NAG Fortran Library Manual
Mark 19

Volume 7

F07 – F08B

F07 – Linear Equations (LAPACK)
F08B – Least-squares and Eigenvalue Problems (LAPACK) (cont'd in Volume 8)

NAG®

NAG Fortran Library Manual, Mark 19

September 1999

ISBN 1-85206-169-3

NAG Ltd
Wilkinson House
Jordan Hill Road
Oxford
OX2 8DR
United Kingdom

Tel: +44 (0)1865 511245
Fax: +44 (0)1865 310139

Nihon NAG KK
Nagashima Building 2F
2-24-3 Higashi
Shibuya-ku
Tokyo
Japan

Tel: +81 (0)3 5485 2901
Fax: +81 (0)3 5485 2903

NAG GmbH
Schleißheimerstraße 5
85748 Garching
Deutschland

Tel: +49 (0)89 3207395
Fax: +49 (0)89 3207396

NAG Inc
1400 Opus Place, Suite 200
Downers Grove, IL 60515-5702
USA

Tel: +1 630 971 2337
Fax: +1 630 971 2706

NAG also has a number of distributors throughout the world. Please contact NAG for further details.

[*NP3390/19*]

Chapter F07 – Linear Equations (LAPACK)

Note. Please refer to the Users' Note for your implementation to check that a routine is available.

Routine Name	Mark of Introduction	Purpose
F07ADF	15	(SGETRF/DGETRF) *LU* factorization of real m by n matrix
F07AEF	15	(SGETRS/DGETRS) Solution of real system of linear equations, multiple right-hand sides, matrix already factorized by F07ADF
F07AGF	15	(SGECON/DGECON) Estimate condition number of real matrix, matrix already factorized by F07ADF
F07AHF	15	(SGERFS/DGERFS) Refined solution with error bounds of real system of linear equations, multiple right-hand sides
F07AJF	15	(SGETRI/DGETRI) Inverse of real matrix, matrix already factorized by F07ADF
F07ARF	15	(CGETRF/ZGETRF) *LU* factorization of complex m by n matrix
F07ASF	15	(CGETRS/ZGETRS) Solution of complex system of linear equations, multiple right-hand sides, matrix already factorized by F07ARF
F07AUF	15	(CGECON/ZGECON) Estimate condition number of complex matrix, matrix already factorized by F07ARF
F07AVF	15	(CGERFS/ZGERFS) Refined solution with error bounds of complex system of linear equations, multiple right-hand sides
F07AWF	15	(CGETRI/ZGETRI) Inverse of complex matrix, matrix already factorized by F07ARF
F07BDF	15	(SGBTRF/DGBTRF) *LU* factorization of real m by n band matrix
F07BEF	15	(SGBTRS/DGBTRS) Solution of real band system of linear equations, multiple right-hand sides, matrix already factorized by F07BDF
F07BGF	15	(SGBCON/DGBCON) Estimate condition number of real band matrix, matrix already factorized by F07BDF
F07BHF	15	(SGBRFS/DGBRFS) Refined solution with error bounds of real band system of linear equations, multiple right-hand sides
F07BRF	15	(CGBTRF/ZGBTRF) *LU* factorization of complex m by n band matrix
F07BSF	15	(CGBTRS/ZGBTRS) Solution of complex band system of linear equations, multiple right-hand sides, matrix already factorized by F07BRF
F07BUF	15	(CGBCON/ZGBCON) Estimate condition number of complex band matrix, matrix already factorized by F07BRF
F07BVF	15	(CGBRFS/ZGBRFS) Refined solution with error bounds of complex band system of linear equations, multiple right-hand sides
F07FDF	15	(SPOTRF/DPOTRF) Cholesky factorization of real symmetric positive-definite matrix
F07FEF	15	(SPOTRS/DPOTRS) Solution of real symmetric positive-definite system of linear equations, multiple right-hand sides, matrix already factorized by F07FDF
F07FGF	15	(SPOCON/DPOCON) Estimate condition number of real symmetric positive-definite matrix, matrix already factorized by F07FDF
F07FHF	15	(SPORFS/DPORFS) Refined solution with error bounds of real symmetric positive-definite system of linear equations, multiple right-hand sides
F07FJF	15	(SPOTRI/DPOTRI) Inverse of real symmetric positive-definite matrix, matrix already factorized by F07FDF
F07FRF	15	(CPOTRF/ZPOTRF) Cholesky factorization of complex Hermitian positive-definite matrix
F07FSF	15	(CPOTRS/ZPOTRS) Solution of complex Hermitian positive-definite system of linear equations, multiple right-hand sides, matrix already factorized by F07FRF

F07FUF	15	(CPOCON/ZPOCON) Estimate condition number of complex Hermitian positive-definite matrix, matrix already factorized by F07FRF
F07FVF	15	(CPORFS/ZPORFS) Refined solution with error bounds of complex Hermitian positive-definite system of linear equations, multiple right-hand sides
F07FWF	15	(CPOTRI/ZPOTRI) Inverse of complex Hermitian positive-definite matrix, matrix already factorized by F07FRF
F07GDF	15	(SPPTRF/DPPTRF) Cholesky factorization of real symmetric positive-definite matrix, packed storage
F07GEF	15	(SPPTRS/DPPTRS) Solution of real symmetric positive-definite system of linear equations, multiple right-hand sides, matrix already factorized by F07GDF, packed storage
F07GGF	15	(SPPCON/DPPCON) Estimate condition number of real symmetric positive-definite matrix, matrix already factorized by F07GDF, packed storage
F07GHF	15	(SPPRFS/DPPRFS) Refined solution with error bounds of real symmetric positive-definite system of linear equations, multiple right-hand sides, packed storage
F07GJF	15	(SPPTRI/DPPTRI) Inverse of real symmetric positive-definite matrix, matrix already factorized by F07GDF, packed storage
F07GRF	15	(CPPTRF/ZPPTRF) Cholesky factorization of complex Hermitian positive-definite matrix, packed storage
F07GSF	15	(CPPTRS/ZPPTRS) Solution of complex Hermitian positive-definite system of linear equations, multiple right-hand sides, matrix already factorized by F07GRF, packed storage
F07GUF	15	(CPPCON/ZPPCON) Estimate condition number of complex Hermitian positive-definite matrix, matrix already factorized by F07GRF, packed storage
F07GVF	15	(CPPRFS/ZPPRFS) Refined solution with error bounds of complex Hermitian positive-definite system of linear equations, multiple right-hand sides, packed storage
F07GWF	15	(CPPTRI/ZPPTRI) Inverse of complex Hermitian positive-definite matrix, matrix already factorized by F07GRF, packed storage
F07HDF	15	(SPBTRF/DPBTRF) Cholesky factorization of real symmetric positive-definite band matrix
F07HEF	15	(SPBTRS/DPBTRS) Solution of real symmetric positive-definite band system of linear equations, multiple right-hand sides, matrix already factorized by F07HDF
F07HGF	15	(SPBCON/DPBCON) Estimate condition number of real symmetric positive-definite band matrix, matrix already factorized by F07HDF
F07HHF	15	(SPBRFS/DPBRFS) Refined solution with error bounds of real symmetric positive-definite band system of linear equations, multiple right-hand sides
F07HRF	15	(CPBTRF/ZPBTRF) Cholesky factorization of complex Hermitian positive-definite band matrix
F07HSF	15	(CPBTRS/ZPBTRS) Solution of complex Hermitian positive-definite band system of linear equations, multiple right-hand sides, matrix already factorized by F07HRF
F07HUF	15	(CPBCON/ZPBCON) Estimate condition number of complex Hermitian positive-definite band matrix, matrix already factorized by F07HRF
F07HVF	15	(CPBRFS/ZPBRFS) Refined solution with error bounds of complex Hermitian positive-definite band system of linear equations, multiple right-hand sides
F07MDF	15	(SSYTRF/DSYTRF) Bunch–Kaufman factorization of real symmetric indefinite matrix
F07MEF	15	(SSYTRS/DSYTRS) Solution of real symmetric indefinite system of linear equations, multiple right-hand sides, matrix already factorized by F07MDF

F07MGF	15	(SSYCON/DSYCON) Estimate condition number of real symmetric indefinite matrix, matrix already factorized by F07MDF
F07MHF	15	(SSYRFS/DSYRFS) Refined solution with error bounds of real symmetric indefinite system of linear equations, multiple right-hand sides
F07MJF	15	(SSYTRI/DSYTRI) Inverse of real symmetric indefinite matrix, matrix already factorized by F07MDF
F07MRF	15	(CHETRF/ZHETRF) Bunch–Kaufman factorization of complex Hermitian indefinite matrix
F07MSF	15	(CHETRS/ZHETRS) Solution of complex Hermitian indefinite system of linear equations, multiple right-hand sides, matrix already factorized by F07MRF
F07MUF	15	(CHECON/ZHECON) Estimate condition number of complex Hermitian indefinite matrix, matrix already factorized by F07MRF
F07MVF	15	(CHERFS/ZHERFS) Refined solution with error bounds of complex Hermitian indefinite system of linear equations, multiple right-hand sides
F07MWF	15	(CHETRI/ZHETRI) Inverse of complex Hermitian indefinite matrix, matrix already factorized by F07MRF
F07NRF	15	(CSYTRF/ZSYTRF) Bunch–Kaufman factorization of complex symmetric matrix
F07NSF	15	(CSYTRS/ZSYTRS) Solution of complex symmetric system of linear equations, multiple right-hand sides, matrix already factorized by F07NRF
F07NUF	15	(CSYCON/ZSYCON) Estimate condition number of complex symmetric matrix, matrix already factorized by F07NRF
F07NVF	15	(CSYRFS/ZSYRFS) Refined solution with error bounds of complex symmetric system of linear equations, multiple right-hand sides
F07NWF	15	(CSYTRI/ZSYTRI) Inverse of complex symmetric matrix, matrix already factorized by F07NRF
F07PDF	15	(SSPTRF/DSPTRF) Bunch–Kaufman factorization of real symmetric indefinite matrix, packed storage
F07PEF	15	(SSPTRS/DSPTRS) Solution of real symmetric indefinite system of linear equations, multiple right-hand sides, matrix already factorized by F07PDF, packed storage
F07PGF	15	(SSPCON/DSPCON) Estimate condition number of real symmetric indefinite matrix, matrix already factorized by F07PDF, packed storage
F07PHF	15	(SSPRFS/DSPRFS) Refined solution with error bounds of real symmetric indefinite system of linear equations, multiple right-hand sides, packed storage
F07PJF	15	(SSPTRI/DSPTRI) Inverse of real symmetric indefinite matrix, matrix already factorized by F07PDF, packed storage
F07PRF	15	(CHPTRF/ZHPTRF) Bunch–Kaufman factorization of complex Hermitian indefinite matrix, packed storage
F07PSF	15	(CHPTRS/ZHPTRS) Solution of complex Hermitian indefinite system of linear equations, multiple right-hand sides, matrix already factorized by F07PRF, packed storage
F07PUF	15	(CHPCON/ZHPCON) Estimate condition number of complex Hermitian indefinite matrix, matrix already factorized by F07PRF, packed storage
F07PVF	15	(CHPRFS/ZHPRFS) Refined solution with error bounds of complex Hermitian indefinite system of linear equations, multiple right-hand sides, packed storage
F07PWF	15	(CHPTRI/ZHPTRI) Inverse of complex Hermitian indefinite matrix, matrix already factorized by F07PRF, packed storage
F07QRF	15	(CSPTRF/ZSPTRF) Bunch–Kaufman factorization of complex symmetric matrix, packed storage
F07QSF	15	(CSPTRS/ZSPTRS) Solution of complex symmetric system of linear equations, multiple right-hand sides, matrix already factorized by F07QRF, packed storage

F07QUF	15	(CSPCON/ZSPCON) Estimate condition number of complex symmetric matrix, matrix already factorized by F07QRF, packed storage
F07QVF	15	(CSPRFS/ZSPRFS) Refined solution with error bounds of complex symmetric system of linear equations, multiple right-hand sides, packed storage
F07QWF	15	(CSPTRI/ZSPTRI) Inverse of complex symmetric matrix, matrix already factorized by F07QRF, packed storage
F07TEF	15	(STRTRS/DTRTRS) Solution of real triangular system of linear equations, multiple right-hand sides
F07TGF	15	(STRCON/DTRCON) Estimate condition number of real triangular matrix
F07THF	15	(STRRFS/DTRRFS) Error bounds for solution of real triangular system of linear equations, multiple right-hand sides
F07TJF	15	(STRTRI/DTRTRI) Inverse of real triangular matrix
F07TSF	15	(CTRTRS/ZTRTRS) Solution of complex triangular system of linear equations, multiple right-hand sides
F07TUF	15	(CTRCON/ZTRCON) Estimate condition number of complex triangular matrix
F07TVF	15	(CTRRFS/ZTRRFS) Error bounds for solution of complex triangular system of linear equations, multiple right-hand sides
F07TWF	15	(CTRTRI/ZTRTRI) Inverse of complex triangular matrix
F07UEF	15	(STPTRS/DTPTRS) Solution of real triangular system of linear equations, multiple right-hand sides, packed storage
F07UGF	15	(STPCON/DTPCON) Estimate condition number of real triangular matrix, packed storage
F07UHF	15	(STPRFS/DTPRFS) Error bounds for solution of real triangular system of linear equations, multiple right-hand sides, packed storage
F07UJF	15	(STPTRI/DTPTRI) Inverse of real triangular matrix, packed storage
F07USF	15	(CTPTRS/ZTPTRS) Solution of complex triangular system of linear equations, multiple right-hand sides, packed storage
F07UUF	15	(CTPCON/ZTPCON) Estimate condition number of complex triangular matrix, packed storage
F07UVF	15	(CTPRFS/ZTPRFS) Error bounds for solution of complex triangular system of linear equations, multiple right-hand sides, packed storage
F07UWF	15	(CTPTRI/ZTPTRI) Inverse of complex triangular matrix, packed storage
F07VEF	15	(STBTRS/DTBTRS) Solution of real band triangular system of linear equations, multiple right-hand sides
F07VGF	15	(STBCON/DTBCON) Estimate condition number of real band triangular matrix
F07VHF	15	(STBRFS/DTBRFS) Error bounds for solution of real band triangular system of linear equations, multiple right-hand sides
F07VSF	15	(CTBTRS/ZTBTRS) Solution of complex band triangular system of linear equations, multiple right-hand sides
F07VUF	15	(CTBCON/ZTBCON) Estimate condition number of complex band triangular matrix
F07VVF	15	(CTBRFS/ZTBRFS) Error bounds for solution of complex band triangular system of linear equations, multiple right-hand sides

Chapter F07

Linear Equations (LAPACK)

Contents

1 Scope of the Chapter

This chapter provides routines for the solution of systems of simultaneous linear equations, and associated computations. It provides routines for:

- matrix factorizations
- solution of linear equations
- estimating matrix condition numbers
- computing error bounds for the solution of linear equations
- matrix inversion

Routines are provided for both *real* and *complex* data.

For a general introduction to the solution of systems of linear equations, you should turn first to the F04 Chapter Introduction. The decision trees, at the end of the F04 Chapter Introduction, direct you to the most appropriate routines in Chapter F04 or F07, for solving your particular problem. In particular, Chapter F04 contains *Black Box* routines which enable some standard types of problem to be solved by a call to a single routine. Where possible, routines in Chapter F04 call F07 routines to perform the necessary computational tasks.

The routines in this chapter (F07) handle only *dense* and *band* matrices (not matrices with more specialized structures, or general sparse matrices).

The routines in this chapter have all been derived from the LAPACK project (see Anderson *et al.* [1]). They have been designed to be efficient on a wide range of high-performance computers, without compromising efficiency on conventional serial machines.

2 Background to the Problems

This section is only a brief introduction to the numerical solution of systems of linear equations. Consult a standard textbook for a more thorough discussion, for example Golub and Van Loan [2].

2.1 Notation

We use the standard notation for a system of simultaneous linear equations:

$$Ax = b \tag{1}$$

where A is the *coefficient matrix*, b is the *right-hand side*, and x is the *solution*. A is assumed to be a square matrix of order n.

If there are several right-hand sides, we write

$$AX = B \tag{2}$$

where the columns of B are the individual right-hand sides, and the columns of X are the corresponding solutions.

We also use the following notation, both here and in the routine documents:

$\hat{x}$	a *computed* solution to $Ax = b$, (which usually differs from the exact solution x because of round-off error)
$r = b - A\hat{x}$	the *residual* corresponding to the computed solution $\hat{x}$
$\|x\|_\infty = \max_i \|x_i\|$	the infinity-norm of the vector x
$\|A\|_\infty = \max_i \sum_j \|a_{ij}\|$	the infinity-norm of the vector A
$\|x\|$	the vector with elements $\|x_i\|$
$\|A\|$	the matrix with elements $\|a_{ij}\|$

Inequalities of the form $|A| \leq |B|$ are interpreted componentwise, that is $|a_{ij}| \leq |b_{ij}|$ for all i, j.

2.2 Matrix Factorizations

If A is upper or lower triangular, $Ax = b$ can be solved by a straightforward process of backward or forward substitution.

Otherwise, the solution is obtained after first factorizing A, as follows:

General matrices (*LU* factorization with partial pivoting):

$$A = PLU$$

where P is a permutation matrix, L is lower-triangular with diagonal elements equal to 1, and U is upper-triangular; the permutation matrix P (which represents row interchanges) is needed to ensure numerical stability.

Symmetric positive-definite matrices (Cholesky factorization):

$$A = U^T U \text{ or } A = LL^T$$

where U is upper triangular and L is lower triangular.

Symmetric indefinite matrices (Bunch–Kaufman factorization):

$$A = PUDU^T P^T \text{ or } A = PLDL^T P^T$$

where P is a permutation matrix, U is upper triangular, L is lower triangular, and D is a block diagonal matrix with diagonal blocks of order 1 or 2; U and L have diagonal elements equal to 1, and have 2 by 2 unit matrices on the diagonal corresponding to the 2 by 2 blocks of D. The permutation matrix P (which represents symmetric row-and-column interchanges) and the 2 by 2 blocks in D are needed to ensure numerical stability. If A is in fact positive-definite, no interchanges are needed and the factorization reduces to $A = UDU^T$ or $A = LDL^T$ with diagonal D, which is simply a variant form of the Cholesky factorization.

2.3 Solution of Systems of Equations

Given one of the above matrix factorizations, it is straightforward to compute a solution to $Ax = b$ by solving two subproblems, as shown below, first for y and then for x. Each subproblem consists essentially of solving a triangular system of equations by forward or backward substitution; the permutation matrix P and the block diagonal matrix D introduce only a little extra complication:

General matrices (*LU* factorization):

$$\begin{aligned} Ly &= P^T b \\ Ux &= y \end{aligned}$$

Symmetric positive-definite matrices (Cholesky factorization):

$$\begin{aligned} U^T y &= b \\ Ux &= y \end{aligned} \text{ or } Ly = bL^T x = y$$

Symmetric indefinite matrices (Bunch–Kaufman factorization):

$$\begin{aligned} PUDy &= b \\ U^T P^T x &= y \end{aligned} \text{ or } \begin{aligned} PLDy &= b \\ L^T P^T x &= y \end{aligned}$$

2.4 Sensitivity and Error Analysis

2.4.1 Normwise error bounds

Frequently in practical problems, the data A and b are not known exactly, and it is then important to understand how uncertainties or perturbations in the data can affect the solution.

If x is the exact solution to $Ax = b$, and $x + \delta x$ is the exact solution to a perturbed problem $(A + \delta A)(x + \delta x) = (b + \delta b)$, then:

$$\frac{||\delta x||}{||x||} \leq \kappa(A) \left(\frac{||\delta A||}{||A||} + \frac{||\delta b||}{||b||} \right) + \ldots \text{(2nd order terms)}$$

where $\kappa(A)$ is the *condition number* of A defined by:

$$\kappa(A) = \|A\|.\|A^{-1}\|. \tag{3}$$

In other words, relative errors in A or b may be amplified in x by a factor $\kappa(A)$. Section 2.4.2 discusses how to compute or estimate $\kappa(A)$.

Similar considerations apply when we study the effects of *rounding errors* introduced by computation in finite precision. The effects of rounding errors can be shown to be equivalent to perturbations in the original data, such that $\frac{\|\delta A\|}{\|A\|}$ and $\frac{\|\delta b\|}{\|b\|}$ are usually at most $p(n)\epsilon$, where ϵ is the ***machine precision*** and $p(n)$ is an increasing function of n which is seldom larger than $10n$ (although in theory it can be as large as 2^{n-1}).

In other words, the computed solution $\hat{x}$ is the exact solution of a linear system $(A+\delta A)\hat{x} = b+\delta b$ which is close to the original system in a normwise sense.

2.4.2 Estimating condition numbers

The previous section has emphasized the usefulness of the quantity $\kappa(A)$ in understanding the sensitivity of the solution of $Ax = b$. To compute the value of $\kappa(A)$ from equation (3) is more expensive than solving $Ax = b$ in the first place. Hence it is standard practice to *estimate* $\kappa(A)$, in either the 1-norm or the ∞-norm, by a method which only requires $O(n^2)$ additional operations, assuming that a suitable factorization of A is available.

The method used in this chapter is Higham's modification of Hager's method [3]. It yields an estimate which is never larger than the true value, but which seldom falls short by more than a factor of 3 (although artificial examples can be constructed where it is much smaller). This is acceptable since it is the order of magnitude of $\kappa(A)$ which is important rather than its precise value.

Because $\kappa(A)$ is infinite if A is singular, the routines in this chapter actually return the *reciprocal* of $\kappa(A)$.

2.4.3 Componentwise error bounds

A disadvantage of normwise error bounds is that they do not reflect any special structure in the data A and b – that is, a pattern of elements which are known to be zero – and the bounds are dominated by the largest elements in the data.

Componentwise error bounds overcome these limitations. Instead of the normwise relative error, we can bound the relative error in *each component* of A and b:

$$\max_{ijk}\left(\frac{|\delta a_{ij}|}{|a_{ij}|}, \frac{|\delta b_k|}{|b_k|}\right) \leq \omega$$

where the *componentwise backward error bound* ω is given by:

$$\omega = \max_i \frac{|r_i|}{(|A|.|\hat{x}| + |b|)_i}.$$

Routines are provided in this chapter which compute ω, and also compute a *forward error bound* which is sometimes much sharper than the normwise bound given earlier:

$$\frac{\|x - \hat{x}\|_\infty}{\|x\|_\infty} \leq \frac{\| \, |A^{-1}|.|r| \, \|_\infty}{\|x\|_\infty}.$$

Care is taken when computing this bound to allow for rounding errors in computing r. The norm $\| \, |A^{-1}|.|r| \, \|_\infty$ is estimated cheaply (without computing A^{-1}) by a modification of the method used to estimate $\kappa(A)$.

2.4.4 Iterative refinement of the solution

If $\hat{x}$ is an approximate computed solution to $Ax = b$, and r is the corresponding residual, then a procedure for *iterative refinement* of $\hat{x}$ can be defined as follows, starting with $x_0 = \hat{x}$:

for $i = 0, 1, \ldots$, until convergence

compute $r_i = b - Ax_i$
solve $Ad_i = r_i$
compute $x_{i+1} = x_i + d_i$

In Chapter F04, routines are provided which perform this procedure using ***additional precision*** to compute r, and are thus able to reduce the *forward error* to the level of ***machine precision***.

The routines in this chapter do *not* use ***additional precision*** to compute r, and cannot guarantee a small forward error, but can guarantee a *small backward error* (except in rare cases when A is very ill-conditioned, or when A and x are sparse in such a way that $|A|.|x|$ has a zero or very small component). The iterations continue until the backward error has been reduced as much as possible; usually only one iteration is needed, and at most five iterations are allowed.

2.5 Matrix Inversion

It is seldom necessary to compute an explicit inverse of a matrix. In particular, do *not* attempt to solve $Ax = b$ by first computing A^{-1} and then forming the matrix-vector product $x = A^{-1}b$; the procedure described in Section 2.3 is more efficient and more accurate.

However, routines are provided for the rare occasions when an inverse is needed, using one of the factorizations described in Section 2.2.

2.6 Packed Storage

Routines which handle symmetric matrices are usually designed so that they use either the upper or lower triangle of the matrix; it is not necessary to store the whole matrix. If the upper or lower triangle is stored conventionally in the upper or lower triangle of a 2-dimensional array, the remaining elements of the array can be used to store other useful data. However, that is not always convenient, and if it is important to economize on storage, the upper or lower triangle can be stored in a 1-dimensional array of length $n(n+1)/2$ - in other words, the storage is almost halved.

This storage format is referred to as *packed storage;* it is described in Section 3.3.2. It may also be used for triangular matrices.

Routines designed for packed storage perform the same number of arithmetic operations as routines which use conventional storage, but they are usually less efficient, especially on high-performance computers, so there is then a trade-off between storage and efficiency.

2.7 Band Matrices

A *band* matrix is one whose non-zero elements are confined to a relatively small number of sub-diagonals or super-diagonals on either side of the main diagonal. Algorithms can take advantage of bandedness to reduce the amount of work and storage required. The storage scheme used for band matrices is described in Section 3.3.3.

The LU factorization for general matrices, and the Cholesky factorization for symmetric positive-definite matrices both preserve bandedness. Hence routines are provided which take advantage of the band structure when solving systems of linear equations.

The Cholesky factorization preserves bandedness in a very precise sense: the factor U or L has the same number of super-diagonals or sub-diagonals as the original matrix. In the LU factorization, the row-interchanges modify the band structure: if A has k_l sub-diagonals and k_u super-diagonals, then L is not a band matrix but still has at most k_l non-zero elements below the diagonal in each column; and U has at most $k_l + k_u$ super-diagonals.

The Bunch–Kaufman factorization does not preserve bandedness, because of the need for symmetric row-and-column permutations; hence no routines are provided for symmetric indefinite band matrices.

The inverse of a band matrix does not in general have a band structure, so no routines are provided for computing inverses of band matrices.

2.8 Block Algorithms

Many of the routines in this chapter use what is termed a *block algorithm*. This means that at each major step of the algorithm a *block* of rows or columns is updated, and most of the computation is performed by matrix-matrix operations on these blocks. The matrix-matrix operations are performed by calls to the Level 3 BLAS (see Chapter F06), which are the key to achieving high performance on many modern computers. See Golub and Van Loan [2] or Anderson *et al.* [1] for more about block algorithms.

The performance of a block algorithm varies to some extent with the ***blocksize*** – that is, the number of rows or columns per block. This is a machine-dependent parameter, which is set to a suitable value when the library is implemented on each range of machines. Users of the library do not normally need to be aware of what value is being used. Different block sizes may be used for different routines. Values in the range 16 to 64 are typical.

On more conventional machines there is often no advantage from using a block algorithm, and then the routines use an *unblocked* algorithm (effectively a blocksize of 1), relying solely on calls to the Level 2 BLAS (see Chapter F06 again).

The only situation in which a user needs some awareness of the block size is when it affects the amount of workspace to be supplied to a particular routine. This is discussed in Section 3.4.3.

3 Recommendations on Choice and Use of Available Routines

Note. Refer to the Users' Note for your implementation to check that a routine is available.

3.1 Available Routines

Tables 1 and 2 in Section 3.5 show the routines which are provided for performing different computations on different types of matrices. Table 1 shows routines for real matrices; Table 2 shows routines for complex matrices. Each entry in the table gives the NAG routine name, the LAPACK single precision name, and the LAPACK double precision name (see Section 3.2).

Routines are provided for the following types of matrix:

general

general band

symmetric or Hermitian positive-definite

symmetric or Hermitian positive-definite (packed storage)

symmetric or Hermitian positive-definite band

symmetric or Hermitian indefinite

symmetric or Hermitian indefinite (packed storage)

triangular

triangular (packed storage)

triangular band

For each of the above types of matrix (except where indicated), routines are provided to perform the following computations:

(a) (except for triangular matrices) factorize the matrix (see Section 2.2).
(b) solve a system of linear equations, using the factorization (see Section 2.3).
(c) estimate the condition number of the matrix, using the factorization (see Section 2.4.2); these routines also require the norm of the original matrix (except when the matrix is triangular) which may be computed by a routine in Chapter F06.
(d) refine the solution and compute forward and backward error bounds (see Section 2.4.3 and Section 2.4.4); these routines require the original matrix and right-hand side, as well as the factorization returned from (a) and the solution returned from (b).
(e) (except for band matrices) invert the matrix, using the factorization (see Section 2.5).

Thus, to solve a particular problem, it is usually necessary to call two or more routines in succession. This is illustrated in the example programs in the routine documents.

3.2 NAG Names and LAPACK Names

As well as the NAG routine name (beginning F07-), Tables 1 and 2 show the LAPACK routine names in both single and double precision.

The routines may be called either by their NAG names or by their LAPACK names. When using a single precision implementation of the NAG Library, the single precision form of the LAPACK name must be used (beginning with S- or C-); when using a double precision implementation of the NAG Library, the double precision form of the LAPACK name must be used (beginning with D- or Z-).

References to F07 routines in the Manual normally include the LAPACK single and double precision names, in that order – for example, F07ADF (SGETRF/DGETRF).

The LAPACK routine names follow a simple scheme (which is similar to that used for the BLAS in Chapter F06). Each name has the structure **XYYZZZ**, where the components have the following meanings:

- the initial letter **X** indicates the data type (real or complex) and precision:
  - S – real, single precision (in Fortran 77, `REAL`)
  - D – real, double precision (in Fortran 77, `DOUBLE PRECISION`)
  - C – complex, single precision (in Fortran 77, `COMPLEX`)
  - Z – complex, double precision (in Fortran 77, `COMPLEX*16` or `DOUBLE COMPLEX`)
- the 2nd and 3rd letters **YY** indicate the type of the matrix A (and in some cases its storage scheme):
  - GE – general
  - GB – general band
  - PO – symmetric or Hermitian positive-definite
  - PP – symmetric or Hermitian positive-definite (packed storage)
  - PB – symmetric or Hermitian positive-definite band
  - SY – symmetric indefinite
  - SP – symmetric indefinite (packed storage)
  - HE – (complex) Hermitian indefinite
  - HP – (complex) Hermitian indefinite (packed storage)
  - TR – triangular
  - TP – triangular (packed storage)
  - TB – triangular band
- the last 3 letters **ZZZ** indicate the computation performed:
  - TRF – triangular factorization
  - TRS – solution of linear equations, using the factorization
  - CON – estimate condition number
  - RFS – refine solution and compute error bounds
  - TRI – compute inverse, using the factorization

Thus the routine SGETRF performs a triangular factorization of a real general matrix in a single precision implementation of the Library; the corresponding routine in a double precision implementation is DGETRF.

Some sections of the routine documents – Section 2 (Specification) and Section 9.1 (Example program) – print the LAPACK name in ***bold italics***, according to the NAG convention of using bold italics for precision-dependent terms – for example, ***sgetrf***, which should be interpreted as either SGETRF (in single precision) or DGETRF (in double precision).

3.3 Matrix Storage Schemes

In this chapter the following different storage schemes are used for matrices:

- conventional storage in a 2-dimensional array;
- packed storage for symmetric, Hermitian or triangular matrices;

– band storage for band matrices;

These storage schemes are compatible with those used in Chapter F06 (especially in the BLAS) and Chapter F08, but different schemes for packed or band storage are used in a few older routines in Chapters F01, F02, F03 and F04.

In the examples below, $*$ indicates an array element which need not be set and is not referenced by the routines. The examples illustrate only the relevant leading rows and columns of the arrays; array arguments may of course have additional rows or columns, according to the usual rules for passing array arguments in Fortran 77.

3.3.1 Conventional storage

The default scheme for storing matrices is the obvious one: a matrix A is stored in a 2-dimensional array A, with matrix element a_{ij} stored in array element $A(i,j)$.

If a matrix is **triangular** (upper or lower, as specified by the argument UPLO), only the elements of the relevant triangle are stored; the remaining elements of the array need not be set. Such elements are indicated by $*$ in the examples below. For example, when $n = 4$:

UPLO	Triangular matrix A	Storage in array A
'U'	$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ & a_{22} & a_{23} & a_{24} \\ & & a_{33} & a_{34} \\ & & & a_{44} \end{pmatrix}$	$\begin{matrix} a_{11} & a_{12} & a_{13} & a_{14} \\ * & a_{22} & a_{23} & a_{24} \\ * & * & a_{33} & a_{34} \\ * & * & * & a_{44} \end{matrix}$
'L'	$\begin{pmatrix} a_{11} & & & \\ a_{21} & a_{22} & & \\ a_{31} & a_{32} & a_{33} & \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$	$\begin{matrix} a_{11} & * & * & * \\ a_{21} & a_{22} & * & * \\ a_{31} & a_{32} & a_{33} & * \\ a_{41} & a_{42} & a_{43} & a_{44} \end{matrix}$

Routines which handle **symmetric** or **Hermitian** matrices allow for either the upper or lower triangle of the matrix (as specified by UPLO) to be stored in the corresponding elements of the array; the remaining elements of the array need not be set. For example, when $n = 4$:

UPLO	Hermitian matrix A	Storage in array A
'U'	$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ \bar{a}_{12} & a_{22} & a_{23} & a_{24} \\ \bar{a}_{13} & \bar{a}_{23} & a_{33} & a_{34} \\ \bar{a}_{14} & \bar{a}_{24} & \bar{a}_{34} & a_{44} \end{pmatrix}$	$\begin{matrix} a_{11} & a_{12} & a_{13} & a_{14} \\ * & a_{22} & a_{23} & a_{24} \\ * & * & a_{33} & a_{34} \\ * & * & * & a_{44} \end{matrix}$
'L'	$\begin{pmatrix} a_{11} & \bar{a}_{21} & \bar{a}_{31} & \bar{a}_{41} \\ a_{21} & a_{22} & \bar{a}_{32} & \bar{a}_{42} \\ a_{31} & a_{32} & a_{33} & \bar{a}_{43} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$	$\begin{matrix} a_{11} & * & * & * \\ a_{21} & a_{22} & * & * \\ a_{31} & a_{32} & a_{33} & * \\ a_{41} & a_{42} & a_{43} & a_{44} \end{matrix}$

3.3.2 Packed Storage

Symmetric, Hermitian or triangular matrices may be stored more compactly, if the relevant triangle (again as specified by UPLO) is packed by columns in a 1-dimensional array. In Chapters F07 and F08, arrays which hold matrices in packed storage, have names ending in P. So:

if UPLO = 'U', a_{ij} is stored in $AP(i + j(j-1)/2)$ for $i \leq j$;

if UPLO = 'L', a_{ij} is stored in $AP(i + (2n-j)(j-1)/2)$ for $j \leq i$.

For example:

UPLO	Triangle of matrix A	Packed storage in array AP
'U'	$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ & a_{22} & a_{23} & a_{24} \\ & & a_{33} & a_{34} \\ & & & a_{44} \end{pmatrix}$	$a_{11}\ \underbrace{a_{12}a_{22}}\ \underbrace{a_{13}a_{23}a_{33}}\ \underbrace{a_{14}a_{24}a_{34}a_{44}}$
'L'	$\begin{pmatrix} a_{11} & & & \\ a_{21} & a_{22} & & \\ a_{31} & a_{32} & a_{33} & \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$	$\underbrace{a_{11}a_{21}a_{31}a_{41}}\ \underbrace{a_{22}a_{32}a_{42}}\ \underbrace{a_{33}a_{43}}\ a_{44}$

Note that for real symmetric matrices, packing the upper triangle by columns is equivalent to packing the lower triangle by rows; packing the lower triangle by columns is equivalent to packing the upper triangle by rows. (For complex Hermitian matrices, the only difference is that the off-diagonal elements are conjugated.)

3.3.3 Band storage

A band matrix with k_l sub-diagonals and k_u super-diagonals may be stored compactly in a 2-dimensional array with $k_l + k_u + 1$ rows and n columns. Columns of the matrix are stored in corresponding columns of the array, and diagonals of the matrix are stored in rows of the array. This storage scheme should be used in practice only if k_l, $k_u \ll n$, although the routines in Chapters F07 and F08 work correctly for all values of k_l and k_u. In Chapters F07 and F08 arrays which hold matrices in band storage have names ending in B.

To be precise, a_{ij} is stored in $\mathrm{AB}(k_u + 1 + i - j, j)$ for $\max(1, j - k_u) \leq i \leq \min(n, j + k_l)$. For example, when $n = 5$, $k_l = 2$ and $k_u = 1$:

Band matrix A	Band storage in array AB
$\begin{pmatrix} a_{11} & a_{12} & & & \\ a_{21} & a_{22} & a_{23} & & \\ a_{31} & a_{32} & a_{33} & a_{34} & \\ & a_{42} & a_{43} & a_{44} & a_{45} \\ & & a_{53} & a_{54} & a_{55} \end{pmatrix}$	$\begin{pmatrix} & & & & \\ * & a_{12} & a_{23} & a_{34} & a_{45} \\ a_{11} & a_{22} & a_{33} & a_{44} & a_{55} \\ a_{21} & a_{32} & a_{43} & a_{54} & * \\ a_{31} & a_{42} & a_{53} & * & * \end{pmatrix}$

The elements marked $*$ in the upper left and lower right corners of the array AB need not be set, and are not referenced by the routines.

Note. when a general band matrix is supplied for LU factorization, space must be allowed to store an additional k_l super-diagonals, generated by fill-in as a result of row interchanges. This means that the matrix is stored according to the above scheme, but with $k_l + k_u$ super-diagonals.

Triangular band matrices are stored in the same format, with either $k_l = 0$ if upper triangular, or $k_u = 0$ if lower triangular.

For symmetric or Hermitian band matrices with k sub-diagonals or super-diagonals, only the upper or lower triangle (as specified by UPLO) need be stored:

if UPLO = 'U', a_{ij} is stored in $\mathrm{AB}(k + 1 + i - j, j)$ for $\max(1, j - k) \leq i \leq j$;

if UPLO = 'L', a_{ij} is stored in $\mathrm{AB}(1 + i - j, j)$ for $j \leq i \leq \min(n, j + k)$.

For example, when $n = 5$ and $k = 2$:

UPLO	Hermitian band matrix A	Band storage in array A
'U'	$\begin{pmatrix} a_{11} & a_{12} & a_{13} & & \\ \bar{a}_{12} & a_{22} & a_{23} & a_{24} & \\ \bar{a}_{13} & \bar{a}_{23} & a_{33} & a_{34} & a_{35} \\ & \bar{a}_{24} & \bar{a}_{34} & a_{44} & a_{45} \\ & & \bar{a}_{35} & \bar{a}_{45} & a_{55} \end{pmatrix}$	$\begin{pmatrix} * & * & a_{13} & a_{24} & a_{35} \\ * & a_{12} & a_{23} & a_{34} & a_{45} \\ a_{11} & a_{22} & a_{33} & a_{44} & a_{55} \end{pmatrix}$
'L'	$\begin{pmatrix} a_{11} & \bar{a}_{21} & \bar{a}_{31} & & \\ a_{21} & a_{22} & \bar{a}_{32} & \bar{a}_{42} & \\ a_{31} & a_{32} & a_{33} & \bar{a}_{43} & \bar{a}_{53} \\ & a_{42} & a_{43} & a_{44} & \bar{a}_{54} \\ & & a_{53} & a_{54} & a_{55} \end{pmatrix}$	$\begin{pmatrix} a_{11} & a_{22} & a_{33} & a_{44} & a_{55} \\ a_{21} & a_{32} & a_{43} & a_{54} & * \\ a_{31} & a_{42} & a_{53} & * & * \end{pmatrix}$

Note that different storage schemes for band matrices are used by some routines in Chapters F01, F02, F03 and F04.

3.3.4 Unit triangular matrices

Some routines in this chapter have an option to handle unit triangular matrices (that is, triangular matrices with diagonal elements = 1). This option is specified by an argument DIAG. If DIAG = 'U' (Unit triangular), the diagonal elements of the matrix need not be stored, and the corresponding array elements are not referenced by the routines. The storage scheme for the rest of the matrix (whether conventional, packed or band) remains unchanged.

3.3.5 Real diagonal elements of complex matrices

Complex Hermitian matrices have diagonal elements that are by definition purely real. In addition, complex triangular matrices which arise in Cholesky factorization are defined by the algorithm to have real diagonal elements.

If such matrices are supplied as input to routines in this chapter, the imaginary parts of the diagonal elements are not referenced, but are assumed to be zero. If such matrices are returned as output by the routines, the computed imaginary parts are explicitly set to zero.

3.4 Parameter Conventions

3.4.1 Option parameters

Most routines in this chapter have one or more option parameters, of type CHARACTER. The descriptions in Section 5 of the routine documents refer only to upper-case values (for example 'U' or 'L'); however, in every case, the corresponding lower-case characters may be supplied (with the same meaning). Any other value is illegal.

A longer character string can be passed as the actual parameter, making the calling program more readable, but only the first character is significant. (This is a feature of Fortran 77.) For example:

```
CALL SGETRS ('Transpose', . . . )
```

3.4.2 Problem dimensions

It is permissible for the problem dimensions (for example, M, N or NRHS) to be passed as zero, in which case the computation (or part of it) is skipped. Negative dimensions are regarded as an error.

3.4.3 Length of work arrays

A few routines implementing block algorithms require workspace sufficient to hold one block of rows or columns of the matrix if they are to achieve optimum levels of performance — for example, workspace

of size $n \times nb$, where nb is the optimum block size. In such cases, the actual declared length of the work array must be passed as a separate parameter LWORK, which immediately follows WORK in the parameter-list.

The routine will still perform correctly when less workspace is provided: it uses the largest block size allowed by the amount of workspace supplied, as long as this is likely to give better performance than the unblocked algorithm. On exit, WORK(1) contains the minimum value of LWORK which would allow the routine to use the optimum block size; this value of LWORK can be used for subsequent runs.

If LWORK indicates that there is insufficient workspace to perform the unblocked algorithm, this is regarded as an illegal value of LWORK, and is treated like any other illegal parameter value (see Section 3.4.4).

If you are in doubt how much workspace to supply and are concerned to achieve optimum performance, supply a generous amount (assume a block size of 64, say), and then examine the value of WORK(1) on exit.

3.4.4 Error-handling and the diagnostic parameter INFO

Routines in this chapter do not use the usual NAG Library error-handling mechanism, involving the parameter IFAIL. Instead they have a diagnostic parameter INFO. (Thus they preserve complete compatibility with the LAPACK specification.)

Whereas IFAIL is an *Input/Output* parameter and must be set before calling a routine, INFO is purely an *Output* parameter and need not be set before entry.

INFO indicates the success or failure of the computation, as follows:

INFO = 0: successful termination

INFO > 0: failure in the course of computation, control returned to the calling program

If the routine document specfies that the routine may terminate with INFO>0, then it is **essential** to **test INFO on exit** from the routine. (This corresponds to a *soft failure* in terms of the usual NAG error-handling terminology.) No error message is output.

All routines check that input parameters such as N or LDA or option parameters of type CHARACTER have permitted values. If an illegal value of the ith parameter is detected, INFO is set to $-i$, a message is output, and execution of the program is terminated. (This corresponds to a *hard failure* in the usual NAG terminology.)

