of the matrix A. $N \geq 0$.
IPIVOT	On exit, a vector of pivot indices.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO > 0 Returns a value k if the kth pivot block is singular to indicate that xSISL or xSIDI will divide by zero if called.

Sample Program

```
PROGRAM TEST
IMPLICIT NONE
C
INTEGER LDA, N
```

```

C      PARAMETER      (N = 4)
C      PARAMETER      (LDA = N)
C      DOUBLE PRECISION A(LDA,N), B(N)
C      INTEGER         ICOL, INFO, IPIVOT(N), IROW
C      EXTERNAL        DSIFA, DSISL
C      Initialize the array A to store the matrix A shown below.
C      Initialize the array B to store the vector B shown below.
C
C      A =
C      -1.5 -1.5 -2.5 -3.5
C      -1.5 -1.5 -2.5 -3.5
C      -1.5 -1.5 -2.5 -3.5
C      -1.5 -1.5 -2.5 -3.5
C
C      DATA A / -5.0D-1, 3*8D8, -5.0D-1, -1.5D0, 2*8D8, -5.0D-1,
$              -1.5D0, -2.5D0, 8D8, -5.0D-1, -1.5D0, -2.5D0,
$              -3.5D0 /
C      DATA B / 6.0D0, 1.2D1, 6.0D0, 1.2D1 /
C
C      PRINT 1000
C      DO 100, IROW = 1, N
C          PRINT 1010, (A(ICOL,IROW), ICOL = 1, IROW),
$              (A(IROW,ICOL), ICOL = IROW + 1, N)
100 CONTINUE
C      PRINT 1020
C      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
C      PRINT 1030
C      PRINT 1040, B
C      CALL DSIFA (A, LDA, N, IPIVOT, INFO)
C      IF (INFO .EQ. 0) THEN
C          CALL DSISL (A, LDA, N, IPIVOT, B)
C          PRINT 1050
C          PRINT 1040, B
C      ELSE
C          PRINT 1060
C      END IF
C
C      1000 FORMAT (1X, 'A in full form:')
C      1010 FORMAT (4(3X, F5.1))
C      1020 FORMAT (/1X, 'A in symmetric form: (* in unused elements)')
C      1030 FORMAT (/1X, 'b:')
C      1040 FORMAT (3X, F5.1)
C      1050 FORMAT (/1X, 'A*(-1) * b:')
C      1060 FORMAT (/1X, 'A is too poorly conditioned.')
C
C      END

```

Sample Output

```
A in full form:
-0.5  -0.5  -0.5  -0.5
-0.5  -1.5  -1.5  -1.5
-0.5  -1.5  -2.5  -2.5
-0.5  -1.5  -2.5  -3.5

A in symmetric form: (* in unused elements)
-0.5  -0.5  -0.5  -0.5
***** -1.5  -1.5  -1.5
***** ***** -2.5  -2.5
***** ***** ***** -3.5

b:
  6.0
 12.0
  6.0
 12.0

A**(-1) * b:
 -6.0
-12.0
 12.0
 -6.0
```

Solution to a Linear System in a UDU-Factored Symmetric Matrix

The subroutines described in this section solve the linear system $Ax = b$ for a symmetric matrix A , which has been UDU-factored by xSICO or xSIFA, and vectors b and x .

Calling Sequence

```
CALL DSISL  (DA, LDA, N, IPIVOT, DB)
CALL SSISL  (SA, LDA, N, IPIVOT, SB)
CALL ZSISL  (ZA, LDA, N, IPIVOT, ZB)
CALL CSISL  (CA, LDA, N, IPIVOT, CB)

void dsisl  (double *da, int lda, int n, int *ipivot, double *db)
void ssisl  (float *sa, int lda, int n, int *ipivot, float *sb)
void zsisl  (doublecomplex *za, int lda, int n, int *ipivot,
             doublecomplex *zb)
void csisl  (complex *ca, int lda, int n, int *ipivot, complex *cb)
```

Arguments

xA	UDU factorization of the matrix A , as computed by xSICO or xSIFA.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
N	Order of the matrix A . $N \geq 0$.
IPIVOT	Pivot vector as computed by xSICO or xSIFA.
xB	On entry, the right-hand side vector b .
	On exit, the solution vector x .

Sample Program

```
PROGRAM TEST
IMPLICIT NONE

C      INTEGER          LDA, N
C      PARAMETER        (N = 4)
C      PARAMETER        (LDA = N)

C      DOUBLE PRECISION A(LDA,N), B(N), RCOND, WORK(N)
C      INTEGER          ICOL, IPIVOT(N), IROW

C      EXTERNAL          DSICO, DSISL
```

```

C
C      Initialize the array A to store the matrix A shown below.
C      Initialize the array B to store the vector B shown below.
C
C      A =
C      -0.5  -0.5  -0.5  -0.5      12
C      -0.5  -1.5  -1.5  -1.5      6
C      -0.5  -1.5  -2.5  -2.5      6
C      -0.5  -1.5  -2.5  -3.5      12
C
C      DATA A / 4*-5.0D-1, -5.0D-1, 3*-1.5D0, -5.0D-1,
$          -1.5D0, 2*-2.5D0, -5.0D-1, -1.5D0, -2.5D0, -3.5D0 /
C      DATA B / 1.2D1, 6.0D0, 6.0D0, 1.2D1 /
C
C      PRINT 1000
C      DO 100, IROW = 1, N
C          PRINT 1010, (A(ICOL,IROW), ICOL = 1, IROW),
$              (A(IROW,ICOL), ICOL = IROW + 1, N)
100  CONTINUE
C      PRINT 1020
C      PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
C      PRINT 1030
C      PRINT 1040, B
C      CALL DSICO (A, LDA, N, IPIVOT, RCOND, WORK)
C      IF ((RCOND + 1.0D0) .EQ. RCOND) THEN
C          PRINT 1050
C      END IF
C      CALL DSISL (A, LDA, N, IPIVOT, B)
C      PRINT 1060
C      PRINT 1040, B
C
C      1000 FORMAT (1X, 'A in full form:')
C      1010 FORMAT (4(3X, F5.1))
C      1020 FORMAT (/1X, 'A in symmetric form: (* in unused elements)')
C      1030 FORMAT (/1X, 'b:')
C      1040 FORMAT (3X, F5.1)
C      1050 FORMAT (/1X, 'A may be singular to working precision.')
C      1060 FORMAT (/1X, 'A**(-1) * b:')
C
C      END

```

Sample Output

```

A in full form:
-0.5  -0.5  -0.5  -0.5
-0.5  -1.5  -1.5  -1.5
-0.5  -1.5  -2.5  -2.5
-0.5  -1.5  -2.5  -3.5

```

```
A in symmetric form: (* in unused elements)
```

```
-0.5   -0.5   -0.5   -0.5  
-0.5   -1.5   -1.5   -1.5  
-0.5   -1.5   -2.5   -2.5  
-0.5   -1.5   -2.5   -3.5
```

```
b:
```

```
12.0  
6.0  
6.0  
12.0
```

```
A**(-1) * b:
```

```
-30.0  
6.0  
6.0  
-6.0
```

UDU Factorization and Condition Number of a Symmetric Matrix in Packed Storage

The subroutines described in this section compute the UDU factorization and condition number of a symmetric matrix A in packed storage. If the condition number is not needed then xSPFA is slightly faster. It is typical to follow a call to xSPCO with a call to xSPSL to solve $Ax = b$ or to xSPDI to compute the determinant, inverse, and inertia of A.

Calling Sequence

```
CALL DSPCO (DA, N, IPIVOT, DRCOND, DWORK)
CALL SSPCO (SA, N, IPIVOT, SRCOND, SWORK)
CALL ZSPCO (ZA, N, IPIVOT, DRCOND, ZWORK)
CALL CSPCO (CA, N, IPIVOT, SRCOND, CWORK)

void dspco (double *da, int n, int *ipivot, double *drcond)
void sspco (float *sa, int n, int *ipivot, float *srcond)
void zspco (doublecomplex *za, int n, int *ipivot, double *drcond)
void cspco (complex *ca, int n, int *ipivot, float *srcond)
```

Arguments

xA	On entry, the upper triangle of the matrix A. On exit, a UDU factorization of the matrix A.
N	Order of the matrix A. $N \geq 0$.
IPIVOT	On exit, a vector of pivot indices.
xRCOND	On exit, an estimate of the reciprocal condition number of A. $0.0 \leq RCOND \leq 1.0$. As the value of RCOND gets smaller, operations with A such as solving $Ax = b$ may become less stable. If RCOND satisfies $RCOND + 1.0 = 1.0$ then A may be singular to working precision.
xWORK	Scratch array with a dimension of N.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER          LENGTA, N
  PARAMETER        (N = 3)
  PARAMETER        (LENGTA = (N * N + N) / 2)

C
  DOUBLE PRECISION A(LENGTA), B(N), RCOND, WORK(N)
  INTEGER          IPIVOT(N)

C
  EXTERNAL          DSPCO, DSPSL

C
C  Initialize the array A to store in packed symmetric format
C  the matrix A shown below. Initialize the array B to store
C  the vector b shown below.
C
C      1  0  4      30
C  A = 0  2  0      b =  4
C      4  0  1      15
C
  DATA A / 1.0D0, 0.0D0, 2.0D0, 4.0D0, 0.0D0, 1.0D0 /
  DATA B / 3.0D1, 4.0D0, 1.5D1 /

C
  PRINT 1000
  PRINT 1010, A(1), A(2), A(4)
  PRINT 1010, A(2), A(3), A(5)
  PRINT 1010, A(4), A(5), A(6)
  PRINT 1020
  PRINT 1030, B
  CALL DSPCO (A, N, IPIVOT, RCOND, WORK)
  IF ((RCOND + 1.0D0) .EQ. RCOND) THEN
    PRINT 1040
  END IF
  CALL DSPSL (A, N, IPIVOT, B)
  PRINT 1050
  PRINT 1030, B

C
1000 FORMAT (1X, 'A:')
1010 FORMAT (3(3X, F4.1))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (3X, F4.1)
1040 FORMAT (1X, 'A may be singular to working precision.')
1050 FORMAT (/1X, 'A**(-1) * b:')

C
  END

```

Sample Output

```
A:
  1.0    0.0    4.0
  0.0    2.0    0.0
  4.0    0.0    1.0

b:
 30.0
  4.0
 15.0

A**(-1) * b:
  2.0
  2.0
  7.0
```

Determinant, Inertia, and Inverse of a UDU-Factored Symmetric Matrix in Packed Storage

The subroutines described in this section compute the determinant, inertia, and inverse of a symmetric matrix A in packed storage, which has been UDU-factored by xSPCO or xSPFA.

Calling Sequence

```
CALL DSPDI  (DA, N, IPIVOT, DDET, INERT, DWORK, JOB)
CALL SSPDI  (SA, N, IPIVOT, SDET, INERT, SWORK, JOB)
CALL ZSPDI  (ZA, N, IPIVOT, ZDET, ZWORK, JOB)
CALL CSPDI  (CA, N, IPIVOT, CDET, CWORK, JOB)

void dspdi  (double *da, int n, int *ipivot, double *ddet, int
             *inert, int job)
void sspdi  (float *sa, int n, int *ipivot, float *sdet, int
             *inert, int job)
void zspdi  (doublecomplex *za, int n, int *ipivot, doublecomplex
             *zdet, int job)
void cspdi  (complex *ca, int n, int *ipivot, complex *cdet, int
             job)
```

Arguments

xA	On entry, the UDU factorization of the matrix A, as computed by xSPCO or SPFA. On exit, if the c digit of JOB $\neq 0$, then A contains the upper triangle of the inverse of the original matrix A; otherwise unchanged.
N	Order of the original matrix A. $N \geq 0$.
IPIVOT	Pivot vector as computed by xSPCO or xSPFA.
xDET	On exit, if the b digit of JOB $\neq 0$, then DET contains the determinant of the matrix A. The determinant is stored as $b \times 10^{\text{expon}}$ where b is stored in DET(1) and expon is stored in DET(2). $1.0 \leq \text{DET}(1) < 10.0$ or $\text{DET}(1) = 0.0$. If the b digit of JOB = 0, DET is not referenced.
INERT	On exit, if the a digit of JOB $\neq 0$ then INERT contains an integer triplet where: INERT(1) = number of positive eigenvalues INERT(2) = number of negative eigenvalues INERT(3) = number of zero eigenvalues

xWORK If the *a* digit of JOB = 0 then INERT is not referenced.
 JOB Scratch array with a dimension of N.
 Integer in the form abc for real subroutines or bc for complex
 subroutines; determines operation the subroutine will perform:
 a ≠ 0 Compute the inertia.
 b ≠ 0 Compute the determinant.
 c ≠ 0 Compute the inverse.
 Note that the inverse should not be computed if xSPCO has set
 RCOND = 0 or if xSPFA has set INFO ≠ 0.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
  INTEGER          IDODET, IDOINR, IDOINV, LENGTA, N
  PARAMETER        (IDODET = 10)
  PARAMETER        (IDOINR = 100)
  PARAMETER        (IDOINV = 1)
  PARAMETER        (N = 3)
  PARAMETER        (LENGTA = (N * N + N) / 2)
C
  DOUBLE PRECISION A(LENGTA), DET(2), RCOND, WORK(N)
  INTEGER          INERT(3), IPIVOT(N), JOB
C
  EXTERNAL          DSPCO, DSPDI
C
  C Initialize the array A to store in packed symmetric format
  C the matrix A shown below.
  C
  C      1  0  4
  C  A = 0  2  0
  C      4  0  3
  C
  DATA A / 3.0D0, 2.0D0, 2.0D0, 1.0D0, 1.0D0, 1.0D0 /
C
  PRINT 1000
  PRINT 1010, A(1), A(2), A(4)
  PRINT 1010, A(2), A(3), A(5)
  PRINT 1010, A(4), A(5), A(6)
  CALL DSPCO (A, N, IPIVOT, RCOND, WORK)
  IF ((RCOND + 1.0D0) .EQ. RCOND) THEN
    PRINT 1020
  END IF
  JOB = IDODET + IDOINV + IDOINR
  CALL DSPDI (A, N, IPIVOT, DET, INERT, WORK, JOB)
  PRINT 1030, DET(1) * (10.0D0 ** DET(2))
  PRINT 1040, INERT
  PRINT 1050
  PRINT 1010, A(1), A(2), A(4)

```

```
      PRINT 1010, A(2), A(3), A(5)
      PRINT 1010, A(4), A(5), A(6)
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (3(3X, F5.2))
1020 FORMAT (1X, 'A may be singular to working precision.')
1030 FORMAT (/1X, 'Determinant of A: ', F7.3)
1040 FORMAT (1X, 'Inertia of A: <', I1, ', ', I1, ', ', I1, '>')
1050 FORMAT (/1X, 'A**(-1):')
C
      END
```

Sample Output

```
A:
  3.00    2.00    1.00
  2.00    2.00    1.00
  1.00    1.00    1.00

Determinant of A:  1.000
Inertia of A: <3,0,0>

A**(-1):
  1.00   -1.00    0.00
 -1.00    2.00   -1.00
  0.00   -1.00    2.00
```

UDU Factorization of a Symmetric Matrix in Packed Storage

The subroutines described in this section compute the UDU factorization of a symmetric matrix A in packed storage. It is typical to follow a call to xSPFA with a call to xSPSL to solve $Ax = b$ or to xSPDI to compute the determinant, inverse, and inertia of A.

Calling Sequence

```
CALL DSPFA  (DA, N, IPIVOT, INFO)
CALL SSPFA  (SA, N, IPIVOT, INFO)
CALL ZSPFA  (ZA, N, IPIVOT, INFO)
CALL CSPFA  (CA, N, IPIVOT, INFO)
```

```
void dspfa  (double *da, int n, int *ipivot, int *info)
void sspfa  (float *sa, int n, int *ipivot, int *info)
void zspfa  (doublecomplex *za, int n, int *ipivot, int *info)
void cspfa  (complex *ca, int n, int *ipivot, int *info)
```

Arguments

xA	On entry, the upper triangle of the matrix A. On exit, a UDU factorization of the matrix A.
N	Order of the matrix A. $N \geq 0$.
IPIVOT	On exit, a vector of pivot indices.
INFO	On exit: INFO = 0 Subroutine completed normally. INFO > 0 Returns a value k if the kth pivot block is singular to indicate that xSPSL or xSPDI will divide by zero if called.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
INTEGER          IDODET, IDOINR, IDOINV, LENGTA, N
PARAMETER        (IDODET = 10)
PARAMETER        (IDOINR = 100)
PARAMETER        (IDOINV = 1)
PARAMETER        (N = 3)
PARAMETER        (LENGTA = (N * N + N) / 2)

C
DOUBLE PRECISION A(LENGTA), DET(2), WORK(N)
INTEGER          INERT(3), INFO, IPIVOT(N), JOB

C
EXTERNAL         DSPDI, DSPFA

C
C Initialize the array A to store in packed symmetric format
C the matrix A shown below.
C
C      1   0   4
C  A =  0   2   0
C      4   0   3
C
DATA A / 3.0D0, 2.0D0, 2.0D0, 1.0D0, 1.0D0, 1.0D0 /

C
PRINT 1000
PRINT 1010, A(1), A(2), A(4)
PRINT 1010, A(2), A(3), A(5)
PRINT 1010, A(4), A(5), A(6)
CALL DSPFA (A, N, IPIVOT, INFO)
IF (INFO .EQ. 0) THEN
    JOB = 111
    CALL DSPDI (A, N, IPIVOT, DET, INERT, WORK, JOB)
    PRINT 1020, DET(1) * (10.0D0 ** DET(2))
    PRINT 1030, INERT
    PRINT 1040
    PRINT 1010, A(1), A(2), A(4)
    PRINT 1010, A(2), A(3), A(5)
    PRINT 1010, A(4), A(5), A(6)
ELSE
    PRINT 1050
END IF

C
1000 FORMAT (1X, 'A:')
1010 FORMAT (3(3X, F5.2))
1020 FORMAT (/1X, 'Determinant of A: ', F7.3)
1030 FORMAT (1X, 'Inertia of A: <', I1, ', ', I1, ', ', I1, '>')
1040 FORMAT (/1X, 'A*(-1):')
1050 FORMAT (1X, 'A is too ill-conditioned.')

C
END

```

Sample Output

```
A:
  3.00    2.00    1.00
  2.00    2.00    1.00
  1.00    1.00    1.00

Determinant of A:  1.000
Inertia of A: <3,0,0>

A**(-1):
  1.00   -1.00    0.00
 -1.00    2.00   -1.00
  0.00   -1.00    2.00
```


Solution to a Linear System in a UDU-Factored Symmetric Matrix in Packed Storage

The subroutines described in this section solve the linear system $Ax = b$ for a symmetric matrix A in packed storage, which has been UDU-factored by xSPCO or xSPFA, and vectors b and x .

Calling Sequence

```
CALL DSPSL  (DA, N, IPIVOT, DB)
CALL SSPSL  (SA, N, IPIVOT, SB)
CALL ZSPSL  (ZA, N, IPIVOT, ZB)
CALL CSPSL  (CA, N, IPIVOT, CB)

void dspsl  (double *da, int n, int *ipivot, double *db)
void sspsl  (float *sa, int n, int *ipivot, float *sb)
void zpsl   (doublecomplex *za, int n, int *ipivot, doublecomplex
             *zb)
void cpsl   (complex *ca, int n, int *ipivot, complex *cb)
```

Arguments

xA	UDU factorization of the matrix A , as computed by xSPCO or xSPFA.
N	Order of the matrix A . $N \geq 0$.
IPIVOT	Pivot vector as computed by xSPCO or xSPFA.
xB	On entry, the right-hand side vector b . On exit, the solution vector x .

Sample Program

```
PROGRAM TEST
IMPLICIT NONE

C
C   INTEGER          LENGTA, N
C   PARAMETER        (N = 3)
C   PARAMETER        (LENGTA = (N * N + N) / 2)
C
C   DOUBLE PRECISION A(LENGTA), B(N), RCOND, WORK(N)
C   INTEGER          IPIVOT(N)
C
C   EXTERNAL          DSPCO, DSPSL
C
```

```

C      Initialize the array A to store in packed symmetric format
C      the matrix A shown below. Initialize the array B to store
C      the vector b shown below.
C
C      1  0  4      15
C      A = 0  2  0      b = 12
C      4  0  1      15
C
C      DATA A / 1.0D0, 0.0D0, 2.0D0, 4.0D0, 0.0D0, 1.0D0 /
C      DATA B / 1.5D1, 1.2D1, 1.5D1 /
C
C      PRINT 1000
C      PRINT 1010, A(1), A(2), A(4)
C      PRINT 1010, A(2), A(3), A(5)
C      PRINT 1010, A(4), A(5), A(6)
C      PRINT 1020
C      PRINT 1030, B
C      CALL DSPCO (A, N, IPIVOT, RCOND, WORK)
C      IF ((RCOND + 1.0D0) .EQ. RCOND) THEN
C          PRINT 1040
C      END IF
C      CALL DSPSL (A, N, IPIVOT, B)
C      PRINT 1050
C      PRINT 1030, B
C
C      1000 FORMAT (1X, 'A:')
C      1010 FORMAT (3(3X, F4.1))
C      1020 FORMAT (/1X, 'b:')
C      1030 FORMAT (3X, F4.1)
C      1040 FORMAT (1X, 'A may be singular to working precision.')
C      1050 FORMAT (/1X, 'A**(-1) * b:')
C
C      END

```

Sample Output

```

A:
  1.0    0.0    4.0
  0.0    2.0    0.0
  4.0    0.0    1.0

b:
 15.0
 12.0
 15.0

A**(-1) * b:
  3.0
  6.0
  3.0

```

Singular Value Decomposition of a General Matrix

The subroutines described in this section compute the singular value decomposition of a general matrix A.

Calling Sequence

CALL DSVDC	(DA, LDA, N, P, DSVALS, DE, DLSVEC, LDL, DRSVEC, LDR, DWORK, JOB, INFO)
CALL SSVDC	(SA, LDA, N, P, SSVALS, SE, SLSVEC, LDL, SRSVEC, LDR, SWORK, JOB, INFO)
CALL ZSVDC	(ZA, LDA, N, P, ZSVALS, ZE, ZLSVEC, LDL, ZRSVEC, LDR, ZWORK, JOB, INFO)
CALL CSVDC	(CA, LDA, N, P, CSVALS, CE, CLSVEC, LDL, CRSVEC, LDR, CWORK, JOB, INFO)
void dsvdc	(double *da, int lda, int n, int p, double *dsvals, double *de, double *dlsvec, int ldL, double *drsvec, int ldr, int job, int *info)
void ssvdc	(float *sa, int lda, int n, int p, float *ssvals, float *se, float *slsvec, int ldL, float *srsvec, int ldr, int job, int *info)
void zsvdc	(doublecomplex *za, int lda, int n, int p, doublecomplex *zsvals, doublecomplex *ze, doublecomplex *zlsvec, int ldL, doublecomplex *zrsvec, int ldr, int job, int *info)
void csvdc	(complex *ca, int lda, int n, int p, complex *csvals, complex *ce, complex *clsvec, int ldL, complex *crsvec, int ldr, int job, int *info)

Arguments

xA	Matrix A.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
N	Number of rows of the matrix A. $N \geq 0$.
P	Number of columns of the matrix A. $P \geq 0$.
xSVALS	On exit, the singular values of A arranged in descending order of magnitude.
xE	On exit, normally contains zero, but see INFO below.
xLSVEC	On exit, the matrix of left singular vectors; not referenced if the <i>a</i> digit of JOB = 0.

LDL	Leading dimension of LSVEC as specified in a dimension or type statement. $LDL \geq \max(1,N)$.
xRSVEC	On exit, the matrix of right singular vectors; not referenced if the <i>b</i> digit of JOB = 0.
LDR	Leading dimension of RSVEC as specified in a dimension or type statement. $LDR \geq \max(1,P)$.
xWORK	Scratch array with a dimension of N.
JOB	Integer in the form ab; determines operation subroutine will perform: a = 0 do not compute the left singular vectors a = 1 return the n left singular vectors in LSVEC $a \geq 2$ return the first min(N,P) singular vectors in LSVEC b = 0 do not compute the right singular vectors b = 1 return the right singular vectors in RSVEC
INFO	On exit: The singular values and their corresponding singular vectors SVALS(INFO+1), SVALS(INFO+2),...,SVALS(min(N,P)) are correct. If INFO = 0 then all singular values and singular vectors are correct. The matrix B defined as $LSVEC^T \times A \times RSVEC$ is the bidiagonal matrix with S on its diagonal and E on its superdiagonal. Therefore the singular values of A and B are the same.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER          JOB, LDL, LDR, LDX, N, P
  PARAMETER        (JOB = 21)
  PARAMETER        (N = 3)
  PARAMETER        (P = 3)
  PARAMETER        (LDL = N)
  PARAMETER        (LDR = P)
  PARAMETER        (LDX = N)

C
  DOUBLE PRECISION EPSILON, EXCEPT(P), LSVALS(LDL,N)
  DOUBLE PRECISION RSVALS(LDR,P), SVALS(N), WORK(N), X(LDX,P)
  INTEGER          I, ICOL, INFO, IRANK, IROW

C
  EXTERNAL          DSVDC
  INTRINSIC         ABS, SQRT

C
  Initialize the array X to store the matrix X shown below.
C
C      1  1  3
C      X = 0  1  1
C      1  0  1

```

```
C
      DATA X / 1.0D0, 0.0D0, 1.0D0, 1.0D0, 1.0D0, 0.0D0,
$         3.0D0, 1.0D0, 1.0D0 /
C
      PRINT 1000
      PRINT 1010, ((X(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
      CALL DSVDC (X, LDX, N, P, SVALS, EXCEPT, LSVALS, LDL,
$         RSVALS, LDR, WORK, JOB, INFO)
      PRINT 1020
      PRINT 1030, SVALS
C
C      Compute the unit roundoff
C
      EPSLON = 1.0D0
10  IF (DBLE (1.0D0 + EPSLON) .NE. 1.0D0) THEN
      EPSLON = EPSLON / 2.0D0
      GO TO 10
      END IF
C
C      Make a conservative estimate of the rank of A.
C
      IRANK = 0
      EPSLON = SQRT (EPSLON + EPSLON)
      DO 20, I = 1, N
      IF (ABS(SVALS(I)) .GT. EPSLON) THEN
      IRANK = IRANK + 1
      END IF
20  CONTINUE
      PRINT 1040, IRANK
C
1000 FORMAT (1X, 'X:')
1010 FORMAT (3(3X, F4.1))
1020 FORMAT (/1X, 'Singular values of X:')
1030 FORMAT (3X, F4.1)
1040 FORMAT (/1X, 'The rank of X is ', I1)
C
      END
```

Sample Output

```
X:
  1.0    1.0    3.0
  0.0    1.0    1.0
  1.0    0.0    1.0

Singular values of X:
  3.7
  1.0
  0.3

The rank of X is 3
```

Condition Number of a Triangular Matrix

The subroutines described in this section estimate the condition number of a triangular matrix A. It is typical to follow a call to xTRCO with a call to xTRSL to solve $Ax = b$ or to xTRDI to compute the determinant and inverse of A.

Calling Sequence

```
CALL DTRCO (DA, LDA, N, DRCOND, DWORK, JOB)
CALL STRCO (SA, LDA, N, SRCOND, SWORK, JOB)
CALL ZTRCO (ZA, LDA, N, DRCOND, ZWORK, JOB)
CALL CTRCO (CA, LDA, N, SRCOND, CWORK, JOB)

void dtrco (double *da, int lda, int n, double *drcond, int job)
void strco (float *sa, int lda, int n, float *srcond, int job)
void ztrco (doublecomplex *za, int lda, int n, double *drcond, int
job)
void ctrco (complex *ca, int lda, int n, float *srcond, int job)
```

Arguments

xA	Matrix A.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
N	Order of the matrix A. $N \geq 0$.
xRCOND	On exit, an estimate of the reciprocal condition number of A. $0.0 \leq RCOND \leq 1.0$. As the value of RCOND gets smaller, operations with A such as solving $Ax = b$ may become less stable. If RCOND satisfies $RCOND + 1.0 = 1.0$ then A may be singular to working precision.
xWORK	Scratch array with a dimension of N.
JOB	If JOB = 0, then A is lower triangular; otherwise A is upper triangular.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
INTEGER          ILOWER, LDA, N
PARAMETER        ( ILOWER = 0 )
PARAMETER        ( N = 5 )
PARAMETER        ( LDA = N )

C
DOUBLE PRECISION A(LDA,N), RCOND, WORK(N)
INTEGER          ICOL, IROW, JOB

C
EXTERNAL          DTRCO

C
C Initialize the array A to store the 5x5 triangular matrix A
C shown below.
C
C      1
C      1  1
C  A = 1  1  1
C      1  1  1  1
C      1  1  1  1  1
C
C      DATA A / 5*1.0D0, 8D8, 4*1.0D0, 2*8D8, 3*1.0D0, 3*8D8,
$              2*1.0D0, 4*8D8, 1.0D0 /

C
C Print the initial values of the arrays.
C
PRINT 1000
DO 100, IROW = 1, N
  PRINT 1010, (A(IROW,ICOL), ICOL = 1, IROW)
100 CONTINUE
PRINT 1020
PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, LDA)

C
C Estimate the condition number of the matrix.
C
JOB = ILOWER
CALL DTRCO (A, LDA, N, RCOND, WORK, JOB)
PRINT 1030, RCOND

C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (5(3X, F4.1))
1020 FORMAT (/1X, 'A in triangular form: (* in unused elements)')
1030 FORMAT (/1X, 'Reciprocal of the condition number:', F4.1)

C
END

```

Sample Output

```
A in full form:
1.0
1.0 1.0
1.0 1.0 1.0
1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0

A in triangular form: (* in unused elements)
1.0 **** *
1.0 1.0 **** *
1.0 1.0 1.0 **** *
1.0 1.0 1.0 1.0 ****
1.0 1.0 1.0 1.0 1.0

Reciprocal of the condition number: 0.1
```


Determinant and Inverse of a Triangular Matrix

The subroutines described in this section compute the determinant and inverse of a triangular matrix A.

Calling Sequence

```
CALL DTRDI  (DA, LDA, N, DDET, JOB, INFO)
CALL STRDI  (SA, LDA, N, SDET, JOB, INFO)
CALL ZTRDI  (ZA, LDA, N, ZDET, JOB, INFO)
CALL CTRDI  (CA, LDA, N, CDET, JOB, INFO)

void dtrdi  (double *da, int lda, int n, double *ddet, int job, int
             *info)
void strdi  (float *sa, int lda, int n, float *sdet, int job, int
             *info)
void ztrdi  (doublecomplex *za, int lda, int n, doublecomplex
             *zdet, int job, int *info)
void ctrdi  (complex *ca, int lda, int n, complex *cdet, int job,
             int *info)
```

Arguments

xA	On entry, the matrix A. On exit, the inverse of the original matrix A if the inverse was requested, otherwise unchanged.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
N	Order of the original matrix A. $N \geq 0$.
xDET	On exit, the determinant of the matrix A. The determinant is stored as $b \times 10^{\text{expon}}$ where b is stored in DET(1) and expon is stored in DET(2). $1.0 \leq \text{DET}(1) < 10.0$ or $\text{DET}(1) = 0.0$.
JOB	Determines which operation the subroutine will perform: 010 no determinant, inverse of lower triangular A 011 no determinant, inverse of upper triangular A 100 determinant, no inverse 110 determinant, inverse of lower triangular A 111 determinant, inverse of upper triangular A
INFO	On exit:

INFO = 0 Subroutine completed normally.
 INFO > 0 Contains the index of a zero element of A if the inverse is requested and A is singular.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
  INTEGER          INDTLO, LDA, N
  PARAMETER        (INDTLO = 110)
  PARAMETER        (N = 5)
  PARAMETER        (LDA = N)
C
  DOUBLE PRECISION A(LDA,N), DET(2)
  INTEGER          ICOL, INFO, IROW, JOB
C
  EXTERNAL          DTRDI
C
  C Initialize the array A to store the 5x5 triangular matrix A
  C shown below.
  C
  C      1
  C      1  -1
  C  A = 1  -2  1
  C      1  -3  3  -1
  C      1  -4  6  -4  1
  C
  DATA A / 5*1.0D0, 8D8, -1.0D0, -2.0D0, -3.0D0, -4.0D0,
$          2*8D8, 1.0D0, 3.0D0, 6.0D0, 3*8D8, -1.0D0,
$          -4.0D0, 4*8D8, 1.0D0 /
C
  C Print the initial values of the arrays.
  C
  PRINT 1000
  DO 100, IROW = 1, N
    PRINT 1010, (A(IROW,ICOL), ICOL = 1, IROW)
  100 CONTINUE
  PRINT 1020
  PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, LDA)
C
  C Factor the matrix in banded form.
  C
  JOB = INDTLO
  CALL DTRDI (A, LDA, N, DET, JOB, INFO)
  IF (INFO .EQ. 0) THEN
    PRINT 1030, DET(1) * (10.0 ** DET(2))
    PRINT 1040
    DO 110, IROW = 1, N
      PRINT 1010, (A(IROW,ICOL), ICOL = 1, IROW)
  110 CONTINUE
  ELSE

```

```

      PRINT 1050, INFO
      END IF
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (5(3X, F4.1))
1020 FORMAT (/1X, 'A in triangular form: (* in unused elements)')
1030 FORMAT (/1X, 'det(A) = ', F4.1)
1040 FORMAT (/1X, 'A**(-1):')
1050 FORMAT (1X, 'A appears singular at ', I2)
C
      END

```

Sample Output

```

A in full form:
 1.0
 1.0 -1.0
 1.0 -2.0 1.0
 1.0 -3.0 3.0 -1.0
 1.0 -4.0 6.0 -4.0 1.0

A in triangular form: (* in unused elements)
 1.0 ****
 1.0 -1.0 ****
 1.0 -2.0 1.0 ****
 1.0 -3.0 3.0 -1.0 ****
 1.0 -4.0 6.0 -4.0 1.0

det(A) = 1.0

A**(-1):
 1.0
 1.0 -1.0
 1.0 -2.0 1.0
 1.0 -3.0 3.0 -1.0
 1.0 -4.0 6.0 -4.0 1.0

```

Solution to a Linear System in a Triangular Matrix

The subroutines described in this section solve the linear system $Ax = b$ for a triangular matrix A and vectors b and x .

Calling Sequence

```
CALL DTRSL (DA, LDA, N, DB, JOB, INFO)
CALL STRSL (SA, LDA, N, SB, JOB, INFO)
CALL ZTRSL (ZA, LDA, N, ZB, JOB, INFO)
CALL CTRSL (CA, LDA, N, CB, JOB, INFO)

void dtrsl (double *da, int lda, int n, double *db, int job, int
            *info)
void strsl (float *sa, int lda, int n, float *sb, int job, int
            *info)
void ztrsl (doublecomplex *za, int lda, int n, doublecomplex *zb,
            int job, int *info)
void ctrsl (complex *ca, int lda, int n, complex *cb, int job, int
            *info)
```

Arguments

xA	Matrix A.								
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.								
N	Order of the matrix A. $N \geq 0$.								
xB	On entry, the right-hand side vector b. On exit, the solution vector x.								
JOB	Determines which operation the subroutine will perform: <table> <tbody> <tr> <td>00</td> <td>solve $Ax = b$, A lower triangular</td> </tr> <tr> <td>01</td> <td>solve $Ax = b$, A upper triangular</td> </tr> <tr> <td>10</td> <td>solve $A^H x = b$, A lower triangular</td> </tr> <tr> <td>11</td> <td>solve $A^H x = b$, A upper triangular</td> </tr> </tbody> </table> Note that $A^T x = A^H x$ for real matrices.	00	solve $Ax = b$, A lower triangular	01	solve $Ax = b$, A upper triangular	10	solve $A^H x = b$, A lower triangular	11	solve $A^H x = b$, A upper triangular
00	solve $Ax = b$, A lower triangular								
01	solve $Ax = b$, A upper triangular								
10	solve $A^H x = b$, A lower triangular								
11	solve $A^H x = b$, A upper triangular								
INFO	On exit: <table> <tbody> <tr> <td>INFO = 0</td> <td>Subroutine completed normally.</td> </tr> <tr> <td>INFO > 0</td> <td>Returns the index of the first zero diagonal element of A.</td> </tr> </tbody> </table>	INFO = 0	Subroutine completed normally.	INFO > 0	Returns the index of the first zero diagonal element of A.				
INFO = 0	Subroutine completed normally.								
INFO > 0	Returns the index of the first zero diagonal element of A.								

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
INTEGER          LDA, LOTRAN, N
PARAMETER        (LOTRAN = 10)
PARAMETER        (N = 5)
PARAMETER        (LDA = N)

C
DOUBLE PRECISION A(LDA,N), B(N)
INTEGER          ICOL, INFO, IROW, JOB

C
EXTERNAL          DTRSL

C
C Initialize the array A to store the 5x5 triangular matrix A
C shown below.
C
C      1          5
C      1  1          4
C  A = 1  1  1          b = 3
C      1  1  1  1      2
C      1  1  1  1  1    1
C
C      DATA A / 5*1.0D0, 8D8, 4*1.0D0, 2*8D8, 3*1.0D0, 3*8D8,
$      2*1.0D0, 4*8D8, 1.0D0 /
C      DATA B / 5.0D0, 4.0D0, 3.0D0, 2.0D0, 1.0D0 /

C
C Print the initial values of the arrays.
C
PRINT 1000
DO 100, IROW = 1, N
  PRINT 1010, (A(IROW,ICOL), ICOL = 1, IROW)
100 CONTINUE
PRINT 1020
PRINT 1010, ((A(IROW,ICOL), ICOL = 1, N), IROW = 1, LDA)
PRINT 1030
PRINT 1040, B

C
C Solve the matrix in banded form.
C
JOB = LOTRAN
CALL DTRSL (A, LDA, N, B, JOB, INFO)
IF (INFO .EQ. 0) THEN
  PRINT 1050
  PRINT 1040, B
ELSE
  PRINT 1060, INFO
END IF

C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (5(3X, F4.1))
1020 FORMAT (/1X, 'A in triangular form: (* in unused elements)')
1030 FORMAT (/1X, 'b:')

```

```

1040 FORMAT (3X, F4.1)
1050 FORMAT (/1X, 'A' '**(-1) * b:')
1060 FORMAT (1X, 'A appears singular at ', I2)
C
      END

```

Sample Output

```

A in full form:
  1.0
  1.0  1.0
  1.0  1.0  1.0
  1.0  1.0  1.0  1.0
  1.0  1.0  1.0  1.0  1.0

A in triangular form: (* in unused elements)
  1.0  ****  ****  ****  ****
  1.0  1.0  ****  ****  ****
  1.0  1.0  1.0  ****  ****
  1.0  1.0  1.0  1.0  ****
  1.0  1.0  1.0  1.0  1.0

b:
  5.0
  4.0
  3.0
  2.0
  1.0

A' **(-1) * b:
  1.0
  1.0
  1.0
  1.0
  1.0

```

Basic Linear Algebra Subprograms, Level 1 (BLAS1)

3 

This chapter describes Basic Linear Algebra Subprograms Level 1 (BLAS1) as they are implemented in the Sun Performance Library. BLAS1 is a collection of linear algebra subprograms to perform common linear algebra operations with vector operands. For more information about BLAS1, refer to the information in this chapter or see "Basic Linear Algebra Subprograms for FORTRAN Usage" written by Lawson *et al* and published in 1979 in the *ACM Transactions on Mathematical Software* (Vol. 5, pp. 308-323).

The first two sections of this chapter describe:

- conventions followed in naming arguments and subroutines and conventions used for passing arguments into or out of subroutines
- extensions to the standard versions of BLAS1.

The remaining sections of this chapter describe the operation of each of the BLAS1 subprograms. For each subprogram the interface is shown, the operation of the subprogram is described, and a short FORTRAN 77 program is given with sample output.

Conventions

The types of all variables and functions are determined as shown below:

```
IMPLICIT COMPLEX (C)
IMPLICIT DOUBLE PRECISION (D)
IMPLICIT INTEGER (I-N)
IMPLICIT COMPLEX*32 Q
IMPLICIT REAL (S)
IMPLICIT DOUBLE COMPLEX (Z)
```

The first one or two letters of a subroutine name identifies the types of the arguments. In most cases, only the first letter is used to identify type. The remaining letters that are not used to identify the type will identify the operation that the subroutine performs. For example, CSCAL will multiply the elements of a complex vector by a complex constant. The second character in CSSCAL indicates that this subroutine will multiply the elements of a complex vector by a real constant.

The first letter of a function name identifies the type of the function, and in some cases the next two letters specify the types of the arguments. In most cases, the types of the arguments match the types of the return type of the function. For example, DDOT will return the double precision dot product of two double precision vectors. The second character in DSDOT indicates that this function will return the double precision dot product of two single precision vectors.

To specify the group of subroutines that perform the same operation but are of different types, this manual uses a lowercase x. For example, xAXPY refers to the subroutines DAXPY, SAXPY, ZAXPY, and CAXPY. This same convention is used for arguments: xY refers to the arguments DY, SY, ZY, and CY.

Vector arguments always are identified by X or Y. The type of the vector is indicated by its first letter, so DX is a double precision vector and CX is a complex vector. The size of the X array is $\max(1, N*|INCX|)$ and the size of the Y array is $\max(1, N*|INCY|)$.

Extensions to the Basic Linear Algebra Subprograms

The following extensions to BLAS1 are implemented in the Sun Performance Library.

1. The Sun Performance Library includes the double complex subprograms suggested in the original paper by Lawson.
2. The format in which the QC argument to DQDOTx is stored differs from the format used in the model implementation on some machines. In general this is not a problem because programmers almost never examine the value of the QC argument.
3. If a non-positive value is supplied for the INCX argument in one of the subprograms that require INCX to be positive, then the subprogram will return immediately. This is compatible with the specification in Lawson's original paper, but not with Lawson's model implementation. This is consistent with changes to BLAS made by Dongarra for LAPACK.
4. This version of Sun Performance Library contains a native C/C++ interface to BLAS. Conventions for this interface are described in detail in the preface to this manual.

Sum of the Absolute Values of a Vector

DASUM and SASUM compute

$$asum \leftarrow \sum_{i=1}^n |x_i|$$

which is the l_1 -norm of a vector x .

DZASUM and SCASUM compute

$$asum \leftarrow \sum_{i=1}^n |\Re x_i| + |\Im x_i|$$

Note that DZASUM and SCASUM do not compute

$$\sum_{i=1}^n [\Re x_i^2 + \Im x_i^2]^{1/2}$$

which is the l_1 -norm of a complex vector.

Calling Sequence

```
DL1 = DASUM (N, DX, INCX)
SL1 = SASUM (N, SX, INCX)
DL1 = DZASUM (N, ZX, INCX)
SL1 = SCASUM (N, CX, INCX)
```

```
double dasum (int n, double *dx, int incx)
real sasum (int n, float *sx, int incx)
double dzasum (int n, doublecomplex *zx, int incx)
real scasum (int n, complex *cx, int incx)
```

Arguments

N	Number of elements in the vector. $N \geq 0$.
xX	Input vector X; the size of array X must be at least $\max(1, N* INCX)$.
INCX	Specifies the storage spacing between successive elements of the vector X. A value of one indicates that the elements of the vector are consecutive in memory. $INCX > 0$. If X refers to a two-dimensional array and the value of INCX is equal to the leading dimension of X then the elements of the vector will be a row of X. If X refers to a two-dimensional array and the value of INCX is equal to the leading dimension of X plus one then the elements of the vector will be a diagonal of X.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C   INTEGER          N
C   PARAMETER      (N = 4)
C
C   DOUBLE PRECISION SUM, X(N)
C   INTEGER        I, INCX
C
C   DOUBLE PRECISION DASUM
C   EXTERNAL      DASUM
C
C   Initialize the array X to store the vector x shown below.
C
C   1.0
C   x = 2.0
C   3.0
C   4.0
C
C   DATA X / 1.0D0, 2.0D0, 3.0D0, 4.0D0 /
C
C   PRINT 1000
C   PRINT 1010, (X(I), I = 1, N)
C   INCX = 1
C   SUM = DASUM (N, X, INCX)
C   PRINT 1020, SUM
C
C   1000 FORMAT (1X, 'x:')
C   1010 FORMAT (3X, F3.1)
C   1020 FORMAT (/1X, 'The L-1 norm of x is ', F4.1)
C
C   END

```

Sample Output

```
x:
  1.0
  2.0
  3.0
  4.0

The L-1 norm of x is 10.0
```

Product of a Scalar and Vector Plus a Vector

The subroutines in this section compute

$$y \leftarrow \alpha x + y$$

Calling Sequence

```
CALL DAXPY (N, DALPHA, DX, INCX, DY, INCY)
CALL SAXPY (N, SALPHA, SX, INCX, SY, INCY)
CALL ZAXPY (N, ZALPHA, ZX, INCX, ZY, INCY)
CALL CAXPY (N, CALPHA, CX, INCX, CY, INCY)
```

```
void daxpy (int n, double da, double *dx, int incx, double *dy,
            (int incy)
void saxpy (int n, float sa, float *sx, int incx, float *sy, int
            incy)
void zaxpy (int n, doublecomplex *za, doublecomplex *zx, int incx,
            doublecomplex *zy, int incy)
void caxpy (int n, complex *ca, complex *cx, int incx, complex
            *cy, int incy)
```

Arguments

N	Number of elements in the vector. $N \geq 0$.
xALPHA	Scalar that will be applied to vector X.
xX	Input vector X; the size of array X must be at least $\max(1, N* INCX)$.
INCX	Specifies the storage spacing between successive elements of the vector X. A value of one indicates that the elements of the vector are consecutive in memory. If X refers to a two-dimensional array and the value of INCX is equal to the leading dimension of X then the elements of the vector will be a row of X. If X refers to a two-dimensional array and the value of INCX is equal to the leading dimension of X plus one then the elements of the vector will be a diagonal of X.
xY	Result vector Y; the size of array Y must be at least $\max(1, N* INCY)$.
INCY	Specifies the storage spacing between successive elements of the vector Y. A value of one indicates that the elements of the vector are consecutive in memory.

If Y refers to a two-dimensional array and the value of INCY is equal to the leading dimension of Y then the elements of the vector will be a row of Y.

If Y refers to a two-dimensional array and the value of INCY is equal to the leading dimension of Y plus one then the elements of the vector will be a diagonal of Y.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
  INTEGER          N
  PARAMETER        (N = 3)
C
  INTEGER          ICOL, INCX, INCY, IROW
  DOUBLE PRECISION SCALE, X(N,N)
C
  EXTERNAL          DAXPY
C
  Initialize the array X to contain the matrix X shown below.
C
C      5.0  4.0  3.0
C      X = 2.5 3.0 3.5
C      1.0  1.5  2.0
C
  DATA X / 5.0D0, 2.5D0, 1.0D0, 4.0D0, 3.0D0, 1.5D0,
$          3.0D0, 3.5D0, 2.0D0 /
C
  PRINT 1000
  PRINT 1010, ((X(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
  INCX = N
  INCY = N
  SCALE = - X(2,1) / X(1,1)
  CALL DAXPY (N, SCALE, X(1,1), INCX, X(2,1), INCY)
  PRINT 1020
  PRINT 1010, ((X(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
C
1000 FORMAT (1X, 'X:')
1010 FORMAT (3(2X, F4.2))
1020 FORMAT (/1X, 'After one elimination:')
C
  END

```

Sample Output

```
X:
5.00  4.00  3.00
2.50  3.00  3.50
1.00  1.50  2.00

After one elimination:
5.00  4.00  3.00
0.00  1.00  2.00
1.00  1.50  2.00
```

Copy a Vector

The subroutines described in this section copy vector x into vector y.

Calling Sequence

```
CALL DCOPY (N, DX, INCX, DY, INCY)
CALL SCOPY (N, SX, INCX, SY, INCY)
CALL ZCOPY (N, ZX, INCX, ZY, INCY)
CALL CCOPY (N, CX, INCX, CY, INCY)

void dcopy (int n, double *dx, int incx, double *dy, int incy)
void scopy (int n, float *sx, int incx, float *sy, int incy)
void zcopy (int n, doublecomplex *zx, int incx, doublecomplex *zy,
            int incy)
void ccopy (int n, complex *cx, int incx, complex *cy, int incy)
```

Arguments

N	Number of elements in the vector. $N \geq 0$.
xX	Input vector X; the size of array X must be at least $\max(1, N* INCX)$.
INCX	Specifies the storage spacing between successive elements of the vector X. A value of one indicates that the elements of the vector are consecutive in memory. If X refers to a two-dimensional array and the value of INCX is equal to the leading dimension of X then the elements of the vector will be a row of X. If X refers to a two-dimensional array and the value of INCX is equal to the leading dimension of X plus one then the elements of the vector will be a diagonal of X.
xY	Result vector Y; the size of array Y must be at least $\max(1, N* INCY)$.
INCY	Specifies the storage spacing between successive elements of the vector Y. A value of one indicates that the elements of the vector are consecutive in memory. If Y refers to a two-dimensional array and the value of INCY is equal to the leading dimension of Y then the elements of the vector will be a row of Y. If Y refers to a two-dimensional array and the value of INCY is equal to the leading dimension of Y plus one then the elements of the vector will be a diagonal of Y.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C      INTEGER          N
C      PARAMETER        (N = 3)

C      INTEGER          I, INCX, INCY, J
C      DOUBLE PRECISION A(N,N), B(N,N)

C      EXTERNAL          DCOPY

C      Initialize the array A to store the matrix A shown below.
C
C      A = 1  4  7
C          2  5  8
C          3  6  9
C
C      DATA A / 1.0D0, 2.0D0, 3.0D0, 4.0D0, 5.0D0, 6.0D0,
$             7.0D0, 8.0D0, 9.0D0 /

C      PRINT 1000
C      PRINT 1010, ((A(I,J), J = 1, N), I = 1, N)
C      INCX = 0
C      INCY = 1

C      Use DCOPY to initialize all element of B to zero.
C
C      CALL DCOPY (N * N, 0.0D0, INCX, B, INCY)

C      Copy A' into B.
C
C      INCX = 1
C      INCY = N
C      DO 10, I = 1, N
C          CALL DCOPY (N, A(1,I), INCX, B(I,1), INCY)
10  CONTINUE
C      PRINT 1020
C      PRINT 1010, ((B(I,J), J = 1, N), I = 1, N)

C      1000 FORMAT (1X, 'A:')
C      1010 FORMAT (3(3(2X, F3.1) / ))
C      1020 FORMAT (1X, 'A':')

C      END

```

Sample Output

```
A:
 1.0  4.0  7.0
 2.0  5.0  8.0
 3.0  6.0  9.0

A':
 1.0  2.0  3.0
 4.0  5.0  6.0
 7.0  8.0  9.0
```

Dot Product (Inner Product)

The subroutines described in this section compute variations on the dot product of the vectors x and y . DDOT, SDOT, ZDOTU, and CDOTU compute:

$$\text{dot} = \sum_{i=1}^n (x_i y_i)$$

ZDOTC and CDOTC compute the dot product of the complex conjugate of x with y :

$$\text{dot} = \sum_{i=1}^n (\bar{x}_i y_i)$$

DSDOT computes the dot product of two real vectors x and y and accumulates the sum into a double precision accumulator.

$$\text{dot} = \sum_{i=1}^n (x_i y_i)$$

SDSDOT computes the dot product of two real vectors x and y , adds a real constant to the result, and then returns the result rounded to single precision.

$$\text{dot} = b + \sum_{i=1}^n (x_i y_i)$$

DQDOTI computes the dot product of two double precision vectors x and y , adds a real constant, stores the result in the extended precision variable C , and then returns the value of C rounded to double precision as the result of the function.

$$\text{dot} = b + \sum_{i=1}^n (x_i y_i)$$

DQDOTA computes the dot product of two double precision vectors x and y, adds B and C, stores the result in the extended precision variable C, and then returns the value of C rounded to double precision as the result of the function.

$$\text{dot} = b + c + \sum_{i=1}^n (x_i y_i)$$

Calling Sequence

```
DINNER = DDOT (N, DX, INCX, DY, INCY)
SINNER = SDOT (N, SX, INCX, SY, INCY)
ZINNER = ZDOTU (N, ZX, INCX, ZY, INCY)
CINNER = CDOTU (N, CX, INCX, CY, INCY)
ZINNER = ZDOTC (N, ZX, INCX, ZY, INCY)
CINNER = CDOTC (N, CX, INCX, CY, INCY)
DINNER = DSDOT (N, SX, INCX, SY, INCY)
SINNER = SDSDOT (N, SB, SX, INCX, SY, INCY)
DINNER = DQDOTI (N, DB, QC, DX, INCX, DY, INCY)
DINNER = DQDOTA (N, DB, QC, DX, INCX, DY, INCY)
```

```
double ddot(int n, double *dx, int incx, double *dy, int incy)
float sdot(int n, float *sx, int incx, float *sy, int incy)
doublecomplex *zdotu(int n, doublecomplex *cx, int incx,
doublecomplex *cy, int incy)
complex *cdotu(int n, complex *cx, int incx, complex *cy, int incy)
doublecomplex *zdotc(int n, doublecomplex *cx, int incx,
doublecomplex *cy, int incy)
complex *cdotc (int n, complex *cx, int incx, complex *cy, int incy)
```

Arguments

N	Number of elements in the vector. $N \geq 0$.
xB	Only used in SDSDOT, DQDOTI, and DQDOTA. Scalar added to dot product in extended precision.
QC	Only used in DQDOTI and DQDOTA. Extended precision result of an operation. The representation of QC varies from one machine to the next, so directly using the value of QC in a way that depends on the representation of QC can inhibit portability.
xX	Input vector X; the size of array X must be at least $\max(1, N* INCX)$.

INCX	Specifies the storage spacing between successive elements of the vector X. A value of one indicates that the elements of the vector are consecutive in memory. If X refers to a two-dimensional array and the value of INCX is equal to the leading dimension of X then the elements of the vector will be a row of X. If X refers to a two-dimensional array and the value of INCX is equal to the leading dimension of X plus one then the elements of the vector will be a diagonal of X.
xY	Result vector Y; the size of array Y must be at least $\max(1, N * INCX)$.
INCY	Specifies the storage spacing between successive elements of the vector Y. A value of one indicates that the elements of the vector are consecutive in memory. If Y refers to a two-dimensional array and the value of INCY is equal to the leading dimension of Y then the elements of the vector will be a row of Y. If Y refers to a two-dimensional array and the value of INCY is equal to the leading dimension of Y plus one then the elements of the vector will be a diagonal of Y.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C      INTEGER          N
C      PARAMETER        (N = 3)
C
C      INTEGER          I, INCX, INCY
C      DOUBLE PRECISION DOTPRD, X(N), Y(N)
C
C      DOUBLE PRECISION DDOT
C      EXTERNAL         DDOT
C
C      Initialize the arrays X and Y to store the vectors x and y
C      shown below.
C
C      1      3
C      x = 2      y = 0
C      3      -1
C
C      DATA X / 1.0D0, 2.0D0, 3.0D0 /
C      DATA Y / 3.0D0, 0.0D0, -1.0D0 /
C
C      PRINT 1000
C      PRINT 1010, (X(I), Y(I), I = 1, 3)
C      INCX = 1

```

```

        INCY = 1
        DOTPRD = DDOT (N, X, INCX, Y, INCY)
        IF (DOTPRD .EQ. 0.0D0) THEN
            PRINT 1020
        ELSE
            PRINT 1030, DOTPRD
        END IF
C
1000 FORMAT (3X, 'x', 5X, 'y')
1010 FORMAT (1X, F4.1, 2X, F4.1)
1020 FORMAT (/1X, 'Vectors x and y are orthogonal.')
1030 FORMAT (/1X, 'dot (x,y) = ', F5.1)
C
        END

```

Sample Output

```

        x      y
    1.0    3.0
    2.0    0.0
    3.0   -1.0

    Vectors x and y are orthogonal.

```

Givens Plane Rotation

The subroutines in this section construct a Given's plane rotation. It is typical to follow the call to xROTG with a call to xROT to apply the rotation to annihilate an element of the input vector.

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix} \times \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}$$

Calling Sequence

```
CALL DROTG  (DA, DB, DC, DS)
CALL SROTG  (SA, SB, SC, SS)
```

```
void drotg (double *da, double *db, double *dc, double *ds)
void srotg (float *sa, float *sb, float *sc, float *ss)
```

Arguments

xA	On entry, the value a shown in the equation above. On exit, the values shown here: If $ a > b $ $\text{sign}(a) \times (a^2 + b^2)^{1/2}$ Otherwise $\text{sign}(b) \times (a^2 + b^2)^{1/2}$
xB	On entry, the value b shown in the equation above. On exit, the values shown here: If $ a > b $ s If $ b \geq a $ and $c \neq 0$ $1/c$ If $c = 0$ 1
xC	On exit, the value c shown in the equation above.
xS	On exit, the value s shown in the equation above.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER          N
  PARAMETER       (N = 5)
C
  DOUBLE PRECISION A, B, C, S, X(N), Y(N)
  INTEGER          I, INCX, INCY
C
  EXTERNAL         DROT, DROTG
C
  Initialize the arrays X and Y to store the vectors x and y
  shown below.
C
C      1      2
C      2      4
C      x = 3    y = 8
C      4      16
C      5      32
C
  DATA X / 1.0D0, 2.0D0, 3.0D0, 4.0D0, 5.0D0 /
  DATA Y / 2.0D0, 4.0D0, 8.0D0, 1.6D1, 3.2D1 /
C
  PRINT 1000
  PRINT 1010, (X(I), Y(I), I = 1, N)
C
  Set up the rotation that will eliminate the last element.
C
  A = X(N)
  B = Y(N)
  CALL DROTG (A, B, C, S)
C
  Apply the rotation to eliminate the last element and print
  the results.
C
  INCX = 1
  INCY = 1
  CALL DROT (N, X, INCX, Y, INCY, C, S)
  PRINT 1020
  PRINT 1010, (X(I), Y(I), I = 1, N)
C
1000 FORMAT (7X, 'x ', 8X, 'y')
1010 FORMAT (2(3X, F7.4))
1020 FORMAT (/7X, 'x''', 8X, 'y''')
C
  END

```


Sample Output

x	y
1.0000	2.0000
2.0000	4.0000
3.0000	8.0000
4.0000	16.0000
5.0000	32.0000

x'	y'
2.1304	-0.6793
4.2608	-1.3585
8.3672	-1.7290
16.4257	-1.4820
32.3883	0.0000

Applied Given's Plane Rotation

The subroutines in this section apply a Given's plane rotation constructed by xROTG:

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} \leftarrow \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \times \begin{bmatrix} x_i \\ y_i \end{bmatrix} \quad \text{for } i = 1, \dots, N$$

Calling Sequence

```
CALL DROT    (N, DX, INCX, DY, INCY, DC, DS)
CALL SROT    (N, SX, INCX, SY, INCY, SC, SS)

void drot    (int n, double *dx, int incx, double *dy, int incy,
              double dc, double ds)
void srot    (int n, float *sx, int incx, float *sy, int incy, float
              sc, float ss)
```

Arguments

N	Number of elements in the vector. $N \geq 0$.
xX	Input vector X; the size of array X must be at least $\max(1, N* INCX)$.
INCX	Specifies the storage spacing between successive elements of the vector X. A value of one indicates that the elements of the vector are consecutive in memory. If X refers to a two-dimensional array and the value of INCX is equal to the leading dimension of X then the elements of the vector will be a row of X. If X refers to a two-dimensional array and the value of INCX is equal to the leading dimension of X plus one then the elements of the vector will be a diagonal of X.
xY	Result vector Y; the size of array Y must be at least $\max(1, N* INCY)$.
INCY	Specifies the storage spacing between successive elements of the vector Y. A value of one indicates that the elements of the vector are consecutive in memory. If Y refers to a two-dimensional array and the value of INCY is equal to the leading dimension of Y then the elements of the vector will be a row of Y.

If Y refers to a two-dimensional array and the value of INCY is equal to the leading dimension of Y plus one then the elements of the vector will be a diagonal of Y.

xC Value of the C argument from xROTG.
 xS Value of the S argument from xROTG.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C      INTEGER          N
C      PARAMETER      (N = 5)
C
C      DOUBLE PRECISION A, B, C, S, X(N), Y(N)
C      INTEGER          I, INCX, INCY
C
C      EXTERNAL          DROT, DROTG
C
C      Initialize the arrays X and Y to store the vectors x and y
C      shown below.
C
C          1          2
C          2          4
C      x = 3          y = 8
C          4          16
C          5          32
C
C      DATA X / 1.0D0, 2.0D0, 3.0D0, 4.0D0, 5.0D0 /
C      DATA Y / 2.0D0, 4.0D0, 8.0D0, 1.6D1, 3.2D1 /
C
C      PRINT 1000
C      PRINT 1010, (X(I), Y(I), I = 1, N)
C
C      Set up the rotation that will eliminate the last element.
C
C      A = X(N)
C      B = Y(N)
C      CALL DROTG (A, B, C, S)
C
C      Apply the rotation to eliminate the last element and print
C      the results.
C
C      INCX = 1
C      INCY = 1
C      CALL DROT (N, X, INCX, Y, INCY, C, S)
C      PRINT 1020
C      PRINT 1010, (X(I), Y(I), I = 1, N)
C
C      1000 FORMAT (7X, 'x', 9X, 'y')
C      1010 FORMAT (2(3X, F7.4))

```

```
1020 FORMAT (/7X, 'x'', 8X, 'y'')
C
      END
```

Sample Output

x	y
1.0000	2.0000
2.0000	4.0000
3.0000	8.0000
4.0000	16.0000
5.0000	32.0000
x'	y'
2.1304	-0.6793
4.2608	-1.3585
8.3672	-1.7290
16.4257	-1.4820
32.3883	0.0000

Modified Given's Plane Rotation

The subroutines in this section construct Gentleman's modified Given's plane rotation. It is typical to follow the call to xROTMG with a call to xROTM to apply the plane rotation to annihilate an element of the input vector.

$$\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} d_1^{1/2} & 0 \\ 0 & d_2^{1/2} \end{bmatrix} \times \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

Calling Sequence

```
CALL DROTMG (DD1, DD2, DB1, DB2, DPARAM)
CALL SROTMG (SD1, SD2, SB1, SB2, SPARAM)

void drotmg (double *dd1, double *dd2, double *db1, double *db2,
double *dparam)
void srotmg (float *sd1, float *sd2, float *sb1, float *sb2, float
*sparam)
```

Arguments

xD1	On entry, contains the value d_1 shown in the equation above. On exit, changed to represent the effect of the transformation.
xD2	On entry, contains the value d_2 shown in the equation above. On exit, changed to represent the effect of the transformation.
xB1	On entry, contains the value b_1 shown in the equation above. On exit, changed to represent the effect of the transformation.
xB2	On entry, contains the value b_2 shown in the equation above. On exit, changed to represent the effect of the transformation.
xPARAM	On entry, PARAM(1) indicates the form of the H matrix. The H matrix is defined in the section that describes the subroutines xROTM. On exit, elements 2 through 5 of PARAM contain the H matrix as defined above. Elements of PARAM that are not listed are not changed.

If PARAM(1) = 1 then $h_{12} = 1$, $h_{21} = -1$, PARAM(2) = h_{11} , and PARAM(5) = h_{22} .
If PARAM(1) = 0 then $h_{11} = h_{22} = 1$, PARAM(3) = h_{21} , and PARAM(4) = h_{12} .

If $\text{PARAM}(1) = -1$ then $\text{PARAM}(2) = h_{11}$, $\text{PARAM}(3) = h_{21}$,
 $\text{PARAM}(4) = h_{12}$, and $\text{PARAM}(5) = h_{22}$.
 If $\text{PARAM}(1) = -2$ then $H = I$ and no elements of PARAM are
 modified.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C      INTEGER          N
C      PARAMETER        (N = 4)
C
C      DOUBLE PRECISION B1, B2, D1, D2, PARAM(5), X(N), Y(N)
C      INTEGER          I, INCX, INCY
C
C      EXTERNAL          DROTM, DROTMG
C
C      Initialize the arrays X and Y to store the vectors x and y
C      shown below.
C
C      5      2
C      x = 5   y = 4
C      5      8
C      5      16
C
C      DATA X / 5.0D0, 5.0D0, 5.0D0, 5.0D0 /
C      DATA Y / 2.0D0, 4.0D0, 8.0D0, 1.6D1 /
C
C      PARAM(1) = -2
C      PRINT 1000
C      PRINT 1010, (X(I), Y(I), I = 1, N)
C
C      Set up the rotation that will eliminate the third element.
C
C      D1 = 1.0D0
C      D2 = 1.0D0
C      B1 = X(3)
C      B2 = Y(3)
C      CALL DROTMG (D1, D2, B1, B2, PARAM)
C
C      Apply the rotation to eliminate the third element and print
C      the results.
C
C      INCX = 1
C      INCY = 1
C      CALL DROTM (N, X, INCX, Y, INCY, PARAM)
C      PRINT 1020
C      PRINT 1010, (X(I), Y(I), I = 1, N)
C
C      1000 FORMAT (7X, 'x ', 8X, 'y')
```

```
1010 FORMAT (2(3X, F7.4))  
1020 FORMAT (/7X, 'x''', 8X, 'y''')  
C  
      END
```

Sample Output

x	y
5.0000	2.0000
5.0000	4.0000
5.0000	8.0000
5.0000	16.0000

x'	y'
5.1250	-3.7500
7.1250	-2.5000
11.1250	0.0000
19.1250	5.0000

Applied Modified Given's Transformation

The subroutines in this section apply Gentleman's modified Given's plane rotation as defined by:

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} \leftarrow H \begin{bmatrix} x_i \\ y_i \end{bmatrix} \quad \text{for } i = 1, \dots, N$$

Calling Sequence

```
CALL DROTM (N, DX, INCX, DY, INCY, DPARAM)
CALL SROTM (N, SX, INCX, SY, INCY, SPARAM)

void drotm (int n, double *dx, int incx, double *dy, int incy,
            double *dparam)
void srotm (int n, float *sx, int incx, float *sy, int incy, float
            sc, float *sparam)
```

Arguments

N	Number of elements in the vector. $N \geq 0$.
xX	Input vector X; the size of array X must be at least $\max(1, N* INCX)$.
INCX	Specifies the storage spacing between successive elements of the vector X. A value of one indicates that the elements of the vector are consecutive in memory. If X refers to a two-dimensional array and the value of INCX is equal to the leading dimension of X then the elements of the vector will be a row of X. If X refers to a two-dimensional array and the value of INCX is equal to the leading dimension of X plus one then the elements of the vector will be a diagonal of X.
xY	Result vector Y; the size of array Y must be at least $\max(1, N* INCY)$.
INCY	Specifies the storage spacing between successive elements of the vector Y. A value of one indicates that the elements of the vector are consecutive in memory. If Y refers to a two-dimensional array and the value of INCY is equal to the leading dimension of Y then the elements of the vector will be a row of Y.