3.5 Tables of Available Routines

Routines for real matrices					
Type of matrix and storage scheme	**factorize**	**solve**	**condition number**	**error estimate**	**invert**
general	F07ADF SGETRF DGETRF	F07AEF SGETRS DGETRS	F07AGF SGECON DGECON	F07AHF SGERFS DGERFS	F07AJF SGETRI DGETRI
general band	F07BDF SGBTRF DGBTRF	F07BEF SGBTRS DGBTRS	F07BGF SGBCON DGBCON	F07BHF SGBRFS DGBRFS	
symmetric positive-definite	F07FDF SPOTRF DPOTRF	F07FEF SPOTRS DPOTRS	F07FGF SPOCON DPOCON	F07FHF SPORFS DPORFS	F07FJF SPOTRI DPOTRI
symmetric positive-definite (packed storage)	F07GDF SPPTRF DPPTRF	F07GEF SPPTRS DPPTRS	F07GGF SPPCON DPPCON	F07GHF SPPRFS DPPRFS	F07GJF SPPTRI DPPTRI
symmetric positive-definite band	F07HDF SPBTRF DPBTRF	F07HEF SPBTRS DPBTRS	F07HGF SPBCON DPBCON	F07HHF SPBRFS DPBRFS	
symmetric indefinite	F07MDF SSYTRF DSYTRF	F07MEF SSYTRS DSYTRS	F07MGF SSYCON DSYCON	F07MHF SSYRFS DSYRFS	F07MJF SSYTRI DSYTRI
symmetric indefinite (packed storage)	F07PDF SSPTRF DSPTRF	F07PEF SSPTRS DSPTRS	F07PGF SSPCON DSPCON	F07PHF SSPRFS DSPRFS	F07PJF SSPTRI DSPTRI
triangular		F07TEF STRTRS DTRTRS	F07TGF STRCON DTRCON	F07THF STRRFS DTRRFS	F07TJF STRTRI DTRTRI
triangular (packed storage)		F07UEF STPTRS DTPTRS	F07UGF STPCON DTPCON	F07UHF STPRFS DTPRFS	F07UJF STPTRI DTPTRI
triangular band		F07VEF STBTRS DTBTRS	F07VGF STBCON DTBCON	F07VHF STBRFS DTBRFS	

Table 1

Each entry gives:

the NAG routine name
the LAPACK routine name in a single precision implementation
the LAPACK routine name in a double precision implementation

Routines for complex matrices					
Type of matrix and storage scheme	**factorize**	**solve**	**condition number**	**error estimate**	**invert**
general	F07ARF CGETRF ZGETRF	F07ASF CGETRS ZGETRS	F07AUF CGECON ZGECON	F07AVF CGERFS ZGERFS	F07AWF CGETRI ZGETRI
general band	F07BRF CGBTRF ZGBTRF	F07BSF CGBTRS ZGBTRS	F07BUF CGBCON ZGBCON	F07BVF CGBRFS ZGBRFS	
Hermitian positive-definite	F07FRF CPOTRF ZPOTRF	F07FSF CPOTRS ZPOTRS	F07FUF CPOCON ZPOCON	F07FVF CPORFS ZPORFS	F07FWF CPOTRI ZPOTRI
Hermitian positive-definite (packed storage)	F07GRF CPPTRF ZPPTRF	F07GSF CPPTRS ZPPTRS	F07GUF CPPCON ZPPCON	F07GVF CPPRFS ZPPRFS	F07GWF CPPTRI ZPPTRI
Hermitian positive-definite band	F07HRF CPBTRF ZPBTRF	F07HSF CPBTRS ZPBTRS	F07HUF CPBCON ZPBCON	F07HVF CPBRFS ZPBRFS	
Hermitian indefinite	F07MRF CHETRF ZHETRF	F07MSF CHETRS ZHETRS	F07MUF CHECON ZHECON	F07MVF CHERFS ZHERFS	F07MWF CHETRI ZHETRI
symmetric indefinite	F07NRF CSYTRF ZSYTRF	F07NSF CSYTRS ZSYTRS	F07NUF CSYCON ZSYCON	F07NVF CSYRFS ZSYRFS	F07NWF CSYTRI ZSYTRI
Hermitian indefinite (packed storage)	F07PRF CHPTRF ZHPTRF	F07PSF CHPTRS ZHPTRS	F07PUF CHPCON ZHPCON	F07PVF CHPRFS ZHPRFS	F07PWF CHPTRI ZHPTRI
symmetric indefinite (packed storage)	F07QRF CSPTRF ZSPTRF	F07QSF CSPTRS ZSPTRS	F07QUF CSPCON ZSPCON	F07QVF CSPRFS ZSPRFS	F07QWF CSPTRI ZSPTRI
triangular		F07TSF CTRTRS ZTRTRS	F07TUF CTRCON ZTRCON	F07TVF CTRRFS ZTRRFS	F07TWF CTRTRI ZTRTRI
triangular (packed storage)		F07USF CTPTRS ZTPTRS	F07UUF CTPCON ZTPCON	F07UVF CTPRFS ZTPRFS	F07UWF CTPTRI ZTPTRI
triangular band		F07VSF CTBTRS ZTBTRS	F07VUF CTBCON ZTBCON	F07VVF CTBRFS ZTBRFS	

Table 2

Each entry gives:

The NAG routine name
the LAPACK routine name in a single precision implementation
the LAPACK routine name in a double precision implementation

4 Indexes of LAPACK routines

Real Matrices		
LAPACK single precision	LAPACK double precision	NAG
SGBCON	DGBCON	F07BGF
SGBRFS	DGBRFS	F07BHF
SGBTRF	DGBTRF	F07BDF
SGBTRS	DGBTRS	F07BEF
SGECON	DGECON	F07AGF
SGERFS	DGERFS	F07AHF
SGETRF	DGETRF	F07ADF
SGETRI	DGETRI	F07AJF
SGETRS	DGETRS	F07AEF
SPBCON	DPBCON	F07HGF
SPBRFS	DPBRFS	F07HHF
SPBTRF	DPBTRF	F07HDF
SPBTRS	DPBTRS	F07HEF
SPOCON	DPOCON	F07FGF
SPORFS	DPORFS	F07FHF
SPOTRF	DPOTRF	F07FDF
SPOTRI	DPOTRI	F07FJF
SPOTRS	DPOTRS	F07FEF
SPPCON	DPPCON	F07GGF
SPPRFS	DPPRFS	F07GHF
SPPTRF	DPPTRF	F07GDF
SPPTRI	DPPTRI	F07GJF
SPPTRS	DPPTRS	F07GEF
SSPCON	DSPCON	F07PGF
SSPRFS	DSPRFS	F07PHF
SSPTRF	DSPTRF	F07PDF
SSPTRI	DSPTRI	F07PJF
SSPTRS	DSPTRS	F07PEF
SSYCON	DSYCON	F07MGF
SSYRFS	DSYRFS	F07MHF
SSYTRF	DSYTRF	F07MDF
SSYTRI	DSYTRI	F07MJF
SSYTRS	DSYTRS	F07MEF
STBCON	DTBCON	F07VGF
STBRFS	DTBRFS	F07VHF
STBTRS	DTBTRS	F07VEF
STPCON	DTPCON	F07UGF
STPRFS	DTPRFS	F07UHF
STPTRI	DTPTRI	F07UJF
STPTRS	DTPTRS	F07UEF
STRCON	DTRCON	F07TGF
STRRFS	DTRRFS	F07THF
STRTRI	DTRTRI	F07TJF
STRTRS	DTRTRS	F07TEF

Complex Matrices		
LAPACK single precision	LAPACK double precision	NAG
CGBCON	ZGBCON	F07BUF
CGBRFS	ZGBRFS	F07BVF
CGBTRF	ZGBTRF	F07BRF
CGBTRS	ZGBTRS	F07BSF
CGECON	ZGECON	F07AUF
CGERFS	ZGERFS	F07AVF
CGETRF	ZGETRF	F07ARF
CGETRI	ZGETRI	F07AWF
CGETRS	ZGETRS	F07ASF
CHECON	ZHECON	F07MUF
CHERFS	ZHERFS	F07MVF
CHETRF	ZHETRF	F07MRF
CHETRI	ZHETRI	F07MWF
CHETRS	ZHETRS	F07MSF
CHPCON	ZHPCON	F07PUF
CHPRFS	ZHPRFS	F07PVF
CHPTRF	ZHPTRF	F07PRF
CHPTRI	ZHPTRI	F07PWF
CHPTRS	ZHPTRS	F07PSF
CPBCON	ZPBCON	F07HUF
CPBRFS	ZPBRFS	F07HVF
CPBTRF	ZPBTRF	F07HRF
CPBTRS	ZPBTRS	F07HSF
CPOCON	ZPOCON	F07FUF
CPORFS	ZPORFS	F07FVF
CPOTRF	ZPOTRF	F07FRF
CPOTRI	ZPOTRI	F07FWF
CPOTRS	ZPOTRS	F07FSF
CPPCON	ZPPCON	F07GUF
CPPRFS	ZPPRFS	F07GVF
CPPTRF	ZPPTRF	F07GRF
CPPTRI	ZPPTRI	F07GWF
CPPTRS	ZPPTRS	F07GSF
CSPCON	ZSPCON	F07QUF
CSPRFS	ZSPRFS	F07QVF
CSPTRF	ZSPTRF	F07QRF
CSPTRI	ZSPTRI	F07QWF
CSPTRS	ZSPTRS	F07QSF
CSYCON	ZSYCON	F07NUF
CSYRFS	ZSYRFS	F07NVF
CSYTRF	ZSYTRF	F07NRF
CSYTRI	ZSYTRI	F07NWF
CSYTRS	ZSYTRS	F07NSF
CTBCON	ZTBCON	F07VUF
CTBRFS	ZTBRFS	F07VVF
CTBTRS	ZTBTRS	F07VSF
CTPCON	ZTPCON	F07UUF
CTPRFS	ZTPRFS	F07UVF
CTPTRI	ZTPTRI	F07UWF
CTPTRS	ZTPTRS	F07USF
CTRCON	ZTRCON	F07TUF
CTRRFS	ZTRRFS	F07TVF
CTRTRI	ZTRTRI	F07TWF
CTRTRS	ZTRTRS	F07TSF

Table 3

5 References

[1] Anderson E, Bai Z, Bischof C, Demmel J, Dongarra J J, Du Croz J J, Greenbaum A, Hammarling S, McKenney A, Ostrouchov S and Sorensen D (1995) *LAPACK Users' Guide* (2nd Edition) SIAM, Philadelphia

[2] Golub G H and Van Loan C F (1989) *Matrix Computations* Johns Hopkins University Press (2nd Edition), Baltimore

[3] Higham N J (1988) Algorithm 674: Fortran codes for estimating the one-norm of a real or complex matrix, with applications to condition estimation *ACM Trans. Math. Software* **14** 381–396

F07ADF (SGETRF/DGETRF) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07ADF (SGETRF/DGETRF) computes the LU factorization of a real m by n matrix.

2. Specification

```
SUBROUTINE F07ADF (M, N, A, LDA, IPIV, INFO)
ENTRY      sgetrf  (M, N, A, LDA, IPIV, INFO)
INTEGER        M, N, LDA, IPIV(*), INFO
real           A(LDA,*)
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine forms the LU factorization of a real m by n matrix A as $A = PLU$, where P is a permutation matrix, L is lower triangular with unit diagonal elements (lower trapezoidal if $m > n$) and U is upper triangular (upper trapezoidal if $m < n$). Usually A is square ($m = n$), and both L and U are triangular. The routine uses partial pivoting, with row interchanges.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, Chapter 3.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: M – INTEGER. *Input*

On entry: m, the number of rows of the matrix A.

Constraint: M $\geq$ 0.

2: N – INTEGER. *Input*

On entry: n, the number of columns of the matrix A.

Constraint: N $\geq$ 0.

3: A(LDA,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the m by n matrix A.

On exit: A is overwritten by the factors L and U; the unit diagonal elements of L are not stored.

4: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07ADF (SGETRF/DGETRF) is called.

Constraint: LDA $\geq$ max(1,M).

5: IPIV(*) – INTEGER array. *Output*

Note: the dimension of the array IPIV must be at least max(1,min(M,N)).

On exit: the pivot indices. Row i of the matrix A was interchanged with row IPIV(i) for i = 1,2,...,min(m,n).

6: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, u_{ii} is exactly zero. The factorization has been completed but the factor U is exactly singular, and division by zero will occur if it is subsequently used to solve a system of linear equations or to compute A^{-1}.

7. Accuracy

The computed factors L and U are the exact factors of a perturbed matrix $A + E$, where

$$|E| \leq c(\min(m,n))\varepsilon P|L||U|,$$

$c(n)$ is a modest linear function of n, and ε is the ***machine precision***.

8. Further Comments

The total number of floating-point operations is approximately $\frac{2}{3}n^3$ if $m = n$ (the usual case), $\frac{1}{2}n^2(3m-n)$ if $m > n$ and $\frac{1}{2}m^2(3n-m)$ if $m < n$.

A call to this routine with $m = n$ may be followed by calls to the routines:

F07AEF (SGETRS/DGETRS) to solve $AX = B$ or $A^TX = B$;
F07AGF (SGECON/DGECON) to estimate the condition number of A;
F07AJF (SGETRI/DGETRI) to compute the inverse of A.

The complex analogue of this routine is F07ARF (CGETRF/ZGETRF).

9. Example

To compute the LU factorization of the matrix A, where

$$A = \begin{pmatrix} 1.80 & 2.88 & 2.05 & -0.89 \\ 5.25 & -2.95 & -0.95 & -3.80 \\ 1.58 & -2.69 & -2.90 & -1.04 \\ -1.11 & -0.66 & -0.59 & 0.80 \end{pmatrix}.$$

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07ADF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          MMAX, NMAX, LDA
      PARAMETER        (MMAX=8,NMAX=8,LDA=MMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, M, N
*     .. Local Arrays ..
      real             A(LDA,NMAX)
      INTEGER          IPIV(NMAX)
*     .. External Subroutines ..
      EXTERNAL         sgetrf, X04CAF
*     .. Intrinsic Functions ..
      INTRINSIC        MIN
```

```
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07ADF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) M, N
      IF (M.LE.MMAX .AND. N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) ((A(I,J),J=1,N),I=1,M)
*
*        Factorize A
*
         CALL sgetrf(M,N,A,LDA,IPIV,INFO)
*
*        Print details of factorization
*
         WRITE (NOUT,*)
         IFAIL = 0
         CALL X04CAF('General',' ',M,N,A,LDA,'Details of factorization',
     +               IFAIL)
*
*        Print pivot indices
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'IPIV'
         WRITE (NOUT,99999) (IPIV(I),I=1,MIN(M,N))
*
         IF (INFO.NE.0) WRITE (NOUT,*) 'The factor U is singular'
*
      END IF
      STOP
*
99999 FORMAT ((3X,7I11))
      END
```

9.2. Program Data

```
F07ADF Example Program Data
  4  4                          :Values of M and N
  1.80   2.88   2.05  -0.89
  5.25  -2.95  -0.95  -3.80
  1.58  -2.69  -2.90  -1.04
 -1.11  -0.66  -0.59   0.80    :End of matrix A
```

9.3. Program Results

```
 F07ADF Example Program Results

 Details of factorization
              1           2           3           4
  1      5.2500     -2.9500     -0.9500     -3.8000
  2      0.3429      3.8914      2.3757      0.4129
  3      0.3010     -0.4631     -1.5139      0.2948
  4     -0.2114     -0.3299      0.0047      0.1314

 IPIV
             2           2           3           4
```

F07AEF (SGETRS/DGETRS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07AEF (SGETRS/DGETRS) solves a real system of linear equations with multiple right-hand sides, $AX = B$ or $A^TX = B$, where A has been factorized by F07ADF (SGETRF/DGETRF).

2. Specification

```
SUBROUTINE F07AEF (TRANS, N, NRHS, A, LDA, IPIV, B, LDB, INFO)
ENTRY      sgetrs  (TRANS, N, NRHS, A, LDA, IPIV, B, LDB, INFO)
INTEGER          N, NRHS, LDA, IPIV(*), LDB, INFO
real             A(LDA,*), B(LDB,*)
CHARACTER*1      TRANS
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

To solve a real system of linear equations $AX = B$ or $A^TX = B$, this routine must be preceded by a call to F07ADF (SGETRF/DGETRF) which computes the LU factorization of A as $A = PLU$. The solution is computed by forward and backward substitution.

If TRANS = 'N', the solution is computed by solving $PLY = B$ and then $UX = Y$.

If TRANS = 'T' or 'C', the solution is computed by solving $U^TY = B$ and then $L^TP^TX = Y$.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, Chapter 3.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: TRANS – CHARACTER*1. *Input*

On entry: indicates the form of the equations as follows:

if TRANS = 'N', then $AX = B$ is solved for X;

if TRANS = 'T' or 'C', then $A^TX = B$ is solved for X.

Constraint: TRANS = 'N', 'T' or 'C'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS $\geq$ 0.

4: A(LDA,*) – ***real*** array. *Input*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the LU factorization of A, as returned by F07ADF (SGETRF/DGETRF).

5: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07AEF (SGETRS/DGETRS) is called.

Constraint: LDA ≥ max(1,N).

6: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: the pivot indices, as returned by F07ADF (SGETRF/DGETRF).

7: B(LDB,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

On exit: the n by r solution matrix X.

8: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07AEF (SGETRS/DGETRS) is called.

Constraint: LDB ≥ max(1,N).

9: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

For each right-hand side vector b, the computed solution x is the exact solution of a perturbed system of equations $(A+E)x = b$, where

$$|E| \leq c(n)\varepsilon P|L||U|,$$

$c(n)$ is a modest linear function of n, and ε is the ***machine precision***.

If $\hat{x}$ is the true solution, then the computed solution x satisfies a forward error bound of the form

$$\frac{\|x-\hat{x}\|_\infty}{\|x\|_\infty} \leq c(n)\text{cond}(A,x)\varepsilon$$

where $\text{cond}(A,x) = \||A^{-1}||A||x|\|_\infty / \|x\|_\infty \leq \text{cond}(A) = \||A^{-1}||A|\|_\infty \leq \kappa_\infty(A)$. Note that $\text{cond}(A,x)$ can be much smaller than $\text{cond}(A)$, and $\text{cond}(A^T)$ can be much larger (or smaller) than $\text{cond}(A)$.

Forward and backward error bounds can be computed by calling F07AHF (SGERFS/DGERFS), and an estimate for $\kappa_\infty(A)$ can be obtained by calling F07AGF (SGECON/DGECON) with NORM = 'I'.

8. Further Comments

The total number of floating-point operations is approximately $2n^2r$.

This routine may be followed by a call to F07AHF (SGERFS/DGERFS) to refine the solution and return an error estimate.

The complex analogue of this routine is F07ASF (CGETRS/ZGETRS).

9. Example

To solve the system of equations $AX = B$, where

$$A = \begin{pmatrix} 1.80 & 2.88 & 2.05 & -0.89 \\ 5.25 & -2.95 & -0.95 & -3.80 \\ 1.58 & -2.69 & -2.90 & -1.04 \\ -1.11 & -0.66 & -0.59 & 0.80 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 9.52 & 18.47 \\ 24.35 & 2.25 \\ 0.77 & -13.28 \\ -6.22 & -6.21 \end{pmatrix}.$$

Here A is unsymmetric and must first be factorized by F07ADF (SGETRF/DGETRF).

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07AEF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, NRHMAX, LDB
      PARAMETER        (NMAX=8,LDA=NMAX,NRHMAX=NMAX,LDB=NMAX)
      CHARACTER        TRANS
      PARAMETER        (TRANS='N')
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
*     .. Local Arrays ..
      real             A(LDA,NMAX), B(LDB,NRHMAX)
      INTEGER          IPIV(NMAX)
*     .. External Subroutines ..
      EXTERNAL         sgetrf, sgetrs, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07AEF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file
*
         READ (NIN,*) ((A(I,J),J=1,N),I=1,N)
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
*
*        Factorize A
*
         CALL sgetrf(N,N,A,LDA,IPIV,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute solution
*
            CALL sgetrs(TRANS,N,NRHS,A,LDA,IPIV,B,LDB,INFO)
*
*           Print solution
*
            IFAIL = 0
            CALL X04CAF('General',' ',N,NRHS,B,LDB,'Solution(s)',IFAIL)
         ELSE
            WRITE (NOUT,*) 'The factor U is singular'
         END IF
      END IF
      STOP
*
      END
```

9.2. Program Data

```
F07AEF Example Program Data
  4  2                                :Values of N and NRHS
  1.80    2.88    2.05   -0.89
  5.25   -2.95   -0.95   -3.80
  1.58   -2.69   -2.90   -1.04
 -1.11   -0.66   -0.59    0.80       :End of matrix A
  9.52   18.47
 24.35    2.25
  0.77  -13.28
 -6.22   -6.21                        :End of matrix B
```

9.3. Program Results

```
F07AEF Example Program Results

Solution(s)
                  1           2
  1       1.0000      3.0000
  2      -1.0000      2.0000
  3       3.0000      4.0000
  4      -5.0000      1.0000
```

F07AGF (SGECON/DGECON) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07AGF (SGECON/DGECON) estimates the condition number of a real matrix A, where A has been factorized by F07ADF (SGETRF/DGETRF).

2. Specification

```
SUBROUTINE F07AGF (NORM, N, A, LDA, ANORM, RCOND, WORK, IWORK, INFO)
ENTRY      sgecon (NORM, N, A, LDA, ANORM, RCOND, WORK, IWORK, INFO)
INTEGER          N, LDA, IWORK(*), INFO
real             A(LDA,*), ANORM, RCOND, WORK(*)
CHARACTER*1      NORM
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine estimates the condition number of a real matrix A, in either the 1-norm or the infinity-norm:

$$\kappa_1(A) = \|A\|_1\|A^{-1}\|_1 \text{ or } \kappa_\infty(A) = \|A\|_\infty\|A^{-1}\|_\infty.$$

Note that $\kappa_\infty(A) = \kappa_1(A^T)$.

Because the condition number is infinite if A is singular, the routine actually returns an estimate of the **reciprocal** of the condition number.

The routine should be preceded by a call to F06RAF to compute $\|A\|_1$ or $\|A\|_\infty$, and a call to F07ADF (SGETRF/DGETRF) to compute the LU factorization of A. The routine then uses Higham's implementation of Hager's method [1] to estimate $\|A^{-1}\|_1$ or $\|A^{-1}\|_\infty$.

4. References

[1] HIGHAM, N.J.
FORTRAN Codes for Estimating the One-norm of a Real or Complex Matrix, with Applications to Condition Estimation.
ACM Trans. Math. Softw., 14, pp. 381-396, 1988.

5. Parameters

1: NORM – CHARACTER*1. *Input*

On entry: indicates whether $\kappa_1(A)$ or $\kappa_\infty(A)$ is estimated as follows:

if NORM = '1' or 'O', then $\kappa_1(A)$ is estimated;

if NORM = 'I', then $\kappa_\infty(A)$ is estimated.

Constraint: NORM = '1', 'O' or 'I'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: A(LDA,*) – ***real*** array. *Input*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the LU factorization of A, as returned by F07ADF (SGETRF/DGETRF).

4: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07AGF (SGECON/DGECON) is called.

Constraint: LDA ≥ max(1,N).

5: ANORM – ***real***. *Input*

On entry: if NORM = '1' or 'O', the 1-norm of the **original** matrix A; if NORM = 'I', the infinity-norm of the **original** matrix A. ANORM may be computed by calling F06RAF with the same value for the parameter NORM. ANORM must be computed either **before** calling F07ADF (SGETRF/DGETRF) or else from a **copy** of the original matrix A.

Constraint: ANORM ≥ 0.0.

6: RCOND – ***real***. *Output*

On exit: an estimate of the reciprocal of the condition number of A. RCOND is set to zero if exact singularity is detected or the estimate underflows. If RCOND is less than ***machine precision***, then A is singular to working precision.

7: WORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,4*N).

8: IWORK(*) – INTEGER array. *Workspace*

Note: the dimension of the array IWORK must be at least max(1,N).

9: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed estimate RCOND is never less than the true value ρ, and in practice is nearly always less than 10ρ, although examples can be constructed where RCOND is much larger.

8. Further Comments

A call to this routine involves solving a number of systems of linear equations of the form $Ax = b$ or $A^Tx = b$; the number is usually 4 or 5 and never more than 11. Each solution involves approximately $2n^2$ floating-point operations but takes considerably longer than a call to F07AEF (SGETRS/DGETRS) with 1 right-hand side, because extra care is taken to avoid overflow when A is approximately singular.

The complex analogue of this routine is F07AUF (CGECON/ZGECON).

9. Example

To estimate the condition number in the 1-norm of the matrix A, where

$$A = \begin{pmatrix} 1.80 & 2.88 & 2.05 & -0.89 \\ 5.25 & -2.95 & -0.95 & -3.80 \\ 1.58 & -2.69 & -2.90 & -1.04 \\ -1.11 & -0.66 & -0.59 & 0.80 \end{pmatrix}.$$

Here A is unsymmetric and must first be factorized by F07ADF (SGETRF/DGETRF). The true condition number in the 1-norm is 152.16.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07AGF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA
      PARAMETER        (NMAX=8,LDA=NMAX)
      CHARACTER        NORM
      PARAMETER        (NORM='1')
*     .. Local Scalars ..
      real             ANORM, RCOND
      INTEGER          I, INFO, J, N
*     .. Local Arrays ..
      real             A(LDA,NMAX), WORK(4*NMAX)
      INTEGER          IPIV(NMAX), IWORK(NMAX)
*     .. External Functions ..
      real             F06RAF, X02AJF
      EXTERNAL         F06RAF, X02AJF
*     .. External Subroutines ..
      EXTERNAL         sgecon, sgetrf
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07AGF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) ((A(I,J),J=1,N),I=1,N)
*
*        Compute norm of A
*
         ANORM = F06RAF(NORM,N,N,A,LDA,WORK)
*
*        Factorize A
*
         CALL sgetrf(N,N,A,LDA,IPIV,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Estimate condition number
*
            CALL sgecon(NORM,N,A,LDA,ANORM,RCOND,WORK,IWORK,INFO)
*
            IF (RCOND.GE.X02AJF()) THEN
               WRITE (NOUT,99999) 'Estimate of condition number =',
     +           1.0e0/RCOND
            ELSE
               WRITE (NOUT,*) 'A is singular to working precision'
            END IF
         ELSE
            WRITE (NOUT,*) 'The factor U is singular'
         END IF
      END IF
      STOP
*
99999 FORMAT (1X,A,1P,e10.2)
      END
```

9.2. Program Data

```
F07AGF Example Program Data
  4                                   :Value of N
  1.80    2.88    2.05   -0.89
  5.25   -2.95   -0.95   -3.80
  1.58   -2.69   -2.90   -1.04
 -1.11   -0.66   -0.59    0.80       :End of matrix A
```

9.3. Program Results

```
F07AGF Example Program Results

Estimate of condition number =  1.52E+02
```

F07AHF (SGERFS/DGERFS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07AHF (SGERFS/DGERFS) returns error bounds for the solution of a real system of linear equations with multiple right-hand sides, $AX = B$ or $A^TX = B$. It improves the solution by iterative refinement, in order to reduce the backward error as much as possible.

2. Specification

```
 SUBROUTINE F07AHF (TRANS, N, NRHS, A, LDA, AF, LDAF, IPIV, B, LDB, X,
1                   LDX, FERR, BERR, WORK, IWORK, INFO)
 ENTRY       sgerfs  (TRANS, N, NRHS, A, LDA, AF, LDAF, IPIV, B, LDB, X,
1                   LDX, FERR, BERR, WORK, IWORK, INFO)
 INTEGER          N, NRHS, LDA, LDAF, IPIV(*), LDB, LDX, IWORK(*), INFO
 real             A(LDA,*), AF(LDAF,*), B(LDB,*), X(LDX,*), FERR(*),
1                 BERR(*), WORK(*)
 CHARACTER*1      TRANS
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine returns the backward errors and estimated bounds on the forward errors for the solution of a real system of linear equations with multiple right-hand sides $AX = B$ or $A^TX = B$. The routine handles each right-hand side vector (stored as a column of the matrix B) independently, so we describe the function of the routine in terms of a single right-hand side b and solution x.

Given a computed solution x, the routine computes the *component-wise backward error* β. This is the size of the smallest relative perturbation in each element of A and b such that x is the exact solution of a perturbed system

$$(A+\delta A)x = b + \delta b$$

$$|\delta a_{ij}| \leq \beta |a_{ij}| \text{ and } |\delta b_i| \leq \beta |b_i|.$$

Then the routine estimates a bound for the *component-wise forward error* in the computed solution, defined by:

$$\max_i |x_i - \hat{x}_i| / \max_i |x_i|$$

where $\hat{x}$ is the true solution.

For details of the method, see the the Chapter Introduction.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: TRANS – CHARACTER*1. *Input*

On entry: indicates the form of the linear equations for which X is the computed solution as follows:

if TRANS = 'N', then the linear equations are of the form $AX = B$;

if TRANS = 'T' or 'C', then the linear equations are of the form $A^TX = B$.

Constraint: TRANS = 'N', 'T' or 'C'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS ≥ 0.

4: A(LDA,*) – ***real*** array. *Input*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the n by n original matrix A as supplied to F07ADF (SGETRF/DGETRF).

5: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07AHF (SGERFS/DGERFS) is called.

Constraint: LDA ≥ max(1,N).

6: AF(LDAF,*) – ***real*** array. *Input*

Note: the second dimension of the array AF must be at least max(1,N).

On entry: the LU factorization of A, as returned by F07ADF (SGETRF/DGETRF).

7: LDAF – INTEGER. *Input*

On entry: the first dimension of the array AF as declared in the (sub)program from which F07AHF (SGERFS/DGERFS) is called.

Constraint: LDAF ≥ max(1,N).

8: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: the pivot indices, as returned by F07ADF (SGETRF/DGETRF).

9: B(LDB,*) – ***real*** array. *Input*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

10: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07AHF (SGERFS/DGERFS) is called.

Constraint: LDB ≥ max(1,N).

11: X(LDX,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array X must be at least max(1,NRHS).

On entry: the n by r solution matrix X, as returned by F07AEF (SGETRS/DGETRS).

On exit: the improved solution matrix X.

12: LDX – INTEGER. *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which F07AHF (SGERFS/DGERFS) is called.

Constraint: LDX ≥ max(1,N).

13: FERR(*) – ***real*** array. *Output*

Note: the dimension of the array FERR must be at least max(1,NRHS).

On exit: FERR(j) contains an estimated error bound for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

14: BERR(*) – ***real*** array. *Output*

Note: the dimension of the array BERR must be at least max(1,NRHS).

On exit: BERR(j) contains the component-wise backward error bound β for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

15: WORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,3*N).

16: IWORK(*) – INTEGER array. *Workspace*

Note: the dimension of the array IWORK must be at least max(1,N).

17: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The bounds returned in FERR are not rigorous, because they are estimated, not computed exactly; but in practice they almost always overestimate the actual error.

8. Further Comments

For each right-hand side, computation of the backward error involves a minimum of $4n^2$ floating-point operations. Each step of iterative refinement involves an additional $6n^2$ operations. At most 5 steps of iterative refinement are performed, but usually only 1 or 2 steps are required.

Estimating the forward error involves solving a number of systems of linear equations of the form $Ax = b$ or $A^Tx = b$; the number is usually 4 or 5 and never more than 11. Each solution involves approximately $2n^2$ operations.

The complex analogue of this routine is F07AVF (CGERFS/ZGERFS).

9. Example

To solve the system of equations $AX = B$ using iterative refinement and to compute the forward and backward error bounds, where

$$A = \begin{pmatrix} 1.80 & 2.88 & 2.05 & -0.89 \\ 5.25 & -2.95 & -0.95 & -3.80 \\ 1.58 & -2.69 & -2.90 & -1.04 \\ -1.11 & -0.66 & -0.59 & 0.80 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 9.52 & 18.47 \\ 24.35 & 2.25 \\ 0.77 & -13.28 \\ -6.22 & -6.21 \end{pmatrix}.$$

Here A is unsymmetric and must first be factorized by F07ADF (SGETRF/DGETRF).

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07AHF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, NRHMAX, LDA, LDAF, LDB, LDX
      PARAMETER        (NMAX=8,NRHMAX=NMAX,LDA=NMAX,LDAF=NMAX,LDB=NMAX,
     +                 LDX=NMAX)
      CHARACTER        TRANS
      PARAMETER        (TRANS='N')
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
*     .. Local Arrays ..
      real             A(LDA,NMAX), AF(LDAF,NMAX), B(LDB,NRHMAX),
     +                 BERR(NRHMAX), FERR(NRHMAX), WORK(3*NMAX),
     +                 X(LDX,NMAX)
      INTEGER          IPIV(NMAX), IWORK(NMAX)
*     .. External Subroutines ..
      EXTERNAL         F06QFF, sgerfs, sgetrf, sgetrs, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07AHF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file, and copy A to AF and B to X
*
         READ (NIN,*) ((A(I,J),J=1,N),I=1,N)
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
*
         CALL F06QFF('General',N,N,A,LDA,AF,LDAF)
*
         CALL F06QFF('General',N,NRHS,B,LDB,X,LDX)
*
*        Factorize A in the array AF
*
         CALL sgetrf(N,N,AF,LDAF,IPIV,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute solution in the array X
*
            CALL sgetrs(TRANS,N,NRHS,AF,LDAF,IPIV,X,LDX,INFO)
*
*           Improve solution, and compute backward errors and
*           estimated bounds on the forward errors
*
            CALL sgerfs(TRANS,N,NRHS,A,LDA,AF,LDAF,IPIV,B,LDB,X,LDX,
     +                  FERR,BERR,WORK,IWORK,INFO)
*
*           Print solution
*
            IFAIL = 0
*
            CALL X04CAF('General',' ',N,NRHS,X,LDX,'Solution(s)',IFAIL)
*
            WRITE (NOUT,*)
            WRITE (NOUT,*) 'Backward errors (machine-dependent)'
            WRITE (NOUT,99999) (BERR(J),J=1,NRHS)
            WRITE (NOUT,*)
     +        'Estimated forward error bounds (machine-dependent)'
            WRITE (NOUT,99999) (FERR(J),J=1,NRHS)
```

```
         ELSE
            WRITE (NOUT,*) 'The factor U is singular'
         END IF
      END IF
      STOP
*
99999 FORMAT ((3X,1P,7e11.1))
      END
```

9.2. Program Data

```
F07AHF Example Program Data
  4  2                              :Values of N and NRHS
  1.80   2.88   2.05  -0.89
  5.25  -2.95  -0.95  -3.80
  1.58  -2.69  -2.90  -1.04
 -1.11  -0.66  -0.59   0.80         :End of matrix A
  9.52  18.47
 24.35   2.25
  0.77 -13.28
 -6.22  -6.21                       :End of matrix B
```

9.3. Program Results

```
 F07AHF Example Program Results

 Solution(s)
                  1           2
  1       1.0000      3.0000
  2      -1.0000      2.0000
  3       3.0000      4.0000
  4      -5.0000      1.0000

 Backward errors (machine-dependent)
       2.3E-17     7.5E-17
 Estimated forward error bounds (machine-dependent)
       2.4E-14     3.3E-14
```

F07AJF (SGETRI/DGETRI) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07AJF (SGETRI/DGETRI) computes the inverse of a real matrix A, where A has been factorized by F07ADF (SGETRF/DGETRF).

2. Specification

```
SUBROUTINE F07AJF (N, A, LDA, IPIV, WORK, LWORK, INFO)
ENTRY      sgetri  (N, A, LDA, IPIV, WORK, LWORK, INFO)
INTEGER        N, LDA, IPIV(*), LWORK, INFO
real           A(LDA,*), WORK(LWORK)
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

To compute the inverse of a real matrix A, the routine must be preceded by a call to F07ADF (SGETRF/DGETRF), which computes the LU factorization of A as $A = PLU$. The inverse of A is computed by forming U^{-1} and then solving the equation $XPL = U^{-1}$ for X.

4. References

[1] DU CROZ, J.J. and HIGHAM, N.J.
Stability of Methods for Matrix Inversion.
LAPACK Working Note No. 27, University of Tennessee, Knoxville, 1990.

5. Parameters

1: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

2: A(LDA,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the LU factorization of A, as returned by F07ADF (SGETRF/DGETRF).

On exit: the factorization is overwritten by the n by n matrix A^{-1}.

3: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07AJF (SGETRI/DGETRI) is called.

Constraint: LDA ≥ max(1,N).

4: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: the pivot indices, as returned by F07ADF (SGETRF/DGETRF).

5: WORK(LWORK) – ***real*** array. *Workspace*

On exit: if INFO = 0, WORK(1) contains the minimum value of LWORK required for optimum performance.

6: LWORK – INTEGER. *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F07AJF (SGETRI/DGETRI) is called.

Suggested value: for optimum performance LWORK should be at least N×*nb*, where *nb* is the ***blocksize***.

Constraint: LWORK ≥ N.

7: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, the ith diagonal element of the factor U is zero, U is singular, and the inverse of A cannot be computed.

7. Accuracy

The computed inverse X satisfies a bound of the form:

$$|XA-I| \leq c(n)\,\varepsilon|X|P|L||U|,$$

where $c(n)$ is a modest linear function of n, and ε is the ***machine precision***.

Note: a similar bound for $|AX-I|$ cannot be guaranteed, although it is almost always satisfied. See Du Croz and Higham [1].

8. Further Comments

The total number of floating-point operations is approximately $\frac{4}{3}n^3$.

The complex analogue of this routine is F07AWF (CGETRI/ZGETRI).

9. Example

To compute the inverse of the matrix A, where

$$A = \begin{pmatrix} 1.80 & 2.88 & 2.05 & -0.89 \\ 5.25 & -2.95 & -0.95 & -3.80 \\ 1.58 & -2.69 & -2.90 & -1.04 \\ -1.11 & -0.66 & -0.59 & 0.80 \end{pmatrix}.$$

Here A is unsymmetric and must first be factorized by F07ADF (SGETRF/DGETRF).

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07AJF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, LWORK
      PARAMETER        (NMAX=8,LDA=NMAX,LWORK=64*NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
*     .. Local Arrays ..
      real             A(LDA,NMAX), WORK(LWORK)
      INTEGER          IPIV(NMAX)
```

```
*     .. External Subroutines ..
      EXTERNAL          sgetrf, sgetri, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07AJF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) ((A(I,J),J=1,N),I=1,N)
*
*        Factorize A
*
         CALL sgetrf(N,N,A,LDA,IPIV,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute inverse of A
*
            CALL sgetri(N,A,LDA,IPIV,WORK,LWORK,INFO)
*
*           Print inverse
*
            IFAIL = 0
            CALL X04CAF('General',' ',N,N,A,LDA,'Inverse',IFAIL)
         ELSE
            WRITE (NOUT,*) 'The factor U is singular'
         END IF
      END IF
      STOP
*
      END
```

9.2. Program Data

```
F07AJF Example Program Data
  4                                 :Value of N
 1.80   2.88   2.05  -0.89
 5.25  -2.95  -0.95  -3.80
 1.58  -2.69  -2.90  -1.04
-1.11  -0.66  -0.59   0.80          :End of matrix A
```

9.3. Program Results

```
F07AJF Example Program Results

Inverse
                1          2          3          4
 1         1.7720     0.5757     0.0843     4.8155
 2        -0.1175    -0.4456     0.4114    -1.7126
 3         0.1799     0.4527    -0.6676     1.4824
 4         2.4944     0.7650    -0.0360     7.6119
```

F07ARF (CGETRF/ZGETRF) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07ARF (CGETRF/ZGETRF) computes the *LU* factorization of a complex *m* by *n* matrix.

2. Specification

```
SUBROUTINE F07ARF (M, N, A, LDA, IPIV, INFO)
ENTRY      cgetrf  (M, N, A, LDA, IPIV, INFO)
INTEGER        M, N, LDA, IPIV(*), INFO
complex        A(LDA,*)
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine forms the *LU* factorization of a complex *m* by *n* matrix *A* as $A = PLU$, where *P* is a permutation matrix, *L* is lower triangular with unit diagonal elements (lower trapezoidal if $m > n$) and *U* is upper triangular (upper trapezoidal if $m < n$). Usually *A* is square ($m = n$), and both *L* and *U* are triangular. The routine uses partial pivoting, with row interchanges.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, Chapter 3.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: M – INTEGER. *Input*

On entry: *m*, the number of rows of the matrix *A*.

Constraint: $M \geq 0$.

2: N – INTEGER. *Input*

On entry: *n*, the number of columns of the matrix *A*.

Constraint: $N \geq 0$.

3: A(LDA,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the *m* by *n* matrix *A*.

On exit: *A* is overwritten by the factors *L* and *U*; the unit diagonal elements of *L* are not stored.

4: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07ARF (CGETRF/ZGETRF) is called.

Constraint: LDA ≥ max(1,M).

5: IPIV(*) – INTEGER array. *Output*

Note: the dimension of the array IPIV must be at least max(1,min(M,N)).

On exit: the pivot indices. Row *i* of the matrix *A* was interchanged with row IPIV(*i*), for $i = 1,2,...,\min(m,n)$.

6: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, u_{ii} is exactly zero. The factorization has been completed but the factor U is exactly singular, and division by zero will occur if it is subsequently used to solve a system of linear equations or to compute A^{-1}.

7. Accuracy

The computed factors L and U are the exact factors of a perturbed matrix $A + E$, where

$$|E| \leq c(\min(m,n))\varepsilon P|L||U|,$$

$c(n)$ is a modest linear function of n, and ε is the ***machine precision***.

8. Further Comments

The total number of real floating-point operations is approximately $\frac{8}{3}n^3$ if $m = n$ (the usual case), $\frac{4}{3}n^2(3m-n)$ if $m > n$ and $\frac{4}{3}m^2(3n-m)$ if $m < n$.

A call to this routine with $m = n$ may be followed by calls to the routines:

F07ASF (CGETRS/ZGETRS) to solve $AX = B$, $A^TX = B$ or $A^HX = B$;
F07AUF (CGECON/ZGECON) to estimate the condition number of A;
F07AWF (CGETRI/ZGETRI) to compute the inverse of A.

The real analogue of this routine is F07ADF (SGETRF/DGETRF).

9. Example

To compute the LU factorization of the matrix A, where

$$A = \begin{pmatrix} -1.34 + 2.55i & 0.28 + 3.17i & -6.39 - 2.20i & 0.72 - 0.92i \\ -0.17 - 1.41i & 3.31 - 0.15i & -0.15 + 1.34i & 1.29 + 1.38i \\ -3.29 - 2.39i & -1.91 + 4.42i & -0.14 - 1.35i & 1.72 + 1.35i \\ 2.41 + 0.39i & -0.56 + 1.47i & -0.83 - 0.69i & -1.96 + 0.67i \end{pmatrix}.$$

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07ARF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          MMAX, NMAX, LDA
      PARAMETER        (MMAX=8,NMAX=8,LDA=MMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, M, N
*     .. Local Arrays ..
      complex          A(LDA,NMAX)
      INTEGER          IPIV(NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         cgetrf, X04DBF
*     .. Intrinsic Functions ..
      INTRINSIC        MIN
```

```
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07ARF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) M, N
      IF (M.LE.MMAX .AND. N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) ((A(I,J),J=1,N),I=1,M)
*
*        Factorize A
*
         CALL cgetrf(M,N,A,LDA,IPIV,INFO)
*
*        Print details of factorization
*
         WRITE (NOUT,*)
         IFAIL = 0
         CALL X04DBF('General',' ',M,N,A,LDA,'Bracketed','F7.4',
     +               'Details of factorization','Integer',RLABS,
     +               'Integer',CLABS,80,0,IFAIL)
*
*        Print pivot indices
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'IPIV'
         WRITE (NOUT,99999) (IPIV(I),I=1,MIN(M,N))
*
         IF (INFO.NE.0) WRITE (NOUT,*) 'The factor U is singular'
*
      END IF
      STOP
*
99999 FORMAT ((1X,I12,3I18))
      END
```

9.2. Program Data

```
F07ARF Example Program Data
  4  4                                                    :Values of M and N
 (-1.34, 2.55) ( 0.28, 3.17) (-6.39,-2.20) ( 0.72,-0.92)
 (-0.17,-1.41) ( 3.31,-0.15) (-0.15, 1.34) ( 1.29, 1.38)
 (-3.29,-2.39) (-1.91, 4.42) (-0.14,-1.35) ( 1.72, 1.35)
 ( 2.41, 0.39) (-0.56, 1.47) (-0.83,-0.69) (-1.96, 0.67) :End of matrix A
```

9.3. Program Results

```
F07ARF Example Program Results

Details of factorization
                    1                   2                   3                   4
 1  (-3.2900,-2.3900) (-1.9100, 4.4200) (-0.1400,-1.3500) ( 1.7200, 1.3500)
 2  ( 0.2376, 0.2560) ( 4.8952,-0.7114) (-0.4623, 1.6966) ( 1.2269, 0.6190)
 3  (-0.1020,-0.7010) (-0.6691, 0.3689) (-5.1414,-1.1300) ( 0.9983, 0.3850)
 4  (-0.5359, 0.2707) (-0.2040, 0.8601) ( 0.0082, 0.1211) ( 0.1482,-0.1252)

IPIV
           3                   2                   3                   4
```

F07ASF (CGETRS/ZGETRS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07ASF (CGETRS/ZGETRS) solves a complex system of linear equations with multiple right-hand sides, $AX = B$, $A^TX = B$ or $A^HX = B$, where A has been factorized by F07ARF (CGETRF/ZGETRF).

2. Specification

```
SUBROUTINE F07ASF (TRANS, N, NRHS, A, LDA, IPIV, B, LDB, INFO)
ENTRY      cgetrs  (TRANS, N, NRHS, A, LDA, IPIV, B, LDB, INFO)
INTEGER          N, NRHS, LDA, IPIV(*), LDB, INFO
complex          A(LDA,*), B(LDB,*)
CHARACTER*1      TRANS
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

To solve a complex system of linear equations $AX = B$, $A^TX = B$ or $A^HX = B$, this routine must be preceded by a call to F07ARF (CGETRF/ZGETRF) which computes the LU factorization of A as $A = PLU$. The solution is computed by forward and backward substitution.

If TRANS = 'N', the solution is computed by solving $PLY = B$ and then $UX = Y$.

If TRANS = 'T', the solution is computed by solving $U^TY = B$ and then $L^TP^TX = Y$.

If TRANS = 'C', the solution is computed by solving $U^HY = B$ and then $L^HP^TX = Y$.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, Chapter 3.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: TRANS – CHARACTER*1. *Input*

On entry: indicates the form of the equations as follows:

if TRANS = 'N', then $AX = B$ is solved for X;

if TRANS = 'T', then $A^TX = B$ is solved for X;

if TRANS = 'C', then $A^HX = B$ is solved for X.

Constraint: TRANS = 'N', 'T' or 'C'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS $\geq$ 0.

4: A(LDA,*) – ***complex*** array. *Input*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the LU factorization of A, as returned by F07ARF (CGETRF/ZGETRF).

5: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07ASF (CGETRS/ZGETRS) is called.

Constraint: LDA ≥ max(1,N).

6: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: the pivot indices, as returned by F07ARF (CGETRF/ZGETRF).

7: B(LDB,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

On exit: the n by r solution matrix X.

8: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07ASF (CGETRS/ZGETRS) is called.

Constraint: LDB ≥ max(1,N).

9: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

For each right-hand side vector b, the computed solution x is the exact solution of a perturbed system of equations $(A+E)x = b$, where

$$|E| \leq c(n)\varepsilon P|L||U|,$$

$c(n)$ is a modest linear function of n, and ε is the ***machine precision***.

If $\hat{x}$ is the true solution, then the computed solution x satisfies a forward error bound of the form

$$\frac{\|x-\hat{x}\|_\infty}{\|x\|_\infty} \leq c(n)\text{cond}(A,x)\varepsilon$$

where $\text{cond}(A,x) = \||A^{-1}||A||x|\|_\infty / \|x\|_\infty \leq \text{cond}(A) = \||A^{-1}||A|\|_\infty \leq \kappa_\infty(A)$. Note that $\text{cond}(A,x)$ can be much smaller than $\text{cond}(A)$, and $\text{cond}(A^H)$ (which is the same as $\text{cond}(A^T)$) can be much larger (or smaller) than $\text{cond}(A)$.

Forward and backward error bounds can be computed by calling F07AVF (CGERFS/ZGERFS), and an estimate for $\kappa_\infty(A)$ can be obtained by calling F07AUF (CGECON/ZGECON) with NORM = 'I'.

8. Further Comments

The total number of real floating-point operations is approximately $8n^2r$.

This routine may be followed by a call to F07AVF (CGERFS/ZGERFS) to refine the solution and return an error estimate.

The real analogue of this routine is F07AEF (SGETRS/DGETRS).

9. Example

To solve the system of equations $AX = B$, where

$$A = \begin{pmatrix} -1.34 + 2.55i & 0.28 + 3.17i & -6.39 - 2.20i & 0.72 - 0.92i \\ -0.17 - 1.41i & 3.31 - 0.15i & -0.15 + 1.34i & 1.29 + 1.38i \\ -3.29 - 2.39i & -1.91 + 4.42i & -0.14 - 1.35i & 1.72 + 1.35i \\ 2.41 + 0.39i & -0.56 + 1.47i & -0.83 - 0.69i & -1.96 + 0.67i \end{pmatrix} \text{ and}$$

$$B = \begin{pmatrix} 26.26 + 51.78i & 31.32 - 6.70i \\ 6.43 - 8.68i & 15.86 - 1.42i \\ -5.75 + 25.31i & -2.15 + 30.19i \\ 1.16 + 2.57i & -2.56 + 7.55i \end{pmatrix}.$$

Here A is unsymmetric and must first be factorized by F07ARF (CGETRF/ZGETRF).

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07ASF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, NRHMAX, LDB
      PARAMETER        (NMAX=8,LDA=NMAX,NRHMAX=NMAX,LDB=NMAX)
      CHARACTER        TRANS
      PARAMETER        (TRANS='N')
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
*     .. Local Arrays ..
      complex          A(LDA,NMAX), B(LDB,NRHMAX)
      INTEGER          IPIV(NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         cgetrf, cgetrs, X04DBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07ASF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file
*
         READ (NIN,*) ((A(I,J),J=1,N),I=1,N)
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
*
*        Factorize A
*
         CALL cgetrf(N,N,A,LDA,IPIV,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute solution
*
            CALL cgetrs(TRANS,N,NRHS,A,LDA,IPIV,B,LDB,INFO)
*
```

```
*           Print solution
*
            IFAIL = 0
            CALL X04DBF('General',' ',N,NRHS,B,LDB,'Bracketed','F7.4',
     +                  'Solution(s)','Integer',RLABS,'Integer',CLABS,
     +                  80,0,IFAIL)
         ELSE
            WRITE (NOUT,*) 'The factor U is singular'
         END IF
       END IF
       STOP
*
       END
```

9.2. Program Data

```
F07ASF Example Program Data
  4  2                                                        :Values of N and NRHS
 (-1.34, 2.55) ( 0.28, 3.17) (-6.39,-2.20) ( 0.72,-0.92)
 (-0.17,-1.41) ( 3.31,-0.15) (-0.15, 1.34) ( 1.29, 1.38)
 (-3.29,-2.39) (-1.91, 4.42) (-0.14,-1.35) ( 1.72, 1.35)
 ( 2.41, 0.39) (-0.56, 1.47) (-0.83,-0.69) (-1.96, 0.67) :End of matrix A
 (26.26, 51.78) (31.32, -6.70)
 ( 6.43, -8.68) (15.86, -1.42)
 (-5.75, 25.31) (-2.15, 30.19)
 ( 1.16,  2.57) (-2.56,  7.55)                            :End of matrix B
```

9.3. Program Results

```
F07ASF Example Program Results

Solution(s)
                      1                      2
  1   ( 1.0000, 1.0000) (-1.0000,-2.0000)
  2   ( 2.0000,-3.0000) ( 5.0000, 1.0000)
  3   (-4.0000,-5.0000) (-3.0000, 4.0000)
  4   ( 0.0000, 6.0000) ( 2.0000,-3.0000)
```

F07AUF (CGECON/ZGECON) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07AUF (CGECON/ZGECON) estimates the condition number of a complex matrix A, where A has been factorized by F07ARF (CGETRF/ZGETRF).