If Y refers to a two-dimensional array and the value of INCY is equal to the leading dimension of Y plus one then the elements of the vector will be a diagonal of Y.

xPARAM Value of the PARAM argument from xROTMG.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C      INTEGER          N
C      PARAMETER        (N = 4)
C
C      DOUBLE PRECISION B1, B2, D1, D2, PARAM(5), X(N), Y(N)
C      INTEGER          I, INCX, INCY
C
C      EXTERNAL DROTM, DROTMG
C
C      Initialize the arrays X and Y to store the vectors x and y
C      shown below.
C
C      5      2
C      x = 5   y = 4
C      5      8
C      5      16
C
C      DATA X / 5.0D0, 5.0D0, 5.0D0, 5.0D0 /
C      DATA Y / 2.0D0, 4.0D0, 8.0D0, 1.6D1 /
C
C      PARAM(1) = -2
C      PRINT 1000
C      PRINT 1010, (X(I), Y(I), I = 1, N)
C
C      Set up the rotation that will eliminate the third element.
C
C      D1 = 1.0D0
C      D2 = 1.0D0
C      B1 = X(3)
C      B2 = Y(3)
C      CALL DROTMG (D1, D2, B1, B2, PARAM)
C
C      Apply the rotation to eliminate the third element and print
C      the results.
C
C      INCX = 1
C      INCY = 1
C      CALL DROTM (N, X, INCX, Y, INCY, PARAM)
C      PRINT 1020
C      PRINT 1010, (X(I), Y(I), I = 1, N)
C
C      1000 FORMAT (1X, 6X, 'x', 9X, 'y')

```

```

1010 FORMAT (2(3X, F7.4))
1020 FORMAT (/1X, 6X, 'x'', 8X, 'y'')
C
      END

```

Sample Output

x	y
5.0000	2.0000
5.0000	4.0000
5.0000	8.0000
5.0000	16.0000

x'	y'
5.1250	-3.7500
7.1250	-2.5000
11.1250	0.0000
19.1250	5.0000

Index of Element with Maximum Absolute Value

IDAMAX and ISAMAX return the smallest index i where

$$|x_i| = \max_{j=1}^n |x_j|$$

which is the l_∞ -norm of a vector x .

IZAMAX and ICAMAX return the smallest index i where

$$|\Re x_i| + |\Im x_i| = \max_{j=1}^n [|\Re x_j| + |\Im x_j|]$$

Note that this does not return an index i such that

$$\Re x_i^2 + \Im x_i^2 = \max_{j=1}^n [|\Re x_j|^2 + |\Im x_j|^2]^{1/2}$$

which is the l_∞ -norm for a complex vector.

Note also that the FORTRAN interfaces return an index based at one while the C interfaces return an index based at zero. This is in keeping with the convention of each language in which the first element of a FORTRAN array is usually 1 and the first element of a C array is 0.

Calling Sequence

`I = IDAMAX (N, DX, INCX)`

`I = ISAMAX (N, SX, INCX)`

`I = IZAMAX (N, ZX, INCX)`

`I = ICAMAX (N, CX, INCX)`

`int idamax (int n, double *dx, int incx)`

`int isamax (int n, float *sx, int incx)`

`int izamax (int n, doublecomplex *zx, int incx)`

`int icamax (int n, complex *cx, int incx)`

Arguments

<code>N</code>	Number of elements in the vector. $N \geq 0$.
<code>xX</code>	Input vector X; the size of array X must be at least $\max(1, N* INCX)$.

INCX Specifies the storage spacing between successive elements of the vector X. A value of one indicates that the elements of the vector are consecutive in memory. $INCX > 0$.

If X refers to a two-dimensional array and the value of INCX is equal to the leading dimension of X then the elements of the vector will be a row of X.

If X refers to a two-dimensional array and the value of INCX is equal to the leading dimension of X plus one then the elements of the vector will be a diagonal of X.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
  INTEGER      N
  PARAMETER (N = 5)
C
  COMPLEX      X(N)
  INTEGER      I, ILARGE, INCX
C
  INTEGER      ICAMAX
  EXTERNAL     ICAMAX
C
C   Initialize the array X to store the vector x shown below.
C
C       5-4i
C      -3-2i
C   x =  5-4i
C       6+0i
C       0+8i
C
C   DATA X / (5.0, -4.0), (-3.0, -2.0), (5.0, -4.0), (6.0, 0.0),
$          (0.0, 8.0) /
C
  INCX = 1
  ILARGE = ICAMAX (N, X, INCX)
  PRINT 1000
  PRINT 1010, (X(I), I = 1, N)
  PRINT 1020, ILARGE
C
1000 FORMAT (1X, 'x:')
1010 FORMAT (3X, '(', F4.1, ', ', F4.1, ')')
1020 FORMAT (/1X, 'Largest element in the vector x is item ', I1)
C
  END

```

Sample Output

```
x:
( 5.0, -4.0)
(-3.0, -2.0)
( 5.0, -4.0)
( 6.0,  0.0)
( 0.0,  8.0)

Largest element in the vector x is item 1
```

Euclidean Norm of a Vector

The subroutines described in this section compute the l2-norm of the vector x:

$$nrm2 \leftarrow \left[\sum_{i=1}^n |x_i|^2 \right]^{1/2}$$

Calling Sequence

```
DL2 = DNRM2 (N, DX, INCX)
SL2 = SNRM2 (N, SX, INCX)
DL2 = DZNRM2 (N, ZX, INCX)
SL2 = SCNRM2 (N, CX, INCX)
```

```
double dnr2(int n, double *dx, int incx)
float snrm2(int n, float *sx, int incx)
double dznrm2(int n, doublecomplex *zx, int incx)
float scnrm2(int n, complex *cx, int incx)
```

Arguments

N	Number of elements in the vector. $N \geq 0$.
xX	Input vector X; the size of array X must be at least $\max(1, N* INCX)$.
INCX	Specifies the storage spacing between successive elements of the vector X. A value of one indicates that the elements of the vector are consecutive in memory. $INCX > 0$. If X refers to a two-dimensional array and the value of INCX is equal to the leading dimension of X then the elements of the vector will be a row of X. If X refers to a two-dimensional array and the value of INCX is equal to the leading dimension of X plus one then the elements of the vector will be a diagonal of X.

Sample Program

```
PROGRAM TEST
IMPLICIT NONE
C
INTEGER          N
```

```

C      PARAMETER      (N = 4)
C      DOUBLE PRECISION  TWONRM, X(N)
C      INTEGER           INCX
C      DOUBLE PRECISION  DNRM2
C      EXTERNAL          DNRM2
C      Initialize the array X to store the vector x shown below.
C      x =
C      1
C      5
C      3
C      1
C      DATA X / 1.0D0, 5.0D0, 3.0D0, 1.0D0 /
C      PRINT 1000
C      PRINT 1010, X
C      INCX = 1
C      TWONRM = DNRM2 (N, X, INCX)
C      PRINT 1020, TWONRM
1000 FORMAT (1X, 'x:')
1010 FORMAT (4X, F3.1)
1020 FORMAT (/1X, 'The L-2 norm of x is ', F3.1)
C      END

```

Sample Output

```

x:
  1.0
  5.0
  3.0
  1.0

The L-2 norm of x is 6.0

```

Scale a Vector

The subroutines described in this section compute

$$y \leftarrow \alpha y_i$$

Calling Sequence

```
CALL DSCAL  (N, DALPHA, DY, INCY)
CALL SSCAL  (N, SALPHA, SY, INCY)
CALL ZSCAL  (N, ZALPHA, ZY, INCY)
CALL CSCAL  (N, CALPHA, CY, INCY)
CALL ZDSCAL (N, DALPHA, ZY, INCY)
CALL CSSCAL (N, SALPHA, CY, INCY)
```

```
void dscal (int n, double da, double *dx, int incx)
void sscal (int n, float sa, float *sx, int incx)
void zscal (int n, doublecomplex *za, doublecomplex *zx, int incx)
void cscal (int n, complex *ca, complex *cx, int incx)
void zdscal (int n, double da, doublecomplex *zx, int incx)
void csscal (int n, float sa, complex *cx, int incx)
```

Arguments

N	Number of elements in the vector. $N \geq 0$.
xALPHA	Scalar that will be applied to Y.
xY	Input vector Y; the size of array Y must be at least $\max(1, N* INCY)$.
INCY	Specifies the storage spacing between successive elements of the vector Y. A value of one indicates that the elements of the vector are consecutive in memory. $INCY > 0$. If Y refers to a two-dimensional array and the value of INCY is equal to the leading dimension of Y then the elements of the vector will be a row of Y. If Y refers to a two-dimensional array and the value of INCY is equal to the leading dimension of Y plus one then the elements of the vector will be a diagonal of Y.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
  INTEGER          N
  PARAMETER        (N = 3)
C
  INTEGER          ICOL, INCX, IROW
  DOUBLE PRECISION SCALE, X(N,N)
C
  EXTERNAL          DSCAL
C
  Initialize the array X to store the vector x shown below.
C
C      5.0  5.5  6.0
C      x = 2.5  3.0  3.5
C      1.0  1.5  2.0
C
  DATA X / 5.0D0, 2.5D0, 1.0D0, 5.5D0, 3.0D0, 1.5D0,
$          6.0D0, 3.5D0, 2.0D0 /
C
  PRINT 1000
  PRINT 1010, ((X(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
  INCX = N
  SCALE = X(1,1) / X(2,1)
  CALL DSCAL (N, SCALE, X(2,1), INCX)
  PRINT 1020
  PRINT 1010, ((X(IROW,ICOL), ICOL = 1, N), IROW = 1, N)
C
1000 FORMAT (1X, 'Before scaling row 2:')
1010 FORMAT (3(3X, F3.1))
1020 FORMAT (/1X, 'After scaling row 2:')
C
  END

```

Sample Output

```

Before scaling row 2:
  5.0  5.5  6.0
  2.5  3.0  3.5
  1.0  1.5  2.0

After scaling row 2:
  5.0  5.5  6.0
  5.0  6.0  7.0
  1.0  1.5  2.0

```

Swap Two Vectors

The subroutines described in this section exchange the vectors *x* and *y*.

Calling Sequence

```
CALL DSWAP (N, DX, INCX, DY, INCY)
CALL SSWAP (N, SX, INCX, SY, INCY)
CALL ZSWAP (N, ZX, INCX, ZY, INCY)
CALL CSWAP (N, CX, INCX, CY, INCY)

void dswap (int n, double *dx, int incx, double *dy, int incy)
void sswap (int n, float *sx, int incx, float *sy, int incy)
void zswap (int n, doublecomplex *zx, int incx, doublecomplex *zy,
            int incy)
void cswap (int n, complex *cx, int incx, complex *cy, int incy)
```

Arguments

N	Number of elements in the vector. $N \geq 0$.
xX	Input vector X; the size of array X must be at least $\max(1, N* INCX)$.
INCX	Specifies the storage spacing between successive elements of the vector X. A value of one indicates that the elements of the vector are consecutive in memory. If X refers to a two-dimensional array and the value of INCX is equal to the leading dimension of X then the elements of the vector will be a row of X. If X refers to a two-dimensional array and the value of INCX is equal to the leading dimension of X plus one then the elements of the vector will be a diagonal of X.
xY	Result vector Y; the size of array Y must be at least $\max(1, N* INCY)$.
INCY	Specifies the storage spacing between successive elements of the vector Y. A value of one indicates that the elements of the vector are consecutive in memory. If Y refers to a two-dimensional array and the value of INCY is equal to the leading dimension of Y then the elements of the vector will be a row of Y. If Y refers to a two-dimensional array and the value of INCY is equal to the leading dimension of Y plus one then the elements of the vector will be a diagonal of Y.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
INTEGER N
PARAMETER (N = 3)
INTEGER I, INCX, INCY, J, LARGE
DOUBLE PRECISION A(N,N)
INTEGER IDAMAX
EXTERNAL DSWAP, IDAMAX
C
C   Initialize the array A to store the matrix A shown below.
C
C   A =
C   1.0  1.5  4.5
C   3.0  2.5  9.5
C   2.0  3.5  5.5
C
DATA A / 1.0D0, 3.0D0, 2.0D0, 1.5D0, 2.5D0, 3.5D0,
$        4.5D0, 9.5D0, 5.5D0 /
C
PRINT 1000
PRINT 1010, ((A(I,J), J = 1, N), I = 1, N)
C
C   Find the largest element in column 1.
C
INCX = 1
LARGE = IDAMAX (N, A, INCX)
C
C   Swap the first row with the row containing the largest
C   element in column 1.
C
INCX = N
INCY = N
CALL DSWAP (N, A(1,1), INCX, A(LARGE,1), INCY)
PRINT 1020
PRINT 1010, ((A(I,J), J = 1, N), I = 1, N)
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (3(3(2X, F3.1) / ))
1020 FORMAT (1X, 'A':')
C
END

```

Sample Output

```

A:
1.0  1.5  4.5
3.0  2.5  9.5
2.0  3.5  5.5

```

A ¹ :		
3.0	2.5	9.5
1.0	1.5	4.5
2.0	3.5	5.5

Basic Linear Algebra Subprograms, Level 2 (BLAS2)



This chapter describes Basic Linear Algebra Subprograms Level 2 (BLAS2) as they are implemented in the Sun Performance Library. BLAS2 is a collection of linear algebra subprograms to perform common linear algebra operations with matrix and vector operands. For more information about BLAS2, refer to the information in this chapter or see “An Extended Set of FORTRAN Basic Linear Algebra Subprograms” written by Dongarra *et al* and published in the March 1988 *ACM Transactions on Mathematical Software* (Vol. 14, pp. 1-17).

The first four sections of this chapter describe:

- conventions followed in naming arguments and subroutines
- conventions used for passing arguments into and out of subroutines
- error handling done by the subroutines
- extensions to the standard versions of BLAS2

The remaining sections describe the operation of each of the BLAS2 subroutines. For each subroutine, the interface is shown, the operation of the subprogram is described, and a short FORTRAN 77 program is given with sample output.

Naming Conventions

The types of all arguments follow the notation in Dongarra's original paper as shown below:

IMPLICIT COMPLEX (C)

```

IMPLICIT DOUBLE PRECISION (D)
IMPLICIT INTEGER (I-N)
IMPLICIT REAL*16 (Q)
IMPLICIT REAL (S)
IMPLICIT DOUBLE COMPLEX (Z)
CHARACTER UPLO, TRANSA, DIAG

```

The first letter of a subroutine name identifies the type of the result. The next two letters indicate the form of matrix involved. The possible forms are:

GB	General in banded storage
GE	General
HB	Hermitian in banded storage
HE	Hermitian
HP	Hermitian in packed storage
SB	Symmetric in banded storage
SP	Symmetric in packed storage
SY	Symmetric
TB	Triangular in banded storage
TP	Triangular in packed storage
TR	Triangular

The fourth and fifth letters indicate the operation that the subroutine performs:

MV	Matrix-vector product
R	Rank-one update
R2	Rank-two update
SV	Solve a triangular system of linear equations

To specify the group of subroutines that perform the same operation but are of different types, this manual uses a lowercase x. For example, xGEMV refers to the subroutines DGEMV, SGEMV, ZGEMV, and CGEMV. This same convention is used for arguments: xA refers to the arguments DA, SA, ZA, and CA.

A^{-1} denotes the inverse of the square matrix stored in the array A. Whenever the symbol A^{-1} appears it is assumed that the numerical inverse of A exists. A^T denotes the transpose of A and A^H denotes the conjugate transpose of A for a complex A.

Argument Conventions

Only the first character of each character argument is used. This means that you can specify for a character argument any string that begins with a legal character. For example, many of the sample programs pass the string 'UNIT DIAGONAL A' as the UNIT argument because it is easier to understand than simply 'U'.

Although the FORTRAN 77 standard prohibits aliasing arguments, the Sun Performance Library allows the programmer to alias two or more arguments as long as all aliased arguments have the same base type and one of the arguments being aliased is not the result array. A real variable and a complex variable have the same base type (real), while a real variable and a double precision variable have different base types. Aliasing arguments having different base types or aliasing any argument with the result array will produce undefined results at best. It is not even guaranteed that these undefined results will be the same from one run to the next.

Error Handling

If the programmer calls a BLAS2 subroutine with an illegal argument value then the subroutine will call the error handling subroutine XERBLA. XERBLA will print the name of the subroutine, the number of the first argument found to be in error, and the values of the scalar arguments. The values of the arguments with the same types are printed together. As an example, consider the following call to DGBMV:

```
CALL DGBMV ('ILLEGAL', -1, 3, 1, 1, 2.0D0, A, LDA, X, 1, 3.0D0, Y, 1)
```

In the call above, the value of TRANSA is illegal. DGBMV will call XERBLA, which will print the report shown below and then terminate with a STOP statement.

```

** On entry to DGBMV  parameter number  1 had an illegal value

CHARACTER PARAMETERS:
  I

INTEGER PARAMETERS:
    -1      3      1      1      3      1
     1

REAL PARAMETERS:
    0.2000E+01    0.3000E+01

STOP

```

The Sun Performance Library will detect all instances in which the calling subprogram specifies one or more illegal argument values, but there are many types of interface errors that the Sun Performance Library will not detect. Included in the list of interface errors that the Sun Performance Library will not detect are too many or too few arguments, an argument of the wrong type, and an argument aliased with the result argument.

Extensions to the Basic Linear Algebra Subprograms

The following extensions to BLAS2 are implemented in the Sun Performance Library:

1. The Sun Performance Library includes the double complex subprograms suggested in the original paper by Dongarra.
2. Lowercase letters are allowed as values for the arguments DIAG, TRANSA, UNIT, and UPLO.
3. This version of Sun Performance Library contains a native C/C++ interface to BLAS. Conventions for this interface are described in detail in the preface to this manual.

Product of a Matrix in Banded Storage and a Vector

The subroutines described in this section compute one of the following results for a matrix A in banded storage and a vector x:

$$y \leftarrow \alpha Ax + \beta y$$

$$y \leftarrow \alpha A^T x + \beta y$$

$$y \leftarrow \alpha \bar{A}^T x + \beta y$$

Calling Sequence

CALL DGBMV	(TRANSA, M, N, NSUB, NSUPER, DALPHA, DA, LDA, DX, INCX, DBETA, DY, INCY)
CALL SGBMV	(TRANSA, M, N, NSUB, NSUPER, SALPHA, SA, LDA, SX, INCX, SBETA, SY, INCY)
CALL ZGBMV	(TRANSA, M, N, NSUB, NSUPER, ZALPHA, ZA, LDA, ZX, INCX, ZBETA, ZY, INCY)
CALL CGBMV	(TRANSA, M, N, NSUB, NSUPER, CALPHA, CA, LDA, CX, INCX, CBETA, CY, INCY)
void dgbmv	(char trans, int m, int n, int nsub, int nsuper, double dalpha, double *da, int lda, double *dx, int incx, double dbeta, double *dy, int incy)
void sgbmv	(char trans, int m, int n, int nsub, int nsuper, float salpha, float *sa, int lda, float *sx, int incx, float sbeta, float *sy, int incy)
void zgbmv	(char trans, int m, int n, int nsub, int nsuper, doublecomplex *zalpha, doublecomplex *za, int lda, doublecomplex *zx, int incx, doublecomplex *zbeta, doublecomplex *zy, int incy)
void cgbmv	(char trans, int m, int n, int nsub, int nsuper, complex *calpha, complex *ca, int lda, complex *cx, int incx, complex *cbeta, complex *cy, int incy)

Arguments

TRANSA	Indicates how to use the matrix stored in the array argument. The legal values for TRANSA are listed below. Any value not listed below is illegal.
'N' or 'n'	Use the matrix as it is stored in the array.
'T' or 't'	Use the transpose of the matrix.
'C' or 'c'	Use the conjugate transpose of the matrix.

M	Number of rows in the matrix A. $M \geq 0$.
N	Number of columns in the matrix A. $N \geq 0$.
NSUB	Number of subdiagonals of A. $N-1 \geq \text{NSUB} \geq 0$ but if $N = 0$ then $\text{NSUB} = 0$.
NSUPER	Number of superdiagonals of A. $N-1 \geq \text{NSUPER} \geq 0$ but if $N = 0$ then $\text{NSUPER} = 0$.
xALPHA	Scalar that scales the input value of the matrix A.
xA	Two-dimensional array that contains the input matrix.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $\text{LDA} \geq \text{NSUB} + 1 + \text{NSUPER}$.
xX	X and INCX describe a vector of length N if $\text{TRANSA} = 'N'$ or $'n'$, otherwise of length M. X contains an input vector.
INCX	Scalar that contains the storage spacing between successive elements of the vector. $\text{INCX} \neq 0$. If $\text{INCX} = 1$, then elements of the vector are contiguous in memory. INCX may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations. If X is a one-dimensional array and $\text{INCX} = -1$ then the array will be accessed in reverse order. If X is a two-dimensional array and $\text{INCX} = \text{LDA}$ then the vector will be a row of the array. If X is a two-dimensional array and $\text{INCX} = \text{LDA}+1$ then the vector will be a diagonal of the array.
xBETA	Scalar that scales the input value of the vector Y.
xY	Y and INCY describe a vector of length M if $\text{TRANSA} = 'N'$ or $'n'$, otherwise of length N. On entry, an input vector. On exit, a result vector.
INCY	Scalar that contains the storage spacing between successive elements of the vector Y. $\text{INCY} \neq 0$. If $\text{INCY} = 1$, then elements of the vector are contiguous in memory. INCY may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations. If Y is a one-dimensional array and $\text{INCY} = -1$ then the array will be accessed in reverse order. If Y is a two-dimensional array and $\text{INCY} = \text{LDA}$ then the vector will be a row of the array. If Y is a two-dimensional array and $\text{INCY} = \text{LDA}+1$ then the vector will be a diagonal of the array.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER          LDA, M, N, NSUB, NSUPER
  PARAMETER        (M = 5)
  PARAMETER        (N = 5)
  PARAMETER        (NSUB = 0)
  PARAMETER        (NSUPER = 2)
  PARAMETER        (LDA = NSUB + 1 + NSUPER)

C
  INTEGER          I, J
  DOUBLE PRECISION A(LDA,N), ALPHA, BETA, X(N), Y(N)

C
  EXTERNAL DGBMV
  INTRINSIC MIN

C
C Initialize the array A to store in banded form the matrix A
C with two superdiagonals and no subdiagonals shown below.
C Initialize the arrays X and Y to store the vectors x and y
C shown below.
C
C      1  1  1          1          1
C      2  2  2          2          1
C      3  3  3          3          1
C      4  4          4          1
C      5          5          1
C
C      A =          x = 3      y = 1
C
C
C      DATA A / 8.0D8, 8.0D8, 1.0D0, 8.0D8, 1.0D0, 2.0D0,
C $              1.0D0, 2.0D0, 3.0D0, 2.0D0, 3.0D0, 4.0D0,
C $              3.0D0, 4.0D0, 5.0D0 /
C      DATA X / 1.0D0, 2.0D0, 3.0D0, 4.0D0, 5.0D0 /
C      DATA Y / 1.0D0, 1.0D0, 1.0D0, 1.0D0, 1.0D0 /

C
  PRINT 1000
  DO 10, I = 1, N
    PRINT 1010, (0.0D0, J = 1, I - 1),
C $      (A(3 - J, J + I), J = 0, MIN(5 - I, 2)),
C $      (0.0D0, J = 1, 3 - I)
10 CONTINUE
  PRINT 1020
  DO 20, I = 1, 3
    PRINT 1010, (A(I,J), J = 1, N)
20 CONTINUE
  PRINT 1030
  PRINT 1040, (X(I), Y(I), I = 1, N)
  ALPHA = 2.0D0
  BETA = 3.0D0
  CALL DGBMV ('NO TRANSPOSE A', M, N, NSUB, NSUPER, ALPHA,
C $      A, LDA, X, 1, BETA, Y, 1)
  PRINT 1050
  PRINT 1060, Y
C

```

```

1000 FORMAT (/1X, 'Array A in full form:')
1010 FORMAT (1X, 5(2X, F4.1))
1020 FORMAT (/1X, 'Array A in packed form: ',
$      ' (* in unused fields)')
1030 FORMAT (/5X, 'x', 5X, 'y')
1040 FORMAT (3X, F4.1, 2X, F4.1)
1050 FORMAT (/1X, 'Ax + y:')
1060 FORMAT (3X, F4.1)
C
      END

```

Sample Output

```

Array A in full form:
  1.0  1.0  1.0  0.0  0.0
  0.0  2.0  2.0  2.0  0.0
  0.0  0.0  3.0  3.0  3.0
  0.0  0.0  0.0  4.0  4.0
  0.0  0.0  0.0  0.0  5.0

Array A in packed form: (* in unused fields)
****  ****  1.0  2.0  3.0
****  1.0  2.0  3.0  4.0
  1.0  2.0  3.0  4.0  5.0

      x      y
  1.0  1.0
  2.0  1.0
  3.0  1.0
  4.0  1.0
  5.0  1.0

Ax + y:
 15.0
 39.0
 75.0
 75.0
 53.0

```

Product of a General Matrix and a Vector

The subroutines described in this section compute one of the following results for a general matrix A and vectors x and y:

$$y \leftarrow \alpha Ax + \beta y$$

$$y \leftarrow \alpha A^T x + \beta y$$

$$y \leftarrow \alpha \bar{A}^T x + \beta y$$

Calling Sequence

CALL DGEMV	(TRANSA, M, N, DALPHA, DA, LDA, DX, INCX, DBETA, DY, INCY)
CALL SGEMV	(TRANSA, M, N, SALPHA, SA, LDA, SX, INCX, SBETA, SY, INCY)
CALL ZGEMV	(TRANSA, M, N, ZALPHA, ZA, LDA, ZX, INCX, ZBETA, ZY, INCY)
CALL CGEMV	(TRANSA, M, N, CALPHA, CA, LDA, CX, INCX, CBETA, CY, INCY)
void dgemv	(char trans, int m, int n, double dalpha, double *da, int lda, double *dx, int incx, double dbeta, double *dy, int incy)
void sgemv	(char trans, int m, int n, float salpha, float *sa, int lda, float *sx, int incx, float sbeta, float *sy, int incy)
void zgemv	(char trans, int m, int n, doublecomplex *zalpha, doublecomplex *za, int lda, doublecomplex *zx, int incx, doublecomplex *zbeta, doublecomplex *zy, int incy)
void cgemv	(char trans, int m, int n, complex *calpha, complex *ca, int lda, complex *cx, int incx, complex *cbeta, complex *cy, int incy)

Arguments

TRANSA	Indicates how to use the matrix stored in the array argument. The legal values for TRANSA are listed below. Any value not listed below is illegal.
'N' or 'n'	Use the matrix as it is stored in the array.
'T' or 't'	Use the transpose of the matrix.
'C' or 'c'	Use the conjugate transpose of the matrix.

M	Number of rows in the matrix A. $M \geq 0$.
N	Number of columns in the matrix A. $N \geq 0$.
xALPHA	Scalar that scales the input value of the matrix A.
xA	Two-dimensional array that contains the input matrix.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1,M)$.
xX	X and INCX describe a vector of length N if TRANSA = 'N' or 'n', otherwise of length M. X contains an input vector.
INCX	Scalar that contains the storage spacing between successive elements of the vector. $INCX \neq 0$. If $INCX = 1$, then elements of the vector are contiguous in memory. INCX may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations. If X is a one-dimensional array and $INCX = -1$ then the array will be accessed in reverse order. If X is a two-dimensional array and $INCX = LDA$ then the vector will be a row of the array. If X is a two-dimensional array and $INCX = LDA+1$ then the vector will be a diagonal of the array.
xBETA	Scalar that scales the input value of the vector Y.
xY	Y and INCY describe a vector of length M if TRANSA = 'N' or 'n', otherwise of length N. On entry, an input vector. On exit, a result vector.
INCY	Scalar that contains the storage spacing between successive elements of the vector Y. $INCY \neq 0$. If $INCY = 1$, then elements of the vector are contiguous in memory. INCY may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations. If Y is a one-dimensional array and $INCY = -1$ then the array will be accessed in reverse order. If Y is a two-dimensional array and $INCY = LDA$ then the vector will be a row of the array. If Y is a two-dimensional array and $INCY = LDA+1$ then the vector will be a diagonal of the array.

Sample Program

PROGRAM TEST		
IMPLICIT NONE		
C	INTEGER	LDRROT, M, N

```

      PARAMETER      (M = 2)
      PARAMETER      (LDROT = M)
      PARAMETER      (N = 2)
C
      DOUBLE PRECISION ALPHA, PI, ROT(LDROT,N), SCALE, THETA
      DOUBLE PRECISION VEC1(N), VEC2(N)
      INTEGER          INCX, INCY, M, N
C
      EXTERNAL          DGEMV
      INTRINSIC          ATAN, COS, SIN
C
      PI = 4.0D0 * ATAN (1.0D0)
      THETA = PI / 2.0D0
C
C      Initialize the array ROT to contain the rotation matrix to
C      be applied to the vector stored in VEC1.
C
      ROT(1,1) = COS (THETA)
      ROT(2,1) = SIN (THETA)
      ROT(1,2) = -SIN (THETA)
      ROT(2,2) = COS (THETA)
      VEC1(1) = 3.0D0
      VEC1(2) = 4.0D0
      PRINT 1000
      PRINT 1010, VEC1
      SCALE = 5.0D-1
      INCX = 1
      ALPHA = 0.0D0
      INCY = 1
      CALL DGEMV ('NOT TRANSPOSED ROT', M, N, SCALE, ROT, LDROT,
$              VEC1, INCX, ALPHA, VEC2, INCY)
      PRINT 1020
      PRINT 1010, VEC2
C
1000 FORMAT (1X, 'Original vector:')
1010 FORMAT (3X, F4.1)
1020 FORMAT (/1X, 'Rotated vector:')
C
      END

```

Sample Output

```

Original vector:
  3.0
  4.0

Rotated vector:
 -2.0
  1.5

```

Rank-1 Update to a General Matrix

The subroutines described in this section compute one of the following results for a general matrix A and vectors x and y.

DGER, SGER, ZGERU, and CGERU compute:

$$A \leftarrow \alpha xy^T + A$$

ZGERC and CGERC compute:

$$A \leftarrow \alpha x\bar{y}^T + A$$

Calling Sequence

```
CALL DGER      (M, N, DALPHA, DX, INCX, DY, INCY, DA, LDA)
CALL SGER      (M, N, SALPHA, SX, INCX, SY, INCY, SA, LDA)
CALL ZGERC     (M, N, ZALPHA, ZX, INCX, ZY, INCY, ZA, LDA)
CALL ZGERU     (M, N, ZALPHA, ZX, INCX, ZY, INCY, ZA, LDA)
CALL CGERC     (M, N, CALPHA, CX, INCX, CY, INCY, CA, LDA)
CALL CGERU     (M, N, CALPHA, CX, INCX, CY, INCY, CA, LDA)

void dger      (int m, int n, double dalpha, double *dx, int incx,
               double *dy, int incy, double *da, int lda)
void sger      (int m, int n, float salpha, float *sx, int incx, float
               *sy, int incy, float *sa, int lda)
void zgerc     (int m, int n, doublecomplex *zalpha, doublecomplex
               *zx, int incx, doublecomplex *zy, int incy,
               doublecomplex *za, int lda)
void zgeru     (int m, int n, doublecomplex *zalpha, doublecomplex
               *zx, int incx, doublecomplex *zy, int incy,
               doublecomplex *za, int lda)
void cgerc     (int m, int n, complex *calpha, complex *cx, int incx,
               complex *cy, int incy, complex *ca, int lda)
void cgeru     (int m, int n, complex *calpha, complex *cx, int incx,
               complex *cy, int incy, complex *ca, int lda)
```

Arguments

M	Number of rows in the matrix A. $M \geq 0$.
N	Number of columns in the matrix A. $N \geq 0$.
xALPHA	Scalar that scales the input value of the matrix A.

xX	X and INCX describe a vector of length M. X contains an input vector.
INCX	Scalar that contains the storage spacing between successive elements of the vector. $INCX \neq 0$. If $INCX = 1$, then elements of the vector are contiguous in memory. INCX may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations. If X is a one-dimensional array and $INCX = -1$ then the array will be accessed in reverse order. If X is a two-dimensional array and $INCX = LDA$ then the vector will be a row of the array. If X is a two-dimensional array and $INCX = LDA+1$ then the vector will be a diagonal of the array.
xY	Y and INCY describe a vector of length N. Y contains an input vector.
INCY	Scalar that contains the storage spacing between successive elements of the vector Y. $INCY \neq 0$. If $INCY = 1$, then elements of the vector are contiguous in memory. INCY may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations. If Y is a one-dimensional array and $INCY = -1$ then the array will be accessed in reverse order. If Y is a two-dimensional array and $INCY = LDA$ then the vector will be a row of the array. If Y is a two-dimensional array and $INCY = LDA+1$ then the vector will be a diagonal of the array.
xA	Two-dimensional array. On entry, the input matrix. On exit, the result matrix.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1,M)$.

Sample Program

```
PROGRAM TEST
IMPLICIT NONE
C
C   INTEGER LDA, M, N
C   PARAMETER      (M = 4)
C   PARAMETER      (LDA = M)
C   PARAMETER      (N = 3)
C
C   DOUBLE PRECISION A(LDA,N), ALPHA, X(M), Y(N)
C   INTEGER          I, J
C
C   EXTERNAL          DGER
```

```

C      INTRINSIC      DBLE
C
C      Initialize the array A to store the general matrix A, and
C      initialize the arrays X and Y to store the vectors x and y
C      shown below:
C
C      100  100  100      x = 1      y = 1
C      A = 100  100  100      x = 2      y = 2
C      100  100  100      3      3
C      100  100  100      4
C
      DO 110, J = 1, N
        DO 100, I = 1, M
          A(I,J) = 1.0D2
100    CONTINUE
        Y(J) = DBLE (J)
110    CONTINUE
      DO 120, I = 1, M
        X(I) = DBLE (I)
120    CONTINUE
      PRINT 1000
      DO 130, I = 1, M
        PRINT 1010, (A(I,J), J = 1, N)
130    CONTINUE
      PRINT 1020
      DO 140, I = 1, N
        PRINT 1030, X(I), Y(I)
140    CONTINUE
      PRINT 1030, X(M)
      ALPHA = 1.0D0
      CALL DGER (M, N, ALPHA, X, 1, Y, 1, A, LDA)
      PRINT 1040
      DO 150, I = 1, M
        PRINT 1010, (A(I,J), J = 1, N)
150    CONTINUE
C
1000  FORMAT (1X, 'A:')
1010  FORMAT (1X, 3(2X, F5.1))
1020  FORMAT (/1X, '      x ', 2X, '      y ')
1030  FORMAT (1X, F5.1, 2X, F5.1)
1040  FORMAT (/1X, 'A + xy'':')
C
      END

```

Sample Output

```

A:
 100.0  100.0  100.0
 100.0  100.0  100.0
 100.0  100.0  100.0
 100.0  100.0  100.0

```

x	y		
1.0	1.0		
2.0	2.0		
3.0	3.0		
4.0			
A + xy':			
101.0	102.0	103.0	
102.0	104.0	106.0	
103.0	106.0	109.0	
104.0	108.0	112.0	

Product of a Hermitian Matrix in Banded Storage and a Vector

The subroutines described in this section compute the following result for a Hermitian matrix A in banded storage and vectors x and y:

$$y \leftarrow \alpha Ax + \beta y$$

Calling Sequence

```
CALL ZHBMV (UPLO, N, NDIAG, ZALPHA, ZA, LDA, ZX, INCX, ZBETA, ZY,
            INCY)
CALL CHBMV (UPLO, N, NDIAG, CALPHA, CA, LDA, CX, INCX, CBETA, CY,
            INCY)

void zhbmv (char uplo, int n, int ndiag, doublecomplex *zalpha,
            doublecomplex *za, int lda, doublecomplex *zx, int
            incx, doublecomplex *zbeta, doublecomplex *zy, int
            incy)
void chbm (char uplo, int n, int ndiag, complex *calpha, complex
            *ca, int lda, complex *cx, int incx, complex *cbeta,
            complex *cy, int incy)
```

Arguments

UPLO	Indicates whether the values in a matrix reside in the upper or lower triangle of the array in which the matrix is stored. The legal values for UPLO are listed below. Any value not listed below is illegal. 'L' or 'l' Only the lower triangle of the array will be referenced. 'U' or 'u' Only the upper triangle of the array will be referenced.
N	Size of a matrix with N rows and N columns. $N \geq 0$.
NDIAG	Bandwidth of a Hermitian matrix in banded storage with NDIAG superdiagonals and NDIAG subdiagonals. $N-1 \geq \text{NDIAG} \geq 0$ but if $N = 0$ then $\text{NDIAG} = 0$.
xALPHA	Scalar that scales the input value of the matrix A.
xA	Two-dimensional array that contains the input matrix.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $\text{LDA} \geq \text{NDIAG} + 1$.
xX	X and INCX describe a vector of length N. X contains an input vector.

INCX	Scalar that contains the storage spacing between successive elements of the vector. $INCX \neq 0$. If $INCX = 1$, then elements of the vector are contiguous in memory. $INCX$ may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations. If X is a one-dimensional array and $INCX = -1$ then the array will be accessed in reverse order. If X is a two-dimensional array and $INCX = LDA$ then the vector will be a row of the array. If X is a two-dimensional array and $INCX = LDA+1$ then the vector will be a diagonal of the array.
xBETA	Scalar that scales the input value of the vector Y .
xY	Y and $INCY$ describe a vector of length N. On entry, an input vector. On exit, a result vector.
INCY	Scalar that contains the storage spacing between successive elements of the vector Y. $INCY \neq 0$. If $INCY = 1$, then elements of the vector are contiguous in memory. $INCY$ may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations. If Y is a one-dimensional array and $INCY = -1$ then the array will be accessed in reverse order. If Y is a two-dimensional array and $INCY = LDA$ then the vector will be a row of the array. If Y is a two-dimensional array and $INCY = LDA+1$ then the vector will be a diagonal of the array.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
REAL      RZERO
COMPLEX   ZERO
INTEGER   LDA, N, NDIAG
PARAMETER (LDA = 3)
PARAMETER (N = 4)
PARAMETER (NDIAG = 2)
PARAMETER (RZERO = 0.0E0)
PARAMETER (ZERO = (0.0E0, 0.0E0))
C
INTEGER   I, J
COMPLEX   A(LDA,N), ALPHA, BETA, X(N), Y(N)
C
EXTERNAL  CHBMV

```

```

C      INTRINSIC  CMPLX, CONJG, REAL
C
C      Initialize the array A to store in Hermitian banded form the
C      matrix A with two diagonals shown below. Initialize the
C      arrays X and Y to store the vectors x and y shown below.
C
C      A =      1+0i  2+1i  3+2i      1+4i      1+1i
C      2-1i  4+0i  5+3i  6+4i      x = 2+3i      y = 1+2i
C      3-2i  5-3i  7+0i  8+5i      3+2i      1+3i
C      6-4i  8-5i  9+0i      4+1i      1+4i
C
C      DATA A /(8E8,8E8), (8E8,8E8), (1.0,8E8),
$      (8E8,8E8), (2.0,1.0), (4.0,8E8),
$      (3.0,2.0), (5.0,3.0), (7.0,8E8),
$      (6.0,4.0), (8.0,5.0), (9.0,8E8) /
C
C      DO 10, I = 1, N
C      X(I) = CMPLX (REAL (I), REAL (5-I))
C      Y(I) = CMPLX (1.0, REAL (I))
10 CONTINUE
C      ALPHA = CMPLX (1.0,0.0)
C      BETA = CMPLX (1.0, 0.0)
C      PRINT 1000
C      PRINT 1010, REAL (A(3,1)), RZERO, A(2,2), A(1,3), ZERO
C      PRINT 1010, CONJG (A(2,2)), REAL (A(3,2)), RZERO, A(2,3),
$      A(1,4)
C      PRINT 1010, CONJG (A(1,3)), CONJG (A(2,3)),
$      REAL (A(3,3)), RZERO, A(2,4)
C      PRINT 1010, ZERO, CONJG (A(1,4)), CONJG (A(2,4)),
$      REAL (A(3,4)), RZERO
C      PRINT 1020
C      DO 20, I = 1, LDA
C      PRINT 1010, (A(I,J), J = 1, N)
20 CONTINUE
C      PRINT 1030
C      PRINT 1040, (X(I), Y(I), I = 1, N)
C      CALL CHBMV ('UPPER TRIANGULAR A', N, NDIAG, ALPHA, A, LDA,
$      X, 1, BETA, Y, 1)
C      PRINT 1050
C      PRINT 1060, (Y(I), I = 1, N)
C
C      1000 FORMAT (/1X, 'Array A in full form:')
C      1010 FORMAT (1X, 4(2X, '(', F5.1, ', ', F5.1, ')'))
C      1020 FORMAT (/1X, 'Array A in banded form: ',
$      '(* in unused elements)')
C      1030 FORMAT (/10X, 'x', 15X, 'y')
C      1040 FORMAT (2(3X, '(', F5.1, ', ', F5.1, ')'))
C      1050 FORMAT (/1X, 'Ax + y:')
C      1060 FORMAT (1X, 2X, '(', F5.1, ', ', F5.1, ')')
C
C      END

```

Sample Output

```
Array A in full form:
( 1.0, 0.0) ( 2.0, 1.0) ( 3.0, 2.0) ( 0.0, 0.0)
( 2.0, -1.0) ( 4.0, 0.0) ( 5.0, 3.0) ( 6.0, 4.0)
( 3.0, -2.0) ( 5.0, -3.0) ( 7.0, 0.0) ( 8.0, 5.0)
( 0.0, 0.0) ( 6.0, -4.0) ( 8.0, -5.0) ( 9.0, 0.0)

Array A in banded form: (* in unused elements)
(*****,*****) (*****,*****) ( 3.0, 2.0) ( 6.0, 4.0)
(*****,*****) ( 2.0, 1.0) ( 5.0, 3.0) ( 8.0, 5.0)
( 1.0,*****) ( 4.0,*****) ( 7.0,*****) ( 9.0,*****)

      x      y
( 1.0, 4.0) ( 1.0, 1.0)
( 2.0, 3.0) ( 1.0, 2.0)
( 3.0, 2.0) ( 1.0, 3.0)
( 4.0, 1.0) ( 1.0, 4.0)

Ax + y:
( 8.0, 25.0)
( 44.0, 62.0)
( 79.0, 64.0)
( 95.0, 24.0)
```

Product of a Hermitian Matrix and a Vector

The subroutines described in this section compute the following result for a Hermitian matrix A and vectors x and y:

$$y \leftarrow \alpha Ax + \beta y$$

Calling Sequence

```
CALL ZHEMV (UPLO, N, ZALPHA, ZA, LDA, ZX, INCX, ZBETA, ZY, INCY)
CALL CHEMV (UPLO, N, CALPHA, CA, LDA, CX, INCX, CBETA, CY, INCY)

void zhemv (char uplo, int n, doublecomplex *zalpha,
doublecomplex *za, int lda, doublecomplex *zx, int
incx, doublecomplex *zbeta, doublecomplex *zy, int
incy)

void chemv (char uplo, int n, complex *calpha, complex *ca, int
lda, complex *cx, int incx, complex *cbeta, complex
*cy, int incy)
```

Arguments

UPLO	Indicates whether the values in a matrix reside in the upper or lower triangle of the array in which the matrix is stored. The legal values for UPLO are listed below. Any value not listed below is illegal. 'L' or 'l' Only the lower triangle of the array will be referenced. 'U' or 'u' Only the upper triangle of the array will be referenced.
N	Size of a matrix with N rows and N columns. $N \geq 0$.
xALPHA	Scalar that scales the input value of the matrix A.
xA	Two-dimensional array that contains the input matrix.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
xX	X and INCX describe a vector of length N. X contains an input vector.
INCX	Scalar that contains the storage spacing between successive elements of the vector. $INCX \neq 0$. If $INCX = 1$, then elements of the vector are contiguous in memory. INCX may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations. If X is a one-dimensional array and $INCX = -1$ then the array will be accessed in reverse order. If X is a two-dimensional array and $INCX = LDA$ then the vector will be a row of the array.

	If X is a two-dimensional array and $INCX = LDA+1$ then the vector will be a diagonal of the array.
xBETA	Scalar that scales the input value of the vector Y.
xY	Y and INCY describe a vector of length N. On entry, an input vector. On exit, a result vector.
INCY	Scalar that contains the storage spacing between successive elements of the vector Y. $INCY \neq 0$. If $INCY = 1$, then elements of the vector are contiguous in memory. INCY may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations. If Y is a one-dimensional array and $INCY = -1$ then the array will be accessed in reverse order. If Y is a two-dimensional array and $INCY = LDA$ then the vector will be a row of the array. If Y is a two-dimensional array and $INCY = LDA+1$ then the vector will be a diagonal of the array.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C      REAL      RZERO
C      INTEGER    LDA, N
C      PARAMETER (N = 4)
C      PARAMETER (LDA = N)
C      PARAMETER (RZERO = 0.0E0)
C
C      COMPLEX    A(LDA,N), ALPHA, BETA, X(N), Y(N)
C      INTEGER    I, J
C
C      EXTERNAL   CHEMV
C      INTRINSIC  CMPLX, CONJG, REAL
C
C      Initialize the array A to store in Hermitian form the matrix
C      A shown below. Initialize the arrays x and y to store the
C      vectors x and y shown below.
C
C      1+0i  2+3i  3+4i  4+5i      10+20i      -10-2i
C      A = 2-3i  5+0i  6+7i  7+8i      x =  1+ 2i      y = -20-3i
C      3-4i  6-7i  3+0i  9+0i      .1+.2i      -30-4i
C      4-5i  7-8i  9+0i  1+0i      .01+.02i      -40-5i
C
C      DATA A / (1.0,8E8), (8E8,8E8), (8E8,8E8), (8E8,8E8),
C      $          (2.0,3.0), (5.0,8E8), (8E8,8E8), (8E8,8E8),
C      $          (3.0,4.0), (6.0,7.0), (8.0,8E8), (8E8,8E8),

```

```

$      (4.0,5.0), (7.0,8.0), (9.0,0.0), (1.0,8E8) /
DATA X / (10.0,20.0), (1.0,2.0), (0.1,0.2), (0.01,0.02) /
DATA Y / (-10.0,-2.0), (-20.0,-3.0), (-30.0,-4.0),
$      (-40.0,-5.0) /
C
  PRINT 1000
  DO 100, I = 1, N
    PRINT 1010, (A(J,I), J = 1, I - 1), REAL(A(I,I)), RZERO,
$      (CONJG(A(I,J)), J = I + 1, N)
100 CONTINUE
  PRINT 1020
  DO 110, I = 1, N
    PRINT 1010, (A(I,J), J = 1, N)
110 CONTINUE
  PRINT 1030
  PRINT 1040, (X(I), Y(I), I = 1, N)
  ALPHA = CMPLX (2.0,3.0)
  BETA = CMPLX (1.0,-1.0)
  CALL CHEMV ('UPPER TRIANGULAR A', N, ALPHA, A, LDA, X, 1,
$      BETA, Y, 1)
  PRINT 1050
  PRINT 1060, (Y(I), I = 1, N)
C
1000 FORMAT (1X, 'Matrix A in full form:')
1010 FORMAT (1X, 4(2X, '(', F6.2, ', ', F6.2, ')'))
1020 FORMAT (/1X, 'Matrix A in Hermitian form: ',
$      '(* in unused elements)')
1030 FORMAT (/10X, 'x', 17X, 'y')
1040 FORMAT (2(3X, '(', F6.2, ', ', F6.2, ')'))
1050 FORMAT (/1X, 'Ax + y:')
1060 FORMAT (3X, '(', F6.1, ', ', F6.1, ')')
C
  END

```

Sample Output

```

Matrix A in full form:
( 1.00, 0.00) ( 2.00, -3.00) ( 3.00, -4.00) ( 4.00, -5.00)
( 2.00, 3.00) ( 5.00, 0.00) ( 6.00, -7.00) ( 7.00, -8.00)
( 3.00, 4.00) ( 6.00, 7.00) ( 8.00, 0.00) ( 9.00, 0.00)
( 4.00, 5.00) ( 7.00, 8.00) ( 9.00, 0.00) ( 1.00, 0.00)

Matrix A in Hermitian form: (* in unused elements)
( 1.00, *****) ( 2.00, 3.00) ( 3.00, 4.00) ( 4.00, 5.00)
( *****, *****) ( 5.00, *****) ( 6.00, 7.00) ( 7.00, 8.00)
( *****, *****) ( *****, *****) ( 8.00, *****) ( 9.00, 0.00)
( *****, *****) ( *****, *****) ( *****, *****) ( 1.00, *****)

      x      y
( 10.00, 20.00) (-10.00, -2.00)
( 1.00, 2.00) (-20.00, -3.00)
( 0.10, 0.20) (-30.00, -4.00)

```

```
( 0.01, 0.02) (-40.00, -5.00)
Ax + y:
( -85.5, 80.6)
( 78.9, 313.6)
( 147.4, 472.2)
( 169.4, 602.4)
```

Rank-1 Update to a Hermitian Matrix

The subroutines described in this section compute the following result for a Hermitian matrix A and a vector x:

$$A \leftarrow \alpha x \bar{x}^T + A$$

Calling Sequence

```
CALL ZHER    (UPLO, N, DALPHA, ZX, INCX, ZA, LDA)
CALL CHER    (UPLO, N, SALPHA, CX, INCX, CA, LDA)

void zher    (char uplo, int n, double dalpha, doublecomplex *zx,
              int incx, doublecomplex *za, int lda)
void cher    (char uplo, int n, float salpha, complex *cx, int incx,
              complex *ca, int lda)
```

Note – The type of the scalar xALPHA is real.

Arguments

UPLO	Indicates whether the values in a matrix reside in the upper or lower triangle of the array in which the matrix is stored. The legal values for UPLO are listed below. Any value not listed below is illegal. 'L' or 'l' Only the lower triangle of the array will be referenced. 'U' or 'u' Only the upper triangle of the array will be referenced.
N	Size of a matrix with N rows and N columns. $N \geq 0$.
xALPHA	Scalar that scales the input value of the matrix A.
xX	X and INCX describe a vector of length N. X contains an input vector.
INCX	Scalar that contains the storage spacing between successive elements of the vector. $INCX \neq 0$. If $INCX = 1$, then elements of the vector are contiguous in memory. INCX may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations. If X is a one-dimensional array and $INCX = -1$ then the array will be accessed in reverse order. If X is a two-dimensional array and $INCX = LDA$ then the vector will be a row of the array. If X is a two-dimensional array and $INCX = LDA+1$ then the vector will be a diagonal of the array.

xA	Two-dimensional array. On entry, the input matrix. On exit, the result matrix.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
REAL      RZERO
COMPLEX   ONE
INTEGER   LDA, N
PARAMETER (N = 3)
PARAMETER (LDA = N)
PARAMETER (ONE = (1.0E0, 0.0E0))
PARAMETER (RZERO = 0.0E0)
C
COMPLEX   A(LDA, N), ONE, X(N)
INTEGER   I, J
C
EXTERNAL  CHER
INTRINSIC CONJG, REAL
C
C Initialize the array A to store in Hermitian form the matrix
C A shown below. Initialize the array X to store the vector X
C shown below.
C
C      1+0i  2+2i  3+3i      1-1i
C  A = 2-2i  2+0i  3+3i      x = 2-2i
C      3-3i  3-3i  3+0i      3-3i
C
DATA A / (1.0, 8E8), (8E8, 8E8), (8E8, 8E8),
$        (2.0, 2.0), (2.0, 8E8), (8E8, 8E8),
$        (3.0, 3.0), (3.0, 3.0), (3.0, 8E8) /
DATA X / (1.0, -1.0), (2.0, -2.0), (3.0, -3.0) /
C
PRINT 1000
DO 100, I = 1, N
  PRINT 1010, (A(J, I), J = 1, I - 1), REAL(A(I, I)), RZERO,
$    (CONJG(A(I, J)), J = I + 1, N)
100 CONTINUE
PRINT 1020
DO 110, I = 1, N
  PRINT 1010, (A(I, J), J = 1, N)
110 CONTINUE
PRINT 1030
DO 120, I = 1, N
  PRINT 1040, X(I)
120 CONTINUE

```

```

      CALL CHER ('UPPER TRIANGULAR A', N, ONE, X, 1, A, LDA)
      PRINT 1050
      DO 130, I = 1, N
        PRINT 1010, (A(J,I), J = 1, I - 1), REAL(A(I,I)), RZERO,
$          (CONJG(A(I,J)), J = I + 1, N)
      130 CONTINUE
      PRINT 1060
      DO 140, I = 1, N
        PRINT 1010, (A(I,J), J = 1, N)
      140 CONTINUE
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (1X, 3(2X, '(', F5.1, ', ', F5.1, ')'))
1020 FORMAT (/1X, 'A in Hermitian form: ',
$          '(* in unused elements)')
1030 FORMAT (/1X, 'x:')
1040 FORMAT (3X, '(', F5.1, ', ', F5.1, ')')
1050 FORMAT (/1X, 'A + xx' in full form:')
1060 FORMAT (/1X, 'A + xx' in Hermitian form: ',
$          '(* in unused elements)')
C
      END

```

Sample Output

```

A in full form:
(  1.0,  0.0) (  2.0, -2.0) (  3.0, -3.0)
(  2.0,  2.0) (  2.0,  0.0) (  3.0, -3.0)
(  3.0,  3.0) (  3.0,  3.0) (  3.0,  0.0)

A in Hermitian form: (* in unused elements)
(  1.0, *****) (  2.0,  2.0) (  3.0,  3.0)
(*****, *****) (  2.0, *****) (  3.0,  3.0)
(*****, *****) (*****, *****) (  3.0, *****)

x:
(  1.0, -1.0)
(  2.0, -2.0)
(  3.0, -3.0)

A + xx' in full form:
(  3.0,  0.0) (  6.0, -2.0) (  9.0, -3.0)
(  6.0,  2.0) ( 10.0,  0.0) ( 15.0, -3.0)
(  9.0,  3.0) ( 15.0,  3.0) ( 21.0,  0.0)

A + xx' in Hermitian form: (* in unused elements)
(  3.0,  0.0) (  6.0,  2.0) (  9.0,  3.0)
(*****, *****) ( 10.0,  0.0) ( 15.0,  3.0)
(*****, *****) (*****, *****) ( 21.0,  0.0)

```

Rank-2 Update to a Hermitian Matrix

The subroutines described in this section compute the following result for the Hermitian matrix A and vectors x and y:

$$A \leftarrow \alpha \bar{x}y^T + \bar{\alpha} y\bar{x}^T + A$$

Calling Sequence

```
CALL ZHER2 (UPLO, N, ZALPHA, ZX, INCX, ZY, INCY, ZA, LDA)
CALL CHER2 (UPLO, N, CALPHA, CX, INCX, CY, INCY, CA, LDA)

void zher2 (char uplo, int n, doublecomplex *zalpha, doublecomplex
            *zx, int incx, doublecomplex *zy, int incy,
            doublecomplex *za, int lda)
void cher2 (char uplo, int n, complex *calpha, complex *cx, int
            incx, complex *cy, int incy, complex *ca, int lda)
```

Arguments

UPLO	Indicates whether the values in a matrix reside in the upper or lower triangle of the array in which the matrix is stored. The legal values for UPLO are listed below. Any value not listed below is illegal. 'L' or 'l' Only the lower triangle of the array will be referenced. 'U' or 'u' Only the upper triangle of the array will be referenced.
N	Size of a matrix with N rows and N columns. $N \geq 0$.
xALPHA	Scalar that scales the input value of the matrix A.
xX	X and INCX describe a vector of length N. X contains an input vector.
INCX	Scalar that contains the storage spacing between successive elements of the vector. $INCX \neq 0$. If $INCX = 1$, then elements of the vector are contiguous in memory. INCX may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations. If X is a one-dimensional array and $INCX = -1$ then the array will be accessed in reverse order. If X is a two-dimensional array and $INCX = LDA$ then the vector will be a row of the array. If X is a two-dimensional array and $INCX = LDA+1$ then the vector will be a diagonal of the array.
xY	Y and INCY describe a vector of length N. Y contains an input vector.

INCY	Scalar that contains the storage spacing between successive elements of the vector Y. $INCY \neq 0$. If $INCY = 1$, then elements of the vector are contiguous in memory. INCY may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations. If Y is a one-dimensional array and $INCY = -1$ then the array will be accessed in reverse order. If Y is a two-dimensional array and $INCY = LDA$ then the vector will be a row of the array. If Y is a two-dimensional array and $INCY = LDA+1$ then the vector will be a diagonal of the array.
xA	Two-dimensional array. On entry, the input matrix. On exit, the result matrix.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1,N)$.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C      REAL      RZERO
C      INTEGER   LDA, N
C      PARAMETER (N = 3)
C      PARAMETER (LDA = N)
C      PARAMETER (RZERO = 0.0E0)
C
C      COMPLEX   A(LDA,N), X(N), Y(N)
C      INTEGER   I, J
C
C      EXTERNAL  CHER2
C      INTRINSIC CONJG, REAL
C
C      Initialize the array A to store in symmetric form the
C      Hermitian matrix A shown below. Initialize the arrays
C      X and Y to store the vectors x and y shown below.
C
C      A =      1+0i  2+3i  5+6i      x =      1-1i      4-4i
C      2-3i  4+0i  7+8i      y =      2-2i      4-4i
C      5-6i  7-8i  9+0i      3-3i      4-4i
C
C      DATA A / (1.0,8E8), (8E8,8E8), (8E8,8E8),
C      $          (2.0,3.0), (4.0,8E8), (8E8,8E8),
C      $          (5.0,6.0), (7.0,8.0), (9.0,8E8) /
C      DATA X / (1.0,-1.0), (2.0,-2.0), (3.0,-3.0) /
C      DATA Y / (4.0,-4.0), (4.0,-4.0), (4.0,-4.0) /
C

```



```

      PRINT 1000
      DO 100, I = 1, N
        PRINT 1010, (A(J,I), J = 1, I - 1), REAL(A(I,I)), RZERO,
$      (CONJG(A(I,J)), J = I + 1, N)
100  CONTINUE
      PRINT 1020
      DO 110, I = 1, N
        PRINT 1010, (A(I,J), J = 1, N)
110  CONTINUE
      PRINT 1030
      DO 120, I = 1, N
        PRINT 1040, X(I), Y(I)
120  CONTINUE
      CALL CHER2 ('UPPER TRIANGULAR A', 3, 1.0D0, X, 1, Y, 1,
$      A, 3)
      PRINT 1050
      DO 130, I = 1, N
        PRINT 1010, (A(J,I), J = 1, I - 1), REAL(A(I,I)), RZERO,
$      (CONJG(A(I,J)), J = I + 1, N)
130  CONTINUE
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (1X, 3(2X, '(', F5.1, ', ', F5.1, ')'))
1020 FORMAT (/1X, 'A in Hermitian form: (* in unused elements)')
1030 FORMAT (/10X, 'x ', 13X, 'y')
1040 FORMAT (1X, 2(2X, '(', F5.1, ', ', F5.1, ')'))
1050 FORMAT (/1X, 'A + xy' + yx')
C
      END

```

Sample Output

```

A in full form:
( 1.0, 0.0) ( 2.0, -3.0) ( 5.0, -6.0)
( 2.0, 3.0) ( 4.0, 0.0) ( 7.0, -8.0)
( 5.0, 6.0) ( 7.0, 8.0) ( 9.0, 0.0)

A in Hermitian form: (* in unused elements)
( 1.0, *****) ( 2.0, 3.0) ( 5.0, 6.0)
( *****, *****) ( 4.0, *****) ( 7.0, 8.0)
( *****, *****) ( *****, *****) ( 9.0, *****)

      x      y
( 1.0, -1.0) ( 4.0, -4.0)
( 2.0, -2.0) ( 4.0, -4.0)
( 3.0, -3.0) ( 4.0, -4.0)

A + xy' + yx'
( 31.0, 0.0) ( 47.0, -3.0) ( 65.0, -6.0)
( 47.0, 3.0) ( 64.0, 0.0) ( 82.0, -8.0)
( 65.0, 6.0) ( 82.0, 8.0) ( 99.0, 0.0)

```

Product of a Hermitian Matrix in Packed Storage and a Vector

The subroutines described in this section compute the following result for a Hermitian matrix A in packed storage and vectors x and y:

$$y \leftarrow \alpha Ax + \beta y$$

Calling Sequence

```
CALL ZHPMV (UPLO, N, ZALPHA, ZA, ZX, INCX, ZBETA, ZY, INCY)
CALL CHPMV (UPLO, N, CALPHA, CA, CX, INCX, CBETA, CY, INCY)

void zhpvm (char uplo, int n, doublecomplex *zalp,
doublecomplex *za, doublecomplex *zx, int incx,
doublecomplex *zbet, doublecomplex *zy, int incy)
void chpmv (char uplo, int n, complex *calp, complex *ca,
complex *cx, int incx, complex *cbet, complex *cy, int
incy)
```

Arguments

UPLO	Indicates whether the values in a matrix reside in the upper or lower triangle of the array in which the matrix is stored. The legal values for UPLO are listed below. Any value not listed below is illegal. 'L' or 'l' Only the lower triangle of the array will be referenced. 'U' or 'u' Only the upper triangle of the array will be referenced.
N	Size of a matrix with N rows and N columns. $N \geq 0$.
xALPHA	Scalar that scales the input value of the matrix A.
xA	One-dimensional array that contains the input matrix. The matrix is in packed storage; the array must have at least enough elements to store the non-redundant elements of the matrix.
xX	X and INCX describe a vector of length N. X contains an input vector.
INCX	Scalar that contains the storage spacing between successive elements of the vector. $INCX \neq 0$. If $INCX = 1$, then elements of the vector are contiguous in memory. INCX may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations. If X is a one-dimensional array and $INCX = -1$ then the array will be accessed in reverse order. If X is a two-dimensional array and $INCX = LDA$ then the vector will be a row of the array.

	If X is a two-dimensional array and INCX = LDA+1 then the vector will be a diagonal of the array.
xBETA	Scalar that scales the input value of the vector Y.
xY	Y and INCY describe a vector of length N. On entry, an input vector. On exit, a result vector.
INCY	Scalar that contains the storage spacing between successive elements of the vector Y. INCY $\neq$ 0. If INCY = 1, then elements of the vector are contiguous in memory. INCY may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations. If Y is a one-dimensional array and INCY = -1 then the array will be accessed in reverse order. If Y is a two-dimensional array and INCY = LDA then the vector will be a row of the array. If Y is a two-dimensional array and INCY = LDA+1 then the vector will be a diagonal of the array.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
REAL      RZERO
INTEGER   LENGTA, N
PARAMETER (N = 4)
PARAMETER (LENGTA = (N * N + N) / 2)
PARAMETER (RZERO = 0.0E0)
C
COMPLEX   ALPHA, BETA
COMPLEX   AP(LENGTA), X(N), Y(N)
INTEGER   I, J
C
EXTERNAL  CHPMV
INTRINSIC CMPLX, CONJG, REAL
C
C      Initialize the array AP to store in packed Hermitian form
C      the matrix A shown below. Initialize the arrays X and Y
C      to store the vectors x and y shown below.
C
C      A =
C      1+0i  1-2i  1-2i  1-2i      x = 1+4i      y = 1+1i
C      1+2i  2+0i  2-3i  2-3i      x = 2+3i      y = 1+1i
C      1+2i  2+3i  3+0i  3-4i      x = 3+2i      y = 1+1i
C      1+2i  2+3i  3+4i  5+0i      x = 4+1i      y = 1+1i
C
C      DATA AP / (1.0,8E8), (1.0,2.0), (1.0,2.0), (1.0,2.0),
$              (2.0,8E8), (2.0,3.0), (2.0,3.0),

```

```

$          (3.0,8E8), (3.0,4.0), (5.0,8E8) /
C
      DO 10, I = 1, N
        X(I) = CMPLX (REAL (I), REAL (5-I))
        Y(I) = CMPLX (1.0, 1.0)
10    CONTINUE
      PRINT 1000
      PRINT 1010, REAL (AP(1)), RZERO, (CONJG (AP(J)), J = 2, 4)
      PRINT 1010, AP(2), REAL (AP(5)), RZERO, (CONJG (AP(J)),
$      J = 6, 7)
      PRINT 1010, AP(3), AP(6), REAL (AP(8)), RZERO, CONJG (AP(9))
      PRINT 1010, AP(4), AP(7), AP(9), REAL (AP(10)), RZERO
      PRINT 1030
      PRINT 1040, (X(I), Y(I), I = 1, 4)
      ALPHA = (2.0E0, 0.0E0)
      BETA = (1.0E0, 0.0E0)
      CALL CHPMV ('LOWER TRIANGULAR A', N, ALPHA, AP, X, 1, BETA,
$      Y, 1)
      PRINT 1050
      PRINT 1060, (Y(I), I = 1, N)
C
1000  FORMAT (/1X, 'Array A in full form:')
1010  FORMAT (1X, 4(2X, '(', F5.1, ', ', F5.1, ')'))
1020  FORMAT (/1X, 'Array A in packed form:')
1030  FORMAT (/7X, 'x', 14X, 'y')
1040  FORMAT (2(3X, '(', F5.1, ', ', F5.1, ')'))
1050  FORMAT (/1X, 'Ax + y:')
1060  FORMAT (3X, '(', F5.1, ', ', F5.1, ')')
C
      END

```

Sample Output

```

Array A in full form:
( 1.0, 0.0) ( 1.0, -2.0) ( 1.0, -2.0) ( 1.0, -2.0)
( 1.0, 2.0) ( 2.0, 0.0) ( 2.0, -3.0) ( 2.0, -3.0)
( 1.0, 2.0) ( 2.0, 3.0) ( 3.0, 0.0) ( 3.0, -4.0)
( 1.0, 2.0) ( 2.0, 3.0) ( 3.0, 4.0) ( 5.0, 0.0)

      x      y
( 1.0, 4.0) ( 1.0, 1.0)
( 2.0, 3.0) ( 1.0, 1.0)
( 3.0, 2.0) ( 1.0, 1.0)
( 4.0, 1.0) ( 1.0, 1.0)

Ax + y:
( 45.0, -15.0)
( 41.0, -5.0)
( 27.0, 23.0)
( 19.0, 83.0)

```

Rank-1 Update to a Hermitian Matrix in Packed Storage

The subroutines described in this section compute the following result for a Hermitian matrix A in packed storage and a vector x :

$$A \leftarrow \alpha x \bar{x}^T + A$$

Calling Sequence

```
CALL ZHPR    (UPLO, N, DALPHA, ZX, INCX, ZA)
CALL CHPR    (UPLO, N, SALPHA, CX, INCX, CA)

void zhpr    (char uplo, int n, double dalpha, doublecomplex *zx,
              int incx, doublecomplex *za)
void chpr    (char uplo, int n, float salpha, complex *cx, int incx,
              complex *ca)
```

Note – The type of the scalar α is real.