2. Specification

```
SUBROUTINE F07AUF (NORM, N, A, LDA, ANORM, RCOND, WORK, RWORK, INFO)
ENTRY      cgecon  (NORM, N, A, LDA, ANORM, RCOND, WORK, RWORK, INFO)
INTEGER          N, LDA, INFO
real             ANORM, RCOND, RWORK(*)
complex          A(LDA,*), WORK(*)
CHARACTER*1      NORM
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine estimates the condition number of a complex matrix A, in either the 1-norm or the infinity-norm:

$$\kappa_1(A) = \|A\|_1\|A^{-1}\|_1 \quad \text{or} \quad \kappa_\infty(A) = \|A\|_\infty\|A^{-1}\|_\infty.$$

Note that $\kappa_\infty(A) = \kappa_1(A^H)$.

Because the condition number is infinite if A is singular, the routine actually returns an estimate of the **reciprocal** of the condition number.

The routine should be preceded by a call to F06UAF to compute $\|A\|_1$ or $\|A\|_\infty$, and a call to F07ARF (CGETRF/ZGETRF) to compute the LU factorization of A. The routine then uses Higham's implementation of Hager's method [1] to estimate $\|A^{-1}\|_1$ or $\|A^{-1}\|_\infty$.

4. References

[1] HIGHAM, N.J.
FORTRAN Codes for Estimating the One-norm of a Real or Complex Matrix, with Applications to Condition Estimation.
ACM Trans. Math. Softw., 14, pp. 381-396, 1988.

5. Parameters

1: NORM – CHARACTER*1. *Input*

On entry: indicates whether $\kappa_1(A)$ or $\kappa_\infty(A)$ is estimated as follows:

if NORM = '1' or 'O', then $\kappa_1(A)$ is estimated;

if NORM = 'I', then $\kappa_\infty(A)$ is estimated.

Constraint: NORM = '1', 'O' or 'I'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: A(LDA,*) – ***complex*** array. *Input*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the LU factorization of A, as returned by F07ARF (CGETRF/ZGETRF).

4: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07AUF (CGECON/ZGECON) is called.

Constraint: LDA ≥ max(1,N).

5: ANORM – ***real***. *Input*

On entry: if NORM = '1' or 'O', the 1-norm of the **original** matrix A; if NORM = 'I', the infinity-norm of the **original** matrix A. ANORM may be computed by calling F06UAF with the same value for the parameter NORM. ANORM must be computed either **before** calling F07ARF (CGETRF/ZGETRF) or else from a **copy** of the original matrix A.

Constraint: ANORM ≥ 0.0.

6: RCOND – ***real***. *Output*

On exit: an estimate of the reciprocal of the condition number of A. RCOND is set to zero if exact singularity is detected or the estimate underflows. If RCOND is less than ***machine precision***, then A is singular to working precision.

7: WORK(*) – ***complex*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,2*N).

8: RWORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array RWORK must be at least max(1,2*N).

9: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed estimate RCOND is never less than the true value ρ, and in practice is nearly always less than 10ρ, although examples can be constructed where RCOND is much larger.

8. Further Comments

A call to this routine involves solving a number of systems of linear equations of the form $Ax = b$ or $A^H x = b$; the number is usually 5 and never more than 11. Each solution involves approximately $8n^2$ real floating-point operations but takes considerably longer than a call to F07ASF (CGETRS/ZGETRS) with 1 right-hand side, because extra care is taken to avoid overflow when A is approximately singular.

The real analogue of this routine is F07AGF (SGECON/DGECON).

9. Example

To estimate the condition number in the 1-norm of the matrix A, where

$$A = \begin{pmatrix} -1.34 + 2.55i & 0.28 + 3.17i & -6.39 - 2.20i & 0.72 - 0.92i \\ -0.17 - 1.41i & 3.31 - 0.15i & -0.15 + 1.34i & 1.29 + 1.38i \\ -3.29 - 2.39i & -1.91 + 4.42i & -0.14 - 1.35i & 1.72 + 1.35i \\ 2.41 + 0.39i & -0.56 + 1.47i & -0.83 - 0.69i & -1.96 + 0.67i \end{pmatrix}.$$

Here A is unsymmetric and must first be factorized by F07ARF (CGETRF/ZGETRF). The true condition number in the 1-norm is 231.86.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07AUF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA
      PARAMETER        (NMAX=8,LDA=NMAX)
      CHARACTER        NORM
      PARAMETER        (NORM='1')
*     .. Local Scalars ..
      real             ANORM, RCOND
      INTEGER          I, INFO, J, N
*     .. Local Arrays ..
      complex          A(LDA,NMAX), WORK(2*NMAX)
      real             RWORK(2*NMAX)
      INTEGER          IPIV(NMAX)
*     .. External Functions ..
      real             F06UAF, X02AJF
      EXTERNAL         F06UAF, X02AJF
*     .. External Subroutines ..
      EXTERNAL         cgecon, cgetrf
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07AUF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) ((A(I,J),J=1,N),I=1,N)
*
*        Compute norm of A
*
         ANORM = F06UAF(NORM,N,N,A,LDA,RWORK)
*
*        Factorize A
*
         CALL cgetrf(N,N,A,LDA,IPIV,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Estimate condition number
*
            CALL cgecon(NORM,N,A,LDA,ANORM,RCOND,WORK,RWORK,INFO)
*
            IF (RCOND.GE.X02AJF()) THEN
               WRITE (NOUT,99999) 'Estimate of condition number =',
     +           1.0e0/RCOND
            ELSE
               WRITE (NOUT,*) 'A is singular to working precision'
            END IF
         ELSE
            WRITE (NOUT,*) 'The factor U is singular'
         END IF
      END IF
      STOP
*
99999 FORMAT (1X,A,1P,e10.2)
      END
```

9.2. Program Data

```
F07AUF Example Program Data
  4                                                                  :Value of N
 (-1.34, 2.55) ( 0.28, 3.17) (-6.39,-2.20) ( 0.72,-0.92)
 (-0.17,-1.41) ( 3.31,-0.15) (-0.15, 1.34) ( 1.29, 1.38)
 (-3.29,-2.39) (-1.91, 4.42) (-0.14,-1.35) ( 1.72, 1.35)
 ( 2.41, 0.39) (-0.56, 1.47) (-0.83,-0.69) (-1.96, 0.67)   :End of matrix A
```

9.3. Program Results

```
F07AUF Example Program Results

Estimate of condition number =  1.50E+02
```

F07AVF (CGERFS/ZGERFS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07AVF (CGERFS/ZGERFS) returns error bounds for the solution of a complex system of linear equations with multiple right-hand sides, $AX = B$, $A^TX = B$ or $A^HX = B$. It improves the solution by iterative refinement, in order to reduce the backward error as much as possible.

2. Specification

```
      SUBROUTINE F07AVF (TRANS, N, NRHS, A, LDA, AF, LDAF, IPIV, B, LDB, X,
     1                   LDX, FERR, BERR, WORK, RWORK, INFO)
      ENTRY       cgerfs  (TRANS, N, NRHS, A, LDA, AF, LDAF, IPIV, B, LDB, X,
     1                   LDX, FERR, BERR, WORK, RWORK, INFO)
      INTEGER           N, NRHS, LDA, LDAF, IPIV(*), LDB, LDX, INFO
      real              FERR(*), BERR(*), RWORK(*)
      complex           A(LDA,*), AF(LDAF,*), B(LDB,*), X(LDX,*), WORK(*)
      CHARACTER*1       TRANS
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine returns the backward errors and estimated bounds on the forward errors for the solution of a complex system of linear equations with multiple right-hand sides $AX = B$, $A^TX = B$ or $A^HX = B$. The routine handles each right-hand side vector (stored as a column of the matrix B) independently, so we describe the function of the routine in terms of a single right-hand side b and solution x.

Given a computed solution x, the routine computes the *component-wise backward error* β. This is the size of the smallest relative perturbation in each element of A and b such that x is the exact solution of a perturbed system

$$(A+\delta A)x = b + \delta b$$

$$|\delta a_{ij}| \leq \beta |a_{ij}| \quad \text{and} \quad |\delta b_i| \leq \beta |b_i|.$$

Then the routine estimates a bound for the *component-wise forward error* in the computed solution, defined by:

$$\max_i |x_i - \hat{x}_i| / \max_i |x_i|$$

where $\hat{x}$ is the true solution.

For details of the method, see the Chapter Introduction.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: TRANS – CHARACTER*1. *Input*

On entry: indicates the form of the linear equations for which X is the computed solution as follows:

if TRANS = 'N', then the linear equations are of the form $AX = B$;

if TRANS = 'T', then the linear equations are of the form $A^TX = B$;

if TRANS = 'C', then the linear equations are of the form $A^H X = B$.

Constraint: TRANS = 'N', 'T' or 'C'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS ≥ 0.

4: A(LDA,*) – ***complex*** array. *Input*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the n by n original matrix A as supplied to F07ARF (CGETRF/ZGETRF).

5: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07AVF (CGERFS/ZGERFS) is called.

Constraint: LDA ≥ max(1,N).

6: AF(LDAF,*) – ***complex*** array. *Input*

Note: the second dimension of the array AF must be at least max(1,N).

On entry: the LU factorization of A, as returned by F07ARF (CGETRF/ZGETRF).

7: LDAF – INTEGER. *Input*

On entry: the first dimension of the array AF as declared in the (sub)program from which F07AVF (CGERFS/ZGERFS) is called.

Constraint: LDAF ≥ max(1,N).

8: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: the pivot indices, as returned by F07ARF (CGETRF/ZGETRF).

9: B(LDB,*) – ***complex*** array. *Input*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

10: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07AVF (CGERFS/ZGERFS) is called.

Constraint: LDB ≥ max(1,N).

11: X(LDX,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array X must be at least max(1,NRHS).

On entry: the n by r solution matrix X, as returned by F07ASF (CGETRS/ZGETRS).

On exit: the improved solution matrix X.

12: LDX – INTEGER. *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which F07AVF (CGERFS/ZGERFS) is called.

Constraint: LDX ≥ max(1,N).

13: FERR(*) – ***real*** array. *Output*

Note: the dimension of the array FERR must be at least max(1,NRHS).

On exit: FERR(j) contains an estimated error bound for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

14: BERR(*) – ***real*** array. *Output*

Note: the dimension of the array BERR must be at least max(1,NRHS).

On exit: BERR(j) contains the component-wise backward error bound β for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

15: WORK(*) – ***complex*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,2*N).

16: RWORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array RWORK must be at least max(1,N).

17: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The bounds returned in FERR are not rigorous, because they are estimated, not computed exactly; but in practice they almost always overestimate the actual error.

8. Further Comments

For each right-hand side, computation of the backward error involves a minimum of $16n^2$ real floating-point operations. Each step of iterative refinement involves an additional $24n^2$ real operations. At most 5 steps of iterative refinement are performed, but usually only 1 or 2 steps are required.

Estimating the forward error involves solving a number of systems of linear equations of the form $Ax = b$ or $A^Hx = b$; the number is usually 5 and never more than 11. Each solution involves approximately $8n^2$ real operations.

The real analogue of this routine is F07AHF (SGERFS/DGERFS).

9. Example

To solve the system of equations $AX = B$ using iterative refinement and to compute the forward and backward error bounds, where

$$A = \begin{pmatrix} -1.34 + 2.55i & 0.28 + 3.17i & -6.39 - 2.20i & 0.72 - 0.92i \\ -0.17 - 1.41i & 3.31 - 0.15i & -0.15 + 1.34i & 1.29 + 1.38i \\ -3.29 - 2.39i & -1.91 + 4.42i & -0.14 - 1.35i & 1.72 + 1.35i \\ 2.41 + 0.39i & -0.56 + 1.47i & -0.83 - 0.69i & -1.96 + 0.67i \end{pmatrix} \text{ and}$$

$$B = \begin{pmatrix} 26.26 + 51.78i & 31.32 - 6.70i \\ 6.43 - 8.68i & 15.86 - 1.42i \\ -5.75 + 25.31i & -2.15 + 30.19i \\ 1.16 + 2.57i & -2.56 + 7.55i \end{pmatrix}.$$

Here A is unsymmetric and must first be factorized by F07ARF (CGETRF/ZGETRF).

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07AVF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, NRHMAX, LDA, LDAF, LDB, LDX
      PARAMETER        (NMAX=8,NRHMAX=NMAX,LDA=NMAX,LDAF=NMAX,LDB=NMAX,
     +                 LDX=NMAX)
      CHARACTER        TRANS
      PARAMETER        (TRANS='N')
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
*     .. Local Arrays ..
      complex          A(LDA,NMAX), AF(LDAF,NMAX), B(LDB,NRHMAX),
     +                 WORK(2*NMAX), X(LDX,NMAX)
      real             BERR(NRHMAX), FERR(NRHMAX), RWORK(NMAX)
      INTEGER          IPIV(NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         cgerfs, cgetrf, cgetrs, F06TFF, X04DBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07AVF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file, and copy A to AF and B to X
*
         READ (NIN,*) ((A(I,J),J=1,N),I=1,N)
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
         CALL F06TFF('General',N,N,A,LDA,AF,LDAF)
         CALL F06TFF('General',N,NRHS,B,LDB,X,LDX)
*
*        Factorize A in the array AF
*
         CALL cgetrf(N,N,AF,LDAF,IPIV,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
```

```
*           Compute solution in the array X
*
            CALL cgetrs(TRANS,N,NRHS,AF,LDAF,IPIV,X,LDX,INFO)
*
*           Improve solution, and compute backward errors and
*           estimated bounds on the forward errors
*
            CALL cgerfs(TRANS,N,NRHS,A,LDA,AF,LDAF,IPIV,B,LDB,X,LDX,
     +                  FERR,BERR,WORK,RWORK,INFO)
*
*           Print solution
*
            IFAIL = 0
            CALL X04DBF('General',' ',N,NRHS,X,LDX,'Bracketed','F7.4',
     +                  'Solution(s)','Integer',RLABS,'Integer',CLABS,
     +                  80,0,IFAIL)
            WRITE (NOUT,*)
            WRITE (NOUT,*) 'Backward errors (machine-dependent)'
            WRITE (NOUT,99999) (BERR(J),J=1,NRHS)
            WRITE (NOUT,*)
     +        'Estimated forward error bounds (machine-dependent)'
            WRITE (NOUT,99999) (FERR(J),J=1,NRHS)
          ELSE
            WRITE (NOUT,*) 'The factor U is singular'
          END IF
        END IF
        STOP
*
99999 FORMAT ((5X,1P,4(e11.1,7X)))
      END
```

9.2. Program Data

```
F07AVF Example Program Data
  4  2                                                  :Values of N and NRHS
 (-1.34, 2.55) ( 0.28, 3.17) (-6.39,-2.20) ( 0.72,-0.92)
 (-0.17,-1.41) ( 3.31,-0.15) (-0.15, 1.34) ( 1.29, 1.38)
 (-3.29,-2.39) (-1.91, 4.42) (-0.14,-1.35) ( 1.72, 1.35)
 ( 2.41, 0.39) (-0.56, 1.47) (-0.83,-0.69) (-1.96, 0.67) :End of matrix A
 (26.26, 51.78) (31.32, -6.70)
 ( 6.43, -8.68) (15.86, -1.42)
 (-5.75, 25.31) (-2.15, 30.19)
 ( 1.16,  2.57) (-2.56,  7.55)                          :End of matrix B
```

9.3. Program Results

```
F07AVF Example Program Results

Solution(s)
                     1                   2
 1   ( 1.0000, 1.0000) (-1.0000,-2.0000)
 2   ( 2.0000,-3.0000) ( 5.0000, 1.0000)
 3   (-4.0000,-5.0000) (-3.0000, 4.0000)
 4   ( 0.0000, 6.0000) ( 2.0000,-3.0000)

Backward errors (machine-dependent)
        5.7E-17           5.2E-17
Estimated forward error bounds (machine-dependent)
        5.8E-14           7.7E-14
```

F07AWF (CGETRI/ZGETRI) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07AWF (CGETRI/ZGETRI) computes the inverse of a complex matrix A, where A has been factorized by F07ARF (CGETRF/ZGETRF).

2. Specification

```
SUBROUTINE F07AWF (N, A, LDA, IPIV, WORK, LWORK, INFO)
ENTRY      cgetri  (N, A, LDA, IPIV, WORK, LWORK, INFO)
INTEGER        N, LDA, IPIV(*), LWORK, INFO
complex        A(LDA,*), WORK(LWORK)
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

To compute the inverse of a complex matrix A, the routine must be preceded by a call to F07ARF (CGETRF/ZGETRF), which computes the LU factorization of A as $A = PLU$. The inverse of A is computed by forming U^{-1} and then solving the equation $XPL = U^{-1}$ for X.

4. References

[1] DU CROZ, J.J. and HIGHAM, N.J.
Stability of Methods for Matrix Inversion.
LAPACK Working Note No. 27, University of Tennessee, Knoxville, 1990.

5. Parameters

1: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

2: A(LDA,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the LU factorization of A, as returned by F07ARF (CGETRF/ZGETRF).

On exit: the factorization is overwritten by the n by n matrix A^{-1}.

3: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07AWF (CGETRI/ZGETRI) is called.

Constraint: LDA ≥ max(1,N).

4: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: the pivot indices, as returned by F07ARF (CGETRF/ZGETRF).

5: WORK(LWORK) – ***complex*** array. *Workspace*

On exit: if INFO = 0, WORK(1) contains the minimum value of LWORK required for optimum performance.

6: LWORK – INTEGER. *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F07AWF (CGETRI/ZGETRI) is called.

Suggested value: for optimum performance LWORK should be at least N×*nb*, where *nb* is the ***blocksize***.

Constraint: LWORK ≥ N.

7: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, the ith diagonal element of the factor U is zero, U is singular, and the inverse of A cannot be computed.

7. Accuracy

The computed inverse X satisfies a bound of the form:

$$|XA-I| \leq c(n)\,\varepsilon|X|P|L||U|,$$

where $c(n)$ is a modest linear function of n, and ε is the ***machine precision.***

Note that a similar bound for $|AX-I|$ cannot be guaranteed, although it is almost always satisfied. See Du Croz and Higham [1].

8. Further Comments

The total number of real floating-point operations is approximately $\frac{16n^3}{3}$.

The real analogue of this routine is F07AJF (SGETRI/DGETRI).

9. Example

To compute the inverse of the matrix A, where

$$A = \begin{pmatrix} -1.34 + 2.55i & 0.28 + 3.17i & -6.39 - 2.20i & 0.72 - 0.92i \\ -0.17 - 1.41i & 3.31 - 0.15i & -0.15 + 1.34i & 1.29 + 1.38i \\ -3.29 - 2.39i & -1.91 + 4.42i & -0.14 - 1.35i & 1.72 + 1.35i \\ 2.41 + 0.39i & -0.56 + 1.47i & -0.83 - 0.69i & -1.96 + 0.67i \end{pmatrix}.$$

Here A is unsymmetric and must first be factorized by F07ARF (CGETRF/ZGETRF).

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07AWF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, LWORK
      PARAMETER        (NMAX=8,LDA=NMAX,LWORK=64*NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
```

```
*     .. Local Arrays ..
      complex           A(LDA,NMAX), WORK(LWORK)
      INTEGER           IPIV(NMAX)
      CHARACTER         CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL          cgetrf, cgetri, X04DBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07AWF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) ((A(I,J),J=1,N),I=1,N)
*
*        Factorize A
*
         CALL cgetrf(N,N,A,LDA,IPIV,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute inverse of A
*
            CALL cgetri(N,A,LDA,IPIV,WORK,LWORK,INFO)
*
*           Print inverse
*
            IFAIL = 0
            CALL X04DBF('General',' ',N,N,A,LDA,'Bracketed','F7.4',
     +                  'Inverse','Integer',RLABS,'Integer',CLABS,80,0,
     +                  IFAIL)
         ELSE
            WRITE (NOUT,*) 'The factor U is singular'
         END IF
      END IF
      STOP
*
      END
```

9.2. Program Data

```
F07AWF Example Program Data
  4                                                          :Value of N
 (-1.34, 2.55) ( 0.28, 3.17) (-6.39,-2.20) ( 0.72,-0.92)
 (-0.17,-1.41) ( 3.31,-0.15) (-0.15, 1.34) ( 1.29, 1.38)
 (-3.29,-2.39) (-1.91, 4.42) (-0.14,-1.35) ( 1.72, 1.35)
 ( 2.41, 0.39) (-0.56, 1.47) (-0.83,-0.69) (-1.96, 0.67)   :End of matrix A
```

9.3. Program Results

```
 F07AWF Example Program Results

 Inverse
                      1                   2                   3                   4
  1  ( 0.0757,-0.4324) ( 1.6512,-3.1342) ( 1.2663, 0.0418) ( 3.8181, 1.1195)
  2  (-0.1942, 0.0798) (-1.1900,-0.1426) (-0.2401,-0.5889) (-0.0101,-1.4969)
  3  (-0.0957,-0.0491) ( 0.7371,-0.4290) ( 0.3224, 0.0776) ( 0.6887, 0.7891)
  4  ( 0.3702,-0.5040) ( 3.7253,-3.1813) ( 1.7014, 0.7267) ( 3.9367, 3.3255)
```

F07BDF (SGBTRF/DGBTRF) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07BDF (SGBTRF/DGBTRF) computes the *LU* factorization of a real m by n band matrix.

2. Specification

```
SUBROUTINE F07BDF (M, N, KL, KU, AB, LDAB, IPIV, INFO)
ENTRY      sgbtrf  (M, N, KL, KU, AB, LDAB, IPIV, INFO)
INTEGER        M, N, KL, KU, LDAB, IPIV(*), INFO
real           AB(LDAB,*)
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine forms the *LU* factorization of a real m by n band matrix A using partial pivoting, with row interchanges. Usually $m = n$, and then, if A has k_l non-zero sub-diagonals and k_u non-zero super-diagonals, the factorization has the form $A = PLU$ where:

P is a permutation matrix;
L is a lower triangular matrix with unit diagonal elements and at most k_l non-zero elements in each column; and
U is an upper triangular band matrix with $k_l + k_u$ super-diagonals.

Note that L is not a band matrix, but the non-zero elements of L can be stored in the same space as the sub-diagonal elements of A. U is a band matrix but with k_l additional super-diagonals compared with A. These additional super-diagonals are created by the row interchanges.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §4.3.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: M – INTEGER. *Input*

On entry: m, the number of rows of the matrix A.

Constraint: M $\geq$ 0.

2: N – INTEGER. *Input*

On entry: n, the number of columns of the matrix A.

Constraint: N $\geq$ 0.

3: KL – INTEGER. *Input*

On entry: k_l, the number of sub-diagonals within the band of A.

Constraint: KL $\geq$ 0.

4: KU – INTEGER. *Input*

On entry: k_u, the number of super-diagonals within the band of A.

Constraint: KU $\geq$ 0.

5: AB(LDAB,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array AB must be at least max(1,N).

On entry: the m by n band matrix A, stored in rows (k_l+1) to $(2k_l+k_u+1)$; the first k_l rows need not be set. More precisely, element a_{ij} must be stored in $AB(k_l+k_u+i-j+1,j)$ for $\max(j-k_u,1) \leq i \leq \min(j+k_l,m)$.

On exit: A is overwritten by details of the factorization: the upper triangular band matrix U with $k_l + k_u$ super-diagonals is stored in rows 1 to (k_l+k_u+1) of the array, and the multipliers used to form the matrix L are stored in rows (k_l+k_u+2) to $(2k_l+k_u+1)$.

6: LDAB – INTEGER. *Input*

On entry: the first dimension of the array AB as declared in the (sub)program from which F07BDF (SGBTRF/DGBTRF) is called.

Constraint: LDAB $\geq$ 2×KL + KU + 1.

7: IPIV(*) – INTEGER array. *Output*

Note: the dimension of the array IPIV must be at least max(1,min(M,N)).

On exit: the pivot indices. Row i of the matrix A was interchanged with row IPIV(i), for $i = 1,2,...,\min(m,n)$.

8: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, u_{ii} is exactly zero. The factorization has been completed but the factor U is exactly singular, and division by zero will occur if it is subsequently used to solve a system of linear equations.

7. Accuracy

The computed factors L and U are the exact factors of a perturbed matrix $A + E$, where

$$|E| \leq c(k)\varepsilon P|L||U|,$$

$c(k)$ is a modest linear function of $k = k_l+k_u+1$, and ε is the ***machine precision***. This assumes $k \ll \min(m,n)$.

8. Further Comments

The total number of floating-point operations varies between approximately $2nk_l(k_u+1)$ and $2nk_l(k_l+k_u+1)$, depending on the interchanges, assuming $m = n \gg k_l$ and $n \gg k_u$.

A call to this routine may be followed by calls to the routines:

F07BEF (SGBTRS/DGBTRS) to solve $AX = B$ or $A^TX = B$;
F07BGF (SGBCON/DGBCON) to estimate the condition number of A.

The complex analogue of this routine is F07BRF (CGBTRF/ZGBTRF).

9. Example

To compute the LU factorization of the matrix A, where

$$A = \begin{pmatrix} -0.23 & 2.54 & -3.66 & 0.00 \\ -6.98 & 2.46 & -2.73 & -2.13 \\ 0.00 & 2.56 & 2.46 & 4.07 \\ 0.00 & 0.00 & -4.78 & -3.82 \end{pmatrix}.$$

Here A is treated as a band matrix with 1 sub-diagonal and 2 super-diagonals.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07BDF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          MMAX, NMAX, KLMAX, KUMAX, LDAB
      PARAMETER        (MMAX=8,NMAX=8,KLMAX=8,KUMAX=8,
     +                 LDAB=2*KLMAX+KUMAX+1)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, K, KL, KU, M, N
*     .. Local Arrays ..
      real             AB(LDAB,NMAX)
      INTEGER          IPIV(NMAX)
*     .. External Subroutines ..
      EXTERNAL         sgbtrf, X04CEF
*     .. Intrinsic Functions ..
      INTRINSIC        MAX, MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07BDF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) M, N, KL, KU
      IF (M.LE.MMAX .AND. N.LE.NMAX .AND. KL.LE.KLMAX .AND. KU.LE.KUMAX)
     +    THEN
*
*        Read A from data file
*
         K = KL + KU + 1
         READ (NIN,*) ((AB(K+I-J,J),J=MAX(I-KL,1),MIN(I+KU,N)),I=1,M)
*
*        Factorize A
*
         CALL sgbtrf(M,N,KL,KU,AB,LDAB,IPIV,INFO)
*
*        Print details of factorization
*
         WRITE (NOUT,*)
         IFAIL = 0
*
         CALL X04CEF(M,N,KL,KL+KU,AB,LDAB,'Details of factorization',
     +               IFAIL)
*
*        Print pivot indices
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'IPIV'
         WRITE (NOUT,99999) (IPIV(I),I=1,MIN(M,N))
*
```

```
         IF (INFO.NE.0) WRITE (NOUT,*) 'The factor U is singular'
*
      END IF
      STOP
*
99999 FORMAT ((3X,7I11))
      END
```

9.2. Program Data

```
F07BDF Example Program Data
  4  4  1  2                    :Values of M, N, KL and KU
 -0.23   2.54  -3.66
 -6.98   2.46  -2.73  -2.13
         2.56   2.46   4.07
               -4.78  -3.82    :End of matrix A
```

9.3. Program Results

```
F07BDF Example Program Results

Details of factorization
                1           2           3           4
 1     -6.9800      2.4600     -2.7300     -2.1300
 2      0.0330      2.5600      2.4600      4.0700
 3                  0.9605     -5.9329     -3.8391
 4                              0.8057     -0.7269

IPIV
             2           3           3           4
```

F07BEF (SGBTRS/DGBTRS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07BEF (SGBTRS/DGBTRS) solves a real band system of linear equations with multiple right-hand sides, $AX = B$ or $A^TX = B$, where A has been factorized by F07BDF (SGBTRF/DGBTRF).

2. Specification

```
SUBROUTINE F07BEF (TRANS, N, KL, KU, NRHS, AB, LDAB, IPIV, B, LDB, INFO)
ENTRY      sgbtrs  (TRANS, N, KL, KU, NRHS, AB, LDAB, IPIV, B, LDB, INFO)
INTEGER          N, KL, KU, NRHS, LDAB, IPIV(*), LDB, INFO
real             AB(LDAB,*), B(LDB,*)
CHARACTER*1      TRANS
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

To solve a real band system of linear equations $AX = B$ or $A^TX = B$, this routine must be preceded by a call to F07BDF (SGBTRF/DGBTRF) which computes the LU factorization of A as $A = PLU$. The solution is computed by forward and backward substitution.

If TRANS = 'N', the solution is computed by solving $PLY = B$ and then $UX = Y$.

If TRANS = 'T' or 'C', the solution is computed by solving $U^TY = B$ and then $L^TP^TX = Y$.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §4.3.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: TRANS – CHARACTER*1. *Input*

On entry: indicates the form of the equations as follows:

if TRANS = 'N', then $AX = B$ is solved for X;

if TRANS = 'T' or 'C', then $A^TX = B$ is solved for X.

Constraint: TRANS = 'N', 'T' or 'C'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: KL – INTEGER. *Input*

On entry: k_l, the number of sub-diagonals within the band of A.

Constraint: KL $\geq$ 0.

4: KU – INTEGER. *Input*

On entry: k_u, the number of super-diagonals within the band of A.

Constraint: KU $\geq$ 0.

5: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS $\geq$ 0.

6: AB(LDAB,*) – ***real*** array. *Input*

Note: the second dimension of the array AB must be at least max(1,N).

On entry: the LU factorization of A, as returned by F07BDF (SGBTRF/DGBTRF).

7: LDAB – INTEGER. *Input*

On entry: the first dimension of the array AB as declared in the (sub)program from which F07BEF (SGBTRS/DGBTRS) is called.

Constraint: LDAB $\geq$ 2×KL + KU + 1.

8: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: the pivot indices, as returned by F07BDF (SGBTRF/DGBTRF).

9: B(LDB,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

On exit: the n by r solution matrix X.

10: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07BEF (SGBTRS/DGBTRS) is called.

Constraint: LDB $\geq$ max(1,N).

11: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

For each right-hand side vector b, the computed solution x is the exact solution of a perturbed system of equations $(A+E)x = b$, where

$$|E| \leq c(k)\varepsilon P|L||U|,$$

$c(k)$ is a modest linear function of $k = k_l+k_u+1$, and ε is the ***machine precision***. This assumes $k \ll n$.

If $\hat{x}$ is the true solution, then the computed solution x satisfies a forward error bound of the form

$$\frac{\|x-\hat{x}\|_\infty}{\|x\|_\infty} \leq c(k)\text{cond}(A,x)\varepsilon$$

where $\text{cond}(A,x) = \||A^{-1}||A||x|\|_\infty/\|x\|_\infty \leq \text{cond}(A) = \||A^{-1}||A|\|_\infty \leq \kappa_\infty(A)$. Note that $\text{cond}(A,x)$ can be much smaller than $\text{cond}(A)$, and $\text{cond}(A^T)$ can be much larger (or smaller) than $\text{cond}(A)$.

Forward and backward error bounds can be computed by calling F07BHF (SGBRFS/DGBRFS), and an estimate for $\kappa_\infty(A)$ can be obtained by calling F07BGF (SGBCON/DGBCON) with NORM = 'I'.

8. Further Comments

The total number of floating-point operations is approximately $2n(2k_l+k_u)r$, assuming $n \gg k_l$ and $n \gg k_u$.

This routine may be followed by a call to F07BHF (SGBRFS/DGBRFS) to refine the solution and return an error estimate.

The complex analogue of this routine is F07BSF (CGBTRS/ZGBTRS).

9. Example

To solve the system of equations $AX = B$, where

$$A = \begin{pmatrix} -0.23 & 2.54 & -3.66 & 0.00 \\ -6.98 & 2.46 & -2.73 & -2.13 \\ 0.00 & 2.56 & 2.46 & 4.07 \\ 0.00 & 0.00 & -4.78 & -3.82 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 4.42 & -36.01 \\ 27.13 & -31.67 \\ -6.14 & -1.16 \\ 10.50 & -25.82 \end{pmatrix}.$$

Here A is unsymmetric and is treated as a band matrix, which must first be factorized by F07BDF (SGBTRF/DGBTRF).

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07BEF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, KLMAX, KUMAX, LDAB, NRHMAX, LDB
      PARAMETER        (NMAX=8,KLMAX=8,KUMAX=8,LDAB=2*KLMAX+KUMAX+1,
     +                 NRHMAX=NMAX,LDB=NMAX)
      CHARACTER        TRANS
      PARAMETER        (TRANS='N')
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, K, KL, KU, N, NRHS
*     .. Local Arrays ..
      real             AB(LDAB,NMAX), B(LDB,NRHMAX)
      INTEGER          IPIV(NMAX)
*     .. External Subroutines ..
      EXTERNAL         sgbtrf, sgbtrs, X04CAF
*     .. Intrinsic Functions ..
      INTRINSIC        MAX, MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07BEF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS, KL, KU
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX .AND. KL.LE.KLMAX .AND. KU.LE.
     +    KUMAX) THEN
*
*        Read A and B from data file
*
         K = KL + KU + 1
         READ (NIN,*) ((AB(K+I-J,J),J=MAX(I-KL,1),MIN(I+KU,N)),I=1,N)
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
*
*        Factorize A
*
         CALL sgbtrf(N,N,KL,KU,AB,LDAB,IPIV,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
```

```
*           Compute solution
*
            CALL sgbtrs(TRANS,N,KL,KU,NRHS,AB,LDAB,IPIV,B,LDB,INFO)
*
*           Print solution
*
            IFAIL = 0
*
            CALL X04CAF('General',' ',N,NRHS,B,LDB,'Solution(s)',IFAIL)
*
         ELSE
            WRITE (NOUT,*) 'The factor U is singular'
         END IF
      END IF
      STOP
*
      END
```

9.2. Program Data

```
F07BEF Example Program Data
  4  2  1  2                    :Values of N, NRHS, KL and KU
 -0.23   2.54  -3.66
 -6.98   2.46  -2.73  -2.13
         2.56   2.46   4.07
               -4.78  -3.82    :End of matrix A
  4.42 -36.01
 27.13 -31.67
 -6.14  -1.16
 10.50 -25.82                  :End of matrix B
```

9.3. Program Results

```
F07BEF Example Program Results

Solution(s)
                1           2
  1     -2.0000      1.0000
  2      3.0000     -4.0000
  3      1.0000      7.0000
  4     -4.0000     -2.0000
```

F07BGF (SGBCON/DGBCON) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07BGF (SGBCON/DGBCON) estimates the condition number of a real band matrix A, where A has been factorized by F07BDF (SGBTRF/DGBTRF).

2. Specification

```
SUBROUTINE F07BGF (NORM, N, KL, KU, AB, LDAB, IPIV, ANORM, RCOND, WORK,
1                  IWORK, INFO)
ENTRY       sgbcon (NORM, N, KL, KU, AB, LDAB, IPIV, ANORM, RCOND, WORK,
1                  IWORK, INFO)
INTEGER          N, KL, KU, LDAB, IPIV(*), IWORK(*), INFO
real             AB(LDAB,*), ANORM, RCOND, WORK(*)
CHARACTER*1      NORM
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine estimates the condition number of a real band matrix A, in either the 1-norm or the infinity-norm:

$$\kappa_1(A) = \|A\|_1\|A^{-1}\|_1 \text{ or } \kappa_\infty(A) = \|A\|_\infty\|A^{-1}\|_\infty.$$

Note that $\kappa_\infty(A) = \kappa_1(A^T)$.

Because the condition number is infinite if A is singular, the routine actually returns an estimate of the **reciprocal** of the condition number.

The routine should be preceded by a call to F06RBF to compute $\|A\|_1$ or $\|A\|_\infty$, and a call to F07BDF (SGBTRF/DGBTRF) to compute the LU factorization of A. The routine then uses Higham's implementation of Hager's method [1] to estimate $\|A^{-1}\|_1$ or $\|A^{-1}\|_\infty$.

4. References

[1] HIGHAM, N.J.
FORTRAN Codes for Estimating the One-norm of a Real or Complex Matrix, with Applications to Condition Estimation.
ACM Trans. Math. Softw., 14, pp. 381-396, 1988.

5. Parameters

1: NORM – CHARACTER*1. *Input*

On entry: indicates whether $\kappa_1(A)$ or $\kappa_\infty(A)$ is estimated as follows:

if NORM = '1' or 'O', then $\kappa_1(A)$ is estimated;

if NORM = 'I', then $\kappa_\infty(A)$ is estimated.

Constraint: NORM = '1', 'O' or 'I'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: KL – INTEGER. *Input*

On entry: k_l, the number of sub-diagonals within the band of A.

Constraint: KL $\geq$ 0.

4: KU – INTEGER. *Input*

On entry: k_u, the number of super-diagonals within the band of A.

Constraint: KU ≥ 0.

5: AB(LDAB,*) – ***real*** array. *Input*

Note: the second dimension of the array AB must be at least max(1,N).

On entry: the LU factorization of A, as returned by F07BDF (SGBTRF/DGBTRF).

6: LDAB – INTEGER. *Input*

On entry: the first dimension of the array AB as declared in the (sub)program from which F07BGF (SGBCON/DGBCON) is called.

Constraint: LDAB ≥ 2×KL + KU + 1.

7: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: the pivot indices, as returned by F07BDF (SGBTRF/DGBTRF).

8: ANORM – ***real***. *Input*

On entry: if NORM = '1' or 'O', the 1-norm of the **original** matrix A; if NORM = 'I', the infinity-norm of the **original** matrix A. ANORM may be computed by calling F06RBF with the same value for the parameter NORM. ANORM must be computed either **before** calling F07BDF (SGBTRF/DGBTRF) or else from a **copy** of the original matrix A.

Constraint: ANORM ≥ 0.0.

9: RCOND – ***real***. *Output*

On exit: an estimate of the reciprocal of the condition number of A. RCOND is set to zero if exact singularity is detected or the estimate underflows. If RCOND is less than ***machine precision***, then A is singular to working precision.

10: WORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,3*N).

11: IWORK(*) – INTEGER array. *Workspace*

Note: the dimension of the array IWORK must be at least max(1,N).

12: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed estimate RCOND is never less than the true value ρ, and in practice is nearly always less than 10ρ, although examples can be constructed where RCOND is much larger.

8. Further Comments

A call to this routine involves solving a number of systems of linear equations of the form $Ax = b$ or $A^T x = b$; the number is usually 4 or 5 and never more than 11. Each solution involves approximately $2n(2k_l+k_u)$ floating-point operations (assuming $n \gg k_l$ and $n \gg k_u$) but takes considerably longer than a call to F07BEF (SGBTRS/DGBTRS) with 1 right-hand side, because extra care is taken to avoid overflow when A is approximately singular.

The complex analogue of this routine is F07BUF (CGBCON/ZGBCON).

9. Example

To estimate the condition number in the 1-norm of the matrix A, where

$$A = \begin{pmatrix} -0.23 & 2.54 & -3.66 & 0.00 \\ -6.98 & 2.46 & -2.73 & -2.13 \\ 0.00 & 2.56 & 2.46 & 4.07 \\ 0.00 & 0.00 & -4.78 & -3.82 \end{pmatrix}.$$

Here A is unsymmetric and is treated as a band matrix, which must first be factorized by F07BDF (SGBTRF/DGBTRF). The true condition number in the 1-norm is 56.40.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07BGF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, KLMAX, KUMAX, LDAB
      PARAMETER        (NMAX=8,KLMAX=8,KUMAX=8,LDAB=2*KLMAX+KUMAX+1)
      CHARACTER        NORM
      PARAMETER        (NORM='1')
*     .. Local Scalars ..
      real             ANORM, RCOND
      INTEGER          I, INFO, J, K, KL, KU, N
*     .. Local Arrays ..
      real             AB(LDAB,NMAX), WORK(3*NMAX)
      INTEGER          IPIV(NMAX), IWORK(NMAX)
*     .. External Functions ..
      real             F06RBF, X02AJF
      EXTERNAL         F06RBF, X02AJF
*     .. External Subroutines ..
      EXTERNAL         sgbcon, sgbtrf
*     .. Intrinsic Functions ..
      INTRINSIC        MAX, MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07BGF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, KL, KU
      IF (N.LE.NMAX .AND. KL.LE.KLMAX .AND. KU.LE.KUMAX) THEN
*
*        Read A from data file
*
         K = KL + KU + 1
         READ (NIN,*) ((AB(K+I-J,J),J=MAX(I-KL,1),MIN(I+KU,N)),I=1,N)
*
*        Compute norm of A
*
         ANORM = F06RBF(NORM,N,KL,KU,AB(KL+1,1),LDAB,WORK)
*
```

```
*        Factorize A
*
         CALL sgbtrf(N,N,KL,KU,AB,LDAB,IPIV,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Estimate condition number
*
            CALL sgbcon(NORM,N,KL,KU,AB,LDAB,IPIV,ANORM,RCOND,WORK,
     +                  IWORK,INFO)
*
            IF (RCOND.GE.X02AJF()) THEN
               WRITE (NOUT,99999) 'Estimate of condition number =',
     +           1.0e0/RCOND
            ELSE
               WRITE (NOUT,*) 'A is singular to working precision'
            END IF
         ELSE
            WRITE (NOUT,*) 'The factor U is singular'
         END IF
      END IF
      STOP
*
99999 FORMAT (1X,A,1P,e10.2)
      END
```

9.2. Program Data

```
F07BGF Example Program Data
  4  1  2                        :Values of N, KL and KU
 -0.23   2.54  -3.66
 -6.98   2.46  -2.73  -2.13
         2.56   2.46   4.07
               -4.78  -3.82      :End of matrix A
```

9.3. Program Results

```
 F07BGF Example Program Results

 Estimate of condition number =  5.64E+01
```

F07BHF (SGBRFS/DGBRFS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07BHF (SGBRFS/DGBRFS) returns error bounds for the solution of a real band system of linear equations with multiple right-hand sides, $AX = B$ or $A^TX = B$. It improves the solution by iterative refinement, in order to reduce the backward error as much as possible.