Arguments

UPLO	Indicates whether the values in a matrix reside in the upper or lower triangle of the array in which the matrix is stored. The legal values for UPLO are listed below. Any value not listed below is illegal. 'L' or 'l' Only the lower triangle of the array will be referenced. 'U' or 'u' Only the upper triangle of the array will be referenced.
N	Size of a matrix with N rows and N columns. $N \geq 0$.
α	Scalar that scales the input value of the matrix A.
X	X and INCX describe a vector of length N. X contains an input vector.
INCX	Scalar that contains the storage spacing between successive elements of the vector. $INCX \neq 0$. If $INCX = 1$, then elements of the vector are contiguous in memory. INCX may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations. If X is a one-dimensional array and $INCX = -1$ then the array will be accessed in reverse order. If X is a two-dimensional array and $INCX = LDA$ then the vector will be a row of the array. If X is a two-dimensional array and $INCX = LDA+1$ then the vector will be a diagonal of the array.

xA One-dimensional array that contains the input matrix. The matrix is in packed storage; the array must have at least enough elements to store the non-redundant elements of the matrix.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
REAL      RZERO
COMPLEX   ONE
INTEGER   LENGTA, N
PARAMETER (N = 3)
PARAMETER (LENGTA = (N * N + N) / 2)
PARAMETER (ONE = (1.0E0,0.0E0))
PARAMETER (RZERO = 0.0E0)
C
COMPLEX   AP(LENGTA), ONE, X(N)
REAL      RZERO
INTEGER   I
C
EXTERNAL  CHPR
INTRINSIC CMPLX, CONJG, REAL
C
C Initialize the array A to store in packed Hermitian form
C the matrix A shown below. Initialize the array X to
C store the vector x shown below.
C
C      1+0i  2-3i  4-5i      2-2i
C      A = 2+3i  6+0i  7-8i      x = 2-2i
C      4+5i  7+8i  9+0i      2-2i
C
DATA AP / (1.0,8E8), (2.0,3.0), (4.0,5.0), (6.0,8E8),
$      (7.0,8.0), (9.0,8E8) /
DATA X / (2.0,-2.0), (2.0,-2.0), (2.0,-2.0) /
C
PRINT 1000
PRINT 1010, REAL (AP(1)), RZERO, CONJG (AP(2)),
$ CONJG (AP(3))
PRINT 1010, AP(2), REAL (AP(4)), RZERO, CONJG (AP(5))
PRINT 1010, AP(3), AP(5), REAL (AP(6)), RZERO
PRINT 1020
PRINT 1030, (X(I), I = 1, N)
CALL CHPR ('LOWER TRIANGULAR A', N, ONE, X, 1, AP)
PRINT 1040
PRINT 1010, AP(1), CONJG (AP(2)), CONJG (AP(3))
PRINT 1010, AP(2), AP(4), CONJG (AP(5))
PRINT 1010, AP(3), AP(5), AP(6)
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (1X, 3(2X, '(', F5.1, ', ', F5.1, ')'))
1020 FORMAT (/1X, 'x:')

```

```

1030 FORMAT (3X, '(', F5.1, ', ', F5.1, ', ')')
1040 FORMAT (/1X, 'Axx':')
      END

```

Sample Output

```

A in full form:
( 1.0, 0.0) ( 2.0, -3.0) ( 4.0, -5.0)
( 2.0, 3.0) ( 6.0, 0.0) ( 7.0, -8.0)
( 4.0, 5.0) ( 7.0, 8.0) ( 9.0, 0.0)

x:
( 2.0, -2.0)
( 2.0, -2.0)
( 2.0, -2.0)

Axx':
( 9.0, 0.0) ( 10.0, -3.0) ( 12.0, -5.0)
( 10.0, 3.0) ( 14.0, 0.0) ( 15.0, -8.0)
( 12.0, 5.0) ( 15.0, 8.0) ( 17.0, 0.0)

```

Rank-2 Update to a Hermitian Matrix in Packed Storage

The subroutines described in this section compute the following result for the Hermitian matrix A in packed storage and vectors x and y:

$$A \leftarrow \alpha \bar{x}y^T + \bar{\alpha} y\bar{x}^T + A$$

Calling Sequence

```
CALL ZHPR2 (UPLO, N, ZALPHA, ZX, INCX, ZY, INCY, ZA)
CALL CHPR2 (UPLO, N, CALPHA, CX, INCX, CY, INCY, CA)

void zhpr2 (char uplo, int n, doublecomplex *zalpha,
doublecomplex *zx, int incx, doublecomplex *zy, int
incy, doublecomplex *za)
void chpr2 (char uplo, int n, complex *calpha, complex *x, int
incx, complex *cy, int incy, complex *ca)
```

Arguments

UPLO	Indicates whether the values in a matrix reside in the upper or lower triangle of the array in which the matrix is stored. The legal values for UPLO are listed below. Any value not listed below is illegal. 'L' or 'l' Only the lower triangle of the array will be referenced. 'U' or 'u' Only the upper triangle of the array will be referenced.
N	Size of a matrix with N rows and N columns. $N \geq 0$.
xALPHA	Scalar that scales the input value of the matrix A.
xX	X and INCX describe a vector of length N. X contains an input vector.
INCX	Scalar that contains the storage spacing between successive elements of the vector. $INCX \neq 0$. If $INCX = 1$, then elements of the vector are contiguous in memory. INCX may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations. If X is a one-dimensional array and $INCX = -1$ then the array will be accessed in reverse order. If X is a two-dimensional array and $INCX = LDA$ then the vector will be a row of the array. If X is a two-dimensional array and $INCX = LDA+1$ then the vector will be a diagonal of the array.
xY	Y and INCY describe a vector of length N. Y contains an input vector.

x_A

Sample Program

C
C
C
C
C
C
C
C
C
C

```

        PRINT 1010, REAL (AP(1)), RZERO, AP(2), AP(4)
        PRINT 1010, CONJG (AP(2)), REAL (AP(3)), RZERO, AP(5)
        PRINT 1010, CONJG (AP(4)), CONJG (AP(5)), REAL (AP(6)),
$      RZERO
        PRINT 1020
        PRINT 1030, AP
        PRINT 1040
        DO 20, I = 1, N
            PRINT 1050, X(I), Y(I)
20    CONTINUE
        CALL CHPR2 ('UPPER TRIANGULAR A', N, ONE, X, 1, Y, 1, AP)
        PRINT 1060
        PRINT 1010, REAL (AP(1)), RZERO, AP(2), AP(4)
        PRINT 1010, CONJG (AP(2)), REAL (AP(3)), RZERO, AP(5)
        PRINT 1010, CONJG (AP(4)), CONJG (AP(5)), REAL (AP(6)),
$      RZERO
C
1000  FORMAT (1X, 'A in full form:')
1010  FORMAT (1X, 3(2X, '(', F5.1, ', ', F5.1, ')'))
1020  FORMAT (/1X, 'A in packed form: (* in unused elements)')
1030  FORMAT (4(: 3X, '(', F5.1, ', ', F5.1, ')'))
1040  FORMAT (/10X, 'x ', 13X, 'y')
1050  FORMAT (1X, 2(2X, '(', F5.1, ', ', F5.1, ')'))
1060  FORMAT (/1X, 'A + xy' + yx'')
C
        END

```

Sample Output

```

A in full form:
(  1.0,   0.0) (  2.0,   3.0) (  5.0,   6.0)
(  2.0,  -3.0) (  4.0,   0.0) (  7.0,   8.0)
(  5.0,  -6.0) (  7.0,  -8.0) (  9.0,   0.0)

A in packed form: (* in unused elements)
(  1.0, *****) (  2.0,   3.0) (  4.0, *****) (  5.0,   6.0)
(  7.0,   8.0) (  9.0, *****)

          x          y
(  1.0,  -1.0) (  4.0,  -4.0)
(  2.0,  -2.0) (  4.0,  -4.0)
(  3.0,  -3.0) (  4.0,  -4.0)

A + xy' + yx':
(  1.0,   0.0) (  2.0,   3.0) (  5.0,   6.0)
(  2.0,  -3.0) (  4.0,   0.0) (  7.0,   8.0)
(  5.0,  -6.0) (  7.0,  -8.0) (  9.0,   0.0)

```

Product of a Symmetric Matrix in Banded Storage and a Vector

The subroutines described in this section compute the following result for a symmetric matrix A in banded storage and vectors x and y :

$$y \leftarrow \alpha Ax + \beta y$$

Calling Sequence

CALL DSBMV	(UPLO, N, NDIAG, DALPHA, ZA, LDA, ZX, INCX, DBETA, ZY, INCY)
CALL SSBMV	(UPLO, N, NDIAG, SALPHA, CA, LDA, CX, INCX, SBETA, CY, INCY)
void dsbmv	(char uplo, int n, int ndiag, double dalpha, double *da, int lda, double *dx, int incx, double dbeta, double *dy, int incy)
void ssbmv	(char uplo, int n, int ndiag, float salpha, float *sa, int lda, float *sx, int incx, float sbeta, float *sy, int incy)

Arguments

UPLO	Indicates whether the values in a matrix reside in the upper or lower triangle of the array in which the matrix is stored. The legal values for UPLO are listed below. Any value not listed below is illegal. 'L' or 'l' Only the lower triangle of the array will be referenced. 'U' or 'u' Only the upper triangle of the array will be referenced.
N	Size of a matrix with N rows and N columns. $N \geq 0$.
NDIAG	Bandwidth of a symmetric matrix in banded storage with NDIAG superdiagonals and NDIAG subdiagonals. $N-1 \geq \text{NDIAG} \geq 0$ but if $N = 0$ then $\text{NDIAG} = 0$.
xALPHA	Scalar that scales the input value of the matrix A .
xA	Two-dimensional array that contains the input matrix.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $\text{LDA} \geq \text{NDIAG} + 1$.
xX	X and INCX describe a vector of length N . X contains an input vector.
INCX	Scalar that contains the storage spacing between successive elements of the vector. $\text{INCX} \neq 0$. If $\text{INCX} = 1$, then elements of the vector are contiguous in memory. INCX may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations.

	If X is a one-dimensional array and INCX = -1 then the array will be accessed in reverse order.
	If X is a two-dimensional array and INCX = LDA then the vector will be a row of the array.
	If X is a two-dimensional array and INCX = LDA+1 then the vector will be a diagonal of the array.
xBETA	Scalar that scales the input value of the vector Y.
xY	Y and INCY describe a vector of length N.
	On entry, an input vector.
	On exit, a result vector.
INCY	Scalar that contains the storage spacing between successive elements of the vector Y. INCY ≠ 0. If INCY = 1, then elements of the vector are contiguous in memory. INCY may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations.
	If Y is a one-dimensional array and INCY = -1 then the array will be accessed in reverse order.
	If Y is a two-dimensional array and INCY = LDA then the vector will be a row of the array.
	If Y is a two-dimensional array and INCY = LDA+1 then the vector will be a diagonal of the array.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
C   INTEGER          LDA, N, NDIAG
C   PARAMETER        (N = 5)
C   PARAMETER        (NDIAG = 1)
C   PARAMETER        (LDA = NDIAG + 1)

C
C   INTEGER          I, J
C   DOUBLE PRECISION A(LDA,N), ALPHA, BETA, X(N), Y(N)

C
C   EXTERNAL DSBMV

C
C   Initialize the array A to store in symmetric banded form
C   the matrix A shown below. Initialize the arrays X and Y
C   to store the vectors X and Y shown below.
C
C
C           1  2          1      1
C           2  3  4          2      1
C   A =      4  5  6      x = 3      y = 1
C           6  7  8          4      1
C           8  9          5      1

```

```

C      DATA A /      8D8, 1.0D0, 2.0D0, 3.0D0, 4.0D0, 5.0D0, 6.0D0,
$      7.0D0, 8.0D0, 9.0D0 /
C      DATA X / 1.0D0, 2.0D0, 3.0D0, 4.0D0, 5.0D0 /
C      DATA Y / 1.0D0, 1.0D0, 1.0D0, 1.0D0, 1.0D0 /

C      PRINT 1000
C      PRINT 1010, A(2,1), A(1,2), 0.0D0, 0.0D0, 0.0D0
C      PRINT 1010, A(1,2), A(2,2), A(1,3), 0.0D0, 0.0D0
C      PRINT 1010, 0.0D0, A(1,3), A(2,3), A(1,4), 0.0D0
C      PRINT 1010, 0.0D0, 0.0D0, A(1,4), A(2,4), A(1,5)
C      PRINT 1010, 0.0D0, 0.0D0, 0.0D0, A(1,5), A(2,5)
C      PRINT 1020
C      DO 20, I = 1, LDA
C          PRINT 1010, (A(I,J), J = 1, N)
20 CONTINUE
C      PRINT 1030
C      PRINT 1040, (X(I), Y(I), I = 1, N)
C      ALPHA = 2.0D0
C      BETA = 3.0D0
C      CALL DSBMV ('UPPER TRIANGULAR A', N, NDIAG, ALPHA, A, LDA,
$      X, 1, BETA, Y, 1)
C      PRINT 1050
C      PRINT 1060, (Y(I), I = 1, N)

C      1000 FORMAT (1X, 'A in full form:')
C      1010 FORMAT (1X, 5(2X, F6.1))
C      1020 FORMAT (/1X, 'A in banded form: (* in unused fields)')
C      1030 FORMAT (/7X, 'x', 8X, 'y')
C      1040 FORMAT (3X, F6.1, 3X, F6.1)
C      1050 FORMAT (/1X, 'Ax + y:')
C      1060 FORMAT (1X, 3X, F6.1)

C      END

```

Sample Output

```

A in full form:
  1.0    2.0    0.0    0.0    0.0
  2.0    3.0    4.0    0.0    0.0
  0.0    4.0    5.0    6.0    0.0
  0.0    0.0    6.0    7.0    8.0
  0.0    0.0    0.0    8.0    9.0

A in banded form: (* in unused fields)
*****
  1.0    3.0    5.0    7.0    9.0

```

x	y
1.0	1.0
2.0	1.0
3.0	1.0
4.0	1.0
5.0	1.0
Ax + y:	
13.0	
43.0	
97.0	
175.0	
157.0	

Product of a Symmetric Matrix in Packed Storage and a Vector

The subroutines described in this section compute the following result for a symmetric matrix A in packed storage and vectors x and y:

$$y \leftarrow \alpha Ax + \beta y$$

Calling Sequence

```
CALL DSPMV (UPLO, N, DALPHA, DA, DX, INCX, DBETA, DY, INCY)
CALL SSPMV (UPLO, N, SALPHA, SA, SX, INCX, SBETA, SY, INCY)

void dspmv (char uplo, int n, double dalpha, double *da, double
            *dx, int incx, double dbeta, double *dy, int incy)
void sspmv (char uplo, int n, float salpha, float *sa, float *sx,
            int incx, float sbeta, float *sy, int incy)
```

Arguments

UPLO	Indicates whether the values in a matrix reside in the upper or lower triangle of the array in which the matrix is stored. The legal values for UPLO are listed below. Any value not listed below is illegal. 'L' or 'l' Only the lower triangle of the array will be referenced. 'U' or 'u' Only the upper triangle of the array will be referenced.
N	Size of a matrix with N rows and N columns. $N \geq 0$.
xALPHA	Scalar that scales the input value of the matrix A.
xA	One-dimensional array that contains the input matrix. The matrix is in packed storage; the array must have at least enough elements to store the non-redundant elements of the matrix.
xX	X and INCX describe a vector of length N. X contains an input vector.
INCX	Scalar that contains the storage spacing between successive elements of the vector. $INCX \neq 0$. If $INCX = 1$, then elements of the vector are contiguous in memory. INCX may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations. If X is a one-dimensional array and $INCX = -1$ then the array will be accessed in reverse order. If X is a two-dimensional array and $INCX = LDA$ then the vector will be a row of the array. If X is a two-dimensional array and $INCX = LDA+1$ then the vector will be a diagonal of the array.
xBETA	Scalar that scales the input value of the vector Y.

xY	Y and INCY describe a vector of length N. On entry, an input vector. On exit, a result vector.
INCY	Scalar that contains the storage spacing between successive elements of the vector Y. $INCY \neq 0$. If $INCY = 1$, then elements of the vector are contiguous in memory. INCY may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations. If Y is a one-dimensional array and $INCY = -1$ then the array will be accessed in reverse order. If Y is a two-dimensional array and $INCY = LDA$ then the vector will be a row of the array. If Y is a two-dimensional array and $INCY = LDA+1$ then the vector will be a diagonal of the array.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER          LENGTA, N
  PARAMETER        (N = 5)
  PARAMETER        (LENGTA = (N * N + N) / 2)
C
  DOUBLE PRECISION ALPHA, AP(LENGTA), BETA, X(N), Y(N)
  INTEGER          I
C
  EXTERNAL DSPMV
C
  C Initialize the array AP to store in packed symmetric form
  C the matrix A shown below. Initialize the arrays X and Y
  C to store the vectors x and y shown below.
C
C      1   2   3   4   5           1           1000
C      2   6   7   8   9           2           2000
C  A =  3   7  10  11  12       x =  3           3000
C      4   8  11  13  14           4           4000
C      5   9  12  14  15           5           5000
C
  DATA AP / 1.0D0, 2.0D0, 3.0D0, 4.0D0, 5.0D0,
$          6.0D0, 7.0D0, 8.0D0, 9.0D0,
$          1.0D1, 1.1D1, 1.2D1,
$          1.3D1, 1.4D1,
$          1.5D1 /
  DATA X / 1.0D0, 2.0D0, 3.0D0, 4.0D0, 5.0D0 /
  DATA Y / 1.0D3, 2.0D3, 3.0D3, 4.0D3, 5.0D3 /
C
  PRINT 1000

```



```

      PRINT 1010, AP(1), AP(2), AP(3), AP(4), AP(5)
      PRINT 1010, AP(2), AP(6), AP(7), AP(8), AP(9)
      PRINT 1010, AP(3), AP(7), AP(8), AP(9), AP(10)
      PRINT 1010, AP(4), AP(8), AP(9), AP(10), AP(11)
      PRINT 1010, AP(5), AP(9), AP(10), AP(11), AP(12)
      PRINT 1020
      PRINT 1030, (X(I), Y(I), I = 1, N)
      ALPHA = 1.0D0
      BETA = 0.0D0
      CALL DSPMV ('LOWER TRIANGULAR A', N, ALPHA, AP, X, 1,
$      BETA, Y, 1)
      PRINT 1040
      PRINT 1050, (Y(I), I = 1, N)
C
1000 FORMAT (1X, 'Array A in full form:')
1010 FORMAT (1X, 5(2X, F6.1))
1020 FORMAT (/1X, '      x      y')
1030 FORMAT (1X, F6.1, 2X, F6.1)
1040 FORMAT (/1X, 'Ax + y:')
1050 FORMAT (1X, F6.1)
C
      END

```

Sample Output

```

Array A in full form:
  1.0   2.0   3.0   4.0   5.0
  2.0   6.0   7.0   8.0   9.0
  3.0   7.0   8.0   9.0  10.0
  4.0   8.0   9.0  10.0  11.0
  5.0   9.0  10.0  11.0  12.0

      x      y
  1.0 1000.0
  2.0 2000.0
  3.0 3000.0
  4.0 4000.0
  5.0 5000.0

Ax + y:
  55.0
 112.0
 151.0
 175.0
 190.0

```

Rank-1 Update to a Real Symmetric Matrix in Packed Storage

The subroutines described in this section compute the following result for a real symmetric matrix A in packed storage and a vector x:

$$A \leftarrow \alpha x x^T + A$$

Calling Sequence

```
CALL DSPR    (UPLO, N, DALPHA, DX, INCX, DA)
CALL SSPR    (UPLO, N, SALPHA, SX, INCX, SA)
```

```
void dspr    (char uplo, int n, double dalpha, double *dx, int incx,
              double *da)
void sspr    (char uplo, int n, float salpha, float *sx, int incx,
              float *sa)
```

Arguments

UPLO	Indicates whether the values in a matrix reside in the upper or lower triangle of the array in which the matrix is stored. The legal values for UPLO are listed below. Any value not listed below is illegal. 'L' or 'l' Only the lower triangle of the array will be referenced. 'U' or 'u' Only the upper triangle of the array will be referenced.
N	Size of a matrix with N rows and N columns. $N \geq 0$.
xALPHA	Scalar that scales the input value of the matrix A.
xX	X and INCX describe a vector of length N. X contains an input vector.
INCX	Scalar that contains the storage spacing between successive elements of the vector. $INCX \neq 0$. If $INCX = 1$, then elements of the vector are contiguous in memory. INCX may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations. If X is a one-dimensional array and $INCX = -1$ then the array will be accessed in reverse order. If X is a two-dimensional array and $INCX = LDA$ then the vector will be a row of the array. If X is a two-dimensional array and $INCX = LDA+1$ then the vector will be a diagonal of the array.
xA	One-dimensional array that contains the input matrix. The matrix is in packed storage; the array must have at least enough elements to store the non-redundant elements of the matrix.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
INTEGER          LENGTA, N
PARAMETER        (N = 3)
PARAMETER        (LENGTA = (N * N + N) / 2)

C
DOUBLE PRECISION ALPHA, AP(LENGTA), X(N)
INTEGER          I

C
EXTERNAL DSPR

C
C Initialize the array A to store in symmetric packed form the
C matrix A shown below.  Initialize the array X to store the
C vector x shown below.
C
C      100   200   300           1
C  A =  200   400   500      x =  2
C      300   500   600           3
C
C      DATA AP / 100.0, 200.0, 300.0, 400.0, 500.0, 600.0 /
C      DATA X / 1.0, 2.0, 3.0 /

C
PRINT 1000
PRINT 1010, AP(1), AP(2), AP(3)
PRINT 1010, AP(2), AP(4), AP(5)
PRINT 1010, AP(3), AP(5), AP(6)
PRINT 1020
PRINT 1030, AP
PRINT 1040
PRINT 1050, (X(I), I = 1, N)
ALPHA = 1.0D0
CALL DSPR ('LOWER TRIANGULAR A', N, ALPHA, X, 1, AP)
PRINT 1060
PRINT 1010, AP(1), AP(2), AP(3)
PRINT 1010, AP(2), AP(4), AP(5)
PRINT 1010, AP(3), AP(5), AP(6)

C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (1X, 3(2X, F5.1))
1020 FORMAT (/1X, 'A in packed form:')
1030 FORMAT (1X, 6(2X, F5.1))
1040 FORMAT (/1X, 'x:')
1050 FORMAT (1X, 2X, F5.1)
1060 FORMAT (/1X, 'Axx'':')

C
END

```

Sample Output

```

A in full form:
 100.0  200.0  300.0
 200.0  400.0  500.0
 300.0  500.0  600.0

A in packed form:
100.0  200.0  300.0  400.0  500.0  600.0

x:
  1.0
  2.0
  3.0

Axx':
101.0  202.0  303.0
202.0  404.0  506.0
303.0  506.0  609.0

```

Rank-2 Update to a Real Symmetric Matrix in Packed Storage

The subroutines described in this section compute the following result for the real symmetric matrix A in packed storage and vectors x and y:

$$A \leftarrow \alpha xy^T + \alpha yx^T + A$$

Calling Sequence

```
CALL DSPR2 (UPLO, N, DALPHA, DX, INCX, DY, INCY, DA)
CALL SSPR2 (UPLO, N, SALPHA, SX, INCX, SY, INCY, SA)

void dspr2 (char uplo, int n, double dalpha, double *dx, int incx,
            double *dy, int incy, double *da)
void sspr2 (char uplo, int n, float salpha, float *sx, int incx,
            float *sy, int incy, float *sa)
```

Arguments

UPLO	Indicates whether the values in a matrix reside in the upper or lower triangle of the array in which the matrix is stored. The legal values for UPLO are listed below. Any value not listed below is illegal. 'L' or 'l' Only the lower triangle of the array will be referenced. 'U' or 'u' Only the upper triangle of the array will be referenced.
N	Size of a matrix with N rows and N columns. $N \geq 0$.
xALPHA	Scalar that scales the input value of the matrix A.
xX	X and INCX describe a vector of length N. X contains an input vector.
INCX	Scalar that contains the storage spacing between successive elements of the vector. $INCX \neq 0$. If $INCX = 1$, then elements of the vector are contiguous in memory. INCX may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations. If X is a one-dimensional array and $INCX = -1$ then the array will be accessed in reverse order. If X is a two-dimensional array and $INCX = LDA$ then the vector will be a row of the array. If X is a two-dimensional array and $INCX = LDA+1$ then the vector will be a diagonal of the array.
xY	Y and INCY describe a vector of length N. Y contains an input vector.

INCY	Scalar that contains the storage spacing between successive elements of the vector Y. $INCY \neq 0$. If $INCY = 1$, then elements of the vector are contiguous in memory. INCY may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations. If Y is a one-dimensional array and $INCY = -1$ then the array will be accessed in reverse order. If Y is a two-dimensional array and $INCY = LDA$ then the vector will be a row of the array. If Y is a two-dimensional array and $INCY = LDA+1$ then the vector will be a diagonal of the array.
xA	One-dimensional array that contains the input matrix on entry to the subroutine and the result matrix on exit. The matrix is in packed storage; the array must have at least enough elements to store the non-redundant elements of the matrix.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER          LENGTA, N
  PARAMETER        (N = 3)
  PARAMETER        (LENGTA = (N * N + N) / 2)
C
  DOUBLE PRECISION ALPHA, AP(LENGTA), X(N), Y(N)
  INTEGER          I
C
  EXTERNAL          DSPR2
C
  C Initialize the array to store in packed symmetric form the
  C symmetric matrix A shown below. Initialize the arrays X
  C and Y to store the vectors x and y shown below.
  C
  C      10  20  30      1      -4
  C      20  40  50      x = 2      y = -4
  C      30  50  60      3      -4
  C
  DATA AP / 1.0D1, 2.0D1, 3.0D1, 4.0D1, 5.0D1, 6.0D1 /
  DATA X / 1.0D0, 2.0D0, 3.0D0 /
  DATA Y / -4.0D0, -4.0D0, -4.0D0 /
C
  PRINT 1000
  PRINT 1010, AP(1), AP(2), AP(4)
  PRINT 1010, AP(2), AP(3), AP(5)
  PRINT 1010, AP(4), AP(5), AP(6)
  PRINT 1020
  DO 20, I = 1, N

```

```

      PRINT 1030, X(I), Y(I)
20 CONTINUE
  ALPHA = 1.0D1
  CALL DSPR2 ('UPPER TRIANGULAR A', N, ALPHA, X, 1, Y, 1, AP)
  PRINT 1040
  PRINT 1010, AP(1), AP(2), AP(4)
  PRINT 1010, AP(2), AP(3), AP(5)
  PRINT 1010, AP(4), AP(5), AP(6)
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (1X, 3(2X, F6.1))
1020 FORMAT (/7X, 'x', 7X, 'y')
1030 FORMAT (1X, 2(2X, F6.1))
1040 FORMAT (/1X, 'A + xy'' + yx'')
C
      END

```

Sample Output

```

A in full form:
  10.0   20.0   40.0
  20.0   30.0   50.0
  40.0   50.0   60.0

      x      y
      1.0   -4.0
      2.0   -4.0
      3.0   -4.0

A + xy' + yx'
 -70.0  -100.0  -120.0
-100.0  -130.0  -150.0
-120.0  -150.0  -180.0

```

Product of a Symmetric Matrix and a Vector

The subroutines described in this section compute the following result for a real symmetric matrix A and vectors x and y:

$$y \leftarrow \alpha Ax + \beta y$$

Calling Sequence

```
CALL DSYMV (UPLO, N, DALPHA, DA, LDA, DX, INCX, DBETA, DY, INCY)
CALL SSYMV (UPLO, N, SALPHA, SA, LDA, SX, INCX, SBETA, SY, INCY)

void dsymv (char uplo, int n, double dalpha, double *da, int lda,
            double *dx, int incx, double dbeta, double *dy, int
            incy)
void ssymv (char uplo, int n, float salpha, float *sa, int lda,
            float *sx, int incx, float sbeta, float *sy, int incy)
```

Arguments

UPLO	Indicates whether the values in a matrix reside in the upper or lower triangle of the array in which the matrix is stored. The legal values for UPLO are listed below. Any value not listed below is illegal. 'L' or 'l' Only the lower triangle of the array will be referenced. 'U' or 'u' Only the upper triangle of the array will be referenced.
N	Size of a matrix with N rows and N columns. $N \geq 0$.
xALPHA	Scalar that scales the input value of the matrix A.
xA	Two-dimensional array that contains the input matrix.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
xX	X and INCX describe a vector of length N. X contains an input vector.
INCX	Scalar that contains the storage spacing between successive elements of the vector. $INCX \neq 0$. If $INCX = 1$, then elements of the vector are contiguous in memory. INCX may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations.

If X is a one-dimensional array and $INCX = -1$ then the array will be accessed in reverse order.

If X is a two-dimensional array and $INCX = LDA$ then the vector will be a row of the array.

	If X is a two-dimensional array and $INCX = LDA+1$ then the vector will be a diagonal of the array.
xBETA	Scalar that scales the input value of the vector Y.
xY	Y and INCY describe a vector of length N. On entry, an input vector. On exit, a result vector.
INCY	Scalar that contains the storage spacing between successive elements of the vector Y. $INCY \neq 0$. If $INCY = 1$, then elements of the vector are contiguous in memory. INCY may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations. If Y is a one-dimensional array and $INCY = -1$ then the array will be accessed in reverse order. If Y is a two-dimensional array and $INCY = LDA$ then the vector will be a row of the array. If Y is a two-dimensional array and $INCY = LDA+1$ then the vector will be a diagonal of the array.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
C   INTEGER          LDA, N
C   PARAMETER       (N = 4)
C   PARAMETER       (LDA = N)
C
C   DOUBLE PRECISION A(LDA,N), ALPHA, BETA, X(N), Y(N)
C   INTEGER          I, J
C
C   EXTERNAL         DSYMV
C
C   Initialize the array A to store in symmetric form the
C   matrix A shown below. Initialize the arrays X and Y to
C   store the vectors X and Y shown below.
C
C   1  2  3  4      2      4000
C   A = 2  5  6  7  x = 2  y = 5000
C       3  6  8  9      2      6000
C       4  7  9 10      2      7000
C
C   DATA A / 1.0D0, -8.0D8, -8.0D8, -8.0D8,
$           2.0D0,  5.0D0, -8.0D8, -8.0D8,
$           3.0D0,  6.0D0,  8.0D0, -8.0D8,
$           4.0D0,  7.0D0,  9.0D0,  1.0D1 /
C   DATA X / 2.0D0, 2.0D0, 2.0D0, 2.0D0 /
C   DATA Y / 4.0D3, 5.0D3, 6.0D3, 7.0D3 /

```

```

C
    PRINT 1000
    DO 10, I = 1, N
        PRINT 1010, (A(J,I), J = 1, I), (A(I,J), J = I + 1, N)
10  CONTINUE
    PRINT 1020
    PRINT 1030, (X(I), Y(I), I = 1, N)
    ALPHA = 1.0D0
    BETA = 0.0D0
    CALL DSYMV ('UPPER TRIANGULAR A', N, ALPHA, A, LDA, X, 1,
$           BETA, Y, 1)
    PRINT 1040
    PRINT 1050, Y
C
1000 FORMAT (1X, 'Matrix A in full form:')
1010 FORMAT (1X, 4(2X, F6.1))
1020 FORMAT (/7X, 'x', 6X, 'y')
1030 FORMAT (3X, F6.1, 2X, F6.1)
1040 FORMAT (/1X, 'Ax + y:')
1050 FORMAT (3X, F6.1)
C
    END

```

Sample Output

```

Matrix A in full form:
    1.0    2.0    3.0    4.0
    2.0    5.0    6.0    7.0
    3.0    6.0    8.0    9.0
    4.0    7.0    9.0   10.0

      x      y
    2.0  4000.0
    2.0  5000.0
    2.0  6000.0
    2.0  7000.0

Ax + y:
    20.0
    40.0
    52.0
    60.0

```

Rank-1 Update to a Real Symmetric Matrix

The subroutines described in this section compute the following result for a real symmetric matrix A and a vector x :

$$A \leftarrow \alpha x x^T + A$$

Calling Sequence

CALL DSYR	(UPLO, N, DALPHA, DX, INCX, DA, LDA)
CALL SSYR	(UPLO, N, SALPHA, SX, INCX, SA, LDA)
void dsyr	(char uplo, int n, double dalpha, double *dx, int incx, double *da, int lda)
void ssyr	(char uplo, int n, float salpha, float *sx, int incx, float *sa, int lda)

Arguments

UPLO	Indicates whether the values in a matrix reside in the upper or lower triangle of the array in which the matrix is stored. The legal values for UPLO are listed below. Any value not listed below is illegal. 'L' or 'l' Only the lower triangle of the array will be referenced. 'U' or 'u' Only the upper triangle of the array will be referenced.
N	Size of a matrix with N rows and N columns. $N \geq 0$.
xALPHA	Scalar that scales the input value of the matrix A.
xX	X and INCX describe a vector of length N. X contains an input vector.
INCX	Scalar that contains the storage spacing between successive elements of the vector. $INCX \neq 0$. If $INCX = 1$, then elements of the vector are contiguous in memory. INCX may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations. If X is a one-dimensional array and $INCX = -1$ then the array will be accessed in reverse order. If X is a two-dimensional array and $INCX = LDA$ then the vector will be a row of the array. If X is a two-dimensional array and $INCX = LDA+1$ then the vector will be a diagonal of the array.
xA	Two-dimensional array. On entry, the input matrix. On exit, the result matrix.

LDA Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.

Sample Program

```

C      PROGRAM TEST
C      IMPLICIT NONE
C
C      INTEGER          LDA, N
C      PARAMETER        (N = 3)
C      PARAMETER        (LDA = N)
C
C      DOUBLE PRECISION A(LDA,N), ALPHA, X(N)
C      INTEGER          I, J
C
C      EXTERNAL          DSYR
C
C      Initialize the array A to store in symmetric form the
C      symmetric matrix A shown below. Initialize the arrays
C      X and Y to store the vectors x and y shown below.
C
C      A =   100  200  400      x = 1
C           200  300  500      x = 2
C           400  500  600      x = 3
C
C      DATA A / 1.0D2, 8D8, 8D8, 2.0D2, 3.0D2, 8D8,
C      $ 4.0D2, 5.0D2, 6.0D2 /
C      DATA X / 1.0D0, 2.0D0, 3.0D0 /
C
C      PRINT 1000
C      DO 100, I = 1, N
C          PRINT 1010, (A(J,I), J = 1, I), (A(I,J), J = I + 1, N)
100  CONTINUE
C      PRINT 1020
C      DO 110, I = 1, N
C          PRINT 1010, (A(I,J), J = 1, N)
110  CONTINUE
C      PRINT 1030
C      DO 120, I = 1, N
C          PRINT 1040, X(I)
120  CONTINUE
C      ALPHA = 1.0D0
C      CALL DSYR ('UPPER TRIANGULAR A', N, ALPHA, X, 1, A, LDA)
C      PRINT 1050
C      DO 130, I = 1, N
C          PRINT 1010, (A(J,I), J = 1, I), (A(I,J), J = I + 1, N)
130  CONTINUE
C
C      1000 FORMAT (1X, 'A in full form:')
C      1010 FORMAT (1X, 3(2X, F5.1))
C      1020 FORMAT (/1X, 'A in symmetric form: (* in unused elements)')
C      1030 FORMAT (/1X, 'x:')

```

```
1040 FORMAT (3X, F6.1)
1050 FORMAT (/1X, 'A + xx':')
C
      END
```

Sample Output

```
A in full form:
100.0  200.0  400.0
200.0  300.0  500.0
400.0  500.0  600.0

A in symmetric form: (* in unused elements)
100.0  200.0  400.0
***** 300.0  500.0
***** ***** 600.0

x:
      1.0
      2.0
      3.0

A + xx':
101.0  202.0  403.0
202.0  304.0  506.0
403.0  506.0  609.0
```

Rank-2 Update to a Real Symmetric Matrix

The subroutines described in this section compute the following result for the real symmetric matrix A and vectors x and y:

$$A \leftarrow \alpha xy^T + \alpha yx^T + A$$

Calling Sequence

```
CALL DSUR2 (UPLO, N, DALPHA, DX, INCX, DY, INCY, DA, LDA)
CALL SSUR2 (UPLO, N, SALPHA, SX, INCX, SY, INCY, SA, LDA)
```

```
void dsur2 (char uplo, int n, double dalpha, double *dx, int incx,
            double *dy, int incy, double *da, int lda)
void ssur2 (char uplo, int n, float salpha, float *sx, int incx,
            float *sy, int incy, float *sa, int lda)
```

Arguments

UPLO	Indicates whether the values in a matrix reside in the upper or lower triangle of the array in which the matrix is stored. The legal values for UPLO are listed below. Any value not listed below is illegal. 'L' or 'l' Only the lower triangle of the array will be referenced. 'U' or 'u' Only the upper triangle of the array will be referenced.
N	Size of a matrix with N rows and N columns. $N \geq 0$.
xALPHA	Scalar that scales the input value of the matrix A.
xX	X and INCX describe a vector of length N. X contains an input vector.
INCX	Scalar that contains the storage spacing between successive elements of the vector. $INCX \neq 0$. If $INCX = 1$, then elements of the vector are contiguous in memory. INCX may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations. If X is a one-dimensional array and $INCX = -1$ then the array will be accessed in reverse order. If X is a two-dimensional array and $INCX = LDA$ then the vector will be a row of the array. If X is a two-dimensional array and $INCX = LDA+1$ then the vector will be a diagonal of the array.
xY	Y and INCY describe a vector of length N. Y contains an input vector.

INCY	Scalar that contains the storage spacing between successive elements of the vector Y. $INCY \neq 0$. If $INCY = 1$, then elements of the vector are contiguous in memory. INCY may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations. If Y is a one-dimensional array and $INCY = -1$ then the array will be accessed in reverse order. If Y is a two-dimensional array and $INCY = LDA$ then the vector will be a row of the array. If Y is a two-dimensional array and $INCY = LDA+1$ then the vector will be a diagonal of the array.
xA	Two-dimensional array. On entry, the input matrix. On exit, the result matrix.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C   INTEGER          LDA, N
C   PARAMETER       (N = 3)
C   PARAMETER       (LDA = N)
C
C   DOUBLE PRECISION A(LDA,N), ALPHA, X(N), Y(N)
C   INTEGER          I, J
C
C   EXTERNAL         DSQR2
C
C   Initialize the array A to store in symmetric form the
C   symmetric matrix A shown below. Initialize the arrays X and
C   Y to store the vectors x and y shown below.
C
C       20  10  10      1      4
C   A = 10  20  10    x = 2    y = 5
C       10  10  20      3      6
C
C   DATA A / 20.0, 8D8, 8D8, 10.0, 20.0, 8D8, 10.0, 10.0, 20.0 /
C   DATA X / 1.0, 2.0, 3.0 /
C   DATA Y / 4.0, 5.0, 6.0 /
C   PRINT 1000
C   DO 100, I = 1, N
C       PRINT 1010, (A(J,I), J = 1, I), (A(I,J), J = I + 1, N)
100 CONTINUE
C   PRINT 1020
C   DO 110, I = 1, N

```

```

        PRINT 1010, (A(I,J), J = 1, N)
110 CONTINUE
    PRINT 1030
    DO 120, I = 1, N
        PRINT 1040, X(I), Y(I)
120 CONTINUE
    ALPHA = 1.0D0
    CALL DSYR2 ('UPPER TRIANGULAR A', N, ALPHA, X, 1, Y, 1,
$      A, LDA)
    PRINT 1050
    DO 130, I = 1, N
        PRINT 1010, (A(J,I), J = 1, I), (A(I,J), J = I + 1, N)
130 CONTINUE
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (1X, 3(2X, F5.1))
1020 FORMAT (/1X, 'A in symmetric form: (* in unused elements)')
1030 FORMAT (/3X, '  x  ', 2X, '  y')
1040 FORMAT (1X, 2(2X, F5.1))
1050 FORMAT (/1X, 'A + xy'' + yx''')
C
        END

```

Sample Output

```

A in full form:
  20.0   10.0   10.0
  10.0   20.0   10.0
  10.0   10.0   20.0

A in symmetric form: (* in unused elements)
  20.0   10.0   10.0
  *****  20.0   10.0
  *****  *****  20.0

      x      y
      1.0    4.0
      2.0    5.0
      3.0    6.0

A + xy' + yx'
  28.0   23.0   28.0
  23.0   40.0   37.0
  28.0   37.0   56.0

```


Product of a Triangular Matrix in Banded Storage and a Vector

The subroutines described in this section compute one of the following results for a triangular matrix A in banded storage and a vector y :

$$y \leftarrow Ay$$

$$y \leftarrow A^T y$$

$$y \leftarrow \bar{A}^T y$$

Calling Sequence

```
CALL DTBMV (UPLO, TRANSA, DIAG, N, NDIAG, DA, LDA, DY, INCY)
CALL STBMV (UPLO, TRANSA, DIAG, N, NDIAG, SA, LDA, SY, INCY)
CALL ZTBMV (UPLO, TRANSA, DIAG, N, NDIAG, ZA, LDA, ZY, INCY)
CALL CTBMV (UPLO, TRANSA, DIAG, N, NDIAG, CA, LDA, CY, INCY)
```

```
void dtbmv (char uplo, char trans, char diag, int n, int ndiag,
double *da, int lda, double *dx, int incx)
void stbmv (char uplo, char trans, char diag, int n, int ndiag,
float *sa, int lda, float *sx, int incx)
void ztbmv (char uplo, char trans, char diag, int n, int ndiag,
doublecomplex *za, int lda, doublecomplex *zx, int
incx)
void ctbmv (char uplo, char trans, char diag, int n, int ndiag,
complex *ca, int lda, complex *cx, int incx)
```

Arguments

UPLO	Indicates whether the values in a matrix reside in the upper or lower triangle of the array in which the matrix is stored. The legal values for UPLO are listed below. Any value not listed below is illegal. 'L' or 'l' Only the lower triangle of the array will be referenced. 'U' or 'u' Only the upper triangle of the array will be referenced.
TRANSA	Indicates how to use the matrix stored in the array argument. The legal values for TRANSA are listed below. Any value not listed below is illegal. 'N' or 'n' Use the matrix as it is stored in the array. 'T' or 't' Use the transpose of the matrix. 'C' or 'c' Use the conjugate transpose of the matrix.

DIAG	Indicates whether or not a triangular matrix is unit triangular. If the matrix is unit triangular then none of the diagonal elements of the array will be accessed. The legal values for DIAG are listed below. Any value not listed below is illegal. 'N' or 'n' The matrix is not unit triangular. 'U' or 'u' The matrix is unit triangular.
N	Size of a matrix with N rows and N columns. $N \geq 0$.
NDIAG	Bandwidth of a matrix with NDIAG superdiagonals or subdiagonals. $N-1 \geq \text{NDIAG} \geq 0$ but if $N = 0$ then $\text{NDIAG} = 0$.
xA	Two-dimensional array that contains the input matrix.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $\text{LDA} \geq \text{NDIAG} + 1$.
xY	Y and INCY describe a vector of length N. On entry, an input vector. On exit, a result vector.
INCY	Scalar that contains the storage spacing between successive elements of the vector Y. $\text{INCY} \neq 0$. If INCY is one then elements of the vector are contiguous in memory. INCY may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations. If Y is a one-dimensional array and $\text{INCY} = -1$ then the array will be accessed in reverse order. If Y is a two-dimensional array and $\text{INCY} = \text{LDA}$ then the vector will be a row of the array. If Y is a two-dimensional array and $\text{INCY} = \text{LDA} + 1$ then the vector will be a diagonal of the array.

Sample Program

	PROGRAM TEST
	IMPLICIT NONE
C	INTEGER LDA, N, NDIAG
	PARAMETER (N = 4)
	PARAMETER (NDIAG = 2)
	PARAMETER (LDA = NDIAG + 1)
C	DOUBLE PRECISION A(LDA,N), X(N)
	INTEGER I, J
C	EXTERNAL DTBMV
	INTRINSIC DBLE
C	

```

C      Initialize the array A to store in triangular banded form
C      the matrix A shown below. Initialize the array X to store
C      the vector x shown below.
C
C      1           4
C      A = 2  4           x = 3
C      3  5  7           2
C      6  8  9           1
C
C      DATA A /1.0D0, 2.0D0, 3.0D0, 4.0D0, 5.0D0, 6.0D0,
$      7.0D0, 8.0D0, 8.8D8, 9.0D0, 8.8D8, 8.8D8 /
C
C      DO 10, I = 1, N
C      X(I) = DBLE (5 - I)
10 CONTINUE
C      PRINT 1000
C      PRINT 1010, A(1,1)
C      PRINT 1010, A(2,1), A(1,2)
C      PRINT 1010, A(3,1), A(2,2), A(1,3)
C      PRINT 1020, A(3,2), A(2,3), A(1,4)
C      PRINT 1030
C      DO 20, I = 1, LDA
C      PRINT 1010, (A(I,J), J = 1, N)
20 CONTINUE
C      PRINT 1040
C      PRINT 1050, X
C      CALL DTBMV ('LOWER TRIANGULAR A', 'NOT TRANSPOSED A',
$      'NOT UNIT DIAGONAL A', N, NDIAG, A, LDA, X, 1)
C      PRINT 1060
C      PRINT 1050, X
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (1X, 4(2X, F5.1))
1020 FORMAT (1X, 7X, 3(2X, F5.1))
1030 FORMAT (/1X, 'A in banded form: (* in unused elements)')
1040 FORMAT (/1X, 'Original x vector:')
1050 FORMAT (1X, 2X, F5.1)
1060 FORMAT (/1X, 'Ax:')
C
C      END

```

Sample Output

```

A in full form:
  1.0
  2.0  4.0
  3.0  5.0  7.0
        6.0  8.0  9.0

```

A in banded form: (* in unused elements)

```
1.0  4.0  7.0  9.0
2.0  5.0  8.0  *****
3.0  6.0  *****  *****
```

Original x vector:

```
4.0
3.0
2.0
1.0
```

Ax:

```
4.0
20.0
41.0
43.0
```

Solution to a Triangular System in Banded Storage of Linear Equations

The subroutines described in this section compute one of the following results for a triangular matrix A in banded storage and a vector y :

$$y \leftarrow A^{-1}y$$

$$y \leftarrow A^{-T}y$$

$$y \leftarrow \bar{A}^{-T}y$$

Calling Sequence

```
CALL DTBSV (UPLO, TRANSA, DIAG, N, NDIAG, DA, LDA, DY, INCY)
CALL STBSV (UPLO, TRANSA, DIAG, N, NDIAG, SA, LDA, SY, INCY)
CALL ZTBSV (UPLO, TRANSA, DIAG, N, NDIAG, ZA, LDA, ZY, INCY)
CALL CTBSV (UPLO, TRANSA, DIAG, N, NDIAG, CA, LDA, CY, INCY)
```

```
void dtbsv (char uplo, char trans, char diag, int n, int ndiag,
double *da, int lda, double *dx, int incx)
void stbsv (char uplo, char trans, char diag, int n, int ndiag,
float *sa, int lda, float *sx, int incx)
void ztbsv (char uplo, char trans, char diag, int n, int ndiag,
doublecomplex *za, int lda, doublecomplex *zx, int
incx)
void ctbsv (char uplo, char trans, char diag, int n, int ndiag,
complex *ca, int lda, complex *cx, int incx)
```

Arguments

UPLO	Indicates whether the values in a matrix reside in the upper or lower triangle of the array in which the matrix is stored. The legal values for UPLO are listed below. Any value not listed below is illegal. 'L' or 'l' Only the lower triangle of the array will be referenced. 'U' or 'u' Only the upper triangle of the array will be referenced.
TRANSA	Indicates how to use the matrix stored in the array argument. The legal values for TRANSA are listed below. Any value not listed below is illegal. 'N' or 'n' Use the matrix as it is stored in the array. 'T' or 't' Use the transpose of the matrix. 'C' or 'c' Use the conjugate transpose of the matrix.

DIAG	Indicates whether or not a triangular matrix is unit triangular. If the matrix is unit triangular then none of the diagonal elements of the array will be accessed. The legal values for DIAG are listed below. Any value not listed below is illegal. 'N' or 'n' The matrix is not unit triangular. 'U' or 'u' The matrix is unit triangular.
N	Size of a matrix with N rows and N columns. $N \geq 0$.
NDIAG	Bandwidth of a matrix with NDIAG superdiagonals or subdiagonals. $N-1 \geq \text{NDIAG} \geq 0$ but if $N = 0$ then $\text{NDIAG} = 0$.
xA	Two-dimensional array that contains the input matrix.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $\text{LDA} \geq \text{NDIAG} + 1$.
xY	Y and INCY describe a vector of length N. On entry, an input vector. On exit, a result vector.
INCY	Scalar that contains the storage spacing between successive elements of the vector Y. $\text{INCY} \neq 0$. If INCY is one then elements of the vector are contiguous in memory. INCY may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations. If Y is a one-dimensional array and $\text{INCY} = -1$ then the array will be accessed in reverse order. If Y is a two-dimensional array and $\text{INCY} = \text{LDA}$ then the vector will be a row of the array. If Y is a two-dimensional array and $\text{INCY} = \text{LDA} + 1$ then the vector will be a diagonal of the array.

Sample Program

	PROGRAM TEST	
	IMPLICIT NONE	
C	DOUBLE PRECISION	ZERO
	INTEGER	LDA, N, NDIAG
	PARAMETER	(N = 4)
	PARAMETER	(NDIAG = 1)
	PARAMETER	(LDA = NDIAG + 1)
	PARAMETER	(ZERO = 0.0D0)
C	DOUBLE PRECISION	A(LDA,N), B(N)
	INTEGER	I, J
C	EXTERNAL	DTBSV

```

C
C      Initialize the array A to store in triangular banded form
C      the triangular banded matrix A shown below. Initialize
C      the array B to store the vector b shown below.
C
C      A =  $\begin{bmatrix} 1 & & & \\ 2 & 3 & & \\ & 4 & 5 & \\ & & 6 & 7 \end{bmatrix}$       b =  $\begin{bmatrix} 8 \\ 43 \\ 86 \\ 137 \end{bmatrix}$ 
C
C      DATA A / 1.0D0, 2.0D0, 3.0D0, 4.0D0, 5.0D0, 6.0D0, 7.0D0,
$      8D8 /
C      DATA B / 8.0D0, 4.3D1, 8.6D1, 1.37D2 /
C
C      PRINT 1000
C      PRINT 1010, A(1,1)
C      PRINT 1010, A(2,1), A(1,2)
C      PRINT 1020,          A(2,2), A(1,3)
C      PRINT 1030,          A(2,3), A(1,4)
C      PRINT 1040
C      DO 10, I = 1, LDA
C          PRINT 1010, (A(I,J), J = 1, N)
10  CONTINUE
C      PRINT 1050
C      PRINT 1060, B
C      CALL DTBSV ('LOWER TRIANGULAR A', 'NOT TRANSPOSED A',
$      'NOT UNIT DIAGONAL A', N, 1, A, LDA, B, 1)
C      PRINT 1070
C      PRINT 1060, B
C
C      1000 FORMAT (1X, 'A in full form:')
C      1010 FORMAT (1X, 4(2X, F5.1))
C      1020 FORMAT (8X, 2(2X, F5.1))
C      1030 FORMAT (15X, 2(2X, F5.1))
C      1040 FORMAT (/1X, 'A in triangular banded form: ',
$      '(* in unused elements)')
C      1050 FORMAT (/1X, 'b:')
C      1060 FORMAT (3X, F5.1)
C      1070 FORMAT (/1X, 'A**(-1) * b:')
C
C      END

```

Sample Output

```

A in full form:
  1.0
  2.0   3.0
        4.0   5.0
              6.0   7.0

```

```
A in triangular banded form: (* in unused elements)
  1.0   3.0   5.0   7.0
  2.0   4.0   6.0 *****

b:
  8.0
 43.0
 86.0
137.0

A**(-1) * b:
  8.0
  9.0
 10.0
 11.0
```


Product of a Triangular Matrix in Packed Storage and a Vector

The subroutines described in this section compute one of the following results for a triangular matrix A in packed storage and a vector y :

$$y \leftarrow Ay$$

$$y \leftarrow A^T y$$

$$y \leftarrow \bar{A}^T y$$

Calling Sequence

```
CALL DTPMV (UPLO, TRANSA, DIAG, N, DA, DY, INCY)
CALL STPMV (UPLO, TRANSA, DIAG, N, SA, SY, INCY)
CALL ZTPMV (UPLO, TRANSA, DIAG, N, ZA, ZY, INCY)
CALL CTPMV (UPLO, TRANSA, DIAG, N, CA, CY, INCY)
```

```
void dtpmv (char uplo, char trans, char diag, int n, double *da,
double *dx, int incx)
void stpmv (char uplo, char trans, char diag, int n, float *sa,
float *sx, int incx)
void ztpmv (char uplo, char trans, char diag, int n, doublecomplex
*za, doublecomplex *zx, int incx)
void ctpmv (char uplo, char trans, char diag, int n, complex *ca,
complex *cx, int incx)
```

Arguments

UPLO	Indicates whether the values in a matrix reside in the upper or lower triangle of the array in which the matrix is stored. The legal values for UPLO are listed below. Any value not listed below is illegal. 'L' or 'l' Only the lower triangle of the array will be referenced. 'U' or 'u' Only the upper triangle of the array will be referenced.
TRANSA	Indicates how to use the matrix stored in the array argument. The legal values for TRANSA are listed below. Any value not listed below is illegal. 'N' or 'n' Use the matrix as it is stored in the array. 'T' or 't' Use the transpose of the matrix. 'C' or 'c' Use the conjugate transpose of the matrix.

DIAG	Indicates whether or not a triangular matrix is unit triangular. If the matrix is unit triangular then none of the diagonal elements of the array will be accessed. The legal values for DIAG are listed below. Any value not listed below is illegal. 'N' or 'n' The matrix is not unit triangular. 'U' or 'u' The matrix is unit triangular.
N	Size of a matrix with N rows and N columns. $N \geq 0$.
xA	One-dimensional array that contains the input matrix. The matrix is in packed storage; the array must have at least enough elements to store the non-redundant elements of the matrix.
xY	Y and INCY describe a vector of length N. On entry, an input vector. On exit, a result vector.
INCY	Scalar that contains the storage spacing between successive elements of the vector Y. $INCY \neq 0$. If INCY is one then elements of the vector are contiguous in memory. INCY may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations. If Y is a one-dimensional array and $INCY = -1$ then the array will be accessed in reverse order. If Y is a two-dimensional array and $INCY = LDA$ then the vector will be a row of the array. If Y is a two-dimensional array and $INCY = LDA+1$ then the vector will be a diagonal of the array.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C      INTEGER          LENGTA, N
C      PARAMETER        (N = 3)
C      PARAMETER        (LENGTA = (N * N + N) / 2)

C      DOUBLE PRECISION AP(LENGTA), X(N)
C      INTEGER          I

C      EXTERNAL          DTPMV
C      INTRINSIC         DBLE

C      Initialize the array AP to store in packed triangular form
C      the triangular matrix A shown below. Initialize the array
C      X to store the vector x shown below.
C

```

```

C      1      3
C      A = 2  4      x = 2
C      3  5  6      1
C
      DO 10, I = 1, LENGTA
      AP(I) = DBLE (I)
10  CONTINUE
      DO 20, I = 1, N
      X(I) = DBLE (4 - I)
20  CONTINUE
      PRINT 1000
      PRINT 1010, AP(1)
      PRINT 1010, AP(2), AP(3)
      PRINT 1010, AP(4), AP(5), AP(6)
      PRINT 1020
      PRINT 1030, AP
      PRINT 1040
      PRINT 1050, X
      CALL DTPMV ('LOWER TRIANGULAR A', 'NO TRANSPOSE A',
$      'NOT UNIT DIAGONAL A', N, AP, X, 1)
      PRINT 1060
      PRINT 1050, X
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (1X, 3(2X, F5.1))
1020 FORMAT (/1X, 'A in packed form:')
1030 FORMAT (1X, 6(2X, F5.1))
1040 FORMAT (/1X, 'x:')
1050 FORMAT (3X, F5.1)
1060 FORMAT (/1X, 'Ax:')
C
      END

```

Sample Output

```

A in full form:
  1.0
  2.0   3.0
  4.0   5.0   6.0

A in packed form:
  1.0   2.0   3.0   4.0   5.0   6.0

x:
  3.0
  2.0
  1.0

```

Ax:
3.0
14.0
25.0

Solution to a Triangular System of Linear Equations in Packed Storage

The subroutines described in this section compute one of the following results for a triangular matrix A in packed storage and a vector y :

$$y \leftarrow A^{-1}y$$

$$y \leftarrow A^{-T}y$$

$$y \leftarrow \bar{A}^{-T}y$$

Calling Sequence

```
CALL DTPSV (UPLO, TRANSA, DIAG, N, DA, DY, INCY)
CALL STPSV (UPLO, TRANSA, DIAG, N, SA, SY, INCY)
CALL ZTPSV (UPLO, TRANSA, DIAG, N, ZA, ZY, INCY)
CALL CTPSV (UPLO, TRANSA, DIAG, N, CA, CY, INCY)
```

```
void dtpsv (char uplo, char trans, char diag, int n, double *da,
double *dx, int incx)
void stpsv (char uplo, char trans, char diag, int n, float *sa,
float *sx, int incx)
void ztpsv (char uplo, char trans, char diag, int n, doublecomplex
*za, doublecomplex *zx, int incx)
void ctpsv (char uplo, char trans, char diag, int n, complex *ca,
complex *cx, int incx)
```

Arguments

UPLO	Indicates whether the values in a matrix reside in the upper or lower triangle of the array in which the matrix is stored. The legal values for UPLO are listed below. Any value not listed below is illegal. 'L' or 'l' Only the lower triangle of the array will be referenced. 'U' or 'u' Only the upper triangle of the array will be referenced.
TRANSA	Indicates how to use the matrix stored in the array argument. The legal values for TRANSA are listed below. Any value not listed below is illegal. 'N' or 'n' Use the matrix as it is stored in the array. 'T' or 't' Use the transpose of the matrix. 'C' or 'c' Use the conjugate transpose of the matrix.

DIAG	Indicates whether or not a triangular matrix is unit triangular. If the matrix is unit triangular then none of the diagonal elements of the array will be accessed. The legal values for DIAG are listed below. Any value not listed below is illegal. 'N' or 'n' The matrix is not unit triangular. 'U' or 'u' The matrix is unit triangular.
N	Size of a matrix with N rows and N columns. $N \geq 0$.
xA	One-dimensional array that contains the input matrix. The matrix is in packed storage; the array must have at least enough elements to store the non-redundant elements of the matrix.
xY	Y and INCY describe a vector of length N. On entry, an input vector. On exit, a result vector.
INCY	Scalar that contains the storage spacing between successive elements of the vector Y. $INCY \neq 0$. If INCY is one then elements of the vector are contiguous in memory. INCY may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations. If Y is a one-dimensional array and $INCY = -1$ then the array will be accessed in reverse order. If Y is a two-dimensional array and $INCY = LDA$ then the vector will be a row of the array. If Y is a two-dimensional array and $INCY = LDA+1$ then the vector will be a diagonal of the array.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  DOUBLE PRECISION  ONE
  INTEGER            LENGTA, N
  PARAMETER          (N = 4)
  PARAMETER          (LENGTA = (N * N + N) / 2)
  PARAMETER          (ONE = 1.0D0)

C
  DOUBLE PRECISION AP(LENGTA), B(N)
  INTEGER I
  EXTERNAL DTPSV

C
C  Initialize the array AP to store in packed triangular form
C  the unit diagonal triangular matrix A shown below.
C  Initialize the array B to store the vector b shown below.
C

```

```

C      1.0      8.0
C      A = 2.0  1.0      b = 25.0
C      3.0  5.0  1.0      79.0
C      4.0  6.0  7.0  1.0      167.0
C
C      DATA AP / 8D8, 2.0D0, 3.0D0, 4.0D0, 8D8, 5.0D0,
$              6.0D0, 8D8, 7.0D0, 1.0D0 /
C      DATA B / 8.0D0, 2.5D1, 7.9D1, 1.67D2 /
C
C      PRINT 1000
C      PRINT 1010, ONE
C      PRINT 1010, AP(2), ONE
C      PRINT 1010, AP(3), AP(6), ONE
C      PRINT 1010, AP(4), AP(7), AP(9), ONE
C      PRINT 1020
C      PRINT 1030, (AP(I), I = 1, LENGTA)
C      PRINT 1040
C      PRINT 1050, (B(I), I = 1, N)
C      CALL DTPSV ('LOWER TRIANGULAR A', 'NOT TRANSPOSED A',
$              'UNIT DIAGONAL A', N, AP, B, 1)
C      PRINT 1060
C      PRINT 1050, (B(I), I = 1, N)
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (1X, 4(2X, F5.1))
1020 FORMAT (/1X, 'A in packed form: (* in unused elements)')
1030 FORMAT (1X, 9(2X, F5.1))
1040 FORMAT (/1X, 'b:')
1050 FORMAT (3X, F5.1)
1060 FORMAT (/1X, 'A**(-1) * b:')
C
      END

```

Sample Output

```

A in full form:
  1.0
  2.0  1.0
  3.0  5.0  1.0
  4.0  6.0  7.0  1.0

A in packed form: (* in unused elements)
*****  2.0  3.0  4.0  *****  5.0  6.0  *****  7.0
  1.0

b:
  8.0
 25.0
 79.0
167.0

```

```
A**(-1) * b:  
 8.0  
 9.0  
10.0  
11.0
```


Product of a Triangular Matrix and a Vector

The subroutines described in this section compute one of the following results for a triangular matrix A and a vector y :

$$y \leftarrow \alpha Ay$$

$$y \leftarrow \alpha A^T y$$

$$y \leftarrow \alpha \bar{A}^T y$$

Calling Sequence

```
CALL DTRMV (UPLO, TRANSA, DIAG, N, DA, LDA, DY, INCY)
CALL STRMV (UPLO, TRANSA, DIAG, N, SA, LDA, SY, INCY)
CALL ZTRMV (UPLO, TRANSA, DIAG, N, ZA, LDA, ZY, INCY)
CALL CTRMV (UPLO, TRANSA, DIAG, N, CA, LDA, CY, INCY)
```

```
void dtrmv (char uplo, char trans, char diag, int n, double *da,
            int lda, double *dx, int incx)
void strmv (char uplo, char trans, char diag, int n, float *sa,
            int lda, float *sx, int incx)
void ztrmv (char uplo, char trans, char diag, int n, doublecomplex
            *za, int lda, doublecomplex *zx, int incx)
void ctrmv (char uplo, char trans, char diag, int n, complex *ca,
            int lda, complex *cx, int incx)
```

Arguments

UPLO	Indicates whether the values in a matrix reside in the upper or lower triangle of the array in which the matrix is stored. The legal values for UPLO are listed below. Any value not listed below is illegal. 'L' or 'l' Only the lower triangle of the array will be referenced. 'U' or 'u' Only the upper triangle of the array will be referenced.
TRANSA	Indicates how to use the matrix stored in the array argument. The legal values for TRANSA are listed below. Any value not listed below is illegal. 'N' or 'n' Use the matrix as it is stored in the array. 'T' or 't' Use the transpose of the matrix. 'C' or 'c' Use the conjugate transpose of the matrix.

DIAG	Indicates whether or not a triangular matrix is unit triangular. If the matrix is unit triangular then none of the diagonal elements of the array will be accessed. The legal values for DIAG are listed below. Any value not listed below is illegal. 'N' or 'n' The matrix is not unit triangular. 'U' or 'u' The matrix is unit triangular.
N	Size of a matrix with N rows and N columns. $N \geq 0$.
xA	Two-dimensional array that contains the input matrix.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
xY	Y and INCY describe a vector of length N. On entry, an input vector. On exit, a result vector.
INCY	Scalar that contains the storage spacing between successive elements of the vector Y. $INCY \neq 0$. If INCY is one then elements of the vector are contiguous in memory. INCY may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations. If Y is a one-dimensional array and $INCY = -1$ then the array will be accessed in reverse order. If Y is a two-dimensional array and $INCY = LDA$ then the vector will be a row of the array. If Y is a two-dimensional array and $INCY = LDA + 1$ then the vector will be a diagonal of the array.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C      INTEGER          LDA, N
C      PARAMETER       (N = 3)
C      PARAMETER       (LDA = N)
C
C      DOUBLE PRECISION A(LDA,N), Y(N)
C      INTEGER          I, INCY, J
C
C      EXTERNAL DTRMV
C
C      Initialize the array A to store in triangular form the
C      matrix A shown below. Initialize the array Y to store
C      the vector y shown below.
C
C      1              7

```

```

C      A = 2  4      y = 8
C          3  5  6      9
C
C      DATA A / 1.0D0, 2.0D0, 3.0D0, 8D8, 4.0D0, 5.0D0, 8D8, 8D8,
$          6.0D0 /
C      DATA Y / 7.0D0, 8.0D0, 9.0D0 /
C
C      PRINT 1000
C      DO 10, I = 1, N
C          PRINT 1010, (A(I,J), J = 1, I)
10  CONTINUE
C      PRINT 1020
C      PRINT 1030, Y
C      INCY = 1
C      CALL DTRMV ('LOWER TRIANGULAR A', 'NOT TRANSPOSED A',
$              'NOT UNIT DIAGONAL A', N, A, LDA, Y, INCY)
C      PRINT 1040
C      PRINT 1030, Y
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (1X, 3(2X, F5.1))
1020 FORMAT (/1X, 'y:')
1030 FORMAT (1X, 2X, F5.1)
1040 FORMAT (/1X, 'A*y:')
C
C      END

```

Sample Output

```

A:
  1.0
  2.0   4.0
  3.0   5.0   6.0

y:
  7.0
  8.0
  9.0

A*y:
  7.0
 46.0
115.0

```

Solution to a Triangular System of Linear Equations

The subroutines described in this section compute one of the following results for a triangular matrix A and a vector y:

$$y \leftarrow A^{-1}y$$

$$y \leftarrow A^{-T}y$$

$$y \leftarrow \bar{A}^{-T}y$$

Calling Sequence

```
CALL DTRSV (UPLO, TRANSA, DIAG, N, DA, LDA, DY, INCY)
CALL STRSV (UPLO, TRANSA, DIAG, N, SA, LDA, SY, INCY)
CALL ZTRSV (UPLO, TRANSA, DIAG, N, ZA, LDA, ZY, INCY)
CALL CTRSV (UPLO, TRANSA, DIAG, N, CA, LDA, CY, INCY)
```

```
void dtrsv (char uplo, char trans, char diag, int n, double *da,
            int lda, double *dx, int incx)
void strsv (char uplo, char trans, char diag, int n, float *sa,
            int lda, float *sx, int incx)
void ztrsv (char uplo, char trans, char diag, int n, doublecomplex
            *za, int lda, doublecomplex *zx, int incx)
void ctrsv (char uplo, char trans, char diag, int n, complex *ca,
            int lda, complex *cx, int incx)
```

Arguments

UPLO	Indicates whether the values in a matrix reside in the upper or lower triangle of the array in which the matrix is stored. The legal values for UPLO are listed below. Any value not listed below is illegal. 'L' or 'l' Only the lower triangle of the array will be referenced. 'U' or 'u' Only the upper triangle of the array will be referenced.
TRANSA	Indicates how to use the matrix stored in the array argument. The legal values for TRANSA are listed below. Any value not listed below is illegal. 'N' or 'n' Use the matrix as it is stored in the array. 'T' or 't' Use the transpose of the matrix. 'C' or 'c' Use the conjugate transpose of the matrix.

DIAG	Indicates whether or not a triangular matrix is unit triangular. If the matrix is unit triangular then none of the diagonal elements of the array will be accessed. The legal values for DIAG are listed below. Any value not listed below is illegal. 'N' or 'n' The matrix is not unit triangular. 'U' or 'u' The matrix is unit triangular.
N	Size of a matrix with N rows and N columns. $N \geq 0$.
xA	Two-dimensional array that contains the input matrix.
LDA	Leading dimension of the array A as specified in a dimension or type statement. $LDA \geq \max(1, N)$.
xY	Y and INCY describe a vector of length N. On entry, an input vector. On exit, a result vector.
INCY	Scalar that contains the storage spacing between successive elements of the vector Y. $INCY \neq 0$. If INCY is one then elements of the vector are contiguous in memory. INCY may take on values besides 1 to allow the programmer to extract from a matrix a vector that is not stored in contiguous memory locations. If Y is a one-dimensional array and $INCY = -1$ then the array will be accessed in reverse order. If Y is a two-dimensional array and $INCY = LDA$ then the vector will be a row of the array. If Y is a two-dimensional array and $INCY = LDA + 1$ then the vector will be a diagonal of the array.

Sample Program

```
PROGRAM TEST
IMPLICIT NONE
C
C      INTEGER          LDA, N
C      PARAMETER       (N = 3)
C      PARAMETER       (LDA = N)
C
C      DOUBLE PRECISION A(LDA,N+1)
C      INTEGER          I
C
C      EXTERNAL DTRSV
C
C      Initialize the first 3x3 elements of the array A to store
C      in triangular form the matrix A shown below. Initialize
C      the last column of A to store the vector b shown below.
C
C      1              7
```

```

C      A = 2  4      b = 46      b is stored in the
C          3  5  7      115      last column of A
C
      DATA A /1.0D0, 2.0D0, 3.0D0, 8.0D8, 4.0D0, 5.0D0,
$          8.0D8, 8.0D8, 6.0D0, 7.0D0, 4.6D1, 1.15D2 /
      PRINT 1000
      PRINT 1010, A(1,1)
      PRINT 1010, A(2,1), A(2,2)
      PRINT 1010, A(3,1), A(3,2), A(3,3)
      PRINT 1020
      PRINT 1030, (A(I,4), I = 1, N)
      CALL DTRSV ('LOWER TRIANGULAR A', 'NOT TRANSPOSED A',
$              'NOT UNIT DIAGONAL A', N, A, LDA, A(1,4), 1)
      PRINT 1040
      PRINT 1030, (A(I,4), I = 1, N)
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (1X, 3(2X, F5.1))
1020 FORMAT (/1X, 'b:')
1030 FORMAT (3X, F5.1)
1040 FORMAT (/1X, 'A**(-1) * b:')
C
      END

```

Sample Output

```

A:
  1.0
  2.0   4.0
  3.0   5.0   6.0

b:
  7.0
 46.0
115.0

A**(-1) * b:
  7.0
  8.0
  9.0

```

Basic Linear Algebra Subprograms, Level 3 (BLAS3)

5 

This chapter describes the Basic Linear Algebra Subprograms Level 3 (BLAS3) as they are implemented in the Sun Performance Library. BLAS3 is a collection of linear algebra subroutines to perform common linear algebra operations with matrix operands. For more information about BLAS3, refer to the information in this chapter or see "A Set of Level 3 Basic Linear Algebra Subprograms" written by Dongarra *et al* and published in the *ACM Transactions on Mathematical Software* (Vol. 16, pp. 1-17).

The first four sections of this chapter describe:

- conventions followed in naming arguments and subroutines
- conventions used for passing arguments into or out of subroutines
- error handling done by the subroutines
- extensions to the standard versions of BLAS3.

The remaining sections of this chapter describe the operation of each of the BLAS3 subroutines. For each subroutine the interface is shown, the operation of the subroutine is described, and a short FORTRAN 77 program is given with sample output.

Naming Conventions

The types of all arguments follow the notation in Dongarra's original paper as shown below:

```

IMPLICIT COMPLEX (C)
IMPLICIT DOUBLE PRECISION (D)
IMPLICIT INTEGER (I-N)
IMPLICIT REAL*16 (Q)
IMPLICIT REAL (S)
IMPLICIT DOUBLE COMPLEX (Z)
CHARACTER SIDE, UPLO, TRANSA, TRANSB, DIAG

```

The first letter of a subroutine name identifies the type of the result. The second and third letters indicate the form of matrix involved. The forms are:

GE	General
HE	Hermitian
SY	Symmetric
TR	Triangular

The remaining letters indicate the operation that the subroutine performs. The operations are:

MM	Matrix-matrix product
RK	Rank-k update of a matrix
R2K	Rank-2k update of a matrix
SM	Solve a triangular system of linear equations with multiple right-hand sides

To specify the group of subroutines that perform the same operation but are of different types, this manual uses a lowercase x. For example, xGEMM refers to the subroutines DGEMM, SGEMM, ZGEMM, and CGEMM. This same convention is used for arguments: xALPHA refers to the arguments DALPHA, SALPHA, ZALPHA, and CALPHA.

A^{-1} denotes the inverse of the square matrix stored in the array A. Whenever the symbol A^{-1} appears it is assumed that the numerical inverse of A exists. A^T denotes the transpose of A and A^H denotes the conjugate transpose of A for a complex A.

Argument Conventions

Only the first character of each character argument is used. This means that you can specify for a character argument any string that begins with a legal character. For example, many of the sample programs pass the string 'UNIT DIAGONAL A' as the UNIT argument because it is easier for a reader to understand than simply 'U'.

Although the FORTRAN 77 standard prohibits aliasing arguments, the Sun Performance Library allows the programmer to alias two or more arguments as long as all aliased arguments have the same base type and none of the arguments being aliased is an output argument. A real variable and a complex variable have the same base type (real), while a real variable and a double precision variable have different base types. Aliasing arguments having different base types or aliasing any argument with an output argument will produce undefined results at best. It is not even guaranteed that these undefined results will be the same from one run to the next.

Error Handling

If the programmer calls a BLAS3 subroutine with an illegal argument value then the subroutine will call the error handling subroutine XERBLA. XERBLA will print the name of the subroutine in which the error was detected, the number of the first argument found to be in error, and the values of the scalar arguments. The values of the arguments with the same types are printed together. As an example, consider the following call to SGEMM:

```
CALL SGEMM ('N', 'ILLEGAL', -10, 20, 30, 5.0, A, 11, B, 31, 9.0, C, 12)
```

In the call above, the value of TRANSB is illegal. SGEMM will call XERBLA, which will print the report shown below and then terminate with a STOP statement.

```
** On entry to SGEMM parameter number 2 had an illegal value
CHARACTER PARAMETERS:
      N      I
INTEGER PARAMETERS:
      -10    20    30
```

	11	31	12
REAL	PARAMETERS:		
	0.5000E+01	0.9000E+01	
STOP			

The Sun Performance Library will detect all instances in which the calling subprogram specifies one or more illegal parameter values, but there are many types of interface errors that the Sun Performance Library will not detect. Included in the list of interface errors that the Sun Performance Library will not detect is specifying too many or too few arguments, specifying an argument of the wrong type, and aliasing an argument with the result argument.

Extensions to the Basic Linear Algebra Subprograms

The Sun Performance Library includes the double complex subprograms suggested in the original paper by Dongarra. Also, this version of Sun Performance Library contains a native C/C++ interface to BLAS. Conventions for this interface are described in detail in the preface to this manual.

Product of Two General Matrices

The subroutines described in this section compute one of the following results for the general matrices A, B, and C, and the scalars α and β :

Operation	TRANSA	TRANSB	Matrix A	Matrix B
$C \leftarrow \alpha AB + \beta C$	'N' or 'n'	'N' or 'n'	$m \times k$	$k \times n$
$C \leftarrow \alpha A^T B + \beta C$	'T' or 't'	'N' or 'n'	$k \times m$	$k \times n$
$C \leftarrow \alpha AB^T + \beta C$	'N' or 'n'	'T' or 't'	$m \times k$	$n \times k$
$C \leftarrow \alpha A^T B^T + \beta C$	'T' or 't'	'T' or 't'	$k \times m$	$n \times k$
$C \leftarrow \alpha A^H B + \beta C$	'C' or 'c'	'N' or 'n'	$k \times m$	$k \times n$
$C \leftarrow \alpha AB^H + \beta C$	'N' or 'n'	'C' or 'c'	$m \times k$	$n \times k$
$C \leftarrow \alpha A^H B^H + \beta C$	'C' or 'c'	'C' or 'c'	$k \times m$	$n \times k$
$C \leftarrow \alpha A^T B^H + \beta C$	'T' or 't'	'C' or 'c'	$k \times m$	$n \times k$
$C \leftarrow \alpha A^H B^T + \beta C$	'C' or 'c'	'T' or 't'	$k \times m$	$n \times k$

Calling Sequence

CALL DGEMM	(TRANSA, TRANSB, M, N, K, DALPHA, DA, LDA, DB, LDB, DBETA, DC, LDC)
CALL SGEMM	(TRANSA, TRANSB, M, N, K, SALPHA, SA, LDA, SB, LDB, SBETA, SC, LDC)
CALL ZGEMM	(TRANSA, TRANSB, M, N, K, ZALPHA, ZA, LDA, ZB, LDB, ZBETA, ZC, LDC)
CALL CGEMM	(TRANSA, TRANSB, M, N, K, CALPHA, CA, LDA, CB, LDB, CBETA, CC, LDC)
void dgemm	(char transa, char transb, int m, int n, int k, double dalpha, double *da, int lda, double *db, int ldb, double dbeta, double *dc, int ldc)
void sgemm	(char transa, char transb, int m, int n, int k, float salpha, float *sa, int lda, float *sb, int ldb, float sbeta, float *sc, int ldc)
void zgemm	(char transa, char transb, int m, int n, int k, doublecomplex *zalpha, doublecomplex *za, int lda, doublecomplex *zb, int ldb, doublecomplex *zbeta, doublecomplex *zc, int ldc)
void cgemm	(char transa, char transb, int m, int n, int k, complex *calpha, complex *ca, int lda, complex *cb, int ldb, complex *cbeta, complex *cc, int ldc)

Arguments

TRANSA	Indicates how to use the matrix stored in the array A. The legal values for TRANSA are listed below. Any value not listed below is illegal. 'N' or 'n' Use the matrix as it is stored in A. 'T' or 't' Use the transpose of the matrix stored in A. 'C' or 'c' Use the conjugate transpose of the matrix stored in A.
TRANSB	Indicates how to use the matrix stored in the array B. The legal values for TRANSB are listed below. Any value not listed below is illegal. 'N' or 'n' Use the matrix as it is stored in B. 'T' or 't' Use the transpose of the matrix stored in B. 'C' or 'c' Use the conjugate transpose of the matrix stored in B.
M, N, K	Indicate the size of the matrices as shown in the table above. $M, N, K \geq 0$.
xALPHA	Scalar that scales the input value of the matrix in A.
xA	Two-dimensional array in which one of the input matrices is stored.
LDA	Leading dimension of the array A as specified in the dimension or type statement. $LDA \geq \max(1, K)$. If $TRANSA = 'N'$ or $'n'$, $LDA \geq \max(1, M)$.
xB	Two-dimensional array in which one of the input matrices is stored.
LDB	Leading dimension of the array B as specified in the dimension or type statement. $LDB \geq \max(1, N)$. If $TRANSB = 'N'$ or $'n'$, $LDB \geq \max(1, K)$.
xBETA	Scalar that scales the input value of the matrix stored in C.
xC	An $M \times N$ array. On entry, an input matrix. On exit, the result matrix.
LDC	Leading dimension of the array C as specified in the dimension or type statement. $LDC \geq \max(1, M)$.

Sample Program

```

C      PROGRAM TEST
C      IMPLICIT NONE
C
C      INTEGER          K, LDA, LDB, LDC, M, N
C      PARAMETER        (K = 3)
C      PARAMETER        (M = 3)
C      PARAMETER        (N = 2)
C      PARAMETER        (LDA = M)
C      PARAMETER        (LDB = K)
C      PARAMETER        (LDC = M)

```

```

C      DOUBLE PRECISION  A(LDA,K), ALPHA, B(LDB,N), BETA, C(LDC,N)
C      INTEGER          I, J
C
C      EXTERNAL          DGEMM
C
C      Initialize the arrays A, B, and C to store the matrices A,
C      B, and C shown below.
C
C      A =   0  0  1      2  5      100  100
C           0  1  0      3  6      100  100
C           1  0  0      4  7      100  100
C
C      DATA A / 0.0D0, 0.0D0, 1.0D0, 0.0D0, 1.0D0, 0.0D0,
$           1.0D0, 0.0D0, 0.0D0 /
C      DATA B / 2.0D0, 3.0D0, 4.0D0, 5.0D0, 6.0D0, 7.0D0 /
C      DATA C / 6*1.0D2 /
C
C      PRINT 1000
C      PRINT 1010, ((A(I,J), J = 1, K), I = 1, M)
C      PRINT 1020
C      PRINT 1030, ((B(I,J), J = 1, N), I = 1, K)
C      PRINT 1040
C      PRINT 1030, ((C(I,J), J = 1, N), I = 1, M)
C      ALPHA = 2.0D0
C      BETA = 3.0D0
C      CALL DGEMM ('NOT TRANSPOSED A', 'NOT TRANSPOSED B', M, N,
$      K, ALPHA, A, LDA, B, LDB, BETA, C, LDC)
C      PRINT 1050
C      PRINT 1030, ((C(I,J), J = 1, N), I = 1, M)
C
C      1000 FORMAT (1X, 'A:')
C      1010 FORMAT (3(3X, F5.1))
C      1020 FORMAT (/1X, 'B:')
C      1030 FORMAT (2(3X, F5.1))
C      1040 FORMAT (/1X, 'C:')
C      1050 FORMAT (/1X, 'AB + C')
C
C      END

```

Sample Output

```

A:
  0.0    0.0    1.0
  0.0    1.0    0.0
  1.0    0.0    0.0

B:
  2.0    5.0
  3.0    6.0
  4.0    7.0

```

C:		
100.0	100.0	
100.0	100.0	
100.0	100.0	
AB + C		
308.0	314.0	
306.0	312.0	
304.0	310.0	

Product of a Hermitian Matrix and a General Matrix

The subroutines described in this section compute one of the following results for the Hermitian matrix A, the general matrices B and C, and the scalars α and β :

Operation	SIDE	Matrix A	Matrix B
$C \leftarrow \alpha AB + \beta C$	'L' or 'l'	$m \times m$	$m \times n$
$C \leftarrow \alpha BA + \beta C$	'R' or 'r'	$n \times n$	$m \times n$

Calling Sequence

CALL ZHEMM	(SIDE, UPLO, M, N, ZALPHA, ZA, LDA, ZB, LDB, ZBETA, ZC, LDC)
CALL CHEMM	(SIDE, UPLO, M, N, CALPHA, CA, LDA, CB, LDB, CBETA, CC, LDC)
void zhemm	(char side, char uplo, int m, int n, doublecomplex *zalp, doublecomplex *za, int lda, doublecomplex *b, int ldb, doublecomplex *zbeta, doublecomplex *zc, int ldc)
void chemm	(char side, char uplo, int m, int n, complex *calp, complex *ca, int lda, complex *cb, int ldb, complex *cbeta, complex *cc, int ldc)

Arguments

SIDE	Indicates whether the product involving matrices A and B should be formed as AB or BA. The legal values for SIDE are listed below. Any value not listed below is illegal. 'L' or 'l' Compute AB. 'R' or 'r' Compute BA.
UPLO	Indicates whether the values in a matrix reside in the upper or lower triangle of the array in which it is stored. The legal values for UPLO are listed below. Any value not listed below is illegal. 'L' or 'l' Only lower triangle of array will be referenced. 'U' or 'u' Only upper triangle of array will be referenced.
M, N	Indicate the size of the matrices as shown in the table above.
xALPHA	Scalar that scales the input value of the matrix stored in A.
xA	Two-dimensional array in which one of the input matrices is stored.
LDA	Leading dimension of the array A as specified in the dimension or type statement. $LDA \geq \max(1, N)$.

	If SIDE = 'L' or 'I', $LDA \geq \max(1, M)$.
xB	Two-dimensional array in which one of the input matrices is stored.
LDB	Leading dimension of the array B as specified in the dimension or type statement. $LDB \geq \max(1, M)$.
xBETA	Scalar that scales the input value of the matrix stored in C.
xC	Two-dimensional array. On entry, an input matrix. On exit, the result matrix.
LDC	Leading dimension of the array C as specified in the dimension or type statement. $LDC \geq \max(1, M)$.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
REAL      RZERO
INTEGER   LDA, LDB, LDC, M, N
PARAMETER (M = 3)
PARAMETER (N = 2)
PARAMETER (LDA = M)
PARAMETER (LDB = M)
PARAMETER (LDC = M)
PARAMETER (RZERO = 0.0E0)

C
COMPLEX   A(LDA,M), ALPHA, B(LDB,N), BETA, C(LDC,N)
INTEGER   I, J

C
EXTERNAL  CHEMM
INTRINSIC CMPLX, CONJG, REAL

C
C Initialize the array A to store in Hermitian form the matrix
C A shown below. Initialize the arrays B and C to store the
C matrices B and C shown below.
C
C      1+0i  2-3i  4-5i      0-1i  0-2i      100+100i  100+100i
C  A = 2+3i  6+0i  7-8i  B = 0-1i  0-2i  C = 100+100i  100+100i
C      4+5i  7+8i  9+0i      0-1i  0-2i      100+100i  100+100i
C
C      DATA A / (1.0,8E8), (2.0,3.0), (4.0,5.0),
C      $          (8E8,8E8), (6.0,8E8), (7.0,8.0),
C      $          (8E8,8E8), (8E8,8E8), (9.0,8E8) /
C      DATA B / 3*(0.0D0,-1.0D0), 3*(0.0D0, -2.0D0) /
C      DATA C / 6*(1.0E2,1.0E2) /

C
ALPHA = CMPLX (1.0E0,0.0E0)
BETA = CMPLX (5.0E0,0.0E0)
PRINT 1000
PRINT 1010, REAL (A(1,1)), RZERO, CONJG (A(2,1)),

```



```

$   CONJG (A(3,1))
   PRINT 1010, A(2,1), REAL (A(2,2)), RZERO, CONJG (A(3,2))
   PRINT 1010, A(3,1), A(3,2), REAL (A(3,3)), RZERO
   PRINT 1020
   PRINT 1010, ((A(I,J), J = 1, M), I = 1, M)
   PRINT 1030
   PRINT 1040, ((B(I,J), J = 1, N), I = 1, M)
   PRINT 1050
   PRINT 1040, ((C(I,J), J = 1, N), I = 1, M)
   CALL CHEMM ('LEFT SIDE A', 'LOWER TRIANGULAR A', M, N,
$   ALPHA, A, LDA, B, LDB, BETA, C, LDC)
   PRINT 1060
   PRINT 1040, ((C(I,J), J = 1, N), I = 1, M)
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (3(3X, '(', F5.1, ', ', F5.1, ')'))
1020 FORMAT (/1X, 'A in Hermitian form: (* in unused elements)')
1030 FORMAT (/1X, 'B:')
1040 FORMAT (2(3X, '(', F5.1, ', ', F5.1, ')'))
1050 FORMAT (/1X, 'C:')
1060 FORMAT (/1X, 'AB + C:')
C
   END

```

Sample Output

```

A in full form:
( 1.0, 0.0) ( 2.0, -3.0) ( 4.0, -5.0)
( 2.0, 3.0) ( 6.0, 0.0) ( 7.0, -8.0)
( 4.0, 5.0) ( 7.0, 8.0) ( 9.0, 0.0)

A in Hermitian form: (* in unused elements)
( 1.0, ***** ) ( ***** , ***** ) ( ***** , ***** )
( 2.0, 3.0 ) ( 6.0 , ***** ) ( ***** , ***** )
( 4.0, 5.0 ) ( 7.0, 8.0 ) ( 9.0 , ***** )

B:
( 0.0, -1.0) ( 0.0, -2.0)
( 0.0, -1.0) ( 0.0, -2.0)
( 0.0, -1.0) ( 0.0, -2.0)

C:
(100.0,100.0) (100.0,100.0)
(100.0,100.0) (100.0,100.0)
(100.0,100.0) (100.0,100.0)

AB + C:
(492.0,493.0) (484.0,486.0)
(495.0,485.0) (490.0,470.0)
(513.0,480.0) (526.0,460.0)

```

Rank-k Update of a Hermitian Matrix

The subroutines described in this section compute one of the following results for a general matrix A and a Hermitian matrix C:

Operation	TRANS A	Matrix A
$C \leftarrow \alpha A A^H + \beta C$	'N' or 'n'	$n \times k$
$C \leftarrow \alpha A^H A + \beta C$	'C' or 'c'	$k \times n$

Calling Sequence

CALL ZHERK	(UPLO, TRANS A, N, K, DALPHA, ZA, LDA, DBETA, ZC, LDC)
CALL CHERK	(UPLO, TRANS A, N, K, SALPHA, CA, LDA, SBETA, CC, LDC)
void zherk	(char uplo, char trans, int n, int k, double dalpha, doublecomplex *za, int lda, double dbeta, doublecomplex *zc, int ldc)
void cherk	(char uplo, char trans, int n, int k, float salpha, complex *ca, int lda, float sbeta, complex *cc, int ldc)

Arguments

UPLO	Indicates whether the values in a matrix reside in the upper or lower triangle of the array in which it is stored. The legal values for UPLO are listed below. Any value not listed below is illegal. 'L' or 'l' Only lower triangle of array will be referenced. 'U' or 'u' Only upper triangle of array will be referenced.
TRANS A	Indicates how to use the matrix stored in the array A. The legal values for TRANS A are listed below. Any value not listed below is illegal. 'N' or 'n' Use the matrix as it is stored in A. 'C' or 'c' Use the conjugate transpose of the matrix stored in A.
N, K	Indicate the size of matrix A as shown in the table above. $N, K \geq 0$.
xALPHA	Scalar that scales the input value of the matrix stored in A. Note that xALPHA is real.
xA	Two-dimensional array in which one of the input matrices is stored.
LDA	Leading dimension of the array A as specified in the dimension or type statement. $LDA \geq \max(1, K)$. If TRANS A = 'N' or 'n', $LDA \geq \max(1, N)$.

xBETA	Scalar that scales the input value of the matrix stored in C. Note that xBETA is real.
xC	Two-dimensional array. On entry, an input matrix. On exit, the result matrix.
LDC	Leading dimension of the array C as specified in the dimension or type statement. $LDC \geq \max(1, N)$

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
REAL      RZERO
INTEGER    K, LDA, LDC, N
PARAMETER (K = 2)
PARAMETER (N = 3)
PARAMETER (LDA = N)
PARAMETER (LDC = N)
PARAMETER (RZERO = 0.0E0)
C
COMPLEX    A(LDA,K), ALPHA, BETA, C(LDC,N)
INTEGER    I, J
C
EXTERNAL   CHERK
INTRINSIC  CMPLX, CONJG, REAL
C
C      Initialize the array A to store the matrix A shown below.
C      Initialize the array C to store in symmetric form the
C      Hermitian matrix C shown below.
C
C      A =
C      1+1i  1+2i      1+0i  2-3i  4-5i
C      2+1i  2+2i      2+3i  6+0i  7-8i
C      3+1i  3+2i      4+5i  7+8i  9+0i
C
C      DATA A / (1.0,1.0), (2.0,1.0), (3.0,1.0),
C      $          (1.0,2.0), (2.0,2.0), (3.0,2.0) /
C      DATA C / (1.0,8E8), (2.0,3.0), (4.0,5.0),
C      $          (8E8,8E8), (6.0,8E8), (7.0,8.0),
C      $          (8E8,8E8), (8E8,8E8), (9.0,8E8) /
C
ALPHA = CMPLX (1.0E0,0.0E0)
BETA = CMPLX (5.0E0,0.0E0)
PRINT 1000
DO 100, I = 1, N
    PRINT 1010, (A(I,J), J = 1, K)
100 CONTINUE
PRINT 1020
DO 110, I = 1, N
    PRINT 1010, (C(I,J), J = 1, I - 1), REAL (C(I,I)), RZERO,

```

```

$          (CONJG (C(J,I)), J = I + 1, N)
110 CONTINUE
    PRINT 1030
    PRINT 1010, ((C(I,J), J = 1, 3), I = 1, 3)
    CALL CHERK ('LOWER TRIANGULAR C', 'NOT TRANSPOSED C', N, K,
$             ALPHA, A, LDA, BETA, C, LDC)
    PRINT 1040
    DO 120, I = 1, N
        PRINT 1010, (C(I,J), J = 1, I), (CONJG (C(J,I)),
$             J = I + 1, N)
    120 CONTINUE
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (3(3X, : '(', F5.1, ', ', F5.1, ')'))
1020 FORMAT (/1X, 'C in full form:')
1030 FORMAT (/1X, 'C in Hermitian form: (* in unused elements)')
1040 FORMAT (/1X, 'AA' + C:')
C
    END

```

Sample Output

```

A:
( 1.0, 1.0) ( 1.0, 2.0)
( 2.0, 1.0) ( 2.0, 2.0)
( 3.0, 1.0) ( 3.0, 2.0)

C in full form:
( 1.0, 0.0) ( 2.0, -3.0) ( 4.0, -5.0)
( 2.0, 3.0) ( 6.0, 0.0) ( 7.0, -8.0)
( 4.0, 5.0) ( 7.0, 8.0) ( 9.0, 0.0)

C in Hermitian form: (* in unused elements)
( 1.0, ****) (****, ****) (****, ****)
( 2.0, 3.0) ( 6.0, ****) (****, ****)
( 4.0, 5.0) ( 7.0, 8.0) ( 9.0, ****)

AA' + C:
( 12.0, 0.0) ( 19.0, -12.0) ( 31.0, -19.0)
( 19.0, 12.0) ( 43.0, 0.0) ( 52.0, -37.0)
( 31.0, 19.0) ( 52.0, 37.0) ( 68.0, 0.0)

```

Rank-2k Update of a Hermitian Matrix

The subroutines described in this section compute one of the following results for general matrices A and B and the Hermitian matrix C:

Operation	TRANS A	Matrices A and B
$C \leftarrow \alpha AB^H + \bar{\alpha} BA^H + \beta C$	'N' or 'n'	n × k
$C \leftarrow \alpha A^H B + \bar{\alpha} B^H A + \beta C$	'C' or 'c'	k × n

Calling Sequence

```
CALL ZHER2K (UPLO, TRANS A, N, K, ZALPHA, ZA, LDA, ZB, LDB, DBETA,
             ZC, LDC)
CALL CHER2K (UPLO, TRANS A, N, K, CALPHA, CA, LDA, CB, LDB, SBETA,
             CC, LDC)

void zher2k (char uplo, char trans, int n, int k, doublecomplex
            *zalp, doublecomplex *za, int lda, doublecomplex
            *zb, int ldb, double dbeta, doublecomplex *zc, int ldc)
void cher2k (char uplo, char trans, int n, int k, complex *calp,
            complex *ca, int lda, complex *cb, int ldb, float
            sbeta, complex *cc, int ldc)
```

Arguments

UPLO	Indicates whether the values in a matrix reside in the upper or lower triangle of the array in which it is stored. The legal values for UPLO are listed below. Any value not listed below is illegal. 'L' or 'l' Only lower triangle of array will be referenced. 'U' or 'u' Only upper triangle of array will be referenced.
TRANS A	Indicates how to use the matrix. The legal values for TRANS A are listed below. Any value not listed below is illegal. 'N' or 'n' Use the matrix as it is stored. 'C' or 'c' Use the conjugate transpose of the matrix.
N, K	Indicate the size of matrix A as shown in the table above. N, K ≥ 0.
xALPHA	Scalar that scales the input value of the matrix stored in A.
xA	Two-dimensional array in which one of the input matrices is stored.
LDA	Leading dimension of the array A as specified in the dimension or type statement. LDA ≥ max (1,K). If TRANS A = 'N' or 'n', LDA ≥ max (1,N).
xB	Two-dimensional array in which one of the input matrices is stored.

LDB	Leading dimension of the array B as specified in the dimension or type statement. $LDB \geq \max(1, K)$. If $TRANSB = 'N'$ or $'n'$, $LDB \geq \max(1, N)$.
xBETA	Scalar that scales the input value of the matrix stored in C. Note that xBETA is real.
xC	Two-dimensional array. On entry, an input matrix. On exit, the result matrix.
LDC	Leading dimension of the array C as specified in the dimension or type statement. $LDC \geq \max(1, N)$.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
REAL      RZERO
INTEGER   K, LDA, LDB, LDC, N
PARAMETER (K = 2)
PARAMETER (N = 3)
PARAMETER (LDA = N)
PARAMETER (LDB = N)
PARAMETER (LDC = N)
PARAMETER (RZERO = 0.0E0)
C
COMPLEX   A(LDA,K), ALPHA, B(LDB,K), BETA, C(LDC,N)
INTEGER   I, J
C
EXTERNAL  CHER2K
INTRINSIC CMPLX, CONJG, REAL
C
C      Initialize the arrays A and B to store the matrices A and B
C      shown below. Initialize the array C to store in symmetric
C      form the Hermitian matrix C shown below.
C
C      A =  $\begin{bmatrix} 1+1i & 1+2i \\ 2+1i & 2+2i \\ 3+1i & 3+2i \end{bmatrix}$  B =  $\begin{bmatrix} 2+1i & 3+2i & 4+3i \\ 3+1i & 4+2i & 5+3i \\ 4+1i & 5+2i & 6+3i \end{bmatrix}$  C =  $\begin{bmatrix} 1+0i & 2-3i & 4-5i \\ 2+3i & 6+0i & 7-8i \\ 4+5i & 7+8i & 9+0i \end{bmatrix}$ 
C
C      DATA A / (1.0,1.0), (2.0,1.0), (3.0,1.0),
C      $          (1.0,2.0), (2.0,2.0), (3.0,2.0) /
C      DATA B / (2.0,1.0), (3.0,1.0), (4.0,1.0), (3.0,2.0),
C      $          (4.0,2.0), (5.0,2.0) /
C      DATA C / (1.0,8E8), (2.0,3.0), (4.0,5.0),
C      $          (8E8,8E8), (6.0,8E8), (7.0,8.0),
C      $          (8E8,8E8), (8E8,8E8), (9.0,8E8) /
C
C      ALPHA = CMPLX (1.0E0,0.0E0)
C      BETA = CMPLX (0.0E0,3.0E0)

```

```

        PRINT 1000
        DO 100, I = 1, N
            PRINT 1010, (A(I,J), J = 1, K)
100    CONTINUE
        PRINT 1020
        DO 110, I = 1, N
            PRINT 1010, (B(J,I), J = 1, K)
110    CONTINUE
        PRINT 1030
        DO 120, I = 1, N
            PRINT 1010, (C(I,J), J = 1, I - 1), REAL (C(I,I)), RZERO,
$          (CONJG (A(J,I)), J = I + 1, N)
120    CONTINUE
        PRINT 1040
        DO 130, I = 1, N
            PRINT 1010, (C(I,J), J = 1, N)
130    CONTINUE
        CALL CHER2K ('LOWER TRIANGULAR C', 'NOT TRANSPOSED C', N, K,
$          ALPHA, A, LDA, B, LDB, BETA, C, LDC)
        PRINT 1050
        DO 140, I = 1, N
            PRINT 1010, (C(I,J), J = 1, I), (CONJG (C(J,I)),
$          J = I + 1, N)
140    CONTINUE
        PRINT 1060
        DO 150, I = 1, N
            PRINT 1010, (C(I,J), J = 1, N)
150    CONTINUE
C
1000  FORMAT (1X, 'A:')
1010  FORMAT (3(3X, : '(', F5.1, ', ', F5.1, ')'))
1020  FORMAT (/1X, 'B:')
1030  FORMAT (/1X, 'C in full form:')
1040  FORMAT (/1X, 'C in Hermitian form: (* in unused elements)')
1050  FORMAT (/1X, 'AB'' + BA'' + C in full form:')
1060  FORMAT (/1X, 'AB'' + BA'' + C in Hermitian form:')
C
        END

```

Sample Output

```

A:
(  1.0,  1.0)  (  1.0,  2.0)
(  2.0,  1.0)  (  2.0,  2.0)
(  3.0,  1.0)  (  3.0,  2.0)

B:
(  2.0,  1.0)  (  3.0,  1.0)
(  3.0,  2.0)  (  4.0,  2.0)
(  1.0,  ***** )  (  2.0,  3.0)

```

```

C in full form:
( 1.0, 0.0) ( 2.0, -1.0) ( 3.0, -1.0)
( 2.0, 3.0) ( 6.0, 0.0) ( 3.0, -2.0)
( 4.0, 5.0) ( 7.0, 8.0) ( 9.0, 0.0)

C in Hermitian form: (* in unused elements)
( 1.0, *****) (***** ,***** ) (***** ,***** )
( 2.0, 3.0) ( 6.0, *****) (***** ,***** )
( 4.0, 5.0) ( 7.0, 8.0) ( 9.0, *****)

AB' + BA' + C in full form:
( 20.0, 0.0) ( 27.0, 6.0) ( 34.0, 12.0)
( 27.0, -6.0) ( 38.0, 0.0) ( 49.0, 6.0)
( 34.0, -12.0) ( 49.0, -6.0) ( 64.0, 0.0)

AB' + BA' + C in Hermitian form:
( 20.0, 0.0) (***** ,***** ) (***** ,***** )
( 27.0, -6.0) ( 38.0, 0.0) (***** ,***** )
( 34.0, -12.0) ( 49.0, -6.0) ( 64.0, 0.0)

```


Product of a Symmetric Matrix and a General Matrix

The subroutines described in this section compute one of the following results for the symmetric matrix A, the general matrices B and C, and the scalars α and β :

Operation	SIDE	Matrix A	Matrix B
$C \leftarrow \alpha AB + \beta C$	'L' or 'l'	m × m	m × n
$C \leftarrow \alpha BA + \beta C$	'R' or 'r'	n × n	m × n

Calling Sequence

CALL DSYMM	(SIDE, UPLO, M, N, DALPHA, DA, LDA, DB, LDB, DBETA, DC, LDC)
CALL SSYMM	(SIDE, UPLO, M, N, SALPHA, SA, LDA, SB, LDB, SBETA, SC, LDC)
CALL ZSYMM	(SIDE, UPLO, M, N, ZALPHA, ZA, LDA, ZB, LDB, ZBETA, ZC, LDC)
CALL CSYMM	(SIDE, UPLO, M, N, CALPHA, CA, LDA, CB, LDB, CBETA, CC, LDC)
void dsymm	(char side, char uplo, int m, int n, double dalpha, double *da, int lda, double *db, int ldb, double dbeta, double *dc, int ldc)
void ssymm	(char side, char uplo, int m, int n, float salpha, float *sa, int lda, float *sb, int ldb, float sbeta, float *sc, int ldc)
void zsymm	(char side, char uplo, int m, int n, doublecomplex *zalpha, doublecomplex *za, int lda, doublecomplex *b, int ldb, doublecomplex *zbeta, doublecomplex *zc, int ldc)
void csymm	(char side, char uplo, int m, int n, complex *calpha, complex *ca, int lda, complex *cb, int ldb, complex *cbeta, complex *cc, int ldc)

Arguments

SIDE	Indicates whether the product involving matrices A and B should be formed as AB or BA. The legal values for SIDE are listed below. Any value not listed below is illegal.
'L' or 'l'	Compute AB.
'R' or 'r'	Compute BA.

UPLO	Indicates whether the values in a matrix reside in the upper or lower triangle of the array in which it is stored. The legal values for UPLO are listed below. Any value not listed below is illegal. 'L' or 'l' Only lower triangle of array will be referenced. 'U' or 'u' Only upper triangle of array will be referenced.
M, N	Indicate the size of the matrices as shown in the table above.
xALPHA	Scalar that scales the input value of the matrix stored in A.
xA	Two-dimensional array in which one of the input matrices is stored.
LDA	Leading dimension of the array A as specified in the dimension or type statement. $LDA \geq \max(1, N)$. If SIDE = 'L' or 'l', $LDA \geq \max(1, M)$.
xB	Two-dimensional array in which one of the input matrices is stored.
LDB	Leading dimension of the array B as specified in the dimension or type statement. $LDB \geq \max(1, M)$.
xBETA	Scalar that scales the input value of the matrix stored in C.
xC	Two-dimensional array. On entry, an input matrix. On exit, the result matrix.
LDC	Leading dimension of the array C as specified in the dimension or type statement. $LDC \geq \max(1, M)$.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
C   INTEGER          LDA, LDB, LDC, M, N
C   PARAMETER        (M = 3)
C   PARAMETER        (N = 2)
C   PARAMETER        (LDA = M)
C   PARAMETER        (LDB = M)
C   PARAMETER        (LDC = M)
C
C   DOUBLE PRECISION A(LDA,M), ALPHA, B(LDB,N), BETA, C(LDC,N)
C   INTEGER          I, J
C
C   EXTERNAL          DSymm
C
C   Initialize the array A to store in symmetric form the
C   symmetric matrix A shown below. Initialize the arrays
C   B and C to store the matrices B and C shown below.
C
C   A =   1  2  3   B =   0  1   C =   100  100
C         2  4  5     1  0     100  100
C         3  5  6     2  2     100  100
C

```

```

      DATA A / 1.0D0, 2.0D0, 3.0D0, 8D8, 4.0D0, 5.0D0, 8D8, 8D8,
$      6.0D0 /
      DATA B / 0.0D0, 1.0D0, 2.0D0, 1.0D0, 0.0D0, 2.0D0 /
      DATA C / 6*1.0D2 /
C
      ALPHA = 1.0D0
      BETA = 1.0D0
      PRINT 1000
      DO 100, I = 1, M
        PRINT 1010, (A(I,J), J = 1, I), (A(J,I), J = I + 1, M)
100  CONTINUE
      PRINT 1020
      PRINT 1010, ((A(I,J), J = 1, M), I = 1, M)
      PRINT 1030
      PRINT 1040, ((B(I,J), J = 1, N), I = 1, M)
      PRINT 1050
      PRINT 1040, ((C(I,J), J = 1, N), I = 1, M)
      CALL DSYMM ('LEFT SIDE A', 'LOWER TRIANGULAR A', M, N,
$      ALPHA, A, LDA, B, LDB, BETA, C, LDC)
      PRINT 1060
      PRINT 1040, ((C(I,J), J = 1, N), I = 1, M)
C
1000 FORMAT (1X, 'A in full form:')
1010 FORMAT (3(3X, F5.1))
1020 FORMAT (/1X, 'A in symmetric form: (* in unused elements)')
1030 FORMAT (/1X, 'B:')
1040 FORMAT (2(3X, F5.1))
1050 FORMAT (/1X, 'C:')
1060 FORMAT (/1X, 'AB + C:')
C
      END

```

Sample Output

```

A in full form:
  1.0    2.0    3.0
  2.0    4.0    5.0
  3.0    5.0    6.0

A in symmetric form: (* in unused elements)
  1.0    *****
  2.0    4.0    *****
  3.0    5.0    6.0

B:
  0.0    1.0
  1.0    0.0
  2.0    2.0

```

C:	
100.0	100.0
100.0	100.0
100.0	100.0
AB + C:	
108.0	107.0
114.0	112.0
117.0	115.0

Rank-k Update of a Symmetric Matrix

This subroutines described in this section compute the rank-k update to a symmetric (or Hermitian) matrix. SSYRK, DSYRK, and CSYRK are subprograms to compute the following functions of the general matrix A and the symmetric matrix C:

Operation	TRANSA	Matrix A
$C \leftarrow \alpha AA^T + \beta C$	'N' or 'n'	$n \times k$
$C \leftarrow \alpha A^T A + \beta C$	'T' or 't'	$k \times n$

Calling Sequence

```
CALL DSYRK (UPLO, TRANSA, N, K, DALPHA, DA, LDA, DBETA, DC, LDC)
CALL SSYRK (UPLO, TRANSA, N, K, SALPHA, SA, LDA, SBETA, SC, LDC)
CALL ZSYRK (UPLO, TRANSA, N, K, ZALPHA, ZA, LDA, ZBETA, ZC, LDC)
CALL CSYRK (UPLO, TRANSA, N, K, CALPHA, CA, LDA, CBETA, CC, LDC)

void dsyrk (char uplo, char trans, int n, int k, double dalpha,
double *da, int lda, double dbeta, double *dc, int ldc)
void ssyrk (char uplo, char trans, int n, int k, float salpha,
float *sa, int lda, float sbeta, float *sc, int ldc)
void zsyk (char uplo, char trans, int n, int k, doublecomplex
*zalpha, doublecomplex *za, int lda, doublecomplex
*zbeta, doublecomplex *zc, int ldc)
void csyrk (char uplo, char trans, int n, int k, complex *calpha,
complex *ca, int lda, complex *cbeta, complex *cc, int
ldc)
```

Arguments

UPLO	Indicates whether the values in a matrix reside in the upper or lower triangle of the array in which it is stored. The legal values for UPLO are listed below. Any value not listed below is illegal. 'L' or 'l' Only lower triangle of array will be referenced. 'U' or 'u' Only upper triangle of array will be referenced.
TRANSA	Indicates how to use the matrix stored in the array A. The legal values for TRANSA are listed below. Any value not listed below is illegal. 'N' or 'n' Use the matrix as it is stored in A. 'T' or 't' Use the transpose of the matrix stored in A

N, K	Indicate the size of matrix A as shown in the table above. $N, K \geq 0$.
xALPHA	Scalar that scales the input value of the matrix stored in A.
xA	Two-dimensional array in which one of the input matrix is stored.
LDA	Leading dimension of the array A as specified in the dimension or type statement. $LDA \geq \max(1, K)$. If $TRANSA = 'N'$ or $'n'$, $LDA \geq \max(1, N)$.
xBETA	Scalar that scales the input value of the matrix stored in C.
xC	Two-dimensional array. On entry, an input matrix. On exit, the result matrix.
LDC	Leading dimension of the array C as specified in the dimension or type statement. $LDC \geq \max(1, N)$.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE
C
C      INTEGER          K, LDA, LDC, N
C      PARAMETER        (K = 2)
C      PARAMETER        (N = 3)
C      PARAMETER        (LDA = N)
C      PARAMETER        (LDC = N)
C
C      DOUBLE PRECISION A(LDA,K), ALPHA, BETA, C(LDC,N)
C      INTEGER          I, J
C
C      EXTERNAL          DSYRK
C      INTRINSIC         DBLE
C
C      Initialize the array A to store the matrix A shown below.
C      Initialize the array C to store in symmetric form the
C      symmetric matrix C shown below.
C
C      A =   3  4      1  2  3
C      5  6      C =  2  4  5
C      7  8      3  5  6
C
C      DATA A / 3.0D0, 5.0D0, 7.0D0, 4.0D0, 6.0D0, 8.0D0 /
C      DATA C / 1.0D0, 2.0D0, 3.0D0, 8D8, 4.0D0, 5.0D0, 8D8, 8D8,
C      $        6.0D0 /
C
C      ALPHA = 1.0D0
C      BETA = 3.0D0
C      PRINT 1000
C      DO 100, I = 1, N
C          PRINT 1010, (A(I, J), J = 1, K)
100 CONTINUE
C      PRINT 1020

```

```
      DO 110, I = 1, N
        PRINT 1010, (C(I,J), J = 1, I), (C(J,I), J = I + 1, N)
110    CONTINUE
      PRINT 1030
      PRINT 1010, ((C(I,J), J = 1, N), I = 1, N)
      CALL DSYRK ('LOWER TRIANGULAR C', 'NOT TRANSPOSED C', N, K,
$          ALPHA, A, LDA, BETA, C, LDC)
      PRINT 1040
      DO 130, I = 1, N
        PRINT 1010, (C(I,J), J = 1, I), (C(J,I), J = I + 1, N)
130    CONTINUE
C
1000  FORMAT (1X, 'A:')
1010  FORMAT (3(2X, F6.1))
1020  FORMAT (/1X, 'C in full form:')
1030  FORMAT (/1X, 'C in symmetric form: (* in unused elements)')
1040  FORMAT (/1X, 'AA' + C:')
C
      END
```

Sample Output

```
A:
    3.0    4.0
    5.0    6.0
    7.0    8.0

C in full form:
    1.0    2.0    3.0
    2.0    4.0    5.0
    3.0    5.0    6.0

C in symmetric form: (* in unused elements)
    1.0 *****
    2.0    4.0 *****
    3.0    5.0    6.0

AA' + C:
    28.0    45.0    62.0
    45.0    73.0    98.0
    62.0    98.0   131.0
```

Rank-2k Update of a Symmetric Matrix

The subroutines described in this section compute one of the following results for general matrices A and B and a symmetric matrix C:

Operation	TRANSA	Matrices A and B
$C \leftarrow \alpha AB^T + \alpha BA^T + \beta C$	'N' or 'n'	n × k
$C \leftarrow \alpha A^T B + \alpha B^T A + \beta C$	'T' or 't'	k × n

Calling Sequence

```
CALL DSYR2K (UPLO, TRANSA, N, K, DALPHA, DA, LDA, DB, LDB, DBETA,
             DC, LDC)
CALL SSYR2K (UPLO, TRANSA, N, K, SALPHA, SA, LDA, SB, LDB, SBETA,
             SC, LDC)
CALL ZSYR2K (UPLO, TRANSA, N, K, ZALPHA, ZA, LDA, ZB, LDB, ZBETA,
             ZC, LDC)
CALL CSYR2K (UPLO, TRANSA, N, K, CALPHA, CA, LDA, CB, LDB, CBETA,
             CC, LDC)

void dsyr2k (char uplo, char trans, int n, int k, double dalpha,
             double *da, int lda, double *db, int ldb, double dbeta,
             double *dc, int ldc)
void ssyr2k (char uplo, char trans, int n, int k, float salpha,
             float *sa, int lda, float *sb, int ldb, float sbeta,
             float *sc, int ldc)
void zsy2k (char uplo, char trans, int n, int k, doublecomplex
             *zalpha, doublecomplex *za, int lda, doublecomplex
             *zb, int ldb, doublecomplex *zbeta, doublecomplex *zc,
             int ldc)
void csyr2k (char uplo, char trans, int n, int k, complex *calpha,
             complex *ca, int lda, complex *cb, int ldb, complex
             *cbeta, complex *cc, int ldc)
```

Arguments

UPLO	Indicates whether the values in a matrix reside in the upper or lower triangle of the array in which it is stored. The legal values for UPLO are listed below. Any value not listed below is illegal.
'L' or 'l'	Only lower triangle of array will be referenced.
'U' or 'u'	Only upper triangle of array will be referenced.

TRANSA	Indicates how to use the matrix. The legal values for TRANSA are listed below. Any value not listed below is illegal. 'N' or 'n' Use the matrix as it is stored. 'T' or 't' Use the transpose of the matrix
N, K	Indicate the size of matrix A as shown in the table above. $N, K \geq 0$.
xALPHA	Scalar that scales the input value of the matrix stored in A.
xA	Two-dimensional array in which one of the input matrices is stored.
LDA	Leading dimension of the array A as specified in the dimension or type statement. $LDA \geq \max(1, K)$. If TRANSA = 'N' or 'n', $LDA \geq \max(1, N)$.
xB	Two-dimensional array in which one of the input matrices is stored.
LDB	Leading dimension of the array B as specified in the dimension or type statement. $LDB \geq \max(1, K)$. If TRANSB = 'N' or 'n', $LDB \geq \max(1, N)$.
xBETA	Scalar that scales the input value of the matrix stored in C.
xC	Two-dimensional array. On entry, an input matrix. On exit, the result matrix.
LDC	Leading dimension of the array C as specified in the dimension or type statement. $LDC \geq \max(1, N)$.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
C   INTEGER          K, LDA, LDB, LDC, N
C   PARAMETER       (K = 2)
C   PARAMETER       (N = 3)
C   PARAMETER       (LDA = N)
C   PARAMETER       (LDB = N)
C   PARAMETER       (LDC = N)

C
C   DOUBLE PRECISION A(LDA,K), ALPHA, B(LDB,K), BETA, C(LDC,N)
C   INTEGER          I, J

C
C   EXTERNAL         DSQR2K
C   INTRINSIC        DBLE

C
C   Initialize the arrays A and B to store the matrices A and B
C   shown below. Initialize the array C to store in symmetric
C   form the symmetric matrix C shown below.
C
C
C   A =  3  4      B =  4  7      C =  1  2  3
C       5  6      B =  5  8      C =  2  4  5
C       7  8      B =  6  9      C =  3  5  6

```

```

C      DATA A / 1.0D0, 1.0D0, 1.0D0, 2.0D0, 2.0D0, 2.0D0 /
      DATA B / 1.0D0, 2.0D0, 1.0D0, 2.0D0, 1.0D0, 2.0D0 /
      DATA C / 1.0D2, 1.0D2, 1.0D2, 8D8, 1.0D2, 1.0D2, 8D8, 8D8,
$      1.0D2 /
C
      ALPHA = 1.0D0
      BETA = 1.0D0
      PRINT 1000
      DO 100, I = 1, N
        PRINT 1010, (C(I,J), J = 1, I), (C(J,I), J = I + 1, N)
100    CONTINUE
      PRINT 1020
      PRINT 1010, ((C(I,J), J = 1, N), I = 1, N)
      PRINT 1030
      DO 110, I = 1, N
        PRINT 1010, (A(I,J), J = 1, K)
110    CONTINUE
      PRINT 1040
      DO 120, I = 1, N
        PRINT 1010, (B(I,J), J = 1, K)
120    CONTINUE
      CALL DSYR2K ('LOWER TRIANGULAR C', 'NOT TRANSPOSED C', N, K,
$      ALPHA, A, LDA, B, LDB, BETA, C, LDC)
      PRINT 1050
      DO 130, I = 1, N
        PRINT 1010, (C(I,J), J = 1, I), (C(J,I), J = I + 1, N)
130    CONTINUE
C
1000  FORMAT (1X, 'C in full form:')
1010  FORMAT (3(2X, F6.1))
1020  FORMAT (/1X, 'C in symmetric form: (* in unused elements)')
1030  FORMAT (/1X, 'A:')
1040  FORMAT (/1X, 'B:')
1050  FORMAT (/1X, 'AB'' + BA'' + C:')
C
      END

```

Sample Output

```

C in full form:
100.0  100.0  100.0
100.0  100.0  100.0
100.0  100.0  100.0

C in symmetric form: (* in unused elements)
100.0  *****
100.0  100.0  *****
100.0  100.0  100.0

```

```
A:
  1.0    2.0
  1.0    2.0
  1.0    2.0

B:
  1.0    2.0
  2.0    1.0
  1.0    2.0

AB' + BA' + C:
 110.0   109.0   110.0
 109.0   108.0   109.0
 110.0   109.0   110.0
```

Product of a Triangular Matrix and a General Matrix

This subroutines described in this section compute the following functions of the triangular matrix A and the general matrix B:

Operation	TRANS A	SIDE	Matrix A
$B \leftarrow \alpha AB$	'N' or 'n'	'L' or 'l'	m × m
$B \leftarrow \alpha A^T B$	'T' or 't'	'L' or 'l'	m × m
$B \leftarrow \alpha A^H B$	'C' or 'c'	'L' or 'l'	m × m
$B \leftarrow \alpha BA$	'N' or 'n'	'R' or 'r'	n × n
$B \leftarrow \alpha BA^T$	'T' or 't'	'R' or 'r'	n × n
$B \leftarrow \alpha BA^H$	'C' or 'c'	'R' or 'r'	n × n

Calling Sequence

CALL DTRMM	(SIDE, UPLO, TRANS A, DIAG, M, N, DALPHA, DA, LDA, DB, LDB)
CALL STRMM	(SIDE, UPLO, TRANS A, DIAG, M, N, SALPHA, SA, LDA, SB, LDB)
CALL ZTRMM	(SIDE, UPLO, TRANS A, DIAG, M, N, ZALPHA, ZA, LDA, ZB, LDB)
CALL CTRMM	(SIDE, UPLO, TRANS A, DIAG, M, N, CALPHA, CA, LDA, CB, LDB)
void dtrmm	(char side, char uplo, char transa, char diag, int m, int n, double dalpha, double *da, int lda, double *db, int ldb)
void strmm	(char side, char uplo, char transa, char diag, int m, int n, float salpha, float *sa, int lda, float *sb, int ldb)
void ztrmm	(char side, char uplo, char transa, char diag, int m, int n, doublecomplex *zalpha, doublecomplex *za, int lda, doublecomplex *zb, int ldb)
void ctrmm	(char side, char uplo, char transa, char diag, int m, int n, complex *calpha, complex *ca, int lda, complex *cb, int ldb)

Arguments

SIDE	Indicates whether the product involving matrices A and B should be formed as AB or BA. The legal values for SIDE are listed below. Any value not listed below is illegal. 'L' or 'l' Compute AB. 'R' or 'r' Compute BA.
UPLO	Indicates whether the values in a matrix reside in the upper or lower triangle of the array in which it is stored. The legal values for UPLO are listed below. Any value not listed below is illegal. 'L' or 'l' Only lower triangle of array will be referenced. 'U' or 'u' Only upper triangle of array will be referenced.
TRANSA	Indicates how to use the matrix stored in A. The legal values for TRANSA are listed below. Any value not listed below is illegal. 'N' or 'n' Use the matrix as it is stored in A. 'T' or 't' Use the transpose of the matrix stored in A. 'C' or 'c' Use the conjugate transpose of the matrix stored in A.
DIAG	Indicates whether the matrix is unit triangular: 'U' or 'u' Unit triangular 'N' or 'n' Not unit triangular
M, N	Indicate the size of matrix A as shown in the table above.
xALPHA	Scalar that scales the input value of the matrix stored in A.
xA	Two-dimensional array in which one of the input matrices is stored.
LDA	Leading dimension of the array A as specified in the dimension or type statement. $LDA \geq \max(1, N)$. If SIDE = 'L' or 'l', $LDA \geq \max(1, M)$.
xB	Two-dimensional array. On entry, one of the input matrices. On exit, the result matrix.
LDB	Leading dimension of the array B as specified in the dimension or type statement. $LDB \geq \max(1, M)$.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
    INTEGER LDA, LDC, M, N
    PARAMETER (M = 3)
    PARAMETER (LDA = M)
    PARAMETER (LDC = M)
    PARAMETER (N = 2)
C

```

```

C      DOUBLE PRECISION  A(LDA,M), ALPHA, C(LDA,N)
C      INTEGER          I, J
C
C      EXTERNAL          DTRMM
C      INTRINSIC          DBLE
C
C      Initialize the array A to store in triangular form the
C      matrix A shown below. Initialize the array C to store
C      the matrix C shown below.
C
C      A =   1           C =  11  12
C      2   4           21  22
C      3   5   6       31  32
C
C      DATA A / 1.0D0, 2.0D0, 3.0D0, 8D8, 4.0D0, 5.0D0, 8D8,
$           8D8, 6.0D0 /
C      DATA C / 1.1D1, 2.1D1, 3.1D1, 1.2D1, 2.2D1, 3.2D1 /
C
C      PRINT 1000
C      DO 100, I = 1, M
C          PRINT 1010, (A(I,J), J = 1, I)
100  CONTINUE
C      PRINT 1020
C      PRINT 1030, ((C(I,J), J = 1, N), I = 1, M)
C      ALPHA = 2.0D0
C      CALL DTRMM ('LEFT SIDE A', 'LOWER TRIANGULAR A',
$              'NO TRANSPOSE A', 'NO UNIT DIAGONAL A', M, N,
$              ALPHA, A, LDA, C, LDC)
C      PRINT 1040
C      PRINT 1030, ((C(I,J), J = 1, N), I = 1, M)
C
C      1000 FORMAT (1X, 'A:')
C      1010 FORMAT (3(2X, F5.1))
C      1020 FORMAT (/1X, 'C:')
C      1030 FORMAT (2(2X, F5.1))
C      1040 FORMAT (/1X, 'AC:')
C
C      END

```

Sample Output

```

A:
  1.0
  2.0   4.0
  3.0   5.0   6.0

C:
 11.0  12.0
 21.0  22.0
 31.0  32.0

AC:

```

22.0	24.0
212.0	224.0
648.0	676.0

Solution for a Triangular System of Equations

This subroutines described in this section compute the following functions of the triangular matrix A and the general matrix B:

Operation	TRANSA	SIDE	Matrix A
$B \leftarrow \alpha A^{-1} B$	'N' or 'n'	'L' or 'l'	$m \times m$
$B \leftarrow \alpha A^{-T} B$	'T' or 't'	'L' or 'l'	$m \times m$
$B \leftarrow \alpha A^{-H} B$	'C' or 'c'	'L' or 'l'	$m \times m$
$B \leftarrow \alpha B A^{-1}$	'N' or 'n'	'R' or 'r'	$n \times n$
$B \leftarrow \alpha B A^{-T}$	'T' or 't'	'R' or 'r'	$n \times n$
$B \leftarrow \alpha B A^{-H}$	'C' or 'c'	'R' or 'r'	$n \times n$

Calling Sequence

CALL DTRSM	(SIDE, UPLO, TRANSA, DIAG, M, N, DALPHA, DA, LDA, DB, LDB)
CALL STRSM	(SIDE, UPLO, TRANSA, DIAG, M, N, SALPHA, SA, LDA, SB, LDB)
CALL ZTRSM	(SIDE, UPLO, TRANSA, DIAG, M, N, ZALPHA, ZA, LDA, ZB, LDB)
CALL CTRSM	(SIDE, UPLO, TRANSA, DIAG, M, N, CALPHA, CA, LDA, CB, LDB)
void dtrsm	(char side, char uplo, char transa, char diag, int m, int n, double dalpha, double *da, int lda, double *db, int ldb)
void strsm	(char side, char uplo, char transa, char diag, int m, int n, float salpha, float *sa, int lda, float *sb, int ldb)
void ztrsm	(char side, char uplo, char transa, char diag, int m, int n, doublecomplex *zalpha, doublecomplex *za, int lda, doublecomplex *zb, int ldb)
void ctrsm	(char side, char uplo, char transa, char diag, int m, int n, complex *calpha, complex *ca, int lda, complex *cb, int ldb)

Arguments

SIDE	Indicates how to multiply matrices: 'L' or 'l' Multiply general matrix by triangular matrix on the left. 'R' or 'r' Multiply general matrix by triangular matrix on the right.
UPLO	Indicates whether the values in a matrix reside in the upper or lower triangle of the array in which it is stored. The legal values for UPLO are listed below. Any value not listed below is illegal. 'L' or 'l' Only lower triangle of array will be referenced. 'U' or 'u' Only upper triangle of array will be referenced.
TRANSA	Indicates how to use the matrix stored in A. The legal values for TRANSA are listed below. Any value not listed below is illegal. 'N' or 'n' Use the matrix as it is stored in A. 'T' or 't' Use the transpose of the matrix stored in A. 'C' or 'c' Use the conjugate transpose of the matrix stored in A.
DIAG	Indicates whether the matrix is unit triangular: 'U' or 'u' Unit triangular 'N' or 'n' Not unit triangular
M, N	Indicate the size of matrix A as shown in the table above.
xALPHA	Scalar that scales the input value of the matrix stored in A.
xA	Two-dimensional array in which one of the input matrices is stored.
LDA	Leading dimension of the array A as specified in the dimension or type statement. $LDA \geq \max(1, N)$. If $SIDE = 'L' \text{ or } 'l'$, $LDA \geq \max(1, M)$.
xB	Two-dimensional array. On entry, one of the input matrices. On exit, the result matrix.
LDB	Leading dimension of the array B as specified in the dimension or type statement. $LDB \geq \max(1, M)$.

Sample Program

```

      PROGRAM TEST
      IMPLICIT NONE
C
      INTEGER          LDA, LDB, M, N
      PARAMETER        (M = 3)
      PARAMETER        (LDA = M)
      PARAMETER        (LDB = M)
      PARAMETER        (N = 2)
C
      DOUBLE PRECISION A(LDA,M), ALPHA, B(LDB,N)

```

```

C      INTEGER          I, J
C
C      EXTERNAL          DTRSM
C      INTRINSIC          DBLE
C
C      Initialize the array A to store in triangular form the
C      matrix A shown below. Initialize the array B to store
C      the matrix B shown below.
C
C      1  2  4          1  2
C      A =   3  5      B =  1  4
C           6          1  6
C
C      DATA A / 1.0D0, 8D8, 8D8, 2.0D0, 3.0D0, 8D8, 4.0D0, 5.0D0,
$              6.0D0 /
C      DATA B / 7.0D0, 8.0D0, 6.0D0, 3.4D1, 4.2D1, 3.6D1 /
C
C      PRINT 1000
C      PRINT 1010, A(1,1), A(1,2), A(1,3)
C      PRINT 1020, A(2,2), A(2,3)
C      PRINT 1030, A(3,3)
C      PRINT 1040
C      PRINT 1050, ((B(I,J), J = 1, N), I = 1, M)
C      ALPHA = 1.0D0
C      CALL DTRSM ('LEFT SIDE A', 'UPPER TRIANGULAR A',
$              'NOT TRANSPOSED A', 'NOT UNIT DIAGONAL A', M, N,
$              ALPHA, A, LDA, B, LDB)
C      PRINT 1060
C      PRINT 1050, ((B(I,J), J = 1, N), I = 1, M)
C
1000 FORMAT (1X, 'A:')
1010 FORMAT (1X, 3(2X, F5.1))
1020 FORMAT (1X, 7X, 2(2X, F5.1))
1030 FORMAT (1X, 14X, 2X, F5.1)
1040 FORMAT (/1X, 'B:')
1050 FORMAT (2(2X, F5.1))
1060 FORMAT (/1X, 'A**(-1) * B:')
C
      END

```

Sample Output

```

A:
    1.0    2.0    4.0
         3.0    5.0
             6.0

B:
    7.0   34.0
    8.0   42.0
    6.0   36.0

```

A**(-1) * B:	
1.0	2.0
1.0	4.0
1.0	6.0

Fast Fourier Transform Subroutines (FFTPACK and VFFTPACK)

6 

This chapter describes the subroutines in FFTPACK and VFFTPACK as they are implemented in the Sun Performance Library. FFTPACK and VFFTPACK are two collections of subroutines to perform Fourier transforms and Fourier synthesis on real and complex data sets.

The first two sections of this chapter describe:

- conventions followed in naming arguments and subroutines
- extensions to the standard versions of FFTPACK and VFFTPACK.

The remaining sections of this chapter describe the operation of each of the FFT and VFFT subroutines. For each subroutine the interface is shown, the operation of the subroutine is described, and a short FORTRAN 77 program is given with sample output.

Naming Conventions

The types of all arguments follow the notation shown below:

```

IMPLICIT REAL (R)
IMPLICIT DOUBLE PRECISION (D)
IMPLICIT COMPLEX (C)
IMPLICIT DOUBLE COMPLEX (Z)
IMPLICIT INTEGER (I-N)

```

The base name of the subroutine identifies the operation performed:

COSQB	Backward cosine quarter-wave transform (synthesis)
COSQF	Cosine quarter-wave transform
COSQI	Initialize cosine quarter-wave transform or synthesis
COST	Cosine even-wave transform
COSTI	Initialize cosine even-wave transform
EZFFTB	Backward EZ transform (synthesis)
EZFFTF	EZ transform
EZFFTI	Initialize EZ transform
FFTB	Backward transform (synthesis)
FFTF	Forward transform
FFTI	Initialize before applying a transform or synthesis
SINQB	Backward sine quarter-wave transform (synthesis)
SINQF	Sine quarter-wave transform
SINQI	Initialize sine quarter-wave transform or synthesis
SINT	Sine odd-wave transform
SINTI	Initialize sine odd-wave transform

These subroutine names may have a one or two-letter prefix that identifies the type of operands processed:

No prefix:	Operands are real.
R	Operands are real.
D	Operands are double precision.
C	Operands are complex.
Z	Operands are double complex.
VR	Subroutine operates on multiple vectors supplied simultaneously; operands are real.
VD	Subroutine operates on multiple vectors supplied simultaneously; operands are double precision.

Extensions to FFTPACK and VFFTPACK

There are two extensions to the standard FFTPACK and VFFTPACK libraries. First, Sun Performance Library contains double precision and double complex transforms. Subroutines to process double precision and double complex data are not available with the standard package from Netlib, so calls to these subroutine may not be portable. Second, this version of Sun Performance Library contains a native C/C++ interface to FFTPACK and VFFTPACK. Conventions for this interface are described in detail in the preface to this manual.

Transpose Array

The subroutines described in this section perform an in-place or out-of-place transpose on a two-dimensional array.

Calling Sequence

```
CALL DTRANS (PLACE, DSCALE, DSOURCE, M, N, DDEST)
CALL STRANS (PLACE, SSCALE, SSOURCE, M, N, SDEST)
CALL ZTRANS (PLACE, ZSCALE, ZSOURCE, M, N, ZDEST)
CALL CTRANS (PLACE, CSCALE, CSOURCE, M, N, CDEST)

void dtrans (char place, double dscale, double *dsource, int m, int
n, double *ddest)
void strans (char place, float sscale, float *ssource, int m, int
n, float *sdest)
void ztrans (char place, doublecomplex zscale, doublecomplex
*zsource, int m, int n, doublecomplex *zdest)
void ctrans (char place, complex cscale, complex *csource, int m,
int n, complex *cdest)
```

Arguments

PLACE	Indicates whether transpose is in-place or out-of-place. 'I' or 'i' In-place; transpose from SOURCE into SOURCE. 'O' or 'o' Out-of-place; transpose from SOURCE into DEST.
xSCALE	The scale factor for the transpose.
xSOURCE	On entry, the matrix to be transposed. On exit, if PLACE = 'I' or 'i', the transposed matrix; otherwise PLACE is not used.
M	Number of rows in xSOURCE.
N	Number of columns in xSOURCE.
xDEST	On exit, if PLACE = 'O' or 'o', the transposed matrix. If PLACE = 'I' or 'i', xDEST is not used.

Sample Program

PROGRAM TEST	
IMPLICIT NONE	
C	
INTEGER	M, N