2. Specification

```
 SUBROUTINE F07BHF (TRANS, N, KL, KU, NRHS, AB, LDAB, AFB, LDAFB, IPIV,
1                   B, LDB, X, LDX, FERR, BERR, WORK, IWORK, INFO)
 ENTRY       sgbrfs  (TRANS, N, KL, KU, NRHS, AB, LDAB, AFB, LDAFB, IPIV,
1                   B, LDB, X, LDX, FERR, BERR, WORK, IWORK, INFO)
 INTEGER          N, KL, KU, NRHS, LDAB, LDAFB, IPIV(*), LDB, LDX, IWORK(*),
1                 INFO
 real             AB(LDAB,*), AFB(LDAFB,*), B(LDB,*), X(LDX,*), FERR(*),
1                 BERR(*), WORK(*)
 CHARACTER*1      TRANS
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine returns the backward errors and estimated bounds on the forward errors for the solution of a real band system of linear equations with multiple right-hand sides $AX = B$ or $A^TX = B$. The routine handles each right-hand side vector (stored as a column of the matrix B) independently, so we describe the function of the routine in terms of a single right-hand side b and solution x.

Given a computed solution x, the routine computes the *component-wise backward error* β. This is the size of the smallest relative perturbation in each element of A and b such that x is the exact solution of a perturbed system

$$(A+\delta A)x = b + \delta b$$

$$|\delta a_{ij}| \leq \beta |a_{ij}| \text{ and } |\delta b_i| \leq \beta |b_i|.$$

Then the routine estimates a bound for the *component-wise forward error* in the computed solution, defined by:

$$\max_i |x_i - \hat{x}_i| / \max_i |x_i|$$

where $\hat{x}$ is the true solution.

For details of the method, see the Chapter Introduction.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: TRANS – CHARACTER*1. *Input*

On entry: indicates the form of the linear equations for which X is the computed solution as follows:

if TRANS = 'N', then the linear equations are of the form $AX = B$;

if TRANS = 'T' or 'C', then the linear equations are of the form $A^TX = B$.

Constraint: TRANS = 'N', 'T' or 'C'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: KL – INTEGER. *Input*

On entry: k_l, the number of sub-diagonals within the band of A.

Constraint: KL ≥ 0.

4: KU – INTEGER. *Input*

On entry: k_u, the number of super-diagonals within the band of A.

Constraint: KU ≥ 0.

5: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS ≥ 0.

6: AB(LDAB,*) – ***real*** array. *Input*

Note: the second dimension of the array AB must be at least max(1,N).

On entry: the n by n original band matrix A as supplied to F07BDF (SGBTRF/DGBTRF), but stored in rows 1 to (k_l+k_u+1) of the array rather than in rows (k_l+1) to $(2k_l+k_u+1)$.

7: LDAB – INTEGER. *Input*

On entry: the first dimension of the array AB as declared in the (sub)program from which F07BHF (SGBRFS/DGBRFS) is called.

Constraint: LDAB ≥ KL + KU + 1.

8: AFB(LDAFB,*) – ***real*** array. *Input*

Note: the second dimension of the array AFB must be at least max(1,N).

On entry: the LU factorization of A, as returned by F07BDF (SGBTRF/DGBTRF).

9: LDAFB – INTEGER. *Input*

On entry: the first dimension of the array AFB as declared in the (sub)program from which F07BHF (SGBRFS/DGBRFS) is called.

Constraint: LDAFB ≥ 2×KL + KU + 1.

10: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: the pivot indices, as returned by F07BDF (SGBTRF/DGBTRF).

11: B(LDB,*) – ***real*** array. *Input*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

12: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07BHF (SGBRFS/DGBRFS) is called.

Constraint: LDB ≥ max(1,N).

13: X(LDX,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array X must be at least max(1,NRHS).

On entry: the n by r solution matrix X, as returned by F07BEF (SGBTRS/DGBTRS).

On exit: the improved solution matrix X.

14: LDX – INTEGER. *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which F07BHF (SGBRFS/DGBRFS) is called.

Constraint: LDX ≥ max(1,N).

15: FERR(*) – ***real*** array. *Output*

Note: the dimension of the array FERR must be at least max(1,NRHS).

On exit: FERR(j) contains an estimated error bound for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

16: BERR(*) – ***real*** array. *Output*

Note: the dimension of the array BERR must be at least max(1,NRHS).

On exit: BERR(j) contains the component-wise backward error bound β for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

17: WORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,3*N).

18: IWORK(*) – INTEGER array. *Workspace*

Note: the dimension of the array IWORK must be at least max(1,N).

19: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The bounds returned in FERR are not rigorous, because they are estimated, not computed exactly; but in practice they almost always overestimate the actual error.

8. Further Comments

For each right-hand side, computation of the backward error involves a minimum of $4n(k_l+k_u)$ floating-point operations. Each step of iterative refinement involves an additional $2n(4k_l+3k_u)$ operations. This assumes $n \gg k_l$ and $n \gg k_u$. At most 5 steps of iterative refinement are performed, but usually only 1 or 2 steps are required.

Estimating the forward error involves solving a number of systems of linear equations of the form $Ax = b$ or $A^Tx = b$; the number is usually 4 or 5 and never more than 11. Each solution involves approximately $2n(2k_l+k_u)$ operations.

The complex analogue of this routine is F07BVF (CGBRFS/ZGBRFS).

9. Example

To solve the system of equations $AX = B$ using iterative refinement and to compute the forward and backward error bounds, where

$$A = \begin{pmatrix} -0.23 & 2.54 & -3.66 & 0.00 \\ -6.98 & 2.46 & -2.73 & -2.13 \\ 0.00 & 2.56 & 2.46 & 4.07 \\ 0.00 & 0.00 & -4.78 & -3.82 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 4.42 & -36.01 \\ 27.13 & -31.67 \\ -6.14 & -1.16 \\ 10.50 & -25.82 \end{pmatrix}.$$

Here A is unsymmetric and is treated as a band matrix, which must first be factorized by F07BDF (SGBTRF/DGBTRF).

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07BHF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      real             ZERO
      PARAMETER        (ZERO=0.0e0)
      INTEGER          NMAX, NRHMAX, KLMAX, KUMAX, LDAB, LDAFB, LDB, LDX
      PARAMETER        (NMAX=8,NRHMAX=NMAX,KLMAX=8,KUMAX=8,
     +                 LDAB=KLMAX+KUMAX+1,LDAFB=2*KLMAX+KUMAX+1,
     +                 LDB=NMAX,LDX=NMAX)
      CHARACTER        TRANS
      PARAMETER        (TRANS='N')
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, K, KL, KU, N, NRHS
*     .. Local Arrays ..
      real             AB(LDAB,NMAX), AFB(LDAFB,NMAX), B(LDB,NRHMAX),
     +                 BERR(NRHMAX), FERR(NRHMAX), WORK(3*NMAX),
     +                 X(LDX,NMAX)
      INTEGER          IPIV(NMAX), IWORK(NMAX)
*     .. External Subroutines ..
      EXTERNAL         F06QFF, F06QHF, sgbrfs, sgbtrf, sgbtrs, X04CAF
*     .. Intrinsic Functions ..
      INTRINSIC        MAX, MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07BHF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS, KL, KU
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX .AND. KL.LE.KLMAX .AND. KU.LE.
     +    KUMAX) THEN
*
*        Set A to zero to avoid referencing uninitialized elements
*
         CALL F06QHF('General',KL+KU+1,N,ZERO,ZERO,AB,LDAB)
*
*        Read A and B from data file, and copy A to AFB and B to X
*
         K = KU + 1
         READ (NIN,*) ((AB(K+I-J,J),J=MAX(I-KL,1),MIN(I+KU,N)),I=1,N)
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
*
         CALL F06QFF('General',KL+KU+1,N,AB,LDAB,AFB(KL+1,1),LDAFB)
*
         CALL F06QFF('General',N,NRHS,B,LDB,X,LDX)
*
```

```
*        Factorize A in the array AFB
*
         CALL sgbtrf(N,N,KL,KU,AFB,LDAFB,IPIV,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute solution in the array X
*
            CALL sgbtrs(TRANS,N,KL,KU,NRHS,AFB,LDAFB,IPIV,X,LDX,INFO)
*
*           Improve solution, and compute backward errors and
*           estimated bounds on the forward errors
*
            CALL sgbrfs(TRANS,N,KL,KU,NRHS,AB,LDAB,AFB,LDAFB,IPIV,B,LDB,
     +                  X,LDX,FERR,BERR,WORK,IWORK,INFO)
*
*           Print solution
*
            IFAIL = 0
*
            CALL X04CAF('General',' ',N,NRHS,X,LDX,'Solution(s)',IFAIL)
*
            WRITE (NOUT,*)
            WRITE (NOUT,*) 'Backward errors (machine-dependent)'
            WRITE (NOUT,99999) (BERR(J),J=1,NRHS)
            WRITE (NOUT,*)
     +        'Estimated forward error bounds (machine-dependent)'
            WRITE (NOUT,99999) (FERR(J),J=1,NRHS)
         ELSE
            WRITE (NOUT,*) 'The factor U is singular'
         END IF
       END IF
       STOP
*
99999 FORMAT ((3X,1P,7e11.1))
      END
```

9.2. Program Data

```
F07BHF Example Program Data
  4  2  1  2                  :Values of N, NRHS, KL and KU
 -0.23   2.54  -3.66
 -6.98   2.46  -2.73  -2.13
         2.56   2.46   4.07
               -4.78  -3.82   :End of matrix A
  4.42 -36.01
 27.13 -31.67
 -6.14  -1.16
 10.50 -25.82                 :End of matrix B
```

9.3. Program Results

```
F07BHF Example Program Results

Solution(s)
                1           2
 1      -2.0000      1.0000
 2       3.0000     -4.0000
 3       1.0000      7.0000
 4      -4.0000     -2.0000

Backward errors (machine-dependent)
      1.0E-16    8.2E-17
Estimated forward error bounds (machine-dependent)
      1.5E-14    1.8E-14
```

F07BRF (CGBTRF/ZGBTRF) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07BRF (CGBTRF/ZGBTRF) computes the *LU* factorization of a complex m by n band matrix.

2. Specification

```
SUBROUTINE F07BRF (M, N, KL, KU, AB, LDAB, IPIV, INFO)
ENTRY      cgbtrf  (M, N, KL, KU, AB, LDAB, IPIV, INFO)
INTEGER        M, N, KL, KU, LDAB, IPIV(*), INFO
complex        AB(LDAB,*)
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine forms the *LU* factorization of a complex m by n band matrix A using partial pivoting, with row interchanges. Usually $m = n$, and then, if A has k_l non-zero sub-diagonals and k_u non-zero super-diagonals, the factorization has the form $A = PLU$ where:

P is a permutation matrix;
L is a lower triangular matrix with unit diagonal elements and at most k_l non-zero elements in each column; and
U is an upper triangular band matrix with $k_l + k_u$ super-diagonals.

Note that L is not a band matrix, but the non-zero elements of L can be stored in the same space as the sub-diagonal elements of A. U is a band matrix but with k_l additional super-diagonals compared with A. These additional super-diagonals are created by the row interchanges.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §4.3.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: M – INTEGER. *Input*

On entry: m, the number of rows of the matrix A.

Constraint: M $\geq$ 0.

2: N – INTEGER. *Input*

On entry: n, the number of columns of the matrix A.

Constraint: N $\geq$ 0.

3: KL – INTEGER. *Input*

On entry: k_l, the number of sub-diagonals within the band of A.

Constraint: KL $\geq$ 0.

4: KU – INTEGER. *Input*

On entry: k_u, the number of super-diagonals within the band of A.

Constraint: KU $\geq$ 0.

5: AB(LDAB,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array AB must be at least max(1,N).

On entry: the m by n band matrix A, stored in rows (k_l+1) to $(2k_l+k_u+1)$; the first k_l rows need not be set. More precisely, element a_{ij} must be stored in $\mathrm{AB}(k_l+k_u+i-j+1,j)$ for $\max(j-k_u,1) \leq i \leq \min(j+k_l,m)$.

On exit: A is overwritten by details of the factorization: the upper triangular band matrix U with $k_l + k_u$ super-diagonals is stored in rows 1 to (k_l+k_u+1) of the array, and the multipliers used to form the matrix L are stored in rows (k_l+k_u+2) to $(2k_l+k_u+1)$.

6: LDAB – INTEGER. *Input*

On entry: the first dimension of the array AB as declared in the (sub)program from which F07BRF (CGBTRF/ZGBTRF) is called.

Constraint: LDAB ≥ 2×KL + KU + 1.

7: IPIV(*) – INTEGER array. *Output*

Note: the dimension of the array IPIV must be at least max(1,min(M,N)).

On exit: the pivot indices. Row i of the matrix A was interchanged with row IPIV(i), for $i = 1,2,...,\min(m,n)$.

8: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, u_{ii} is exactly zero. The factorization has been completed but the factor U is exactly singular, and division by zero will occur if it is subsequently used to solve a system of linear equations.

7. Accuracy

The computed factors L and U are the exact factors of a perturbed matrix $A + E$, where

$$|E| \leq c(k)\varepsilon P|L||U|,$$

$c(k)$ is a modest linear function of $k = k_l+k_u+1$, and ε is the ***machine precision***. This assumes $k \ll \min(m,n)$.

8. Further Comments

The total number of real floating-point operations varies between approximately $8nk_l(k_u+1)$ and $8nk_l(k_l+k_u+1)$, depending on the interchanges, assuming $m = n \gg k_l$ and $n \gg k_u$.

A call to this routine may be followed by calls to the routines:

F07BSF (CGBTRS/ZGBTRS) to solve $AX = B$, $A^TX = B$ or $A^HX = B$;
F07BUF (CGBCON/ZGBCON) to estimate the condition number of A.

The real analogue of this routine is F07BDF (SGBTRF/DGBTRF).

9. Example

To compute the LU factorization of the matrix A, where

$$A = \begin{pmatrix} -1.65 + 2.26i & -2.05 - 0.85i & 0.97 - 2.84i & 0.00 + 0.00i \\ 0.00 + 6.30i & -1.48 - 1.75i & -3.99 + 4.01i & 0.59 - 0.48i \\ 0.00 + 0.00i & -0.77 + 2.83i & -1.06 + 1.94i & 3.33 - 1.04i \\ 0.00 + 0.00i & 0.00 + 0.00i & 4.48 - 1.09i & -0.46 - 1.72i \end{pmatrix}.$$

Here A is treated as a band matrix with 1 sub-diagonal and 2 super-diagonals.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07BRF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          MMAX, NMAX, KLMAX, KUMAX, LDAB
      PARAMETER        (MMAX=8,NMAX=8,KLMAX=8,KUMAX=8,
     +                 LDAB=2*KLMAX+KUMAX+1)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, K, KL, KU, M, N
*     .. Local Arrays ..
      complex          AB(LDAB,NMAX)
      INTEGER          IPIV(NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         cgbtrf, X04DFF
*     .. Intrinsic Functions ..
      INTRINSIC        MAX, MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07BRF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) M, N, KL, KU
      IF (M.LE.MMAX .AND. N.LE.NMAX .AND. KL.LE.KLMAX .AND. KU.LE.KUMAX)
     +    THEN
*
*        Read A from data file
*
         K = KL + KU + 1
         READ (NIN,*) ((AB(K+I-J,J),J=MAX(I-KL,1),MIN(I+KU,N)),I=1,M)
*
*        Factorize A
*
         CALL cgbtrf(M,N,KL,KU,AB,LDAB,IPIV,INFO)
*
*        Print details of factorization
*
         WRITE (NOUT,*)
         IFAIL = 0
         CALL X04DFF(M,N,KL,KL+KU,AB,LDAB,'Bracketed','F7.4',
     +              'Details of factorization','Integer',RLABS,
     +              'Integer',CLABS,80,0,IFAIL)
*
*        Print pivot indices
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'IPIV'
         WRITE (NOUT,99999) (IPIV(I),I=1,MIN(M,N))
*
```

```
         IF (INFO.NE.0) WRITE (NOUT,*) 'The factor U is singular'
*
      END IF
      STOP
*
99999 FORMAT ((1X,I12,3I18))
      END
```

9.2. Program Data

```
F07BRF Example Program Data
  4  4  1  2                                          :Values of M, N, KL and KU
 (-1.65, 2.26) (-2.05,-0.85) ( 0.97,-2.84)
 ( 0.00, 6.30) (-1.48,-1.75) (-3.99, 4.01) ( 0.59,-0.48)
               (-0.77, 2.83) (-1.06, 1.94) ( 3.33,-1.04)
                             ( 4.48,-1.09) (-0.46,-1.72) :End of matrix A
```

9.3. Program Results

```
 F07BRF Example Program Results

 Details of factorization
                        1                     2                     3                     4
  1   ( 0.0000, 6.3000) (-1.4800,-1.7500) (-3.9900, 4.0100) ( 0.5900,-0.4800)
  2   ( 0.3587, 0.2619) (-0.7700, 2.8300) (-1.0600, 1.9400) ( 3.3300,-1.0400)
  3                     ( 0.2314, 0.6358) ( 4.9303,-3.0086) (-1.7692,-1.8587)
  4                                       ( 0.7604, 0.2429) ( 0.4338, 0.1233)

 IPIV
            2                     3                     3                     4
```

F07BSF (CGBTRS/ZGBTRS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07BSF (CGBTRS/ZGBTRS) solves a complex band system of linear equations with multiple right-hand sides, $AX = B$, $A^TX = B$ or $A^HX = B$, where A has been factorized by F07BRF (CGBTRF/ZGBTRF).

2. Specification

```
SUBROUTINE F07BSF (TRANS, N, KL, KU, NRHS, AB, LDAB, IPIV, B, LDB, INFO)
ENTRY      cgbtrs  (TRANS, N, KL, KU, NRHS, AB, LDAB, IPIV, B, LDB, INFO)
INTEGER        N, KL, KU, NRHS, LDAB, IPIV(*), LDB, INFO
complex        AB(LDAB,*), B(LDB,*)
CHARACTER*1    TRANS
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

To solve a complex band system of linear equations $AX = B$, $A^TX = B$ or $A^HX = B$, this routine must be preceded by a call to F07BRF (CGBTRF/ZGBTRF) which computes the LU factorization of A as $A = PLU$. The solution is computed by forward and backward substitution.

If TRANS = 'N', the solution is computed by solving $PLY = B$ and then $UX = Y$.

If TRANS = 'T', the solution is computed by solving $U^TY = B$ and then $L^TP^TX = Y$.

If TRANS = 'C', the solution is computed by solving $U^HY = B$ and then $L^HP^TX = Y$.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §4.3.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: TRANS – CHARACTER*1. *Input*

On entry: indicates the form of the equations as follows:

if TRANS = 'N', then $AX = B$ is solved for X;

if TRANS = 'T', then $A^TX = B$ is solved for X;

if TRANS = 'C', then $A^HX = B$ is solved for X.

Constraint: TRANS = 'N', 'T' or 'C'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: KL – INTEGER. *Input*

On entry: k_l, the number of sub-diagonals within the band of A.

Constraint: KL $\geq$ 0.

4: KU – INTEGER. *Input*

On entry: k_u, the number of super-diagonals within the band of A.

Constraint: KU $\geq$ 0.

5: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS ≥ 0.

6: AB(LDAB,*) – ***complex*** array. *Input*

Note: the second dimension of the array AB must be at least max(1,N).

On entry: the LU factorization of A, as returned by F07BRF (CGBTRF/ZGBTRF).

7: LDAB – INTEGER. *Input*

On entry: the first dimension of the array AB as declared in the (sub)program from which F07BSF (CGBTRS/ZGBTRS) is called.

Constraint: LDAB $\geq$ 2×KL + KU + 1.

8: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: the pivot indices, as returned by F07BRF (CGBTRF/ZGBTRF).

9: B(LDB,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

On exit: the n by r solution matrix X.

10: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07BSF (CGBTRS/ZGBTRS) is called.

Constraint: LDB $\geq$ max(1,N).

11: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

For each right-hand side vector b, the computed solution x is the exact solution of a perturbed system of equations $(A+E)x = b$, where

$$|E| \leq c(k)\varepsilon P|L||U|,$$

$c(k)$ is a modest linear function of $k = k_l+k_u+1$, and ε is the ***machine precision***. This assumes $k \ll n$.

If $\hat{x}$ is the true solution, then the computed solution x satisfies a forward error bound of the form

$$\frac{\|x-\hat{x}\|_\infty}{\|x\|_\infty} \leq c(k)\mathrm{cond}(A,x)\varepsilon$$

where $\mathrm{cond}(A,x) = \||A^{-1}||A||x|\|_\infty/\|x\|_\infty \leq \mathrm{cond}(A) = \||A^{-1}||A|\|_\infty \leq \kappa_\infty(A)$. Note that $\mathrm{cond}(A,x)$ can be much smaller than $\mathrm{cond}(A)$, and $\mathrm{cond}(A^H)$ (which is the same as $\mathrm{cond}(A^T)$) can be much larger (or smaller) than $\mathrm{cond}(A)$.

Forward and backward error bounds can be computed by calling F07BVF (CGBRFS/ZGBRFS), and an estimate for $\kappa_\infty(A)$ can be obtained by calling F07BUF (CGBCON/ZGBCON) with NORM = 'I'.

8. Further Comments

The total number of real floating-point operations is approximately $8n(2k_l+k_u)r$, assuming $n \gg k_l$ and $n \gg k_u$.

This routine may be followed by a call to F07BVF (CGBRFS/ZGBRFS) to refine the solution and return an error estimate.

The real analogue of this routine is F07BEF (SGBTRS/DGBTRS).

9. Example

To solve the system of equations $AX = B$, where

$$A = \begin{pmatrix} -1.65 + 2.26i & -2.05 - 0.85i & 0.97 - 2.84i & 0.00 + 0.00i \\ 0.00 + 6.30i & -1.48 - 1.75i & -3.99 + 4.01i & 0.59 - 0.48i \\ 0.00 + 0.00i & -0.77 + 2.83i & -1.06 + 1.94i & 3.33 - 1.04i \\ 0.00 + 0.00i & 0.00 + 0.00i & 4.48 - 1.09i & -0.46 - 1.72i \end{pmatrix} \quad \text{and}$$

$$B = \begin{pmatrix} -1.06 + 21.50i & 12.85 + 2.84i \\ -22.72 - 53.90i & -70.22 + 21.57i \\ 28.24 - 38.60i & -20.73 - 1.23i \\ -34.56 + 16.73i & 26.01 + 31.97i \end{pmatrix}.$$

Here A is unsymmetric and is treated as a band matrix, which must first be factorized by F07BRF (CGBTRF/ZGBTRF).

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07BSF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, KLMAX, KUMAX, LDAB, NRHMAX, LDB
      PARAMETER        (NMAX=8,KLMAX=8,KUMAX=8,LDAB=2*KLMAX+KUMAX+1,
     +                 NRHMAX=NMAX,LDB=NMAX)
      CHARACTER        TRANS
      PARAMETER        (TRANS='N')
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, K, KL, KU, N, NRHS
*     .. Local Arrays ..
      complex          AB(LDAB,NMAX), B(LDB,NRHMAX)
      INTEGER          IPIV(NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         cgbtrf, cgbtrs, X04DBF
*     .. Intrinsic Functions ..
      INTRINSIC        MAX, MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07BSF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS, KL, KU
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX .AND. KL.LE.KLMAX .AND. KU.LE.
     +    KUMAX) THEN
*
*        Read A and B from data file
*
         K = KL + KU + 1
         READ (NIN,*) ((AB(K+I-J,J),J=MAX(I-KL,1),MIN(I+KU,N)),I=1,N)
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
*
```

```
*        Factorize A
*
         CALL cgbtrf(N,N,KL,KU,AB,LDAB,IPIV,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute solution
*
            CALL cgbtrs(TRANS,N,KL,KU,NRHS,AB,LDAB,IPIV,B,LDB,INFO)
*
*           Print solution
*
            IFAIL = 0
            CALL X04DBF('General',' ',N,NRHS,B,LDB,'Bracketed','F7.4',
     +                  'Solution(s)','Integer',RLABS,'Integer',CLABS,
     +                  80,0,IFAIL)
         ELSE
            WRITE (NOUT,*) 'The factor U is singular'
         END IF
       END IF
       STOP
*
       END
```

9.2. Program Data

```
F07BSF Example Program Data
  4  2  1  2                                        :Values of N, NRHS, KL and KU
 (-1.65, 2.26) (-2.05,-0.85) ( 0.97,-2.84)
 ( 0.00, 6.30) (-1.48,-1.75) (-3.99, 4.01) ( 0.59,-0.48)
               (-0.77, 2.83) (-1.06, 1.94) ( 3.33,-1.04)
                             ( 4.48,-1.09) (-0.46,-1.72)  :End of matrix A
 ( -1.06, 21.50) ( 12.85,  2.84)
 (-22.72,-53.90) (-70.22, 21.57)
 ( 28.24,-38.60) (-20.73, -1.23)
 (-34.56, 16.73) ( 26.01, 31.97)                         :End of matrix B
```

9.3. Program Results

```
F07BSF Example Program Results

Solution(s)
                        1                   2
  1   (-3.0000, 2.0000) ( 1.0000, 6.0000)
  2   ( 1.0000,-7.0000) (-7.0000,-4.0000)
  3   (-5.0000, 4.0000) ( 3.0000, 5.0000)
  4   ( 6.0000,-8.0000) (-8.0000, 2.0000)
```

F07BUF (CGBCON/ZGBCON) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07BUF (CGBCON/ZGBCON) estimates the condition number of a complex band matrix A, where A has been factorized by F07BRF (CGBTRF/ZGBTRF).

2. Specification

```
SUBROUTINE F07BUF (NORM, N, KL, KU, AB, LDAB, IPIV, ANORM, RCOND, WORK,
1                  RWORK, INFO)
ENTRY       cgbcon (NORM, N, KL, KU, AB, LDAB, IPIV, ANORM, RCOND, WORK,
1                  RWORK, INFO)
INTEGER          N, KL, KU, LDAB, IPIV(*), INFO
real             ANORM, RCOND, RWORK(*)
complex          AB(LDAB,*), WORK(*)
CHARACTER*1      NORM
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine estimates the condition number of a complex band matrix A, in either the 1-norm or the infinity-norm:

$$\kappa_1(A) = \|A\|_1\|A^{-1}\|_1 \quad \text{or} \quad \kappa_\infty(A) = \|A\|_\infty\|A^{-1}\|_\infty.$$

Note that $\kappa_\infty(A) = \kappa_1(A^H)$.

Because the condition number is infinite if A is singular, the routine actually returns an estimate of the **reciprocal** of the condition number.

The routine should be preceded by a call to F06UBF to compute $\|A\|_1$ or $\|A\|_\infty$, and a call to F07BRF (CGBTRF/ZGBTRF) to compute the LU factorization of A. The routine then uses Higham's implementation of Hager's method [1] to estimate $\|A^{-1}\|_1$ or $\|A^{-1}\|_\infty$.

4. References

[1] HIGHAM, N.J.
FORTRAN Codes for Estimating the One-norm of a Real or Complex Matrix, with Applications to Condition Estimation.
ACM Trans. Math. Softw., 14, pp. 381-396, 1988.

5. Parameters

1: NORM – CHARACTER*1. *Input*

On entry: indicates whether $\kappa_1(A)$ or $\kappa_\infty(A)$ is estimated as follows:

if NORM = '1' or 'O', then $\kappa_1(A)$ is estimated;

if NORM = 'I', then $\kappa_\infty(A)$ is estimated.

Constraint: NORM = '1', 'O' or 'I'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: KL – INTEGER. *Input*

On entry: k_l, the number of sub-diagonals within the band of A.

Constraint: KL ≥ 0.

4: KU – INTEGER. *Input*

On entry: k_u, the number of super-diagonals within the band of A.

Constraint: KU ≥ 0.

5: AB(LDAB,*) – ***complex*** array. *Input*

Note: the second dimension of the array AB must be at least max(1,N).

On entry: the LU factorization of A, as returned by F07BRF (CGBTRF/ZGBTRF).

6: LDAB – INTEGER. *Input*

On entry: the first dimension of the array AB as declared in the (sub)program from which F07BUF (CGBCON/ZGBCON) is called.

Constraint: LDAB ≥ 2×KL + KU + 1.

7: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: the pivot indices, as returned by F07BRF (CGBTRF/ZGBTRF).

8: ANORM – ***real.*** *Input*

On entry: if NORM = '1' or 'O', the 1-norm of the **original** matrix A; if NORM = 'I', the infinity-norm of the **original** matrix A. ANORM may be computed by calling F06UBF with the same value for the parameter NORM. ANORM must be computed either **before** calling F07BRF (CGBTRF/ZGBTRF) or else from a **copy** of the original matrix A.

Constraint: ANORM ≥ 0.0.

9: RCOND – ***real.*** *Output*

On exit: an estimate of the reciprocal of the condition number of A. RCOND is set to zero if exact singularity is detected or the estimate underflows. If RCOND is less than ***machine precision***, then A is singular to working precision.

10: WORK(*) – ***complex*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,2*N).

11: RWORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array RWORK must be at least max(1,N).

12: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed estimate RCOND is never less than the true value ρ, and in practice is nearly always less than 10ρ, although examples can be constructed where RCOND is much larger.

8. Further Comments

A call to this routine involves solving a number of systems of linear equations of the form $Ax = b$ or $A^H x = b$; the number is usually 5 and never more than 11. Each solution involves approximately $8n(2k_l+k_u)$ real floating-point operations (assuming $n \gg k_l$ and $n \gg k_u$) but takes considerably longer than a call to F07BSF (CGBTRS/ZGBTRS) with 1 right-hand side, because extra care is taken to avoid overflow when A is approximately singular.

The real analogue of this routine is F07BGF (SGBCON/DGBCON).

9. Example

To estimate the condition number in the 1-norm of the matrix A, where

$$A = \begin{pmatrix} -1.65 + 2.26i & -2.05 - 0.85i & 0.97 - 2.84i & 0.00 + 0.00i \\ 0.00 + 6.30i & -1.48 - 1.75i & -3.99 + 4.01i & 0.59 - 0.48i \\ 0.00 + 0.00i & -0.77 + 2.83i & -1.06 + 1.94i & 3.33 - 1.04i \\ 0.00 + 0.00i & 0.00 + 0.00i & 4.48 - 1.09i & -0.46 - 1.72i \end{pmatrix}.$$

Here A is unsymmetric and is treated as a band matrix, which must first be factorized by F07BRF (CGBTRF/ZGBTRF). The true condition number in the 1-norm is 181.98.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07BUF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, KLMAX, KUMAX, LDAB
      PARAMETER        (NMAX=8,KLMAX=8,KUMAX=8,LDAB=2*KLMAX+KUMAX+1)
      CHARACTER        NORM
      PARAMETER        (NORM='1')
*     .. Local Scalars ..
      real             ANORM, RCOND
      INTEGER          I, INFO, J, K, KL, KU, N
*     .. Local Arrays ..
      complex          AB(LDAB,NMAX), WORK(2*NMAX)
      real             RWORK(NMAX)
      INTEGER          IPIV(NMAX)
*     .. External Functions ..
      real             F06UBF, X02AJF
      EXTERNAL         F06UBF, X02AJF
*     .. External Subroutines ..
      EXTERNAL         cgbcon, cgbtrf
*     .. Intrinsic Functions ..
      INTRINSIC        MAX, MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07BUF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, KL, KU
      IF (N.LE.NMAX .AND. KL.LE.KLMAX .AND. KU.LE.KUMAX) THEN
*
*        Read A from data file
*
         K = KL + KU + 1
         READ (NIN,*) ((AB(K+I-J,J),J=MAX(I-KL,1),MIN(I+KU,N)),I=1,N)
*
*        Compute norm of A
*
         ANORM = F06UBF(NORM,N,KL,KU,AB(KL+1,1),LDAB,RWORK)
*
```

```
*        Factorize A
*
         CALL cgbtrf(N,N,KL,KU,AB,LDAB,IPIV,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Estimate condition number
*
            CALL cgbcon(NORM,N,KL,KU,AB,LDAB,IPIV,ANORM,RCOND,WORK,
     +                  RWORK,INFO)
*
            IF (RCOND.GE.X02AJF()) THEN
               WRITE (NOUT,99999) 'Estimate of condition number =',
     +           1.0e0/RCOND
            ELSE
               WRITE (NOUT,*) 'A is singular to working precision'
            END IF
         ELSE
            WRITE (NOUT,*) 'The factor U is singular'
         END IF
      END IF
      STOP
*
99999 FORMAT (1X,A,1P,e10.2)
      END
```

9.2. Program Data

```
F07BUF Example Program Data
  4  1  2                                          :Values of N, KL and KU
 (-1.65, 2.26) (-2.05,-0.85) ( 0.97,-2.84)
 ( 0.00, 6.30) (-1.48,-1.75) (-3.99, 4.01) ( 0.59,-0.48)
               (-0.77, 2.83) (-1.06, 1.94) ( 3.33,-1.04)
                             ( 4.48,-1.09) (-0.46,-1.72) :End of matrix A
```

9.3. Program Results

```
 F07BUF Example Program Results

 Estimate of condition number =  1.04E+02
```

F07BVF (CGBRFS/ZGBRFS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07BVF (CGBRFS/ZGBRFS) returns error bounds for the solution of a complex band system of linear equations with multiple right-hand sides, $AX = B$, $A^TX = B$ or $A^HX = B$. It improves the solution by iterative refinement, in order to reduce the backward error as much as possible.

2. Specification

```
 SUBROUTINE F07BVF (TRANS, N, KL, KU, NRHS, AB, LDAB, AFB, LDAFB, IPIV,
1                   B, LDB, X, LDX, FERR, BERR, WORK, RWORK, INFO)
 ENTRY       cgbrfs  (TRANS, N, KL, KU, NRHS, AB, LDAB, AFB, LDAFB, IPIV,
1                   B, LDB, X, LDX, FERR, BERR, WORK, RWORK, INFO)
 INTEGER          N, KL, KU, NRHS, LDAB, LDAFB, IPIV(*), LDB, LDX, INFO
 real             FERR(*), BERR(*), RWORK(*)
 complex          AB(LDAB,*), AFB(LDAFB,*), B(LDB,*), X(LDX,*), WORK(*)
 CHARACTER*1      TRANS
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine returns the backward errors and estimated bounds on the forward errors for the solution of a complex band system of linear equations with multiple right-hand sides $AX = B$, $A^TX = B$ or $A^HX = B$. The routine handles each right-hand side vector (stored as a column of the matrix B) independently, so we describe the function of the routine in terms of a single right-hand side b and solution x.

Given a computed solution x, the routine computes the *component-wise backward error* β. This is the size of the smallest relative perturbation in each element of A and b such that x is the exact solution of a perturbed system

$$(A+\delta A)x = b + \delta b$$

$$|\delta a_{ij}| \leq \beta |a_{ij}| \quad \text{and} \quad |\delta b_i| \leq \beta |b_i|.$$

Then the routine estimates a bound for the *component-wise forward error* in the computed solution, defined by:

$$\max_i |x_i - \hat{x}_i| / \max_i |x_i|$$

where $\hat{x}$ is the true solution.

For details of the method, see the Chapter Introduction.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: TRANS – CHARACTER*1. *Input*

On entry: indicates the form of the linear equations for which X is the computed solution as follows:

if TRANS = 'N', then the linear equations are of the form $AX = B$;

if TRANS = 'T', then the linear equations are of the form $A^TX = B$;

if TRANS = 'C', then the linear equations are of the form $A^H X = B$.

Constraint: TRANS = 'N', 'T' or 'C'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: KL – INTEGER. *Input*

On entry: k_l, the number of sub-diagonals within the band of A.

Constraint: KL ≥ 0.

4: KU – INTEGER. *Input*

On entry: k_u, the number of super-diagonals within the band of A.

Constraint: KU ≥ 0.

5: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS ≥ 0.

6: AB(LDAB,*) – ***complex*** array. *Input*

Note: the second dimension of the array AB must be at least max(1,N).

On entry: the n by n original band matrix A as supplied to F07BRF (CGBTRF/ZGBTRF), but stored in rows 1 to (k_l+k_u+1) of the array rather than in rows (k_l+1) to $(2k_l+k_u+1)$.

7: LDAB – INTEGER. *Input*

On entry: the first dimension of the array AB as declared in the (sub)program from which F07BVF (CGBRFS/ZGBRFS) is called.

Constraint: LDAB ≥ KL + KU + 1.

8: AFB(LDAFB,*) – ***complex*** array. *Input*

Note: the second dimension of the array AFB must be at least max(1,N).

On entry: the LU factorization of A, as returned by F07BRF (CGBTRF/ZGBTRF).

9: LDAFB – INTEGER. *Input*

On entry: the first dimension of the array AFB as declared in the (sub)program from which F07BVF (CGBRFS/ZGBRFS) is called.

Constraint: LDAFB ≥ 2×KL + KU + 1.

10: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: the pivot indices, as returned by F07BRF (CGBTRF/ZGBTRF).

11: B(LDB,*) – ***complex*** array. *Input*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

12: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07BVF (CGBRFS/ZGBRFS) is called.

Constraint: LDB ≥ max(1,N).

13: X(LDX,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array X must be at least max(1,NRHS).

On entry: the n by r solution matrix X, as returned by F07BSF (CGBTRS/ZGBTRS).

On exit: the improved solution matrix X.

14: LDX – INTEGER. *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which F07BVF (CGBRFS/ZGBRFS) is called.

Constraint: LDX ≥ max(1,N).

15: FERR(*) – ***real*** array. *Output*

Note: the dimension of the array FERR must be at least max(1,NRHS).

On exit: FERR(j) contains an estimated error bound for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

16: BERR(*) – ***real*** array. *Output*

Note: the dimension of the array BERR must be at least max(1,NRHS).

On exit: BERR(j) contains the component-wise backward error bound β for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

17: WORK(*) – ***complex*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,2*N).

18: RWORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array RWORK must be at least max(1,N).

19: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The bounds returned in FERR are not rigorous, because they are estimated, not computed exactly; but in practice they almost always overestimate the actual error.

8. Further Comments

For each right-hand side, computation of the backward error involves a minimum of $16n(k_l+k_u)$ real floating-point operations. Each step of iterative refinement involves an additional $8n(4k_l+3k_u)$ real operations. This assumes $n \gg k_l$ and $n \gg k_u$. At most 5 steps of iterative refinement are performed, but usually only 1 or 2 steps are required.

Estimating the forward error involves solving a number of systems of linear equations of the form $Ax = b$ or $A^Hx = b$; the number is usually 5 and never more than 11. Each solution involves approximately $8n(2k_l+k_u)$ real operations.

The real analogue of this routine is F07BHF (SGBRFS/DGBRFS).

9. Example

To solve the system of equations $AX = B$ using iterative refinement and to compute the forward and backward error bounds, where

$$A = \begin{pmatrix} -1.65 + 2.26i & -2.05 - 0.85i & 0.97 - 2.84i & 0.00 + 0.00i \\ 0.00 + 6.30i & -1.48 - 1.75i & -3.99 + 4.01i & 0.59 - 0.48i \\ 0.00 + 0.00i & -0.77 + 2.83i & -1.06 + 1.94i & 3.33 - 1.04i \\ 0.00 + 0.00i & 0.00 + 0.00i & 4.48 - 1.09i & -0.46 - 1.72i \end{pmatrix} \text{ and}$$

$$B = \begin{pmatrix} -1.06 + 21.50i & 12.85 + 2.84i \\ -22.72 - 53.90i & -70.22 + 21.57i \\ 28.24 - 38.60i & -20.73 - 1.23i \\ -34.56 + 16.73i & 26.01 + 31.97i \end{pmatrix}.$$

Here A is unsymmetric and is treated as a band matrix, which must first be factorized by F07BRF (CGBTRF/ZGBTRF).

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07BVF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      complex          ZERO
      PARAMETER        (ZERO=(0.0e0,0.0e0))
      INTEGER          NMAX, NRHMAX, KLMAX, KUMAX, LDAB, LDAFB, LDB, LDX
      PARAMETER        (NMAX=8,NRHMAX=NMAX,KLMAX=8,KUMAX=8,
     +                 LDAB=KLMAX+KUMAX+1,LDAFB=2*KLMAX+KUMAX+1,
     +                 LDB=NMAX,LDX=NMAX)
      CHARACTER        TRANS
      PARAMETER        (TRANS='N')
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, K, KL, KU, N, NRHS
*     .. Local Arrays ..
      complex          AB(LDAB,NMAX), AFB(LDAFB,NMAX), B(LDB,NRHMAX),
     +                 WORK(2*NMAX), X(LDX,NMAX)
      real             BERR(NRHMAX), FERR(NRHMAX), RWORK(NMAX)
      INTEGER          IPIV(NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         cgbrfs, cgbtrf, cgbtrs, F06TFF, F06THF, X04DBF
*     .. Intrinsic Functions ..
      INTRINSIC        MAX, MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07BVF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS, KL, KU
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX .AND. KL.LE.KLMAX .AND. KU.LE.
     +    KUMAX) THEN
*
*        Set A to zero to avoid referencing uninitialized elements
*
         CALL F06THF('General',KL+KU+1,N,ZERO,ZERO,AB,LDAB)
*
*        Read A and B from data file, and copy A to AFB and B to X
*
         K = KU + 1
         READ (NIN,*) ((AB(K+I-J,J),J=MAX(I-KL,1),MIN(I+KU,N)),I=1,N)
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
         CALL F06TFF('General',KL+KU+1,N,AB,LDAB,AFB(KL+1,1),LDAFB)
         CALL F06TFF('General',N,NRHS,B,LDB,X,LDX)
*
```

```
*        Factorize A in the array AFB
*
         CALL cgbtrf(N,N,KL,KU,AFB,LDAFB,IPIV,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute solution in the array X
*
            CALL cgbtrs(TRANS,N,KL,KU,NRHS,AFB,LDAFB,IPIV,X,LDX,INFO)
*
*           Improve solution, and compute backward errors and
*           estimated bounds on the forward errors
*
            CALL cgbrfs(TRANS,N,KL,KU,NRHS,AB,LDAB,AFB,LDAFB,IPIV,B,
     +                  LDB,X,LDX,FERR,BERR,WORK,RWORK,INFO)
*
*           Print solution
*
            IFAIL = 0
            CALL X04DBF('General',' ',N,NRHS,X,LDX,'Bracketed','F7.4',
     +                  'Solution(s)','Integer',RLABS,'Integer',CLABS,
     +                  80,0,IFAIL)
            WRITE (NOUT,*)
            WRITE (NOUT,*) 'Backward errors (machine-dependent)'
            WRITE (NOUT,99999) (BERR(J),J=1,NRHS)
            WRITE (NOUT,*)
     +        'Estimated forward error bounds (machine-dependent)'
            WRITE (NOUT,99999) (FERR(J),J=1,NRHS)
         ELSE
            WRITE (NOUT,*) 'The factor U is singular'
         END IF
       END IF
       STOP
*
99999  FORMAT ((5X,1P,4(e11.1,7X)))
       END
```

9.2. Program Data

```
F07BVF Example Program Data
  4  2  1  2                                   :Values of N, NRHS, KL and KU
 (-1.65, 2.26) (-2.05,-0.85) ( 0.97,-2.84)
 ( 0.00, 6.30) (-1.48,-1.75) (-3.99, 4.01) ( 0.59,-0.48)
               (-0.77, 2.83) (-1.06, 1.94) ( 3.33,-1.04)
                             ( 4.48,-1.09) (-0.46,-1.72)  :End of matrix A
 ( -1.06, 21.50) ( 12.85,  2.84)
 (-22.72,-53.90) (-70.22, 21.57)
 ( 28.24,-38.60) (-20.73, -1.23)
 (-34.56, 16.73) ( 26.01, 31.97)                          :End of matrix B
```

9.3. Program Results

```
 F07BVF Example Program Results

 Solution(s)
                        1                   2
  1  (-3.0000, 2.0000) ( 1.0000, 6.0000)
  2  ( 1.0000,-7.0000) (-7.0000,-4.0000)
  3  (-5.0000, 4.0000) ( 3.0000, 5.0000)
  4  ( 6.0000,-8.0000) (-8.0000, 2.0000)

 Backward errors (machine-dependent)
         3.8E-17           4.3E-17
 Estimated forward error bounds (machine-dependent)
         3.6E-14           4.3E-14
```

F07FDF (SPOTRF/DPOTRF) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07FDF (SPOTRF/DPOTRF) computes the Cholesky factorization of a real symmetric positive-definite matrix.

2. Specification

```
SUBROUTINE F07FDF (UPLO, N, A, LDA, INFO)
ENTRY      spotrf  (UPLO, N, A, LDA, INFO)
INTEGER          N, LDA, INFO
real             A(LDA,*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine forms the Cholesky factorization of a real symmetric positive-definite matrix A either as $A = U^T U$ if UPLO = 'U', or $A = LL^T$ if UPLO = 'L', where U is an upper triangular matrix and L is lower triangular.

4. References

[1] DEMMEL, J.W.
On Floating-point Errors in Cholesky.
LAPACK Working Note No. 14, University of Tennessee, Knoxville, 1989.

[2] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §4.2.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether the upper or lower triangular part of A is stored and how A is factorized, as follows:

if UPLO = 'U', then the upper triangular part of A is stored and A is factorized as $U^T U$, where U is upper triangular;

if UPLO = 'L', then the lower triangular part of A is stored and A is factorized as LL^T, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: A(LDA,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the n by n symmetric positive-definite matrix A. If UPLO = 'U', the upper triangle of A must be stored and the elements of the array below the diagonal are not referenced; if UPLO = 'L', the lower triangle of A must be stored and the elements of the array above the diagonal are not referenced.

On exit: the upper or lower triangle of A is overwritten by the Cholesky factor U or L as specified by UPLO.

4: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07FDF (SPOTRF/DPOTRF) is called.

Constraint: LDA ≥ max(1,N).

5: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, the leading minor of order i is not positive-definite and the factorization could not be completed. Hence A itself is not positive-definite. This may indicate an error in forming the matrix A. To factorize a symmetric matrix which is not positive-definite, call F07MDF (SSYTRF/DSYTRF) instead.

7. Accuracy

If UPLO = 'U', the computed factor U is the exact factor of a perturbed matrix $A + E$, where

$$|E| \leq c(n)\varepsilon|U^T||U|,$$

$c(n)$ is a modest linear function of n, and ε is the ***machine precision***. If UPLO = 'L', a similar statement holds for the computed factor L. It follows that $|e_{ij}| \leq c(n)\varepsilon\sqrt{a_{ii}a_{jj}}$.

8. Further Comments

The total number of floating-point operations is approximately $\frac{1}{3}n^3$.

A call to this routine may be followed by calls to the routines:

F07FEF (SPOTRS/DPOTRS) to solve $AX = B$;
F07FGF (SPOCON/DPOCON) to estimate the condition number of A;
F07FJF (SPOTRI/DPOTRI) to compute the inverse of A.

The complex analogue of this routine is F07FRF (CPOTRF/ZPOTRF).

9. Example

To compute the Cholesky factorization of the matrix A, where

$$A = \begin{pmatrix} 4.16 & -3.12 & 0.56 & -0.10 \\ -3.12 & 5.03 & -0.83 & 1.18 \\ 0.56 & -0.83 & 0.76 & 0.34 \\ -0.10 & 1.18 & 0.34 & 1.18 \end{pmatrix}.$$

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07FDF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA
      PARAMETER        (NMAX=8,LDA=NMAX)
```

```
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             A(LDA,NMAX)
*     .. External Subroutines ..
      EXTERNAL         spotrf, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07FDF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
*
*        Factorize A
*
         CALL spotrf(UPLO,N,A,LDA,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Print factor
*
            IFAIL = 0
            CALL X04CAF(UPLO,'Nonunit',N,N,A,LDA,'Factor',IFAIL)
         ELSE
            WRITE (NOUT,*) 'A is not positive-definite'
         END IF
      END IF
      STOP
*
      END
```

9.2. Program Data

```
F07FDF Example Program Data
  4                              :Value of N
  'L'                            :Value of UPLO
  4.16
 -3.12   5.03
  0.56  -0.83   0.76
 -0.10   1.18   0.34   1.18     :End of matrix A
```

9.3. Program Results

```
 F07FDF Example Program Results

 Factor
              1          2          3          4
  1      2.0396
  2     -1.5297     1.6401
  3      0.2746    -0.2500     0.7887
  4     -0.0490     0.6737     0.6617     0.5347
```

F07FEF (SPOTRS/DPOTRS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07FEF (SPOTRS/DPOTRS) solves a real symmetric positive-definite system of linear equations with multiple right-hand sides, $AX = B$, where A has been factorized by F07FDF (SPOTRF/DPOTRF).

2. Specification

```
SUBROUTINE F07FEF (UPLO, N, NRHS, A, LDA, B, LDB, INFO)
ENTRY      spotrs  (UPLO, N, NRHS, A, LDA, B, LDB, INFO)
INTEGER          N, NRHS, LDA, LDB, INFO
real             A(LDA,*), B(LDB,*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

To solve a real symmetric positive-definite system of linear equations $AX = B$, this routine must be preceded by a call to F07FDF (SPOTRF/DPOTRF) which computes the Cholesky factorization of A. The solution X is computed by forward and backward substitution.

If UPLO = 'U', $A = U^TU$, where U is upper triangular; the solution X is computed by solving $U^TY = B$ and then $UX = Y$.

If UPLO = 'L', $A = LL^T$, where L is lower triangular; the solution X is computed by solving $LY = B$ and then $L^TX = Y$.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §4.2.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A has been factorized as U^TU or LL^T as follows:

if UPLO = 'U', then $A = U^TU$, where U is upper triangular;

if UPLO = 'L', then $A = LL^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS ≥ 0.

4: A(LDA,*) – ***real*** array. *Input*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the Cholesky factor of A, as returned by F07FDF (SPOTRF/DPOTRF).

5: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07FEF (SPOTRS/DPOTRS) is called.

Constraint: LDA ≥ max(1,N).

6: B(LDB,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

On exit: the n by r solution matrix X.

7: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07FEF (SPOTRS/DPOTRS) is called.

Constraint: LDB ≥ max(1,N).

8: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

For each right-hand side vector b, the computed solution x is the exact solution of a perturbed system of equations $(A+E)x = b$, where

$|E| \leq c(n)\varepsilon|U^T||U|$ if UPLO = 'U',

$|E| \leq c(n)\varepsilon|L||L^T|$ if UPLO = 'L',

$c(n)$ is a modest linear function of n, and ε is the ***machine precision***.

If $\hat{x}$ is the true solution, then the computed solution x satisfies a forward error bound of the form

$$\frac{\|x-\hat{x}\|_\infty}{\|x\|_\infty} \leq c(n)\text{cond}(A,x)\varepsilon$$

where $\text{cond}(A,x) = \||A^{-1}||A||x|\|_\infty / \|x\|_\infty \leq \text{cond}(A) = \||A^{-1}||A|\|_\infty \leq \kappa_\infty(A)$. Note that $\text{cond}(A,x)$ can be much smaller than $\text{cond}(A)$.

Forward and backward error bounds can be computed by calling F07FHF (SPORFS/DPORFS), and an estimate for $\kappa_\infty(A)$ $(= \kappa_1(A))$ can be obtained by calling F07FGF (SPOCON/DPOCON).

8. Further Comments

The total number of floating-point operations is approximately $2n^2r$.

This routine may be followed by a call to F07FHF (SPORFS/DPORFS) to refine the solution and return an error estimate.

The complex analogue of this routine is F07FSF (CPOTRS/ZPOTRS).

9. Example

To solve the system of equations $AX = B$, where

$$A = \begin{pmatrix} 4.16 & -3.12 & 0.56 & -0.10 \\ -3.12 & 5.03 & -0.83 & 1.18 \\ 0.56 & -0.83 & 0.76 & 0.34 \\ -0.10 & 1.18 & 0.34 & 1.18 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 8.70 & 8.30 \\ -13.35 & 2.13 \\ 1.89 & 1.61 \\ -4.14 & 5.00 \end{pmatrix}.$$

Here A is symmetric positive-definite and must first be factorized by F07FDF (SPOTRF/DPOTRF).

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07FEF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, NRHMAX, LDB
      PARAMETER        (NMAX=8,LDA=NMAX,NRHMAX=NMAX,LDB=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             A(LDA,NMAX), B(LDB,NRHMAX)
*     .. External Subroutines ..
      EXTERNAL         spotrf, spotrs, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07FEF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
*
*        Factorize A
*
         CALL spotrf(UPLO,N,A,LDA,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute solution
*
            CALL spotrs(UPLO,N,NRHS,A,LDA,B,LDB,INFO)
*
*           Print solution
*
            IFAIL = 0
            CALL X04CAF('General',' ',N,NRHS,B,LDB,'Solution(s)',IFAIL)
         ELSE
```

```
            WRITE (NOUT,*) 'A is not positive-definite'
          END IF
        END IF
        STOP
*
        END
```

9.2. Program Data

```
F07FEF Example Program Data
  4  2                          :Values of N and NRHS
  'L'                           :Value of UPLO
  4.16
 -3.12   5.03
  0.56  -0.83   0.76
 -0.10   1.18   0.34   1.18    :End of matrix A
  8.70   8.30
-13.35   2.13
  1.89   1.61
 -4.14   5.00                   :End of matrix B
```

9.3. Program Results

```
 F07FEF Example Program Results

 Solution(s)
                1           2
  1      1.0000      4.0000
  2     -1.0000      3.0000
  3      2.0000      2.0000
  4     -3.0000      1.0000
```

F07FGF (SPOCON/DPOCON) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07FGF (SPOCON/DPOCON) estimates the condition number of a real symmetric positive-definite matrix A, where A has been factorized by F07FDF (SPOTRF/DPOTRF).

2. Specification

```
SUBROUTINE F07FGF (UPLO, N, A, LDA, ANORM, RCOND, WORK, IWORK, INFO)
ENTRY      spocon (UPLO, N, A, LDA, ANORM, RCOND, WORK, IWORK, INFO)
INTEGER          N, LDA, IWORK(*), INFO
real             A(LDA,*), ANORM, RCOND, WORK(*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine estimates the condition number (in the 1-norm) of a real symmetric positive-definite matrix A:

$$\kappa_1(A) = \|A\|_1 \|A^{-1}\|_1.$$

Since A is symmetric, $\kappa_1(A) = \kappa_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty$.

Because $\kappa_1(A)$ is infinite if A is singular, the routine actually returns an estimate of the **reciprocal** of $\kappa_1(A)$.

The routine should be preceded by a call to F06RCF to compute $\|A\|_1$ and a call to F07FDF (SPOTRF/DPOTRF) to compute the Cholesky factorization of A. The routine then uses Higham's implementation of Hager's method [1] to estimate $\|A^{-1}\|_1$.

4. References

[1] HIGHAM, N.J.
FORTRAN Codes for Estimating the One-norm of a Real or Complex Matrix, with Applications to Condition Estimation.
ACM Trans. Math. Softw., 14, pp. 381-396, 1988.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A has been factorized as U^TU or LL^T as follows:

if UPLO = 'U', then $A = U^TU$, where U is upper triangular;

if UPLO = 'L', then $A = LL^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: A(LDA,*) – ***real*** array. *Input*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the Cholesky factor of A, as returned by F07FDF (SPOTRF/DPOTRF).

4: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07FGF (SPOCON/DPOCON) is called.

Constraint: LDA ≥ max(1,N).

5: ANORM – ***real***. *Input*

On entry: the 1-norm of the **original** matrix A, which may be computed by calling F06RCF. ANORM must be computed either **before** calling F07FDF (SPOTRF/DPOTRF) or else from a copy of the original matrix A.

Constraint: ANORM ≥ 0.0.

6: RCOND – ***real***. *Output*

On exit: an estimate of the reciprocal of the condition number of A. RCOND is set to zero if exact singularity is detected or the estimate underflows. If RCOND is less than ***machine precision***, then A is singular to working precision.

7: WORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,3*N).

8: IWORK(*) – INTEGER array. *Workspace*

Note: the dimension of the array IWORK must be at least max(1,N).

9: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed estimate RCOND is never less than the true value ρ, and in practice is nearly always less than 10ρ, although examples can be constructed where RCOND is much larger.

8. Further Comments

A call to this routine involves solving a number of systems of linear equations of the form $Ax = b$; the number is usually 4 or 5 and never more than 11. Each solution involves approximately $2n^2$ floating-point operations but takes considerably longer than a call to F07FEF (SPOTRS/DPOTRS) with 1 right-hand side, because extra care is taken to avoid overflow when A is approximately singular.

The complex analogue of this routine is F07FUF (CPOCON/ZPOCON).

9. Example

To estimate the condition number in the 1-norm (or infinity-norm) of the matrix A, where

$$A = \begin{pmatrix} 4.16 & -3.12 & 0.56 & -0.10 \\ -3.12 & 5.03 & -0.83 & 1.18 \\ 0.56 & -0.83 & 0.76 & 0.34 \\ -0.10 & 1.18 & 0.34 & 1.18 \end{pmatrix}.$$

Here A is symmetric positive-definite and must first be factorized by F07FDF (SPOTRF/DPOTRF). The true condition number in the 1-norm is 97.32.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07FGF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA
      PARAMETER        (NMAX=8,LDA=NMAX)
*     .. Local Scalars ..
      real             ANORM, RCOND
      INTEGER          I, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             A(LDA,NMAX), WORK(3*NMAX)
      INTEGER          IWORK(NMAX)
*     .. External Functions ..
      real             F06RCF, X02AJF
      EXTERNAL         F06RCF, X02AJF
*     .. External Subroutines ..
      EXTERNAL         spocon, spotrf
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07FGF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
*
*        Compute norm of A
*
         ANORM = F06RCF('1-norm',UPLO,N,A,LDA,WORK)
*
*        Factorize A
*
         CALL spotrf(UPLO,N,A,LDA,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Estimate condition number
*
            CALL spocon(UPLO,N,A,LDA,ANORM,RCOND,WORK,IWORK,INFO)
*
            IF (RCOND.GE.X02AJF()) THEN
               WRITE (NOUT,99999) 'Estimate of condition number =',
     +           1.0e0/RCOND
            ELSE
               WRITE (NOUT,*) 'A is singular to working precision'
            END IF
         ELSE
            WRITE (NOUT,*) 'A is not positive-definite'
         END IF
      END IF
      STOP
*
99999 FORMAT (1X,A,1P,e10.2)
      END
```

9.2. Program Data

```
F07FGF Example Program Data
  4                                :Value of N
  'L'                              :Value of UPLO
  4.16
 -3.12   5.03
  0.56  -0.83   0.76
 -0.10   1.18   0.34   1.18    :End of matrix A
```

9.3. Program Results

```
F07FGF Example Program Results

Estimate of condition number =  9.73E+01
```

F07FHF (SPORFS/DPORFS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07FHF (SPORFS/DPORFS) returns error bounds for the solution of a real symmetric positive-definite system of linear equations with multiple right-hand sides, $AX = B$. It improves the solution by iterative refinement, in order to reduce the backward error as much as possible.

2. Specification

```
      SUBROUTINE F07FHF (UPLO, N, NRHS, A, LDA, AF, LDAF, B, LDB, X, LDX, FERR,
     1                   BERR, WORK, IWORK, INFO)
      ENTRY      sporfs  (UPLO, N, NRHS, A, LDA, AF, LDAF, B, LDB, X, LDX, FERR,
     1                   BERR, WORK, IWORK, INFO)
      INTEGER            N, NRHS, LDA, LDAF, LDB, LDX, IWORK(*), INFO
      real               A(LDA,*), AF(LDAF,*), B(LDB,*), X(LDX,*), FERR(*),
     1                   BERR(*), WORK(*)
      CHARACTER*1        UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine returns the backward errors and estimated bounds on the forward errors for the solution of a real symmetric positive-definite system of linear equations with multiple right-hand sides $AX = B$. The routine handles each right-hand side vector (stored as a column of the matrix B) independently, so we describe the function of the routine in terms of a single right-hand side b and solution x.

Given a computed solution x, the routine computes the *component-wise backward error* β. This is the size of the smallest relative perturbation in each element of A and b such that x is the exact solution of a perturbed system

$$(A+\delta A)x = b + \delta b$$

$$|\delta a_{ij}| \leq \beta |a_{ij}| \text{ and } |\delta b_i| \leq \beta |b_i|.$$

Then the routine estimates a bound for the *component-wise forward error* in the computed solution, defined by:

$$\max_i |x_i - \hat{x}_i| / \max_i |x_i|$$

where $\hat{x}$ is the true solution.

For details of the method, see the Chapter Introduction.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether the upper or lower triangular part of A is stored and how A has been factorized, as follows:

if UPLO = 'U', then the upper triangular part of A is stored and A is factorized as $U^T U$, where U is upper triangular;

if UPLO = 'L', then the lower triangular part of A is stored and A is factorized as LL^T, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS ≥ 0.

4: A(LDA,*) – ***real*** array. *Input*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the n by n original symmetric positive-definite matrix A as supplied to F07FDF (SPOTRF/DPOTRF).

5: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07FHF (SPORFS/DPORFS) is called.

Constraint: LDA ≥ max(1,N).

6: AF(LDAF,*) – ***real*** array. *Input*

Note: the second dimension of the array AF must be at least max(1,N).

On entry: the Cholesky factor of A, as returned by F07FDF (SPOTRF/DPOTRF).

7: LDAF – INTEGER. *Input*

On entry: the first dimension of the array AF as declared in the (sub)program from which F07FHF (SPORFS/DPORFS) is called.

Constraint: LDAF ≥ max(1,N).

8: B(LDB,*) – ***real*** array. *Input*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

9: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07FHF (SPORFS/DPORFS) is called.

Constraint: LDB ≥ max(1,N).

10: X(LDX,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array X must be at least max(1,NRHS).

On entry: the n by r solution matrix X, as returned by F07FEF (SPOTRS/DPOTRS).

On exit: the improved solution matrix X.

11: LDX – INTEGER. *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which F07FHF (SPORFS/DPORFS) is called.

Constraint: LDX ≥ max(1,N).

12: FERR(*) – ***real*** array. *Output*

Note: the dimension of the array FERR must be at least max(1,NRHS).

On exit: FERR(j) contains an estimated error bound for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

13: BERR(*) – ***real*** array. *Output*

Note: the dimension of the array BERR must be at least max(1,NRHS).

On exit: BERR(j) contains the component-wise backward error bound β for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

14: WORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,3*N).

15: IWORK(*) – INTEGER array. *Workspace*

Note: the dimension of the array IWORK must be at least max(1,N).

16: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The bounds returned in FERR are not rigorous, because they are estimated, not computed exactly; but in practice they almost always overestimate the actual error.

8. Further Comments

For each right-hand side, computation of the backward error involves a minimum of $4n^2$ floating-point operations. Each step of iterative refinement involves an additional $6n^2$ operations. At most 5 steps of iterative refinement are performed, but usually only 1 or 2 steps are required.

Estimating the forward error involves solving a number of systems of linear equations of the form $Ax = b$; the number is usually 4 or 5 and never more than 11. Each solution involves approximately $2n^2$ operations.

The complex analogue of this routine is F07FVF (CPORFS/ZPORFS).

9. Example

To solve the system of equations $AX = B$ using iterative refinement and to compute the forward and backward error bounds, where

$$A = \begin{pmatrix} 4.16 & -3.12 & 0.56 & -0.10 \\ -3.12 & 5.03 & -0.83 & 1.18 \\ 0.56 & -0.83 & 0.76 & 0.34 \\ -0.10 & 1.18 & 0.34 & 1.18 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 8.70 & 8.30 \\ -13.35 & 2.13 \\ 1.89 & 1.61 \\ -4.14 & 5.00 \end{pmatrix}.$$

Here A is symmetric positive-definite and must first be factorized by F07FDF (SPOTRF/DPOTRF).

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07FHF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, NRHMAX, LDA, LDAF, LDB, LDX
      PARAMETER        (NMAX=8,NRHMAX=NMAX,LDA=NMAX,LDAF=NMAX,LDB=NMAX,
     +                 LDX=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             A(LDA,NMAX), AF(LDAF,NMAX), B(LDB,NRHMAX),
     +                 BERR(NRHMAX), FERR(NRHMAX), WORK(3*NMAX),
     +                 X(LDX,NMAX)
      INTEGER          IWORK(NMAX)
*     .. External Subroutines ..
      EXTERNAL         sporfs, spotrf, spotrs,
     +                 F06QFF, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07FHF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file, and copy A to AF and B to X
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
*
         CALL F06QFF(UPLO,N,N,A,LDA,AF,LDAF)
*
         CALL F06QFF('General',N,NRHS,B,LDB,X,LDX)
*
*        Factorize A in the array AF
*
         CALL spotrf(UPLO,N,AF,LDAF,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute solution in the array X
*
            CALL spotrs(UPLO,N,NRHS,AF,LDAF,X,LDX,INFO)
*
*           Improve solution, and compute backward errors and
*           estimated bounds on the forward errors
*
            CALL sporfs(UPLO,N,NRHS,A,LDA,AF,LDAF,B,LDB,X,LDX,FERR,BERR,
     +                  WORK,IWORK,INFO)
*
*           Print solution
*
            IFAIL = 0
*
```

```
            CALL X04CAF('General',' ',N,NRHS,X,LDX,'Solution(s)',IFAIL)
*
            WRITE (NOUT,*)
            WRITE (NOUT,*) 'Backward errors (machine-dependent)'
            WRITE (NOUT,99999) (BERR(J),J=1,NRHS)
            WRITE (NOUT,*)
     +        'Estimated forward error bounds (machine-dependent)'
            WRITE (NOUT,99999) (FERR(J),J=1,NRHS)
          ELSE
            WRITE (NOUT,*) 'A is not positive-definite'
          END IF
        END IF
        STOP
*
99999 FORMAT ((3X,1P,7e11.1))
      END
```

9.2. Program Data

```
F07FHF Example Program Data
  4  2                          :Values of N and NRHS
  'L'                           :Value of UPLO
  4.16
 -3.12   5.03
  0.56  -0.83   0.76
 -0.10   1.18   0.34   1.18    :End of matrix A
  8.70   8.30
-13.35   2.13
  1.89   1.61
 -4.14   5.00                   :End of matrix B
```

9.3. Program Results

```
 F07FHF Example Program Results

 Solution(s)
                1          2
  1      1.0000     4.0000
  2     -1.0000     3.0000
  3      2.0000     2.0000
  4     -3.0000     1.0000

 Backward errors (machine-dependent)
       4.0E-17    5.0E-17
 Estimated forward error bounds (machine-dependent)
       2.3E-14    2.3E-14
```

F07FJF (SPOTRI/DPOTRI) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07FJF (SPOTRI/DPOTRI) computes the inverse of a real symmetric positive-definite matrix A, where A has been factorized by F07FDF (SPOTRF/DPOTRF).

2. Specification

```
SUBROUTINE F07FJF (UPLO, N, A, LDA, INFO)
ENTRY      spotri  (UPLO, N, A, LDA, INFO)
INTEGER            N, LDA, INFO
real               A(LDA,*)
CHARACTER*1        UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

To compute the inverse of a real symmetric positive-definite matrix A, this routine must be preceded by a call to F07FDF (SPOTRF/DPOTRF), which computes the Cholesky factorization of A.

If UPLO = 'U', $A = U^TU$ and A^{-1} is computed by first inverting U and then forming $(U^{-1})(U^{-1})^T$.

If UPLO = 'L', $A = LL^T$ and A^{-1} is computed by first inverting L and then forming $(L^{-1})^T(L^{-1})$.

4. References

[1] DU CROZ, J.J. and HIGHAM, N.J.
Stability of Methods for Matrix Inversion.
LAPACK Working Note No. 27, University of Tennessee, Knoxville, 1990.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A has been factorized as U^TU or LL^T as follows:

if UPLO = 'U', then $A = U^TU$, where U is upper triangular;

if UPLO = 'L', then $A = LL^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: $N \geq 0$.

3: A(LDA,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the upper triangular matrix U if UPLO = 'U' or the lower triangular matrix L if UPLO = 'L', as returned by F07FDF (SPOTRF/DPOTRF).

On exit: U is overwritten by the upper triangle of A^{-1} if UPLO = 'U'; L is overwritten by the lower triangle of A^{-1} if UPLO = 'L'.

F07FRF (CPOTRF/ZPOTRF) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07FRF (CPOTRF/ZPOTRF) computes the Cholesky factorization of a complex Hermitian positive-definite matrix.

2. Specification

```
SUBROUTINE F07FRF (UPLO, N, A, LDA, INFO)
ENTRY      cpotrf  (UPLO, N, A, LDA, INFO)
INTEGER          N, LDA, INFO
complex          A(LDA,*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine forms the Cholesky factorization of a complex Hermitian positive-definite matrix A either as $A = U^H U$ if UPLO = 'U', or $A = LL^H$ if UPLO = 'L', where U is an upper triangular matrix and L is lower triangular.

4. References

[1] DEMMEL, J.W.
On Floating-point Errors in Cholesky.
LAPACK Working Note No. 14, University of Tennessee, Knoxville, 1989.

[2] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §4.2.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether the upper or lower triangular part of A is stored and how A is factorized, as follows:

if UPLO = 'U', then the upper triangular part of A is stored and A is factorized as $U^H U$, where U is upper triangular;

if UPLO = 'L', then the lower triangular part of A is stored and A is factorized as LL^H, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: A(LDA,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the n by n Hermitian positive-definite matrix A. If UPLO = 'U', the upper triangle of A must be stored and the elements of the array below the diagonal are not referenced; if UPLO = 'L', the lower triangle of A must be stored and the elements of the array above the diagonal are not referenced.

On exit: the upper or lower triangle of A is overwritten by the Cholesky factor U or L as specified by UPLO.

4: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07FRF (CPOTRF/ZPOTRF) is called.

Constraint: LDA ≥ max(1,N).

5: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, the leading minor of order i is not positive-definite and the factorization could not be completed. Hence A itself is not positive-definite. This may indicate an error in forming the matrix A. To factorize a Hermitian matrix which is not positive-definite, call F07MRF (CHETRF/ZHETRF) instead.

7. Accuracy

If UPLO = 'U', the computed factor U is the exact factor of a perturbed matrix $A + E$, where

$$|E| \leq c(n)\varepsilon|U^H||U|,$$

$c(n)$ is a modest linear function of n, and ε is the ***machine precision***. If UPLO = 'L', a similar statement holds for the computed factor L. It follows that $|e_{ij}| \leq c(n)\varepsilon\sqrt{a_{ii}a_{jj}}$.

8. Further Comments

The total number of real floating-point operations is approximately $\frac{4}{3}n^3$.

A call to this routine may be followed by calls to the routines:

F07FSF (CPOTRS/ZPOTRS) to solve $AX = B$;
F07FUF (CPOCON/ZPOCON) to estimate the condition number of A;
F07FWF (CPOTRI/ZPOTRI) to compute the inverse of A.

The real analogue of this routine is F07FDF (SPOTRF/DPOTRF).

9. Example

To compute the Cholesky factorization of the matrix A, where

$$A = \begin{pmatrix} 3.23 + 0.00i & 1.51 - 1.92i & 1.90 + 0.84i & 0.42 + 2.50i \\ 1.51 + 1.92i & 3.58 + 0.00i & -0.23 + 1.11i & -1.18 + 1.37i \\ 1.90 - 0.84i & -0.23 - 1.11i & 4.09 + 0.00i & 2.33 - 0.14i \\ 0.42 - 2.50i & -1.18 - 1.37i & 2.33 + 0.14i & 4.29 + 0.00i \end{pmatrix}.$$

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07FRF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA
      PARAMETER        (NMAX=8,LDA=NMAX)
```

```
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          A(LDA,NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         X04DBF, cpotrf
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07FRF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
*
*        Factorize A
*
         CALL cpotrf(UPLO,N,A,LDA,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Print factor
*
            IFAIL = 0
            CALL X04DBF(UPLO,'Nonunit',N,N,A,LDA,'Bracketed','F7.4',
     +                  'Factor','Integer',RLABS,'Integer',CLABS,80,0,
     +                  IFAIL)
         ELSE
            WRITE (NOUT,*) 'A is not positive-definite'
         END IF
      END IF
      STOP
*
      END
```

9.2. Program Data

```
F07FRF Example Program Data
  4                                                          :Value of N
  'L'                                                        :Value of UPLO
 (3.23, 0.00)
 (1.51, 1.92) ( 3.58, 0.00)
 (1.90,-0.84) (-0.23,-1.11) ( 4.09, 0.00)
 (0.42,-2.50) (-1.18,-1.37) ( 2.33, 0.14) ( 4.29, 0.00)  :End of matrix A
```

9.3. Program Results

```
 F07FRF Example Program Results

 Factor
                      1                  2                  3                  4
  1  ( 1.7972, 0.0000)
  2  ( 0.8402, 1.0683) ( 1.3164, 0.0000)
  3  ( 1.0572,-0.4674) (-0.4702, 0.3131) ( 1.5604, 0.0000)
  4  ( 0.2337,-1.3910) ( 0.0834, 0.0368) ( 0.9360, 0.9900) ( 0.6603, 0.0000)
```

F07FSF (CPOTRS/ZPOTRS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07FSF (CPOTRS/ZPOTRS) solves a complex Hermitian positive-definite system of linear equations with multiple right-hand sides, $AX = B$, where A has been factorized by F07FRF (CPOTRF/ZPOTRF).

2. Specification

```
SUBROUTINE F07FSF (UPLO, N, NRHS, A, LDA, B, LDB, INFO)
ENTRY      cpotrs  (UPLO, N, NRHS, A, LDA, B, LDB, INFO)
INTEGER          N, NRHS, LDA, LDB, INFO
complex          A(LDA,*), B(LDB,*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

To solve a complex Hermitian positive-definite system of linear equations $AX = B$, this routine must be preceded by a call to F07FRF (CPOTRF/ZPOTRF) which computes the Cholesky factorization of A. The solution X is computed by forward and backward substitution.

If UPLO = 'U', $A = U^H U$, where U is upper triangular; the solution X is computed by solving $U^H Y = B$ and then $UX = Y$.

If UPLO = 'L', $A = LL^H$, where L is lower triangular; the solution X is computed by solving $LY = B$ and then $L^H X = Y$.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §4.2.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A has been factorized as $U^H U$ or LL^H as follows:

if UPLO = 'U', then $A = U^H U$, where U is upper triangular;

if UPLO = 'L', then $A = LL^H$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: $N \geq 0$.

3: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: $NRHS \geq 0$.

4: A(LDA,*) – ***complex*** array. *Input*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the Cholesky factor of A, as returned by F07FRF (CPOTRF/ZPOTRF).

5: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07FSF (CPOTRS/ZPOTRS) is called.

Constraint: LDA ≥ max(1,N).

6: B(LDB,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

On exit: the n by r solution matrix X.

7: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07FSF (CPOTRS/ZPOTRS) is called.

Constraint: LDB ≥ max(1,N).

8: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

For each right-hand side vector b, the computed solution x is the exact solution of a perturbed system of equations $(A+E)x = b$, where

$|E| \leq c(n)\varepsilon|U^H||U|$ if UPLO = 'U',

$|E| \leq c(n)\varepsilon|L||L^H|$ if UPLO = 'L',

$c(n)$ is a modest linear function of n, and ε is the ***machine precision***.

If $\hat{x}$ is the true solution, then the computed solution x satisfies a forward error bound of the form

$$\frac{\|x-\hat{x}\|_\infty}{\|x\|_\infty} \leq c(n)\text{cond}(A,x)\varepsilon$$

where $\text{cond}(A,x) = \||A^{-1}||A||x|\|_\infty / \|x\|_\infty \leq \text{cond}(A) = \||A^{-1}||A|\|_\infty \leq \kappa_\infty(A)$. Note that $\text{cond}(A,x)$ can be much smaller than $\text{cond}(A)$.

Forward and backward error bounds can be computed by calling F07FVF (CPORFS/ZPORFS), and an estimate for $\kappa_\infty(A)$ $(= \kappa_1(A))$ can be obtained by calling F07FUF (CPOCON/ZPOCON).

8. Further Comments

The total number of real floating-point operations is approximately $8n^2r$.

This routine may be followed by a call to F07FVF (CPORFS/ZPORFS) to refine the solution and return an error estimate.

The real analogue of this routine is F07FEF (SPOTRS/DPOTRS).

9. Example

To solve the system of equations $AX = B$, where

$$A = \begin{pmatrix} 3.23 + 0.00i & 1.51 - 1.92i & 1.90 + 0.84i & 0.42 + 2.50i \\ 1.51 + 1.92i & 3.58 + 0.00i & -0.23 + 1.11i & -1.18 + 1.37i \\ 1.90 - 0.84i & -0.23 - 1.11i & 4.09 + 0.00i & 2.33 - 0.14i \\ 0.42 - 2.50i & -1.18 - 1.37i & 2.33 + 0.14i & 4.29 + 0.00i \end{pmatrix} \quad \text{and}$$

$$B = \begin{pmatrix} 3.93 - 6.14i & 1.48 + 6.58i \\ 6.17 + 9.42i & 4.65 - 4.75i \\ -7.17 - 21.83i & -4.91 + 2.29i \\ 1.99 - 14.38i & 7.64 - 10.79i \end{pmatrix}.$$

Here A is Hermitian positive-definite and must first be factorized by F07FRF (CPOTRF/ZPOTRF).

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07FSF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, NRHMAX, LDB
      PARAMETER        (NMAX=8,LDA=NMAX,NRHMAX=NMAX,LDB=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          A(LDA,NMAX), B(LDB,NRHMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         X04DBF, cpotrf, cpotrs
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07FSF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
*
*        Factorize A
*
         CALL cpotrf(UPLO,N,A,LDA,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute solution
*
            CALL cpotrs(UPLO,N,NRHS,A,LDA,B,LDB,INFO)
*
```

```
*           Print solution
*
            IFAIL = 0
            CALL X04DBF('General',' ',N,NRHS,B,LDB,'Bracketed','F7.4',
     +                  'Solution(s)','Integer',RLABS,'Integer',CLABS,
     +                  80,0,IFAIL)
         ELSE
            WRITE (NOUT,*) 'A is not positive-definite'
         END IF
       END IF
       STOP
*
       END
```

9.2. Program Data

```
F07FSF Example Program Data
  4  2                                                        :Values of N and NRHS
  'L'                                                            :Value of UPLO
 (3.23, 0.00)
 (1.51, 1.92) ( 3.58, 0.00)
 (1.90,-0.84) (-0.23,-1.11) ( 4.09, 0.00)
 (0.42,-2.50) (-1.18,-1.37) ( 2.33, 0.14) ( 4.29, 0.00)  :End of matrix A
 ( 3.93, -6.14) ( 1.48,  6.58)
 ( 6.17,  9.42) ( 4.65, -4.75)
 (-7.17,-21.83) (-4.91,  2.29)
 ( 1.99,-14.38) ( 7.64,-10.79)                           :End of matrix B
```

9.3. Program Results

```
F07FSF Example Program Results

Solution(s)
                      1                   2
 1   ( 1.0000,-1.0000) (-1.0000, 2.0000)
 2   ( 0.0000, 3.0000) ( 3.0000,-4.0000)
 3   (-4.0000,-5.0000) (-2.0000, 3.0000)
 4   ( 2.0000, 1.0000) ( 4.0000,-5.0000)
```

F07FUF (CPOCON/ZPOCON) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07FUF (CPOCON/ZPOCON) estimates the condition number of a complex Hermitian positive-definite matrix A, where A has been factorized by F07FRF (CPOTRF/ZPOTRF).

2. Specification

```
SUBROUTINE F07FUF (UPLO, N, A, LDA, ANORM, RCOND, WORK, RWORK, INFO)
ENTRY      cpocon (UPLO, N, A, LDA, ANORM, RCOND, WORK, RWORK, INFO)
INTEGER          N, LDA, INFO
real             ANORM, RCOND, RWORK(*)
complex          A(LDA,*), WORK(*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine estimates the condition number (in the 1-norm) of a complex Hermitian positive-definite matrix A:

$$\kappa_1(A) = \|A\|_1 \|A^{-1}\|_1.$$

Since A is Hermitian, $\kappa_1(A) = \kappa_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty$.

Because $\kappa_1(A)$ is infinite if A is singular, the routine actually returns an estimate of the **reciprocal** of $\kappa_1(A)$.

The routine should be preceded by a call to F06UCF to compute $\|A\|_1$ and a call to F07FRF (CPOTRF/ZPOTRF) to compute the Cholesky factorization of A. The routine then uses Higham's implementation of Hager's method [1] to estimate $\|A^{-1}\|_1$.

4. References

[1] HIGHAM, N.J.
FORTRAN Codes for Estimating the One-norm of a Real or Complex Matrix, with Applications to Condition Estimation.
ACM Trans. Math. Softw., 14, pp. 381-396, 1988.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A has been factorized as $U^H U$ or LL^H as follows:

if UPLO = 'U', then $A = U^H U$, where U is upper triangular;

if UPLO = 'L', then $A = LL^H$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: A(LDA,*) – ***complex*** array. *Input*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the Cholesky factor of A, as returned by F07FRF (CPOTRF/ZPOTRF).

4: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07FUF (CPOCON/ZPOCON) is called.

Constraint: LDA ≥ max(1,N).

5: ANORM – ***real***. *Input*

On entry: the 1-norm of the **original** matrix A, which may be computed by calling F06UCF. ANORM must be computed either **before** calling F07FRF (CPOTRF/ZPOTRF) or else from a copy of the original matrix A.

Constraint: ANORM ≥ 0.0.

6: RCOND – ***real***. *Output*

On exit: an estimate of the reciprocal of the condition number of A. RCOND is set to zero if exact singularity is detected or the estimate underflows. If RCOND is less than ***machine precision***, then A is singular to working precision.

7: WORK(*) – ***complex*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,2*N).

8: RWORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array RWORK must be at least max(1,N).

9: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed estimate RCOND is never less than the true value ρ, and in practice is nearly always less than 10ρ, although examples can be constructed where RCOND is much larger.

8. Further Comments

A call to this routine involves solving a number of systems of linear equations of the form $Ax = b$; the number is usually 5 and never more than 11. Each solution involves approximately $8n^2$ real floating-point operations but takes considerably longer than a call to F07FSF (CPOTRS/ZPOTRS) with 1 right-hand side, because extra care is taken to avoid overflow when A is approximately singular.

The real analogue of this routine is F07FGF (SPOCON/DPOCON).

9. Example

To estimate the condition number in the 1-norm (or infinity-norm) of the matrix A, where

$$A = \begin{pmatrix} 3.23 + 0.00i & 1.51 - 1.92i & 1.90 + 0.84i & 0.42 + 2.50i \\ 1.51 + 1.92i & 3.58 + 0.00i & -0.23 + 1.11i & -1.18 + 1.37i \\ 1.90 - 0.84i & -0.23 - 1.11i & 4.09 + 0.00i & 2.33 - 0.14i \\ 0.42 - 2.50i & -1.18 - 1.37i & 2.33 + 0.14i & 4.29 + 0.00i \end{pmatrix}.$$

Here A is Hermitian positive-definite and must first be factorized by F07FRF (CPOTRF/ZPOTRF). The true condition number in the 1-norm is 201.92.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07FUF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA
      PARAMETER        (NMAX=8,LDA=NMAX)
*     .. Local Scalars ..
      real             ANORM, RCOND
      INTEGER          I, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          A(LDA,NMAX), WORK(2*NMAX)
      real             RWORK(NMAX)
*     .. External Functions ..
      real             F06UCF, X02AJF
      EXTERNAL         F06UCF, X02AJF
*     .. External Subroutines ..
      EXTERNAL         cpocon, cpotrf
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07FUF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
*
*        Compute norm of A
*
         ANORM = F06UCF('1-norm',UPLO,N,A,LDA,RWORK)
*
*        Factorize A
*
         CALL cpotrf(UPLO,N,A,LDA,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Estimate condition number
*
            CALL cpocon(UPLO,N,A,LDA,ANORM,RCOND,WORK,RWORK,INFO)
*
            IF (RCOND.GE.X02AJF()) THEN
               WRITE (NOUT,99999) 'Estimate of condition number =',
     +           1.0e0/RCOND
            ELSE
               WRITE (NOUT,*) 'A is singular to working precision'
            END IF
         ELSE
            WRITE (NOUT,*) 'A is not positive-definite'
         END IF
      END IF
      STOP
*
99999 FORMAT (1X,A,1P,e10.2)
      END
```

9.2. Program Data

```
F07FUF Example Program Data
  4                                                          :Value of N
  'L'                                                        :Value of UPLO
 (3.23, 0.00)
 (1.51, 1.92) ( 3.58, 0.00)
 (1.90,-0.84) (-0.23,-1.11) ( 4.09, 0.00)
 (0.42,-2.50) (-1.18,-1.37) ( 2.33, 0.14) ( 4.29, 0.00)  :End of matrix A
```

9.3. Program Results

```
F07FUF Example Program Results

Estimate of condition number =  1.51E+02
```

F07FVF (CPORFS/ZPORFS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07FVF (CPORFS/ZPORFS) returns error bounds for the solution of a complex Hermitian positive-definite system of linear equations with multiple right-hand sides, $AX = B$. It improves the solution by iterative refinement, in order to reduce the backward error as much as possible.

2. Specification

```
      SUBROUTINE F07FVF (UPLO, N, NRHS, A, LDA, AF, LDAF, B, LDB, X, LDX, FERR,
     1                   BERR, WORK, RWORK, INFO)
      ENTRY       cporfs  (UPLO, N, NRHS, A, LDA, AF, LDAF, B, LDB, X, LDX, FERR,
     1                   BERR, WORK, RWORK, INFO)
      INTEGER           N, NRHS, LDA, LDAF, LDB, LDX, INFO
      real              FERR(*), BERR(*), RWORK(*)
      complex           A(LDA,*), AF(LDAF,*), B(LDB,*), X(LDX,*), WORK(*)
      CHARACTER*1       UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine returns the backward errors and estimated bounds on the forward errors for the solution of a complex Hermitian positive-definite system of linear equations with multiple right-hand sides $AX = B$. The routine handles each right-hand side vector (stored as a column of the matrix B) independently, so we describe the function of the routine in terms of a single right-hand side b and solution x.

Given a computed solution x, the routine computes the *component-wise backward error* β. This is the size of the smallest relative perturbation in each element of A and b such that x is the exact solution of a perturbed system

$$(A+\delta A)x = b + \delta b$$

$$|\delta a_{ij}| \leq \beta |a_{ij}| \quad \text{and} \quad |\delta b_i| \leq \beta |b_i|.$$

Then the routine estimates a bound for the *component-wise forward error* in the computed solution, defined by:

$$\max_i |x_i - \hat{x}_i| / \max_i |x_i|$$

where $\hat{x}$ is the true solution.

For details of the method, see the Chapter Introduction.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether the upper or lower triangular part of A is stored and how A has been factorized, as follows:

if UPLO = 'U', then the upper triangular part of A is stored and A is factorized as $U^H U$, where U is upper triangular;

if UPLO = 'L', then the lower triangular part of A is stored and A is factorized as LL^H, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS ≥ 0.

4: A(LDA,*) – ***complex*** array. *Input*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the n by n original Hermitian positive-definite matrix A as supplied to F07FRF (CPOTRF/ZPOTRF).

5: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07FVF (CPORFS/ZPORFS) is called.

Constraint: LDA ≥ max(1,N).

6: AF(LDAF,*) – ***complex*** array. *Input*

Note: the second dimension of the array AF must be at least max(1,N).

On entry: the Cholesky factor of A, as returned by F07FRF (CPOTRF/ZPOTRF).

7: LDAF – INTEGER. *Input*

On entry: the first dimension of the array AF as declared in the (sub)program from which F07FVF (CPORFS/ZPORFS) is called.

Constraint: LDAF ≥ max(1,N).

8: B(LDB,*) – ***complex*** array. *Input*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

9: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07FVF (CPORFS/ZPORFS) is called.

Constraint: LDB ≥ max(1,N).

10: X(LDX,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array X must be at least max(1,NRHS).

On entry: the n by r solution matrix X, as returned by F07FSF (CPOTRS/ZPOTRS).

On exit: the improved solution matrix X.

11: LDX – INTEGER. *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which F07FVF (CPORFS/ZPORFS) is called.

Constraint: LDX ≥ max(1,N).

12: FERR(*) – ***real*** array. *Output*

Note: the dimension of the array FERR must be at least max(1,NRHS).

On exit: FERR(j) contains an estimated error bound for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

13: BERR(*) – ***real*** array. *Output*

Note: the dimension of the array BERR must be at least max(1,NRHS).

On exit: BERR(j) contains the component-wise backward error bound β for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

14: WORK(*) – ***complex*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,2*N).

15: RWORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array RWORK must be at least max(1,N).

16: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The bounds returned in FERR are not rigorous, because they are estimated, not computed exactly; but in practice they almost always overestimate the actual error.

8. Further Comments

For each right-hand side, computation of the backward error involves a minimum of $16n^2$ real floating-point operations. Each step of iterative refinement involves an additional $24n^2$ real operations. At most 5 steps of iterative refinement are performed, but usually only 1 or 2 steps are required.

Estimating the forward error involves solving a number of systems of linear equations of the form $Ax = b$; the number is usually 5 and never more than 11. Each solution involves approximately $8n^2$ real operations.

The real analogue of this routine is F07FHF (SPORFS/DPORFS).

9. Example

To solve the system of equations $AX = B$ using iterative refinement and to compute the forward and backward error bounds, where

$$A = \begin{pmatrix} 3.23 + 0.00i & 1.51 - 1.92i & 1.90 + 0.84i & 0.42 + 2.50i \\ 1.51 + 1.92i & 3.58 + 0.00i & -0.23 + 1.11i & -1.18 + 1.37i \\ 1.90 - 0.84i & -0.23 - 1.11i & 4.09 + 0.00i & 2.33 - 0.14i \\ 0.42 - 2.50i & -1.18 - 1.37i & 2.33 + 0.14i & 4.29 + 0.00i \end{pmatrix} \text{ and}$$

$$B = \begin{pmatrix} 3.93 - 6.14i & 1.48 + 6.58i \\ 6.17 + 9.42i & 4.65 - 4.75i \\ -7.17 - 21.83i & -4.91 + 2.29i \\ 1.99 - 14.38i & 7.64 - 10.79i \end{pmatrix}.$$

Here A is Hermitian positive-definite and must first be factorized by F07FRF (CPOTRF/ZPOTRF).

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07FVF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, NRHMAX, LDA, LDAF, LDB, LDX
      PARAMETER        (NMAX=8,NRHMAX=NMAX,LDA=NMAX,LDAF=NMAX,LDB=NMAX,
     +                 LDX=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          A(LDA,NMAX), AF(LDAF,NMAX), B(LDB,NRHMAX),
     +                 WORK(2*NMAX), X(LDX,NMAX)
      real             BERR(NRHMAX), FERR(NRHMAX), RWORK(NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         F06TFF, X04DBF, cporfs,
                       cpotrf, cpotrs
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07FVF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file, and copy A to AF and B to X
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
         CALL F06TFF(UPLO,N,N,A,LDA,AF,LDAF)
         CALL F06TFF('General',N,NRHS,B,LDB,X,LDX)
*
*        Factorize A in the array AF
*
         CALL cpotrf(UPLO,N,AF,LDAF,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute solution in the array X
*
            CALL cpotrs(UPLO,N,NRHS,AF,LDAF,X,LDX,INFO)
*
*           Improve solution, and compute backward errors and
*           estimated bounds on the forward errors
*
            CALL cporfs(UPLO,N,NRHS,A,LDA,AF,LDAF,B,LDB,X,LDX,FERR,BERR,
     +                  WORK,RWORK,INFO)
*
*           Print solution
*
            IFAIL = 0
            CALL X04DBF('General',' ',N,NRHS,X,LDX,'Bracketed','F7.4',
     +                  'Solution(s)','Integer',RLABS,'Integer',CLABS,
     +                  80,0,IFAIL)
            WRITE (NOUT,*)
            WRITE (NOUT,*) 'Backward errors (machine-dependent)'
            WRITE (NOUT,99999) (BERR(J),J=1,NRHS)
```

```
            WRITE (NOUT,*)
     +        'Estimated forward error bounds (machine-dependent)'
            WRITE (NOUT,99999) (FERR(J),J=1,NRHS)
          ELSE
            WRITE (NOUT,*) 'A is not positive-definite'
          END IF
        END IF
        STOP
*
99999 FORMAT ((5X,1P,4(e11.1,7X)))
      END
```

9.2. Program Data

```
F07FVF Example Program Data
  4  2                                                        :Values of N and NRHS
  'L'                                                         :Value of UPLO
 (3.23, 0.00)
 (1.51, 1.92) ( 3.58, 0.00)
 (1.90,-0.84) (-0.23,-1.11) ( 4.09, 0.00)
 (0.42,-2.50) (-1.18,-1.37) ( 2.33, 0.14) ( 4.29, 0.00) :End of matrix A
 ( 3.93, -6.14) ( 1.48,  6.58)
 ( 6.17,  9.42) ( 4.65, -4.75)
 (-7.17,-21.83) (-4.91,  2.29)
 ( 1.99,-14.38) ( 7.64,-10.79)                           :End of matrix B
```

9.3. Program Results

```
 F07FVF Example Program Results

 Solution(s)
                        1                   2
  1   ( 1.0000,-1.0000) (-1.0000, 2.0000)
  2   ( 0.0000, 3.0000) ( 3.0000,-4.0000)
  3   (-4.0000,-5.0000) (-2.0000, 3.0000)
  4   ( 2.0000, 1.0000) ( 4.0000,-5.0000)

 Backward errors (machine-dependent)
         7.5E-17           5.3E-17
 Estimated forward error bounds (machine-dependent)
         6.1E-14           7.2E-14
```

F07FWF (CPOTRI/ZPOTRI) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07FWF (CPOTRI/ZPOTRI) computes the inverse of a complex Hermitian positive-definite matrix A, where A has been factorized by F07FRF (CPOTRF/ZPOTRF).

2. Specification

```
SUBROUTINE F07FWF (UPLO, N, A, LDA, INFO)
ENTRY      cpotri  (UPLO, N, A, LDA, INFO)
INTEGER          N, LDA, INFO
complex          A(LDA,*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

To compute the inverse of a complex Hermitian positive-definite matrix A, the routine must be preceded by a call to F07FRF (CPOTRF/ZPOTRF), which computes the Cholesky factorization of A.

If UPLO = 'U', $A = U^H U$ and A^{-1} is computed by first inverting U and then forming $(U^{-1})(U^{-1})^H$.

If UPLO = 'L', $A = LL^H$ and A^{-1} is computed by first inverting L and then forming $(L^{-1})^H(L^{-1})$.

4. References

[1] DU CROZ, J.J. and HIGHAM, N.J.
Stability of Methods for Matrix Inversion.
LAPACK Working Note No. 27, University of Tennessee, Knoxville, 1990.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A has been factorized as $U^H U$ or LL^H as follows:

if UPLO = 'U', then $A = U^H U$, where U is upper triangular;

if UPLO = 'L', then $A = LL^H$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: A(LDA,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the upper triangular matrix U if UPLO = 'U' or the lower triangular matrix L if UPLO = 'L', as returned by F07FRF (CPOTRF/ZPOTRF).

On exit: U is overwritten by the upper triangle of A^{-1} if UPLO = 'U'; L is overwritten by the lower triangle of A^{-1} if UPLO = 'L'.

4: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07FWF (CPOTRI/ZPOTRI) is called.

Constraint: LDA $\geq$ max(1,N).

5: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, the ith diagonal element of the Cholesky factor is zero; the Cholesky factor is singular and the inverse of A cannot be computed.

7. Accuracy

The computed inverse X satisfies

$$\|XA-I\|_2 \leq c(n)\varepsilon\kappa_2(A) \quad \text{and} \quad \|AX-I\|_2 \leq c(n)\varepsilon\kappa_2(A),$$

where $c(n)$ is a modest function of n, ε is the ***machine precision*** and $\kappa_2(A)$ is the condition number of A defined by

$$\kappa_2(A) = \|A\|_2\|A^{-1}\|_2.$$

8. Further Comments

The total number of real floating-point operations is approximately $\frac{8}{3}n^3$.

The real analogue of this routine is F07FJF (SPOTRI/DPOTRI).

9. Example

To compute the inverse of the matrix A, where

$$A = \begin{pmatrix} 3.23 + 0.00i & 1.51 - 1.92i & 1.90 + 0.84i & 0.42 + 2.50i \\ 1.51 + 1.92i & 3.58 + 0.00i & -0.23 + 1.11i & -1.18 + 1.37i \\ 1.90 - 0.84i & -0.23 - 1.11i & 4.09 + 0.00i & 2.33 - 0.14i \\ 0.42 - 2.50i & -1.18 - 1.37i & 2.33 + 0.14i & 4.29 + 0.00i \end{pmatrix}.$$

Here A is Hermitian positive-definite and must first be factorized by F07FRF (CPOTRF/ZPOTRF).