```

      PARAMETER      (M = 4)
      PARAMETER      (N = 3)
C
      DOUBLE COMPLEX  A(M,N), ALPHA, B(N,M)
      INTEGER         I, J
C
      EXTERNAL        ZTRANS
C
      INTRINSIC       DCMLPX
C
      Initialize the array A to store the general matrix A below:
C
C      A = (1.0, 1.0) (1.0, 2.0) (1.0, 3.0)
C          (2.0, 1.0) (2.0, 2.0) (2.0, 3.0)
C          (3.0, 1.0) (3.0, 2.0) (3.0, 3.0)
C          (4.0, 1.0) (4.0, 2.0) (4.0, 3.0)
C
      DO 110, J = 1, N
        DO 100, I = 1, M
          A(I,J) = DCMLPX (I,J)
100    CONTINUE
110    CONTINUE
      PRINT 1000
      DO 120, I = 1, M
        PRINT 1010, (A(I,J), J = 1, N)
120    CONTINUE
      ALPHA = DCMLPX (1.0D0, 0.0D0)
      CALL ZTRANS ('OUT-OF-PLACE', ALPHA, A, M, N, B)
      PRINT 1020
      DO 130, I = 1, N
        PRINT 1010, (B(I,J), J = 1, M)
130    CONTINUE
C
1000  FORMAT (1X, 'A:')
1010  FORMAT (1X, 4(2X, '(F4.1, ', ', F4.1, ')') :))
1020  FORMAT (/1X, 'TRANS(A):')
C
      END

```

Sample Output

```

A:
( 1.0, 1.0) ( 1.0, 2.0) ( 1.0, 3.0)
( 2.0, 1.0) ( 2.0, 2.0) ( 2.0, 3.0)
( 3.0, 1.0) ( 3.0, 2.0) ( 3.0, 3.0)
( 4.0, 1.0) ( 4.0, 2.0) ( 4.0, 3.0)

TRANS(A):
( 1.0, 1.0) ( 2.0, 1.0) ( 3.0, 1.0) ( 4.0, 1.0)
( 1.0, 2.0) ( 2.0, 2.0) ( 3.0, 2.0) ( 4.0, 2.0)
( 1.0, 3.0) ( 2.0, 3.0) ( 3.0, 3.0) ( 4.0, 3.0)

```

Cosine Quarter-Wave Synthesis

The subroutines described in this section synthesize a Fourier sequence from its representation in terms of a cosine series with odd wave numbers. The transform is defined as:

For $i=1,\dots,n$

$$x(i) = \sum_{k=1}^n 4 \times x(k) \times \cos\left((2 \times k - 1) \times \left((i - 1) \times \frac{\pi}{2 \times n}\right)\right)$$

Note – The xCOSQ operations are unnormalized inverses of themselves, so a call to xCOSQF followed by a call to xCOSQB will multiply the input sequence by $4 \times N$. The VxCOSQ operations are normalized, so a call of VxCOSQF followed by a call of VxCOSQB will return the original sequence.

Calling Sequence

```
CALL COSQB   (N, RX, RWSAVE)
CALL DCOSQB  (N, DX, DWSAVE)
CALL VCOSQB  (M, N, RX, RXT, MDIMX, RWSAVE)
CALL VDCOSQB (M, N, DX, DXT, MDIMX, DWSAVE)

void cosqb   (int n, float *rx, float *rwsave)
void dcosqb  (int n, double *dx, double *dwsave)
void vcosqb  (int m, int n, float *rx, int mdimx, float *rwsave)
void vdcosqb (int m, int n, double *dx, int mdimx, double *dwsave)
```

Arguments

M	(For vector operations only.) The number of sequences to be transformed. $M \geq 0$.
N	Length of the sequence to be transformed. These subroutines are most efficient when N is a product of small primes. $N \geq 0$.
xX	On entry, an array of length N containing the sequence to be transformed. For VxCOSQB, a real two-dimensional array with dimensions of (MDIMX $\times$ N) whose rows contain the sequences to be transformed. On exit, the quarter-wave cosine synthesis of the input.

xXT	(For vector operations only.) A real two-dimensional work array with dimensions of $(MDIMX \times N)$.
MDIMX	(For vector operations only.) Leading dimension of the arrays xX and xXT as specified in a dimension or type statement. $MDIMX \geq M$.
xWSAVE	On entry, an array with dimension of at least $(3 \times N + 15)$ for scalar subroutines or $(2 \times N + 15)$ for vector subroutines, initialized by xCOSQI or VxCOSQI.

Sample Program

```

      PROGRAM TEST
      IMPLICIT NONE
C
      INTEGER          M, N
      PARAMETER        (M = 1)
      PARAMETER        (N = 6)
C
      INTEGER          I
      REAL              PI, WSAVE(N + 15), X(M, N), XT(M, N)
C
      EXTERNAL          VCOSQB, VCOSQF, VCOSQI
      INTRINSIC         ACOS, COS
C
C      Initialize the first row of the array to a real even
C      quarter-wave sequence, that is, it can be expanded in
C      terms of a cosine series with only odd wave numbers.
C
      PI = ACOS (-1.0)
      DO 100, I=1, N
        X(M,I) = 40.0 * COS ((I - 1) * PI / (2.0 * N))
100  CONTINUE
C
      PRINT 1000
      PRINT 1010, (X(M, I), I = 1, N)
      CALL VCOSQI (N, WSAVE)
      CALL VCOSQF (M, N, X, XT, M, WSAVE)
      PRINT 1020
      PRINT 1010, (X(M, I), I = 1, N)
      CALL VCOSQB (M, N, X, XT, M, WSAVE)
      PRINT 1030
      PRINT 1010, (X(M, I), I = 1, N)
C
1000  FORMAT (1X, 'Original Sequence: ')
1010  FORMAT (1X, 100(F5.1, 1X))
1020  FORMAT (1X, 'Transformed Sequence: ')
1030  FORMAT (1X, 'Recovered Sequence: ')
C
      END

```

Sample Output

```
Original Sequence:
40.0 38.6 34.6 28.3 20.0 10.4
Transformed Sequence:
49.0 0.0 0.0 0.0 0.0 0.0
Recovered Sequence:
40.0 38.6 34.6 28.3 20.0 10.4
```

Cosine Quarter-Wave Transform

The subroutines described in this section compute the Fourier coefficients in a cosine series representation with only odd wave numbers. The transform is defined as:

For $i=1,\dots,n$

$$x(i) = x(1) + \sum_{k=2}^n 2 \times x(k) \times \cos\left((2 \times i - 1) \times \left((k - 1) \times \frac{\pi}{2 \times n}\right)\right)$$

Note – The xCOSQ operations are unnormalized inverses of themselves, so a call to xCOSQF followed by a call to xCOSQB will multiply the input sequence by $4 \times N$. The VxCOSQ operations are normalized, so a call of VxCOSQF followed by a call of VxCOSQB will return the original sequence.

Calling Sequence

```
CALL COSQF   (N, RX, RWSAVE)
CALL DCOSQF  (N, DX, RWSAVE)
CALL VCOSQF  (M, N, RX, RXT, MDIMX, RWSAVE)
CALL VDCOSQF (M, N, DX, DXT, MDIMX, DWSAVE)

void cosqf   (int n, float *rx, float *rwsave)
void dcosqf  (int n, double *dx, double *dwsave)
void vcosqf  (int m, int n, float *rx, int mdimx, float *rwsave)
void vdcosqf (int m, int n, double *dx, int mdimx, double *dwsave)
```

Arguments

M	(For vector operations only.) The number of sequences to be transformed. $M \geq 0$.
N	Length of the sequence to be transformed. These subroutines are most efficient when N is a product of small primes. $N \geq 0$.
xX	On entry, an array of length N containing the sequence to be transformed. For VxCOSQF, a real two-dimensional array with dimensions of $(MDIMX \times N)$ whose rows contain the sequences to be transformed. On exit, the quarter-wave cosine transform of the input.
xXT	(For vector operations only.) A real two-dimensional work array with dimensions of $(MDIMX \times N)$.
MDIMX	(For vector operations only.) Leading dimension of the arrays xX and xXT as specified in a dimension or type statement. $MDIMX \geq M$.
xWSAVE	On entry, an array with dimension of at least $(3 \times N + 15)$ for scalar subroutines or $(2 \times N + 15)$ for vector subroutines, initialized by xCOSTI or VxCOSTI.

Sample Program

```

      PROGRAM TEST
      IMPLICIT NONE
C
      INTEGER          M, N
      PARAMETER        (M = 4)
      PARAMETER        (N = 6)
C
      INTEGER          I, J
      REAL              PI, WSAVE(N + 15), X(M, N), XT(M, N)
C
      EXTERNAL          VCOSQB, VCOSQF, VCOSQI
      INTRINSIC         ACOS, COS
C
C      Initialize the array to m real even quarter-wave sequences,
C      that is, they can be expanded in terms of a cosine series
C      with only odd wave numbers.
C
      PI = ACOS (-1.0)
      DO 110, J=1, M
        DO 100, I=1, N
          X(J,I) = 40.0 * J * COS ((I-1) * PI / 2.0 / N )
100    CONTINUE
110  CONTINUE
C
      CALL VCOSQI (N, WSAVE)
      PRINT 1000
      DO 120, J=1, M

```

```
      PRINT 1010, J, (X(J, I), I = 1, N)
120  CONTINUE
      CALL VCOSQF (M, N, X, XT, M, WSAVE)
      PRINT 1020
      DO 130, J=1, M
        PRINT 1010, J, (X(J, I), I = 1, N)
130  CONTINUE
      CALL VCOSQB (M, N, X, XT, M, WSAVE)
      PRINT 1030
      DO 140, J=1, M
        PRINT 1010, J, (X(J, I), I = 1, N)
140  CONTINUE
C
1000 FORMAT (1X, 'Original Sequence: ')
1010 FORMAT(1X, '  Sequence', I2, ': ', 100(F5.1, 1X))
1020 FORMAT (1X, 'Transformed Sequence: ')
1030 FORMAT (1X, 'Recovered Sequence: ')
C
      END
```

Sample Output

```
Original Sequence:
Sequence 1:  40.0  38.6  34.6  28.3  20.0  10.4
Sequence 2:  80.0  77.3  69.3  56.6  40.0  20.7
Sequence 3: 120.0 115.9 103.9  84.9  60.0  31.1
Sequence 4: 160.0 154.5 138.6 113.1  80.0  41.4
Transformed Sequence:
Sequence 1:  49.0   0.0   0.0   0.0   0.0   0.0
Sequence 2:  98.0   0.0   0.0   0.0   0.0   0.0
Sequence 3: 147.0   0.0   0.0   0.0   0.0   0.0
Sequence 4: 196.0   0.0   0.0   0.0   0.0   0.0
Recovered Sequence:
Sequence 1:  40.0  38.6  34.6  28.3  20.0  10.4
Sequence 2:  80.0  77.3  69.3  56.6  40.0  20.7
Sequence 3: 120.0 115.9 103.9  84.9  60.0  31.1
Sequence 4: 160.0 154.5 138.6 113.1  80.0  41.4
```

Initialize Cosine Quarter-Wave Transform and Synthesis

The subroutines described in this section initialize the array xWSAVE, which is used in both xCOSQF and xCOSQB.

Calling Sequence

```
CALL COSQI    (N, RWSAVE)
CALL DCOSQI   (N, DWSAVE)
CALL VCOSQI   (N, RWSAVE)
CALL VDCOSQI  (N, DWSAVE)

void cosqi    (int n, float *rwsave)
void dcosqi   (int n, double *dwsave)
void vcosqi   (int n, float *rwsave)
void vdcosqi  (int n, double *dwsave)
```

Arguments

N	Length of the sequence to be transformed. The method is most efficient when N is a product of small primes.
xWSAVE	On entry, an array with a dimension of at least $(3 \times N + 15)$ for scalar subroutines or $(2 \times N + 15)$ for vector subroutines. The same work array can be used for both xCOSQF and xCOSQB as long as N remains unchanged. Different xWSAVE arrays are required for different values of N. This initialization does not have to be repeated between calls to xCOSQF or xCOSQB as long as N and xWSAVE remain unchanged, thus subsequent transforms can be obtained faster than the first.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C      INTEGER      N
C      PARAMETER    (N = 6)

C      INTEGER      I
C      REAL         PI, WSAVE(3 * N + 15), X(N)

C      EXTERNAL     COSQB, COSQF, COSQI
```



```

      INTRINSIC      ACOS, COS
C
C      Initialize array X to a real even quarter-wave sequence,
C      that is, it can be expanded in terms of a cosine series
C      with only odd wave numbers.
C
      PI = ACOS (-1.0)
      DO 100, I=1, N
        X(I) = COS((I - 1) * PI / (2.0 * N))
100  CONTINUE
C
      CALL COSQI (N, WSAVE)
      PRINT 1000
      PRINT 1010, (X(I), I = 1, N)
      CALL COSQF (N, X, WSAVE)
      PRINT 1020
      PRINT 1010, (X(I), I = 1, N)
      CALL COSQB (N, X, WSAVE)
      PRINT 1030
      PRINT 1010, (X(I), I = 1, N)
C
1000 FORMAT(1X, 'Original Sequence: ')
1010 FORMAT(1X, 100(F7.3, 1X))
1020 FORMAT(1X, 'Transformed Sequence: ')
1030 FORMAT(1X, 'Recovered Sequence: ')
      END

```

Sample Output

```

Original Sequence:
 1.000  0.966  0.866  0.707  0.500  0.259
Transformed Sequence:
 6.000  0.000  0.000  0.000  0.000  0.000
Recovered Sequence:
24.000 23.182 20.785 16.971 12.000  6.212

```

Cosine Even-Wave Transform

The subroutines described in this section compute the discrete Fourier cosine transform of an even sequence. The transform is defined as:

For $i=1,\dots,n$

$$x(i) = x(1) + (-1)^{i-1} \times x(n) + \sum_{k=2}^{n-1} 2 \times x(k) \times \cos\left((k-1) \times (i-1) \times \frac{\pi}{n-1}\right)$$

Note – The xCOST transforms are unnormalized inverses of themselves, so a call of xCOST followed by another call of xCOST will multiply the input sequence by $2 \times (N-1)$. The VxCOST transforms are normalized, so a call of VxCOST followed by a call of VxCOST will return the original sequence.

Calling Sequence

```
CALL COST      (N, RX, RWSAVE)
CALL DCOST     (N, DX, DWSAVE)
CALL VCOST     (M,N, RX, RXT, MDIMX, RWSAVE)
CALL VDCOST    (M,N, DX, DXT, MDIMX, DWSAVE)

void cost      (int n, float *rx, float *rwsave)
void dcost     (int n, double *dx, double *dwsave)
void vcost     (int m, int n, float *rx, int mdimx, float *rwsave)
void vdcost    (int m, int n, double *dx, int mdimx, double *dwsave)
```

Arguments

M	(For vector operations only) The number of sequences to be transformed. $M \geq 0$.
N	Length of the sequence to be transformed. These subroutines are most efficient when $N-1$ is a product of small primes. $N \geq 2$.

xX	On entry, an array of length N containing the sequence to be transformed. For VxCOST, a real two-dimensional array with dimensions of $(MDIMX \times N)$ whose rows contain the sequences to be transformed. On exit, the cosine transform of the input.
xXT	(For vector operations only.) A real two-dimensional work array with dimensions of $(MDIMX \times (N-1))$.
MDIMX	(For vector operations only.) Leading dimension of the arrays xX and xXT as specified in a dimension or type statement. $MDIMX \geq M$.
xWSAVE	On entry, an array with dimension of at least $(3 \times N + 15)$, initialized by xCOSTI or VxCOSTI.

Sample Program

```

      PROGRAM TEST
      IMPLICIT NONE
C
      INTEGER          N
      PARAMETER        (N = 9)
C
      INTEGER          I
      REAL             PI, X(N), WSAVE(3 * N + 15)
C
      EXTERNAL          COST, COSTI
      INTRINSIC         ACOS, COS
C
C      Initialize the array X to an even sequence, that is, it
C      can be expanded in terms of a trigonometric series that
C      contains only cosine terms.
C
      PI = ACOS (-1.0)
      DO 100, I=1, N
        X(I) = COS ((I - 1.0) * 2.0 * PI / (N - 1.0))
100 CONTINUE
C
      CALL COSTI (N, WSAVE)
      PRINT 1000
      PRINT 1010, (X(I), I = 1, N)
      CALL COST (N, X, WSAVE)
      PRINT 1020
      PRINT 1010, (X(I), I = 1, N)
      CALL COST (N, X, WSAVE)
      PRINT 1030
      PRINT 1010, (X(I), I = 1, N)
C
1000 FORMAT (1X, 'Original Sequence: ')
1010 FORMAT (1X, 100(F5.1, 1X))
1020 FORMAT (1X, 'Transformed Sequence: ')

```

```
1030 FORMAT (1X, 'Recovered Sequence: ')
      END
```

Sample Output

```
Original Sequence:
  1.0   0.7   0.0  -0.7  -1.0  -0.7   0.0   0.7   1.0
Transformed Sequence:
  0.0   0.0   8.0   0.0   0.0   0.0   0.0   0.0   0.0
Recovered Sequence:
 16.0  11.3   0.0 -11.3 -16.0 -11.3   0.0  11.3  16.0
```

Initialize Cosine Even-Wave Transform

The subroutines described in this section initialize the array xWSAVE, which is used in xCOST.

Calling Sequence

```
CALL COSTI   (N, RWSAVE)
CALL DCOSTI  (N, DWSAVE)
CALL VCOSTI  (N, RWSAVE)
CALL VDCOSTI (N, DWSAVE)
```

```
void costi   (int n, float *rwsave)
void dcosti  (int n, double *dwsave)
void vcosti  (int n, float *rwsave)
void vdcosti (int n, double *dwsave)
```

Arguments

N	Length of the sequence to be transformed. The method is most efficient when $N-1$ is a product of small primes. $N \geq 2$.
xWSAVE	On entry, an array with a dimension of at least $(3 \times N + 15)$. Different xWSAVE arrays are required for different values of N. This initialization does not have to be repeated between calls to xCOST as long as N and xWSAVE remain unchanged, thus subsequent transforms can be obtained faster than the first.

Sample Program

```
PROGRAM TEST
IMPLICIT NONE
C
C      INTEGER          N
C      PARAMETER       (N = 9)
C
C      INTEGER          I
C      REAL             PI, X(N), WSAVE(3 * N + 15)
C
C      EXTERNAL         COST, COSTI
C      INTRINSIC        ACOS, COS
C
C      Initialize the array X to an even sequence, that is, it
C      can be expanded in terms of a trigonometric series that
```

```

C      contains only cosine terms.
C
      PI = ACOS (-1.0)
      DO 100, I=1, N
        X(I) = COS ((I - 1.0) * 2.0 * PI / (N - 1.0))
100    CONTINUE
C
      CALL COSTI (N, WSAVE)
      PRINT 1000
      PRINT 1010, (X(I), I = 1, N)
      CALL COST (N, X, WSAVE)
      PRINT 1020
      PRINT 1010, (X(I), I = 1, N)
      CALL COST (N, X, WSAVE)
      PRINT 1030
      PRINT 1010, (X(I), I = 1, N)
C
1000  FORMAT (1X, 'Original Sequence: ')
1010  FORMAT (1X, 100(F5.1, 1X))
1020  FORMAT (1X, 'Transformed Sequence: ')
1030  FORMAT (1X, 'Recovered Sequence: ')
      END

```

Sample Output

```

Original Sequence:
 1.0  0.7  0.0 -0.7 -1.0 -0.7  0.0  0.7  1.0
Transformed Sequence:
 0.0  0.0  8.0  0.0  0.0  0.0  0.0  0.0  0.0
Recovered Sequence:
16.0 11.3  0.0 -11.3 -16.0 -11.3  0.0 11.3 16.0

```

EZ Fourier Synthesis

The subroutine described in this section computes a periodic sequence from its Fourier coefficients. The synthesis is defined as:

If N is even let $k_{\max} = n/2$, otherwise let $k_{\max} = (n-1)/2$. Then for $i=1, \dots, n$

$$r(i) = a_0 + \sum_{k=1}^{k_{\max}} \left(a(k) \times \cos\left(k \times (i-1) \times 2 \times \frac{\pi}{n}\right) + b(k) \times \sin\left(k \times (i-1) \times 2 \times \frac{\pi}{n}\right) \right)$$

EZFFTB is a simplified but slower version of RFFTB.

Calling Sequence

```
CALL EZFFTB (N, R, AZERO, A, B, WSAVE)
```

```
void ezfftb (int n, float *r, float *azero, float *a, float *b,
            float *wsave)
```

Arguments

N	Length of the sequence to be synthesized. The method is most efficient when N is the product of small primes. $N \geq 0$.
R	On exit, the Fourier synthesis of the inputs.
AZERO	The constant Fourier coefficient a_0 .
A, B	Arrays that contain the remaining Fourier coefficients. On exit, these arrays are unchanged. If N is even, then A and B are length $N/2$, otherwise A and B are length $(N-1)/2$.
WSAVE	On entry, an array with dimension of at least $(3 \times N + 15)$, initialized by EZFFTI.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER          N
  PARAMETER        (N = 9)

C
  INTEGER          I
  REAL              A(N), B(N), AZERO, PI, R(N)
  REAL              WSAVE(3 * N + 15)

C
  EXTERNAL          EZFFTB, EZFFTF, EZFFTI
  INTRINSIC          ACOS, COS, SIN

C
  Initialize array to a sequence of real numbers.
C
  PI = ACOS (-1.0)
  DO 100, I=1, N
    R(I) = 3.0 + SIN ((I - 1.0) * 2.0 * PI / N) +
$      4.0 * COS ((I - 1.0) * 8.0 * PI / N)
  100 CONTINUE
C
  CALL EZFFTI (N, WSAVE)
  PRINT 1000
  PRINT 1010, (R(I), I = 1, N)
  CALL EZFFTF (N, R, AZERO, A, B, WSAVE)
  PRINT 1020, AZERO
  PRINT 1030
  PRINT 1010, (A(I), I = 1, N)
  PRINT 1040
  PRINT 1010, (B(I), I = 1, N)
  CALL EZFFTB (N, R, AZERO, A, B, WSAVE)
  PRINT 1050
  PRINT 1010, (R(I), I = 1, N)
C
  1000 FORMAT (1X, 'Original Sequence: ')
  1010 FORMAT (100(F6.1, 1X))
  1020 FORMAT (1X, 'Azero = ' F4.1)
  1030 FORMAT (1X, 'A = ')
  1040 FORMAT (1X, 'B = ')
  1050 FORMAT (1X, 'Recovered Sequence: ')
C
  END

```

Sample Output

```

Original Sequence:
  7.0  -0.1  7.0  1.9  4.0  3.4  0.1  5.1  -1.4
Azero = 3.0

```


A =	0.0	0.0	0.0	4.0	0.0	0.0	0.0	0.0	0.0
B =	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Recovered Sequence:	7.0	-0.1	7.0	1.9	4.0	3.4	0.1	5.1	-1.4

EZ Fourier Transform

The subroutine described in this section computes the Fourier coefficients of a periodic sequence. The transform is defined below at output parameters AZERO, A and B. EZFFTF is a simplified but slower version of RFFTF.

Calling Sequence

```
CALL EZFFTF (N, R, AZERO, A, B, WSAVE)
```

```
void ezfftf (int n, float *r, float *azero, float *a, float *b,  
float *wsave)
```

Arguments

N	Length of the sequence to be transformed. The method is most efficient when N is the product of small primes. $N \geq 0$.
R	A real array of length N containing the sequence to be transformed. On exit, R is unchanged.
AZERO	The sum from $i=1$ to $i=n$ of $r(i)/n$
A, B	For N even,

$$b\left(\frac{n}{2}\right) = 0 \quad \text{and} \quad a\left(\frac{n}{2}\right) = \sum_{i=1}^n (-1)^{i-1} \times \frac{r(i)}{n}$$

For N even, define $k_{\max}=n/2-1$ and for N odd, define $k_{\max}=(n-1)/2$, then for $k=1,\dots,k_{\max}$

$$a(k) = \sum_{i=1}^n \frac{2.0}{n} \times r(i) \times \cos\left(k \times (i-1) \times 2 \times \frac{\pi}{n}\right)$$

$$b(k) = \sum_{i=1}^n \frac{2.0}{n} \times r(i) \times \sin\left(k \times (i-1) \times 2 \times \frac{\pi}{n}\right)$$

xWSAVE	On entry, an array with dimension of at least $(3 \times N + 15)$, initialized by EZFFTI.

Sample Program

```

PROGRAM TEST
IMPLICIT NONE

C
  INTEGER          N
  PARAMETER        (N = 9)

C
  INTEGER          I
  REAL              A(N), B(N), AZERO, PI, R(N)
  REAL              WSAVE(3 * N + 15)

C
  EXTERNAL          EZFFTB, EZFFTF, EZFFTI
  INTRINSIC          ACOS, COS, SIN

C
  Initialize array to a sequence of real numbers.
C
  PI = ACOS (-1.0)
  DO 100, I=1, N
    R(I) = 3.0 + SIN ((I - 1.0) * 2.0 * PI / N) +
$      4.0 * COS ((I - 1.0) * 8.0 * PI / N)
  100 CONTINUE
C
  CALL EZFFTI (N, WSAVE)
  PRINT 1000
  PRINT 1010, (R(I), I = 1, N)
  CALL EZFFTF (N, R, AZERO, A, B, WSAVE)
  PRINT 1020, AZERO
  PRINT 1030
  PRINT 1010, (A(I), I = 1, N)
  PRINT 1040
  PRINT 1010, (B(I), I = 1, N)
  CALL EZFFTB (N, R, AZERO, A, B, WSAVE)
  PRINT 1050
  PRINT 1010, (R(I), I = 1, N)
C
  1000 FORMAT (1X, 'Original Sequence: ')
  1010 FORMAT (100(F6.1, 1X))
  1020 FORMAT (1X, 'Azero = ' F4.1)
  1030 FORMAT (1X, 'A = ')
  1040 FORMAT (1X, 'B = ')
  1050 FORMAT (1X, 'Recovered Sequence: ')
C
  END

```

Sample Output

```

Original Sequence:
  7.0  -0.1  7.0  1.9  4.0  3.4  0.1  5.1  -1.4
Azero = 3.0

```

A =								
0.0	0.0	0.0	4.0	0.0	0.0	0.0	0.0	0.0
B =								
1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Recovered Sequence:								
7.0	-0.1	7.0	1.9	4.0	3.4	0.1	5.1	-1.4

Initialize EZ Fourier Transform and Synthesis

The subroutine described in this section initializes the array WSAVE, which is used in both EZFFTf and EZFFTb.

Calling Sequence

```
CALL EZFFTf (N, WSAVE)
```

```
void ezfft (int n, float *wsave)
```

Arguments

N	Length of the sequence to be transformed. $N \geq 0$.
WSAVE	On entry, an array with a dimension of at least $(3 \times N + 15)$. The same work array can be used for both EZFFTf and EZFFTb as long as N remains unchanged. Different WSAVE arrays are required for different values of N. This initialization does not have to be repeated between calls to EZFFTf or EZFFTb as long as N and WSAVE remain unchanged, thus subsequent transforms can be obtained faster than the first.

Sample Program

```

      PROGRAM TEST
      IMPLICIT NONE
C
      INTEGER          N
      PARAMETER        (N = 9)
C
      INTEGER          I
      REAL              A(N), B(N), AZERO, PI, R(N)
      REAL              WSAVE(3 * N + 15)
C
      EXTERNAL          EZFFTb, EZFFTf, EZFFTf
      INTRINSIC          ACOS, COS, SIN
C
C      Initialize array to a sequence of real numbers.
C
      PI = ACOS (-1.0)
      DO 100, I=1, N
        R(I) = 3.0 + SIN ((I - 1.0) * 2.0 * PI / N) +
$           4.0 * COS ((I - 1.0) * 8.0 * PI / N)
      100 CONTINUE
C
```

```

CALL EZFFT1 (N, WSAVE)
PRINT 1000
PRINT 1010, (R(I), I = 1, N)
CALL EZFFT2 (N, R, AZERO, A, B, WSAVE)
PRINT 1020, AZERO
PRINT 1030
PRINT 1010, (A(I), I = 1, N)
PRINT 1040
PRINT 1010, (B(I), I = 1, N)
CALL EZFFT3 (N, R, AZERO, A, B, WSAVE)
PRINT 1050
PRINT 1010, (R(I), I = 1, N)
C
1000 FORMAT (1X, 'Original Sequence: ')
1010 FORMAT (100(F6.1, 1X))
1020 FORMAT (1X, 'Azero = ' F4.1)
1030 FORMAT (1X, 'A = ')
1040 FORMAT (1X, 'B = ')
1050 FORMAT (1X, 'Recovered Sequence: ')
C
END

```

Sample Output

```

Original Sequence:
 7.0  -0.1  7.0  1.9  4.0  3.4  0.1  5.1  -1.4
Azero = 3.0
A =
0.0  0.0  0.0  4.0  0.0  0.0  0.0  0.0  0.0
B =
1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
Recovered Sequence:
 7.0  -0.1  7.0  1.9  4.0  3.4  0.1  5.1  -1.4

```

Fourier Synthesis

The subroutines described in this section compute a periodic sequence from its Fourier coefficients. The transform is defined as:

For N even and for $i = 1, \dots, n$

$$r(i) = r(1) + (-1)^{i-1} \times r(n) + \sum_{k=2}^{n/2} \left(2.0 \times r(2 \times k - 2) \times \cos\left((k-1) \times (i-1) \times 2 \times \frac{\pi}{n}\right) - 2.0 \times r(2 \times k - 1) \times \sin\left((k-1) \times (i-1) \times 2 \times \frac{\pi}{n}\right) \right)$$

For N odd and for $i = 1, \dots, n$

$$r(i) = r(1) + \sum_{k=2}^{(n+1)/2} \left(2.0 \times r(2 \times k - 2) \times \cos\left((k-1) \times (i-1) \times 2 \times \frac{\pi}{n}\right) - 2.0 \times r(2 \times k - 1) \times \sin\left((k-1) \times (i-1) \times 2 \times \frac{\pi}{n}\right) \right)$$

Note – The xFFT operations are unnormalized, so a call of xFFTF followed by a call of xFFTB will multiply the input sequence by N. The VxFFT operations are normalized, so a call of VxFFTF followed by a call of VxFFTB will return the original sequence.

Calling Sequence

```
CALL RFFTB (N, RX, RWSAVE)
CALL DFFTB (N, DX, DWSAVE)
CALL CFFTB (N, RX, RWSAVE)
```

```

CALL ZFFTB      (N, DX, DWSAVE)
CALL VRFFTB     (M, N, RX, RXT, MDIMX, RWSAVE)
CALL VDFFTB     (M, N, DX, DXT, MDIMX, DWSAVE)

void rfftb      (int n, float *rx, float *rwsave)
void dfftb      (int n, double *rx, double *dwsave)
void cfftb      (int n, complex *rx, complex *rwsave)
void zfftb      (int n, doublecomplex *dx, doublecomplex *dwsave)
void vrfftb     (int m, int n, float *rx, int mdimx, float *rwsave)
void vdfftb     (int m, int n, double *rx, int mdimx, double *dwsave)

```

Arguments

M	(For vector operations only.) The number of sequences to be transformed. $M \geq 0$.
N	Length of the sequence to be transformed. These subroutines are most efficient when N is a product of small primes. $N \geq 0$.
xX	On entry, an array of length N containing the sequence to be transformed. For VxFFTE, a real two-dimensional array xX(M,N) whose rows contain the sequences to be transformed.
xXT	(For vector operations only.) A real two-dimensional work array with dimensions of (MDIMX $\times$ N).
MDIMX	(For vector operations only.) Leading dimension of the arrays xX and xXT as specified in a dimension or type statement. $MDIMX \geq M$.
xWSAVE	On entry, an array with dimension for scalar subroutines of at least $(2 \times N + 15)$ for real inputs or $(4 \times N + 15)$ for complex inputs. For vector subroutines, the dimension is at least (N+15). The array must have been initialized by xFFTI or VxFFTI.

Sample Program

	PROGRAM TEST	
	IMPLICIT NONE	
C	INTEGER	N
	PARAMETER	(N = 4)
C	INTEGER	I
	REAL	PI, WSAVE(4 * N + 15), X, Y
	COMPLEX	C(N)
C	EXTERNAL	CMFTB, CMFTF, CMFTI


```

      INTRINSIC      ACOS, CMPLX, COS, SIN
C
C      Initialize the array C to a complex sequence.
C
      PI = ACOS (-1.0)
      DO 100, I=1, N
        X = SIN ((I - 1.0) * 2.0 * PI / N)
        Y = COS ((I - 1.0) * 2.0 * PI / N)
        C(I) = CMPLX (X, Y)
      100 CONTINUE
C
      PRINT 1000
      PRINT 1010, (C(I), I = 1, N)
      CALL CFFTI (N, WSAVE)
      CALL CFFTF (N, C, WSAVE)
      PRINT 1020
      PRINT 1010, (C(I), I = 1, N)
      CALL CFFTB (N, C, WSAVE)
      PRINT 1030
      PRINT 1010, (C(I), I = 1, N)
C
      1000 FORMAT (1X, 'Original Sequence:')
      1010 FORMAT (1X, 100(F5.1' +',F4.1,'i '))
      1020 FORMAT (1X, 'Transformed Sequence:')
      1030 FORMAT (1X, 'Recovered Sequence:')
C
      END

```

Sample Output

```

Original Sequence:
  0.0 + 1.0i   1.0 + 0.0i   0.0 +-1.0i   -1.0 + 0.0i
Transformed Sequence:
  0.0 + 0.0i   0.0 + 0.0i   0.0 + 0.0i   0.0 + 4.0i
Recovered Sequence:
  0.0 + 4.0i   4.0 + 0.0i   0.0 +-4.0i   -4.0 + 0.0i

```

Fourier Transform

The subroutines described in this section compute the Fourier coefficients of a periodic sequence. The transform is defined as:

If N is even then let $l = n/2$, otherwise let $l = (n+1)/2$. Then,

$$r(1) = \sum_{i=1}^n r(i)$$

and for $k = 2, \dots, l$

$$r(2 \times k - 2) = \sum_{i=1}^n r(i) \times \cos\left((k-1) \times (i-1) \times 2 \times \frac{\pi}{n}\right)$$

$$r(2 \times k - 1) = \sum_{i=1}^n -r(i) \times \sin\left((k-1) \times (i-1) \times 2 \times \frac{\pi}{n}\right)$$

If N is even

$$r(n) = \sum_{i=1}^n -1^{i-1} \times r(i)$$

Note – The xFFT operations are unnormalized, so a call of xFFTF followed by a call of xFFTB will multiply the input sequence by N. The VxFFT operations are normalized, so a call of VxFFTF followed by a call of VxFFTB will return the original sequence.

Calling Sequence

```
CALL RFFTF (N, RX, RWSAVE)
CALL DFFTF (N, DX, DWSAVE)
CALL CFFTF (N, CR, CWSAVE)
CALL ZFFTF (N, ZR, ZWSAVE)
```

```

CALL VRFFTF (M, N, RX, RXT, MDIMX, RWSAVE)
CALL VDFFTF (M, N, DX, DXT, MDIMX, DWSAVE)

void rfftf (int n, float *rx, float *rwsave)
void dfftf (int n, double *dx, double *dwsave)
void cfftf (int n, complex *cr, complex *wsave)
void zfftf (int n, doublecomplex *zr, doublecomplex *wsave)
void vrfftf (int m, int n, float *rx, int mdimx, float *rwsave)
void vdfftf (int m, int n, double *dx, int mdimx, double *dwsave)

```

Arguments

M	(For vector operations only.) The number of sequences to be transformed. $M \geq 0$.
N	Length of the sequence to be transformed. These subroutines are most efficient when N is a product of small primes. $N \geq 0$.
xX	On entry, an array of length N containing the sequence to be transformed. For VxFFTF, a real two-dimensional array xX(M,N) whose rows contain the sequences to be transformed.
xXT	(For vector operations only.) A real two-dimensional work array with dimensions of (MDIMX $\times$ N).
MDIMX	(For vector operations only.) Leading dimension of the arrays xX and xXT as specified in a dimension or type statement. $MDIMX \geq M$.
xWSAVE	On entry, an array with dimension for scalar subroutines of at least $(2 \times N + 15)$ for real inputs or $(4 \times N + 15)$ for complex inputs. For vector subroutines, the dimension is at least (N+15). The array must have been initialized by xFFTI or VxFFTI.

Sample Program

	PROGRAM TEST	
	IMPLICIT NONE	
C	INTEGER	N
	PARAMETER	(N = 4)
C	INTEGER	I
	REAL	PI, WSAVE(4 * N + 15), X, Y
	COMPLEX	C(N)
C	EXTERNAL	CFFTB, CFFTF, CFFTI
	INTRINSIC	ACOS, CMPLX, COS, SIN
C		

```

C      Initialize the array C to a complex sequence.
C
      PI = ACOS (-1.0)
      DO 100, I=1, N
        X = SIN ((I - 1.0) * 2.0 * PI / N)
        Y = COS ((I - 1.0) * 2.0 * PI / N)
        C(I) = CMPLX (X, Y)
      100 CONTINUE
C
      PRINT 1000
      PRINT 1010, (C(I), I = 1, N)
      CALL CFFTI (N, WSAVE)
      CALL CFFTF (N, C, WSAVE)
      PRINT 1020
      PRINT 1010, (C(I), I = 1, N)
      CALL CFFTB (N, C, WSAVE)
      PRINT 1030
      PRINT 1010, (C(I), I = 1, N)
C
1000 FORMAT (1X, 'Original Sequence:')
1010 FORMAT (1X, 100(F5.1' +',F4.1,'i '))
1020 FORMAT (1X, 'Transformed Sequence:')
1030 FORMAT (1X, 'Recovered Sequence:')
C
      END

```

Sample Output

```

Original Sequence:
  0.0 + 1.0i    1.0 + 0.0i    0.0 +-1.0i    -1.0 + 0.0i
Transformed Sequence:
  0.0 + 0.0i    0.0 + 0.0i    0.0 + 0.0i    0.0 + 4.0i
Recovered Sequence:
  0.0 + 4.0i    4.0 + 0.0i    0.0 +-4.0i    -4.0 + 0.0i

```

Initialize Fourier Transform and Synthesis

The subroutines described in this section initialize the array xWSAVE, which is used in both xFFTF and xFFTB.

Calling Sequence

```
CALL RFFTI  (N, RWSAVE)
CALL DFFTI  (N, DWSAVE)
CALL CFFTI  (N, RWSAVE)
CALL ZFFTI  (N, DWSAVE)
CALL VRFFTI (N, RWSAVE)
CALL VDFFTI (N, DWSAVE)
```

```
void rffti  (int n, float *rwsave)
void dffti  (int n, double *dwsave)
void cffti  (int n, complex *rwsave)
void zffti  (int n, doublecomplex *dwsave)
void vrffti (int n, float *rwsave)
void vdffti (int n, double *dwsave)
```

Arguments

N	Length of the sequence to be transformed. $N \geq 0$.
xWSAVE	On entry, an array with a dimension for scalar subroutines of at least $(2 \times N + 15)$ for real inputs or $(4 \times N + 15)$ for complex inputs. For vector subroutines, the dimension is at least $(N + 15)$. The same work array can be used for both xFFTF and xFFTB as long as N remains unchanged. Different xWSAVE arrays are required for different values of N. This initialization does not have to be repeated between calls to xFFTF or xFFTB as long as N and xWSAVE remain unchanged, thus subsequent transforms can be obtained faster than the first.

Sample Program

```
PROGRAM TEST
IMPLICIT NONE
C
  INTEGER          N
  PARAMETER        (N = 9)
C
```

```

      INTEGER          I
      REAL             PI, R(N), WSAVE(2 * N + 15)
C
      EXTERNAL         RFFTB, RFFTF, RFFTI
      INTRINSIC        ACOS, SIN
C
C      Initialize array to a real sequence.
C
      PI = ACOS (-1.0)
      DO 100, I=1, N
        R(I) = 3.0 + SIN ((I - 1.0) * 2.0 * PI / N)
100 CONTINUE
C
      PRINT 1000
      PRINT 1010, (R(I), I = 1, N)
      CALL RFFTI (N, WSAVE)
      CALL RFFTF (N, R, WSAVE)
      PRINT 1020
      PRINT 1010, (R(I), I = 1, N)
      CALL RFFTB (N, R, WSAVE)
      PRINT 1030
      PRINT 1010, (R(I), I = 1, N)
C
1000 FORMAT (1X, 'Original Sequence: ')
1010 FORMAT (1X, 100(F4.1, 1X))
1020 FORMAT (1X, 'Transformed Sequence: ')
1030 FORMAT (1X, 'Recovered Sequence: ')
C
      END

```

Sample Output

```

Original Sequence:
 3.0  3.6  4.0  3.9  3.3  2.7  2.1  2.0  2.4
Transformed Sequence:
27.0  0.0 -4.5  0.0  0.0  0.0  0.0  0.0  0.0
Recovered Sequence:
27.0 32.8 35.9 34.8 30.1 23.9 19.2 18.1 21.2

```

Sine Quarter-Wave Synthesis

The subroutines described in this section synthesize a Fourier sequence from its representation in terms of a sine series with odd wave numbers. The transform is defined as:

For $i=1,\dots,n$

$$x(i) = \sum_{k=1}^n 4 \times x(k) \times \sin\left((2 \times k - 1) \times i \times \frac{\pi}{2 \times n}\right)$$

Note – The xSINQ operations are unnormalized inverses of themselves, so a call to xSINQF followed by a call to xSINQB will multiply the input sequence by $4 \times N$. The VxSINQ operations are normalized, so a call of VxSINQF followed by a call of VxSINQB will return the original sequence.

Calling Sequence

```
CALL SINQB   (N, RX, RWSAVE)
CALL DSINQB  (N, DX, DWSAVE)
CALL VSINQB  (M, N, RX, RXT, MDIMX, RWSAVE)
CALL VDSINQB (M, N, DX, DXT, MDIMX, DWSAVE)

void sinqb   (int n, float *rx, float *rwsave)
void dsinqb  (int n, double *dx, double *dwsave)
void vsinqb  (int m, int n, float *rx, int mdimx, float *rwsave)
void vdsinqb (int m, int n, double *dx, int mdimx, double *dwsave)
```

Arguments

M	(For vector operations only.) The number of sequences to be transformed. $M \geq 0$.
N	Length of the sequence to be transformed. These subroutines are most efficient when N is a product of small primes. $N \geq 0$.
xX	On entry, an array of length N containing the sequence to be transformed. For VxSINQB, a real two-dimensional array with dimensions of (MDIMX $\times$ N) whose rows contain the sequences to be transformed. On exit, the quarter-wave sine synthesis of the input.

xXT	(For vector operations only.) A real two-dimensional work array with dimensions of (MDIMX × N).
MDIMX	(For vector operations only.) Leading dimension of the arrays xX and xXT as specified in a dimension or type statement. MDIMX ≥ M.
xWSAVE	On entry, an array with dimension of at least (3 × N + 15) for scalar subroutines or (2 × N + 15) for vector subroutines, initialized by xSINQI or VxSINQI.

Sample Program

```

      PROGRAM TEST
      IMPLICIT NONE
C
      INTEGER          M, N
      PARAMETER        (M = 1)
      PARAMETER        (N = 6)
C
      INTEGER          I
      REAL              PI, WSAVE(N + 15), X(M, N), XT(M, N)
C
      EXTERNAL          VSINQB, VSINQF, VSINQI
      INTRINSIC         ACOS, SIN
C
C      Initialize the first row of the array to a real odd
C      quarter-wave sequence, that is, it can be expanded in
C      terms of a cosine series with only odd wave numbers.
C
      PI = ACOS (-1.0)
      DO 100, I=1, N
         X(M,I) = 40.0 * SIN ((I * PI / (2.0 * N)))
100  CONTINUE
C
      CALL VSINQI (N, WSAVE)
      PRINT 1000
      PRINT 1010, (X(M, I), I = 1, N)
      CALL VSINQF (M, N, X, XT, M, WSAVE)
      PRINT 1020
      PRINT 1010, (X(M, I), I = 1, N)
      CALL VSINQB (M, N, X, XT, M, WSAVE)
      PRINT 1030
      PRINT 1010, (X(M, I), I = 1, N)
C
1000 FORMAT (1X, 'Original Sequence: ')
1010 FORMAT (1X, 100(F5.1, 1X))
1020 FORMAT (1X, 'Transformed Sequence: ')
1030 FORMAT (1X, 'Recovered Sequence: ')

```



```
C
      END
```

Sample Output

```
Original Sequence:
10.4 20.0 28.3 34.6 38.6 40.0
Transformed Sequence:
49.0 0.0 0.0 0.0 0.0 0.0
Recovered Sequence:
10.4 20.0 28.3 34.6 38.6 40.0
```

Sine Quarter-Wave Transform

The subroutines described in this section compute the Fourier coefficients in a sine series representation with only odd wave numbers. The transform is defined as:

For $i=1,\dots,n$

$$x(i) = (-1)^{i-1} \times x(n) + \sum_{k=1}^{n-1} 2 \times x(k) \times \sin\left((2 \times i - 1) \times k \times \frac{\pi}{2 \times n}\right)$$

Note – The xSINQ operations are unnormalized inverses of themselves, so a call to xSINQF followed by a call to xSINQB will multiply the input sequence by $4 \times N$. The VxSINQ operations are normalized, so a call of VxSINQF followed by a call of VxSINQB will return the original sequence.

Calling Sequence

```
CALL SINQF    (N, RX, RWSAVE)
CALL DSINQF   (N, DX, DWSAVE)
CALL VSINQF   (M, N, RX, RXT, MDIMX, RWSAVE)
CALL VDSINQF  (M, N, DX, DXT, MDIMX, DWSAVE)

void singf    (int n, float *rx, float *rwsave)
void dsingf   (int n, double *dx, double *dwsave)
void vsingf   (int m, int n, float *rx, int mdimx, float *rwsave)
void vdsingf  (int m, int n, double *dx, int mdimx, double *dwsave)
```

Arguments

M	(For vector operations only.) The number of sequences to be transformed. $M \geq 0$.
N	Length of the sequence to be transformed. These subroutines are most efficient when N is a product of small primes. $N \geq 0$.
xX	On entry, an array of length N containing the sequence to be transformed. For VxSINQF, a real two-dimensional array with dimensions of (MDIMX $\times$ N) whose rows contain the sequences to be transformed. On exit, the quarter-wave sine transform of the input.

xXT	(For vector operations only.) A real two-dimensional work array with dimensions of (MDIMX $\times$ N).
MDIMX	(For vector operations only.) Leading dimension of the arrays xX and xXT as specified in a dimension or type statement. MDIMX $\geq$ M.
xWSAVE	On entry, an array with dimension of at least (3 $\times$ N + 15) for scalar subroutines or (2 $\times$ N + 15) for vector subroutines, initialized by xSINQI or VxSINQI.

Sample Program

```

      PROGRAM TEST
      IMPLICIT NONE
C
      INTEGER          M, N
      PARAMETER        (M = 4)
      PARAMETER        (N = 6)
C
      INTEGER          I, J
      REAL             PI, WSAVE(N + 15), X(M, N+1), XT(M, N + 1)
C
      EXTERNAL          VSINQB, VSINQF, VSINQI
      INTRINSIC         ACOS, SIN
C
C      Initialize the array to m real odd quarter-wave sequence,
C      that is, they can be expanded in terms of a cosine series
C      with only odd wave numbers.
C
      PI = ACOS (-1.0)
      DO 110, J=1, M
        DO 100, I=1, N
          X(J,I) = 40.0 * J * SIN (I * PI / (2.0 * N))
100    CONTINUE
110   CONTINUE
C
      CALL VSINQI (N, WSAVE)
      PRINT 1000
      DO 120, J=1, M
        PRINT 1010, J, (X(J, I), I = 1, N)
120   CONTINUE
      CALL VSINQF (M, N, X, XT, M, WSAVE)
      PRINT 1020
      DO 130, J=1, M
        PRINT 1010, J, (X(J, I), I = 1, N)
130   CONTINUE
      CALL VSINQB (M, N, X, XT, M, WSAVE)
      PRINT 1030
      DO 140, J=1, M
        PRINT 1010, J, (X(J, I), I = 1, N)
140   CONTINUE

```

```

C
1000 FORMAT (1X, 'Original Sequence: ')
1010 FORMAT (1X, ' Sequence', I2, ': ', 100(F5.1, 1X))
1020 FORMAT (1X, 'Transformed Sequence: ')
1030 FORMAT (1X, 'Recovered Sequence: ')
C
      END

```

Sample Output

```

Original Sequence:
Sequence 1:  10.4  20.0  28.3  34.6  38.6  40.0
Sequence 2:  20.7  40.0  56.6  69.3  77.3  80.0
Sequence 3:  31.1  60.0  84.9 103.9 115.9 120.0
Sequence 4:  41.4  80.0 113.1 138.6 154.5 160.0
Transformed Sequence:
Sequence 1:  49.0   0.0   0.0   0.0   0.0   0.0
Sequence 2:  98.0   0.0   0.0   0.0   0.0   0.0
Sequence 3: 147.0   0.0   0.0   0.0   0.0   0.0
Sequence 4: 196.0   0.0   0.0   0.0   0.0   0.0
Recovered Sequence:
Sequence 1:  10.4  20.0  28.3  34.6  38.6  40.0
Sequence 2:  20.7  40.0  56.6  69.3  77.3  80.0
Sequence 3:  31.1  60.0  84.9 103.9 115.9 120.0
Sequence 4:  41.4  80.0 113.1 138.6 154.5 160.0

```

Initialize Sine Quarter-Wave Transform and Synthesis

The subroutines described in this section initialize the array xWSAVE, which is used in both xSINQF and xSINQB.

Calling Sequence

```
CALL SINQI   (N, RWSAVE)
CALL DSINQI  (N, DWSAVE)
CALL VSINQI  (N, RWSAVE)
CALL VDSINQI (N, DWSAVE)
```

```
void sinqi   (int n, float *rwsave)
void dsinqi  (int n, double *dwsave)
void vsinqi  (int n, float *rwsave)
void vdsinqi (int n, double *dwsave)
```

Arguments

N	Length of the sequence to be transformed. The method is most efficient when N is a product of small primes.
xWSAVE	On entry, an array with a dimension of at least $(3 \times N + 15)$ for scalar subroutines or $(2 \times N + 15)$ for vector subroutines. The same work array can be used for both xSINQF and xSINQB as long as N remains unchanged. Different xWSAVE arrays are required for different values of N. This initialization does not have to be repeated between calls to xCSINF or xCSINB as long as N and xWSAVE remain unchanged, thus subsequent transforms can be obtained faster than the first.

Sample Program

	PROGRAM TEST	
	IMPLICIT NONE	
C	INTEGER	N
	PARAMETER	(N = 6)
C	INTEGER	I
	REAL	PI, WSAVE(3 * N + 15), X(N)
C	EXTERNAL	SINQB, SINQF, SINQI

```

C      INTRINSIC          ACOS, SIN
C
C      Initialize array X to a real odd quarter-wave sequence,
C      that is, it can be expanded in terms of a sine series with
C      only odd wave number.
C
C      PI = ACOS (-1.0)
C      DO 100, I=1, N
C          X(I) = 40.0 * SIN (I * PI / (2.0 * N))
100 CONTINUE
C
C      PRINT 1000
C      PRINT 1010, (X(I), I = 1, N)
C      CALL SINQI (N, WSAVE)
C      CALL SINQF (N, X, WSAVE)
C      PRINT 1020
C      PRINT 1010, (X(I), I = 1, N)
C      CALL SINQB(N, X, WSAVE)
C      PRINT 1030
C      PRINT 1010, (X(I), I = 1, N)
C
C      1000 FORMAT (1X, 'Original Sequence: ')
C      1010 FORMAT (1X, 100(F6.1, 1X))
C      1020 FORMAT (1X, 'Transformed Sequence: ')
C      1030 FORMAT (1X, 'Recovered Sequence: ')
C
C      END

```

Sample Output

```

Original Sequence:
  10.4  20.0  28.3  34.6  38.6  40.0
Transformed Sequence:
 240.0   0.0   0.0   0.0   0.0   0.0
Recovered Sequence:
 248.5 480.0 678.8 831.4 927.3 960.0

```

Sine Odd-Wave Transform

The subroutines described in this section compute the discrete Fourier sine transform of an odd sequence. The transform is defined as:

For $i=1,\dots,n$

$$x(i) = \sum_{k=1}^n 2 \times x(k) \times \sin\left(k \times i \times \frac{\pi}{n+1}\right)$$

Note – The xSINT transforms are unnormalized inverses of themselves, so a call of xSINT followed by another call of xSINT will multiply the input sequence by $2 \times (N+1)$. The VxSINT transforms are normalized, so a call of VxSINT followed by a call of VxSINT will return the original sequence.

Calling Sequence

```
CALL SINT      (N, RX, RWSAVE)
CALL DSINT     (N, DX, DWSAVE)
CALL VSINT     (M, N, RX, RXT, MDIMX, RWSAVE)
CALL VDSINT    (M, N, DX, DXT, MDIMX, DWSAVE)

void sint      (int n, float *rx, float *rwsave)
void dsint     (int n, double *dx, double *dwsave)
void vsint     (int m, int n, float *rx, int mdimx, float *rwsave)
void vdsint    (int m, int n, double *dx, int mdimx, double *dwsave)
```

Arguments

M	(For vector operations only.) The number of sequences to be transformed. $M \geq 0$.
N	Length of the sequence to be transformed. These subroutines are most efficient when $N+1$ is a product of small primes. $N \geq 0$.
xX	On entry, an array of length N containing the sequence to be transformed. For VxSINT, a real two-dimensional array with dimensions of $(MDIMX \times (N+1))$ whose rows contain the sequences to be transformed. On exit, the sine transform of the input.

xXT	(For vector operations only.) A real two-dimensional work array with dimensions of $(MDIMX \times (N+1))$.
MDIMX	(For vector operations only.) Leading dimension of the arrays xX and xXT as specified in a dimension or type statement. $MDIMX \geq M$.
xWSAVE	On entry, an array with dimension of at least $(3 \times N + 15)$ for scalar subroutines or $(2 \times N + 15)$ for vector subroutines, initialized by xCOSQI or VxCOSQI.
xWSAVE	On entry, an array with dimension of at least $\text{int}(2.5 \times N + 15)$ initialized by xSINTI or VxSINTI.

Sample Program

```

      PROGRAM TEST
      IMPLICIT NONE
C
      INTEGER          N
      PARAMETER        (N = 9)
C
      INTEGER          I
      REAL             PI, WSAVE(3 * N + 15), X(N)
C
      EXTERNAL          SINT, SINTI
      INTRINSIC         ACOS, SIN
C
C      Initialize the array X to an odd sequence, that is, it
C      can be expanded in terms of a trigonometric series that
C      contains only sine terms.
C
      PI = ACOS (-1.0)
      DO 100, I=1, N
         X(I) = SIN ( I * 2.0 * PI / (N + 1.0))
100  CONTINUE
C
      PRINT 1000
      PRINT 1010, (X(I), I = 1, N)
      CALL SINTI (N, WSAVE)
      CALL SINT (N, X, WSAVE)
      PRINT 1020
      PRINT 1010, (X(I), I = 1, N)
      CALL SINT (N, X, WSAVE)
      PRINT 1030
      PRINT 1010, (X(I), I = 1, N)
C
1000 FORMAT (1X, 'Original Sequence: ')
1010 FORMAT (1X, 100(F7.3, 1X))
1020 FORMAT (1X, 'Transformed Sequence: ')
1030 FORMAT (1X, 'Recovered Sequence: ')

```



```
C
      END
```

Sample Output

```
Original Sequence:
 0.588  0.951  0.951  0.588  0.000 -0.588 -0.951 -0.951 -0.588
Transformed Sequence:
 0.000 10.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
Recovered Sequence:
11.756 19.021 19.021 11.756  0.000 -11.756 -19.021 -19.021 -11.756
```

Initialize Sine Odd-Wave Transform

The subroutines described in this section initialize the array xWSAVE, which is used in subroutine xSINT.

Calling Sequence

```
CALL SINTI    (N, RWSAVE)
CALL DSINTI   (N, DWSAVE)
CALL VSINTI   (N, WSAVE)
CALL VDSINTI  (N, WSAVE)

void sinti    (int n, float *rwsave)
void dsinti   (int n, double *dwsave)
void vsinti   (int n, float *rwsave)
void vdsinti  (int n, double *dwsave)
```

Arguments

N	Length of the sequence to be transformed. $N \geq 0$.
xWSAVE	On entry, an array with a dimension of at least $(3 \times N + 15)$. Different xWSAVE arrays are required for different values of N. This initialization does not have to be repeated between calls to xSINT as long as N and xWSAVE remain unchanged, thus subsequent transforms can be obtained faster than the first.

Sample Program

```

      PROGRAM TEST
      IMPLICIT NONE
C
      INTEGER          N
      PARAMETER        (N = 9)
C
      INTEGER          I
      REAL              PI, WSAVE(3 * N + 15), X(N)
C
      EXTERNAL          SINT, SINTI
      INTRINSIC          ACOS, SIN
C
C      Initialize the array X to an odd sequence, that is, it
C      can be expanded in terms of a trigonometric series that
C      contains only sine terms.
```

```
C
      PI = ACOS (-1.0)
      DO 100, I=1, N
        X(I) = SIN ( I * 2.0 * PI / (N + 1.0))
100  CONTINUE
C
      PRINT 1000
      PRINT 1010, (X(I), I = 1, N)
      CALL SINTI (N, WSAVE)
      CALL SINT (N, X, WSAVE)
      PRINT 1020
      PRINT 1010, (X(I), I = 1, N)
      CALL SINT (N, X, WSAVE)
      PRINT 1030
      PRINT 1010, (X(I), I = 1, N)
C
1000 FORMAT (1X, 'Original Sequence: ')
1010 FORMAT (1X, 100(F7.3, 1X))
1020 FORMAT (1X, 'Transformed Sequence: ')
1030 FORMAT (1X, 'Recovered Sequence: ')
C
      END
```

Sample Output

```
Original Sequence:
 0.588  0.951  0.951  0.588  0.000 -0.588 -0.951 -0.951 -0.588
Transformed Sequence:
 0.000 10.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
Recovered Sequence:
11.756 19.021 19.021 11.756  0.000 -11.756 -19.021 -19.021 -11.756
```


List of Subprograms by Name



This Appendix lists the subprograms in the seven libraries of Sun Performance Library and their arguments; the list is sorted alphabetically by subprogram name. Note that the Fortran interfaces are shown in uppercase letters, while the C interfaces are shown in lowercase.

C

CALL CGBCON (NORM, N, NSUB, NSUPER, CA, LDA, IPIVOT, SANORM, SRCOND, CWORK, SWORK2, INFO)cgbcon]8

CALL CAXPY (N, CALPHA, CX, INCX, CY, INCY)687

void caxpy (int n, complex *ca, complex *cx, int incx, complex *cy, int incy)687

CALL CCHDC (CA, LDA, N, CWORK, IPIVOT, JOB, INFO)535

void cchdc (complex *ca, int lda, int n, int *ipivot, int job, int *info)535

CALL CCHDD (CA, LDA, N, CX, CZ, LDZ, NZ, CY, SRHO, SCOS, SSIN, INFO)538

void cchdd (complex *ca, int lda, int n, complex *cx, complex *cz, int ldz, int nz, complex *cy, double *drho, double *dcos, double *dsin, int *info)538

CALL CCHEX (ZA, LDA, N, K, L, ZZ, LDZ, NZ, DCOS, ZSIN, JOB)542

void cchex (complex *r, int lda, int n, int k, int l, complex *cz, int ldz, int nz, float *scos, complex *csin, int job)542

CALL CCHUD (CA, LDA, N, CX, CZ, LDZ, NZ, CY, SRHO, SCOS, CSIN)546

void cchud (complex *ca, int lda, int n, complex *cx, complex *cz, int ldz, int nz, complex *cy, float *srho, float *scos, complex *csin)546

CALL CCOPY (N, CX, INCX, CY, INCY)690

void ccopy (int n, complex *cx, int incx, complex *cy, int incy)690

CINNER = CDOTC (N, CX, INCX, CY, INCY)694

complex *cdotc (int n, complex *cx, int incx, complex *cy, int incy)694

CINNER = CDOTU (N, CX, INCX, CY, INCY)694

complex *cdotu (int n, complex *cx, int incx, complex *cy, int incy)694

CALL CFFTB (N, RX, RWSAVE)865

void cfftb (int n, complex *rx, complex *rwsave)865

CALL CFFTF (N, CR, CWSAVE)868

void cfftf (int n, complex *cr, complex *wsave)869

CALL CFFTI (N, RWSAVE)871

void cffti (int n, complex *rwsave)871

CALL CGBCO (CA, LDA, N, NSUB, NSUPER, IPIVOT, SRCOND, CWORK)549

void cgbco (complex *ca, int lda, int n, int nsub, int nsuper, int *ipivot, float *srcond)549

void cgbcon (char norm, int n, int nsub, int nsuper, complex *ca, int lda, int *ipivot, float sanorm, float *srcond, int *info)8

CALL CGBDI (CA, LDA, N, NSUB, NSUPER, IPIVOT, CDET)553

void cgbdi (complex *ca, int lda, int n, int nsub, int nsuper, int *ipivot, complex *cdet)553

CALL CGBEQU (M, N, NSUB, NSUPER, CA, LDA, SROWSC, SCOLSC, SROWCN, SCOLCN, SAMAX, INFO)12

void cgbequ (int m, int n, int nsub, int nsuper, complex *ca, int lda, float *srowsc, float *scolsc, float *srowcn, float *scolcn, float *samax, int *info)12

CALL CGBFA (CA, LDA, N, NSUB, NSUPER, IPIVOT, INFO)556

void cgbfa (complex *ca, int lda, int n, int nsub, int nsuper, int *ipivot, int *info)556

CALL CGBMV (TRANSA, M, N, NSUB, NSUPER, CALPHA, CA, LDA, CX, INCX, CBETA, CY, INCY)723

void cgbmv (char trans, int m, int n, int nsub, int nsuper, complex *calpha, complex *ca, int lda, complex *cx, int incx, complex *cbeta, complex *cy, int incy)723

CALL CGBRFS (TRANSA, N, NSUB, NSUPER, NRHS, CA, LDA, CAF, LDAF, IPIVOT, CB, LDB, CX, LDX, CFERR, CBERR, CWORK, SWORK2, INFO)15

void cgbtrfs (char trans, int n, int nsub, int nsuper, int nrhs, complex *ca, int lda, complex *caf, int ldaf, int *ipivot, complex *cb, int ldb, complex *cx, int ldx, float *sferr, float *sberr, int *info)15

CALL CGBSL (CA, LDA, N, NSUB, NSUPER, IPIVOT, CB, JOB)560

void cgbsl (complex *ca, int lda, int n, int nsub, int nsuper, int *ipivot, complex *cb, int job)560

CALL CGBSV (N, NSUB, NSUPER, NRHS, CA, LDA, IPIVOT, CB, LDB, INFO)21

void cgbsv (int n, int nsub, int nsuper, int nrhs, complex *ca, int lda, int *ipivot, complex *cb, int ldb, int *info)21

CALL CGBSVX (FACT, TRANSA, N, NSUB, NSUPER,
NRHS, CA, LDA, CAF, LDAF, IPIVOT, EQUED,
SROWSC, SCOLSC, CB, LDB, CX, LDX, SRCOND,
SFERR, SBERR, CWORK, SWORK2, INFO)25

void cgbsvx (char fact, char trans, int n, int nsub, int nsuper,
int nrhs, complex *ca, int lda, complex *caf, int ldaf,
int *ipivot, char *equed, float *srowsc, float *scolsc
complex *cb, int ldb, complex *cx, int ldx, float
*srcond, float *sferr, float *sberr, int *info)26

CALL CGBTRF (M, N, NSUB, NSUPER, CA, LDA, IPIVOT,
INFO)33

void cgbtrf (int m, int n, int nsub, int nsuper, complex *ca, int
lda, int *ipivot, int *info)33

CALL CGBTRS (TRANSA, N, NSUB, NSUPER, NRHS, CA,
LDA, IPIVOT, CB, LDB, INFO)37

void cgbtrs (char trans, int n, int nsub, int nsuper, int nrhs,
complex *ca, int lda, int *ipivot, complex *cb, int
ldb, int *info)37

CALL CGECO (CA, LDA, N, IPIVOT, SRCOND, CWORK)
564

void cgeco (complex *ca, int lda, int n, int *ipivot, float
*srcond)564

CALL CGECON (NORM, N, CA, LDA, SANORM,
SRCOND, CWORK, SWORK2, INFO)41

void cgecon (char norm, int n, complex *ca, int lda, float
sanorm, float *srcond, int *info)41

CALL CGEDI (CA, LDA, N, IPIVOT, CDET, CWORK, JOB)
567

void cgedi (complex *ca, int lda, int n, int *ipivot, complex
*cdet, int job)567

CALL CGEEQU (M, N, CA, LDA, SROWSC, SCOLSC,
SROWCN, SCOLCN, SAMAX, INFO)44

void cgeequ (int m, int n, complex *ca, int lda, float *srowsc,
float *scolsc, float *srowcn, float *scolcn, float
*samax, int *info)44

CALL CGEES (JOBZ, SORTEV, SELECT, N, CA, LDA,
NOUT, CW, CZ, LDZ, CWORK, LDWORK,
SWORK2, BWORK3, INFO)48

void cgees (char jobvs, char sort, L_fp select, int n, complex
*ca, int lda, int *sdim, complex *cw, complex *cz, int
ldz, int *info)48

CALL CGEESX (JOBZ, SORTEV, SELECT, SENSE, N, CA,
LDA, NOUT, CW, CZ, LDZ, SRCONE, SRCONV,
CWORK, LDWORK, SWORK2, BWORK3, INFO)53

void cgeesx (char jobvs, char sort, L_fp select, char sense, int
n, complex *ca, int lda, int *sdim, complex *cw,
complex *cz, int ldz, float *srcone, float *srconv, int
*info)54

CALL CGEEV (JOBVL, JOBVR, N, CA, LDA, CW, CVL,
LDVL, CVR, LDVR, CWORK, LDWORK,
SWORK2, INFO)59

void cgeev (char jobvl, char jobvr, int n, complex *ca, int lda,
complex *cw, complex *vl, int ldvl, complex *cvr, int
ldvr, int *info)59

CALL CGEEVX (BALANC, JOBVL, JOBVR, SENSE, N, CA,
LDA, CW, CVL, LDVL, CVR, LDVR, ILO, IHI,
SSCALE, SABNRM, SRCONE, SRCONV, CWORK,
LDWORK, SWORK2, INFO)63

void cgeevx (char balanc, char jobvl, char jobvr, char sense,
int n, complex *ca, int lda, complex *cw, complex
*vl, int ldvl, complex *cvr, int ldvr, int *ilo, int *ihi,
float *sscale, float *sabnrm, float *srcone, float
*srconv, int *info)64

CALL CGEFA (CA, LDA, N, IPIVOT, INFO)570

void cgefa (complex *ca, int lda, int n, int *ipivot, int *info)
570

CALL CGEGS (JOBVSL, JOBVSR, N, CA, LDA, CB, LDB,
CALPHA, CBETA, CVSL, LDVSL, CVSR, LDVSR,
CWORK, LDWORK, SWORK2, INFO)69

void cgegs (char jobvsl, char jobvsr, int n, complex *ca, int
lda, complex *cb, int ldb, complex *calpha, complex
*cbeta, complex *cvsl, int ldvsl, complex *cvsr, int
ldvsr, int *info)70

CALL CGEGV (JOBVL, JOBVR, N, CA, LDA, CB, LDB,
CALPHA, CBETA, CVL, LDVL, CVR, LDVR,
CWORK, LDWORK, SWORK2, INFO)74

void cgegv (char jobvl, char jobvr, int n, complex *ca, int lda,
complex *cb, int ldb, complex *calpha, complex
*cbeta, complex *cvl, int ldvl, complex *cvr, int ldvr,
int *info)75

CALL CGELQF (M, N, CA, LDA, CTAU, CWORK,
LDWORK, INFO)79

void cgelqf (int m, int n, complex *ca, int lda, complex *ctau,
int *info)79

CALL CGELS (TRANSA, M, N, NRHS, CA, LDA, CB, LDB, CWORK, LDWORK, INFO)82

void cgels (char trans, int m, int n, int nrhs, complex *ca, int lda, complex *cb, int ldb, int *info)82

CALL CGELSS (M, N, NRHS, CA, LDA, CB, LDB, SSING, SRCOND, IRANK, CWORK, LDWORK, SWORK2, INFO)86

void cgelss (int m, int n, int nrhs, complex *ca, int lda, complex *cb, int ldb, float *ssing, float rcond, int *rank, int *info)86

CALL CGELSX (M, N, NRHS, CA, LDA, CB, LDB, JPIVOT, SRCOND, IRANK, CWORK, SWORK2, INFO)90

void cgelsx (int m, int n, int nrhs, complex *ca, int lda, complex *cb, int ldb, int *jpivot, float rcond, int *rank, int *info)90

CALL CGEMM (TRANSA, TRANSB, M, N, K, CALPHA, CA, LDA, CB, LDB, CBETA, CC, LDC)805

void cgemm (char transa, char transb, int m, int n, int k, complex *calpha, complex *ca, int lda, complex *cb, int ldb, complex *cbeta, complex *cc, int ldc)805

CALL CGEMV (TRANSA, M, N, CALPHA, CA, LDA, CX, INCX, CBETA, CY, INCY)727

void cgemv (char trans, int m, int n, complex *calpha, complex *ca, int lda, complex *cx, int incx, complex *cbeta, complex *cy, int incy)727

CALL CGEQLF (M, N, CA, LDA, CTAU, CWORK, LDWORK, INFO)94

void cgeqlf (int m, int n, complex *ca, int lda, complex *ctau, int *info)94

CALL CGEQPF (M, N, CA, LDA, JPIVOT, CTAU, CWORK, SWORK2, INFO)97

void cgeqpf (int m, int n, complex *ca, int lda, int *jpivot, complex *ctau, int *info)97

CALL CGEQRF (M, N, CA, LDA, CTAU, CWORK, LDWORK, INFO)100

void cgeqrf (int m, int n, complex *ca, int lda, complex *ctau, int *info)100

CALL CGERC (M, N, CALPHA, CX, INCX, CY, INCY, CA, LDA)730

void cgerc (int m, int n, complex *calpha, complex *cx, int incx, complex *cy, int incy, complex *ca, int lda)730

CALL CGERFS (TRANSA, N, NRHS, CA, LDA, CAF, LDAF, IPIVOT, CB, LDB, CX, LDX, SFERR, SBERR, CWORK, SWORK2, INFO)103

void cgerfs (char trans, int n, int nrhs, complex *ca, int lda, complex *caf, int laf, int *ipivot, complex *cb, int ldb, complex *cx, int ldx, float *sferr, float *sberr, int *info)103

CALL CGERQF (M, N, CA, LDA, CTAU, CWORK, LDWORK, INFO)108

void cgerqf (int m, int n, complex *ca, int lda, complex *ctau, int *info)108

CALL CGERU (M, N, CALPHA, CX, INCX, CY, INCY, CA, LDA)730

void cgeru (int m, int n, complex *calpha, complex *cx, int incx, complex *cy, int incy, complex *ca, int lda)730

CALL CGESL (CA, LDA, N, IPIVOT, CB, JOB)572

void cgesl (complex *ca, int lda, int n, int *ipivot, complex *cb, int job)572

CALL CGESV (N, NRHS, CA, LDA, IPIVOT, CB, LDB, INFO)111

void cgesv (int n, int nrhs, complex *ca, int lda, int *ipivot, complex *cb, int ldb, int *info)111

CALL CGESVD (JOB, JOBVT, M, N, CA, LDA, SSING, CU, LDU, CVT, LDVT, CWORK, LDWORK, SWORK2, INFO)114

void cgesvd (char job, char jobvt, int m, int n, complex *ca, int lda, float *ssing, complex *u, int ldu, complex *cvt, int ldvt, int *info)114

CALL CGESVX (FACT, TRANSA, N, NRHS, CA, LDA, CAF, LDAF, IPIVOT, EQUED, CROWSC, CCOLSC, CB, LDB, CX, LDX, SRCOND, SFERR, SBERR, CWORK, SWORK2, INFO)119

void cgesvx (char fact, char trans, int n, int nrhs, complex *ca, int lda, complex *caf, int laf, int *ipivot, char *eued, float *srowsc, float *scolsc, complex *cb, int ldb, complex *cx, int ldx, float *srcond, float *sferr, float *sberr, int *info)120

CALL CGETRF (M, N, CA, LDA, IPIVOT, INFO)126

void cgetrf (int m, int n, complex *ca, int lda, int *ipivot, int *info)126

CALL CGETRI (N, CA, LDA, IPIVOT, CWORK, LDWORK, INFO)129