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07FWF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA
      PARAMETER        (NMAX=8,LDA=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
      CHARACTER        UPLO
```

```
*     .. Local Arrays ..
      complex           A(LDA,NMAX)
      CHARACTER         CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL          X04DBF, cpotrf, cpotri
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07FWF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
*
*        Factorize A
*
         CALL cpotrf(UPLO,N,A,LDA,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute inverse of A
*
            CALL cpotri(UPLO,N,A,LDA,INFO)
*
*           Print inverse
*
            IFAIL = 0
            CALL X04DBF(UPLO,'Nonunit',N,N,A,LDA,'Bracketed','F7.4',
     +                  'Inverse','Integer',RLABS,'Integer',CLABS,80,0,
     +                  IFAIL)
         ELSE
            WRITE (NOUT,*) 'A is not positive-definite'
         END IF
       END IF
       STOP
*
       END
```

9.2. Program Data

```
F07FWF Example Program Data
  4                                                          :Value of N
  'L'                                                        :Value of UPLO
 (3.23, 0.00)
 (1.51, 1.92) ( 3.58, 0.00)
 (1.90,-0.84) (-0.23,-1.11) ( 4.09, 0.00)
 (0.42,-2.50) (-1.18,-1.37) ( 2.33, 0.14) ( 4.29, 0.00)  :End of matrix A
```

9.3. Program Results

```
 F07FWF Example Program Results

 Inverse
                      1                   2                   3                   4
  1  ( 5.4691, 0.0000)
  2  (-1.2624,-1.5491) ( 1.1024, 0.0000)
  3  (-2.9746,-0.9616) ( 0.8989,-0.5672) ( 2.1589, 0.0000)
  4  ( 1.1962, 2.9772) (-0.9826,-0.2566) (-1.3756,-1.4550) ( 2.2934, 0.0000)
```

F07GDF (SPPTRF/DPPTRF) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07GDF (SPPTRF/DPPTRF) computes the Cholesky factorization of a real symmetric positive-definite matrix, using packed storage.

2. Specification

```
SUBROUTINE F07GDF (UPLO, N, AP, INFO)
ENTRY      spptrf  (UPLO, N, AP, INFO)
INTEGER          N, INFO
real             AP(*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine forms the Cholesky factorization of a real symmetric positive-definite matrix A either as $A = U^TU$ if UPLO = 'U', or $A = LL^T$ if UPLO = 'L', where U is an upper triangular matrix and L is lower triangular, using packed storage.

4. References

[1] DEMMEL, J.W.
On Floating-point Errors in Cholesky.
LAPACK Working Note No. 14, University of Tennessee, Knoxville, 1989.

[2] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §4.2.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether the upper or lower triangular part of A is stored and how A is factorized, as follows:

if UPLO = 'U', then the upper triangular part of A is stored and A is factorized as U^TU, where U is upper triangular;

if UPLO = 'L', then the lower triangular part of A is stored and A is factorized as LL^T, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: AP(*) – ***real*** array. *Input/Output*

Note: the dimension of the array AP must be at least max(1,N*(N+1)/2).

On entry: the n by n symmetric positive-definite matrix A, packed by columns. More precisely, if UPLO = 'U', the upper triangle of A must be stored with element a_{ij} in AP(i+j(j−1)/2) for $i \leq j$; if UPLO = 'L', the lower triangle of A must be stored with element a_{ij} in AP(i+(2n−j)(j−1)/2) for $i \geq j$.

On exit: the upper or lower triangle of A is overwritten by the Cholesky factor U or L as specified by UPLO, using the same packed storage format as described above.

4: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, the leading minor of order i is not positive-definite and the factorization could not be completed. Hence A itself is not positive-definite. This may indicate an error in forming the matrix A. To factorize a symmetric matrix which is not positive-definite, call F07PDF (SSPTRF/DSPTRF) instead.

7. Accuracy

If UPLO = 'U', the computed factor U is the exact factor of a perturbed matrix $A + E$, where

$$|E| \leq c(n)\varepsilon|U^T||U|,$$

$c(n)$ is a modest linear function of n, and ε is the ***machine precision***. If UPLO = 'L', a similar statement holds for the computed factor L. It follows that $|e_{ij}| \leq c(n)\varepsilon\sqrt{a_{ii}a_{jj}}$.

8. Further Comments

The total number of floating-point operations is approximately $\frac{1}{3}n^3$.

A call to this routine may be followed by calls to the routines:

F07GEF (SPPTRS/DPPTRS) to solve $AX = B$;
F07GGF (SPPCON/DPPCON) to estimate the condition number of A;
F07GJF (SPPTRI/DPPTRI) to compute the inverse of A.

The complex analogue of this routine is F07GRF (CPPTRF/ZPPTRF).

9. Example

To compute the Cholesky factorization of the matrix A, where

$$A = \begin{pmatrix} 4.16 & -3.12 & 0.56 & -0.10 \\ -3.12 & 5.03 & -0.83 & 1.18 \\ 0.56 & -0.83 & 0.76 & 0.34 \\ -0.10 & 1.18 & 0.34 & 1.18 \end{pmatrix},$$

using packed storage.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07GDF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=8)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             AP(NMAX*(NMAX+1)/2)
*     .. External Subroutines ..
      EXTERNAL         spptrf, X04CCF
```

```
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07GDF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
*
*        Factorize A
*
         CALL spptrf(UPLO,N,AP,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Print factor
*
            IFAIL = 0
*
            CALL X04CCF(UPLO,'Nonunit',N,AP,'Factor',IFAIL)
*
         ELSE
            WRITE (NOUT,*) 'A is not positive-definite'
         END IF
      END IF
      STOP
*
      END
```

9.2. Program Data

```
F07GDF Example Program Data
  4                                :Value of N
  'L'                              :Value of UPLO
  4.16
 -3.12   5.03
  0.56  -0.83   0.76
 -0.10   1.18   0.34   1.18    :End of matrix A
```

9.3. Program Results

```
F07GDF Example Program Results

Factor
             1          2          3          4
 1      2.0396
 2     -1.5297     1.6401
 3      0.2746    -0.2500     0.7887
 4     -0.0490     0.6737     0.6617     0.5347
```

F07GEF (SPPTRS/DPPTRS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07GEF (SPPTRS/DPPTRS) solves a real symmetric positive-definite system of linear equations with multiple right-hand sides, $AX = B$, where A has been factorized by F07GDF (SPPTRF/DPPTRF), using packed storage.

2. Specification

```
SUBROUTINE F07GEF (UPLO, N, NRHS, AP, B, LDB, INFO)
ENTRY      spptrs  (UPLO, N, NRHS, AP, B, LDB, INFO)
INTEGER          N, NRHS, LDB, INFO
real             AP(*), B(LDB,*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

To solve a real symmetric positive-definite system of linear equations $AX = B$, this routine must be preceded by a call to F07GDF (SPPTRF/DPPTRF) which computes the Cholesky factorization of A using packed storage. The solution X is computed by forward and backward substitution.

If UPLO = 'U', $A = U^TU$, where U is upper triangular; the solution X is computed by solving $U^TY = B$ and then $UX = Y$.

If UPLO = 'L', $A = LL^T$, where L is lower triangular; the solution X is computed by solving $LY = B$ and then $L^TX = Y$.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §4.2.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A has been factorized as U^TU or LL^T as follows:

if UPLO = 'U', then $A = U^TU$, where U is upper triangular;

if UPLO = 'L', then $A = LL^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS $\geq$ 0.

4: AP(*) – ***real*** array. *Input*

Note: the dimension of the array AP must be at least max(1,N*(N+1)/2).

On entry: the Cholesky factor of A stored in packed form, as returned by F07GDF (SPPTRF/DPPTRF).

5: B(LDB,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

On exit: the n by r solution matrix X.

6: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07GEF (SPPTRS/DPPTRS) is called.

Constraint: LDB $\geq$ max(1,N).

7: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

For each right-hand side vector b, the computed solution x is the exact solution of a perturbed system of equations $(A+E)x = b$, where

$|E| \leq c(n)\varepsilon|U^T||U|$ if UPLO = 'U',

$|E| \leq c(n)\varepsilon|L||L^T|$ if UPLO = 'L',

$c(n)$ is a modest linear function of n, and ε is the ***machine precision***.

If $\hat{x}$ is the true solution, then the computed solution x satisfies a forward error bound of the form

$$\frac{\|x-\hat{x}\|_\infty}{\|x\|_\infty} \leq c(n)\text{cond}(A,x)\varepsilon$$

where $\text{cond}(A,x) = \||A^{-1}||A||x|\|_\infty / \|x\|_\infty \leq \text{cond}(A) = \||A^{-1}||A|\|_\infty \leq \kappa_\infty(A)$. Note that $\text{cond}(A,x)$ can be much smaller than $\text{cond}(A)$.

Forward and backward error bounds can be computed by calling F07GHF (SPPRFS/DPPRFS), and an estimate for $\kappa_\infty(A)$ $(= \kappa_1(A))$ can be obtained by calling F07GGF (SPPCON/DPPCON).

8. Further Comments

The total number of floating-point operations is approximately $2n^2r$.

This routine may be followed by a call to F07GHF (SPPRFS/DPPRFS) to refine the solution and return an error estimate.

The complex analogue of this routine is F07GSF (CPPTRS/ZPPTRS).

9. Example

To solve the system of equations $AX = B$, where

$$A = \begin{pmatrix} 4.16 & -3.12 & 0.56 & -0.10 \\ -3.12 & 5.03 & -0.83 & 1.18 \\ 0.56 & -0.83 & 0.76 & 0.34 \\ -0.10 & 1.18 & 0.34 & 1.18 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 8.70 & 8.30 \\ -13.35 & 2.13 \\ 1.89 & 1.61 \\ -4.14 & 5.00 \end{pmatrix}.$$

Here A is symmetric positive-definite, stored in packed form, and must first be factorized by F07GDF (SPPTRF/DPPTRF).

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07GEF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, NRHMAX, LDB
      PARAMETER        (NMAX=8,NRHMAX=NMAX,LDB=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             AP(NMAX*(NMAX+1)/2), B(LDB,NRHMAX)
*     .. External Subroutines ..
      EXTERNAL         spptrf, spptrs, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07GEF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
*
*        Factorize A
*
         CALL spptrf(UPLO,N,AP,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute solution
*
            CALL spptrs(UPLO,N,NRHS,AP,B,LDB,INFO)
*
*           Print solution
*
            IFAIL = 0
*
```

```
            CALL X04CAF('General',' ',N,NRHS,B,LDB,'Solution(s)',IFAIL)
*
         ELSE
            WRITE (NOUT,*) 'A is not positive-definite'
         END IF
      END IF
      STOP
*
      END
```

9.2. Program Data

```
F07GEF Example Program Data
  4  2                          :Values of N and NRHS
  'L'                           :Value of UPLO
  4.16
 -3.12   5.03
  0.56  -0.83   0.76
 -0.10   1.18   0.34   1.18    :End of matrix A
  8.70   8.30
-13.35   2.13
  1.89   1.61
 -4.14   5.00                   :End of matrix B
```

9.3. Program Results

```
 F07GEF Example Program Results

 Solution(s)
                  1           2
  1       1.0000      4.0000
  2      -1.0000      3.0000
  3       2.0000      2.0000
  4      -3.0000      1.0000
```

F07GGF (SPPCON/DPPCON) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07GGF (SPPCON/DPPCON) estimates the condition number of a real symmetric positive-definite matrix A, where A has been factorized by F07GDF (SPPTRF/DPPTRF), using packed storage.

2. Specification

```
SUBROUTINE F07GGF (UPLO, N, AP, ANORM, RCOND, WORK, IWORK, INFO)
ENTRY      sppcon (UPLO, N, AP, ANORM, RCOND, WORK, IWORK, INFO)
INTEGER          N, IWORK(*), INFO
real             AP(*), ANORM, RCOND, WORK(*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine estimates the condition number (in the 1-norm) of a real symmetric positive-definite matrix A:

$$\kappa_1(A) = \|A\|_1 \|A^{-1}\|_1.$$

Since A is symmetric, $\kappa_1(A) = \kappa_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty$.

Because $\kappa_1(A)$ is infinite if A is singular, the routine actually returns an estimate of the **reciprocal** of $\kappa_1(A)$.

The routine should be preceded by a call to F06RDF to compute $\|A\|_1$ and a call to F07GDF (SPPTRF/DPPTRF) to compute the Cholesky factorization of A. The routine then uses Higham's implementation of Hager's method [1] to estimate $\|A^{-1}\|_1$.

4. References

[1] HIGHAM, N.J.
FORTRAN Codes for Estimating the One-norm of a Real or Complex Matrix, with Applications to Condition Estimation.
ACM Trans. Math. Softw., 14, pp. 381-396, 1988.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A has been factorized as U^TU or LL^T as follows:

if UPLO = 'U', then $A = U^TU$, where U is upper triangular;

if UPLO = 'L', then $A = LL^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: AP(*) – ***real*** array. *Input*

Note: the dimension of the array AP must be at least max(1,N*(N+1)/2).

On entry: the Cholesky factor of A stored in packed form, as returned by F07GDF (SPPTRF/DPPTRF).

4: ANORM – ***real.*** *Input*

On entry: the 1-norm of the **original** matrix A, which may be computed by calling F06RDF. ANORM must be computed either **before** calling F07GDF (SPPTRF/DPPTRF) or else from a copy of the original matrix A.

Constraint: ANORM $\geq$ 0.0.

5: RCOND – ***real.*** *Output*

On exit: an estimate of the reciprocal of the condition number of A. RCOND is set to zero if exact singularity is detected or the estimate underflows. If RCOND is less than ***machine precision***, then A is singular to working precision.

6: WORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,3*N).

7: IWORK(*) – INTEGER array. *Workspace*

Note: the dimension of the array IWORK must be at least max(1,N).

8: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed estimate RCOND is never less than the true value ρ, and in practice is nearly always less than 10ρ, although examples can be constructed where RCOND is much larger.

8. Further Comments

A call to this routine involves solving a number of systems of linear equations of the form $Ax = b$; the number is usually 4 or 5 and never more than 11. Each solution involves approximately $2n^2$ floating-point operations but takes considerably longer than a call to F07GEF (SPPTRS/DPPTRS) with 1 right-hand side, because extra care is taken to avoid overflow when A is approximately singular.

The complex analogue of this routine is F07GUF (CPPCON/ZPPCON).

9. Example

To estimate the condition number in the 1-norm (or infinity-norm) of the matrix A, where

$$A = \begin{pmatrix} 4.16 & -3.12 & 0.56 & -0.10 \\ -3.12 & 5.03 & -0.83 & 1.18 \\ 0.56 & -0.83 & 0.76 & 0.34 \\ -0.10 & 1.18 & 0.34 & 1.18 \end{pmatrix}.$$

Here A is symmetric positive-definite, stored in packed form, and must first be factorized by F07GDF (SPPTRF/DPPTRF). The true condition number in the 1-norm is 97.32.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07GGF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=8)
*     .. Local Scalars ..
      real             ANORM, RCOND
      INTEGER          I, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             AP(NMAX*(NMAX+1)/2), WORK(3*NMAX)
      INTEGER          IWORK(NMAX)
*     .. External Functions ..
      real             F06RDF, X02AJF
      EXTERNAL         F06RDF, X02AJF
*     .. External Subroutines ..
      EXTERNAL         sppcon, spptrf
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07GGF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
*
*        Compute norm of A
*
         ANORM = F06RDF('1-norm',UPLO,N,AP,WORK)
*
*        Factorize A
*
         CALL spptrf(UPLO,N,AP,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Estimate condition number
*
            CALL sppcon(UPLO,N,AP,ANORM,RCOND,WORK,IWORK,INFO)
*
            IF (RCOND.GE.X02AJF()) THEN
               WRITE (NOUT,99999) 'Estimate of condition number =',
     +           1.0e0/RCOND
            ELSE
               WRITE (NOUT,*) 'A is singular to working precision'
            END IF
         ELSE
            WRITE (NOUT,*) 'A is not positive-definite'
         END IF
      END IF
      STOP
*
99999 FORMAT (1X,A,1P,e10.2)
      END
```

9.2. Program Data

```
F07GGF Example Program Data
  4                                :Value of N
  'L'                              :Value of UPLO
  4.16
 -3.12   5.03
  0.56  -0.83   0.76
 -0.10   1.18   0.34   1.18    :End of matrix A
```

9.3. Program Results

```
 F07GGF Example Program Results

 Estimate of condition number =  9.73E+01
```

F07GHF (SPPRFS/DPPRFS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07GHF (SPPRFS/DPPRFS) returns error bounds for the solution of a real symmetric positive-definite system of linear equations with multiple right-hand sides, $AX = B$, using packed storage. It improves the solution by iterative refinement, in order to reduce the backward error as much as possible.

2. Specification

```
 SUBROUTINE F07GHF (UPLO, N, NRHS, AP, AFP, B, LDB, X, LDX, FERR, BERR,
1                   WORK, IWORK, INFO)
 ENTRY       spprfs  (UPLO, N, NRHS, AP, AFP, B, LDB, X, LDX, FERR, BERR,
1                   WORK, IWORK, INFO)
 INTEGER          N, NRHS, LDB, LDX, IWORK(*), INFO
 real             AP(*), AFP(*), B(LDB,*), X(LDX,*), FERR(*), BERR(*),
1                 WORK(*)
 CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine returns the backward errors and estimated bounds on the forward errors for the solution of a real symmetric positive-definite system of linear equations with multiple right-hand sides $AX = B$, using packed storage. The routine handles each right-hand side vector (stored as a column of the matrix B) independently, so we describe the function of the routine in terms of a single right-hand side b and solution x.

Given a computed solution x, the routine computes the *component-wise backward error* β. This is the size of the smallest relative perturbation in each element of A and b such that x is the exact solution of a perturbed system

$$(A+\delta A)x = b + \delta b$$

$$|\delta a_{ij}| \leq \beta |a_{ij}| \text{ and } |\delta b_i| \leq \beta |b_i|.$$

Then the routine estimates a bound for the *component-wise forward error* in the computed solution, defined by:

$$\max_i |x_i - \hat{x}_i| / \max_i |x_i|$$

where $\hat{x}$ is the true solution.

For details of the method, see the Chapter Introduction.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether the upper or lower triangular part of A is stored and how A has been factorized, as follows:

if UPLO = 'U', then the upper triangular part of A is stored and A is factorized as $U^T U$, where U is upper triangular;

if UPLO = 'L', then the lower triangular part of A is stored and A is factorized as LL^T, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS ≥ 0.

4: AP(*) – ***real*** array. *Input*

Note: the dimension of the array AP must be at least max(1,N*(N+1)/2).

On entry: the n by n original symmetric positive-definite matrix A as supplied to F07GDF (SPPTRF/DPPTRF).

5: AFP(*) – ***real*** array. *Input*

Note: the dimension of the array AFP must be at least max(1,N*(N+1)/2).

On entry: the Cholesky factor of A stored in packed form, as returned by F07GDF (SPPTRF/DPPTRF).

6: B(LDB,*) – ***real*** array. *Input*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

7: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07GHF (SPPRFS/DPPRFS) is called.

Constraint: LDB ≥ max(1,N).

8: X(LDX,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array X must be at least max(1,NRHS).

On entry: the n by r solution matrix X, as returned by F07GEF (SPPTRS/DPPTRS).

On exit: the improved solution matrix X.

9: LDX – INTEGER. *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which F07GHF (SPPRFS/DPPRFS) is called.

Constraint: LDX ≥ max(1,N).

10: FERR(*) – ***real*** array. *Output*

Note: the dimension of the array FERR must be at least max(1,NRHS).

On exit: FERR(j) contains an estimated error bound for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

11: BERR(*) – ***real*** array. *Output*

Note: the dimension of the array BERR must be at least max(1,NRHS).

On exit: BERR(j) contains the component-wise backward error bound β for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

12: WORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,3*N).

13: IWORK(*) – INTEGER array. *Workspace*

Note: the dimension of the array IWORK must be at least max(1,N).

14: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The bounds returned in FERR are not rigorous, because they are estimated, not computed exactly; but in practice they almost always overestimate the actual error.

8. Further Comments

For each right-hand side, computation of the backward error involves a minimum of $4n^2$ floating-point operations. Each step of iterative refinement involves an additional $6n^2$ operations. At most 5 steps of iterative refinement are performed, but usually only 1 or 2 steps are required.

Estimating the forward error involves solving a number of systems of linear equations of the form $Ax = b$; the number is usually 4 or 5 and never more than 11. Each solution involves approximately $2n^2$ operations.

The complex analogue of this routine is F07GVF (CPPRFS/ZPPRFS).

9. Example

To solve the system of equations $AX = B$ using iterative refinement and to compute the forward and backward error bounds, where

$$A = \begin{pmatrix} 4.16 & -3.12 & 0.56 & -0.10 \\ -3.12 & 5.03 & -0.83 & 1.18 \\ 0.56 & -0.83 & 0.76 & 0.34 \\ -0.10 & 1.18 & 0.34 & 1.18 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 8.70 & 8.30 \\ -13.35 & 2.13 \\ 1.89 & 1.61 \\ -4.14 & 5.00 \end{pmatrix}.$$

Here A is symmetric positive-definite, stored in packed form, and must first be factorized by F07GDF (SPPTRF/DPPTRF).

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07GHF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, NRHMAX, LDB, LDX
      PARAMETER        (NMAX=8,NRHMAX=NMAX,LDB=NMAX,LDX=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
      CHARACTER        UPLO
```

```
*     .. Local Arrays ..
      real               AFP(NMAX*(NMAX+1)/2), AP(NMAX*(NMAX+1)/2),
     +                   B(LDB,NRHMAX), BERR(NRHMAX), FERR(NRHMAX),
     +                   WORK(3*NMAX), X(LDX,NMAX)
      INTEGER            IWORK(NMAX)
*     .. External Subroutines ..
      EXTERNAL           F06QFF, spprfs, spptrf, spptrs, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07GHF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file, and copy A to AFP and B to X
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
         DO 20 I = 1, N*(N+1)/2
            AFP(I) = AP(I)
   20    CONTINUE
*
         CALL F06QFF('General',N,NRHS,B,LDB,X,LDX)
*
*        Factorize A in the array AFP
*
         CALL spptrf(UPLO,N,AFP,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute solution in the array X
*
            CALL spptrs(UPLO,N,NRHS,AFP,X,LDX,INFO)
*
*           Improve solution, and compute backward errors and
*           estimated bounds on the forward errors
*
            CALL spprfs(UPLO,N,NRHS,AP,AFP,B,LDB,X,LDX,FERR,BERR,WORK,
     +                  IWORK,INFO)
*
*           Print solution
*
            IFAIL = 0
*
            CALL X04CAF('General',' ',N,NRHS,X,LDX,'Solution(s)',IFAIL)
*
            WRITE (NOUT,*)
            WRITE (NOUT,*) 'Backward errors (machine-dependent)'
            WRITE (NOUT,99999) (BERR(J),J=1,NRHS)
            WRITE (NOUT,*)
     +        'Estimated forward error bounds (machine-dependent)'
            WRITE (NOUT,99999) (FERR(J),J=1,NRHS)
         ELSE
            WRITE (NOUT,*) 'A is not positive-definite'
         END IF
      END IF
      STOP
*
99999 FORMAT ((3X,1P,7e11.1))
      END
```

9.2. Program Data

```
F07GHF Example Program Data
  4  2                          :Values of N and NRHS
  'L'                           :Value of UPLO
  4.16
 -3.12   5.03
  0.56  -0.83   0.76
 -0.10   1.18   0.34   1.18    :End of matrix A
  8.70   8.30
-13.35   2.13
  1.89   1.61
 -4.14   5.00                   :End of matrix B
```

9.3. Program Results

```
F07GHF Example Program Results

Solution(s)
                1          2
 1        1.0000     4.0000
 2       -1.0000     3.0000
 3        2.0000     2.0000
 4       -3.0000     1.0000

Backward errors (machine-dependent)
      4.0E-17    3.7E-17
Estimated forward error bounds (machine-dependent)
      2.3E-14    2.2E-14
```

F07GJF (SPPTRI/DPPTRI) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07GJF (SPPTRI/DPPTRI) computes the inverse of a real symmetric positive-definite matrix A, where A has been factorized by F07GDF (SPPTRF/DPPTRF), using packed storage.

2. Specification

```
SUBROUTINE F07GJF (UPLO, N, AP, INFO)
ENTRY      spptri  (UPLO, N, AP, INFO)
INTEGER          N, INFO
real             AP(*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

To compute the inverse of a real symmetric positive-definite matrix A, this routine must be preceded by a call to F07GDF (SPPTRF/DPPTRF), which computes the Cholesky factorization of A using packed storage.

If UPLO = 'U', $A = U^TU$ and A^{-1} is computed by first inverting U and then forming $(U^{-1})(U^{-1})^T$.

If UPLO = 'L', $A = LL^T$ and A^{-1} is computed by first inverting L and then forming $(L^{-1})^T(L^{-1})$.

4. References

[1] DU CROZ, J.J. and HIGHAM, N.J.
Stability of Methods for Matrix Inversion.
LAPACK Working Note No. 27, University of Tennessee, Knoxville, 1990.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A has been factorized as U^TU or LL^T as follows:

if UPLO = 'U', then $A = U^TU$, where U is upper triangular;

if UPLO = 'L', then $A = LL^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: AP(*) – ***real*** array. *Input/Output*

Note: the dimension of the array AP must be at least max(1,N*(N+1)/2).

On entry: the upper triangular matrix U stored in packed form if UPLO = 'U' or the lower triangular matrix L stored in packed form if UPLO = 'L', as returned by F07GDF (SPPTRF/DPPTRF).

On exit: U is overwritten by the upper triangle of A^{-1} if UPLO = 'U'; L is overwritten by the lower triangle of A^{-1} if UPLO = 'L'. More precisely, the (i,j)th element of A^{-1} is stored in AP(i+j(j−1)/2) for $i \leq j$ if UPLO = 'U', and in AP(i+(2n−j)(j−1)/2) for $i \geq j$ if UPLO = 'L'.

4: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, the ith diagonal element of the Cholesky factor is zero; the Cholesky factor is singular and the inverse of A cannot be computed.

7. Accuracy

The computed inverse X satisfies

$$\|XA-I\|_2 \leq c(n)\varepsilon\kappa_2(A) \text{ and } \|AX-I\|_2 \leq c(n)\varepsilon\kappa_2(A),$$

where $c(n)$ is a modest function of n, ε is the ***machine precision*** and $\kappa_2(A)$ is the condition number of A defined by

$$\kappa_2(A) = \|A\|_2\|A^{-1}\|_2.$$

8. Further Comments

The total number of floating-point operations is approximately $\frac{2}{3}n^3$.

The complex analogue of this routine is F07GWF (CPPTRI/ZPPTRI).

9. Example

To compute the inverse of the matrix A, where

$$A = \begin{pmatrix} 4.16 & -3.12 & 0.56 & -0.10 \\ -3.12 & 5.03 & -0.83 & 1.18 \\ 0.56 & -0.83 & 0.76 & 0.34 \\ -0.10 & 1.18 & 0.34 & 1.18 \end{pmatrix}.$$

Here A is symmetric positive-definite, stored in packed form, and must first be factorized by F07GDF (SPPTRF/DPPTRF).

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07GJF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=8)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             AP(NMAX*(NMAX+1)/2)
*     .. External Subroutines ..
      EXTERNAL         spptrf, spptri, X04CCF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07GJF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
```

```
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
*
*        Factorize A
*
         CALL spptrf(UPLO,N,AP,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute inverse of A
*
            CALL spptri(UPLO,N,AP,INFO)
*
*           Print inverse
*
            IFAIL = 0
*
            CALL X04CCF(UPLO,'Nonunit',N,AP,'Inverse',IFAIL)
*
         ELSE
            WRITE (NOUT,*) 'A is not positive-definite'
         END IF
      END IF
      STOP
*
      END
```

9.2. Program Data

```
F07GJF Example Program Data
  4                              :Value of N
  'L'                            :Value of UPLO
  4.16
 -3.12   5.03
  0.56  -0.83   0.76
 -0.10   1.18   0.34   1.18     :End of matrix A
```

9.3. Program Results

```
 F07GJF Example Program Results

 Inverse
              1          2          3          4
  1      0.6995
  2      0.7769     1.4239
  3      0.7508     1.8255     4.0688
  4     -0.9340    -1.8841    -2.9342     3.4978
```

F07GRF (CPPTRF/ZPPTRF) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07GRF (CPPTRF/ZPPTRF) computes the Cholesky factorization of a complex Hermitian positive-definite matrix, using packed storage.

2. Specification

```
SUBROUTINE F07GRF (UPLO, N, AP, INFO)
ENTRY      cpptrf  (UPLO, N, AP, INFO)
INTEGER          N, INFO
complex          AP(*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine forms the Cholesky factorization of a complex Hermitian positive-definite matrix A either as $A = U^H U$ if UPLO = 'U', or $A = LL^H$ if UPLO = 'L', where U is an upper triangular matrix and L is lower triangular, using packed storage.

4. References

[1] DEMMEL, J.W.
On Floating-point Errors in Cholesky.
LAPACK Working Note No. 14, University of Tennessee, Knoxville, 1989.

[2] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §4.2.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether the upper or lower triangular part of A is stored and how A is factorized, as follows:

if UPLO = 'U', then the upper triangular part of A is stored and A is factorized as $U^H U$, where U is upper triangular;

if UPLO = 'L', then the lower triangular part of A is stored and A is factorized as LL^H, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: AP(*) – ***complex*** array. *Input/Output*

Note: the dimension of the array AP must be at least max(1,N*(N+1)/2).

On entry: the n by n Hermitian positive-definite matrix A, packed by columns. More precisely, if UPLO = 'U', the upper triangle of A must be stored with element a_{ij} in AP(i+j(j−1)/2) for $i \leq j$; if UPLO = 'L', the lower triangle of A must be stored with element a_{ij} in AP(i+(2n−j)(j−1)/2) for $i \geq j$.

On exit: the upper or lower triangle of A is overwritten by the Cholesky factor U or L as specified by UPLO, using the same packed storage format as described above.

4: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, the leading minor of order i is not positive-definite and the factorization could not be completed. Hence A itself is not positive-definite. This may indicate an error in forming the matrix A. To factorize a Hermitian matrix which is not positive-definite, call F07PRF (CHPTRF/ZHPTRF) instead.

7. Accuracy

If UPLO = 'U', the computed factor U is the exact factor of a perturbed matrix $A + E$, where

$$|E| \leq c(n)\varepsilon|U^H||U|,$$

$c(n)$ is a modest linear function of n, and ε is the ***machine precision***. If UPLO = 'L', a similar statement holds for the computed factor L. It follows that $|e_{ij}| \leq c(n)\varepsilon\sqrt{a_{ii}a_{jj}}$.

8. Further Comments

The total number of real floating-point operations is approximately $\frac{4}{3}n^3$.

A call to this routine may be followed by calls to the routines:

F07GSF (CPPTRS/ZPPTRS) to solve $AX = B$;
F07GUF (CPPCON/ZPPCON) to estimate the condition number of A;
F07GWF (CPPTRI/ZPPTRI) to compute the inverse of A.

The real analogue of this routine is F07GDF (SPPTRF/DPPTRF).

9. Example

To compute the Cholesky factorization of the matrix A, where

$$A = \begin{pmatrix} 3.23 + 0.00i & 1.51 - 1.92i & 1.90 + 0.84i & 0.42 + 2.50i \\ 1.51 + 1.92i & 3.58 + 0.00i & -0.23 + 1.11i & -1.18 + 1.37i \\ 1.90 - 0.84i & -0.23 - 1.11i & 4.09 + 0.00i & 2.33 - 0.14i \\ 0.42 - 2.50i & -1.18 - 1.37i & 2.33 + 0.14i & 4.29 + 0.00i \end{pmatrix},$$

using packed storage.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07GRF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=8)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          AP(NMAX*(NMAX+1)/2)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         cpptrf, X04DDF
```

```
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07GRF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
*
*        Factorize A
*
         CALL cpptrf(UPLO,N,AP,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Print factor
*
            IFAIL = 0
            CALL X04DDF(UPLO,'Nonunit',N,AP,'Bracketed','F7.4','Factor',
     +                  'Integer',RLABS,'Integer',CLABS,80,0,IFAIL)
         ELSE
            WRITE (NOUT,*) 'A is not positive-definite'
         END IF
      END IF
      STOP
*
      END
```

9.2. Program Data

```
F07GRF Example Program Data
  4                                                          :Value of N
  'L'                                                        :Value of UPLO
 (3.23, 0.00)
 (1.51, 1.92) ( 3.58, 0.00)
 (1.90,-0.84) (-0.23,-1.11) ( 4.09, 0.00)
 (0.42,-2.50) (-1.18,-1.37) ( 2.33, 0.14) ( 4.29, 0.00)  :End of matrix A
```

9.3. Program Results

```
 F07GRF Example Program Results

 Factor
                      1                  2                  3                  4
  1  ( 1.7972, 0.0000)
  2  ( 0.8402, 1.0683) ( 1.3164, 0.0000)
  3  ( 1.0572,-0.4674) (-0.4702, 0.3131) ( 1.5604, 0.0000)
  4  ( 0.2337,-1.3910) ( 0.0834, 0.0368) ( 0.9360, 0.9900) ( 0.6603, 0.0000)
```

F07GSF (CPPTRS/ZPPTRS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07GSF (CPPTRS/ZPPTRS) solves a complex Hermitian positive-definite system of linear equations with multiple right-hand sides, $AX = B$, where A has been factorized by F07GRF (CPPTRF/ZPPTRF), using packed storage.

2. Specification

```
SUBROUTINE F07GSF (UPLO, N, NRHS, AP, B, LDB, INFO)
ENTRY      cpptrs  (UPLO, N, NRHS, AP, B, LDB, INFO)
INTEGER          N, NRHS, LDB, INFO
complex          AP(*), B(LDB,*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

To solve a complex Hermitian positive-definite system of linear equations $AX = B$, this routine must be preceded by a call to F07GRF (CPPTRF/ZPPTRF) which computes the Cholesky factorization of A using packed storage. The solution X is computed by forward and backward substitution.

If UPLO = 'U', $A = U^H U$, where U is upper triangular; the solution X is computed by solving $U^H Y = B$ and then $UX = Y$.

If UPLO = 'L', $A = LL^H$, where L is lower triangular; the solution X is computed by solving $LY = B$ and then $L^H X = Y$.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §4.2.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A has been factorized as $U^H U$ or LL^H as follows:

if UPLO = 'U', then $A = U^H U$, where U is upper triangular;

if UPLO = 'L', then $A = LL^H$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS $\geq$ 0.

4: AP(*) – ***complex*** array. *Input*

Note: the dimension of the array AP must be at least max(1,N*(N+1)/2).

On entry: the Cholesky factor of A stored in packed form, as returned by F07GRF (CPPTRF/ZPPTRF).

5: B(LDB,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

On exit: the n by r solution matrix X.

6: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07GSF (CPPTRS/ZPPTRS) is called.

Constraint: LDB ≥ max(1,N).

7: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

For each right-hand side vector b, the computed solution x is the exact solution of a perturbed system of equations $(A+E)x = b$, where

$|E| \leq c(n)\varepsilon|U^H||U|$, if UPLO = 'U',

$|E| \leq c(n)\varepsilon|L||L^H|$, if UPLO = 'L',

$c(n)$ is a modest linear function of n, and ε is the ***machine precision***.

If $\hat{x}$ is the true solution, then the computed solution x satisfies a forward error bound of the form

$$\frac{\|x-\hat{x}\|_\infty}{\|x\|_\infty} \leq c(n)\text{cond}(A,x)\varepsilon$$

where $\text{cond}(A,x) = \||A^{-1}||A||x|\|_\infty / \|x\|_\infty \leq \text{cond}(A) = \||A^{-1}||A|\|_\infty \leq \kappa_\infty(A)$. Note that $\text{cond}(A,x)$ can be much smaller than $\text{cond}(A)$.

Forward and backward error bounds can be computed by calling F07GVF (CPPRFS/ZPPRFS), and an estimate for $\kappa_\infty(A)$ $(= \kappa_1(A))$ can be obtained by calling F07GUF (CPPCON/ZPPCON).

8. Further Comments

The total number of real floating-point operations is approximately $8n^2r$.

This routine may be followed by a call to F07GVF (CPPRFS/ZPPRFS) to refine the solution and return an error estimate.

The real analogue of this routine is F07GEF (SPPTRS/DPPTRS).

9. Example

To solve the system of equations $AX = B$, where

$$A = \begin{pmatrix} 3.23 + 0.00i & 1.51 - 1.92i & 1.90 + 0.84i & 0.42 + 2.50i \\ 1.51 + 1.92i & 3.58 + 0.00i & -0.23 + 1.11i & -1.18 + 1.37i \\ 1.90 - 0.84i & -0.23 - 1.11i & 4.09 + 0.00i & 2.33 - 0.14i \\ 0.42 - 2.50i & -1.18 - 1.37i & 2.33 + 0.14i & 4.29 + 0.00i \end{pmatrix} \quad \text{and}$$

$$B = \begin{pmatrix} 3.93 - 6.14i & 1.48 + 6.58i \\ 6.17 + 9.42i & 4.65 - 4.75i \\ -7.17 - 21.83i & -4.91 + 2.29i \\ 1.99 - 14.38i & 7.64 - 10.79i \end{pmatrix}.$$

Here A is Hermitian positive-definite, stored in packed form, and must first be factorized by F07GRF (CPPTRF/ZPPTRF).

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07GSF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, NRHMAX, LDB
      PARAMETER        (NMAX=8,NRHMAX=NMAX,LDB=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          AP(NMAX*(NMAX+1)/2), B(LDB,NRHMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         cpptrf, cpptrs, X04DBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07GSF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
*
*        Factorize A
*
         CALL cpptrf(UPLO,N,AP,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute solution
*
            CALL cpptrs(UPLO,N,NRHS,AP,B,LDB,INFO)
*
```

```
*           Print solution
*
            IFAIL = 0
            CALL X04DBF('General',' ',N,NRHS,B,LDB,'Bracketed','F7.4',
     +                  'Solution(s)','Integer',RLABS,'Integer',CLABS,
     +                  80,0,IFAIL)
         ELSE
            WRITE (NOUT,*) 'A is not positive-definite'
         END IF
       END IF
       STOP
*
       END
```

9.2. Program Data

```
F07GSF Example Program Data
  4  2                                                    :Values of N and NRHS
  'L'                                                         :Value of UPLO
 (3.23, 0.00)
 (1.51, 1.92) ( 3.58, 0.00)
 (1.90,-0.84) (-0.23,-1.11) ( 4.09, 0.00)
 (0.42,-2.50) (-1.18,-1.37) ( 2.33, 0.14) ( 4.29, 0.00) :End of matrix A
 ( 3.93, -6.14) ( 1.48,  6.58)
 ( 6.17,  9.42) ( 4.65, -4.75)
 (-7.17,-21.83) (-4.91,  2.29)
 ( 1.99,-14.38) ( 7.64,-10.79)                               :End of matrix B
```

9.3. Program Results

```
 F07GSF Example Program Results

 Solution(s)
                      1                   2
  1   ( 1.0000,-1.0000) (-1.0000, 2.0000)
  2   ( 0.0000, 3.0000) ( 3.0000,-4.0000)
  3   (-4.0000,-5.0000) (-2.0000, 3.0000)
  4   ( 2.0000, 1.0000) ( 4.0000,-5.0000)
```

F07GUF (CPPCON/ZPPCON) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07GUF (CPPCON/ZPPCON) estimates the condition number of a complex Hermitian positive-definite matrix A, where A has been factorized by F07GRF (CPPTRF/ZPPTRF), using packed storage.

2. Specification

```
SUBROUTINE F07GUF (UPLO, N, AP, ANORM, RCOND, WORK, RWORK, INFO)
ENTRY      cppcon (UPLO, N, AP, ANORM, RCOND, WORK, RWORK, INFO)
INTEGER          N, INFO
real             ANORM, RCOND, RWORK(*)
complex          AP(*), WORK(*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine estimates the condition number (in the 1-norm) of a complex Hermitian positive-definite matrix A:

$$\kappa_1(A) = \|A\|_1 \|A^{-1}\|_1.$$

Since A is Hermitian, $\kappa_1(A) = \kappa_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty$.

Because $\kappa_1(A)$ is infinite if A is singular, the routine actually returns an estimate of the **reciprocal** of $\kappa_1(A)$.

The routine should be preceded by a call to F06UDF to compute $\|A\|_1$ and a call to F07GRF (CPPTRF/ZPPTRF) to compute the Cholesky factorization of A. The routine then uses Higham's implementation of Hager's method [1] to estimate $\|A^{-1}\|_1$.

4. References

[1] HIGHAM, N.J.
FORTRAN Codes for Estimating the One-norm of a Real or Complex Matrix, with Applications to Condition Estimation.
ACM Trans. Math. Softw., 14, pp. 381-396, 1988.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A has been factorized as $U^H U$ or LL^H as follows:

if UPLO = 'U', then $A = U^H U$, where U is upper triangular;

if UPLO = 'L', then $A = LL^H$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: AP(*) – ***complex*** array. *Input*

Note: the dimension of the array AP must be at least max(1,N*(N+1)/2).

On entry: the Cholesky factor of A stored in packed form, as returned by F07GRF (CPPTRF/ZPPTRF).

4: ANORM – ***real.*** *Input*

On entry: the 1-norm of the **original** matrix A, which may be computed by calling F06UDF. ANORM must be computed either **before** calling F07GRF (CPPTRF/ZPPTRF) or else from a copy of the original matrix A.

Constraint: ANORM $\geq$ 0.0.

5: RCOND – ***real.*** *Output*

On exit: an estimate of the reciprocal of the condition number of A. RCOND is set to zero if exact singularity is detected or the estimate underflows. If RCOND is less than ***machine precision***, then A is singular to working precision.

6: WORK(*) – ***complex*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,2*N).

7: RWORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array RWORK must be at least max(1,N).

8: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed estimate RCOND is never less than the true value ρ, and in practice is nearly always less than 10ρ, although examples can be constructed where RCOND is much larger.

8. Further Comments

A call to this routine involves solving a number of systems of linear equations of the form $Ax = b$; the number is usually 5 and never more than 11. Each solution involves approximately $8n^2$ real floating-point operations but takes considerably longer than a call to F07GSF (CPPTRS/ZPPTRS) with 1 right-hand side, because extra care is taken to avoid overflow when A is approximately singular.

The real analogue of this routine is F07GGF (SPPCON/DPPCON).

9. Example

To estimate the condition number in the 1-norm (or infinity-norm) of the matrix A, where

$$A = \begin{pmatrix} 3.23 + 0.00i & 1.51 - 1.92i & 1.90 + 0.84i & 0.42 + 2.50i \\ 1.51 + 1.92i & 3.58 + 0.00i & -0.23 + 1.11i & -1.18 + 1.37i \\ 1.90 - 0.84i & -0.23 - 1.11i & 4.09 + 0.00i & 2.33 - 0.14i \\ 0.42 - 2.50i & -1.18 - 1.37i & 2.33 + 0.14i & 4.29 + 0.00i \end{pmatrix}.$$

Here A is Hermitian positive-definite, stored in packed form, and must first be factorized by F07GRF (CPPTRF/ZPPTRF). The true condition number in the 1-norm is 201.92.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07GUF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=8)
*     .. Local Scalars ..
      real             ANORM, RCOND
      INTEGER          I, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          AP(NMAX*(NMAX+1)/2), WORK(2*NMAX)
      real             RWORK(NMAX)
*     .. External Functions ..
      real             F06UDF, X02AJF
      EXTERNAL         F06UDF, X02AJF
*     .. External Subroutines ..
      EXTERNAL         cppcon, cpptrf
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07GUF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
*
*        Compute norm of A
*
         ANORM = F06UDF('1-norm',UPLO,N,AP,RWORK)
*
*        Factorize A
*
         CALL cpptrf(UPLO,N,AP,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Estimate condition number
*
            CALL cppcon(UPLO,N,AP,ANORM,RCOND,WORK,RWORK,INFO)
*
            IF (RCOND.GE.X02AJF()) THEN
               WRITE (NOUT,99999) 'Estimate of condition number =',
     +           1.0e0/RCOND
            ELSE
               WRITE (NOUT,*) 'A is singular to working precision'
            END IF
         ELSE
            WRITE (NOUT,*) 'A is not positive-definite'
         END IF
      END IF
      STOP
*
99999 FORMAT (1X,A,1P,e10.2)
      END
```

9.2. Program Data

```
F07GUF Example Program Data
  4                                                          :Value of N
  'L'                                                        :Value of UPLO
 (3.23, 0.00)
 (1.51, 1.92) ( 3.58, 0.00)
 (1.90,-0.84) (-0.23,-1.11) ( 4.09, 0.00)
 (0.42,-2.50) (-1.18,-1.37) ( 2.33, 0.14) ( 4.29, 0.00)  :End of matrix A
```

9.3. Program Results

```
F07GUF Example Program Results

Estimate of condition number =  1.51E+02
```

F07GVF (CPPRFS/ZPPRFS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07GVF (CPPRFS/ZPPRFS) returns error bounds for the solution of a complex Hermitian positive-definite system of linear equations with multiple right-hand sides, $AX = B$, using packed storage. It improves the solution by iterative refinement, in order to reduce the backward error as much as possible.

2. Specification

```
SUBROUTINE F07GVF (UPLO, N, NRHS, AP, AFP, B, LDB, X, LDX, FERR, BERR,
1                  WORK, RWORK, INFO)
ENTRY       cpprfs  (UPLO, N, NRHS, AP, AFP, B, LDB, X, LDX, FERR, BERR,
1                  WORK, RWORK, INFO)
INTEGER          N, NRHS, LDB, LDX, INFO
real             FERR(*), BERR(*), RWORK(*)
complex          AP(*), AFP(*), B(LDB,*), X(LDX,*), WORK(*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine returns the backward errors and estimated bounds on the forward errors for the solution of a complex Hermitian positive-definite system of linear equations with multiple right-hand sides $AX = B$, using packed storage. The routine handles each right-hand side vector (stored as a column of the matrix B) independently, so we describe the function of the routine in terms of a single right-hand side b and solution x.

Given a computed solution x, the routine computes the *component-wise backward error* β. This is the size of the smallest relative perturbation in each element of A and b such that x is the exact solution of a perturbed system

$$(A+\delta A)x = b + \delta b$$

$$|\delta a_{ij}| \leq \beta |a_{ij}| \quad \text{and} \quad |\delta b_i| \leq \beta |b_i|.$$

Then the routine estimates a bound for the *component-wise forward error* in the computed solution, defined by:

$$\max_i |x_i - \hat{x}_i| / \max_i |x_i|$$

where $\hat{x}$ is the true solution.