```

void cgetri (int n, complex *ca, int lda, int *ipivot, int *info)
129
CALL CGETRS (TRANSA, N, NRHS, CA, LDA, IPIVOT, CB,
LDB, INFO)132
void cgetrs (char trans, int n, int nrhs, complex *ca, int lda,
int *ipivot, complex *cb, int ldb, int *info)132
CALL CGGGLM (N, M, P, CA, LDA, CB, LDB, CD, CX, CY,
CWORK, LDWORK, INFO)135
void cggglm (int n, int m, int p, complex *ca, int lda, complex
*cb, int ldb, complex *cd, complex *cx, complex *cy,
int *info)136
CALL CGGLSE (M, N, P, CA, LDA, CB, LDB, CC, CD, CX,
CWORK, LDWORK, INFO)139
void cgglse (int m, int n, int p, complex *ca, int lda, complex
*cb, int ldb, complex *cc, complex *cd, complex *cx,
int *info)139
CALL CGGSVD (JOBV, JOBV, JOBQ, M, N, P, K, L, CA, LDA,
CB, LDB, SALPHA, SBETA, CU, LDU, CV, LDV,
CQ, LDQ, CWORK, SWORK2, IWORK3, INFO)144
void cggsvd (char jobv, char jobv, char jobq, int m, int n, int
p, int *k, int *l, complex *ca, int lda, complex *cb, int
ldb, float *salpha, float *sbeta, complex *cu, int ldu,
complex *cv, int ldv, complex *cq, int ldq, int *info)
145
CALL CGTCON (NORM, N, CLOW, CDIAG, CUP1, CUP2,
IPIVOT, SANORM, SRCOND, CWORK, INFO)149
void cgtcon (char norm, int n, complex *clow, complex
*cdiag, complex *cup1, complex *cup2, int *ipivot,
float sanorm, float *srcond, int *info)149
CALL CGTRFS (TRANSA, N, NRHS, CLOW, CDIAG, CUP,
CLOWE, CDIAGF, CUPF1, CUPF2, IPIVOT, CB,
LDB, CX, LDX, SFERR, SBERR, CWORK, SWORK2,
INFO)153
void cgtrfs (char trans, int n, int nrhs, complex *clow,
complex *cdiag, complex *cup, complex *clowf,
complex *cdiagf, complex *cupf1, complex *cupf2,
int *ipivot, complex *cb, int ldb, complex *cx, int ldx,
float *sferr, float *sberr, int *info)154
CALL CGTSL (N, CSUB, CDIAG, CSUPER, CB, INFO)575
void cgtsl (int n, complex *csub, complex *cdiag, complex
*csuper, complex *cb, int *info)575
CALL CGTSV (N, NRHS, CLOW, CDIAG, CUP, CB, LDB,
INFO)160
void cgtsv (int n, int nrhs, complex *clow, complex *cdiag,
complex *cup, complex *cb, int ldb, int *info)160
CALL CGTSVX (FACT, TRANSA, N, NRHS, CLOW,
CDIAG, CUP, CLOWE, CDIAGF, CUPF1, CUPF2,
IPIVOT, CB, LDB, CX, LDX, SRCOND, SFERR,
SBERR, CWORK, SWORK2, INFO)163
void cgtsvx (char fact, char trans, int n, int nrhs, complex
*clow, complex *cdiag, complex *cup, complex
*clowf, complex *cdiagf, complex *cupf1, complex
*cupf2, int *ipivot, complex *cb, int ldb, complex
*cx, int ldx, float *srcond, float *sferr, float *sberr, int
*info)164
CALL CGTTRF (N, CLOW, CDIAG, CUP1, CUP2, IPIVOT,
INFO)169
void cgttrf (int n, complex *clow, complex *cdiag, complex
*cup1, complex *cup2, int *ipivot, int *info)169
CALL CGTTRS (TRANSA, N, NRHS, CLOW, CDIAG,
CUP1, CUP2, IPIVOT, CB, LDB, INFO)173
void cgttrs (char trans, int n, int nrhs, complex *clow,
complex *cdiag, complex *cup1, complex *cup2, int
*ipivot, complex *cb, int ldb, int *info)173
CALL CHBEV (JOBZ, UPLO, N, NDIAG, CA, LDA, SW, CZ,
LDZ, CWORK, SWORK2, INFO)177
void chbev (char jobz, char uplo, int n, int ndiag, complex
*ca, int lda, float *sw, complex *cz, int ldz, int *info)
177
CALL CHBEVX (JOBZ, RANGE, UPLO, N, NDIAG, CA,
LDA, CQ, LDQ, SVL, SVU, IL, IU, SABTOL,
NFOUND, SW, CZ, LDZ, CWORK, SWORK2,
IWORK3, IFAIL, INFO)181
void chbev (char jobz, char range, char uplo, int n, int
ndiag, complex *ca, int lda, complex *q, int ldq,
float svl, float svu, int il, int iu, float sabtol, int
*nfound, float *sw, complex *cz, int ldz, int *ifail, int
*info)181
CALL CHBMV (UPLO, N, NDIAG, CALPHA, CA, LDA,
CX, INCX, CBETA, CY, INCY)734
void chbm (char uplo, int n, int ndiag, complex *calpha,
complex *ca, int lda, complex *cx, int incx, complex
*cbeta, complex *cy, int incy)734
CALL CHECON (UPLO, N, CA, LDA, IPIVOT, SANORM,
SRCOND, CWORK, INFO)186
void checon (char uplo, int n, complex *ca, int lda, int
*ipivot, float sanorm, float *srcond, int *info)186

```

CALL CHEEV (JOBZ, UPLO, N, CA, LDA, SW, CWORK, LDWORK, SWORK2, INFO)190

void cheev (char jobz, char uplo, int n, complex *ca, int lda, float *sw, int *info)190

CALL CHEEVX (JOBZ, RANGE, UPLO, N, CA, LDA, SVL, SVU, IL, IU, SABTOL, NFOUND, SW, CZ, LDZ, CWORK, LDWORK, SWORK2, IWORK3, IFAIL, INFO)193

void cheevx (char jobz, char range, char uplo, int n, complex *ca, int lda, float svl, float svu, int il, int iu, float sabtol, int *nfound, float *sw, complex *cz, int ldz, int *ifail, int *info)193

CALL CHEGV (ITYPE, JOBZ, UPLO, N, CA, LDA, CB, LDB, SW, CWORK, LDWORK, SWORK2, INFO)198

void chegv (int itype, char jobz, char uplo, int n, complex *ca, int lda, complex *cb, int ldb, float *sw, int *info)198

CALL CHEMM (SIDE, UPLO, M, N, CALPHA, CA, LDA, CB, LDB, CBETA, CC, LDC)809

void chemm (char side, char uplo, int m, int n, complex *calpha, complex *ca, int lda, complex *cb, int ldb, complex *cbeta, complex *cc, int ldc)809

CALL CHEMV (UPLO, N, CALPHA, CA, LDA, CX, INCX, CBETA, CY, INCY)738

void chemv (char uplo, int n, complex *calpha, complex *ca, int lda, complex *cx, int incx, complex *cbeta, complex *cy, int incy)738

CALL CHER (UPLO, N, SALPHA, CX, INCX, CA, LDA)742

void cher (char uplo, int n, float salpha, complex *cx, int incx, complex *ca, int lda)742

CALL CHER2 (UPLO, N, CALPHA, CX, INCX, CY, INCY, CA, LDA)745

void cher2 (char uplo, int n, complex *calpha, complex *cx, int incx, complex *cy, int incy, complex *ca, int lda)745

CALL CHER2K (UPLO, TRANSA, N, K, CALPHA, CA, LDA, CB, LDB, SBETA, CC, LDC)815

void cher2k (char uplo, char trans, int n, int k, complex *calpha, complex *ca, int lda, complex *cb, int ldb, float sbeta, complex *cc, int ldc)815

CALL CHERFS (UPLO, N, NRHS, CA, LDA, CAF, LDAF, IPIVOT, CB, LDB, CX, LDX, SFERR, SBERR, CWORK, SWORK2, INFO)203

void cherfs (char uplo, int n, int nrhs, complex *ca, int lda, complex *caf, int ldaf, int *ipivot, complex *cb, int ldb, complex *cx, int ldx, float *sferr, float *sberr, int *info)203

CALL CHERK (UPLO, TRANSA, N, K, SALPHA, CA, LDA, SBETA, CC, LDC)812

void cherk (char uplo, char trans, int n, int k, float salpha, complex *ca, int lda, float sbeta, complex *cc, int ldc)812

CALL CHESV (UPLO, N, NRHS, CA, LDA, IPIVOT, CB, LDB, SWORK, LDWORK, INFO)209

void chesv (char uplo, int n, int nrhs, complex *ca, int lda, int *ipivot, complex *cb, int ldb, int *info)209

CALL CHESVX (FACT, UPLO, N, NRHS, CA, LDA, CAF, LDAF, IPIVOT, CB, LDB, CX, LDX, SRCOND, SFERR, SBERR, CWORK, LDWORK, SWORK2, INFO)213

void chesvx (char fact, char uplo, int n, int nrhs, complex *ca, int lda, complex *caf, int ldaf, int *ipivot, complex *cb, int ldb, complex *cx, int ldx, float *srcond, float *sferr, float *sberr, int *info)213

CALL CHETRF (UPLO, N, CA, LDA, IPIVOT, CWORK, LDWORK, INFO)218

void chetrf (char uplo, int n, complex *ca, int lda, int *ipivot, int *info)218

CALL CHETRI (UPLO, N, CA, LDA, IPIVOT, CWORK, INFO)222

void chetri (char uplo, int n, complex *ca, int lda, int *ipivot, int *info)222

CALL CHETRS (UPLO, N, NRHS, CA, LDA, IPIVOT, CB, LDB, INFO)225

void chetrs (char uplo, int n, int nrhs, complex *ca, int lda, int *ipivot, complex *cb, int ldb, int *info)225

CALL CHICO (CA, LDA, N, IPIVOT, SRCOND, CWORK)578

void chico (complex *ca, int lda, int n, int *ipivot, float *srcond)578

CALL CHIDI (CA, LDA, N, IPIVOT, SDET, INERT, CWORK, JOB)581

void chidi (complex *ca, int lda, int n, int *ipivot, float *sdet, int *inert, int job)581

CALL CHIFA (CA, LDA, N, IPIVOT, INFO)584

void chifa (complex *ca, int lda, int n, int *ipivot, int *info)
584
CALL CHISL (CA, LDA, N, IPIVOT, CB)587
void chisl (complex *ca, int lda, int n, int *ipivot, complex
*cb)587
CALL CHPCO (CA, N, IPIVOT, SRCOND, CWORK)590
void chpc (complex *ca, int n, int *ipivot, float *srcond)590
CALL CHPCON (UPLO, N, CA, IPIVOT, SANORM,
SRCOND, CWORK, INFO)229
void chpcon (char uplo, int n, complex *ca, int *ipivot, float
sanorm, float *srcond, int *info)229
CALL CHPDI (CA, N, IPIVOT, SDET, INERT, CWORK, JOB)
593
void chpdi (complex *ca, int n, int *ipivot, float *sdet, int
*inert, int job)593
CALL CHPEV (JOBZ, UPLO, N, CA, SW, CZ, LDZ, CWORK,
SWORK2, INFO)233
void chpev (char jobz, char uplo, int n, complex *ca, float
*sw, complex *cz, int ldz, int *info)233
CALL CHPEVX (JOBZ, RANGE, UPLO, N, CA, SVL, SVU,
IL, IU, SABTOL, NFOUND, SW, CZ, LDZ, CWORK,
SWORK2, IWORK3, IFAIL, INFO)236
void chpevx (char jobz, char range, char uplo, int n, complex
*ca, float svl, float svu, int il, int iu, float sabetol, int
*nfound, float *sw, complex *cz, int ldz, int *ifail, int
*info)236
CALL CHPFA (CA, N, IPIVOT, INFO)596
void chpfa (complex *ca, int n, int *ipivot, int *info)596
CALL CHPGV (ITYPE, JOBZ, UPLO, N, CA, CB, SW, CZ,
LDZ, CWORK, SWORK2, INFO)240
void chpgv (int itype, char jobz, char uplo, int n, complex
*ca, complex *cb, float *sw, complex *cz, int ldz, int
*info)240
CALL CHPMV (UPLO, N, CALPHA, CA, CX, INCX,
CBETA, CY, INCY)748
void chpmv (char uplo, int n, complex *calpha, complex *ca,
complex *cx, int incx, complex *cbeta, complex *cy,
int incy)748
CALL CHPR (UPLO, N, SALPHA, CX, INCX, CA)751
void chpr (char uplo, int n, float salpha, complex *cx, int
incx, complex *ca)751
CALL CHPR2 (UPLO, N, CALPHA, CX, INCX, CY, INCY,
CA)754
void chpr2 (char uplo, int n, complex *calpha, complex *x,
int incx, complex *cy, int incy, complex *ca)754
CALL CHPRFS (UPLO, N, NRHS, CA, CAF, IPIVOT, CB,
LDB, CX, LDX, SFERR, SBERR, CWORK, SWORK2,
INFO)244
void chprfs (char uplo, int n, int nrhs, complex *ca, complex
*caf, int *ipivot, complex *cb, int ldb, complex *cx,
int ldx, float *sferr, float *sberr, int *info)244
CALL CHPSL (CA, N, IPIVOT, CB)598
void chpsl (complex *ca, int n, int *ipivot, complex *cb)598
CALL CHPSV (UPLO, N, NRHS, CA, IPIVOT, CB, LDB,
INFO)250
void chpsv (char uplo, int n, int nrhs, complex *ca, int
*ipivot, complex *cb, int ldb, int *info)250
CALL CHPSVX (FACT, UPLO, N, NRHS, CA, CAF, IPIVOT,
CB, LDB, CX, LDX, SRCOND, SFERR, SBERR,
CWORK, SWORK2, INFO)253
void chpsvx (char fact, char uplo, int n, int nrhs, complex *ca,
complex *caf, int *ipivot, complex *cb, int ldb,
complex *cx, int ldx, float *srcond, float *sferr, float
*sberr, int *info)253
CALL CHPTRF (UPLO, N, CA, IPIVOT, INFO)258
void chptrf (char uplo, int n, complex *ca, int *ipivot, int
*info)258
CALL CHPTRI (UPLO, N, CA, IPIVOT, CWORK, INFO)262
void chptri (char uplo, int n, complex *ca, int *ipivot, int
*info)262
CALL CHPTRS (UPLO, N, NRHS, CA, IPIVOT, CB, LDB,
INFO)265
void chptrs (char uplo, int n, int nrhs, complex *ca, int
*ipivot, complex *cb, int ldb, int *info)265
CALL COSQB (N, RX, RWSAVE)844
void cosqb (int n, float *rx, float *rwsave)844
CALL COSQF (N, RX, RWSAVE)847
void cosqf (int n, float *rx, float *rwsave)847
CALL COSQI (N, RWSAVE)850
void cosqi (int n, float *rwsave)850
CALL COST (N, RX, RWSAVE)852
void cost (int n, float *rx, float *rwsave)852
CALL COSTI (N, RWSAVE)855
void costi (int n, float *rwsave)855

CALL CPBCO (CA, LDA, N, NDIAG, SRCOND, CWORK, INFO)601

void cpbco (complex *ca, int lda, int n, int ndiag, float *srcond, int *info)601

CALL CPBCON (UPLO, N, NDIAG, CA, LDA, SANORM, SRCOND, CWORK, SWORK2, INFO)281

void cpbcon (char uplo, int n, int ndiag, complex *ca, int lda, float sanorm, float *srcond, int *info)281

CALL CPBDI (CA, LDA, N, NDIAG, SDET)604

void cpbdi (complex *ca, int lda, int n, int ndiag, float *sdet)604

CALL CPBEQU (UPLO, N, NDIAG, CA, LDA, SSCALE, SSCOND, SAMAX, INFO)284

void cpbequ (char uplo, int n, int ndiag, complex *ca, int lda, float *sscale, float *sscond, float *samax, int *info)284

CALL CPBFA (CA, LDA, N, NDIAG, INFO)607

void cpbfa (complex *ca, int lda, int n, int ndiag, int *info)607

CALL CPBRFS (UPLO, N, NDIAG, NRHS, CA, LDA, CAF, LDAF, CB, LDB, CX, LDX, SFERR, SBERR, CWORK, SWORK2, INFO)288

void cpbrfs (char uplo, int n, int ndiag, int nrhs, complex *ca, int lda, complex *caf, int ldaf, complex *cb, int ldb, complex *cx, int ldx, float *sferr, float *sberr, int *info)288

CALL CPBSL (CA, LDA, N, NDIAG, CB)610

void cpbsl (complex *ca, int lda, int n, int ndiag, complex *cb)610

CALL CPBSV (UPLO, N, NDIAG, NRHS, CA, LDA, CB, LDB, INFO)294

void cpbsv (char uplo, int n, int ndiag, int nrhs, complex *ca, int lda, complex *cb, int ldb, int *info)294

CALL CPBSVX (FACT, UPLO, N, NDIAG, NRHS, CA, LDA, CAF, LDAF, EQUED, SSCALE, CB, LDB, CX, LDX, SRCOND, SFERR, SBERR, CWORK, SWORK2, INFO)297

void cpbsvx (char fact, char uplo, int n, int ndiag, int nrhs, complex *ca, int lda, complex *caf, int ldaf, char *equed, float *sscale, complex *cb, int ldb, complex *cx, int ldx, float *srcond, float *sferr, float *sberr, int *info)298

CALL CPBTRF (UPLO, N, NDIAG, CA, LDA, INFO)303

void cpbtrf (char uplo, int n, int ndiag, complex *ca, int lda, int *info)303

CALL CPBTRS (UPLO, N, NDIAG, NRHS, CA, LDA, CB, LDB, INFO)307

void cpbtrs (char uplo, int n, int ndiag, int nrhs, complex *ca, int lda, complex *cb, int ldb, int *info)307

CALL CPOCO (CA, LDA, N, SRCOND, CWORK, INFO)613

void cpoco (complex *ca, int lda, int n, float *srcond, int *info)613

CALL CPOCON (UPLO, N, CA, LDA, SANORM, SRCOND, CWORK, SWORK2, INFO)311

void cpocon (char uplo, int n, complex *ca, int lda, float sanorm, float *srcond, int *info)311

CALL CPODI (CA, LDA, N, SDET, JOB)616

void cpodi (complex *ca, int lda, int n, float *sdet, int job)616

CALL CPOEQU (N, CA, LDA, SSCALE, SSCOND, SAMAX, INFO)314

void cpoequ (int n, complex *ca, int lda, float *sscale, float *sscond, float *samax, int *info)314

CALL CPOFA (CA, LDA, N, INFO)619

void cpofa (complex *ca, int lda, int n, int *info)619

CALL CPORFS (UPLO, N, NRHS, CA, LDA, CAF, LDAF, CB, LDB, CX, LDX, SFERR, SBERR, CWORK, SWORK2, INFO)318

void cporfs (char uplo, int n, int nrhs, complex *ca, int lda, complex *caf, int ldaf, complex *cb, int ldb, complex *cx, int ldx, float *sferr, float *sberr, int *info)318

CALL CPOSL (CA, LDA, N, CB)622

void cposl (complex *ca, int lda, int n, complex *cb)622

CALL CPOSV (UPLO, N, NRHS, CA, LDA, CB, LDB, INFO)323

void cposv (char uplo, int n, int nrhs, complex *ca, int lda, complex *cb, int ldb, int *info)323

CALL CPOSVX (FACT, UPLO, N, NRHS, CA, LDA, CAF, LDAF, EQUED, SSCALE, CB, LDB, CX, LDX, SRCOND, SFERR, SBERR, CWORK, SWORK2, INFO)326

void cposvx (char fact, char uplo, int n, int nrhs, complex *ca, int lda, complex *caf, int ldaf, char *equed, float *sscale, complex *cb, int ldb, complex *cx, int ldx, float *srcond, float *sferr, float *sberr, int *info)327

CALL CPOTRF (UPLO, N, CA, LDA, INFO)332

void cpotrf (char uplo, int n, complex *ca, int lda, int *info)
332
CALL CPOTRI (UPLO, N, CA, LDA, INFO)335
void cpotri (char uplo, int n, complex *ca, int lda, int *info)
335
CALL CPOTRS (UPLO, N, NRHS, CA, LDA, CB, LDB,
INFO)338
void cpotrs (char uplo, int n, int nrhs, complex *ca, int lda,
complex *cb, int ldb, int *info)338
CALL CPPCO (CA, N, SRCOND, CWORK, INFO)625
void cppco (complex *ca, int n, float *srcond, int *info)625
CALL CPPCON (UPLO, N, CA, SANORM, SRCOND,
CWORK, SWORK2, INFO)342
void cppcon (char uplo, int n, complex *ca, float sanorm,
float *srcond, int *info)342
CALL CPPDI (CA, N, SDET, JOB)628
void cppdi (complex *ca, int n, float *sdet, int job)628
CALL CPPEQU (UPLO, N, CA, SSCALE, SSCOND,
SAMAX, INFO)345
void cppequ (char uplo, int n, complex *ca, float *sscale, float
*scond, float *samax, int *info)345
CALL CPPFA (CA, N, INFO)631
void cppfa (complex *ca, int n, int *info)631
CALL CPPRFS (UPLO, N, NRHS, CA, CAF, CB, LDB, CX,
LDX, SFERR, SBERR, CWORK, SWORK2, INFO)
349
void cpprfs (char uplo, int n, int nrhs, complex *ca, complex
*caf, complex *cb, int ldb, complex *cx, int ldx, float
*sferr, float *sberr, int *info)349
CALL CPPSL (CA, N, CB)634
void cppsl (complex *ca, int n, complex *cb)634
CALL CPPSV (UPLO, N, NRHS, CA, CB, LDB, INFO)354
void cppsv (char uplo, int n, int nrhs, complex *ca, complex
*cb, int ldb, int *info)354
CALL CPPSVX (FACT, UPLO, N, NRHS, CA, CAF, EQUED,
SSCALE, CB, LDB, CX, LDX, SRCOND, SFERR,
SBERR, CWORK, SWORK2, INFO)357
void cppsvx (char fact, char uplo, int n, int nrhs, complex *ca,
complex *caf, char *equed, float *sscale, complex
*cb, int ldb, complex *cx, int ldx, float *srcond, float
*sferr, float *sberr, int *info)357
CALL CPPTRF (UPLO, N, CA, INFO)363
void cptrf (char uplo, int n, complex *ca, int *info)363
CALL CPPTRI (UPLO, N, CA, INFO)366
void cpptri (char uplo, int n, complex *ca, int *info)366
CALL CPPTRS (UPLO, N, NRHS, CA, CB, LDB, INFO)369
void cpptrs (char uplo, int n, int nrhs, complex *ca, complex
*cb, int ldb, int *info)369
CALL CPTCON (N, SDIAG, COFFD, SANORM, SRCOND,
SWORK, INFO)372
void cptcon (int n, float *sdiag, complex *zoffd, float sanorm,
float *srcond, int *info)372
CALL CPTRFS (UPLO, N, NRHS, SDIAG, COFFD, SDIAGF,
COFFDF, CB, LDB, CX, LDX, SFERR, SBERR,
CWORK, SWORK2, INFO)375
void cptrfs (char uplo, int n, int nrhs, float *sdiag, complex
*coffd, float *sdiagf, complex *coffdf, complex *cb,
int ldb, complex *cx, int ldx, float *sferr, float *sberr,
int *info)375
CALL CPTSL (N, CDIAG, COFFD, CB)637
void cptsl (int n, complex *cdiag, complex *coffd, complex
*cb)637
CALL CPTSV (N, NRHS, SDIAG, CSUB, CB, LDB, INFO)381
void cptsv (int n, int nrhs, float *sdiag, complex *ssub,
complex *cb, int ldb, int *info)381
CALL CPTSVX (FACT, N, NRHS, SDIAG, CSUB, SDIAGF,
CSUBF, CB, LDB, CX, LDX, SRCOND, SFERR,
SBERR, CWORK, SWORK2, INFO)384
void cptsvx (char fact, int n, int nrhs, float *sdiag, complex
*csub, float *sdiagf, complex *csubf, complex *cb,
int ldb, complex *cx, int ldx, float *srcond, float
*sferr, float *sberr, int *info)384
CALL CPTTRF (N, SDIAG, COFFD, INFO)389
void cpttrf (int n, float *sdiag, complex *coffd, int *info)389
CALL CPTTRS (N, NRHS, SDIAG, COFFD, CB, LDB, INFO)
392
void cpttrs (char uplo, int n, int nrhs, float *sdiag, complex
*coffd, complex *cb, int ldb, int *info)392
CALL CQRDC (CA, LDA, N, P, CQRAUX, IPIVOT, CWORK,
JOB)639
void qrdc (complex *ca, int lda, int n, int p, complex
*cquaux, int *ipivot, int job)639
CALL CQRSL (CA, LDA, N, K, CQRAUX, ZY, ZQY, CQTY,
CB, CRESID, CAB, JOB, INFO)642

```

void cqrsl (complex *ca, int lda, int n, int k, complex *cqraux,
           complex *cy, complex *cqy, complex *cqty, complex
           *cb, complex *cresid, complex *cab, int job, int
           *info)642
CALL CSCAL (N, CALPHA, CY, INCY)714
void cscal(int n, complex *ca, complex *cx, int incx)714
CALL CSICO (CA, LDA, N, IPIVOT, SRCOND, CWORK)646
void csico (complex *ca, int lda, int n, int *ipivot, float
           *srcond)646
CALL CSIDI (CA, LDA, N, IPIVOT, CDET, CWORK, JOB)
           649
void csidi (complex *ca, int lda, int n, int *ipivot, complex
           *cdet, int job)649
CALL CSIFA (CA, LDA, N, IPIVOT, INFO)652
void csifa (complex *ca, int lda, int n, int *ipivot, int *info)652
CALL CSISL (CA, LDA, N, IPIVOT, CB)655
void csisl (complex *ca, int lda, int n, int *ipivot, complex
           *cb)655
CALL CSPCO (CA, N, IPIVOT, SRCOND, CWORK)658
void cspco (complex *ca, int n, int *ipivot, float *srcond)658
CALL CSPCON (UPLO, N, CA, IPIVOT, SANORM,
            SRCOND, CWORK, INFO)405
void cspcon (char uplo, int n, complex *ca, int *ipivot, float
            sanorm, float *srcond, int *info)405
CALL CSPDI (CA, N, IPIVOT, CDET, CWORK, JOB)661
void cspdi (complex *ca, int n, int *ipivot, complex *cdet, int
            job)661
CALL CSPFA (CA, N, IPIVOT, INFO)664
void cspfa (complex *ca, int n, int *ipivot, int *info)664
CALL CSPRFS (UPLO, N, NRHS, CA, CAF, IPIVOT, CB,
            LDB, CX, LDX, SFERR, SBERR, CWORK, SWORK2,
            INFO)419
void csprfs (char uplo, int n, int nrhs, complex *ca, complex
            *caf, int *ipivot, complex *cb, int ldb, complex *cx,
            int ldx, float *sferr, float *sberr, int *info)419
CALL CSPSL (CA, N, IPIVOT, CB)667
void cspsl (complex *ca, int n, int *ipivot, complex *cb)667
CALL CSPSV (UPLO, N, NRHS, CA, IPIVOT, CB, LDB,
            INFO)425
void cspsv (char uplo, int n, int nrhs, complex *ca, int *ipivot,
            complex *cb, int ldb, int *info)425

```

```

CALL CSPSVX (FACT, UPLO, N, NRHS, CA, CAF, IPIVOT,
            CB, LDB, CX, LDX, SRCOND, SFERR, SBERR,
            CWORK, SWORK2, INFO)428
void cspsvx (char fact, char uplo, int n, int nrhs, complex *ca,
            complex *caf, int *ipivot, complex *cb, int ldb,
            complex *cx, int ldx, float *srcond, float *sferr, float
            *sberr, int *info)428
CALL CSPTRF (UPLO, N, CA, IPIVOT, INFO)433
void csptf (char uplo, int n, complex *ca, int *ipivot, int
            *info)433
CALL CSPTRI (UPLO, N, CA, IPIVOT, CWORK, INFO)436
void csptri (char uplo, int n, complex *ca, int *ipivot, int
            *info)436
CALL CSPTRS (UPLO, N, NRHS, CA, IPIVOT, CB, LDB,
            INFO)439
void csptrs (char uplo, int n, int nrhs, complex *ca, int
            *ipivot, complex *cb, int ldb, int *info)439
CALL CSSCAL (N, SALPHA, CY, INCY)714
void csscal (int n, float sa, complex *cx, int incx)714
CALL CSVDC (CA, LDA, N, P, CSVALS, CE, CLSVEC, LDL,
            CRSVEC, LDR, CWORK, JOB, INFO)669
void csvdc (complex *ca, int lda, int n, int p, complex *csvals,
            complex *ce, complex *clsvec, int ldl, complex
            *crsvec, int ldr, int job, int *info)669
CALL CSWAP (N, CX, INCX, CY, INCY)716
void cswap (int n, complex *cx, int incx, complex *cy, int
            incy)716
CALL CSYCON (UPLO, N, CA, LDA, IPIVOT, SANORM,
            SRCOND, CWORK, INFO)450
void csycon (char uplo, int n, complex *ca, int lda, int *ipivot,
            float sanorm, float *srcond, int *info)450
CALL CSYMM (SIDE, UPLO, M, N, CALPHA, CA, LDA,
            CB, LDB, CBETA, CC, LDC)819
void csymm (char side, char uplo, int m, int n, complex
            *calpha, complex *ca, int lda, complex *cb, int ldb,
            complex *cbeta, complex *cc, int ldc)819
CALL CSYR2K (UPLO, TRANSA, N, K, CALPHA, CA,
            LDA, CB, LDB, CBETA, CC, LDC)826
void csyr2k (char uplo, char trans, int n, int k, complex
            *calpha, complex *ca, int lda, complex *cb, int ldb,
            complex *cbeta, complex *cc, int ldc)826

```


CALL CSYRFS (UPLO, N, NRHS, CA, LDA, CAF, LDAF, IPIVOT, CB, LDB, CX, LDX, SFERR, SBERR, CWORK, SWORK2, INFO)465

void csyrf (char uplo, int n, int nrhs, complex *ca, int lda, complex *caf, int ldaf, int *ipivot, complex *cb, int ldb, complex *cx, int ldx, float *sferr, float *sberr, int *info)465

CALL CSYRK (UPLO, TRANSA, N, K, CALPHA, CA, LDA, CBETA, CC, LDC)823

void csyrk (char uplo, char trans, int n, int k, complex *calpha, complex *ca, int lda, complex *cbeta, complex *cc, int ldc)823

void csysv (char uplo, int n, int nrhs, complex *ca, int lda, int *ipivot, complex *cb, int ldb, int *info)471

CALL CSYSVX (FACT, UPLO, N, NRHS, CA, LDA, CAF, LDAF, IPIVOT, CB, LDB, CX, LDX, SRCOND, SFERR, SBERR, CWORK, LDWORK, SWORK2, INFO)474

void csysvx (char fact, char uplo, int n, int nrhs, complex *ca, int lda, complex *caf, int ldaf, int *ipivot, complex *cb, int ldb, complex *cx, int ldx, float *srcond, float *sferr, float *sberr, int *info)474

CALL CSYTRF (UPLO, N, CA, LDA, IPIVOT, CWORK, LDWORK, INFO)479

void csytrf (char uplo, int n, complex *ca, int lda, int *ipivot, int *info)479

CALL CSYTRI (UPLO, N, CA, LDA, IPIVOT, CWORK, INFO)482

void csytri (char uplo, int n, complex *ca, int lda, int *ipivot, int *info)482

CALL CSYTRS (UPLO, N, NRHS, CA, LDA, IPIVOT, CB, LDB, INFO)485

void csytrs (char uplo, int n, int nrhs, complex *ca, int lda, int *ipivot, complex *cb, int ldb, int *info)485

CALL CTBCON (NORM, UPLO, DIAG, N, NDIAG, CA, LDA, SRCOND, CWORK, SWORK2, INFO)488

void ctbcon (char norm, char uplo, char diag, int n, int ndiag, complex *ca, int lda, float *srcond, int *info)488

CALL CTBMV (UPLO, TRANSA, DIAG, N, NDIAG, CA, LDA, CY, INCY)779

void ctbm (char uplo, char trans, char diag, int n, int ndiag, complex *ca, int lda, complex *cx, int incx)779

CALL CTBRFS (UPLO, TRANSA, DIAG, N, NDIAG, NRHS, CA, LDA, CB, LDB, CX, LDX, SFERR, SBERR, CWORK, SWORK2, INFO)492

void ctbrfs (char uplo, char trans, char diag, int n, int ndiag, int nrhs, complex *ca, int lda, complex *cb, int ldb, complex *cx, int ldx, float *sferr, float *sberr, int *info)492

CALL CTBSV (UPLO, TRANSA, DIAG, N, NDIAG, CA, LDA, CY, INCY)783

void ctbsv (char uplo, char trans, char diag, int n, int ndiag, complex *ca, int lda, complex *cx, int incx)783

CALL CTBTRS (UPLO, TRANSA, DIAG, N, NDIAG, NRHS, CA, LDA, CB, LDB, INFO)497

void ctbtrs (char uplo, char trans, char diag, int n, int ndiag, int nrhs, complex *ca, int lda, complex *cb, int ldb, int *info)497

CALL CTPCON (NORM, UPLO, DIAG, N, CA, SRCOND, CWORK, SWORK2, INFO)501

void ctpcon (char norm, char uplo, char diag, int n, complex *ca, float *srcond, int *info)501

CALL CTPMV (UPLO, TRANSA, DIAG, N, CA, CY, INCY)787

void ctpmv (char uplo, char trans, char diag, int n, complex *ca, complex *cx, int incx)787

CALL CTPRFS (UPLO, TRANSA, DIAG, N, NRHS, CA, CB, LDB, CX, LDX, SFERR, SBERR, CWORK, SWORK2, INFO)504

void ctpfrs (char uplo, char trans, char diag, int n, int nrhs, complex *ca, complex *cb, int ldb, complex *cx, int ldx, float *sferr, float *sberr, int *info)504

CALL CTPSV (UPLO, TRANSA, DIAG, N, CA, CY, INCY)791

void ctpsv (char uplo, char trans, char diag, int n, complex *ca, complex *cx, int incx)791

CALL CTPTRI (UPLO, DIAG, N, CA, INFO)509

void ctptri (char uplo, char diag, int n, complex *ca, int *info)509

CALL CTPTRS (UPLO, TRANSA, DIAG, N, NRHS, CA, CB, LDB, INFO)512

void ctptrs (char uplo, char trans, char diag, int n, int nrhs, complex *ca, complex *cb, int ldb, int *info)512

CALL CTRANS (PLACE, CSCALE, CSOURCE, M, N, CDEST)842

void ctrans (char place, complex cscale, complex *csource,
int m, int n, complex *cdest)842

CALL CTRCO (CA, LDA, N, SRCOND, CWORK, JOB)672

void ctrco (complex *ca, int lda, int n, float *srcond, int job)
672

CALL CTRCON (NORM, UPLO, DIAG, N, CA, LDA,
SRCOND, CWORK, SWORK2, INFO)515

void ctrcon (char norm, char uplo, char diag, int n, complex
*ca, int lda, float *srcond, int *info)515

CALL CTRDI (CA, LDA, N, CDET, JOB, INFO)675

void ctrdi (complex *ca, int lda, int n, complex *cdet, int job,
int *info)675

CALL CTRMM (SIDE, UPLO, TRANSA, DIAG, M, N,
CALPHA, CA, LDA, CB, LDB)830

void ctrmm (char side, char uplo, char transa, char diag, int
m, int n, complex *calpha, complex *ca, int lda,
complex *cb, int ldb)830

CALL CTRMV (UPLO, TRANSA, DIAG, N, CA, LDA, CY,
INCY)795

void ctrmv (char uplo, char trans, char diag, int n, complex
*ca, int lda, complex *cx, int incx)795

CALL CTRRFS (UPLO, TRANSA, DIAG, N, NRHS, CA,
LDA, CB, LDB, CX, LDX, SFERR, SBERR, CWORK,
SWORK2, INFO)518

void ctrrfs (char uplo, char trans, char diag, int n, int nrhs,
complex *ca, int lda, complex *cb, int ldb, complex
*cx, int ldx, float *sferr, float *sberr, int *info)518

CALL CTRSL (CA, LDA, N, CB, JOB, INFO)678

void ctrsl (complex *ca, int lda, int n, complex *cb, int job, int
*info)678

CALL CTRSM (SIDE, UPLO, TRANSA, DIAG, M, N,
CALPHA, CA, LDA, CB, LDB)834

void ctrsm (char side, char uplo, char transa, char diag, int
m, int n, complex *calpha, complex *ca, int lda,
complex *cb, int ldb)834

CALL CTRSV (UPLO, TRANSA, DIAG, N, CA, LDA, CY,
INCY)798

void ctrsv (char uplo, char trans, char diag, int n, complex
*ca, int lda, complex *cx, int incx)798

CALL CTRTRI (UPLO, DIAG, N, CA, LDA, INFO)523

void ctrtri (char uplo, char diag, int n, complex *ca, int lda,
int *info)523

CALL CTRTRS (UPLO, TRANSA, DIAG, N, NRHS, CA,
LDA, CB, LDB, INFO)526

void ctrtrs (char uplo, char trans, char diag, int n, int nrhs,
complex *ca, int lda, complex *cb, int ldb, int *info)
526

D

DL1 = DASUM (N, DX, INCX)684

double dasum (int n, double *dx, int incx)684

CALL DAXPY (N, DALPHA, DX, INCX, DY, INCY)687

void daxpy (int n, double da, double *dx, int incx, double
*dy, int incy)687

CALL DCHDC (DA, LDA, N, DWORK, IPIVOT, JOB, INFO)
535

void dchdc (double *da, int lda, int n, int *ipivot, int job, int
*info)535

CALL DCHDD (DA, LDA, N, DX, DZ, LDZ, NZ, DY, DRHO,
DCOS, DSIN, INFO)538

void dchdd (double *da, int lda, int n, double *dx, double
*dz, int ldz, int nz, double *dy, double *drho,
double *dcos, double *dsin, int *info)538

CALL DCHEX (DA, LDA, N, K, L, DZ, LDZ, NZ, DCOS,
DSIN, JOB)542

void dchex (double *da, int lda, int n, int k, int l, double *dz,
int ldz, int nz, double *dcos, double *dsin, int job)
542

CALL DCHUD (DA, LDA, N, DX, DZ, LDZ, NZ, DY, DRHO,
DCOS, DSIN)546

void dchud (double *da, int lda, int n, double *dx, double
*dz, int ldz, int nz, double *dy, double *drho,
double *dcos, double *dsin)546

CALL DCOPY (N, DX, INCX, DY, INCY)690

void dcopy (int n, double *dx, int incx, double *dy, int incy)
690

CALL DCOSQB (N, DX, DWSAVE)844

void dcosqb (int n, double *dx, double *dwsave)844

CALL DCOSQF (N, DX, RWSAVE)847

void dcosqf (int n, double *dx, double *dwsave)847

CALL DCOSQI (N, DWSAVE)850

void dcosqi (int n, double *dwsave)850

CALL DCOST (N, DX, DWSAVE)852

void dcost (int n, double *dx, double *dwsave)852


```
CALL DCOSTI (N, DWSAVE)855
void dcosti (int n, double *dwsave)855
DINNER = DDOT (N, DX, INCX, DY, INCY)694
double ddot(int n, double *dx, int incx, double *dy, int incy)
694
CALL DFFTB (N, DX, DWSAVE)865
void dfftb (int n, double *rx, double *dwsave)866
CALL DFFTF (N, DX, DWSAVE)868
void dfftf (int n, double *dx, double *dwsave)869
CALL DFFTI (N, DWSAVE)871
void dffti (int n, double *dwsave)871
CALL DGBCO (DA, LDA, N, NSUB, NSUPER, IPIVOT,
DRCOND, DWORK)549
void dgbco (double *da, int lda, int n, int nsub, int nsuper, int
*ipivot, double *drcond)549
CALL DGBCON (NORM, N, NSUB, NSUPER, DA, LDA,
IPIVOT, DANORM, DRCOND, DWORK, IWORK2,
INFO)8
void dgbcon (char norm, int n, int nsub, int nsuper, double
*da, int lda, int *ipivot, double danorm, double
*drcond, int *info)8
CALL DGBDI (DA, LDA, N, NSUB, NSUPER, IPIVOT,
DDET)553
void dgbdi (double *da, int lda, int n, int nsub, int nsuper, int
*ipivot, double *ddet)553
CALL DGBEQU (M, N, NSUB, NSUPER, DA, LDA,
DROWSC, DCOLSC, DROWCN, DCOLCN,
DAMAX, INFO)12
void dgbequ (int m, int n, int nsub, int nsuper, double *da,
int lda, double *drowsc, double *dcolsc, double
*drowcn, double *dcolcn, double *damax, int *info)
12
CALL DGBFA (DA, LDA, N, NSUB, NSUPER, IPIVOT,
INFO)556
void dgbf (double *da, int lda, int n, int nsub, int nsuper, int
*ipivot, int *info)556
CALL DGBMV (TRANSA, M, N, NSUB, NSUPER,
DALPHA, DA, LDA, DX, INCX, DBETA, DY,
INCY)723
void dgbm (char trans, int m, int n, int nsub, int nsuper,
double dalpha, double *da, int lda, double *dx, int
incx, double dbeta, double *dy, int incy)723
CALL DGBRFS (TRANSA, N, NSUB, NSUPER, NRHS, DA,
LDA, DAF, LDAF, IPIVOT, DB, LDB, DX, LDX,
DFERR, DBERR, DWORK, IWORK2, INFO)15
void dgbrfs (char trans, int n, int nsub, int nsuper, int nrhs,
double *da, int lda, double *daf, int ldaf, int *ipivot,
double *db, int ldb, double *dx, int ldx, double
*dferr, double *dberr, int *info)15
CALL DGBSL (DA, LDA, N, NSUB, NSUPER, IPIVOT, DB,
JOB)560
void dgbsl (double *da, int lda, int n, int nsub, int nsuper, int
*ipivot, double *db, int job)560
CALL DGBSV (N, NSUB, NSUPER, NRHS, DA, LDA,
IPIVOT, DB, LDB, INFO)21
void dgbsv (int n, int nsub, int nsuper, int nrhs, double *da,
int lda, int *ipivot, double *db, int ldb, int *info)21
CALL DGBSVX (FACT, TRANSA, N, NSUB, NSUPER,
NRHS, DA, LDA, DAF, LDAF, IPIVOT, EQUED,
DROWSC, DCOLSC, DB, LDB, DX, LDX,
DRCOND, DFERR, DBERR, DWORK, IWORK2,
INFO)25
void dgbsvx (char fact, char trans, int n, int nsub, int nsuper,
int nrhs, double *da, int lda, double *daf, int ldaf,
int *ipivot, char *equed, double *drowsc, double
*dcolsc, double *db, int ldb, double *dx, int ldx,
double *drcond, double *dferr, double *dberr, int
*info)25
CALL DGBTRF (M, N, NSUB, NSUPER, DA, LDA, IPIVOT,
INFO)33
void dgbrf (int m, int n, int nsub, int nsuper, double *da, int
lda, int *ipivot, int *info)33
CALL DGBTRS (TRANSA, N, NSUB, NSUPER, NRHS, DA,
LDA, IPIVOT, DB, LDB, INFO)37
void dgbrs (char trans, int n, int nsub, int nsuper, int nrhs,
double *da, int lda, int *ipivot, double *db, int ldb,
int *info)37
CALL DGECO (DA, LDA, N, IPIVOT, DRCOND, DWORK)
564
void dgeco (double *da, int lda, int n, int *ipivot, double
*drcond)564
CALL DGECON (NORM, N, DA, LDA, DANORM,
DRCOND, DWORK, IWORK2, INFO)41
void dgecon (char norm, int n, double *da, int lda, double
danorm, double *drcond, int *info)41
```

CALL DGEDI (DA, LDA, N, IPIVOT, DDET, DWORK, JOB)
567
void dgedi (double *da, int lda, int n, int *ipivot, double
*ddet, int job)567
CALL DGEEQU (M, N, DA, LDA, DROWSC, DCOLSC,
DROWCN, DCOLCN, DAMAX, INFO)44
void dgeequ (int m, int n, double *da, int lda, double
*drowsc, double *dcolsc, double *drowcn, double
*dcolcn, double *damax, int *info)44
CALL DGEES (JOBZ, SORTEV, SELECT, N, DA, LDA,
NOUT, DWR, DWI, DZ, LDZ, DWORK, LDWORK,
BWORK3, INFO)48
void dgees (char jobvs, char sort, L_fp select, int n, double
*da, int lda, int *sdim, double *dwr, double *dwi,
double *dz, int ldz, int *info)48
CALL DGEESX (JOBZ, SORTEV, SELECT, SENSE, N, DA,
LDA, NOUT, DWR, DWI, DZ, LDZ, DRCONC,
DRCONV, DWORK, LDWORK, IWORK2,
LDWRK2, BWORK3, INFO)53
CALL DGEESX (JOBZ, SORTEV, SELECT, SENSE, N, DA,
LDA, NOUT, DWR, DWI, DZ, LDZ, SRCONE,
DRCONV, DWORK, LDWORK, IWORK2,
LDWRK2, BWORK3, INFO)53
void dgeesx (char jobvs, char sort, L_fp select, char sense, int
n, double *da, int lda, int *sdim, double *dwr,
double *dwi, double *dz, int ldz, double *drcone,
double *drconv, int *info)53
CALL DGEEV (JOBVL, JOBVR, N, DA, LDA, DWR, DWI,
DVL, LDVL, DVR, LDVR, DWORK, LDWORK,
INFO)59
void dgeev (char jobvl, char jobvr, int n, double *da, int lda,
double *dwr, double *dwi, double *dvl, int ldvl,
double *dvr, int ldvr, int *info)59
CALL DGEEVX (BALANC, JOBVL, JOBVR, SENSE, N, DA,
LDA, DWR, DWI, DVL, LDVL, DVR, LDVR, ILO,
IHI, DSCALE, DABNRM, DRCONC, DRCONV,
DWORK, LDWORK, IWORK2, INFO)63
void dgeevx (char balanc, char jobvl, char jobvr, char sense,
int n, double *da, int lda, double *dwr, double *dwi,
double *dvl, int ldvl, double *dvr, int ldvr, int *ilo,
int *ihi, double *dscale, double *dabnrm, double
*drcone, double *drconv, int *info)63
CALL DGEFA (DA, LDA, N, IPIVOT, INFO)570
void dgefa (double *da, int lda, int n, int *ipivot, int *info)570

CALL DGEGB (JOBVSL, JOBVSR, N, DA, LDA, DB, LDB,
DALPHAR, DALPHAI, DBETA, DVSL, LDVSL,
DVSR, LDVSR, DWORK, LDWORK, INFO)69
void dgegb (char jobvsl, char jobvsr, int n, double *da, int lda,
double *db, int ldb, double *dalphar, double
*dalphai, double *dbeta, double *dvsl, int ldvsl,
double *dvsl, int ldvsl, int *info)69
CALL DGEGV (JOBVL, JOBVR, N, DA, LDA, DB, LDB,
DALPHAR, DALPHAI, DBETA, DVL, LDVL, DVR,
LDVR, DWORK, LDWORK, INFO)74
void dgegv (char jobvl, char jobvr, int n, double *da, int lda,
double *db, int ldb, double *dalphar, double
*dalphai, double *dbeta, double *dvl, int ldvl,
double *dvr, int ldvr, int *info)74
CALL DGELQF (M, N, DA, LDA, DTAU, DWORK,
LDWORK, INFO)79
void dgelqf (int m, int n, double *da, int lda, double *dtau,
int *info)79
CALL DGELS (TRANSA, M, N, NRHS, DA, LDA, DB, LDB,
DWORK, LDWORK, INFO)82
void dgels (char trans, int m, int n, int nrhs, double *da, int
lda, double *db, int ldb, int *info)82
CALL DGELSS (M, N, NRHS, DA, LDA, DB, LDB, DSING,
DRCOND, IRANK, DWORK, LDWORK, INFO)86
void dgelss (int m, int n, int nrhs, double *da, int lda, double
*db, int ldb, double *dsing, double rcond, int *rank,
int *info)86
CALL DGELSX (M, N, NRHS, DA, LDA, DB, LDB, JPIVOT,
DRCOND, IRANK, DWORK, INFO)90
void dgelsx (int m, int n, int nrhs, double *da, int lda, double
*db, int ldb, int *jpivot, double rcond, int *rank, int
*info)90
CALL DGEMM (TRANSA, TRANSB, M, N, K, DALPHA,
DA, LDA, DB, LDB, DBETA, DC, LDC)805
void dgemm (char transa, char transb, int m, int n, int k,
double dalpha, double *da, int lda, double *db, int
ldb, double dbeta, double *dc, int ldc)805
CALL DGEMV (TRANSA, M, N, DALPHA, DA, LDA, DX,
INCX, DBETA, DY, INCY)727
void dgemv (char trans, int m, int n, double dalpha, double
*da, int lda, double *dx, int incx, double dbeta,
double *dy, int incy)727

CALL DGEQLF (M, N, DA, LDA, DTAU, DWORK,
LDWORK, INFO)94
void dgeqlf (int m, int n, double *da, int lda, double *dtau,
int *info)94
CALL DGEQPF (M, N, DA, LDA, JPIVOT, DTAU, DWORK,
INFO)97
void dgeqpf (int m, int n, double *da, int lda, int *jpivot,
double *dtau, int *info)97
CALL DGEQRF (M, N, DA, LDA, DTAU, DWORK,
LDWORK, INFO)100
void dgeqrf (int m, int n, double *da, int lda, double *dtau,
int *info)100
CALL DGER (M, N, DALPHA, DX, INCX, DY, INCY, DA,
LDA)730
void dger (int m, int n, double dalpha, double *dx, int incx,
double *dy, int incy, double *da, int lda)730
CALL DGERFS (TRANSA, N, NRHS, DA, LDA, DAF, LDAF,
IPIVOT, DB, LDB, DX, LDX, DFERR, DBERR,
DWORK, IWORK2, INFO)103
void dgerfs (char trans, int n, int nrhs, double *da, int lda,
double *daf, int ldaf, int *ipivot, double *db, int ldb,
double *dx, int ldx, double *dferr, double *dberr, int
*info)103
CALL DGERQF (M, N, DA, LDA, DTAU, DWORK,
LDWORK, INFO)108
void dgerqf (int m, int n, double *da, int lda, double *dtau,
int *info)108
CALL DGESL (DA, LDA, N, IPIVOT, DB, JOB)572
void dgesl (double *da, int lda, int n, int *ipivot, double *db,
int job)572
CALL DGESV (N, NRHS, DA, LDA, IPIVOT, DB, LDB,
INFO)111
void dgesv (int n, int nrhs, double *da, int lda, int *ipivot,
double *db, int ldb, int *info)111
CALL DGESVD (JOBV, JOBT, M, N, DA, LDA, DSING,
DU, LDU, DVT, LDVT, DWORK, LDWORK, INFO)
114
void dgesvd (char jobv, char jobt, int m, int n, double *da,
int lda, double *dsing, double *du, int ldu, double
*dvt, int ldvt, int *info)114
CALL DGESVX (FACT, TRANSA, N, NRHS, DA, LDA, DAF,
LDAF, IPIVOT, EQUED, DROWSC, DCOLSC, DB,
LDB, DX, LDX, DRCOND, DFERR, DBERR,
DWORK, IWORK2, INFO)119
void dgesvx (char fact, char trans, int n, int nrhs, double *da,
int lda, double *daf, int ldaf, int *ipivot, char
*equed, double *drowsc, double *dcolsc, double
*db, int ldb, double *dx, int ldx, double *drcond,
double *dferr, double *dberr, int *info)119
CALL DGETRF (M, N, DA, LDA, IPIVOT, INFO)126
void dgetrf (int m, int n, double *da, int lda, int *ipivot, int
*info)126
CALL DGETRI (N, DA, LDA, IPIVOT, DWORK, LDWORK,
INFO)129
void dgetri (int n, double *da, int lda, int *ipivot, int *info)
129
CALL DGETRS (TRANSA, N, NRHS, DA, LDA, IPIVOT,
DB, LDB, INFO)132
void dgetrs (char trans, int n, int nrhs, double *da, int lda, int
*ipivot, double *db, int ldb, int *info)132
CALL DGGGLM (N, M, P, DA, LDA, DB, LDB, DD, DX, DY,
DWORK, LDWORK, INFO)135
void dggglm (int n, int m, int p, double *da, int lda, double
*db, int ldb, double *dd, double *dx, double *dy, int
*info)135
CALL DGGLSE (M, N, P, DA, LDA, DB, LDB, DC, DD, DX,
DWORK, LDWORK, INFO)139
void dgglse (int m, int n, int p, double *da, int lda, double
*db, int ldb, double *dc, double *dd, double *dx, int
*info)139
CALL DGGSDV (JOBV, JOBT, M, N, P, K, L, DA,
LDA, DB, LDB, DALPHA, DBETA, DU, LDU, DV,
LDV, DQ, LDQ, DWORK, IWORK3, INFO)144
void dggsvd (char jobv, char jobt, char jobq, int m, int n, int
p, int *k, int *l, double *da, int lda, double *db, int
ldb, double *dalp, double *dbeta, double *du, int
ldu, double *dv, int ldv, double *dq, int ldq, int
*info)144
CALL DGTCON (NORM, N, DLOW, DDIAG, DUP1, DUP2,
IPIVOT, DANORM, DRCOND, DWORK, IWORK2,
INFO)149

```

void dgtcon (char norm, int n, double *dlow, double *ddiag,
             double *dup1, double *dup2, int *ipivot, double
             danorm, double *drcond, int *info)149
CALL DGTRFS (TRANSA, N, NRHS, DLOW, DDIAG, DUP,
             DLOWF, DDIAGF, DUPF1, DUPF2, IPIVOT, DB,
             LDB, DX, LDX, DFERR, DBERR, DWORK,
             IWORK2, INFO)153
void dgtrfs (char trans, int n, int nrhs, double *dlow, double
             *ddiag, double *dup, double *dlowf, double
             *ddiagf, double *dupf1, double *dupf2, int *ipivot,
             double *db, int ldb, double *dx, int ldx, double
             *dferr, double *dberr, int *info)153
CALL DGTSL (N, DSUB, DDIAG, DSUPER, DB, INFO)575
void dgtsl (int n, double *dsup, double *ddiag, double
            *dsuper, double *db, int *info)575
CALL DGTSV (N, NRHS, DLOW, DDIAG, DUP, DB, LDB,
            INFO)160
void dgtsv (int n, int nrhs, double *dlow, double *ddiag,
            double *dup, double *db, int ldb, int *info)160
CALL DGTSVX (FACT, TRANSA, N, NRHS, DLOW,
             DDIAG, DUP, DLOWF, DDIAGF, DUPF1, DUPF2,
             IPIVOT, DB, LDB, DX, LDX, DRCOND, DFERR,
             DBERR, DWORK, IWORK2, INFO)163
void dgtsvx (char fact, char trans, int n, int nrhs, double
             *dlow, double *ddiag, double *dup, double *dlowf,
             double *ddiagf, double *dupf1, double *dupf2, int
             *ipivot, double *db, int ldb, double *dx, int ldx,
             double *drcond, double *dferr, double *dberr, int
             *info)163
CALL DGTTRF (N, DLOW, DDIAG, DUP1, DUP2, IPIVOT,
            INFO)169
void dgtrf (int n, double *dlow, double *ddiag, double
            *dup1, double *dup2, int *ipivot, int *info)169
CALL DGTTRS (TRANSA, N, NRHS, DLOW, DDIAG,
            DUP1, DUP2, IPIVOT, DB, LDB, INFO)173
void dgtrs (char trans, int n, int nrhs, double *dlow, double
            *ddiag, double *dup1, double *dup2, int *ipivot,
            double *db, int ldb, int *info)173
DL2 = DNRM2 (N, DX, INCX)712
double dnrm2(int n, double *dx, int incx)712
CALL DORGLQ (M, N, K, DA, LDA, DTAU, DWORK,
            LDWORK, INFO)269

```

```

void dorglq (int m, int n, int k, double *da, int lda, double
            *dtau, int *info)269
CALL DORGQL (M, N, K, DA, LDA, DTAU, DWORK,
            LDWORK, INFO)272
void dorgql (int m, int n, int k, double *da, int lda, double
            *dtau, int *info)272
CALL DORGQR (M, N, K, DA, LDA, DTAU, DWORK,
            LDWORK, INFO)275
void dorgqr (int m, int n, int k, double *da, int lda, double
            *dtau, int *info)275
CALL DORGRQ (M, N, K, DA, LDA, DTAU, DWORK,
            LDWORK, INFO)278
void dorgrq (int m, int n, int k, double *da, int lda, double
            *dtau, int *info)278
CALL DPBCO (DA, LDA, N, NDIAG, DRCOND, DWORK,
            INFO)601
void dpbco (double *da, int lda, int n, int ndiag, double
            *drcond, int *info)601
CALL DPBCON (UPLO, N, NDIAG, DA, LDA, DANORM,
            DRCOND, DWORK, IWORK2, INFO)281
void dpbcon (char uplo, int n, int ndiag, double *da, int lda,
            double danorm, double *drcond, int *info)281
CALL DPBDI (DA, LDA, N, NDIAG, DDET)604
void dpbdi (double *da, int lda, int n, int ndiag, double
            *ddet)604
CALL DPBEQU (UPLO, N, NDIAG, DA, LDA, DSCALE,
            DSCOND, DAMAX, INFO)284
void dpbequ (char uplo, int n, int ndiag, double *da, int lda,
            double *dscale, double *dscond, double *damax, int
            *info)284
CALL DPBFA (DA, LDA, N, NDIAG, INFO)607
void dpbfa (double *da, int lda, int n, int ndiag, int *info)607
CALL DPBRFS (UPLO, N, NDIAG, NRHS, DA, LDA, DAF,
            LDAF, DB, LDB, DX, LDX, DFERR, DBERR,
            DWORK, IWORK2, INFO)288
void dpbrfs (char uplo, int n, int ndiag, int nrhs, double *da,
            int lda, double *daf, int ldaf, double *db, int ldb,
            double *dx, int ldx, double *dferr, double *dberr, int
            *info)288
CALL DPBSL (DA, LDA, N, NDIAG, DB)610
void dpbsl (double *da, int lda, int n, int ndiag, double *db)
610

```

CALL DPBSV (UPLO, N, NDIAG, NRHS, DA, LDA, DB, LDB, INFO)294
void dpbsv (char uplo, int n, int ndiag, int nrhs, double *da, int lda, double *db, int ldb, int *info)294
CALL DPBSVX (FACT, UPLO, N, NDIAG, NRHS, DA, LDA, DAF, LDAF, EQUED, DSCALE, DB, LDB, DX, LDX, DRCOND, DFERR, DBERR, DWORK, IWORK2, INFO)297
void dpbsvx (char fact, char uplo, int n, int ndiag, int nrhs, double *da, int lda, double *daf, int ldaf, char *equed, double *dscale, double *db, int ldb, double *dx, int ldx, double *drcond, double *dferr, double *dberr, int *info)297
CALL DPBTRF (UPLO, N, NDIAG, DA, LDA, INFO)303
void dpbtrf (char uplo, int n, int ndiag, double *da, int lda, int *info)303
CALL DPBTRS (UPLO, N, NDIAG, NRHS, DA, LDA, DB, LDB, INFO)307
void dpbtrs (char uplo, int n, int ndiag, int nrhs, double *da, int lda, double *db, int ldb, int *info)307
CALL DPOCO (DA, LDA, N, DRCOND, DWORK, INFO)613
void dpoco (double *da, int lda, int n, double *drcond, int *info)613
CALL DPOCON (UPLO, N, DA, LDA, DANORM, DRCOND, DWORK, IWORK2, INFO)311
void dpocon (char uplo, int n, double *da, int lda, double danorm, double *drcond, int *info)311
CALL DPODI (DA, LDA, N, DDET, JOB)616
void dpodi (double *da, int lda, int n, double *ddet, int job)616
CALL DPOEQU (N, DA, LDA, DSCALE, DSCOND, DAMAX, INFO)314
void dpoequ (int n, double *da, int lda, double *dscale, double *dscond, double *damax, int *info)314
CALL DPOFA (DA, LDA, N, INFO)619
void dpofa (double *da, int lda, int n, int *info)619
CALL DPORFS (UPLO, N, NRHS, DA, LDA, DAF, LDAF, DB, LDB, DX, LDX, DFERR, DBERR, DWORK, IWORK2, INFO)318
void dporfs (char uplo, int n, int nrhs, double *da, int lda, double *daf, int ldaf, double *db, int ldb, double *dx, int ldx, double *dferr, double *dberr, int *info)318
CALL DPOSL (DA, LDA, N, DB)622
void dposl (double *da, int lda, int n, double *db)622
CALL DPOSV (UPLO, N, NRHS, DA, LDA, DB, LDB, INFO)323
void dposv (char uplo, int n, int nrhs, double *da, int lda, double *db, int ldb, int *info)323
CALL DPOSVX (FACT, UPLO, N, NRHS, DA, LDA, DAF, LDAF, EQUED, DSCALE, DB, LDB, DX, LDX, DRCOND, DFERR, DBERR, DWORK, IWORK2, INFO)326
void dposvx (char fact, char uplo, int n, int nrhs, double *da, int lda, double *daf, int ldaf, char *equed, double *dscale, double *db, int ldb, double *dx, int ldx, double *drcond, double *dferr, double *dberr, int *info)326
CALL DPOTRF (UPLO, N, DA, LDA, INFO)332
void dpotrf (char uplo, int n, double *da, int lda, int *info)332
CALL DPOTRI (UPLO, N, DA, LDA, INFO)335
void dpotri (char uplo, int n, double *da, int lda, int *info)335
CALL DPOTRS (UPLO, N, NRHS, DA, LDA, DB, LDB, INFO)338
void dpotrs (char uplo, int n, int nrhs, double *da, int lda, double *db, int ldb, int *info)338
CALL DPPCO (DA, N, DRCOND, DWORK, INFO)625
void dppco (double *da, int n, double *drcond, int *info)625
CALL DPPCON (UPLO, N, DA, DANORM, DRCOND, DWORK, IWORK2, INFO)342
void dppcon (char uplo, int n, double *da, double danorm, double *drcond, int *info)342
CALL DPPDI (DA, N, DDET, JOB)628
void dppdi (double *da, int n, double *ddet, int job)628
CALL DPPEQU (UPLO, N, DA, DSCALE, DSCOND, DAMAX, INFO)345
void dppequ (char uplo, int n, double *da, double *dscale, double *scond, double *damax, int *info)345
CALL DPPFA (DA, N, INFO)631
void dppfa (double *da, int n, int *info)631

CALL DPPRFS (UPLO, N, NRHS, DA, DAF, DB, LDB, DX, LDX, DFERR, DBERR, DWORK, IWORK2, INFO) 349

void dpprfs (char uplo, int n, int nrhs, double *da, double *daf, double *db, int ldb, double *dx, int ldx, double *dferr, double *dberr, int *info)349

CALL DPPSL (DA, N, DB)634

void dppsl (double *da, int n, double *db)634

CALL DPPSV (UPLO, N, NRHS, DA, DB, LDB, INFO)354

void dppsv (char uplo, int n, int nrhs, double *da, double *db, int ldb, int *info)354

CALL DPPSVX (FACT, UPLO, N, NRHS, DA, DAF, EQUED, DSCALE, DB, LDB, DX, LDX, DRCOND, DFERR, DBERR, DWORK, IWORK2, INFO)357

void dppsvx (char fact, char uplo, int n, int nrhs, double *da, double *daf, char *equed, double *dscale, double *db, int ldb, double *dx, int ldx, double *drcond, double *dferr, double *dberr, int *info)357

CALL DPPTRF (UPLO, N, DA, INFO)363

void dpptrf (char uplo, int n, double *da, int *info)363

CALL DPPTRI (UPLO, N, DA, INFO)366

void dpptri (char uplo, int n, double *da, int *info)366

CALL DPPTRS (UPLO, N, NRHS, DA, DB, LDB, INFO)369

void dpptrs (char uplo, int n, int nrhs, double *da, double *db, int ldb, int *info)369

CALL DPTCON (N, DDIAG, DOFFD, DANORM, DRCOND, DWORK, INFO)372

void dptcon (int n, double *ddiag, double *doffd, double danorm, double *drcond, int *info)372

CALL DPTRFS (N, NRHS, DDIAG, DOFFD, DDIAGF, DOFFDF, DB, LDB, DX, LDX, DFERR, DBERR, DWORK, INFO)375

void dptrfs (int n, int nrhs, double *ddiag, double *doffd, double *ddiagf, double *doffdf, double *db, int ldb, double *dx, int ldx, double *dferr, double *dberr, int *info)375

CALL DPTSL (N, DDIAG, DOFFD, DB)637

void dptsl (int n, double *ddiag, double *doffd, double *db) 637

CALL DPTSV (N, NRHS, DDIAG, DSUB, DB, LDB, INFO) 381

void dptsv (int n, int nrhs, double *ddiag, double *dsb, double *db, int ldb, int *info)381

CALL DPTSVX (FACT, N, NRHS, DDIAG, DSUB, DDIAGF, DSUBF, DB, LDB, DX, LDX, DRCOND, DFERR, DBERR, DWORK, INFO)384

void dptsvx (char fact, int n, int nrhs, double *ddiag, double *dsb, double *ddiagf, double *dsbf, double *db, int ldb, double *dx, int ldx, double *drcond, double *dferr, double *dberr, int *info)384

CALL DPTTRF (N, DDIAG, DOFFD, INFO)389

void dpttrf (int n, double *ddiag, double *doffd, int *info)389

CALL DPTTRS (UPLO, N, NRHS, DDIAG, DOFFD, DB, LDB, INFO)392

void dpttrs (int n, int nrhs, double *ddiag, double *doffd, double *db, int ldb, int *info)392

DINNER = DQDOTA (N, DB, QC, DX, INCX, DY, INCY)694

DINNER = DQDOTI (N, DB, QC, DX, INCX, DY, INCY)694

CALL DQRDC (DA, LDA, N, P, DQRAUX, IPIVOT, DWORK, JOB)639

void dqrdc (double *da, int lda, int n, int p, double *dqraux, int *ipivot, int job)639

CALL DQRSL (DA, LDA, N, K, DQRAUX, DY, DQY, DQTY, DB, DRESID, DAB, JOB, INFO)642

void dqrsl (double *da, int lda, int n, int k, double *dqraux, double *dy, double *dqy, double *dqty, double *db, double *dresid, double *dab, int job, int *info)642

CALL DROT (N, DX, INCX, DY, INCY, DC, DS)700

void drot (int n, double *dx, int incx, double *dy, int incy, double dc, double ds)700

CALL DROTG (DA, DB, DC, DS)697

void drotg (double *da, double *db, double *dc, double *ds) 697

CALL DROTM (N, DX, INCX, DY, INCY, DPARAM)706

void drotm (int n, double *dx, int incx, double *dy, int incy, double *dparam)706

CALL DROTMG (DD1, DD2, DB1, DB2, DPARAM)703

void drotmg (double *dd1, double *dd2, double *db1, double *db2, double *dparam)703

CALL DSBEV (JOBZ, UPLO, N, NDIAG, DA, LDA, DW, DZ, LDZ, DWORK, INFO)396

void dsbev (char jobz, char uplo, int n, int ndiag, double *da, int lda, double *dw, double *dz, int ldz, int *info)396

CALL DSBEVX (JOBZ, RANGE, UPLO, N, NDIAG, DA,
LDA, DQ, LDQ, DVL, DVU, IL, IU, DABTOL,
NFOUND, DW, DZ, LDZ, DWORK, IWORK2,
IFAIL, INFO)400
void dsbev (char jobz, char range, char uplo, int n, int
ndiag, double *da, int lda, double *dq, int ldq,
double dvl, double dvu, int il, int iu, double dabtol,
int *nfound, double *dw, double *dz, int ldz, int
*ifail, int *info)400
CALL DSBMV (UPLO, N, NDIAG, DALPHA, ZA, LDA, ZX,
INCX, DBETA, ZY, INCY)757
void dsbmv (char uplo, int n, int ndiag, double dalpha,
double *da, int lda, double *dx, int incx, double
dbeta, double *dy, int incy)757
CALL DSCAL (N, DALPHA, DY, INCY)714
void dscal (int n, double da, double *dx, int incx)714
DINNER = DSDOT (N, SX, INCX, SY, INCY)694
CALL DSICO (DA, LDA, N, IPIVOT, DRCOND, DWORK)
646
void dsico (double *da, int lda, int n, int *ipivot, double
*drcond)646
CALL DSDI (DA, LDA, N, IPIVOT, DDET, INERT,
DWORK, JOB)649
void dsidi (double *da, int lda, int n, int *ipivot, double
*ddet, int *inert, int job)649
CALL DSIFA (DA, LDA, N, IPIVOT, INFO)652
void dsifa (double *da, int lda, int n, int *ipivot, int *info)652
void dsinqi (int n, double *dwsave)879
CALL DSINQB (N, DX, DWSAVE)873
void dsinqb (int n, double *dx, double *dwsave)873
CALL DSINQF (N, DX, DWSAVE)876
void dsinqf (int n, double *dx, double *dwsave)876
CALL DSINQI (N, DWSAVE)879
CALL DSINT (N, DX, DWSAVE)881
void dsint (int n, double *dx, double *dwsave)881
CALL DSINTI (N, DWSAVE)884
void dsinti (int n, double *dwsave)884
CALL DSISL (DA, LDA, N, IPIVOT, DB)655
void dsisl (double *da, int lda, int n, int *ipivot, double *db)
655
CALL DSPCO (DA, N, IPIVOT, DRCOND, DWORK)658
void dspco (double *da, int n, int *ipivot, double *drcond)
658
CALL DSPCON (UPLO, N, DA, IPIVOT, DANORM,
DRCOND, DWORK, IWORK2, INFO)405
void dspcon (char uplo, int n, double *da, int *ipivot, double
danorm, double *drcond, int *info)405
CALL DSPDI (DA, N, IPIVOT, DDET, INERT, DWORK, JOB)
661
void dspdi (double *da, int n, int *ipivot, double *ddet, int
*inert, int job)661
CALL DSPEV (JOBZ, UPLO, N, DA, DW, DZ, LDZ,
DWORK, INFO)408
void dspev (char jobz, char uplo, int n, double *da, double
*dw, double *dz, int ldz, int *info)408
CALL DSPEVX (JOBZ, RANGE, UPLO, N, DA, DVL, DVU,
IL, IU, DABTOL, NFOUND, DW, DZ, LDZ,
DWORK, IWORK2, IFAIL, INFO)411
void dspevx (char jobz, char range, char uplo, int n, double
*da, double dvl, double dvu, int il, int iu, double
dabtol, int *nfound, double *dw, double *dz, int ldz,
int *ifail, int *info)411
CALL DSPFA (DA, N, IPIVOT, INFO)664
void dspfa (double *da, int n, int *ipivot, int *info)664
CALL DSPGV (ITYPE, JOBZ, UPLO, N, DA, DB, DW, DZ,
LDZ, DWORK, INFO)415
void dspgv (int itype, char jobz, char uplo, int n, double *da,
double *db, double *dw, double *dz, int ldz, int
*info)415
CALL DSPMV (UPLO, N, DALPHA, DA, DX, INCX,
DBETA, DY, INCY)761
void dspmv (char uplo, int n, double dalpha, double *da,
double *dx, int incx, double dbeta, double *dy, int
incy)761
CALL DSPR (UPLO, N, DALPHA, DX, INCX, DA)764
void dspr (char uplo, int n, double dalpha, double *dx, int
incx, double *da)764
CALL DSPR2 (UPLO, N, DALPHA, DX, INCX, DY, INCY,
DA)767
void dspr2 (char uplo, int n, double dalpha, double *dx, int
incx, double *dy, int incy, double *da)767
CALL DSPRFS (UPLO, N, NRHS, DA, DAF, IPIVOT, DB,
LDB, DX, LDX, DFERR, DBERR, DWORK,
IWORK2, INFO)419


```

void dsprfs (char uplo, int n, int nrhs, double *da, double
             *daf, int *ipivot, double *db, int ldb, double *dx, int
             ldx, double *dferr, double *dberr, int *info)419
CALL DSPSL (DA, N, IPIVOT, DB)667
void dspsl (double *da, int n, int *ipivot, double *db)667
CALL DSPSV (UPLO, N, NRHS, DA, IPIVOT, DB, LDB,
             INFO)425
void dspsv (char uplo, int n, int nrhs, double *da, int *ipivot,
             double *db, int ldb, int *info)425
CALL DSPSVX (FACT, UPLO, N, NRHS, DA, DAF, IPIVOT,
             DB, LDB, DX, LDX, DRCOND, DFERR, DBERR,
             DWORK, IWORK2, INFO)428
void dspsvx (char fact, char uplo, int n, int nrhs, double *da,
             double *daf, int *ipivot, double *db, int ldb, double
             *dx, int ldx, double *drcond, double *dferr, double
             *dberr, int *info)428
CALL DSPTRF (UPLO, N, DA, IPIVOT, INFO)433
void dsptrf (char uplo, int n, double *da, int *ipivot, int *info)
             433
CALL DSPTRI (UPLO, N, DA, IPIVOT, DWORK, INFO)436
void dsptri (char uplo, int n, double *da, int *ipivot, int *info)
             436
CALL DSPTRS (UPLO, N, NRHS, DA, IPIVOT, DB, LDB,
             INFO)439
void dsptrs (char uplo, int n, int nrhs, double *da, int *ipivot,
             double *db, int ldb, int *info)439
CALL DSTEV (JOBZ, N, DDIAG, DOFFD, DZ, LDZ,
            DWORK, INFO)442
void dstev (char jobz, int n, double *ddiag, double *doffd,
            double *dz, int ldz, int *info)442
CALL DSTEVX (JOBZ, RANGE, N, DDIAG, DOFFD, DVL,
            DVU, IL, IU, DABTOL, NFOUND, DW, DZ, LDZ,
            DWORK, IWORK2, IFAIL, INFO)445
void dstevx (char jobz, char range, int n, double *ddiag,
            double *doffd, double dvl, double dvu, int il, int iu,
            double dabtol, int *nfound, double *dw, double *dz,
            int ldz, int *ifail, int *info)445
CALL DSVDC (DA, LDA, N, P, DSVALS, DE, DLSVEC, LDL,
            DRSVEC, LDR, DWORK, JOB, INFO)669
void dsvdc (double *da, int lda, int n, int p, double *dsvals,
            double *de, double *dlsvec, int ldl, double *drsvec,
            int ldr, int job, int *info)669
CALL DSWAP (N, DX, INCX, DY, INCY)716

```

```

void dswap (int n, double *dx, int incx, double *dy, int incy)
            716
CALL DSYCON (UPLO, N, DA, LDA, IPIVOT, DANORM,
            DRCOND, DWORK, IWORK2, INFO)450
void dsycon (char uplo, int n, double *da, int lda, int *ipivot,
            double danorm, double *drcond, int *info)450
CALL DSYEV (JOBZ, UPLO, N, DA, LDA, DW, DWORK,
            LDWORK, INFO)453
void dsyev (char jobz, char uplo, int n, double *da, int lda,
            double *dw, int *info)453
CALL DSYEVX (JOBZ, RANGE, UPLO, N, DA, LDA, DVL,
            DVU, IL, IU, DABTOL, NFOUND, DW, DZ, LDZ,
            DWORK, LDWORK, IWORK2, IFAIL, INFO)456
void dsyevx (char jobz, char range, char uplo, int n, double
            *da, int lda, double dvl, double dvu, int il, int iu,
            double dabtol, int *nfound, double *dw, double *dz,
            int ldz, int *ifail, int *info)456
CALL DSYGV (ITYPE, JOBZ, UPLO, N, DA, LDA, DB, LDB,
            DW, DWORK, LDWORK, INFO)461
void dsygv (int itype, char jobz, char uplo, int n, double *da,
            int lda, double *db, int ldb, double *dw, int *info)
            461
CALL DSYMM (SIDE, UPLO, M, N, DALPHA, DA, LDA,
            DB, LDB, DBETA, DC, LDC)819
void dsymm (char side, char uplo, int m, int n, double
            dalpha, double *da, int lda, double *db, int ldb,
            double dbeta, double *dc, int ldc)819
CALL DSYMV (UPLO, N, DALPHA, DA, LDA, DX, INCX,
            DBETA, DY, INCY)770
void dsymv (char uplo, int n, double dalpha, double *da, int
            lda, double *dx, int incx, double dbeta, double *dy,
            int incy)770
CALL DSYSR (UPLO, N, DALPHA, DX, INCX, DA, LDA)773
void dsyr (char uplo, int n, double dalpha, double *dx, int
            incx, double *da, int lda)773
CALL DSYSR2 (UPLO, N, DALPHA, DX, INCX, DY, INCY,
            DA, LDA)776
void dsyr2 (char uplo, int n, double dalpha, double *dx, int
            incx, double *dy, int incy, double *da, int lda)776
CALL DSYSR2K (UPLO, TRANS, N, K, DALPHA, DA,
            LDA, DB, LDB, DBETA, DC, LDC)826

```


void dsyr2k (char uplo, char trans, int n, int k, double
dalpha, double *da, int lda, double *db, int ldb,
double dbeta, double *dc, int ldc)826

CALL DSYRFS (UPLO, N, NRHS, DA, LDA, DAF, LDAF,
IPIVOT, DB, LDB, DX, LDX, DFERR, DBERR,
DWORK, DWORK2, INFO)465

void dsyrfs (char uplo, int n, int nrhs, double *da, int lda,
double *daf, int ldaf, int *ipivot, double *db, int ldb,
double *dx, int ldx, double *dferr, double *dberr, int
*info)465

CALL DSYRK (UPLO, TRANSA, N, K, DALPHA, DA, LDA,
DBETA, DC, LDC)823

void dsyrk (char uplo, char trans, int n, int k, double dalpha,
double *da, int lda, double dbeta, double *dc, int
ldc)823

CALL DSYSV (UPLO, N, NRHS, DA, LDA, IPIVOT, DB,
LDB, DWORK, LDWORK, INFO)471

void dsysv (char uplo, int n, int nrhs, double *da, int lda, int
*ipivot, double *db, int ldb, int *info)471

CALL DSYSVX (FACT, UPLO, N, NRHS, DA, LDA, DAF,
LDAF, IPIVOT, DB, LDB, DX, LDX, DRCOND,
DFERR, DBERR, DWORK, LDWORK, IWORK2,
INFO)474

void dsysvx (char fact, char uplo, int n, int nrhs, double *da,
int lda, double *daf, int ldaf, int *ipivot, double *db,
int ldb, double *dx, int ldx, double *drcond, double
*dferr, double *dberr, int *info)474

CALL DSYTRF (UPLO, N, DA, LDA, IPIVOT, DWORK,
LDWORK, INFO)479

void dsytrf (char uplo, int n, double *da, int lda, int *ipivot,
int *info)479

CALL DSYTRI (UPLO, N, DA, LDA, IPIVOT, DWORK,
INFO)482

void dsytri (char uplo, int n, double *da, int lda, int *ipivot,
int *info)482

CALL DSYTRS (UPLO, N, NRHS, DA, LDA, IPIVOT, DB,
LDB, INFO)485

void dsytrs (char uplo, int n, int nrhs, double *da, int lda, int
*ipivot, double *db, int ldb, int *info)485

CALL DTBCON (NORM, UPLO, DIAG, N, NDIAG, DA,
LDA, DRCOND, DWORK, IWORK2, INFO)488

void dtbcon (char norm, char uplo, char diag, int n, int
ndiag, double *da, int lda, double *drcond, int
*info)488

CALL DTBMV (UPLO, TRANSA, DIAG, N, NDIAG, DA,
LDA, DY, INCY)779

void dtbm (char uplo, char trans, char diag, int n, int ndiag,
double *da, int lda, double *dx, int incx)779

CALL DTBRFS (UPLO, TRANSA, DIAG, N, NDIAG, NRHS,
DA, LDA, DB, LDB, DX, LDX, DFERR, DBERR,
DWORK, IWORK2, INFO)492

void dtbrfs (char uplo, char trans, char diag, int n, int ndiag,
int nrhs, double *da, int lda, double *db, int ldb,
double *dx, int ldx, double *dferr, double *dberr, int
*info)492

CALL DTBSV (UPLO, TRANSA, DIAG, N, NDIAG, DA,
LDA, DY, INCY)783

CALL DTBTRS (UPLO, TRANSA, DIAG, N, NDIAG, NRHS,
DA, LDA, DB, LDB, INFO)497

void dtbtrs (char uplo, char trans, char diag, int n, int ndiag,
int nrhs, double *da, int lda, double *db, int ldb, int
*info)497

CALL DTPCON (NORM, UPLO, DIAG, N, DA, DRCOND,
DWORK, IWORK2, INFO)501

void dtpcon (char norm, char uplo, char diag, int n, double
*da, double *drcond, int *info)501

CALL DTPMV (UPLO, TRANSA, DIAG, N, DA, DY, INCY)
787

void dtpmv (char uplo, char trans, char diag, int n, double
*da, double *dx, int incx)787

CALL DTPRFS (UPLO, TRANSA, DIAG, N, NRHS, DA, DB,
LDB, DX, LDX, DFERR, DBERR, DWORK,
IWORK2, INFO)504

void dtp (char uplo, char trans, char diag, int n, int nrhs,
double *da, double *db, int ldb, double *dx, int ldx,
double *dferr, double *dberr, int *info)504

CALL DTPSV (UPLO, TRANSA, DIAG, N, DA, DY, INCY)
791

void dtpsv (char uplo, char trans, char diag, int n, double
*da, double *dx, int incx)791

CALL DTPTRI (UPLO, DIAG, N, DA, INFO)509

void dtptri (char uplo, char diag, int n, double *da, int *info)
509

CALL DTPTRS (UPLO, TRANSA, DIAG, N, NRHS, DA, DB, LDB, INFO)512

void dtptrs (char uplo, char trans, char diag, int n, int nrhs, double *da, double *db, int ldb, int *info)512

CALL DTRANS (PLACE, DSCALE, DSOURCE, M, N, DDEST)842

void dtrans (char place, double dscale, double *dsource, int m, int n, double *ddest)842

CALL DTRCO (DA, LDA, N, DRCOND, DWORK, JOB)672

void dtrco (double *da, int lda, int n, double *drcond, int job)672

CALL DTRCON (NORM, UPLO, DIAG, N, DA, LDA, DRCOND, DWORK, IWORK2, INFO)515

void dtrcon (char norm, char uplo, char diag, int n, double *da, int lda, double *drcond, int *info)515

CALL DTRDI (DA, LDA, N, DDET, JOB, INFO)675

void dtrdi (double *da, int lda, int n, double *ddet, int job, int *info)675

CALL DTRMM (SIDE, UPLO, TRANSA, DIAG, M, N, DALPHA, DA, LDA, DB, LDB)830

void dtrmm (char side, char uplo, char transa, char diag, int m, int n, double dalpha, double *da, int lda, double *db, int ldb)830

CALL DTRMV (UPLO, TRANSA, DIAG, N, DA, LDA, DY, INCY)795

void dtrmv (char uplo, char trans, char diag, int n, double *da, int lda, double *dx, int incx)795

CALL DTRRFS (UPLO, TRANSA, DIAG, N, NRHS, DA, LDA, DB, LDB, DX, LDX, DFERR, DBERR, DWORK, IWORK2, INFO)518

CALL DTRSL (DA, LDA, N, DB, JOB, INFO)678

void dtrsl (double *da, int lda, int n, double *db, int job, int *info)678

CALL DTRSM (SIDE, UPLO, TRANSA, DIAG, M, N, DALPHA, DA, LDA, DB, LDB)834

void dtrsm (char side, char uplo, char transa, char diag, int m, int n, double dalpha, double *da, int lda, double *db, int ldb)834

CALL DTRSV (UPLO, TRANSA, DIAG, N, DA, LDA, DY, INCY)798

void dtrsv (char uplo, char trans, char diag, int n, double *da, int lda, double *dx, int incx)798

CALL DTRTRI (UPLO, DIAG, N, DA, LDA, INFO)523

void dtrtri (char uplo, char diag, int n, double *da, int lda, int *info)523

CALL DTRTRS (UPLO, TRANSA, DIAG, N, NRHS, DA, LDA, DB, LDB, INFO)526

void dtrtrs (char uplo, char trans, char diag, int n, int nrhs, double *da, int lda, double *db, int ldb, int *info)526

void dtrrfs (char uplo, char trans, char diag, int n, int nrhs, double *da, int lda, double *db, int ldb, double *dx, int ldx, double *dferr, double *dberr, int *info)518

DL1 = DZASUM (N, ZX, INCX)684

double dzasum (int n, doublecomplex *zx, int incx)684

DL2 = DZNRM2 (N, ZX, INCX)712

double dznrm2(int n, doublecomplex *zx, int incx)712

E

CALL EZFFTB (N, R, AZERO, A, B, WSAVE)857

void ezftb (int n, float *r, float *azero, float *a, float *b, float *wsave)857

CALL EZFFTF (N, R, AZERO, A, B, WSAVE)860

void ezftti (int n, float *wsave)863

CALL EZFFTI (N, WSAVE)863

void ezftf (int n, float *r, float *azero, float *a, float *b, float *wsave)860

I

I = ICAMAX (N, CX, INCX)709

int icamax (int n, complex *cx, int incx)709

I = IDAMAX (N, DX, INCX)709

int idamax (int n, double *dx, int incx)709

I = ISAMAX (N, SX, INCX)709

int isamax (int n, float *sx, int incx)709

I = IZAMAX (N, ZX, INCX)709

int izamax (int n, doublecomplex *zx, int incx)709

R

CALL RFFTB (N, RX, RWSAVE)865

void rftb (int n, float *rx, float *rwsave)866

CALL RFFTF (N, RX, RWSAVE)868

void rftf (int n, float *rx, float *rwsave)869

CALL RFFTI (N, RWSAVE)871

void rffti (int n, float *rwsave)871

S

real sasum (int n, float *sx, int incx)684

SL1 = SASUM (N, SX, INCX)684

CALL SAXPY (N, SALPHA, SX, INCX, SY, INCY)687

void saxpy (int n, float sa, float *sx, int incx, float *sy, int incy)687

real scasum (int n, complex *cx, int incx)684

SL1 = SCASUM (N, CX, INCX)684

CALL SCHDC (SA, LDA, N, SWORK, IPIVOT, JOB, INFO)535

void schdc (float *sa, int lda, int n, int *ipivot, int job, int *info)535

CALL SCHDD (SA, LDA, N, SX, SZ, LDZ, NZ, SY, SRHO, SCOS, SSIN, INFO)538

void schdd (float *sa, int lda, int n, float *sx, float *sz, int ldz, int nz, float *sy, float *srho, float *scos, float *ssin, int *info)538

CALL SCHEX (SA, LDA, N, K, L, SZ, LDZ, NZ, SCOS, SSIN, JOB)542

void schex (float *da, int lda, int n, int k, int l, float *sz, int ldz, int nz, float *scos, float *ssin, int job)542

CALL SCHUD (SA, LDA, N, SX, SZ, LDZ, NZ, SY, SRHO, SCOS, SSIN)546

void schud (float *sa, int lda, int n, float *sx, float *sz, int ldz, int nz, float *sy, float *srho, float *scos, float *ssin)546

float scnrm2(int n, complex *cx, int incx)712

SL2 = SCNRM2 (N, CX, INCX)712

CALL SCOPY (N, SX, INCX, SY, INCY)690

void scopy (int n, float *sx, int incx, float *sy, int incy)690

float sdot(int n, float *sx, int incx, float *sy, int incy)694

SINNER = SDOT (N, SX, INCX, SY, INCY)694

SINNER = SDSDOT (N, SB, SX, INCX, SY, INCY)694

CALL SGBCO (SA, LDA, N, NSUB, NSUPER, IPIVOT, SRCOND, SWORK)549

void sgbc (float *sa, int lda, int n, int nsub, int nsuper, int *ipivot, float *srcond)549

CALL SGBCON (NORM, N, NSUB, NSUPER, SA, LDA, IPIVOT, SANORM, SRCOND, SWORK, IWORK2, INFO)8

void sgbcn (char norm, int n, int nsub, int nsuper, float *sa, int lda, int *ipivot, float sanorm, float *srcond, int *info)8

CALL SGBDI (SA, LDA, N, NSUB, NSUPER, IPIVOT, SDET)553

void sgbd (float *sa, int lda, int n, int nsub, int nsuper, int *ipivot, float *sdet)553

CALL SGBEQU (M, N, NSUB, NSUPER, SA, LDA, SROWSC, SCOLSC, SROWCN, SCOLCN, SAMAX, INFO)12

void sgbequ (int m, int n, int nsub, int nsuper, float *sa, int lda, float *srowsc, float *scolsc, float *srowcn, float *scolcn, float *samax, int *info)12

CALL SGBFA (SA, LDA, N, NSUB, NSUPER, IPIVOT, INFO)556

void sgbf (float *sa, int lda, int n, int nsub, int nsuper, int *ipivot, int *info)556

CALL SGBMV (TRANSA, M, N, NSUB, NSUPER, SALPHA, SA, LDA, SX, INCX, SBETA, SY, INCY)723

void sgbm (char trans, int m, int n, int nsub, int nsuper, float salpha, float *sa, int lda, float *sx, int incx, float sbeta, float *sy, int incy)723

CALL SGBRFS (TRANSA, N, NSUB, NSUPER, NRHS, SA, LDA, SAF, LDAF, IPIVOT, SB, LDB, SX, LDX, SFERR, SBERR, SWORK, IWORK2, INFO)15

void sgbrfs (char trans, int n, int nsub, int nsuper, int nrhs, float *sa, int lda, float *saf, int ldaf, int *ipivot, float *sb, int ldb, float *sx, int ldx, float *sferr, float *sberr, int *info)15

CALL SGBSL (SA, LDA, N, NSUB, NSUPER, IPIVOT, SB, JOB)560

void sgbsl (float *sa, int lda, int n, int nsub, int nsuper, int *ipivot, float *sb, int job)560

CALL SGBSV (N, NSUB, NSUPER, NRHS, SA, LDA, IPIVOT, SB, LDB, INFO)21

void sgbsv (int n, int nsub, int nsuper, int nrhs, float *sa, int lda, int *ipivot, float *sb, int ldb, int *info)21

CALL SGBSVX (FACT, TRANSA, N, NSUB, NSUPER, NRHS, SA, LDA, SAF, LDAF, IPIVOT, EQUED, SROWSC, SCOLSC, SB, LDB, SX, LDX, SRCOND, SFERR, SBERR, SWORK, IWORK2, INFO)25

```

void sgbsvx (char fact, char trans, int n, int nsub, int nsuper,
             int nrhs, float *sa, int lda, float *saf, int ldaf, int
             *ipivot, char *equed, float *srowsc, float *scolsc,
             float *sb, int ldb, float *sx, int ldx, float *srcond, float
             *sferr, float *sberr, int *info)25
CALL SGBTRF (M, N, NSUB, NSUPER, SA, LDA, IPIVOT,
             INFO)33
void sgbrtf (int m, int n, int nsub, int nsuper, float *sa, int lda,
             int *ipivot, int *info)33
CALL SGBTRS (TRANSA, N, NSUB, NSUPER, NRHS, SA,
             LDA, IPIVOT, SB, LDB, INFO)37
void sgbrts (char trans, int n, int nsub, int nsuper, int nrhs,
             float *sa, int lda, int *ipivot, float *sb, int ldb, int
             *info)37
CALL SGECO (SA, LDA, N, IPIVOT, SRCOND, SWORK)564
void sgeco (float *sa, int lda, int n, int *ipivot, float *srcond)
             564
CALL SGECON (NORM, N, SA, LDA, SANORM, SRCOND,
             SWORK, IWORK2, INFO)41
void sgecon (char norm, int n, float *sa, int lda, float sanorm,
             float *srcond, int *info)41
CALL SGEDI (SA, LDA, N, IPIVOT, SDet, SWORK, JOB)567
void sgedi (float *sa, int lda, int n, int *ipivot, float *sdet, int
             job)567
CALL SGEEQU (M, N, SA, LDA, SROWSC, SCOLSC,
             SROWCN, SCOLCN, SAMAX, INFO)44
void sgeequ (int m, int n, float *sa, int lda, float *srowsc, float
             *scolsc, float *srowcn, float *scolcn, float *samax, int
             *info)44
CALL SGEES (JOBZ, SORTEV, SELECT, N, SA, LDA, NOUT,
             SWR, SWI, SZ, LDZ, SWORK, LDWORK,
             BWORK3, INFO)48
void sgees (char jobvs, char sort, L_fp select, int n, float *sa,
             int lda, int *sdim, float *swr, float *swi, float *sz, int
             ldz, int *info)48
CALL SGEESX (JOBZ, SORTEV, SELECT, SENSE, N, SA,
             LDA, NOUT, SWR, SWI, SZ, LDZ, SRCONE,
             SRCONV, SWORK, LDWORK, IWORK2,
             LDWRK2, BWORK3, INFO)53
void sgeesx (char jobvs, char sort, L_fp select, char sense, int
             n, float *sa, int lda, int *sdim, float *swr, float *swi,
             float *sz, int ldz, float *srcone, float *srconv, int
             *info)53

```

```

CALL SGEEV (JOBVL, JOBVR, N, SA, LDA, SWR, SWI, SVL,
            LDVL, SVR, LDVR, SWORK, LDWORK, INFO)59
void sgeev (char jobvl, char jobvr, int n, float *sa, int lda, float
            *swr, float *swi, float *svl, int ldvl, float *svr, int
            ldvr, int *info)59
CALL SGEEVX (BALANC, JOBVL, JOBVR, SENSE, N, SA,
            LDA, SWR, SWI, SVL, LDVL, SVR, LDVR, ILO, IHI,
            SSCALE, SABNRM, SRCONE, SRCONV, SWORK,
            LDWORK, IWORK2, INFO)63
void sgeevx (char balanc, char jobvl, char jobvr, char sense,
            int n, float *sa, int lda, float *swr, float *swi, float
            *svl, int ldvl, float *svr, int ldvr, int *ilo, int *ih, float
            *sscale, float *sabnrm, float *srcone, float *srconv,
            int *info)63
CALL SGEFA (SA, LDA, N, IPIVOT, INFO)570
void sgefa (float *sa, int lda, int n, int *ipivot, int *info)570
CALL SGEGS (JOBVSL, JOBVSR, N, SA, LDA, SB, LDB,
            SALPHAR, SALPHAI, SBETA, SVSL, LDVSL,
            SVSR, LDVSR, SWORK, LDWORK, INFO)69
void sgEGS (char jobvsl, char jobvsr, int n, float *sa, int lda,
            float *sb, int ldb, float *salphar, float *dalphai, float
            *sbeta, float *svsl, int ldvsl, float *svsr, int ldvsr, int
            *info)69
CALL SGEGV (JOBVL, JOBVR, N, SA, LDA, SB, LDB,
            SALPHAR, SALPHAI, SBETA, SVL, LDVL, SVR,
            LDVR, SWORK, LDWORK, INFO)74
void sgEGV (char jobvl, char jobvr, int n, float *sa, int lda, float
            *sb, int ldb, float *salphar, float *dalphai, float
            *sbeta, float *svl, int ldvl, float *svr, int ldvr, int
            *info)74
CALL SGELQF (M, N, SA, LDA, STAU, SWORK, LDWORK,
            INFO)79
void sgelqf (int m, int n, float *sa, int lda, float *stau, int *info)
            79
CALL SGELS (TRANSA, M, N, NRHS, SA, LDA, SB, LDB,
            SWORK, LDWORK, INFO)82
void sgels (char trans, int m, int n, int nrhs, float *sa, int lda,
            float *sb, int ldb, int *info)82
CALL SGELSS (M, N, NRHS, SA, LDA, SB, LDB, SSING,
            SRCOND, IRANK, SWORK, LDWORK, INFO)86
void sgelss (int m, int n, int nrhs, float *sa, int lda, float *sb,
            int ldb, float *ssing, float rcond, int *rank, int *info)
            86

```

CALL SGELSX (M, N, NRHS, SA, LDA, SB, LDB, JPIVOT, SRCOND, IRANK, SWORK, INFO)90
void sgelsx (int m, int n, int nrhs, float *sa, int lda, float *sb, int ldb, int *jpivot, float rcond, int *rank, int *info)90
CALL SGEMM (TRANSA, TRANSB, M, N, K, SALPHA, SA, LDA, SB, LDB, SBETA, SC, LDC)805
void sgemm (char transa, char transb, int m, int n, int k, float salpha, float *sa, int lda, float *sb, int ldb, float sbeta, float *sc, int ldc)805
CALL SGEMV (TRANSA, M, N, SALPHA, SA, LDA, SX, INCX, SBETA, SY, INCY)727
void sgemv (char trans, int m, int n, float salpha, float *sa, int lda, float *sx, int incx, float sbeta, float *sy, int incy)727
CALL SGEQLF (M, N, SA, LDA, STAU, SWORK, LDWORK, INFO)94
void sgeqlf (int m, int n, float *sa, int lda, float *stau, int *info)94
CALL SGEQPF (M, N, SA, LDA, JPIVOT, STAU, SWORK, INFO)97
void sgeqpf (int m, int n, float *sa, int lda, int *jpivot, float *stau, int *info)97
CALL SGEQRF (M, N, SA, LDA, STAU, SWORK, LDWORK, INFO)100
void sgeqrf (int m, int n, float *sa, int lda, float *stau, int *info)100
CALL SGER (M, N, SALPHA, SX, INCX, SY, INCY, SA, LDA)730
void sger (int m, int n, float salpha, float *sx, int incx, float *sy, int incy, float *sa, int lda)730
CALL SGERFS (TRANSA, N, NRHS, SA, LDA, SAF, LDAF, IPIVOT, SB, LDB, SX, LDX, SFERR, SBERR, SWORK, IWORK2, INFO)103
void sgerfs (char trans, int n, int nrhs, float *sa, int lda, float *saf, int ldaf, int *ipivot, float *sb, int ldb, float *sx, int ldx, float *sferr, float *sberr, int *info)103
CALL SGERQF (M, N, SA, LDA, STAU, SWORK, LDWORK, INFO)108
void sgerqf (int m, int n, float *sa, int lda, float *stau, int *info)108
CALL SGESL (SA, LDA, N, IPIVOT, SB, JOB)572
void sgesl (float *sa, int lda, int n, int *ipivot, float *sb, int job)572
CALL SGESV (N, NRHS, SA, LDA, IPIVOT, SB, LDB, INFO)111
void sgesv (int n, int nrhs, float *sa, int lda, int *ipivot, float *sb, int ldb, int *info)111
CALL SGESVD (JOBV, JOBT, M, N, SA, LDA, SSING, SU, LDU, SVT, LDVT, SWORK, LDWORK, INFO)114
void sgesvd (char jobv, char jobt, int m, int n, float *sa, int lda, float *ssing, float *su, int ldu, float *svt, int ldvt, int *info)114
CALL SGESVX (FACT, TRANSA, N, NRHS, SA, LDA, SAF, LDAF, IPIVOT, EQUED, SROWSC, SCOLSC, SB, LDB, SX, LDX, SRCOND, SFERR, SBERR, SWORK, IWORK2, INFO)119
void sgesvx (char fact, char trans, int n, int nrhs, float *sa, int lda, float *saf, int ldaf, int *ipivot, char *equed, float *drowsc, float *scolsc, float *sb, int ldb, float *sx, int ldx, float *srcond, float *sferr, float *sberr, int *info)119
CALL SGETRF (M, N, SA, LDA, IPIVOT, INFO)126
void sgetrf (int m, int n, float *sa, int lda, int *ipivot, int *info)126
void sgetri (int n, float *sa, int lda, int *ipivot, int *info)129
CALL SGETRS (TRANSA, N, NRHS, SA, LDA, IPIVOT, SB, LDB, INFO)132
void sgetrs (char trans, int n, int nrhs, float *sa, int lda, int *ipivot, float *sb, int ldb, int *info)132
CALL SGGGLM (N, M, P, SA, LDA, SB, LDB, SD, SX, SY, SWORK, LDWORK, INFO)135
void sgglm (int n, int m, int p, float *sa, int lda, float *sb, int ldb, float *sd, float *sx, float *dy, int *info)135
CALL SGGLSE (M, N, P, SA, LDA, SB, LDB, SC, SD, SX, SWORK, LDWORK, INFO)139
void sgglse (int m, int n, int p, float *sa, int lda, float *sb, int ldb, float *sc, float *sd, float *sx, int *info)139
CALL SGGSDV (JOBV, JOBT, JOBQ, M, N, P, K, L, SA, LDA, SB, LDB, SALPHA, SBETA, SU, LDU, SV, LDV, SQ, LDQ, SWORK, IWORK3, INFO)144
void sggsdv (char jobv, char jobv, char jobq, int m, int n, int p, int *k, int *l, float *sa, int lda, float *sb, int ldb, float *salpha, float *sbeta, float *su, int ldu, float *sv, int ldv, float *q, int ldq, int *info)144

CALL SGTCON (NORM, N, SLOW, SDIAG, SUP1, SUP2, IPIVOT, SANORM, SRCOND, SWORK, IWORK2, INFO)149

void sgtcon (char norm, int n, float *slow, float *sdiag, float *sup1, float *sup2, int *ipivot, float sanorm, float *srcond, int *info)149

CALL SGTRFS (TRANSA, N, NRHS, SLOW, SDIAG, SUP, SLOWF, SDIAGF, SUPF1, SUPF2, IPIVOT, SB, LDB, SX, LDX, SFERR, SBERR, SWORK, IWORK2, INFO)153

void sgtrfs (char trans, int n, int nrhs, float *slow, float *sdiag, float *sup, float *slowf, float *sdiagf, float *supf1, float *supf2, int *ipivot, float *sb, int ldb, float *sx, int ldx, float *sferr, float *sberr, int *info)153

CALL SGTSL (N, SSUB, SDIAG, SSUPER, SB, INFO)575

void sgtsl (int n, float *ssub, float *sdiag, float *ssuper, float *sb, int *info)575

CALL SGTSV (N, NRHS, SLOW, SDIAG, SUP, SB, LDB, INFO)160

void sgtsv (int n, int nrhs, float *slow, float *sdiag, float *sup, float *sb, int ldb, int *info)160

CALL SGTSVX (FACT, TRANSA, N, NRHS, SLOW, SDIAG, SUP, SLOWF, SDIAGF, SUPF1, SUPF2, IPIVOT, SB, LDB, SX, LDX, SRCOND, SFERR, SBERR, SWORK, IWORK2, INFO)163

void sgtsvx (char fact, char trans, int n, int nrhs, float *slow, float *sdiag, float *sup, float *slowf, float *sdiagf, float *supf1, float *supf2, int *ipivot, float *sb, int ldb, float *sx, int ldx, float *srcond1, float *sferr, float *sberr, int *info)163

CALL SGTTRF (N, SLOW, SDIAG, SUP1, SUP2, IPIVOT, INFO)169

void sgtrf (int n, float *slow, float *sdiag, float *sup1, float *sup2, int *ipivot, int *info)169

CALL SGTTRS (TRANSA, N, NRHS, SLOW, SDIAG, SUP1, SUP2, IPIVOT, SB, LDB, INFO)173

void sgtrs (char trans, int n, int nrhs, float *dlow, float *ddiag, float *dup1, float *dup2, int *ipivot, float *sb, int ldb, int *info)173

void sinq (int n, float *rwsave)879

CALL SINGB (N, RX, RWSAVE)873

void sinqb (int n, float *rx, float *rwsave)873

CALL SINQF (N, RX, RWSAVE)876

void sinqf (int n, float *rx, float *rwsave)876

CALL SINQI (N, RWSAVE)879

CALL SINT (N, RX, RWSAVE)881

void sint (int n, float *rx, float *rwsave)881

CALL SINTI (N, RWSAVE)884

void sinti (int n, float *rwsave)884

float snrm2(int n, float *sx, int incx)712

SL2 = SNRM2 (N, SX, INCX)712

CALL SORGLQ (M, N, K, SA, LDA, STAU, SWORK, LDWORK, INFO)269

void sorglq (int m, int n, int k, float *sa, int lda, float *stau, int *info)269

CALL SORGQL (M, N, K, SA, LDA, STAU, SWORK, LDWORK, INFO)272

void sorgql (int m, int n, int k, float *sa, int lda, float *stau, int *info)272

CALL SORGQR (M, N, K, SA, LDA, STAU, SWORK, LDWORK, INFO)275

void sorgqr (int m, int n, int k, float *sa, int lda, float *stau, int *info)275

CALL SORGRQ (M, N, K, SA, LDA, STAU, SWORK, LDWORK, INFO)278

void sorgrq (int m, int n, int k, float *sa, int lda, float *stau, int *info)278

CALL SPBCO (SA, LDA, N, NDIAG, SRCOND, SWORK, INFO)601

void spbco (float *sa, int lda, int n, int ndiag, float *srcond, int *info)601

CALL SPBCON (UPLO, N, NDIAG, SA, LDA, SANORM, SRCOND, SWORK, IWORK2, INFO)281

void spbcon (char uplo, int n, int ndiag, float *sa, int lda, float sanorm, float *srcond, int *info)281

CALL SPBDI (SA, LDA, N, NDIAG, SDET)604

void spbdi (float *sa, int lda, int n, int ndiag, float *sdet)604

CALL SPBEQU (UPLO, N, NDIAG, SA, LDA, SSCAL, SCOND, SAMAX, INFO)284

void spbequ (char uplo, int n, int ndiag, float *sa, int lda, float *sscale, float *sscond, float *samax, int *info)284

CALL SPBFA (SA, LDA, N, NDIAG, INFO)607

void spbfa (float *sa, int lda, int n, int ndiag, int *info)607

CALL SPBRFS (UPLO, N, NDIAG, NRHS, SA, LDA, SAF, LDAF, SB, LDB, SX, LDX, SFERR, SBERR, SWORK, IWORK2, INFO)288
void spbrfs (char uplo, int n, int ndiag, int nrhs, float *sa, int lda, float *saf, int ldaf, float *sb, int ldb, float *sx, int ldx, float *sferr, float *sberr, int *info)288
CALL SPBSL (SA, LDA, N, NDIAG, SB)610
void spbsl (float *sa, int lda, int n, int ndiag, float *sb)610
CALL SPBSV (UPLO, N, NDIAG, NRHS, SA, LDA, SB, LDB, INFO)294
void spbsv (char uplo, int n, int ndiag, int nrhs, float *sa, int lda, float *sb, int ldb, int *info)294
CALL SPBSVX (FACT, UPLO, N, NDIAG, NRHS, SA, LDA, SAF, LDAF, EQUED, SSCAL, SB, LDB, SX, LDX, SRCOND, SFERR, SBERR, SWORK, IWORK2, INFO)297
void spbsvx (char fact, char uplo, int n, int ndiag, int nrhs, float *sa, int lda, float *saf, int ldaf, char *equed, float *sscale, float *sb, int ldb, float *sx, int ldx, float *srcond, float *sferr, float *sberr, int *info)297
CALL SPBTRF (UPLO, N, NDIAG, SA, LDA, INFO)303
void spbtrf (char uplo, int n, int ndiag, float *sa, int lda, int *info)303
CALL SPBTRS (UPLO, N, NDIAG, NRHS, SA, LDA, SB, LDB, INFO)307
void spbtrs (char uplo, int n, int ndiag, int nrhs, float *sa, int lda, float *sb, int ldb, int *info)307
CALL SPOCO (SA, LDA, N, SRCOND, SWORK, INFO)613
void spoco (float *sa, int lda, int n, float *srcond, int *info)613
CALL SPOCON (UPLO, N, SA, LDA, SANORM, SRCOND, SWORK, IWORK2, INFO)311
void spocon (char uplo, int n, float *sa, int lda, float sanorm, float *srcond, int *info)311
CALL SPODI (SA, LDA, N, SD, JOB)616
void spodi (float *sa, int lda, int n, float *sd, int job)616
CALL SPOEQU (N, SA, LDA, SSCAL, SCOND, SAMAX, INFO)314
void spoequ (int n, float *sa, int lda, float *sscale, float *sscond, float *samax, int *info)314
CALL SPOFA (SA, LDA, N, INFO)619
void spofa (float *sa, int lda, int n, int *info)619

CALL SPORFS (UPLO, N, NRHS, SA, LDA, SAF, LDAF, SB, LDB, SX, LDX, SFERR, SBERR, SWORK, IWORK2, INFO)318
void sporfs (char uplo, int n, int nrhs, float *sa, int lda, float *saf, int ldaf, float *sb, int ldb, float *sx, int ldx, float *sferr, float *sberr, int *info)318
CALL SPOSL (SA, LDA, N, SB)622
void sposl (float *sa, int lda, int n, float *sb)622
CALL SPOSV (UPLO, N, NRHS, SA, LDA, SB, LDB, INFO)323
void sposv (char uplo, int n, int nrhs, float *sa, int lda, float *sb, int ldb, int *info)323
CALL SPOSVX (FACT, UPLO, N, NRHS, SA, LDA, SAF, LDAF, EQUED, SSCAL, SB, LDB, SX, LDX, SRCOND, SFERR, SBERR, SWORK, IWORK2, INFO)326
void sposvx (char fact, char uplo, int n, int nrhs, float *sa, int lda, float *saf, int ldaf, char *equed, float *sscale, float *sb, int ldb, float *sx, int ldx, float *srcond, float *sferr, float *sberr, int *info)326
CALL SPOTRF (UPLO, N, SA, LDA, INFO)332
void spotrf (char uplo, int n, float *sa, int lda, int *info)332
CALL SPOTRI (UPLO, N, SA, LDA, INFO)335
void spotri (char uplo, int n, float *sa, int lda, int *info)335
CALL SPOTRS (UPLO, N, NRHS, SA, LDA, SB, LDB, INFO)338
void spotrs (char uplo, int n, int nrhs, float *sa, int lda, float *sb, int ldb, int *info)338
CALL SPPCO (SA, N, SRCOND, SWORK, INFO)625
void sppco (float *sa, int n, float *srcond, int *info)625
CALL SPPCON (UPLO, N, SA, SANORM, SRCOND, SWORK, IWORK2, INFO)342
void sppcon (char uplo, int n, float *sa, float sanorm, float *srcond, int *info)342
CALL SPPDI (SA, N, SD, JOB)628
void sppdi (float *sa, int n, float *sd, int job)628
CALL SPPEQU (UPLO, N, SA, SSCAL, SCOND, SAMAX, INFO)345
void sppequ (char uplo, int n, float *sa, float *sscale, float *scond, float *samax, int *info)345
CALL SPPFA (SA, N, INFO)631
void sppfa (float *sa, int n, int *info)631

CALL SPPRFS (UPLO, N, NRHS, SA, SAF, SB, LDB, SX, LDX, SFERR, SBERR, SWORK, IWORK2, INFO)349

void spprfs (char uplo, int n, int nrhs, float *sa, float *saf, float *sb, int ldb, float *sx, int ldx, float *sferr, float *sberr, int *info)349

CALL SPPSL (SA, N, SB)634

void sppsl (float *sa, int n, float *sb)634

CALL SPPSV (UPLO, N, NRHS, SA, SB, LDB, INFO)354

void sppsv (char uplo, int n, int nrhs, float *sa, float *sb, int ldb, int *info)354

CALL SPPSVX (FACT, UPLO, N, NRHS, SA, SAF, EQUED, SSCAL, SB, LDB, SX, LDX, SRCOND, SFERR, SBERR, SWORK, IWORK2, INFO)357

void sppsvx (char fact, char uplo, int n, int nrhs, float *sa, float *saf, char *equed, float *sscale, float *sb, int ldb, float *sx, int ldx, float *srcond, float *sferr, float *sberr, int *info)357

CALL SPPTRF (UPLO, N, SA, INFO)363

void spptrf (char uplo, int n, float *sa, int *info)363

CALL SPPTRI (UPLO, N, SA, INFO)366

void spptri (char uplo, int n, float *sa, int *info)366

CALL SPPTRS (UPLO, N, NRHS, SA, SB, LDB, INFO)369

void spptrs (char uplo, int n, int nrhs, float *sa, float *sb, int ldb, int *info)369

CALL SPTCON (N, SDIAG, SOFFD, SANORM, SRCOND, SWORK, INFO)372

void sptcon (int n, float *sdiag, float *soffd, float sanorm, float *srcond, int *info)372

CALL SPTRFS (N, NRHS, SDIAG, SOFFD, SDIAGF, SOFFDF, SB, LDB, SX, LDX, SFERR, SBERR, SWORK, INFO)375

void sptrfs (int n, int nrhs, float *sdiag, float *soffd, float *sdiagf, float *soffdf, float *sb, int ldb, float *sx, int ldx, float *sferr, float *sberr, int *info)375

CALL SPTSL (N, SDIAG, SOFFD, SB)637

void sptsl (int n, float *sdiag, float *soffd, float *sb)637

CALL SPTSV (N, NRHS, SDIAG, SSUB, SB, LDB, INFO)381

void sptsv (int n, int nrhs, float *sdiag, float *ssub, float *sb, int ldb, int *info)381

CALL SPTSVX (FACT, N, NRHS, SDIAG, SSUB, SDIAGF, SSUBF, SB, LDB, SX, LDX, SRCOND, SFERR, SBERR, SWORK, INFO)384

void sptsvx (char fact, int n, int nrhs, float *sdiag, float *ssub, float *sdiagf, float *ssubf, float *sb, int ldb, float *sx, int ldx, float *srcond, float *sferr, float *sberr, int *info)384

CALL SPTTRF (N, SDIAG, SOFFD, INFO)389

void spttrf (int n, float *sdiag, float *soffd, int *info)389

CALL SPTTRS (UPLO, N, NRHS, SDIAG, SOFFD, SB, LDB, INFO)392

void spttrs (int n, int nrhs, float *sdiag, float *soffd, float *sb, int ldb, int *info)392

CALL SQRDC (SA, LDA, N, P, SQRAUX, IPIVOT, SWORK, JOB)639

void sqrdc (float *sa, int lda, int n, int p, float *sqraux, int *ipivot, int job)639

CALL SQRSL (SA, LDA, N, K, SQRAUX, SY, SQY, SQTY, SB, SRESID, SAB, JOB, INFO)642

void sqrsi (float *sa, int lda, int n, int k, float *sqraux, float *sy, float *sqy, float *sqty, float *sb, float *sresid, float *sab, int job, int *info)642

CALL SROT (N, SX, INCX, SY, INCY, SC, SS)700

void srot (int n, float *sx, int incx, float *sy, int incy, float sc, float ss)700

CALL SROTG (SA, SB, SC, SS)697

void srotg (float *sa, float *sb, float *sc, float *ss)697

CALL SROTM (N, SX, INCX, SY, INCY, SPARAM)706

void srotm (int n, float *sx, int incx, float *sy, int incy, float sc, float *sparam)706

CALL SROTMG (SD1, SD2, SB1, SB2, SPARAM)703

void srotmg (float *sd1, float *sd2, float *sb1, float *sb2, float *sparam)703

CALL SSBEV (JOBZ, UPLO, N, NDIAG, SA, LDA, SW, SZ, LDZ, SWORK, INFO)396

void ssbev (char jobz, char uplo, int n, int ndiag, float *sa, int lda, float *sw, float *sz, int ldz, int *info)396

CALL SSBEVX (JOBZ, RANGE, UPLO, N, NDIAG, SA, LDA, SQ, LDQ, SVL, SVU, IL, IU, SABTOL, NFOUND, SW, SZ, LDZ, SWORK, IWORK2, IFAIL, INFO)400

void ssbev (char jobz, char range, char uplo, int n, int ndiag, float *sa, int lda, float *sq, int ldq, float svl, float svu, int il, int iu, float sabtol, int *nfound, float *sw, float *sz, int ldz, int *ifail, int *info)400

CALL SSBMV (UPLO, N, NDIAG, SALPHA, CA, LDA, CX, INCX, SBETA, CY, INCY)757

void ssbmv (char uplo, int n, int ndiag, float salpha, float *sa, int lda, float *sx, int incx, float sbeta, float *sy, int incy)757

CALL SSCAL (N, SALPHA, SY, INCY)714

void sscal(int n, float sa, float *sx, int incx)714

CALL SSICO (SA, LDA, N, IPIVOT, SRCOND, SWORK)646

void ssico (float *sa, int lda, int n, int *ipivot, float *srcond)646

CALL SSIDI (SA, LDA, N, IPIVOT, SDET, INERT, SWORK, JOB)649

void ssidi (float *sa, int lda, int n, int *ipivot, float *sdet, int *inert, int job)649

CALL SSIFA (SA, LDA, N, IPIVOT, INFO)652

void ssifa (float *sa, int lda, int n, int *ipivot, int *info)652

CALL SSISL (SA, LDA, N, IPIVOT, SB)655

void ssisl (float *sa, int lda, int n, int *ipivot, float *sb)655

CALL SSPCO (SA, N, IPIVOT, SRCOND, SWORK)658

void sspco (float *sa, int n, int *ipivot, float *srcond)658

CALL SSPCON (UPLO, N, SA, IPIVOT, SANORM, SRCOND, SWORK, IWORK2, INFO)405

void sspcon (char uplo, int n, float *sa, int *ipivot, float sanorm, float *srcond, int *info)405

CALL SSPDI (SA, N, IPIVOT, SDET, INERT, SWORK, JOB)661

void sspdi (float *sa, int n, int *ipivot, float *sdet, int *inert, int job)661

CALL SSPEV (JOBZ, UPLO, N, SA, SW, SZ, LDZ, SWORK, INFO)408

void sspev (char jobz, char uplo, int n, float *sa, float *sw, float *sz, int ldz, int *info)408

CALL SSPEVX (JOBZ, RANGE, UPLO, N, SA, SVL, SVU, IL, IU, SBTOL, NFOUND, SW, SZ, LDZ, SWORK, IWORK2, IFAIL, INFO)411

void sspevx (char jobz, char range, char uplo, int n, float *sa, float svl, float svu, int il, int iu, float sbtol, int *nfound, float *sw, float *sz, int ldz, int *ifail, int *info)411

CALL SSPFA (SA, N, IPIVOT, INFO)664

void sspfa (float *sa, int n, int *ipivot, int *info)664

CALL SSPGV (ITYPE, JOBZ, UPLO, N, SA, SB, SW, SZ, LDZ, SWORK, INFO)415

void sspgv (int itype, char jobz, char uplo, int n, float *sa, float *sb, float *sw, float *sz, int ldz, int *info)415

CALL SSPMV (UPLO, N, SALPHA, SA, SX, INCX, SBETA, SY, INCY)761

void sspmv (char uplo, int n, float salpha, float *sa, float *sx, int incx, float sbeta, float *sy, int incy)761

CALL SSPR (UPLO, N, SALPHA, SX, INCX, SA)764

void sspr (char uplo, int n, float salpha, float *sx, int incx, float *sa)764

CALL SSPR2 (UPLO, N, SALPHA, SX, INCX, SY, INCY, SA)767

void sspr2 (char uplo, int n, float salpha, float *sx, int incx, float *sy, int incy, float *sa)767

CALL SSPRFS (UPLO, N, NRHS, SA, SAF, IPIVOT, SB, LDB, SX, LDX, SFERR, SBERR, SWORK, IWORK2, INFO)419

void ssprfs (char uplo, int n, int nrhs, float *sa, float *saf, int *ipivot, float *sb, int ldb, float *sx, int ldx, float *sferr, float *sberr, int *info)419

CALL SSPSL (SA, N, IPIVOT, SB)667

void sspsl (float *sa, int n, int *ipivot, float *sb)667

CALL SSPSV (UPLO, N, NRHS, SA, IPIVOT, SB, LDB, INFO)425

void sspsv (char uplo, int n, int nrhs, float *sa, int *ipivot, float *sb, int ldb, int *info)425

CALL SSPSVX (FACT, UPLO, N, NRHS, SA, SAF, IPIVOT, SB, LDB, SX, LDX, SRCOND, SFERR, SBERR, SWORK, IWORK2, INFO)428

void sspsvx (char fact, char uplo, int n, int nrhs, float *sa, float *saf, int *ipivot, float *sb, int ldb, float *sx, int ldx, float *srcond, float *sferr, float *sberr, int *info)428

CALL SSPTRF (UPLO, N, SA, IPIVOT, INFO)433

void ssptrf (char uplo, int n, float *sa, int *ipivot, int *info)433

CALL SSPTRI (UPLO, N, SA, IPIVOT, SWORK, INFO)436

void ssptri (char uplo, int n, float *sa, int *ipivot, int *info)436

CALL SSPTRS (UPLO, N, NRHS, SA, IPIVOT, SB, LDB, INFO)439

void ssptrs (char uplo, int n, int nrhs, float *sa, int *ipivot, float *sb, int ldb, int *info)439

CALL SSTEVC (JOBZ, N, SDIAG, SOFFD, SZ, LDZ, SWORK, INFO)442

void sstev (char jobz, int n, float *sdiag, float *soffd, float *sz, int ldz, int *info)442

CALL SSTEVCX (JOBZ, RANGE, N, SDIAG, SOFFD, SVL, SVU, IL, IU, SABTOL, NFOUND, SW, SZ, LDZ, SWORK, IWORK2, IFAIL, INFO)445

void sstevx (char jobz, char range, int n, float *sdiag, float *soffd, float svl, float svu, int il, int iu, float sabsol, int *nfound, float *sw, float *sz, int ldz, int *ifail, int *info)445

CALL SSVDC (SA, LDA, N, P, SSVALS, SE, SLSVEC, LDL, SRSVEC, LDR, SWORK, JOB, INFO)669

void ssvdc (float *sa, int lda, int n, int p, float *ssvals, float *se, float *svec, int ldl, float *srsvec, int ldr, int job, int *info)669

CALL SSWAP (N, SX, INCX, SY, INCY)716

void sswap (int n, float *sx, int incx, float *sy, int incy)716

CALL SSYCON (UPLO, N, SA, LDA, IPIVOT, SANORM, SRCOND, SWORK, IWORK2, INFO)450

void ssycon (char uplo, int n, float *sa, int lda, int *ipivot, float sanorm, float *srcond, int *info)450

CALL SSYEV (JOBZ, UPLO, N, SA, LDA, SW, SWORK, LDWORK, INFO)453

void ssyev (char jobz, char uplo, int n, float *sa, int lda, float *sw, int *info)453

CALL SSYEVX (JOBZ, RANGE, UPLO, N, SA, LDA, SVL, SVU, IL, IU, SABTOL, NFOUND, SW, SZ, LDZ, SWORK, LDWORK, IWORK2, IFAIL, INFO)456

void ssyevx (char jobz, char range, char uplo, int n, float *sa, int lda, float svl, float svu, int il, int iu, float sabsol, int *nfound, float *sw, float *sz, int ldz, int *ifail, int *info)456

CALL SSYGV (ITYPE, JOBZ, UPLO, N, SA, LDA, SB, LDB, SW, SWORK, LDWORK, INFO)461

void ssygv (int itype, char jobz, char uplo, int n, float *sa, int lda, float *sb, int ldb, float *sw, int *info)461

CALL SSYMM (SIDE, UPLO, M, N, SALPHA, SA, LDA, SB, LDB, SBETA, SC, LDC)819

void ssymm (char side, char uplo, int m, int n, float salpha, float *sa, int lda, float *sb, int ldb, float sbeta, float *sc, int ldc)819

CALL SSYMV (UPLO, N, SALPHA, SA, LDA, SX, INCX, SBETA, SY, INCY)770

void ssymv (char uplo, int n, float salpha, float *sa, int lda, float *sx, int incx, float sbeta, float *sy, int incy)770

CALL SSYR (UPLO, N, SALPHA, SX, INCX, SA, LDA)773

void ssyr (char uplo, int n, float salpha, float *sx, int incx, float *sa, int lda)773

CALL SSYR2 (UPLO, N, SALPHA, SX, INCX, SY, INCY, SA, LDA)776

void ssyr2 (char uplo, int n, float salpha, float *sx, int incx, float *sy, int incy, float *sa, int lda)776

CALL SSYR2K (UPLO, TRANSA, N, K, SALPHA, SA, LDA, SB, LDB, SBETA, SC, LDC)826

void ssyr2k (char uplo, char trans, int n, int k, float salpha, float *sa, int lda, float *sb, int ldb, float sbeta, float *sc, int ldc)826

CALL SSYRFS (UPLO, N, NRHS, SA, LDA, SAF, LDAF, IPIVOT, SB, LDB, SX, LDX, SFERR, SBERR, SWORK, SWORK2, INFO)465

void ssyrfs (char uplo, int n, int nrhs, float *sa, int lda, float *saf, int ldaf, int *ipivot, float *sb, int ldb, float *sx, int ldx, float *sferr, float *sberr, int *info)465

CALL SSYRK (UPLO, TRANSA, N, K, SALPHA, SA, LDA, SBETA, SC, LDC)823

void ssyrk (char uplo, char trans, int n, int k, float salpha, float *sa, int lda, float sbeta, float *sc, int ldc)823

CALL SSYSV (UPLO, N, NRHS, CA, LDA, IPIVOT, CB, LDB, CWORK, LDWORK, INFO)471

CALL SSYSV (UPLO, N, NRHS, SA, LDA, IPIVOT, SB, LDB, SWORK, LDWORK, INFO)471

void ssysv (char uplo, int n, int nrhs, float *sa, int lda, int *ipivot, float *sb, int ldb, int *info)471

CALL SSYSVX (FACT, UPLO, N, NRHS, SA, LDA, SAF, LDAF, IPIVOT, SB, LDB, SX, LDX, SRCOND, SFERR, SBERR, SWORK, LDWORK, IWORK2, INFO)474

void ssysvx (char fact, char uplo, int n, int nrhs, float *sa, int lda, float *saf, int ldaf, int *ipivot, float *sb, int ldb, float *sx, int ldx, float *srcond, float *sferr, float *sberr, int *info)474

CALL SSYTRF (UPLO, N, SA, LDA, IPIVOT, SWORK, LDWORK, INFO)479