For details of the method, see the Chapter Introduction.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether the upper or lower triangular part of A is stored and how A has been factorized, as follows:

if UPLO = 'U', then the upper triangular part of A is stored and A is factorized as $U^H U$, where U is upper triangular;

if UPLO = 'L', then the lower triangular part of A is stored and A is factorized as LL^H, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS ≥ 0.

4: AP(*) – ***complex*** array. *Input*

Note: the dimension of the array AP must be at least max(1,N*(N+1)/2).

On entry: the n by n original Hermitian positive-definite matrix A as supplied to F07GRF (CPPTRF/ZPPTRF).

5: AFP(*) – ***complex*** array. *Input*

Note: the dimension of the array AFP must be at least max(1,N*(N+1)/2).

On entry: the Cholesky factor of A stored in packed form, as returned by F07GRF (CPPTRF/ZPPTRF).

6: B(LDB,*) – ***complex*** array. *Input*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

7: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07GVF (CPPRFS/ZPPRFS) is called.

Constraint: LDB ≥ max(1,N).

8: X(LDX,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array X must be at least max(1,NRHS).

On entry: the n by r solution matrix X, as returned by F07GSF (CPPTRS/ZPPTRS).

On exit: the improved solution matrix X.

9: LDX – INTEGER. *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which F07GVF (CPPRFS/ZPPRFS) is called.

Constraint: LDX ≥ max(1,N).

10: FERR(*) – ***real*** array. *Output*

Note: the dimension of the array FERR must be at least max(1,NRHS).

On exit: FERR(j) contains an estimated error bound for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

11: BERR(*) – ***real*** array. *Output*

Note: the dimension of the array BERR must be at least max(1,NRHS).

On exit: BERR(j) contains the component-wise backward error bound β for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

12: WORK(*) – ***complex*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,2*N).

13: RWORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array RWORK must be at least max(1,N).

14: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The bounds returned in FERR are not rigorous, because they are estimated, not computed exactly; but in practice they almost always overestimate the actual error.

8. Further Comments

For each right-hand side, computation of the backward error involves a minimum of $16n^2$ real floating-point operations. Each step of iterative refinement involves an additional $24n^2$ real operations. At most 5 steps of iterative refinement are performed, but usually only 1 or 2 steps are required.

Estimating the forward error involves solving a number of systems of linear equations of the form $Ax = b$; the number is usually 5 and never more than 11. Each solution involves approximately $8n^2$ real operations.

The real analogue of this routine is F07GHF (SPPRFS/DPPRFS).

9. Example

To solve the system of equations $AX = B$ using iterative refinement and to compute the forward and backward error bounds, where

$$A = \begin{pmatrix} 3.23 + 0.00i & 1.51 - 1.92i & 1.90 + 0.84i & 0.42 + 2.50i \\ 1.51 + 1.92i & 3.58 + 0.00i & -0.23 + 1.11i & -1.18 + 1.37i \\ 1.90 - 0.84i & -0.23 - 1.11i & 4.09 + 0.00i & 2.33 - 0.14i \\ 0.42 - 2.50i & -1.18 - 1.37i & 2.33 + 0.14i & 4.29 + 0.00i \end{pmatrix} \quad \text{and}$$

$$B = \begin{pmatrix} 3.93 - 6.14i & 1.48 + 6.58i \\ 6.17 + 9.42i & 4.65 - 4.75i \\ -7.17 - 21.83i & -4.91 + 2.29i \\ 1.99 - 14.38i & 7.64 - 10.79i \end{pmatrix}.$$

Here A is Hermitian positive-definite, stored in packed form, and must first be factorized by F07GRF (CPPTRF/ZPPTRF).

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07GVF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, NRHMAX, LDB, LDX
      PARAMETER        (NMAX=8,NRHMAX=NMAX,LDB=NMAX,LDX=NMAX)
```

```
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          AFP(NMAX*(NMAX+1)/2), AP(NMAX*(NMAX+1)/2),
     +                 B(LDB,NRHMAX), WORK(2*NMAX), X(LDX,NMAX)
      real             BERR(NRHMAX), FERR(NRHMAX), RWORK(NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         cpprfs, cpptrf, cpptrs, F06TFF, X04DBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07GVF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file, and copy A to AFP and B to X
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
         DO 20 I = 1, N*(N+1)/2
            AFP(I) = AP(I)
   20    CONTINUE
         CALL F06TFF('General',N,NRHS,B,LDB,X,LDX)
*
*        Factorize A in the array AFP
*
         CALL cpptrf(UPLO,N,AFP,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute solution in the array X
*
            CALL cpptrs(UPLO,N,NRHS,AFP,X,LDX,INFO)
*
*           Improve solution, and compute backward errors and
*           estimated bounds on the forward errors
*
            CALL cpprfs(UPLO,N,NRHS,AP,AFP,B,LDB,X,LDX,FERR,BERR,WORK,
     +                  RWORK,INFO)
*
*           Print solution
*
            IFAIL = 0
            CALL X04DBF('General',' ',N,NRHS,X,LDX,'Bracketed','F7.4',
     +                  'Solution(s)','Integer',RLABS,'Integer',CLABS,
     +                  80,0,IFAIL)
            WRITE (NOUT,*)
            WRITE (NOUT,*) 'Backward errors (machine-dependent)'
            WRITE (NOUT,99999) (BERR(J),J=1,NRHS)
            WRITE (NOUT,*)
     +        'Estimated forward error bounds (machine-dependent)'
            WRITE (NOUT,99999) (FERR(J),J=1,NRHS)
         ELSE
            WRITE (NOUT,*) 'A is not positive-definite'
         END IF
      END IF
      STOP
*
99999 FORMAT ((5X,1P,4(e11.1,7X)))
      END
```

9.2. Program Data

```
F07GVF Example Program Data
  4  2                                                          :Values of N and NRHS
  'L'                                                           :Value of UPLO
 (3.23, 0.00)
 (1.51, 1.92) ( 3.58, 0.00)
 (1.90,-0.84) (-0.23,-1.11) ( 4.09, 0.00)
 (0.42,-2.50) (-1.18,-1.37) ( 2.33, 0.14) ( 4.29, 0.00)         :End of matrix A
 ( 3.93, -6.14) ( 1.48,  6.58)
 ( 6.17,  9.42) ( 4.65, -4.75)
 (-7.17,-21.83) (-4.91,  2.29)
 ( 1.99,-14.38) ( 7.64,-10.79)                                  :End of matrix B
```

9.3. Program Results

```
F07GVF Example Program Results

Solution(s)
                     1                   2
 1   ( 1.0000,-1.0000) (-1.0000, 2.0000)
 2   ( 0.0000, 3.0000) ( 3.0000,-4.0000)
 3   (-4.0000,-5.0000) (-2.0000, 3.0000)
 4   ( 2.0000, 1.0000) ( 4.0000,-5.0000)

Backward errors (machine-dependent)
         4.7E-17           5.3E-17
Estimated forward error bounds (machine-dependent)
         5.9E-14           7.3E-14
```

4: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, the ith diagonal element of the Cholesky factor is zero; the Cholesky factor is singular and the inverse of A cannot be computed.

7. Accuracy

The computed inverse X satisfies

$$\|XA-I\|_2 \leq c(n)\varepsilon\kappa_2(A) \quad \text{and} \quad \|AX-I\|_2 \leq c(n)\varepsilon\kappa_2(A),$$

where $c(n)$ is a modest function of n, ε is the ***machine precision*** and $\kappa_2(A)$ is the condition number of A defined by

$$\kappa_2(A) = \|A\|_2\|A^{-1}\|_2.$$

8. Further Comments

The total number of real floating-point operations is approximately $\frac{8}{3}n^3$.

The real analogue of this routine is F07GJF (SPPTRI/DPPTRI).

9. Example

To compute the inverse of the matrix A, where

$$A = \begin{pmatrix} 3.23 + 0.00i & 1.51 - 1.92i & 1.90 + 0.84i & 0.42 + 2.50i \\ 1.51 + 1.92i & 3.58 + 0.00i & -0.23 + 1.11i & -1.18 + 1.37i \\ 1.90 - 0.84i & -0.23 - 1.11i & 4.09 + 0.00i & 2.33 - 0.14i \\ 0.42 - 2.50i & -1.18 - 1.37i & 2.33 + 0.14i & 4.29 + 0.00i \end{pmatrix}.$$

Here A is Hermitian positive-definite, stored in packed form, and must first be factorized by F07GRF (CPPTRF/ZPPTRF).

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07GWF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=8)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          AP(NMAX*(NMAX+1)/2)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         cpptrf, cpptri, X04DDF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07GWF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
```

```
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
*
*        Factorize A
*
         CALL cpptrf(UPLO,N,AP,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute inverse of A
*
            CALL cpptri(UPLO,N,AP,INFO)
*
*           Print inverse
*
            IFAIL = 0
            CALL X04DDF(UPLO,'Nonunit',N,AP,'Bracketed','F7.4',
     +                  'Inverse','Integer',RLABS,'Integer',CLABS,80,0,
     +                  IFAIL)
         ELSE
            WRITE (NOUT,*) 'A is not positive-definite'
         END IF
      END IF
      STOP
*
      END
```

9.2. Program Data

```
F07GWF Example Program Data
  4                                                          :Value of N
  'L'                                                        :Value of UPLO
 (3.23, 0.00)
 (1.51, 1.92) ( 3.58, 0.00)
 (1.90,-0.84) (-0.23,-1.11) ( 4.09, 0.00)
 (0.42,-2.50) (-1.18,-1.37) ( 2.33, 0.14) ( 4.29, 0.00)  :End of matrix A
```

9.3. Program Results

```
 F07GWF Example Program Results

 Inverse
                      1                  2                  3                  4
  1  ( 5.4691, 0.0000)
  2  (-1.2624,-1.5491) ( 1.1024, 0.0000)
  3  (-2.9746,-0.9616) ( 0.8989,-0.5672) ( 2.1589, 0.0000)
  4  ( 1.1962, 2.9772) (-0.9826,-0.2566) (-1.3756,-1.4550) ( 2.2934, 0.0000)
```

F07HDF (SPBTRF/DPBTRF) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07HDF (SPBTRF/DPBTRF) computes the Cholesky factorization of a real symmetric positive-definite band matrix.

2. Specification

```
SUBROUTINE F07HDF (UPLO, N, KD, AB, LDAB, INFO)
ENTRY      spbtrf  (UPLO, N, KD, AB, LDAB, INFO)
INTEGER          N, KD, LDAB, INFO
real             AB(LDAB,*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine forms the Cholesky factorization of a real symmetric positive-definite band matrix A either as $A = U^T U$ if UPLO = 'U', or $A = LL^T$ if UPLO = 'L', where U (or L) is an upper (or lower) triangular band matrix with the same number of super-diagonals (or sub-diagonals) as A.

4. References

[1] DEMMEL, J.W.
On Floating-point Errors in Cholesky.
LAPACK Working Note No. 14, University of Tennessee, Knoxville, 1989.

[2] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §4.3.6.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether the upper or lower triangular part of A is stored and how A is factorized, as follows:

if UPLO = 'U', then the upper triangular part of A is stored and A is factorized as $U^T U$, where U is upper triangular;

if UPLO = 'L', then the lower triangular part of A is stored and A is factorized as LL^T, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: KD – INTEGER. *Input*

On entry: k, the number of super-diagonals or sub-diagonals of the matrix A.

Constraint: KD $\geq$ 0.

4: AB(LDAB,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array AB must be at least max(1,N).

On entry: the n by n symmetric band matrix A, stored in rows 1 to k+1. More precisely, if UPLO = 'U', the elements of the upper triangle of A within the band must be stored with element a_{ij} in AB(k+1+i−j,j) for max(1,j−k) ≤ i ≤ j; if UPLO = 'L', the elements of the lower triangle of A within the band must be stored with element a_{ij} in AB(1+i−j,j) for j ≤ i ≤ min(n,j+k).

On exit: the upper or lower triangle of A is overwritten by the Cholesky factor U or L as specified by UPLO, using the same storage format as described above.

5: LDAB – INTEGER. *Input*

On entry: the first dimension of the array AB as declared in the (sub)program from which F07HDF (SPBTRF/DPBTRF) is called.

Constraint: LDAB ≥ KD + 1.

6: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = −i, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, the leading minor of order i is not positive-definite and the factorization could not be completed. Hence A itself is not positive-definite. This may indicate an error in forming the matrix A. There is no routine specifically designed to factorize a symmetric band matrix which is not positive-definite; the matrix must be treated either as an unsymmetric band matrix, by calling F07BDF (SGBTRF/DGBTRF) or as a full symmetric matrix, by calling F07MDF (SSYTRF/DSYTRF).

7. Accuracy

If UPLO = 'U', the computed factor U is the exact factor of a perturbed matrix $A + E$, where

$$|E| \leq c(k+1)\varepsilon|U^T||U|,$$

$c(k+1)$ is a modest linear function of k+1, and ε is the ***machine precision***. If UPLO = 'L', a similar statement holds for the computed factor L. It follows that $|e_{ij}| \leq c(k+1)\varepsilon\sqrt{a_{ii}a_{jj}}$.

8. Further Comments

The total number of floating-point operations is approximately $n(k+1)^2$, assuming $n \gg k$.

A call to this routine may be followed by calls to the routines:

F07HEF (SPBTRS/DPBTRS) to solve $AX = B$;
F07HGF (SPBCON/DPBCON) to estimate the condition number of A.

The complex analogue of this routine is F07HRF (CPBTRF/ZPBTRF).

9. Example

To compute the Cholesky factorization of the matrix A, where

$$A = \begin{pmatrix} 5.49 & 2.68 & 0.00 & 0.00 \\ 2.68 & 5.63 & -2.39 & 0.00 \\ 0.00 & -2.39 & 2.60 & -2.22 \\ 0.00 & 0.00 & -2.22 & 5.17 \end{pmatrix}.$$

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07HDF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, KMAX, LDAB
      PARAMETER        (NMAX=8,KMAX=8,LDAB=KMAX+1)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, KD, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             AB(LDAB,NMAX)
*     .. External Subroutines ..
      EXTERNAL         spbtrf, X04CEF
*     .. Intrinsic Functions ..
      INTRINSIC        MAX, MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07HDF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, KD
      IF (N.LE.NMAX .AND. KD.LE.KMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            DO 20 I = 1, N
               READ (NIN,*) (AB(KD+1+I-J,J),J=I,MIN(N,I+KD))
   20       CONTINUE
         ELSE IF (UPLO.EQ.'L') THEN
            DO 40 I = 1, N
               READ (NIN,*) (AB(1+I-J,J),J=MAX(1,I-KD),I)
   40       CONTINUE
         END IF
*
*        Factorize A
*
         CALL spbtrf(UPLO,N,KD,AB,LDAB,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Print factor
*
            IFAIL = 0
*
            IF (UPLO.EQ.'U') THEN
*
               CALL X04CEF(N,N,0,KD,AB,LDAB,'Factor',IFAIL)
*
            ELSE IF (UPLO.EQ.'L') THEN
*
               CALL X04CEF(N,N,KD,0,AB,LDAB,'Factor',IFAIL)
*
            END IF
*
         ELSE
            WRITE (NOUT,*) 'A is not positive-definite'
         END IF
      END IF
      STOP
*
      END
```

9.2. Program Data

```
F07HDF Example Program Data
  4  1                          :Values of N and KD
  'L'                           :Value of UPLO
  5.49
  2.68   5.63
        -2.39   2.60
                -2.22   5.17    :End of matrix A
```

9.3. Program Results

```
 F07HDF Example Program Results

 Factor
                1          2          3          4
  1      2.3431
  2      1.1438     2.0789
  3                -1.1497     1.1306
  4                           -1.9635     1.1465
```

F07HEF (SPBTRS/DPBTRS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07HEF (SPBTRS/DPBTRS) solves a real symmetric positive-definite band system of linear equations with multiple right-hand sides, $AX = B$, where A has been factorized by F07HDF (SPBTRF/DPBTRF).

2. Specification

```
SUBROUTINE F07HEF (UPLO, N, KD, NRHS, AB, LDAB, B, LDB, INFO)
ENTRY      spbtrs  (UPLO, N, KD, NRHS, AB, LDAB, B, LDB, INFO)
INTEGER          N, KD, NRHS, LDAB, LDB, INFO
real             AB(LDAB,*), B(LDB,*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

To solve a real symmetric positive-definite band system of linear equations $AX = B$, this routine must be preceded by a call to F07HDF (SPBTRF/DPBTRF) which computes the Cholesky factorization of A. The solution X is computed by forward and backward substitution.

If UPLO = 'U', $A = U^TU$, where U is upper triangular; the solution X is computed by solving $U^TY = B$ and then $UX = Y$.

If UPLO = 'L', $A = LL^T$, where L is lower triangular; the solution X is computed by solving $LY = B$ and then $L^TX = Y$.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §4.3.6.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A has been factorized as U^TU or LL^T as follows:

if UPLO = 'U', then $A = U^TU$, where U is upper triangular;

if UPLO = 'L', then $A = LL^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: KD – INTEGER. *Input*

On entry: k, the number of super-diagonals or sub-diagonals of the matrix A.

Constraint: KD $\geq$ 0.

4: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS $\geq$ 0.

5: AB(LDAB,*) – ***real*** array. *Input*

Note: the second dimension of the array AB must be at least max(1,N).

On entry: the Cholesky factor of A, as returned by F07HDF (SPBTRF/DPBTRF).

6: LDAB – INTEGER. *Input*

On entry: the first dimension of the array AB as declared in the (sub)program from which F07HEF (SPBTRS/DPBTRS) is called.

Constraint: LDAB ≥ KD + 1.

7: B(LDB,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

On exit: the n by r solution matrix X.

8: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07HEF (SPBTRS/DPBTRS) is called.

Constraint: LDB ≥ max(1,N).

9: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

For each right-hand side vector b, the computed solution x is the exact solution of a perturbed system of equations $(A+E)x = b$, where

$|E| \leq c(k+1)\varepsilon|U^T||U|$ if UPLO = 'U',

$|E| \leq c(k+1)\varepsilon|L||L^T|$ if UPLO = 'L',

$c(k+1)$ is a modest linear function of $k+1$, and ε is the ***machine precision.***

If $\hat{x}$ is the true solution, then the computed solution x satisfies a forward error bound of the form

$$\frac{\|x-\hat{x}\|_\infty}{\|x\|_\infty} \leq c(k+1)\mathrm{cond}(A,x)\varepsilon$$

where $\mathrm{cond}(A,x) = \||A^{-1}||A||x|\|_\infty / \|x\|_\infty \leq \mathrm{cond}(A) = \||A^{-1}||A|\|_\infty \leq \kappa_\infty(A)$. Note that $\mathrm{cond}(A,x)$ can be much smaller than $\mathrm{cond}(A)$.

Forward and backward error bounds can be computed by calling F07HHF (SPBRFS/DPBRFS), and an estimate for $\kappa_\infty(A)$ $(= \kappa_1(A))$ can be obtained by calling F07HGF (SPBCON/DPBCON).

8. Further Comments

The total number of floating-point operations is approximately $4nkr$, assuming $n \gg k$.

This routine may be followed by a call to F07HHF (SPBRFS/DPBRFS) to refine the solution and return an error estimate.

The complex analogue of this routine is F07HSF (CPBTRS/ZPBTRS).

9. Example

To solve the system of equations $AX = B$, where

$$A = \begin{pmatrix} 5.49 & 2.68 & 0.00 & 0.00 \\ 2.68 & 5.63 & -2.39 & 0.00 \\ 0.00 & -2.39 & 2.60 & -2.22 \\ 0.00 & 0.00 & -2.22 & 5.17 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 22.09 & 5.10 \\ 9.31 & 30.81 \\ -5.24 & -25.82 \\ 11.83 & 22.90 \end{pmatrix}.$$

Here A is symmetric and positive-definite, and is treated as a band matrix, which must first be factorized by F07HDF (SPBTRF/DPBTRF).

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07HEF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, KDMAX, LDAB, NRHMAX, LDB
      PARAMETER        (NMAX=8,KDMAX=8,LDAB=KDMAX+1,NRHMAX=NMAX,
     +                 LDB=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, KD, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             AB(LDAB,NMAX), B(LDB,NRHMAX)
*     .. External Subroutines ..
      EXTERNAL         spbtrf, spbtrs, X04CAF
*     .. Intrinsic Functions ..
      INTRINSIC        MAX, MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07HEF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, KD, NRHS
      IF (N.LE.NMAX .AND. KD.LE.KDMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            DO 20 I = 1, N
               READ (NIN,*) (AB(KD+1+I-J,J),J=I,MIN(N,I+KD))
   20       CONTINUE
         ELSE IF (UPLO.EQ.'L') THEN
            DO 40 I = 1, N
               READ (NIN,*) (AB(1+I-J,J),J=MAX(1,I-KD),I)
   40       CONTINUE
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
*
*        Factorize A
*
         CALL spbtrf(UPLO,N,KD,AB,LDAB,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute solution
*
            CALL spbtrs(UPLO,N,KD,NRHS,AB,LDAB,B,LDB,INFO)
*
```

```
*           Print solution
*
            IFAIL = 0
*
            CALL X04CAF('General',' ',N,NRHS,B,LDB,'Solution(s)',IFAIL)
*
         ELSE
            WRITE (NOUT,*) 'A is not positive-definite'
         END IF
      END IF
      STOP
*
      END
```

9.2. Program Data

```
F07HEF Example Program Data
  4  1  2                        :Values of N, KD and NRHS
  'L'                            :Value of UPLO
  5.49
  2.68   5.63
        -2.39   2.60
                -2.22   5.17     :End of matrix A
 22.09   5.10
  9.31  30.81
 -5.24 -25.82
 11.83  22.90                    :End of matrix B
```

9.3. Program Results

```
 F07HEF Example Program Results

 Solution(s)
                  1           2
  1        5.0000     -2.0000
  2       -2.0000      6.0000
  3       -3.0000     -1.0000
  4        1.0000      4.0000
```

F07HGF (SPBCON/DPBCON) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07HGF (SPBCON/DPBCON) estimates the condition number of a real symmetric positive-definite band matrix A, where A has been factorized by F07HDF (SPBTRF/DPBTRF).

2. Specification

```
      SUBROUTINE F07HGF (UPLO, N, KD, AB, LDAB, ANORM, RCOND, WORK, IWORK,
     1                   INFO)
      ENTRY       spbcon (UPLO, N, KD, AB, LDAB, ANORM, RCOND, WORK, IWORK,
     1                   INFO)
      INTEGER           N, KD, LDAB, IWORK(*), INFO
      real              AB(LDAB,*), ANORM, RCOND, WORK(*)
      CHARACTER*1       UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine estimates the condition number (in the 1-norm) of a real symmetric positive-definite band matrix A:

$$\kappa_1(A) = \|A\|_1 \|A^{-1}\|_1.$$

Since A is symmetric, $\kappa_1(A) = \kappa_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty$.

Because $\kappa_1(A)$ is infinite if A is singular, the routine actually returns an estimate of the **reciprocal** of $\kappa_1(A)$.

The routine should be preceded by a call to F06REF to compute $\|A\|_1$ and a call to F07HDF (SPBTRF/DPBTRF) to compute the Cholesky factorization of A. The routine then uses Higham's implementation of Hager's method [1] to estimate $\|A^{-1}\|_1$.

4. References

[1] HIGHAM, N.J.
FORTRAN Codes for Estimating the One-norm of a Real or Complex Matrix, with Applications to Condition Estimation.
ACM Trans. Math. Softw., 14, pp. 381-396, 1988.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A has been factorized as U^TU or LL^T as follows:

if UPLO = 'U', then $A = U^TU$, where U is upper triangular;

if UPLO = 'L', then $A = LL^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: KD – INTEGER. *Input*

On entry: k, the number of super-diagonals or sub-diagonals of the matrix A.

Constraint: KD $\geq$ 0.

4: AB(LDAB,*) – ***real*** array. *Input*

Note: the second dimension of the array AB must be at least max(1,N).

On entry: the Cholesky factor of A, as returned by F07HDF (SPBTRF/DPBTRF).

5: LDAB – INTEGER. *Input*

On entry: the first dimension of the array AB as declared in the (sub)program from which F07HGF (SPBCON/DPBCON) is called.

Constraint: LDAB $\geq$ KD + 1.

6: ANORM – ***real***. *Input*

On entry: the 1-norm of the **original** matrix A, which may be computed by calling F06REF. ANORM must be computed either **before** calling F07HDF (SPBTRF/DPBTRF) or else from a copy of the original matrix A.

Constraint: ANORM $\geq$ 0.0.

7: RCOND – ***real***. *Output*

On exit: an estimate of the reciprocal of the condition number of A. RCOND is set to zero if exact singularity is detected or the estimate underflows. If RCOND is less than ***machine precision***, then A is singular to working precision.

8: WORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,3*N).

9: IWORK(*) – INTEGER array. *Workspace*

Note: the dimension of the array IWORK must be at least max(1,N).

10: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed estimate RCOND is never less than the true value ρ, and in practice is nearly always less than 10ρ, although examples can be constructed where RCOND is much larger.

8. Further Comments

A call to this routine involves solving a number of systems of linear equations of the form $Ax = b$; the number is usually 4 or 5 and never more than 11. Each solution involves approximately $4nk$ floating-point operations (assuming $n \gg k$) but takes considerably longer than a call to F07HEF (SPBTRS/DPBTRS) with 1 right-hand side, because extra care is taken to avoid overflow when A is approximately singular.

The complex analogue of this routine is F07HUF (CPBCON/ZPBCON).

9. Example

To estimate the condition number in the 1-norm (or infinity-norm) of the matrix A, where

$$A = \begin{pmatrix} 5.49 & 2.68 & 0.00 & 0.00 \\ 2.68 & 5.63 & -2.39 & 0.00 \\ 0.00 & -2.39 & 2.60 & -2.22 \\ 0.00 & 0.00 & -2.22 & 5.17 \end{pmatrix}.$$

Here A is symmetric and positive-definite, and is treated as a band matrix, which must first be factorized by F07HDF (SPBTRF/DPBTRF). The true condition number in the 1-norm is 74.15.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07HGF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, KDMAX, LDAB
      PARAMETER        (NMAX=8,KDMAX=8,LDAB=KDMAX+1)
*     .. Local Scalars ..
      real             ANORM, RCOND
      INTEGER          I, INFO, J, KD, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             AB(LDAB,NMAX), WORK(3*NMAX)
      INTEGER          IWORK(NMAX)
*     .. External Functions ..
      real             F06REF, X02AJF
      EXTERNAL         F06REF, X02AJF
*     .. External Subroutines ..
      EXTERNAL         spbcon, spbtrf
*     .. Intrinsic Functions ..
      INTRINSIC        MAX, MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07HGF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, KD
      IF (N.LE.NMAX .AND. KD.LE.KDMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            DO 20 I = 1, N
               READ (NIN,*) (AB(KD+1+I-J,J),J=I,MIN(N,I+KD))
   20       CONTINUE
         ELSE IF (UPLO.EQ.'L') THEN
            DO 40 I = 1, N
               READ (NIN,*) (AB(1+I-J,J),J=MAX(1,I-KD),I)
   40       CONTINUE
         END IF
*
*        Compute norm of A
*
         ANORM = F06REF('1-norm',UPLO,N,KD,AB,LDAB,WORK)
*
*        Factorize A
*
         CALL spbtrf(UPLO,N,KD,AB,LDAB,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
```

```
*           Estimate condition number
*
            CALL spbcon(UPLO,N,KD,AB,LDAB,ANORM,RCOND,WORK,IWORK,INFO)
*
            IF (RCOND.GE.X02AJF()) THEN
               WRITE (NOUT,99999) 'Estimate of condition number =',
     +           1.0e0/RCOND
            ELSE
               WRITE (NOUT,*) 'A is singular to working precision'
            END IF
         ELSE
            WRITE (NOUT,*) 'A is not positive-definite'
         END IF
      END IF
      STOP
*
99999 FORMAT (1X,A,1P,e10.2)
      END
```

9.2. Program Data

```
F07HGF Example Program Data
  4  1                          :Values of N and KD
  'L'                           :Value of UPLO
  5.49
  2.68   5.63
        -2.39   2.60
                -2.22   5.17    :End of matrix A
```

9.3. Program Results

```
 F07HGF Example Program Results

 Estimate of condition number =  7.42E+01
```

F07HHF (SPBRFS/DPBRFS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07HHF (SPBRFS/DPBRFS) returns error bounds for the solution of a real symmetric positive-definite band system of linear equations with multiple right-hand sides, $AX = B$. It improves the solution by iterative refinement, in order to reduce the backward error as much as possible.

2. Specification

```
 SUBROUTINE F07HHF (UPLO, N, KD, NRHS, AB, LDAB, AFB, LDAFB, B, LDB, X,
1                   LDX, FERR, BERR, WORK, IWORK, INFO)
 ENTRY       spbrfs  (UPLO, N, KD, NRHS, AB, LDAB, AFB, LDAFB, B, LDB, X,
1                   LDX, FERR, BERR, WORK, IWORK, INFO)
 INTEGER          N, KD, NRHS, LDAB, LDAFB, LDB, LDX, IWORK(*), INFO
 real             AB(LDAB,*), AFB(LDAFB,*), B(LDB,*), X(LDX,*), FERR(*),
1                 BERR(*), WORK(*)
 CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine returns the backward errors and estimated bounds on the forward errors for the solution of a real symmetric positive-definite band system of linear equations with multiple right-hand sides $AX = B$. The routine handles each right-hand side vector (stored as a column of the matrix B) independently, so we describe the function of the routine in terms of a single right-hand side b and solution x.

Given a computed solution x, the routine computes the *component-wise backward error* β. This is the size of the smallest relative perturbation in each element of A and b such that x is the exact solution of a perturbed system

$$(A+\delta A)x = b + \delta b$$

$$|\delta a_{ij}| \leq \beta |a_{ij}| \text{ and } |\delta b_i| \leq \beta |b_i|.$$

Then the routine estimates a bound for the *component-wise forward error* in the computed solution, defined by:

$$\max_i |x_i - \hat{x}_i| / \max_i |x_i|$$

where $\hat{x}$ is the true solution.

For details of the method, see the Chapter Introduction.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether the upper or lower triangular part of A is stored and how A has been factorized, as follows:

if UPLO = 'U', then the upper triangular part of A is stored and A is factorized as $U^T U$, where U is upper triangular;

if UPLO = 'L', then the lower triangular part of A is stored and A is factorized as LL^T, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: KD – INTEGER. *Input*

On entry: k, the number of super-diagonals or sub-diagonals of the matrix A.

Constraint: KD ≥ 0.

4: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS ≥ 0.

5: AB(LDAB,*) – ***real*** array. *Input*

Note: the second dimension of the array AB must be at least max(1,N).

On entry: the n by n original symmetric band matrix A as supplied to F07HDF (SPBTRF/DPBTRF).

6: LDAB – INTEGER. *Input*

On entry: the first dimension of the array AB as declared in the (sub)program from which F07HHF (SPBRFS/DPBRFS) is called.

Constraint: LDAB ≥ KD + 1.

7: AFB(LDAFB,*) – ***real*** array. *Input*

Note: the second dimension of the array AFB must be at least max(1,N).

On entry: the Cholesky factor of A, as returned by F07HDF (SPBTRF/DPBTRF).

8: LDAFB – INTEGER. *Input*

On entry: the first dimension of the array AFB as declared in the (sub)program from which F07HHF (SPBRFS/DPBRFS) is called.

Constraint: LDAFB ≥ KD + 1.

9: B(LDB,*) – ***real*** array. *Input*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

10: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07HHF (SPBRFS/DPBRFS) is called.

Constraint: LDB ≥ max(1,N).

11: X(LDX,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array X must be at least max(1,NRHS).

On entry: the n by r solution matrix X, as returned by F07HEF (SPBTRS/DPBTRS).

On exit: the improved solution matrix X.

12: LDX – INTEGER. *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which F07HHF (SPBRFS/DPBRFS) is called.

Constraint: LDX ≥ max(1,N).

13: FERR(*) – ***real*** array. *Output*

Note: the dimension of the array FERR must be at least max(1,NRHS).

On exit: FERR(j) contains an estimated error bound for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

14: BERR(*) – ***real*** array. *Output*

Note: the dimension of the array BERR must be at least max(1,NRHS).

On exit: BERR(j) contains the component-wise backward error bound β for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

15: WORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,3*N).

16: IWORK(*) – INTEGER array. *Workspace*

Note: the dimension of the array IWORK must be at least max(1,N).

17: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The bounds returned in FERR are not rigorous, because they are estimated, not computed exactly; but in practice they almost always overestimate the actual error.

8. Further Comments

For each right-hand side, computation of the backward error involves a minimum of $8nk$ floating-point operations. Each step of iterative refinement involves an additional $12nk$ operations. This assumes $n \gg k$. At most 5 steps of iterative refinement are performed, but usually only 1 or 2 steps are required.

Estimating the forward error involves solving a number of systems of linear equations of the form $Ax = b$; the number is usually 4 or 5 and never more than 11. Each solution involves approximately $4nk$ operations.

The complex analogue of this routine is F07HVF (CPBRFS/ZPBRFS).

9. Example

To solve the system of equations $AX = B$ using iterative refinement and to compute the forward and backward error bounds, where

$$A = \begin{pmatrix} 5.49 & 2.68 & 0.00 & 0.00 \\ 2.68 & 5.63 & -2.39 & 0.00 \\ 0.00 & -2.39 & 2.60 & -2.22 \\ 0.00 & 0.00 & -2.22 & 5.17 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 22.09 & 5.10 \\ 9.31 & 30.81 \\ -5.24 & -25.82 \\ 11.83 & 22.90 \end{pmatrix}.$$

Here A is symmetric and positive-definite, and is treated as a band matrix, which must first be factorized by F07HDF (SPBTRF/DPBTRF).

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07HHF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      real             ZERO
      PARAMETER        (ZERO=0.0e0)
      INTEGER          NMAX, NRHMAX, KDMAX, LDAB, LDAFB, LDB, LDX
      PARAMETER        (NMAX=8,NRHMAX=NMAX,KDMAX=8,LDAB=KDMAX+1,
     +                 LDAFB=KDMAX+1,LDB=NMAX,LDX=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, KD, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             AB(LDAB,NMAX), AFB(LDAFB,NMAX), B(LDB,NRHMAX),
     +                 BERR(NRHMAX), FERR(NRHMAX), WORK(3*NMAX),
     +                 X(LDX,NMAX)
      INTEGER          IWORK(NMAX)
*     .. External Subroutines ..
      EXTERNAL         F06QFF, F06QHF, spbrfs, spbtrf, spbtrs, X04CAF
*     .. Intrinsic Functions ..
      INTRINSIC        MAX, MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07HHF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, KD, NRHS
      IF (N.LE.NMAX .AND. KD.LE.KDMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Set A to zero to avoid referencing uninitialized elements
*
         CALL F06QHF('General',KD+1,N,ZERO,ZERO,AB,LDAB)
*
*        Read A and B from data file, and copy A to AFB and B to X
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            DO 20 I = 1, N
               READ (NIN,*) (AB(KD+1+I-J,J),J=I,MIN(N,I+KD))
   20       CONTINUE
         ELSE IF (UPLO.EQ.'L') THEN
            DO 40 I = 1, N
               READ (NIN,*) (AB(1+I-J,J),J=MAX(1,I-KD),I)
   40       CONTINUE
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
*
         CALL F06QFF('General',KD+1,N,AB,LDAB,AFB,LDAFB)
*
         CALL F06QFF('General',N,NRHS,B,LDB,X,LDX)
*
*        Factorize A in the array AFB
*
         CALL spbtrf(UPLO,N,KD,AFB,LDAFB,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute solution in the array X
*
            CALL spbtrs(UPLO,N,KD,NRHS,AFB,LDAFB,X,LDX,INFO)
*
```

```
*           Improve solution, and compute backward errors and
*           estimated bounds on the forward errors
*
            CALL spbrfs(UPLO,N,KD,NRHS,AB,LDAB,AFB,LDAFB,B,LDB,X,LDX,
     +                  FERR,BERR,WORK,IWORK,INFO)
*
*           Print solution
*
            IFAIL = 0
*
            CALL X04CAF('General',' ',N,NRHS,X,LDX,'Solution(s)',IFAIL)
*
            WRITE (NOUT,*)
            WRITE (NOUT,*) 'Backward errors (machine-dependent)'
            WRITE (NOUT,99999) (BERR(J),J=1,NRHS)
            WRITE (NOUT,*)
     +        'Estimated forward error bounds (machine-dependent)'
            WRITE (NOUT,99999) (FERR(J),J=1,NRHS)
          ELSE
            WRITE (NOUT,*) 'A is not positive-definite'
          END IF
        END IF
        STOP
*
99999 FORMAT ((3X,1P,7e11.1))
      END
```

9.2. Program Data

```
F07HHF Example Program Data
  4  1  2                        :Values of N, KD and NRHS
  'L'                             :Value of UPLO
  5.49
  2.68   5.63
        -2.39   2.60
                -2.22   5.17     :End of matrix A
 22.09   5.10
  9.31  30.81
 -5.24 -25.82
 11.83  22.90                     :End of matrix B
```

9.3. Program Results

```
 F07HHF Example Program Results

 Solution(s)
                  1           2
  1       5.0000     -2.0000
  2      -2.0000      6.0000
  3      -3.0000     -1.0000
  4       1.0000      4.0000

 Backward errors (machine-dependent)
       4.9E-17     6.1E-17
 Estimated forward error bounds (machine-dependent)
       2.0E-14     3.0E-14
```

F07HRF (CPBTRF/ZPBTRF) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07HRF (CPBTRF/ZPBTRF) computes the Cholesky factorization of a complex Hermitian positive-definite band matrix.

2. Specification

```
SUBROUTINE F07HRF (UPLO, N, KD, AB, LDAB, INFO)
ENTRY      cpbtrf  (UPLO, N, KD, AB, LDAB, INFO)
INTEGER        N, KD, LDAB, INFO
complex        AB(LDAB,*)
CHARACTER*1    UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine forms the Cholesky factorization of a complex Hermitian positive-definite band matrix A either as $A = U^H U$ if UPLO = 'U', or $A = LL^H$ if UPLO = 'L', where U (or L) is an upper (or lower) triangular band matrix with the same number of super-diagonals (or sub-diagonals) as A.

4. References

[1] DEMMEL, J.W.
On Floating-point Errors in Cholesky.
LAPACK Working Note No. 14, University of Tennessee, Knoxville, 1989.

[2] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §4.3.6.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether the upper or lower triangular part of A is stored and how A is factorized, as follows:

if UPLO = 'U', then the upper triangular part of A is stored and A is factorized as $U^H U$, where U is upper triangular;

if UPLO = 'L', then the lower triangular part of A is stored and A is factorized as LL^H, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: KD – INTEGER. *Input*

On entry: k, the number of super-diagonals or sub-diagonals of the matrix A.

Constraint: KD $\geq$ 0.

4: AB(LDAB,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array AB must be at least max(1,N).

On entry: the n by n Hermitian positive-definite band matrix A, stored in rows 1 to $k + 1$. More precisely, if UPLO = 'U', the elements of the upper triangle of A within the band must be stored with element a_{ij} in AB(k+1+i−j,j) for max(1,j−k) ≤ i ≤ j; if UPLO = 'L', the elements of the lower triangle of A within the band must be stored with element a_{ij} in AB(1+i−j,j) for j ≤ i ≤ min(n,j+k).

On exit: the upper or lower triangle of A is overwritten by the Cholesky factor U or L as specified by UPLO, using the same storage format as described above.

5: LDAB – INTEGER. *Input*

On entry: the first dimension of the array AB as declared in the (sub)program from which F07HRF (CPBTRF/ZPBTRF) is called.

Constraint: LDAB ≥ KD + 1.

6: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = −i, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, the leading minor of order i is not positive-definite and the factorization could not be completed. Hence A itself is not positive-definite. This may indicate an error in forming the matrix A. There is no routine specifically designed to factorize a Hermitian band matrix which is not positive-definite; the matrix must be treated either as an unsymmetric band matrix, by calling F07BRF (CGBTRF/ZGBTRF) or as a full Hermitian matrix, by calling F07MRF (CHETRF/ZHETRF).

7. Accuracy

If UPLO = 'U', the computed factor U is the exact factor of a perturbed matrix $A + E$, where

$$|E| \leq c(k+1)\varepsilon|U^H||U|,$$

$c(k+1)$ is a modest linear function of $k+1$, and ε is the ***machine precision***. If UPLO = 'L', a similar statement holds for the computed factor L. It follows that $|e_{ij}| \leq c(k+1)\varepsilon\sqrt{a_{ii}a_{jj}}$.

8. Further Comments

The total number of real floating-point operations is approximately $4n(k+1)^2$, assuming $n \gg k$.

A call to this routine may be followed by calls to the routines:

F07HSF (CPBTRS/ZPBTRS) to solve $AX = B$;
F07HUF (CPBCON/ZPBCON) to estimate the condition number of A.

The real analogue of this routine is F07HDF (SPBTRF/DPBTRF).

9. Example

To compute the Cholesky factorization of the matrix A, where

$$A = \begin{pmatrix} 9.39 + 0.00i & 1.08 - 1.73i & 0.00 + 0.00i & 0.00 + 0.00i \\ 1.08 + 1.73i & 1.69 + 0.00i & -0.04 + 0.29i & 0.00 + 0.00i \\ 0.00 + 0.00i & -0.04 - 0.29i & 2.65 + 0.00i & -0.33 + 2.24i \\ 0.00 + 0.00i & 0.00 + 0.00i & -0.33 - 2.24i & 2.17 + 0.00i \end{pmatrix}.$$

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07HRF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, KMAX, LDAB
      PARAMETER        (NMAX=8,KMAX=8,LDAB=KMAX+1)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, KD, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          AB(LDAB,NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         cpbtrf, X04DFF
*     .. Intrinsic Functions ..
      INTRINSIC        MAX, MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07HRF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, KD
      IF (N.LE.NMAX .AND. KD.LE.KMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            DO 20 I = 1, N
               READ (NIN,*) (AB(KD+1+I-J,J),J=I,MIN(N,I+KD))
   20       CONTINUE
         ELSE IF (UPLO.EQ.'L') THEN
            DO 40 I = 1, N
               READ (NIN,*) (AB(1+I-J,J),J=MAX(1,I-KD),I)
   40       CONTINUE
         END IF
*
*        Factorize A
*
         CALL cpbtrf(UPLO,N,KD,AB,LDAB,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Print factor
*
            IFAIL = 0
            IF (UPLO.EQ.'U') THEN
*
               CALL X04DFF(N,N,0,KD,AB,LDAB,'Bracketed','F7.4','Factor',
     +                     'Integer',RLABS,'Integer',CLABS,80,0,IFAIL)
*
            ELSE IF (UPLO.EQ.'L') THEN
*
               CALL X04DFF(N,N,KD,0,AB,LDAB,'Bracketed','F7.4','Factor',
     +                     'Integer',RLABS,'Integer',CLABS,80,0,IFAIL)
*
            END IF
*
```

```
         ELSE
            WRITE (NOUT,*) 'A is not positive-definite'
         END IF
      END IF
      STOP
*
      END
```

9.2. Program Data

```
F07HRF Example Program Data
  4  1                                                    :Values of N and KD
  'L'                                                     :Value of UPLO
 ( 9.39, 0.00)
 ( 1.08, 1.73) ( 1.69, 0.00)
               (-0.04,-0.29) ( 2.65, 0.00)
                             (-0.33,-2.24) ( 2.17, 0.00)  :End of matrix A
```

9.3. Program Results

```
F07HRF Example Program Results

Factor
                     1                  2                  3                  4
 1  ( 3.0643, 0.0000)
 2  ( 0.3524, 0.5646) ( 1.1167, 0.0000)
 3                    (-0.0358,-0.2597) ( 1.6066, 0.0000)
 4                                      (-0.2054,-1.3942) ( 0.4289, 0.0000)
```

F07HSF (CPBTRS/ZPBTRS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07HSF (CPBTRS/ZPBTRS) solves a complex Hermitian positive-definite band system of linear equations with multiple right-hand sides, $AX = B$, where A has been factorized by F07HRF (CPBTRF/ZPBTRF).