```

void ssytrf (char uplo, int n, float *sa, int lda, int *ipivot, int
*info)479
CALL SSYTRI (UPLO, N, SA, LDA, IPIVOT, SWORK, INFO)
482
void ssytri (char uplo, int n, float *sa, int lda, int *ipivot, int
*info)482
CALL SSYTRS (UPLO, N, NRHS, SA, LDA, IPIVOT, SB,
LDB, INFO)485
void ssytrs (char uplo, int n, int nrhs, float *sa, int lda, int
*ipivot, float *sb, int ldb, int *info)485
CALL STBCON (NORM, UPLO, DIAG, N, NDIAG, SA,
LDA, SRCOND, SWORK, IWORK2, INFO)488
void stbcon (char norm, char uplo, char diag, int n, int ndiag,
float *sa, int lda, float *srcond, int *info)488
CALL STBMV (UPLO, TRANSA, DIAG, N, NDIAG, SA,
LDA, SY, INCY)779
void stbm (char uplo, char trans, char diag, int n, int ndiag,
float *sa, int lda, float *sx, int incx)779
CALL STBRFS (UPLO, TRANSA, DIAG, N, NDIAG, NRHS,
SA, LDA, SB, LDB, SX, LDX, SFERR, SBERR,
SWORK, IWORK2, INFO)492
void stbrfs (char uplo, char trans, char diag, int n, int ndiag,
int nrhs, float *sa, int lda, float *sb, int ldb, float *sx,
int ldx, float *sferr, float *sberr, int *info)492
CALL STBSV (UPLO, TRANSA, DIAG, N, NDIAG, SA,
LDA, SY, INCY)783
void dtbsv (char uplo, char trans, char diag, int n, int ndiag,
double *da, int lda, double *dx, int incx)783
void stbsv (char uplo, char trans, char diag, int n, int ndiag,
float *sa, int lda, float *sx, int incx)783
CALL STBTRS (UPLO, TRANSA, DIAG, N, NDIAG, NRHS,
SA, LDA, SB, LDB, INFO)497
void stbtrs (char uplo, char trans, char diag, int n, int ndiag,
int nrhs, float *sa, int lda, float *sb, int ldb, int *info)
497
CALL STPCON (NORM, UPLO, DIAG, N, SA, SRCOND,
SWORK, IWORK2, INFO)501
void stpcon (char norm, char uplo, char diag, int n, float *sa,
float *srcond, int *info)501
CALL STPMV (UPLO, TRANSA, DIAG, N, SA, SY, INCY)
787
void stpmv (char uplo, char trans, char diag, int n, float *sa,
float *sx, int incx)787
CALL STPRFS (UPLO, TRANSA, DIAG, N, NRHS, SA, SB,
LDB, SX, LDX, SFERR, SBERR, SWORK, IWORK2,
INFO)504
void stprfs (char uplo, char trans, char diag, int n, int nrhs,
float *sa, float *sb, int ldb, float *sx, int ldx, float
*sferr, float *sberr, int *info)504
CALL STPSV (UPLO, TRANSA, DIAG, N, SA, SY, INCY)791
void stpsv (char uplo, char trans, char diag, int n, float *sa,
float *sx, int incx)791
CALL STPTRI (UPLO, DIAG, N, SA, INFO)509
void stptri (char uplo, char diag, int n, float *sa, int *info)509
CALL STPTRS (UPLO, TRANSA, DIAG, N, NRHS, SA, SB,
LDB, INFO)512
void stptrs (char uplo, char trans, char diag, int n, int nrhs,
float *sa, float *sb, int ldb, int *info)512
CALL STRANS (PLACE, SSCAL, SSOURCE, M, N, SDEST)
842
void strans (char place, float sscale, float *ssource, int m, int
n, float *sdest)842
CALL STRCO (SA, LDA, N, SRCOND, SWORK, JOB)672
void strco (float *sa, int lda, int n, float *srcond, int job)672
CALL STRCON (NORM, UPLO, DIAG, N, SA, LDA,
SRCOND, SWORK, IWORK2, INFO)515
void strcon (char norm, char uplo, char diag, int n, float *sa,
int lda, float *srcond, int *info)515
CALL STRDI (SA, LDA, N, SDET, JOB, INFO)675
void strdi (float *sa, int lda, int n, float *sdet, int job, int *info)
675
CALL STRMM (SIDE, UPLO, TRANSA, DIAG, M, N,
SALPHA, SA, LDA, SB, LDB)830
void strmm (char side, char uplo, char transa, char diag, int
m, int n, float salpha, float *sa, int lda, float *sb, int
ldb)830
CALL STRMV (UPLO, TRANSA, DIAG, N, SA, LDA, SY,
INCY)795
void strmv (char uplo, char trans, char diag, int n, float *sa,
int lda, float *sx, int incx)795
CALL STRRFS (UPLO, TRANSA, DIAG, N, NRHS, SA,
LDA, SB, LDB, SX, LDX, SFERR, SBERR, SWORK,
IWORK2, INFO)518
void strrfs (char uplo, char trans, char diag, int n, int nrhs,
float *sa, int lda, float *sb, int ldb, float *sx, int ldx,
float *sferr, float *sberr, int *info)518

```

CALL STRSL (SA, LDA, N, SB, JOB, INFO)678
void strsl (float *sa, int lda, int n, float *sb, int job, int *info)
678
CALL STRSM (SIDE, UPLO, TRANSA, DIAG, M, N,
SALPHA, SA, LDA, SB, LDB)834
void strsm (char side, char uplo, char transa, char diag, int
m, int n, float salpha, float *sa, int lda, float *sb, int
ldb)834
CALL STRSV (UPLO, TRANSA, DIAG, N, SA, LDA, SY,
INCY)798
void strsv (char uplo, char trans, char diag, int n, float *sa, int
lda, float *sx, int incx)798
CALL STRTRI (UPLO, DIAG, N, SA, LDA, INFO)523
void strtri (char uplo, char diag, int n, float *sa, int lda, int
*info)523
CALL STRTRS (UPLO, TRANSA, DIAG, N, NRHS, SA,
LDA, SB, LDB, INFO)526
void strtrs (char uplo, char trans, char diag, int n, int nrhs,
float *sa, int lda, float *sb, int ldb, int *info)526

V

CALL VCOSQB (M, N, RX, RXT, MDIMX, RWSAVE)844
void vcosqb (int m, int n, float *rx, int mdimx, float *rwsave)
844
CALL VCOSQF (M, N, RX, RXT, MDIMX, RWSAVE)847
void vcosqf (int m, int n, float *rx, int mdimx, float *rwsave)
847
CALL VCOSQI (N, RWSAVE)850
void vcosqi (int n, float *rwsave)850
CALL VCOST (M,N, RX, RXT, MDIMX, RWSAVE)852
void vcost (int m, int n, float *rx, int mdimx, float *rwsave)
852
CALL VCOSTI (N, RWSAVE)855
void vcosti (int n, float *rwsave)855
CALL VDCOSQB (M, N, DX, DXT, MDIMX, DWSAVE)844
void vdcosqb (int m, int n, double *dx, int mdimx, double
*dwsave)844
CALL VDCOSQF (M, N, DX, DXT, MDIMX, DWSAVE)847
void vdcosqf (int m, int n, double *dx, int mdimx, double
*dwsave)847
CALL VDCOSQI (N, DWSAVE)850

void vdcosqi (int n, double *dwsave)850
CALL VDCOST (M,N, DX, DXT, MDIMX, DWSAVE)852
void vdcost (int m, int n, double *dx, int mdimx, double
*dwsave)852
CALL VDCOSTI (N, DWSAVE)855
void vdcosti (int n, double *dwsave)855
CALL VDDFTB (M, N, DX, DXT, MDIMX, DWSAVE)866
void vdftb (int m, int n, double *rx, int mdimx, double
*dwsave)866
CALL VDDFTF (M, N, DX, DXT, MDIMX, DWSAVE)869
void vdftf (int m, int n, double *dx, int mdimx, double
*dwsave)869
CALL VDDFTI (N, DWSAVE)871
void vdfti (int n, double *dwsave)871
CALL VDSINQB (M, N, DX, DXT, MDIMX, DWSAVE)873
void vdsinqb (int m, int n, double *dx, int mdimx, double
*dwsave)873
CALL VDSINQF (M, N, DX, DXT, MDIMX, DWSAVE)876
void vdsinqf (int m, int n, double *dx, int mdimx, double
*dwsave)876
CALL VDSINQI (N, DWSAVE)879
void vdsinqi (int n, double *dwsave)879
CALL VDSINT (M, N, DX, DXT, MDIMX, DWSAVE)881
void vdsint (int m, int n, double *dx, int mdimx, double
*dwsave)881
CALL VDSINTI (N, WSAVE)884
void vdsinti (int n, double *dwsave)884
CALL VRFFTB (M, N, RX, RXT, MDIMX, RWSAVE)866
void vrftb (int m, int n, float *rx, int mdimx, float *rwsave)
866
CALL VRFFTF (M, N, RX, RXT, MDIMX, RWSAVE)869
void vrftf (int m, int n, float *rx, int mdimx, float *rwsave)
869
CALL VRFFTI (N, RWSAVE)871
void vrfti (int n, float *rwsave)871
CALL VSINQB (M, N, RX, RXT, MDIMX, RWSAVE)873
void vsinqb (int m, int n, float *rx, int mdimx, float *rwsave)
873
CALL VSINQF (M, N, RX, RXT, MDIMX, RWSAVE)876
void vsinqf (int m, int n, float *rx, int mdimx, float *rwsave)
876

CALL VSINQI (N, RWSAVE)879
 void vsinqi (int n, float *rwsave)879
 CALL VSINT (M, N, RX, RXT, MDIMX, RWSAVE)881
 void vsint (int m, int n, float *rx, int mdimx, float *rwsave)
 881
 CALL VSINTI (N, WSAVE)884
 void vsinti (int n, float *rwsave)884

Z

CALL ZAXPY (N, ZALPHA, ZX, INCX, ZY, INCY)687
 void zaxpy (int n, doublecomplex *za, doublecomplex *zx,
 int incx, doublecomplex *zy, int incy)687
 CALL ZCHDC (ZA, LDA, N, ZWORK, IPIVOT, JOB, INFO)
 535
 void zchdc (doublecomplex *za, int lda, int n, int *ipivot, int
 job, int *info)535
 CALL ZCHDD (ZA, LDA, N, ZX, ZZ, LDZ, NZ, ZY, DRHO,
 DCOS, DSIN, INFO)538
 void zchdd (doublecomplex *za, int lda, int n,
 doublecomplex *zx, doublecomplex *zz, int ldz, int
 nz, doublecomplex *zy, double *drho, double *dcos,
 double *dsin, int *info)538
 CALL ZCHEX (CA, LDA, N, K, L, CZ, LDZ, NZ, SCOS,
 CSIN, JOB)542
 void zchex (doublecomplex *za, int lda, int n, int k, int l,
 doublecomplex *zz, int ldz, int nz, double *dcos,
 doublecomplex *zsin, int job)542
 CALL ZCHUD (ZA, LDA, N, ZX, ZZ, LDZ, NZ, ZY, DRHO,
 DCOS, ZSIN)546
 void zchud (doublecomplex *za, int lda, int n,
 doublecomplex *zx, doublecomplex *zz, int ldz, int
 nz, doublecomplex *zy, double *drho, double *dcos,
 doublecomplex *zsin)546
 CALL ZCOPY (N, ZX, INCX, ZY, INCY)690
 void zcopy (int n, doublecomplex *zx, int incx,
 doublecomplex *zy, int incy)690
 doublecomplex *zdotc(int n, doublecomplex *cx, int incx,
 doublecomplex *cy, int incy)694
 ZINNER = ZDOTC (N, ZX, INCX, ZY, INCY)694
 doublecomplex *zdotu(int n, doublecomplex *cx, int incx,
 doublecomplex *cy, int incy)694
 ZINNER = ZDOTU (N, ZX, INCX, ZY, INCY)694
 CALL ZDSCAL (N, DALPHA, ZY, INCY)714
 void zdscal(int n, double da, doublecomplex *zx, int incx)
 714
 CALL ZFFTB (N, DX, DWSAVE)866
 void zfftb (int n, doublecomplex *dx, doublecomplex
 *dwsave)866
 CALL ZFFTF (N, ZR, ZWSAVE)868
 void zfftf (int n, doublecomplex *zr, doublecomplex *wsave)
 869
 CALL ZFFTI (N, DWSAVE)871
 void zffti (int n, doublecomplex *dwsave)871
 CALL ZGBCO (ZA, LDA, N, NSUB, NSUPER, IPIVOT,
 DRCOND, ZWORK)549
 void zgbco (doublecomplex *za, int lda, int n, int nsub, int
 nsuper, int *ipivot, double *drcond)549
 CALL ZGBCON (NORM, N, NSUB, NSUPER, ZA, LDA,
 IPIVOT, DANORM, DRCOND, ZWORK,
 DWORK2, INFO)8
 void zgbcon (char norm, int n, int nsub, int nsuper,
 doublecomplex *za, int lda, int *ipivot, double
 danorm, double *drcond, int *info)8
 CALL ZGBDI (ZA, LDA, N, NSUB, NSUPER, IPIVOT,
 ZDET)553
 void zgbdi (doublecomplex *za, int lda, int n, int nsub, int
 nsuper, int *ipivot, doublecomplex *zdet)553
 CALL ZGBEQU (M, N, NSUB, NSUPER, ZA, LDA,
 DROWSC, DCOLSC, DROWCN, DCOLCN,
 DAMAX, INFO)12
 void zgbegu (int m, int n, int nsub, int nsuper,
 doublecomplex *za, int lda, double *drowsc, double
 *dcolsc, double *drowcn, double *dcolcn, double
 *damax, int *info)12
 CALL ZGBFA (ZA, LDA, N, NSUB, NSUPER, IPIVOT,
 INFO)556
 void zgbfa (doublecomplex *za, int lda, int n, int nsub, int
 nsuper, int *ipivot, int *info)556
 CALL ZGBMV (TRANSA, M, N, NSUB, NSUPER,
 ZALPHA, ZA, LDA, ZX, INCX, ZBETA, ZY, INCY)
 723
 void zgbmv (char trans, int m, int n, int nsub, int nsuper,
 doublecomplex *zalpha, doublecomplex *za, int
 lda, doublecomplex *zx, int incx, doublecomplex
 *zbeta, doublecomplex *zy, int incy)723

CALL ZGBRFS (TRANSA, N, NSUB, NSUPER, NRHS, ZA, LDA, ZAF, LDAF, IPIVOT, ZB, LDB, ZX, LDX, ZFERR, ZBERR, ZWORK, DWORK2, INFO)15

void zgbfrs (char trans, int n, int nsub, int nsuper, int nrhs, doublecomplex *za, int lda, doublecomplex *zaf, int ldaf, int *ipivot, doublecomplex *zb, int ldb, doublecomplex *zx, int ldx, double *dferr, double *dberr, int *info)15

CALL ZGBSL (ZA, LDA, N, NSUB, NSUPER, IPIVOT, ZB, JOB)560

void zgbsl (doublecomplex *za, int lda, int n, int nsub, int nsuper, int *ipivot, doublecomplex *zb, int job)560

CALL ZGBSV (N, NSUB, NSUPER, NRHS, ZA, LDA, IPIVOT, ZB, LDB, INFO)21

void zgbsv (int n, int nsub, int nsuper, int nrhs, doublecomplex *za, int lda, int *ipivot, doublecomplex *zb, int ldb, int *info)21

CALL ZGBSVX (FACT, TRANSA, N, NSUB, NSUPER, NRHS, ZA, LDA, ZAF, LDAF, IPIVOT, EQUED, DROWSC, DCOLSC, ZB, LDB, ZX, LDX, DRCOND, DFERR, DBERR, ZWORK, DWORK2, INFO)25

void zgbsvx (char fact, char trans, int n, int nsub, int nsuper, int nrhs, doublecomplex *za, int lda, doublecomplex *zaf, int ldaf, int *ipivot, char *equed, double *drowsc, double *dcols, doublecomplex *zb, int ldb, doublecomplex *zx, int ldx, double *drcond, double *dferr, double *dberr, int *info)25

CALL ZGBTRF (M, N, NSUB, NSUPER, ZA, LDA, IPIVOT, INFO)33

void zgbtrf (int m, int n, int nsub, int nsuper, doublecomplex *za, int lda, int *ipivot, int *info)33

CALL ZGBTRS (TRANSA, N, NSUB, NSUPER, NRHS, ZA, LDA, IPIVOT, ZB, LDB, INFO)37

void zgbtrs (char trans, int n, int nsub, int nsuper, int nrhs, doublecomplex *za, int lda, int *ipivot, doublecomplex *zb, int ldb, int *info)37

CALL ZGECO (ZA, LDA, N, IPIVOT, DRCOND, ZWORK)564

void zgeco (doublecomplex *za, int lda, int n, int *ipivot, double *drcond)564

CALL ZGECN (NORM, N, ZA, LDA, DANORM, DRCOND, ZWORK, DWORK2, INFO)41

void zgecon (char norm, int n, doublecomplex *za, int lda, double danorm, double *drcond, int *info)41

CALL ZGEDI (ZA, LDA, N, IPIVOT, ZDET, ZWORK, JOB)567

void zgedi (doublecomplex *za, int lda, int n, int *ipivot, doublecomplex *zdet, int job)567

CALL ZGEEQU (M, N, ZA, LDA, DROWSC, DCOLSC, DROWCN, DCOLCN, DAMAX, INFO)44

void zgeequ (int m, int n, doublecomplex *za, int lda, double *drowcn, double *dcolcn, double *drowcn, double *dcolcn, double *damax, int *info)44

CALL ZGEES (JOBZ, SORTEV, SELECT, N, ZA, LDA, NOUT, ZW, ZZ, LDZ, ZWORK, LDWORK, DWORK2, BWORK3, INFO)48

void zgees (char jobvs, char sort, L_fp select, int n, doublecomplex *za, int lda, int *sdim, doublecomplex *zw, doublecomplex *zz, int ldz, int *info)48

CALL ZGEESX (JOBZ, SORTEV, SELECT, SENSE, N, ZA, LDA, NOUT, ZW, ZZ, LDZ, DRCONC, DRCONV, ZWORK, LDWORK, DWORK2, BWORK3, INFO)53

void zgeesx (char jobvs, char sort, L_fp select, char sense, int n, doublecomplex *za, int lda, int *sdim, doublecomplex *zw, doublecomplex *zz, int ldz, double *drcone, double *drconv, int *info)53

CALL ZGEEV (JOBVL, JOBVR, N, ZA, LDA, ZW, ZVL, LDVL, ZVR, LDVR, ZWORK, LDWORK, DWORK2, INFO)59

void zgeev (char jobvl, char jobvr, int n, doublecomplex *za, int lda, doublecomplex *zw, doublecomplex *vl, int ldvl, doublecomplex *zvr, int ldvr, int *info)59

CALL ZGEEVX (BALANC, JOBVL, JOBVR, SENSE, N, ZA, LDA, ZW, ZVL, LDVL, ZVR, LDVR, ILO, IHI, DSCALE, DABNRM, DRCONC, DRCONV, ZWORK, LDWORK, DWORK2, INFO)63

void zgeevx (char balanc, char jobvl, char jobvr, char sense, int n, doublecomplex *za, int lda, doublecomplex *zw, doublecomplex *vl, int ldvl, doublecomplex *zvr, int ldvr, int *ilo, int *ihi, double *dscale, double *dabnrm, double *drcone, double *drconv, int *info)64

CALL ZGEFA (ZA, LDA, N, IPIVOT, INFO)570


```

void zgefa (doublecomplex *za, int lda, int n, int *ipivot, int
*info)570
CALL ZGEGS (JOBVSL, JOBVSR, N, ZA, LDA, ZB, LDB,
ZALPHA, ZBETA, ZVSL, LDVSL, ZVSR, LDVSR,
ZWORK, LDWORK, DWORK2, INFO)69
void zgegs (char jobvsl, char jobvsr, int n, doublecomplex
*za, int lda, doublecomplex *zb, int ldb,
doublecomplex *zalpha, doublecomplex *zbeta,
doublecomplex *zvsl, int ldvsl, doublecomplex
*zvsr, int ldvsr, int *info)69
CALL ZGEGV (JOBVL, JOBVR, N, ZA, LDA, ZB, LDB,
ZALPHA, ZBETA, ZVL, LDVL, ZVR, LDVR,
ZWORK, LDWORK, DWORK2, INFO)74
void zgegv (char jobvl, char jobvr, int n, doublecomplex *za,
int lda, doublecomplex *zb, int ldb, doublecomplex
*zalpha, doublecomplex *zbeta, doublecomplex
*zvl, int ldvl, doublecomplex *zvr, int ldvr, int
*info)74
CALL ZGELQF (M, N, ZA, LDA, ZTAU, ZWORK,
LDWORK, INFO)79
void zgelf (int m, int n, doublecomplex *za, int lda,
doublecomplex *ztau, int *info)79
CALL ZGELS (TRANSA, M, N, NRHS, ZA, LDA, ZB, LDB,
ZWORK, LDWORK, INFO)82
void zgels (char trans, int m, int n, int nrhs, doublecomplex
*za, int lda, doublecomplex *zb, int ldb, int *info)82
CALL ZGELSS (M, N, NRHS, ZA, LDA, ZB, LDB, DSING,
DRCOND, IRANK, ZWORK, LDWORK,
DWORK2, INFO)86
void zgels (int m, int n, int nrhs, doublecomplex *za, int lda,
doublecomplex *zb, int ldb, double *dsing, double
rcond, int *rank, int *info)86
CALL ZGELSX (M, N, NRHS, ZA, LDA, ZB, LDB, JPIVOT,
DRCOND, IRANK, ZWORK, DWORK2, INFO)90
void zgelsx (int m, int n, int nrhs, doublecomplex *za, int lda,
doublecomplex *zb, int ldb, int *jpivot, double
rcond, int *rank, int *info)90
CALL ZGEMM (TRANSA, TRANSB, M, N, K, ZALPHA,
ZA, LDA, ZB, LDB, ZBETA, ZC, LDC)805
void zgemm (char transa, char transb, int m, int n, int k,
doublecomplex *zalpha, doublecomplex *za, int
lda, doublecomplex *zb, int ldb, doublecomplex
*zbeta, doublecomplex *zc, int ldc)805
CALL ZGEMV (TRANSA, M, N, ZALPHA, ZA, LDA, ZX,
INCX, ZBETA, ZY, INCY)727
void zgmv (char trans, int m, int n, doublecomplex *zalpha,
doublecomplex *za, int lda, doublecomplex *zx, int
incx, doublecomplex *zbeta, doublecomplex *zy, int
incy)727
CALL ZGEQLF (M, N, ZA, LDA, ZTAU, ZWORK,
LDWORK, INFO)94
void zgeqlf (int m, int n, doublecomplex *za, int lda,
doublecomplex *ztau, int *info)94
CALL ZGEQPF (M, N, ZA, LDA, JPIVOT, ZTAU, ZWORK,
DWORK2, INFO)97
void zgeqpf (int m, int n, doublecomplex *za, int lda, int
*jpivot, doublecomplex *ztau, int *info)97
CALL ZGEQRF (M, N, ZA, LDA, ZTAU, ZWORK,
LDWORK, INFO)100
void zgeqrf (int m, int n, doublecomplex *za, int lda,
doublecomplex *ztau, int *info)100
CALL ZGERC (M, N, ZALPHA, ZX, INCX, ZY, INCY, ZA,
LDA)730
void zgerc (int m, int n, doublecomplex *zalpha,
doublecomplex *zx, int incx, doublecomplex *zy,
int incy, doublecomplex *za, int lda)730
CALL ZGERFS (TRANSA, N, NRHS, ZA, LDA, ZAF, LDAF,
IPIVOT, ZB, LDB, ZX, LDX, DFERR, DBERR,
ZWORK, DWORK2, INFO)103
void zgerfs (char trans, int n, int nrhs, doublecomplex *za,
int lda, doublecomplex *zaf, int ldaf, int *ipivot,
doublecomplex *zb, int ldb, doublecomplex *zx, int
ldx, double *dferr, double *dberr, int *info)103
CALL ZGERQF (M, N, ZA, LDA, ZTAU, ZWORK,
LDWORK, INFO)108
void zgerqf (int m, int n, doublecomplex *za, int lda,
doublecomplex *ztau, int *info)108
CALL ZGERU (M, N, ZALPHA, ZX, INCX, ZY, INCY, ZA,
LDA)730
void zgeru (int m, int n, doublecomplex *zalpha,
doublecomplex *zx, int incx, doublecomplex *zy,
int incy, doublecomplex *za, int lda)730
CALL ZGESL (ZA, LDA, N, IPIVOT, ZB, JOB)572
void zgesl (doublecomplex *za, int lda, int n, int *ipivot,
doublecomplex *zb, int job)572

```

CALL ZGESV (N, NRHS, ZA, LDA, IPIVOT, ZB, LDB, INFO)111

void zgesv (int n, int nrhs, doublecomplex *za, int lda, int *ipivot, doublecomplex *zb, int ldb, int *info)111

CALL ZGESVD (JOBV, JOBT, M, N, ZA, LDA, DSING, ZU, LDU, ZVT, LDVT, ZWORK, LDWORK, DWORK2, INFO)114

void zgesvd (char jobv, char jobt, int m, int n, doublecomplex *za, int lda, double *dsing, doublecomplex *u, int ldu, doublecomplex *zvt, int ldvt, int *info)114

CALL ZGESVX (FACT, TRANSA, N, NRHS, ZA, LDA, ZAF, LDAF, IPIVOT, EQUED, ZROWSC, ZCOLSC, ZB, LDB, ZX, LDX, DRCOND, DFERR, DBERR, ZWORK, DWORK2, INFO)119

void zgesvx (char fact, char trans, int n, int nrhs, doublecomplex *za, int lda, doublecomplex *zaf, int ldaf, int *ipivot, char *equed, double *drowsc, double *dcolsc, doublecomplex *zb, int ldb, doublecomplex *zx, int ldx, double *drcond, double *dferr, double *dberr, int *info)119

CALL ZGETRF (M, N, ZA, LDA, IPIVOT, INFO)126

void zgetrf (int m, int n, doublecomplex *za, int lda, int *ipivot, int *info)126

CALL SGETRI (N, SA, LDA, IPIVOT, SWORK, LDWORK, INFO)129

CALL ZGETRI (N, ZA, LDA, IPIVOT, ZWORK, LDWORK, INFO)129

void zgetri (int n, doublecomplex *za, int lda, int *ipivot, int *info)129

CALL ZGETRS (TRANSA, N, NRHS, ZA, LDA, IPIVOT, ZB, LDB, INFO)132

void zgetrs (char trans, int n, int nrhs, doublecomplex *za, int lda, int *ipivot, doublecomplex *zb, int ldb, int *info)132

CALL ZGGGLM (N, M, P, ZA, LDA, ZB, LDB, ZD, ZX, ZY, ZWORK, LDWORK, INFO)135

void zggglm (int n, int m, int p, doublecomplex *za, int lda, doublecomplex *zb, int ldb, doublecomplex *zd, doublecomplex *zx, doublecomplex *zy, int *info)135

CALL ZGGLSE (M, N, P, ZA, LDA, ZB, LDB, ZC, ZD, ZX, ZWORK, LDWORK, INFO)139

void zgglse (int m, int n, int p, doublecomplex *za, int lda, doublecomplex *zb, int ldb, doublecomplex *zc, doublecomplex *zd, doublecomplex *zx, int *info)139

CALL ZGGSVD (JOBV, JOBT, M, N, P, K, L, ZA, LDA, ZB, LDB, DALPHA, DBETA, ZU, LDU, ZV, LDV, ZQ, LDQ, ZWORK, DWORK2, IWORK3, INFO)144

void zggsvd (char jobv, char jobt, char jobq, int m, int n, int p, int *k, int *l, doublecomplex *za, int lda, doublecomplex *zb, int ldb, double *dalpha, double *dbeta, doublecomplex *zu, int ldu, doublecomplex *zv, int ldv, doublecomplex *zq, int ldq, int *info)145

CALL ZGTCON (NORM, N, ZLOW, ZDIAG, ZUP1, ZUP2, IPIVOT, DANORM, DRCOND, ZWORK, INFO)149

void zgtcon (char norm, int n, doublecomplex *zlow, doublecomplex *zdiag, doublecomplex *zup1, doublecomplex *zup2, int *ipivot, double danorm, double *drcond, int *info)149

CALL ZGTRFS (TRANSA, N, NRHS, ZLOW, ZDIAG, ZUP, ZLOWF, ZDIAGF, ZUPF1, ZUPF2, IPIVOT, ZB, LDB, ZX, LDX, DFERR, DBERR, ZWORK, DWORK2, INFO)153

void zgtrfs (char trans, int n, int nrhs, doublecomplex *zlow, doublecomplex *zdiag, doublecomplex *zup, doublecomplex *zlowf, doublecomplex *zdiagf, doublecomplex *zupf1, doublecomplex *zupf2, int *ipivot, doublecomplex *zb, int ldb, doublecomplex *zx, int ldx, double *dferr, double *dberr, int *info)153

CALL ZGTSL (N, ZSUB, ZDIAG, ZSUPER, ZB, INFO)575

void zgtsl (int n, doublecomplex *zsub, doublecomplex *zdiag, doublecomplex *zsuper, doublecomplex *zb, int *info)575

CALL ZGTSV (N, NRHS, ZLOW, ZDIAG, ZUP, ZB, LDB, INFO)160

void zgtsv (int n, int nrhs, doublecomplex *zlow, doublecomplex *zdiag, doublecomplex *zup, doublecomplex *zb, int ldb, int *info)160

CALL ZGTSVX (FACT, TRANSA, N, NRHS, ZLOW, ZDIAG, ZUP, ZLOWF, ZDIAGF, ZUPF1, ZUPF2, IPIVOT, ZB, LDB, ZX, LDX, DRCOND, DFERR, DBERR, ZWORK, DWORK2, INFO)163


```

void zgtsvx (char fact, char trans, int n, int nrhs,
             doublecomplex *zlow, doublecomplex *zdiag,
             doublecomplex *zup, doublecomplex *zlowf,
             doublecomplex *zdiagf, doublecomplex *zupf1,
             doublecomplex *zupf2, int *ipivot, doublecomplex
             *zb, int ldb, doublecomplex *zx, int ldx, double
             *drcond, double *dferr, double *dberr, int *info)163
CALL ZGTTRF (N, ZLOW, ZDIAG, ZUP1, ZUP2, IPIVOT,
             INFO)169
void zgtrf (int n, doublecomplex *zlow, doublecomplex
            *zdiag, doublecomplex *zup1, doublecomplex
            *zup2, int *ipivot, int *info)169
CALL ZGTRRS (TRANSA, N, NRHS, ZLOW, ZDIAG, ZUP1,
             ZUP2, IPIVOT, ZB, LDB, INFO)173
void zgtrs (char trans, int n, int nrhs, doublecomplex *zlow,
            doublecomplex *zdiag, doublecomplex *zup1,
            doublecomplex *zup2, int *ipivot, doublecomplex
            *zb, int ldb, int *info)173
CALL ZHBEV (JOBZ, UPLO, N, NDIAG, ZA, LDA, DW, ZZ,
            LDZ, ZWORK, DWORK2, INFO)177
void zhbev (char jobz, char uplo, int n, int ndiag,
            doublecomplex *za, int lda, double *dw,
            doublecomplex *zz, int ldz, int *info)177
CALL ZHBEVX (JOBZ, RANGE, UPLO, N, NDIAG, ZA,
            LDA, ZQ, LDQ, DVL, DVU, IL, IU, DABTOL,
            NFOUND, DW, ZZ, LDZ, ZWORK, DWORK2,
            IWORK3, IFAIL, INFO)181
void zhbevz (char jobz, char range, char uplo, int n, int
            ndiag, doublecomplex *za, int lda, doublecomplex
            *zq, int ldq, double dvl, double dvu, int il, int iu,
            double dabtol, int *nfound, double *dw
            doublecomplex *zz, int ldz, int *ifail, int *info)181
CALL ZHBMV (UPLO, N, NDIAG, ZALPHA, ZA, LDA, ZX,
            INCX, ZBETA, ZY, INCY)734
void zhbmz (char uplo, int n, int ndiag, doublecomplex
            *zalpha, doublecomplex *za, int lda,
            doublecomplex *zx, int incx, doublecomplex
            *zbeta, doublecomplex *zy, int incy)734
CALL ZHECON (UPLO, N, ZA, LDA, IPIVOT, DANORM,
            DRCOND, ZWORK, INFO)186
void zhecon (char uplo, int n, doublecomplex *za, int lda, int
            *ipivot, double danorm, double *drcond, int *info)
            186
CALL ZHEEV (JOBZ, UPLO, N, ZA, LDA, DW, ZWORK,
            LDWORK, DWORK2, INFO)190
CALL ZHEEVX (JOBZ, RANGE, UPLO, N, ZA, LDA, DVL,
            DVU, IL, IU, DABTOL, NFOUND, DW, ZZ, LDZ,
            ZWORK, LDWORK, DWORK2, IWORK3, IFAIL,
            INFO)193
void zheev (char jobz, char uplo, int n, doublecomplex *za,
            int lda, double *dw, int *info)190
void zheevx (char jobz, char range, char uplo, int n,
            doublecomplex *za, int lda, double dvl, double
            dvu, int il, int iu, double dabtol, int *nfound, double
            *dw, doublecomplex *zz, int ldz, int *ifail, int *info)
            193
CALL ZHEGV (ITYPE, JOBZ, UPLO, N, ZA, LDA, ZB, LDB,
            DW, ZWORK, LDWORK, DWORK2, INFO)198
void zhegv (int itype, char jobz, char uplo, int n,
            doublecomplex *za, int lda, doublecomplex *zb, int
            ldb, double *dw, int *info)198
CALL ZHEMM (SIDE, UPLO, M, N, ZALPHA, ZA, LDA,
            ZB, LDB, ZBETA, ZC, LDC)809
void zhemm (char side, char uplo, int m, int n,
            doublecomplex *zalpha, doublecomplex *za, int
            lda, doublecomplex *b, int ldb, doublecomplex
            *zbeta, doublecomplex *zc, int ldc)809
CALL ZHEMV (UPLO, N, ZALPHA, ZA, LDA, ZX, INCX,
            ZBETA, ZY, INCY)738
void zhemv (char uplo, int n, doublecomplex *zalpha,
            doublecomplex *za, int lda, doublecomplex *zx, int
            incx, doublecomplex *zbeta, doublecomplex *zy, int
            incy)738
CALL ZHER (UPLO, N, DALPHA, ZX, INCX, ZA, LDA)742
void zher (char uplo, int n, double dalpha, doublecomplex
            *zx, int incx, doublecomplex *za, int lda)742
CALL ZHER2 (UPLO, N, ZALPHA, ZX, INCX, ZY, INCY,
            ZA, LDA)745
void zher2 (char uplo, int n, doublecomplex *zalpha,
            doublecomplex *zx, int incx, doublecomplex *zy,
            int incy, doublecomplex *za, int lda)745
void zher2k (char uplo, char trans, int n, int k,
            doublecomplex *zalpha, doublecomplex *za, int
            lda, doublecomplex *zb, int ldb, double dbeta,
            doublecomplex *zc, int ldc)815

```

CALL ZHERFS (UPLO, N, NRHS, ZA, LDA, ZAF, LDAF, IPIVOT, ZB, LDB, ZX, LDX, DFERR, DBERR, ZWORK, DWORK2, INFO)203

void zherfs (char uplo, int n, int nrhs, doublecomplex *za, int lda, doublecomplex *zaf, int ldaf, int *ipivot, doublecomplex *zb, int ldb, doublecomplex *zx, int ldx, double *dferr, double *dberr, int *info)203

CALL ZHERK (UPLO, TRANSA, N, K, DALPHA, ZA, LDA, DBETA, ZC, LDC)812

void zherk (char uplo, char trans, int n, int k, double dalpha, doublecomplex *za, int lda, double dbeta, doublecomplex *zc, int ldc)812

CALL ZHESV (UPLO, N, NRHS, ZA, LDA, IPIVOT, ZB, LDB, DWORK, LDWORK, INFO)209

void zhesv (char uplo, int n, int nrhs, doublecomplex *za, int lda, int *ipivot, doublecomplex *zb, int ldb, int *info)209

CALL ZHESVX (FACT, UPLO, N, NRHS, ZA, LDA, ZAF, LDAF, IPIVOT, ZB, LDB, ZX, LDX, DRCOND, DFERR, DBERR, ZWORK, LDWORK, DWORK2, INFO)213

void zhesvx (char fact, char uplo, int n, int nrhs, doublecomplex *za, int lda, doublecomplex *zaf, int ldaf, int *ipivot, doublecomplex *zb, int ldb, doublecomplex *zx, int ldx, double *drcond, double *dferr, double *dberr, int *info)213

CALL ZHETRF (UPLO, N, ZA, LDA, IPIVOT, ZWORK, LDWORK, INFO)218

void zhetrf (char uplo, int n, doublecomplex *za, int lda, int *ipivot, int *info)218

CALL ZHETRI (UPLO, N, ZA, LDA, IPIVOT, ZWORK, INFO)222

void zhetri (char uplo, int n, doublecomplex *za, int lda, int *ipivot, int *info)222

CALL ZHETRS (UPLO, N, NRHS, ZA, LDA, IPIVOT, ZB, LDB, INFO)225

void zhetrs (char uplo, int n, int nrhs, doublecomplex *za, int lda, int *ipivot, doublecomplex *zb, int ldb, int *info)225

CALL ZHICO (ZA, LDA, N, IPIVOT, DRCOND, ZWORK)578

void zhico (doublecomplex *za, int lda, int n, int *ipivot, double *drcond)578

CALL ZHIDI (ZA, LDA, N, IPIVOT, DDET, INERT, ZWORK, JOB)581

void zhidi (doublecomplex *za, int lda, int n, int *ipivot, double *ddet, int *inert, int job)581

CALL ZHIFA (ZA, LDA, N, IPIVOT, INFO)584

void zhifa (doublecomplex *za, int lda, int n, int *ipivot, int *info)584

CALL ZHISL (ZA, LDA, N, IPIVOT, ZB)587

void zhisl (doublecomplex *za, int lda, int n, int *ipivot, doublecomplex *zb)587

CALL ZHPCO (ZA, N, IPIVOT, DRCOND, ZWORK)590

void zhpc (doublecomplex *za, int n, int *ipivot, double *drcond)590

CALL ZHPCON (UPLO, N, ZA, IPIVOT, DANORM, DRCOND, ZWORK, INFO)229

void zhpc (char uplo, int n, doublecomplex *za, int *ipivot, double danorm, double *drcond, int *info)229

CALL ZHPDI (ZA, N, IPIVOT, DDET, INERT, ZWORK, JOB)593

void zhpdi (doublecomplex *za, int n, int *ipivot, double *ddet, int *inert, int job)593

CALL ZHPEV (JOBZ, UPLO, N, ZA, DW, ZZ, LDZ, ZWORK, DWORK2, INFO)233

void zhpev (char jobz, char uplo, int n, doublecomplex *za, double *dw, doublecomplex *zz, int ldz, int *info)233

CALL ZHPEVX (JOBZ, RANGE, UPLO, N, ZA, DVL, DVU, IL, IU, DABTOL, NFOUND, DW, ZZ, LDZ, ZWORK, DWORK2, IWORK3, IFAIL, INFO)236

void zhpevx (char jobz, char range, char uplo, int n, doublecomplex *za, double dvl, double dvu, int il, int iu, double dabt, int *nfound, double *dw, doublecomplex *zz, int ldz, int *ifail, int *info)236

CALL ZHPFA (ZA, N, IPIVOT, INFO)596

void zhpfa (doublecomplex *za, int n, int *ipivot, int *info)596

CALL ZHPGV (ITYPE, JOBZ, UPLO, N, ZA, ZB, DW, ZZ, LDZ, ZWORK, DWORK2, INFO)240

void zhp (int itype, char jobz, char uplo, int n, doublecomplex *za, doublecomplex *zb, double *dw, doublecomplex *zz, int ldz, int *info)240

CALL ZHPMV (UPLO, N, ZALPHA, ZA, ZX, INCX, ZBETA, ZY, INCY)748
void zhpmv (char uplo, int n, doublecomplex *zalpha, doublecomplex *za, doublecomplex *zx, int incx, doublecomplex *zbeta, doublecomplex *zy, int incy)748
CALL ZHPR (UPLO, N, DALPHA, ZX, INCX, ZA)751
void zhpr (char uplo, int n, double dalpha, doublecomplex *zx, int incx, doublecomplex *za)751
CALL ZHPR2 (UPLO, N, ZALPHA, ZX, INCX, ZY, INCY, ZA)754
void zhpr2 (char uplo, int n, doublecomplex *zalpha, doublecomplex *zx, int incx, doublecomplex *zy, int incy, doublecomplex *za)754
CALL ZHPRFS (UPLO, N, NRHS, ZA, ZAF, IPIVOT, ZB, LDB, ZX, LDX, DFERR, DBERR, ZWORK, DWORK2, INFO)244
void zhprfs (char uplo, int n, int nrhs, doublecomplex *za, doublecomplex *zaf, int *ipivot, doublecomplex *zb, int ldb, doublecomplex *zx, int ldx, double *dferr, double *dberr, int *info)244
CALL ZHPSL (ZA, N, IPIVOT, ZB)598
void zhpsl (doublecomplex *za, int n, int *ipivot, doublecomplex *zb)598
CALL ZHPSV (UPLO, N, NRHS, ZA, IPIVOT, ZB, LDB, INFO)250
void zhpsv (char uplo, int n, int nrhs, doublecomplex *za, int *ipivot, doublecomplex *zb, int ldb, int *info)250
CALL ZHPSVX (FACT, UPLO, N, NRHS, ZA, ZAF, IPIVOT, ZB, LDB, ZX, LDX, DRCOND, DFERR, DBERR, ZWORK, DWORK2, INFO)253
void zhpsvx (char fact, char uplo, int n, int nrhs, doublecomplex *za, doublecomplex *zaf, int *ipivot, doublecomplex *zb, int ldb, doublecomplex *zx, int ldx, double *drcond, double *dferr, double *dberr, int *info)253
CALL ZHPTRF (UPLO, N, ZA, IPIVOT, INFO)258
void zhptrf (char uplo, int n, doublecomplex *za, int *ipivot, int *info)258
CALL ZHPTRI (UPLO, N, ZA, IPIVOT, ZWORK, INFO)262
void zhptri (char uplo, int n, doublecomplex *za, int *ipivot, int *info)262
CALL ZHPTRS (UPLO, N, NRHS, ZA, IPIVOT, ZB, LDB, INFO)265
void zhptrs (char uplo, int n, int nrhs, doublecomplex *za, int *ipivot, doublecomplex *zb, int ldb, int *info)265
CALL ZPBCO (ZA, LDA, N, NDIAG, DRCOND, ZWORK, INFO)601
void zpbc (doublecomplex *za, int lda, int n, int ndiag, double *drcond, int *info)601
CALL ZPBCON (UPLO, N, NDIAG, ZA, LDA, DANORM, DRCOND, ZWORK, DWORK2, INFO)281
void zpbc (char uplo, int n, int ndiag, doublecomplex *za, int lda, double danorm, double *drcond, int *info)281
CALL ZPBDI (ZA, LDA, N, NDIAG, DDET)604
void zpbd (doublecomplex *za, int lda, int n, int ndiag, double *ddet)604
CALL ZPBEQU (UPLO, N, NDIAG, ZA, LDA, DSCALE, DSCOND, DAMAX, INFO)284
void zpbequ (char uplo, int n, int ndiag, doublecomplex *za, int lda, double *dscale, double *dscond, double *damax, int *info)284
CALL ZPBFA (ZA, LDA, N, NDIAG, INFO)607
void zpbf (doublecomplex *za, int lda, int n, int ndiag, int *info)607
CALL ZPBRFS (UPLO, N, NDIAG, NRHS, ZA, LDA, ZAF, LDAF, ZB, LDB, ZX, LDX, DFERR, DBERR, ZWORK, DWORK2, INFO)288
void zpbrfs (char uplo, int n, int ndiag, int nrhs, doublecomplex *za, int lda, doublecomplex *zaf, int ldaf, doublecomplex *zb, int ldb, doublecomplex *zx, int ldx, double *dferr, double *dberr, int *info)288
CALL ZPBSL (ZA, LDA, N, NDIAG, ZB)610
void zpbsl (doublecomplex *za, int lda, int n, int ndiag, doublecomplex *zb)610
CALL ZPBSV (UPLO, N, NDIAG, NRHS, ZA, LDA, ZB, LDB, INFO)294
void zpbsv (char uplo, int n, int ndiag, int nrhs, doublecomplex *za, int lda, doublecomplex *zb, int ldb, int *info)294

CALL ZPBSVX (FACT, UPLO, N, NDIAG, NRHS, ZA, LDA, ZAF, LDAF, EQUED, DSCALE, ZB, LDB, ZX, LDX, DRCOND, DFERR, DBERR, ZWORK, DWORK2, INFO)297

void zpbsvx (char fact, char uplo, int n, int ndiag, int nrhs, doublecomplex *za, int lda, doublecomplex *zaf, int ldaf, char *equed, double *dscale, doublecomplex *zb, int ldb, doublecomplex *zx, int ldx, double *drcond, double *dferr, double *dberr, int *info)297

CALL ZPBTRF (UPLO, N, NDIAG, ZA, LDA, INFO)303

void zpbtrf (char uplo, int n, int ndiag, doublecomplex *za, int lda, int *info)303

CALL ZPBTRS (UPLO, N, NDIAG, NRHS, ZA, LDA, ZB, LDB, INFO)307

void zpbtrs (char uplo, int n, int ndiag, int nrhs, doublecomplex *za, int lda, doublecomplex *zb, int ldb, int *info)307

CALL ZPOCO (ZA, LDA, N, DRCOND, ZWORK, INFO)613

void zpoco (doublecomplex *za, int lda, int n, double *drcond, int *info)613

CALL ZPOCON (UPLO, N, ZA, LDA, DANORM, DRCOND, ZWORK, DWORK2, INFO)311

void zpocon (char uplo, int n, doublecomplex *za, int lda, double danorm, double *drcond, int *info)311

CALL ZPODI (ZA, LDA, N, DDET, JOB)616

void zpodi (doublecomplex *za, int lda, int n, double *ddet, int job)616

CALL ZPOEQU (N, ZA, LDA, DSCALE, DSCOND, DAMAX, INFO)314

void zpoequ (int n, doublecomplex *za, int lda, double *dscale, double *dscond, double *damax, int *info)314

CALL ZPOFA (ZA, LDA, N, INFO)619

void zpofa (doublecomplex *za, int lda, int n, int *info)619

CALL ZPORFS (UPLO, N, NRHS, ZA, LDA, ZAF, LDAF, ZB, LDB, ZX, LDX, DFERR, DBERR, ZWORK, DWORK2, INFO)318

void zporfs (char uplo, int n, int nrhs, doublecomplex *za, int lda, doublecomplex *zaf, int ldaf, doublecomplex *zb, int ldb, doublecomplex *zx, int ldx, double *dferr, double *dberr, int *info)318

CALL ZPOSL (ZA, LDA, N, ZB)622

void zposl (doublecomplex *za, int lda, int n, doublecomplex *zb)622

CALL ZPOSV (UPLO, N, NRHS, ZA, LDA, ZB, LDB, INFO)323

void zposv (char uplo, int n, int nrhs, doublecomplex *za, int lda, doublecomplex *zb, int ldb, int *info)323

CALL ZPOSVX (FACT, UPLO, N, NRHS, ZA, LDA, ZAF, LDAF, EQUED, DSCALE, ZB, LDB, ZX, LDX, DRCOND, DFERR, DBERR, ZWORK, DWORK2, INFO)326

void zposvx (char fact, char uplo, int n, int nrhs, doublecomplex *za, int lda, doublecomplex *zaf, int ldaf, char *equed, double *dscale, doublecomplex *zb, int ldb, doublecomplex *zx, int ldx, double *drcond, double *dferr, double *dberr, int *info)326

CALL ZPOTRF (UPLO, N, ZA, LDA, INFO)332

void zpotrf (char uplo, int n, doublecomplex *za, int lda, int *info)332

CALL ZPOTRI (UPLO, N, ZA, LDA, INFO)335

void zpotri (char uplo, int n, doublecomplex *za, int lda, int *info)335

CALL ZPOTRS (UPLO, N, NRHS, ZA, LDA, ZB, LDB, INFO)338

void zpotrs (char uplo, int n, int nrhs, doublecomplex *za, int lda, doublecomplex *zb, int ldb, int *info)338

CALL ZPPCO (ZA, N, DRCOND, ZWORK, INFO)625

void zppco (doublecomplex *za, int n, double *drcond, int *info)625

CALL ZPPCON (UPLO, N, ZA, DANORM, DRCOND, ZWORK, DWORK2, INFO)342

void zppcon (char uplo, int n, doublecomplex *za, double danorm, double *drcond, int *info)342

CALL ZPPDI (ZA, N, DDET, JOB)628

void zppdi (doublecomplex *za, int n, double *ddet, int job)628

CALL ZPPEQU (UPLO, N, ZA, DSCALE, DSCOND, DAMAX, INFO)345

void zppequ (char uplo, int n, doublecomplex *za, double *dscale, double *scond, double *damax, int *info)345

CALL ZPPFA (ZA, N, INFO)631

void zppfa (doublecomplex *za, int n, int *info)631

CALL ZPPRFS (UPLO, N, NRHS, ZA, ZAF, ZB, LDB, ZX,
LDB, DFERR, DBERR, ZWORK, DWORK2, INFO)
349

void zpprfs (char uplo, int n, int nrhs, doublecomplex *za,
doublecomplex *zaf, doublecomplex *zb, int ldb,
doublecomplex *zx, int ldx, double *dferr, double
*dberr, int *info)349

CALL ZPPSL (ZA, N, ZB)634

CALL ZSPSL (ZA, N, IPIVOT, ZB)667

void zppsl (doublecomplex *za, int n, doublecomplex *zb)
634

CALL ZPPSV (UPLO, N, NRHS, ZA, ZB, LDB, INFO)354

void zppsv (char uplo, int n, int nrhs, doublecomplex *za,
doublecomplex *zb, int ldb, int *info)354

CALL ZPPSVX (FACT, UPLO, N, NRHS, ZA, ZAF, EQUED,
DSCALE, ZB, LDB, ZX, LDX, DRCOND, DFERR,
DBERR, ZWORK, DWORK2, INFO)357

void zppsvx (char fact, char uplo, int n, int nrhs,
doublecomplex *za, doublecomplex *zaf, char
*equed, double *dscale, doublecomplex *zb, int ldb,
doublecomplex *zx, int ldx, double *drcond, double
*dferr, double *dberr, int *info)357

CALL ZPPTRF (UPLO, N, ZA, INFO)363

void zpptrf (char uplo, int n, doublecomplex *za, int *info)
363

CALL ZPPTRI (UPLO, N, ZA, INFO)366

void zpptri (char uplo, int n, doublecomplex *za, int *info)
366

CALL ZPPTRS (UPLO, N, NRHS, ZA, ZB, LDB, INFO)369

void zpptrs (char uplo, int n, int nrhs, doublecomplex *za,
doublecomplex *zb, int ldb, int *info)369

CALL ZPTCON (N, DDIAG, ZOFFD, DANORM, DRCOND,
DWORK, INFO)372

void zptcon (int n, double *ddiag, doublecomplex *zoffd,
double danorm, double *drcond, int *info)372

CALL ZPTRFS (UPLO, N, NRHS, DDIAG, ZOFFD,
DDIAGF, ZOFFDF, ZB, LDB, ZX, LDX, DFERR,
DBERR, ZWORK, DWORK2, INFO)375

void zptrfs (char uplo, int n, int nrhs, double *ddiag,
doublecomplex *zoffd, double *ddiagf,
doublecomplex *zoffdf, doublecomplex *zb, int ldb,
doublecomplex *zx, int ldx, double *dferr, double
*dberr, int *info)375

CALL ZPTSL (N, ZDIAG, ZOFFD, ZB)637

void zptsl (int n, doublecomplex *zdiag, doublecomplex
*zoffd, doublecomplex *zb)637

CALL ZPTSV (N, NRHS, DDIAG, ZSUB, ZB, LDB, INFO)381

void zptsv (int n, int nrhs, double *ddiag, doublecomplex
*dsub, doublecomplex *zb, int ldb, int *info)381

CALL ZPTSVX (FACT, N, NRHS, DDIAG, ZSUB, DDIAGF,
ZSUBF, ZB, LDB, ZX, LDX, DRCOND, DFERR,
DBERR, ZWORK, DWORK2, INFO)384

void zptsvx (char fact, int n, int nrhs, double *ddiag,
doublecomplex *zsub, double *ddiagf,
doublecomplex *zsubf, doublecomplex *zb, int ldb,
doublecomplex *zx, int ldx, double *drcond, double
*dferr, double *dberr, int *info)384

CALL ZPTTRF (N, DDIAG, ZOFFD, INFO)389

void zpttrf (int n, double *ddiag, doublecomplex *zoffd, int
*info)389

CALL ZPTTRS (N, NRHS, DDIAG, ZOFFD, ZB, LDB, INFO)
392

void zpttrs (char uplo, int n, int nrhs, double *ddiag,
doublecomplex *zoffd, doublecomplex *zb, int ldb,
int *info)392

CALL ZQRDC (ZA, LDA, N, P, ZQRAUX, IPIVOT, ZWORK,
JOB)639

void zqrdc (doublecomplex *za, int lda, int n, int p,
doublecomplex *zqraux, int *ipivot, int job)639

CALL ZQRSL (ZA, LDA, N, K, ZQRAUX, CY, CQY, ZQTY,
ZB, ZRESID, ZAB, JOB, INFO)642

void zqrsl (doublecomplex *za, int lda, int n, int k,
doublecomplex *zqraux, doublecomplex *zy,
doublecomplex *zqy, doublecomplex *zqty,
doublecomplex *zb, doublecomplex *zresid,
doublecomplex *zab, int job, int *info)642

CALL ZSCAL (N, ZALPHA, ZY, INCY)714

void zscal (int n, doublecomplex *za, doublecomplex *zx, int
incx)714

CALL ZSICO (ZA, LDA, N, IPIVOT, DRCOND, ZWORK)
646

void zsico (doublecomplex *za, int lda, int n, int *ipivot,
double *drcond)646

CALL ZSIDI (ZA, LDA, N, IPIVOT, ZDET, ZWORK, JOB)649

void zsidi (doublecomplex *za, int lda, int n, int *ipivot,
doublecomplex *zdet, int job)649

CALL ZSIFA (ZA, LDA, N, IPIVOT, INFO)652
void zsifa (doublecomplex *za, int lda, int n, int *ipivot, int *info)652

CALL ZSISL (ZA, LDA, N, IPIVOT, ZB)655
void zsisl (doublecomplex *za, int lda, int n, int *ipivot, doublecomplex *zb)655

CALL ZSPCO (ZA, N, IPIVOT, DRCOND, ZWORK)658
void zspco (doublecomplex *za, int n, int *ipivot, double *drcond)658

CALL ZSPCON (UPLO, N, ZA, IPIVOT, DANORM, DRCOND, ZWORK, INFO)405
void zspcon (char uplo, int n, doublecomplex *za, int *ipivot, double danorm, double *drcond, int *info)405

CALL ZSPDI (ZA, N, IPIVOT, ZDET, ZWORK, JOB)661
void zspdi (doublecomplex *za, int n, int *ipivot, doublecomplex *zdet, int job)661

CALL ZSPFA (ZA, N, IPIVOT, INFO)664
void zspfa (doublecomplex *za, int n, int *ipivot, int *info)664

CALL ZSPRFS (UPLO, N, NRHS, ZA, ZAF, IPIVOT, ZB, LDB, ZX, LDX, DFERR, DBERR, ZWORK, DWORK2, INFO)419
void zsprfs (char uplo, int n, int nrhs, doublecomplex *za, doublecomplex *zaf, int *ipivot, doublecomplex *zb, int ldb, doublecomplex *zx, int ldx, double *dferr, double *dberr, int *info)419

void zspsl (doublecomplex *za, int n, int *ipivot, doublecomplex *zb)667

CALL ZSPSV (UPLO, N, NRHS, ZA, IPIVOT, ZB, LDB, INFO)425
void zpsv (char uplo, int n, int nrhs, doublecomplex *za, int *ipivot, doublecomplex *zb, int ldb, int *info)425

CALL ZSPSVX (FACT, UPLO, N, NRHS, ZA, ZAF, IPIVOT, ZB, LDB, ZX, LDX, DRCOND, DFERR, DBERR, ZWORK, DWORK2, INFO)428
void zpsvx (char fact, char uplo, int n, int nrhs, doublecomplex *za, doublecomplex *zaf, int *ipivot, doublecomplex *zb, int ldb, doublecomplex *zx, int ldx, double *drcond, double *dferr, double *dberr, int *info)428

CALL ZSPTRF (UPLO, N, ZA, IPIVOT, INFO)433
void zsptrf (char uplo, int n, doublecomplex *za, int *ipivot, int *info)433

CALL ZSPTRI (UPLO, N, ZA, IPIVOT, ZWORK, INFO)436
void zsptri (char uplo, int n, doublecomplex *za, int *ipivot, int *info)436

CALL ZSPTRS (UPLO, N, NRHS, ZA, IPIVOT, ZB, LDB, INFO)439
void zsptrs (char uplo, int n, int nrhs, doublecomplex *za, int *ipivot, doublecomplex *zb, int ldb, int *info)439

CALL ZSVDC (ZA, LDA, N, P, ZSVALS, ZE, ZLSVEC, LDL, ZRSVEC, LDR, ZWORK, JOB, INFO)669
void zsvdc (doublecomplex *za, int lda, int n, int p, doublecomplex *zsvals, doublecomplex *ze, doublecomplex *zlsvec, int ldl, doublecomplex *zrsvec, int ldr, int job, int *info)669

CALL ZSWAP (N, ZX, INCX, ZY, INCY)716
void zswap (int n, doublecomplex *zx, int incx, doublecomplex *zy, int incy)716

CALL ZSYCON (UPLO, N, ZA, LDA, IPIVOT, DANORM, DRCOND, ZWORK, INFO)450
void zsycon (char uplo, int n, doublecomplex *za, int lda, int *ipivot, double danorm, double *drcond, int *info)450

CALL ZSYMM (SIDE, UPLO, M, N, ZALPHA, ZA, LDA, ZB, LDB, ZBETA, ZC, LDC)819
void zsymm (char side, char uplo, int m, int n, doublecomplex *zalpha, doublecomplex *za, int lda, doublecomplex *b, int ldb, doublecomplex *zbeta, doublecomplex *zc, int ldc)819

CALL ZSYR2K (UPLO, TRANSA, N, K, ZALPHA, ZA, LDA, ZB, LDB, ZBETA, ZC, LDC)826
void zsyr2k (char uplo, char trans, int n, int k, doublecomplex *zalpha, doublecomplex *za, int lda, doublecomplex *zb, int ldb, doublecomplex *zbeta, doublecomplex *zc, int ldc)826

CALL ZSYRFS (UPLO, N, NRHS, ZA, LDA, ZAF, LDAF, IPIVOT, ZB, LDB, ZX, LDX, DFERR, DBERR, ZWORK, DWORK2, INFO)465
void zsyrfs (char uplo, int n, int nrhs, doublecomplex *za, int lda, doublecomplex *zaf, int ldaf, int *ipivot, doublecomplex *zb, int ldb, doublecomplex *zx, int ldx, double *dferr, double *dberr, int *info)465

CALL ZSYRK (UPLO, TRANSA, N, K, ZALPHA, ZA, LDA, ZBETA, ZC, LDC)823


```
void zsyrrk (char uplo, char trans, int n, int k, doublecomplex
    *zalpha, doublecomplex *za, int lda,
    doublecomplex *zbeta, doublecomplex *zc, int ldc)
823
CALL ZSYSV (UPLO, N, NRHS, ZA, LDA, IPIVOT, ZB,
    LDB, ZWORK, LDWORK, INFO)471
void zsysv (char uplo, int n, int nrhs, doublecomplex *za, int
    lda, int *ipivot, doublecomplex *zb, int ldb, int
    *info)471
CALL ZSYSVX (FACT, UPLO, N, NRHS, ZA, LDA, ZAF,
    LDAF, IPIVOT, ZB, LDB, ZX, LDX, DRCOND,
    DFERR, DBERR, ZWORK, LDWORK, DWORK2,
    INFO)474
void zsysvx (char fact, char uplo, int n, int nrhs,
    doublecomplex *za, int lda, doublecomplex *zaf, int
    ldaf, int *ipivot, doublecomplex *zb, int ldb,
    doublecomplex *zx, int ldx, double *drcond, double
    *dferr, double *dberr, int *info)474
CALL ZSYTRF (UPLO, N, ZA, LDA, IPIVOT, ZWORK,
    LDWORK, INFO)479
void zsytrf (char uplo, int n, doublecomplex *za, int lda, int
    *ipivot, int *info)479
CALL ZSYTRI (UPLO, N, ZA, LDA, IPIVOT, ZWORK,
    INFO)482
void zsytri (char uplo, int n, doublecomplex *za, int lda, int
    *ipivot, int *info)482
CALL ZSYTRS (UPLO, N, NRHS, ZA, LDA, IPIVOT, ZB,
    LDB, INFO)485
void zsytrs (char uplo, int n, int nrhs, doublecomplex *za, int
    lda, int *ipivot, doublecomplex *zb, int ldb, int
    *info)485
CALL ZTBCON (NORM, UPLO, DIAG, N, NDIAG, ZA,
    LDA, DRCOND, ZWORK, DWORK2, INFO)488
void ztbcon (char norm, char uplo, char diag, int n, int ndiag,
    doublecomplex *za, int lda, double *drcond, int
    *info)488
CALL ZTBMV (UPLO, TRANSA, DIAG, N, NDIAG, ZA,
    LDA, ZY, INCY)779
void ztbmv (char uplo, char trans, char diag, int n, int ndiag,
    doublecomplex *za, int lda, doublecomplex *zx, int
    incx)779
CALL ZTBRFS (UPLO, TRANSA, DIAG, N, NDIAG, NRHS,
    ZA, LDA, ZB, LDB, ZX, LDX, DFERR, DBERR,
    ZWORK, DWORK2, INFO)492
void ztbrfs (char uplo, char trans, char diag, int n, int ndiag,
    int nrhs, doublecomplex *za, int lda,
    doublecomplex *zb, int ldb, doublecomplex *zx, int
    ldx, double *dferr, double *dberr, int *info)492
CALL ZTBSV (UPLO, TRANSA, DIAG, N, NDIAG, ZA,
    LDA, ZY, INCY)783
void ztbsv (char uplo, char trans, char diag, int n, int ndiag,
    doublecomplex *za, int lda, doublecomplex *zx, int
    incx)783
CALL ZTBTRS (UPLO, TRANSA, DIAG, N, NDIAG, NRHS,
    ZA, LDA, ZB, LDB, INFO)497
void ztbtrs (char uplo, char trans, char diag, int n, int ndiag,
    int nrhs, doublecomplex)497
CALL ZTPCON (NORM, UPLO, DIAG, N, ZA, DRCOND,
    ZWORK, DWORK2, INFO)501
void ztpcon (char norm, char uplo, char diag, int n,
    doublecomplex *za, double *drcond, int *info)501
CALL ZTPMV (UPLO, TRANSA, DIAG, N, ZA, ZY, INCY)
787
void ztpmv (char uplo, char trans, char diag, int n,
    doublecomplex *za, doublecomplex *zx, int incx)
787
CALL ZTPRFS (UPLO, TRANSA, DIAG, N, NRHS, ZA, ZB,
    LDB, ZX, LDX, DFERR, DBERR, ZWORK,
    DWORK2, INFO)504
void ztparfs (char uplo, char trans, char diag, int n, int nrhs,
    doublecomplex *za, doublecomplex *zb, int ldb,
    doublecomplex *zx, int ldx, double *dferr, double
    *dberr, int *info)504
CALL ZTPSV (UPLO, TRANSA, DIAG, N, ZA, ZY, INCY)
791
void ztpsv (char uplo, char trans, char diag, int n,
    doublecomplex *za, doublecomplex *zx, int incx)
791
CALL ZTPTRI (UPLO, DIAG, N, ZA, INFO)509
void ztptri (char uplo, char diag, int n, doublecomplex *za,
    int *info)509
CALL ZTPTRS (UPLO, TRANSA, DIAG, N, NRHS, ZA, ZB,
    LDB, INFO)512
```

≡ A

void ztptrs (char uplo, char trans, char diag, int n, int nrhs,
doublecomplex *za, doublecomplex *zb, int ldb, int
*info)512

CALL ZTRANS (PLACE, ZSCALE, ZSOURCE, M, N,
ZDEST)842

void ztrans (char place, doublecomplex zscale,
doublecomplex *zsource, int m, int n,
doublecomplex *zdest)842

CALL ZTRCO (ZA, LDA, N, DRCOND, ZWORK, JOB)672

void ztrco (doublecomplex *za, int lda, int n, double
*drcond, int job)672

CALL ZTRCON (NORM, UPLO, DIAG, N, ZA, LDA,
DRCOND, ZWORK, DWORK2, INFO)515

void ztrcon (char norm, char uplo, char diag, int n,
doublecomplex *za, int lda, double *drcond, int
*info)515

CALL ZTRDI (ZA, LDA, N, ZDET, JOB, INFO)675

void ztrdi (doublecomplex *za, int lda, int n, doublecomplex
*zdet, int job, int *info)675

CALL ZTRMM (SIDE, UPLO, TRANSA, DIAG, M, N,
ZALPHA, ZA, LDA, ZB, LDB)830

void ztrmm (char side, char uplo, char transa, char diag, int
m, int n, doublecomplex *zalpha, doublecomplex
*za, int lda, doublecomplex *zb, int ldb)830

CALL ZTRMV (UPLO, TRANSA, DIAG, N, ZA, LDA, ZY,
INCY)795

void ztrmv (char uplo, char trans, char diag, int n,
doublecomplex *za, int lda, doublecomplex *zx, int
incx)795

CALL ZTRRFS (UPLO, TRANSA, DIAG, N, NRHS, ZA,
LDA, ZB, LDB, ZX, LDX, DFERR, DBERR,
ZWORK, DWORK2, INFO)518

void ztrrfs (char uplo, char trans, char diag, int n, int nrhs,
doublecomplex *za, int lda, doublecomplex *zb, int
ldb, doublecomplex *zx, int ldx, double *dferr,
double *dberr, int *info)518

CALL ZTRSL (ZA, LDA, N, ZB, JOB, INFO)678

void ztrsl (doublecomplex *za, int lda, int n, doublecomplex
*zb, int job, int *info)678

CALL ZTRSM (SIDE, UPLO, TRANSA, DIAG, M, N,
ZALPHA, ZA, LDA, ZB, LDB)834

void ztrsm (char side, char uplo, char transa, char diag, int
m, int n, doublecomplex *zalpha, doublecomplex
*za, int lda, doublecomplex *zb, int ldb)834

CALL ZTRSV (UPLO, TRANSA, DIAG, N, ZA, LDA, ZY,
INCY)798

void ztrsv (char uplo, char trans, char diag, int n,
doublecomplex *za, int lda, doublecomplex *zx, int
incx)798

CALL ZTRTRI (UPLO, DIAG, N, ZA, LDA, INFO)523

void ztrtri (char uplo, char diag, int n, doublecomplex *za,
int lda, int *info)523

CALL ZTRTRS (UPLO, TRANSA, DIAG, N, NRHS, ZA,
LDA, ZB, LDB, INFO)526

void ztrtrs (char uplo, char trans, char diag, int n, int nrhs,
doublecomplex *za, int lda, doublecomplex *zb, int
ldb, int *info)526

Copyright 1996 Sun Microsystems Inc., 2550 Garcia Avenue, Mountain View, Californie 94043-1100, U.S.A. Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou de sa documentation associée ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a.

Des parties de ce produit pourront être dérivées du système UNIX[®] licencié par Novell, Inc. et du système Berkeley 4.3 BSD licencié par l'Université de Californie. UNIX est une marque enregistrée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company Ltd. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Sun, Sun Microsystems, le logo Sun, SunSoft, Solaris, Sun WorkShop, Sun Performance WorkShop, Sun Performance Library, Sun Visual WorkShop, et Sun WorkShop TeamWare, sont des marques déposées ou enregistrées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC, utilisées sous licence, sont des marques déposées ou enregistrées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

Les interfaces d'utilisation graphique OPEN LOOK[®] et Sun[™] ont été développées par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant aussi les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" SANS GARANTIE D'AUCUNE SORTE, NI EXPRESSE NI IMPLICITE, Y COMPRIS, ET SANS QUE CETTE LISTE NE SOIT LIMITATIVE, DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DES PRODUITS A RÉPONDRE A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ILS NE SOIENT PAS CONTREFAISANTS DE PRODUITS DE TIERS.