2. Specification

```
SUBROUTINE F07HSF (UPLO, N, KD, NRHS, AB, LDAB, B, LDB, INFO)
ENTRY      cpbtrs  (UPLO, N, KD, NRHS, AB, LDAB, B, LDB, INFO)
INTEGER          N, KD, NRHS, LDAB, LDB, INFO
complex          AB(LDAB,*), B(LDB,*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

To solve a complex Hermitian positive-definite band system of linear equations $AX = B$, this routine must be preceded by a call to F07HRF (CPBTRF/ZPBTRF) which computes the Cholesky factorization of A. The solution X is computed by forward and backward substitution.

If UPLO = 'U', $A = U^H U$, where U is upper triangular; the solution X is computed by solving $U^H Y = B$ and then $UX = Y$.

If UPLO = 'L', $A = LL^H$, where L is lower triangular; the solution X is computed by solving $LY = B$ and then $L^H X = Y$.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §4.3.6.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A has been factorized as $U^H U$ or LL^H as follows:

if UPLO = 'U', then $A = U^H U$, where U is upper triangular;

if UPLO = 'L', then $A = LL^H$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: KD – INTEGER. *Input*

On entry: k, the number of super-diagonals or sub-diagonals of the matrix A.

Constraint: KD $\geq$ 0.

4: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS $\geq$ 0.

5: AB(LDAB,*) – ***complex*** array. *Input*

Note: the second dimension of the array AB must be at least max(1,N).

On entry: the Cholesky factor of A, as returned by F07HRF (CPBTRF/ZPBTRF).

6: LDAB – INTEGER. *Input*

On entry: the first dimension of the array AB as declared in the (sub)program from which F07HSF (CPBTRS/ZPBTRS) is called.

Constraint: LDAB ≥ KD + 1.

7: B(LDB,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

On exit: the n by r solution matrix X.

8: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07HSF (CPBTRS/ZPBTRS) is called.

Constraint: LDB ≥ max(1,N).

9: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

For each right-hand side vector b, the computed solution x is the exact solution of a perturbed system of equations $(A+E)x = b$, where

$|E| \leq c(k+1)\varepsilon|U^H||U|$ if UPLO = 'U',

$|E| \leq c(k+1)\varepsilon|L||L^H|$ if UPLO = 'L',

$c(k+1)$ is a modest linear function of $k + 1$, and ε is the ***machine precision***.

If $\hat{x}$ is the true solution, then the computed solution x satisfies a forward error bound of the form

$$\frac{\|x-\hat{x}\|_\infty}{\|x\|_\infty} \leq c(k+1)\mathrm{cond}(A,x)\varepsilon$$

where $\mathrm{cond}(A,x) = \||A^{-1}||A||x|\|_\infty / \|x\|_\infty \leq \mathrm{cond}(A) = \||A^{-1}||A|\|_\infty \leq \kappa_\infty(A)$. Note that $\mathrm{cond}(A,x)$ can be much smaller than $\mathrm{cond}(A)$.

Forward and backward error bounds can be computed by calling F07HVF (CPBRFS/ZPBRFS), and an estimate for $\kappa_\infty(A)$ $(= \kappa_1(A))$ can be obtained by calling F07HUF (CPBCON/ZPBCON).

8. Further Comments

The total number of real floating-point operations is approximately $16nkr$, assuming $n \gg k$.

This routine may be followed by a call to F07HVF (CPBRFS/ZPBRFS) to refine the solution and return an error estimate.

The real analogue of this routine is F07HEF (SPBTRS/DPBTRS).

9. Example

To solve the system of equations $AX = B$, where

$$A = \begin{pmatrix} 9.39 + 0.00i & 1.08 - 1.73i & 0.00 + 0.00i & 0.00 + 0.00i \\ 1.08 + 1.73i & 1.69 + 0.00i & -0.04 + 0.29i & 0.00 + 0.00i \\ 0.00 + 0.00i & -0.04 - 0.29i & 2.65 + 0.00i & -0.33 + 2.24i \\ 0.00 + 0.00i & 0.00 + 0.00i & -0.33 - 2.24i & 2.17 + 0.00i \end{pmatrix} \quad \text{and}$$

$$B = \begin{pmatrix} -12.42 + 68.42i & 54.30 - 56.56i \\ -9.93 + 0.88i & 18.32 + 4.76i \\ -27.30 - 0.01i & -4.40 + 9.97i \\ 5.31 + 23.63i & 9.43 + 1.41i \end{pmatrix}.$$

Here A is Hermitian positive-definite, and is treated as a band matrix, which must first be factorized by F07HRF (CPBTRF/ZPBTRF).

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07HSF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, KDMAX, LDAB, NRHMAX, LDB
      PARAMETER        (NMAX=8,KDMAX=8,LDAB=KDMAX+1,NRHMAX=NMAX,
     +                 LDB=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, KD, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          AB(LDAB,NMAX), B(LDB,NRHMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         cpbtrf, cpbtrs, X04DBF
*     .. Intrinsic Functions ..
      INTRINSIC        MAX, MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07HSF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, KD, NRHS
      IF (N.LE.NMAX .AND. KD.LE.KDMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            DO 20 I = 1, N
               READ (NIN,*) (AB(KD+1+I-J,J),J=I,MIN(N,I+KD))
   20       CONTINUE
         ELSE IF (UPLO.EQ.'L') THEN
            DO 40 I = 1, N
               READ (NIN,*) (AB(1+I-J,J),J=MAX(1,I-KD),I)
   40       CONTINUE
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
*
*        Factorize A
*
         CALL cpbtrf(UPLO,N,KD,AB,LDAB,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
```

```
*           Compute solution
*
            CALL cpbtrs(UPLO,N,KD,NRHS,AB,LDAB,B,LDB,INFO)
*
*           Print solution
*
            IFAIL = 0
            CALL X04DBF('General',' ',N,NRHS,B,LDB,'Bracketed','F7.4',
     +                  'Solution(s)','Integer',RLABS,'Integer',CLABS,
     +                  80,0,IFAIL)
         ELSE
            WRITE (NOUT,*) 'A is not positive-definite'
         END IF
       END IF
       STOP
*
       END
```

9.2. Program Data

```
F07HSF Example Program Data
  4  1  2                                                :Values of N, KD and NRHS
  'L'                                                       :Value of UPLO
 ( 9.39, 0.00)
 ( 1.08, 1.73) ( 1.69, 0.00)
               (-0.04,-0.29) ( 2.65, 0.00)
                             (-0.33,-2.24) ( 2.17, 0.00) :End of matrix A
(-12.42,68.42) (54.30,-56.56)
( -9.93, 0.88) (18.32,  4.76)
(-27.30,-0.01) (-4.40,  9.97)
(  5.31,23.63) ( 9.43,  1.41)                            :End of matrix B
```

9.3. Program Results

```
 F07HSF Example Program Results

 Solution(s)
                        1                   2
  1  (-1.0000, 8.0000) ( 5.0000,-6.0000)
  2  ( 2.0000,-3.0000) ( 2.0000, 3.0000)
  3  (-4.0000,-5.0000) (-8.0000, 4.0000)
  4  ( 7.0000, 6.0000) (-1.0000,-7.0000)
```

F07HUF (CPBCON/ZPBCON) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07HUF (CPBCON/ZPBCON) estimates the condition number of a complex Hermitian positive-definite band matrix A, where A has been factorized by F07HRF (CPBTRF/ZPBTRF).

2. Specification

```
 SUBROUTINE F07HUF (UPLO, N, KD, AB, LDAB, ANORM, RCOND, WORK, RWORK,
1                   INFO)
 ENTRY       cpbcon (UPLO, N, KD, AB, LDAB, ANORM, RCOND, WORK, RWORK,
1                   INFO)
 INTEGER          N, KD, LDAB, INFO
 real             ANORM, RCOND, RWORK(*)
 complex          AB(LDAB,*), WORK(*)
 CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine estimates the condition number (in the 1-norm) of a complex Hermitian positive-definite band matrix A:

$$\kappa_1(A) = \|A\|_1 \|A^{-1}\|_1.$$

Since A is Hermitian, $\kappa_1(A) = \kappa_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty$.

Because $\kappa_1(A)$ is infinite if A is singular, the routine actually returns an estimate of the **reciprocal** of $\kappa_1(A)$.

The routine should be preceded by a call to F06UEF to compute $\|A\|_1$ and a call to F07HRF (CPBTRF/ZPBTRF) to compute the Cholesky factorization of A. The routine then uses Higham's implementation of Hager's method [1] to estimate $\|A^{-1}\|_1$.

4. References

[1] HIGHAM, N.J.
FORTRAN Codes for Estimating the One-norm of a Real or Complex Matrix, with Applications to Condition Estimation.
ACM Trans. Math. Softw., 14, pp. 381-396, 1988.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A has been factorized as $U^H U$ or LL^H as follows:

if UPLO = 'U', then $A = U^H U$, where U is upper triangular;

if UPLO = 'L', then $A = LL^H$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: KD – INTEGER. *Input*

On entry: k, the number of super-diagonals or sub-diagonals of the matrix A.

Constraint: KD $\geq$ 0.

4: AB(LDAB,*) – ***complex*** array. *Input*

Note: the second dimension of the array AB must be at least max(1,N).

On entry: the Cholesky factor of A, as returned by F07HRF (CPBTRF/ZPBTRF).

5: LDAB – INTEGER. *Input*

On entry: the first dimension of the array AB as declared in the (sub)program from which F07HUF (CPBCON/ZPBCON) is called.

Constraint: LDAB $\geq$ KD + 1.

6: ANORM – ***real***. *Input*

On entry: the 1-norm of the **original** matrix A, which may be computed by calling F06UEF. ANORM must be computed either **before** calling F07HRF (CPBTRF/ZPBTRF) or else from a copy of the original matrix A.

Constraint: ANORM $\geq$ 0.0.

7: RCOND – ***real***. *Output*

On exit: an estimate of the reciprocal of the condition number of A. RCOND is set to zero if exact singularity is detected or the estimate underflows. If RCOND is less than ***machine precision***, then A is singular to working precision.

8: WORK(*) – ***complex*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,2*N).

9: RWORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array RWORK must be at least max(1,N).

10: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed estimate RCOND is never less than the true value ρ, and in practice is nearly always less than 10ρ, although examples can be constructed where RCOND is much larger.

8. Further Comments

A call to this routine involves solving a number of systems of linear equations of the form $Ax = b$; the number is usually 5 and never more than 11. Each solution involves approximately $16nk$ real floating-point operations (assuming $n \gg k$) but takes considerably longer than a call to F07HSF (CPBTRS/ZPBTRS) with 1 right-hand side, because extra care is taken to avoid overflow when A is approximately singular.

The real analogue of this routine is F07HGF (SPBCON/DPBCON).

9. Example

To estimate the condition number in the 1-norm (or infinity-norm) of the matrix A, where

$$A = \begin{pmatrix} 9.39 + 0.00i & 1.08 - 1.73i & 0.00 + 0.00i & 0.00 + 0.00i \\ 1.08 + 1.73i & 1.69 + 0.00i & -0.04 + 0.29i & 0.00 + 0.00i \\ 0.00 + 0.00i & -0.04 - 0.29i & 2.65 + 0.00i & -0.33 + 2.24i \\ 0.00 + 0.00i & 0.00 + 0.00i & -0.33 - 2.24i & 2.17 + 0.00i \end{pmatrix}.$$

Here A is Hermitian positive-definite, and is treated as a band matrix, which must first be factorized by F07HRF (CPBTRF/ZPBTRF). The true condition number in the 1-norm is 153.45.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07HUF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, KDMAX, LDAB
      PARAMETER        (NMAX=8,KDMAX=8,LDAB=KDMAX+1)
*     .. Local Scalars ..
      real             ANORM, RCOND
      INTEGER          I, INFO, J, KD, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          AB(LDAB,NMAX), WORK(2*NMAX)
      real             RWORK(NMAX)
*     .. External Functions ..
      real             F06UEF, X02AJF
      EXTERNAL         F06UEF, X02AJF
*     .. External Subroutines ..
      EXTERNAL         cpbcon, cpbtrf
*     .. Intrinsic Functions ..
      INTRINSIC        MAX, MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07HUF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, KD
      IF (N.LE.NMAX .AND. KD.LE.KDMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            DO 20 I = 1, N
               READ (NIN,*) (AB(KD+1+I-J,J),J=I,MIN(N,I+KD))
   20       CONTINUE
         ELSE IF (UPLO.EQ.'L') THEN
            DO 40 I = 1, N
               READ (NIN,*) (AB(1+I-J,J),J=MAX(1,I-KD),I)
   40       CONTINUE
         END IF
*
*        Compute norm of A
*
         ANORM = F06UEF('1-norm',UPLO,N,KD,AB,LDAB,RWORK)
*
*        Factorize A
*
         CALL cpbtrf(UPLO,N,KD,AB,LDAB,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
```

```
*           Estimate condition number
*
            CALL cpbcon(UPLO,N,KD,AB,LDAB,ANORM,RCOND,WORK,RWORK,INFO)
*
            IF (RCOND.GE.X02AJF()) THEN
               WRITE (NOUT,99999) 'Estimate of condition number =',
     +           1.0e0/RCOND
            ELSE
               WRITE (NOUT,*) 'A is singular to working precision'
            END IF
         ELSE
            WRITE (NOUT,*) 'A is not positive-definite'
         END IF
       END IF
       STOP
*
99999 FORMAT (1X,A,1P,e10.2)
      END
```

9.2. Program Data

```
F07HUF Example Program Data
  4  1                                                        :Values of N and KD
  'L'                                                         :Value of UPLO
 ( 9.39, 0.00)
 ( 1.08, 1.73) ( 1.69, 0.00)
               (-0.04,-0.29) ( 2.65, 0.00)
                             (-0.33,-2.24) ( 2.17, 0.00)     :End of matrix A
```

9.3. Program Results

```
F07HUF Example Program Results

Estimate of condition number =  1.22E+02
```

F07HVF (CPBRFS/ZPBRFS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07HVF (CPBRFS/ZPBRFS) returns error bounds for the solution of a complex Hermitian positive-definite band system of linear equations with multiple right-hand sides, $AX = B$. It improves the solution by iterative refinement, in order to reduce the backward error as much as possible.

2. Specification

```
      SUBROUTINE F07HVF (UPLO, N, KD, NRHS, AB, LDAB, AFB, LDAFB, B, LDB, X,
     1                   LDX, FERR, BERR, WORK, RWORK, INFO)
      ENTRY       cpbrfs (UPLO, N, KD, NRHS, AB, LDAB, AFB, LDAFB, B, LDB, X,
     1                   LDX, FERR, BERR, WORK, RWORK, INFO)
      INTEGER            N, KD, NRHS, LDAB, LDAFB, LDB, LDX, INFO
      real               FERR(*), BERR(*), RWORK(*)
      complex            AB(LDAB,*), AFB(LDAFB,*), B(LDB,*), X(LDX,*), WORK(*)
      CHARACTER*1        UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine returns the backward errors and estimated bounds on the forward errors for the solution of a complex Hermitian positive-definite band system of linear equations with multiple right-hand sides $AX = B$. The routine handles each right-hand side vector (stored as a column of the matrix B) independently, so we describe the function of the routine in terms of a single right-hand side b and solution x.

Given a computed solution x, the routine computes the *component-wise backward error* β. This is the size of the smallest relative perturbation in each element of A and b such that x is the exact solution of a perturbed system

$$(A+\delta A)x = b + \delta b$$

$$|\delta a_{ij}| \leq \beta |a_{ij}| \quad \text{and} \quad |\delta b_i| \leq \beta |b_i|.$$

Then the routine estimates a bound for the *component-wise forward error* in the computed solution, defined by:

$$\max_i |x_i - \hat{x}_i| / \max_i |x_i|$$

where $\hat{x}$ is the true solution.

For details of the method, see the Chapter Introduction.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether the upper or lower triangular part of A is stored and how A has been factorized, as follows:

if UPLO = 'U', then the upper triangular part of A is stored and A is factorized as $U^H U$, where U is upper triangular;

if UPLO = 'L', then the lower triangular part of A is stored and A is factorized as LL^H, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: KD – INTEGER. *Input*

On entry: k, the number of super-diagonals or sub-diagonals of the matrix A.

Constraint: KD ≥ 0.

4: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS ≥ 0.

5: AB(LDAB,*) – ***complex*** array. *Input*

Note: the second dimension of the array AB must be at least max(1,N).

On entry: the n by n original Hermitian positive-definite band matrix A as supplied to F07HRF (CPBTRF/ZPBTRF).

6: LDAB – INTEGER. *Input*

On entry: the first dimension of the array AB as declared in the (sub)program from which F07HVF (CPBRFS/ZPBRFS) is called.

Constraint: LDAB ≥ KD + 1.

7: AFB(LDAFB,*) – ***complex*** array. *Input*

Note: the second dimension of the array AFB must be at least max(1,N).

On entry: the Cholesky factor of A, as returned by F07HRF (CPBTRF/ZPBTRF).

8: LDAFB – INTEGER. *Input*

On entry: the first dimension of the array AFB as declared in the (sub)program from which F07HVF (CPBRFS/ZPBRFS) is called.

Constraint: LDAFB ≥ KD + 1.

9: B(LDB,*) – ***complex*** array. *Input*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

10: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07HVF (CPBRFS/ZPBRFS) is called.

Constraint: LDB ≥ max(1,N).

11: X(LDX,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array X must be at least max(1,NRHS).

On entry: the n by r solution matrix X, as returned by F07HSF (CPBTRS/ZPBTRS).

On exit: the improved solution matrix X.

12: LDX – INTEGER. *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which F07HVF (CPBRFS/ZPBRFS) is called.

Constraint: LDX ≥ max(1,N).

13: FERR(*) – ***real*** array. *Output*

Note: the dimension of the array FERR must be at least max(1,NRHS).

On exit: FERR(j) contains an estimated error bound for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

14: BERR(*) – ***real*** array. *Output*

Note: the dimension of the array BERR must be at least max(1,NRHS).

On exit: BERR(j) contains the component-wise backward error bound β for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

15: WORK(*) – ***complex*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,2*N).

16: RWORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array RWORK must be at least max(1,N).

17: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The bounds returned in FERR are not rigorous, because they are estimated, not computed exactly; but in practice they almost always overestimate the actual error.

8. Further Comments

For each right-hand side, computation of the backward error involves a minimum of $32nk$ real floating-point operations. Each step of iterative refinement involves an additional $48nk$ real operations. This assumes $n \gg k$. At most 5 steps of iterative refinement are performed, but usually only 1 or 2 steps are required.

Estimating the forward error involves solving a number of systems of linear equations of the form $Ax = b$; the number is usually 5 and never more than 11. Each solution involves approximately $16nk$ real operations.

The real analogue of this routine is F07HHF (SPBRFS/DPBRFS).

9. Example

To solve the system of equations $AX = B$ using iterative refinement and to compute the forward and backward error bounds, where

$$A = \begin{pmatrix} 9.39 + 0.00i & 1.08 - 1.73i & 0.00 + 0.00i & 0.00 + 0.00i \\ 1.08 + 1.73i & 1.69 + 0.00i & -0.04 + 0.29i & 0.00 + 0.00i \\ 0.00 + 0.00i & -0.04 - 0.29i & 2.65 + 0.00i & -0.33 + 2.24i \\ 0.00 + 0.00i & 0.00 + 0.00i & -0.33 - 2.24i & 2.17 + 0.00i \end{pmatrix} \quad \text{and}$$

$$B = \begin{pmatrix} -12.42 + 68.42i & 54.30 - 56.56i \\ -9.93 + 0.88i & 18.32 + 4.76i \\ -27.30 - 0.01i & -4.40 + 9.97i \\ 5.31 + 23.63i & 9.43 + 1.41i \end{pmatrix}.$$

Here A is Hermitian positive-definite, and is treated as a band matrix, which must first be factorized by F07HRF (CPBTRF/ZPBTRF).

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07HVF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      complex          ZERO
      PARAMETER        (ZERO=(0.0e0,0.0e0))
      INTEGER          NMAX, NRHMAX, KDMAX, LDAB, LDAFB, LDB, LDX
      PARAMETER        (NMAX=8,NRHMAX=NMAX,KDMAX=8,LDAB=KDMAX+1,
     +                 LDAFB=KDMAX+1,LDB=NMAX,LDX=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, KD, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          AB(LDAB,NMAX), AFB(LDAFB,NMAX), B(LDB,NRHMAX),
     +                 WORK(2*NMAX), X(LDX,NMAX)
      real             BERR(NRHMAX), FERR(NRHMAX), RWORK(NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         cpbrfs, cpbtrf, cpbtrs, F06TFF, F06THF, X04DBF
*     .. Intrinsic Functions ..
      INTRINSIC        MAX, MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07HVF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, KD, NRHS
      IF (N.LE.NMAX .AND. KD.LE.KDMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Set A to zero to avoid referencing uninitialized elements
*
         CALL F06THF('General',KD+1,N,ZERO,ZERO,AB,LDAB)
*
*        Read A and B from data file, and copy A to AFB and B to X
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            DO 20 I = 1, N
               READ (NIN,*) (AB(KD+1+I-J,J),J=I,MIN(N,I+KD))
   20       CONTINUE
```

```
            ELSE IF (UPLO.EQ.'L') THEN
               DO 40 I = 1, N
                  READ (NIN,*) (AB(1+I-J,J),J=MAX(1,I-KD),I)
   40          CONTINUE
            END IF
            READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
            CALL F06TFF('General',KD+1,N,AB,LDAB,AFB,LDAFB)
            CALL F06TFF('General',N,NRHS,B,LDB,X,LDX)
*
*           Factorize A in the array AFB
*
            CALL cpbtrf(UPLO,N,KD,AFB,LDAFB,INFO)
*
            WRITE (NOUT,*)
            IF (INFO.EQ.0) THEN
*
*              Compute solution in the array X
*
               CALL cpbtrs(UPLO,N,KD,NRHS,AFB,LDAFB,X,LDX,INFO)
*
*              Improve solution, and compute backward errors and
*              estimated bounds on the forward errors
*
               CALL cpbrfs(UPLO,N,KD,NRHS,AB,LDAB,AFB,LDAFB,B,LDB,X,LDX,
     +                     FERR,BERR,WORK,RWORK,INFO)
*
*              Print solution
*
               IFAIL = 0
               CALL X04DBF('General',' ',N,NRHS,X,LDX,'Bracketed','F7.4',
     +                     'Solution(s)','Integer',RLABS,'Integer',CLABS,
     +                     80,0,IFAIL)
               WRITE (NOUT,*)
               WRITE (NOUT,*) 'Backward errors (machine-dependent)'
               WRITE (NOUT,99999) (BERR(J),J=1,NRHS)
               WRITE (NOUT,*)
     +           'Estimated forward error bounds (machine-dependent)'
               WRITE (NOUT,99999) (FERR(J),J=1,NRHS)
            ELSE
               WRITE (NOUT,*) 'A is not positive-definite'
            END IF
         END IF
         STOP
*
99999 FORMAT ((5X,1P,4(e11.1,7X)))
         END
```

9.2. Program Data

```
F07HVF Example Program Data
  4  1  2                                                  :Values of N, KD and NRHS
  'L'                                                       :Value of UPLO
 ( 9.39, 0.00)
 ( 1.08, 1.73) ( 1.69, 0.00)
               (-0.04,-0.29) ( 2.65, 0.00)
                             (-0.33,-2.24) ( 2.17, 0.00)   :End of matrix A
 (-12.42,68.42) (54.30,-56.56)
 ( -9.93, 0.88) (18.32,  4.76)
 (-27.30,-0.01) (-4.40,  9.97)
 (  5.31,23.63) ( 9.43,  1.41)                             :End of matrix B
```

9.3. Program Results

```
F07HVF Example Program Results

Solution(s)
                     1                   2
 1  (-1.0000, 8.0000) ( 5.0000,-6.0000)
 2  ( 2.0000,-3.0000) ( 2.0000, 3.0000)
 3  (-4.0000,-5.0000) (-8.0000, 4.0000)
 4  ( 7.0000, 6.0000) (-1.0000,-7.0000)

Backward errors (machine-dependent)
        8.2E-17           4.5E-17
Estimated forward error bounds (machine-dependent)
        3.3E-14           2.8E-14
```

F07MDF (SSYTRF/DSYTRF) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07MDF (SSYTRF/DSYTRF) computes the Bunch-Kaufman factorization of a real symmetric indefinite matrix.

2. Specification

```
SUBROUTINE F07MDF (UPLO, N, A, LDA, IPIV, WORK, LWORK, INFO)
ENTRY      ssytrf  (UPLO, N, A, LDA, IPIV, WORK, LWORK, INFO)
INTEGER          N, LDA, IPIV(*), LWORK, INFO
real             A(LDA,*), WORK(LWORK)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine factorizes a real symmetric matrix A, using the Bunch-Kaufman diagonal pivoting method. A is factorized as either $A = PUDU^TP^T$ if UPLO = 'U', or $A = PLDL^TP^T$ if UPLO = 'L', where P is a permutation matrix, U (or L) is a unit upper (or lower) triangular matrix and D is a symmetric block diagonal matrix with 1 by 1 and 2 by 2 diagonal blocks; U (or L) has 2 by 2 unit diagonal blocks corresponding to the 2 by 2 blocks of D. Row and column interchanges are performed to ensure numerical stability while preserving symmetry.

This method is suitable for symmetric matrices which are not known to be positive-definite. If A is in fact positive-definite, no interchanges are performed and no 2 by 2 blocks occur in D.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §4.4.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether the upper or lower triangular part of A is stored and how A is factorized, as follows:

if UPLO = 'U', then the upper triangular part of A is stored and A is factorized as $PUDU^TP^T$, where U is upper triangular;

if UPLO = 'L', then the lower triangular part of A is stored and A is factorized as $PLDL^TP^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: A(LDA,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the n by n symmetric matrix A. If UPLO = 'U', the upper triangle of A must be stored and the elements of the array below the diagonal are not referenced; if UPLO = 'L', the lower triangle of A must be stored and the elements of the array above the diagonal are not referenced.

On exit: the upper or lower triangle of A is overwritten by details of the block diagonal matrix D and the multipliers used to obtain the factor U or L as specified by UPLO.

4: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07MDF (SSYTRF/DSYTRF) is called.

Constraint: LDA $\geq$ max(1,N).

5: IPIV(*) – INTEGER array. *Output*

Note: the dimension of the array IPIV must be at least max(1,N).

On exit: details of the interchanges and the block structure of D. More precisely, if IPIV(i) = $k > 0$, then d_{ii} is a 1 by 1 pivot block and the ith row and column of A was interchanged with the kth row and column. If UPLO = 'U' and IPIV($i-1$) = IPIV(i) = $-l < 0$, then $\begin{pmatrix} d_{i-1,i-1} & d_{i,i-1} \\ d_{i,i-1} & d_{ii} \end{pmatrix}$ is a 2 by 2 pivot block and the $(i-1)$th row and column of A was interchanged with the lth row and column; if UPLO = 'L' and IPIV(i) = IPIV($i+1$) = $-m < 0$, then $\begin{pmatrix} d_{ii} & d_{i+1,i} \\ d_{i+1,i} & d_{i+1,i+1} \end{pmatrix}$ is a 2 by 2 pivot block and the $(i+1)$th row and column of A was interchanged with the mth row and column.

6: WORK(LWORK) – ***real*** array. *Workspace*

On exit: if INFO = 0, WORK(1) contains the minimum value of LWORK required for optimum performance.

7: LWORK – INTEGER. *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F07MDF (SSYTRF/DSYTRF) is called.

Suggested value: for optimum performance LWORK should be at least N×nb, where nb is the ***blocksize***.

Constraint: LWORK $\geq$ 1.

8: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, d_{ii} is exactly zero. The factorization has been completed but the block diagonal matrix D is exactly singular, and division by zero will occur if it is subsequently used to solve a system of linear equations or to compute A^{-1}.

7. Accuracy

If UPLO = 'U', the computed factors U and D are the exact factors of a perturbed matrix $A + E$, where

$$|E| \leq c(n)\varepsilon P|U||D||U^T|P^T,$$

$c(n)$ is a modest linear function of n, and ε is the ***machine precision***. If UPLO = 'L', a similar statement holds for the computed factors L and D.

8. Further Comments

The elements of D overwrite the corresponding elements of A; if D has 2 by 2 blocks, only the upper or lower triangle is stored, as specified by UPLO.

The unit diagonal elements of U or L and the 2 by 2 unit diagonal blocks are not stored. The remaining elements of U and L are stored in the corresponding columns of the array A, but additional row interchanges must be applied to recover U or L explicitly (this is seldom necessary). If IPIV(i) = i, for i = 1,2,...,n (as is the case when A is positive-definite), then U or L are stored explicitly (except for their unit diagonal elements which are equal to 1).

The total number of floating-point operations is approximately $\frac{1}{3}n^3$.

A call to this routine may be followed by calls to the routines:

F07MEF (SSYTRS/DSYTRS) to solve $AX = B$;
F07MGF (SSYCON/DSYCON) to estimate the condition number of A;
F07MJF (SSYTRI/DSYTRI) to compute the inverse of A.

The complex analogues of this routine are F07MRF (CHETRF/ZHETRF) for Hermitian matrices and F07NRF (CSYTRF/ZSYTRF) for symmetric matrices.

9. Example

To compute the Bunch-Kaufman factorization of the matrix A, where

$$A = \begin{pmatrix} 2.07 & 3.87 & 4.20 & -1.15 \\ 3.87 & -0.21 & 1.87 & 0.63 \\ 4.20 & 1.87 & 1.15 & 2.06 \\ -1.15 & 0.63 & 2.06 & -1.81 \end{pmatrix}.$$

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07MDF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, LWORK
      PARAMETER        (NMAX=8,LDA=NMAX,LWORK=64*NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             A(LDA,NMAX), WORK(LWORK)
      INTEGER          IPIV(NMAX)
*     .. External Subroutines ..
      EXTERNAL         ssytrf, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07MDF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
*
```

```
*        Factorize A
*
         CALL ssytrf(UPLO,N,A,LDA,IPIV,WORK,LWORK,INFO)
*
         WRITE (NOUT,*)
*
*        Print details of factorization
*
         IFAIL = 0
         CALL X04CAF(UPLO,'Nonunit',N,N,A,LDA,
     +               'Details of factorization',IFAIL)
*
*        Print pivot indices
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'IPIV'
         WRITE (NOUT,99999) (IPIV(I),I=1,N)
*
         IF (INFO.NE.0) WRITE (NOUT,*) 'The factor D is singular'
*
      END IF
      STOP
*
99999 FORMAT ((3X,7I11))
      END
```

9.2. Program Data

```
F07MDF Example Program Data
  4                              :Value of N
  'L'                            :Value of UPLO
  2.07
  3.87  -0.21
  4.20   1.87   1.15
 -1.15   0.63   2.06  -1.81     :End of matrix A
```

9.3. Program Results

```
F07MDF Example Program Results

Details of factorization
             1           2           3           4
 1      2.0700
 2      4.2000      1.1500
 3      0.2230      0.8115     -2.5907
 4      0.6537     -0.5960      0.3031      0.4074

IPIV
           -3          -3           3           4
```

F07MEF (SSYTRS/DSYTRS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of *bold italicised* terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07MEF (SSYTRS/DSYTRS) solves a real symmetric indefinite system of linear equations with multiple right-hand sides, $AX = B$, where A has been factorized by F07MDF (SSYTRF/DSYTRF).

2. Specification

```
SUBROUTINE F07MEF (UPLO, N, NRHS, A, LDA, IPIV, B, LDB, INFO)
ENTRY      ssytrs  (UPLO, N, NRHS, A, LDA, IPIV, B, LDB, INFO)
INTEGER          N, NRHS, LDA, IPIV(*), LDB, INFO
real             A(LDA,*), B(LDB,*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

To solve a real symmetric indefinite system of linear equations $AX = B$, this routine must be preceded by a call to F07MDF (SSYTRF/DSYTRF) which computes the Bunch-Kaufman factorization of A.

If UPLO = 'U', $A = PUDU^TP^T$, where P is a permutation matrix, U is an upper triangular matrix and D is a symmetric block diagonal matrix with 1 by 1 and 2 by 2 blocks; the solution X is computed by solving $PUDY = B$ and then $U^TP^TX = Y$.

If UPLO = 'L', $A = PLDL^TP^T$, where L is a lower triangular matrix; the solution X is computed by solving $PLDY = B$ and then $L^TP^TX = Y$.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §4.4.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates how A has been factorized as follows:

if UPLO = 'U', then $A = PUDU^TP^T$, where U is upper triangular;

if UPLO = 'L', then $A = PLDL^TP^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS $\geq$ 0.

4: A(LDA,*) – ***real*** array. *Input*

Note: the second dimension of the array A must be at least max(1,N).

On entry: details of the factorization of A, as returned by F07MDF (SSYTRF/DSYTRF).

5: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07MEF (SSYTRS/DSYTRS) is called.

Constraint: LDA $\geq$ max(1,N).

6: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: details of the interchanges and the block structure of D, as returned by F07MDF (SSYTRF/DSYTRF).

7: B(LDB,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

On exit: the n by r solution matrix X.

8: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07MEF (SSYTRS/DSYTRS) is called.

Constraint: LDB $\geq$ max(1,N).

9: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

For each right-hand side vector b, the computed solution x is the exact solution of a perturbed system of equations $(A+E)x = b$, where

$|E| \leq c(n)\varepsilon P|U||D||U^T|P^T$ if UPLO = 'U',

$|E| \leq c(n)\varepsilon P|L||D||L^T|P^T$ if UPLO = 'L',

$c(n)$ is a modest linear function of n, and ε is the ***machine precision***.

If $\hat{x}$ is the true solution, then the computed solution x satisfies a forward error bound of the form

$$\frac{\|x-\hat{x}\|_\infty}{\|x\|_\infty} \leq c(n)\text{cond}(A,x)\varepsilon$$

where $\text{cond}(A,x) = \||A^{-1}||A||x|\|_\infty / \|x\|_\infty \leq \text{cond}(A) = \||A^{-1}||A|\|_\infty \leq \kappa_\infty(A)$. Note that $\text{cond}(A,x)$ can be much smaller than $\text{cond}(A)$.

Forward and backward error bounds can be computed by calling F07MHF (SSYRFS/DSYRFS), and an estimate for $\kappa_\infty(A)$ $(= \kappa_1(A))$ can be obtained by calling F07MGF (SSYCON/DSYCON).

8. Further Comments

The total number of floating-point operations is approximately $2n^2r$.

This routine may be followed by a call to F07MHF (SSYRFS/DSYRFS) to refine the solution and return an error estimate.

The complex analogues of this routine are F07MSF (CHETRS/ZHETRS) for Hermitian matrices and F07NSF (CSYTRS/ZSYTRS) for symmetric matrices.

9. Example

To solve the system of equations $AX = B$, where

$$A = \begin{pmatrix} 2.07 & 3.87 & 4.20 & -1.15 \\ 3.87 & -0.21 & 1.87 & 0.63 \\ 4.20 & 1.87 & 1.15 & 2.06 \\ -1.15 & 0.63 & 2.06 & -1.81 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} -9.50 & 27.85 \\ -8.38 & 9.90 \\ -6.07 & 19.25 \\ -0.96 & 3.93 \end{pmatrix}.$$

Here A is symmetric indefinite and must first be factorized by F07MDF (SSYTRF/DSYTRF).

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07MEF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, LWORK, NRHMAX, LDB
      PARAMETER        (NMAX=8,LDA=NMAX,LWORK=64*NMAX,NRHMAX=NMAX,
     +                 LDB=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             A(LDA,NMAX), B(LDB,NRHMAX), WORK(LWORK)
      INTEGER          IPIV(NMAX)
*     .. External Subroutines ..
      EXTERNAL         ssytrf, ssytrs, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07MEF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
*
*        Factorize A
*
         CALL ssytrf(UPLO,N,A,LDA,IPIV,WORK,LWORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
```

```
*           Compute solution
*
            CALL ssytrs(UPLO,N,NRHS,A,LDA,IPIV,B,LDB,INFO)
*
*           Print solution
*
            IFAIL = 0
            CALL X04CAF('General',' ',N,NRHS,B,LDB,'Solution(s)',IFAIL)
         ELSE
            WRITE (NOUT,*) 'The factor D is singular'
         END IF
      END IF
      STOP
*
      END
```

9.2. Program Data

```
F07MEF Example Program Data
  4  2                          :Values of N and NRHS
  'L'                           :Value of UPLO
  2.07
  3.87  -0.21
  4.20   1.87   1.15
 -1.15   0.63   2.06  -1.81    :End of matrix A
 -9.50  27.85
 -8.38   9.90
 -6.07  19.25
 -0.96   3.93                  :End of matrix B
```

9.3. Program Results

```
 F07MEF Example Program Results

 Solution(s)
                  1           2
  1       -4.0000      1.0000
  2       -1.0000      4.0000
  3        2.0000      3.0000
  4        5.0000      2.0000
```

F07MGF (SSYCON/DSYCON) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07MGF (SSYCON/DSYCON) estimates the condition number of a real symmetric indefinite matrix A, where A has been factorized by F07MDF (SSYTRF/DSYTRF).

2. Specification

```
 SUBROUTINE F07MGF (UPLO, N, A, LDA, IPIV, ANORM, RCOND, WORK, IWORK,
1                   INFO)
 ENTRY       ssycon (UPLO, N, A, LDA, IPIV, ANORM, RCOND, WORK, IWORK,
1                   INFO)
 INTEGER          N, LDA, IPIV(*), IWORK(*), INFO
 real             A(LDA,*), ANORM, RCOND, WORK(*)
 CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine estimates the condition number (in the 1-norm) of a real symmetric indefinite matrix A:

$$\kappa_1(A) = \|A\|_1 \|A^{-1}\|_1.$$

Since A is symmetric, $\kappa_1(A) = \kappa_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty$.

Because $\kappa_1(A)$ is infinite if A is singular, the routine actually returns an estimate of the **reciprocal** of $\kappa_1(A)$.

The routine should be preceded by a call to F06RCF to compute $\|A\|_1$ and a call to F07MDF (SSYTRF/DSYTRF) to compute the Bunch-Kaufman factorization of A. The routine then uses Higham's implementation of Hager's method [1] to estimate $\|A^{-1}\|_1$.

4. References

[1] HIGHAM, N.J.
FORTRAN Codes for Estimating the One-norm of a Real or Complex Matrix, with Applications to Condition Estimation.
ACM Trans. Math. Softw., 14, pp. 381-396, 1988.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates how A has been factorized as follows:

if UPLO = 'U', then $A = PUDU^TP^T$, where U is upper triangular;

if UPLO = 'L', then $A = PLDL^TP^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: A(LDA,*) – ***real*** array. *Input*

Note: the second dimension of the array A must be at least max(1,N).

On entry: details of the factorization of A, as returned by F07MDF (SSYTRF/DSYTRF).

4: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07MGF (SSYCON/DSYCON) is called.

Constraint: LDA ≥ max(1,N).

5: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: details of the interchanges and the block structure of D, as returned by F07MDF (SSYTRF/DSYTRF).

6: ANORM – ***real***. *Input*

On entry: the 1-norm of the **original** matrix A, which may be computed by calling F06RCF. ANORM must be computed either **before** calling F07MDF (SSYTRF/DSYTRF) or else from a copy of the original matrix A.

Constraint: ANORM ≥ 0.0.

7: RCOND – ***real***. *Output*

On exit: an estimate of the reciprocal of the condition number of A. RCOND is set to zero if exact singularity is detected or the estimate underflows. If RCOND is less than ***machine precision***, then A is singular to working precision.

8: WORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,2*N).

9: IWORK(*) – INTEGER array. *Workspace*

Note: the dimension of the array IWORK must be at least max(1,N).

10: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed estimate RCOND is never less than the true value ρ, and in practice is nearly always less than 10ρ, although examples can be constructed where RCOND is much larger.

8. Further Comments

A call to this routine involves solving a number of systems of linear equations of the form $Ax = b$; the number is usually 4 or 5 and never more than 11. Each solution involves approximately $2n^2$ floating-point operations but takes considerably longer than a call to F07MEF (SSYTRS/DSYTRS) with 1 right-hand side, because extra care is taken to avoid overflow when A is approximately singular.

The complex analogues of this routine are F07MUF (CHECON/ZHECON) for Hermitian matrices and F07NUF (CSYCON/ZSYCON) for symmetric matrices.

9. Example

To estimate the condition number in the 1-norm (or infinity-norm) of the matrix A, where

$$A = \begin{pmatrix} 2.07 & 3.87 & 4.20 & -1.15 \\ 3.87 & -0.21 & 1.87 & 0.63 \\ 4.20 & 1.87 & 1.15 & 2.06 \\ -1.15 & 0.63 & 2.06 & -1.81 \end{pmatrix}.$$

Here A is symmetric indefinite and must first be factorized by F07MDF (SSYTRF/DSYTRF). The true condition number in the 1-norm is 75.68.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07MGF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, LWORK
      PARAMETER        (NMAX=8,LDA=NMAX,LWORK=64*NMAX)
*     .. Local Scalars ..
      real             ANORM, RCOND
      INTEGER          I, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             A(LDA,NMAX), WORK(LWORK)
      INTEGER          IPIV(NMAX), IWORK(NMAX)
*     .. External Functions ..
      real             F06RCF, X02AJF
      EXTERNAL         F06RCF, X02AJF
*     .. External Subroutines ..
      EXTERNAL         ssycon, ssytrf
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07MGF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
*
*        Compute norm of A
*
         ANORM = F06RCF('1-norm',UPLO,N,A,LDA,WORK)
*
*        Factorize A
*
         CALL ssytrf(UPLO,N,A,LDA,IPIV,WORK,LWORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
```

```
*           Estimate condition number
*
            CALL ssycon(UPLO,N,A,LDA,IPIV,ANORM,RCOND,WORK,IWORK,INFO)
*
            IF (RCOND.GE.X02AJF()) THEN
               WRITE (NOUT,99999) 'Estimate of condition number =',
     +           1.0e0/RCOND
            ELSE
               WRITE (NOUT,*) 'A is singular to working precision'
            END IF
         ELSE
            WRITE (NOUT,*) 'The factor D is singular'
         END IF
      END IF
      STOP
*
99999 FORMAT (1X,A,1P,e10.2)
      END
```

9.2. Program Data

```
F07MGF Example Program Data
  4                              :Value of N
  'L'                            :Value of UPLO
  2.07
  3.87  -0.21
  4.20   1.87   1.15
 -1.15   0.63   2.06  -1.81     :End of matrix A
```

9.3. Program Results

```
 F07MGF Example Program Results

 Estimate of condition number =  7.57E+01
```

F07MHF (SSYRFS/DSYRFS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07MHF (SSYRFS/DSYRFS) returns error bounds for the solution of a real symmetric indefinite system of linear equations with multiple right-hand sides, $AX = B$. It improves the solution by iterative refinement, in order to reduce the backward error as much as possible.

2. Specification

```
 SUBROUTINE F07MHF (UPLO, N, NRHS, A, LDA, AF, LDAF, IPIV, B, LDB, X, LDX,
1                   FERR, BERR, WORK, IWORK, INFO)
 ENTRY       ssyrfs  (UPLO, N, NRHS, A, LDA, AF, LDAF, IPIV, B, LDB, X, LDX,
1                   FERR, BERR, WORK, IWORK, INFO)
 INTEGER          N, NRHS, LDA, LDAF, IPIV(*), LDB, LDX, IWORK(*), INFO
 real             A(LDA,*), AF(LDAF,*), B(LDB,*), X(LDX,*), FERR(*),
1                 BERR(*), WORK(*)
 CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine returns the backward errors and estimated bounds on the forward errors for the solution of a real symmetric indefinite system of linear equations with multiple right-hand sides $AX = B$. The routine handles each right-hand side vector (stored as a column of the matrix B) independently, so we describe the function of the routine in terms of a single right-hand side b and solution x.

Given a computed solution x, the routine computes the *component-wise backward error* β. This is the size of the smallest relative perturbation in each element of A and b such that x is the exact solution of a perturbed system

$$(A+\delta A)x = b + \delta b$$

$$|\delta a_{ij}| \leq \beta |a_{ij}| \text{ and } |\delta b_i| \leq \beta |b_i|.$$

Then the routine estimates a bound for the *component-wise forward error* in the computed solution, defined by:

$$\max_i |x_i - \hat{x}_i| / \max_i |x_i|$$

where $\hat{x}$ is the true solution.

For details of the method, see the Chapter Introduction.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether the upper or lower triangular part of A is stored and how A has been factorized, as follows:

if UPLO = 'U', then the upper triangular part of A is stored and A is factorized as $PUDU^TP^T$, where U is upper triangular;

if UPLO = 'L', then the lower triangular part of A is stored and A is factorized as $PLDL^TP^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS ≥ 0.

4: A(LDA,*) – ***real*** array. *Input*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the n by n original symmetric matrix A as supplied to F07MDF (SSYTRF/DSYTRF).

5: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07MHF (SSYRFS/DSYRFS) is called.

Constraint: LDA ≥ max(1,N).

6: AF(LDAF,*) – ***real*** array. *Input*

Note: the second dimension of the array AF must be at least max(1,N).

On entry: details of the factorization of A, as returned by F07MDF (SSYTRF/DSYTRF).

7: LDAF – INTEGER. *Input*

On entry: the first dimension of the array AF as declared in the (sub)program from which F07MHF (SSYRFS/DSYRFS) is called.

Constraint: LDAF ≥ max(1,N).

8: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: details of the interchanges and the block structure of D, as returned by F07MDF (SSYTRF/DSYTRF).

9: B(LDB,*) – ***real*** array. *Input*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

10: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07MHF (SSYRFS/DSYRFS) is called.

Constraint: LDB ≥ max(1,N).

11: X(LDX,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array X must be at least max(1,NRHS).

On entry: the n by r solution matrix X, as returned by F07MEF (SSYTRS/DSYTRS).

On exit: the improved solution matrix X.

12: LDX – INTEGER. *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which F07MHF (SSYRFS/DSYRFS) is called.

Constraint: LDX ≥ max(1,N).

13: FERR(*) – ***real*** array. *Output*

Note: the dimension of the array FERR must be at least max(1,NRHS).

On exit: FERR(j) contains an estimated error bound for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

14: BERR(*) – ***real*** array. *Output*

Note: the dimension of the array BERR must be at least max(1,NRHS).

On exit: BERR(j) contains the component-wise backward error bound β for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

15: WORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,3*N).

16: IWORK(*) – INTEGER array. *Workspace*

Note: the dimension of the array IWORK must be at least max(1,N).

17: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The bounds returned in FERR are not rigorous, because they are estimated, not computed exactly; but in practice they almost always overestimate the actual error.

8. Further Comments

For each right-hand side, computation of the backward error involves a minimum of $4n^2$ floating-point operations. Each step of iterative refinement involves an additional $6n^2$ operations. At most 5 steps of iterative refinement are performed, but usually only 1 or 2 steps are required.

Estimating the forward error involves solving a number of systems of linear equations of the form $Ax = b$; the number is usually 4 or 5 and never more than 11. Each solution involves approximately $2n^2$ operations.

The complex analogues of this routine are F07MVF (CHERFS/ZHERFS) for Hermitian matrices and F07NVF (CSYRFS/ZSYRFS) for symmetric matrices.

9. Example

To solve the system of equations $AX = B$ using iterative refinement and to compute the forward and backward error bounds, where

$$A = \begin{pmatrix} 2.07 & 3.87 & 4.20 & -1.15 \\ 3.87 & -0.21 & 1.87 & 0.63 \\ 4.20 & 1.87 & 1.15 & 2.06 \\ -1.15 & 0.63 & 2.06 & -1.81 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} -9.50 & 27.85 \\ -8.38 & 9.90 \\ -6.07 & 19.25 \\ -0.96 & 3.93 \end{pmatrix}.$$

Here A is symmetric indefinite and must first be factorized by F07MDF (SSYTRF/DSYTRF).

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07MHF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, NRHMAX, LDA, LWORK, LDAF, LDB, LDX
      PARAMETER        (NMAX=8,NRHMAX=NMAX,LDA=NMAX,LWORK=64*NMAX,
     +                 LDAF=NMAX,LDB=NMAX,LDX=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             A(LDA,NMAX), AF(LDAF,NMAX), B(LDB,NRHMAX),
     +                 BERR(NRHMAX), FERR(NRHMAX), WORK(LWORK),
     +                 X(LDX,NMAX)
      INTEGER          IPIV(NMAX), IWORK(NMAX)
*     .. External Subroutines ..
      EXTERNAL         F06QFF, ssyrfs, ssytrf, ssytrs, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07MHF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file, and copy A to AF and B to X
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
*
         CALL F06QFF(UPLO,N,N,A,LDA,AF,LDAF)
*
         CALL F06QFF('General',N,NRHS,B,LDB,X,LDX)
*
*        Factorize A in the array AF
*
         CALL ssytrf(UPLO,N,AF,LDAF,IPIV,WORK,LWORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute solution in the array X
*
            CALL ssytrs(UPLO,N,NRHS,AF,LDAF,IPIV,X,LDX,INFO)
*
```

```
*           Improve solution, and compute backward errors and
*           estimated bounds on the forward errors
*
            CALL ssyrfs(UPLO,N,NRHS,A,LDA,AF,LDAF,IPIV,B,LDB,X,LDX,
     +                  FERR,BERR,WORK,IWORK,INFO)
*
*           Print solution
*
            IFAIL = 0
*
            CALL X04CAF('General',' ',N,NRHS,X,LDX,'Solution(s)',IFAIL)
*
            WRITE (NOUT,*)
            WRITE (NOUT,*) 'Backward errors (machine-dependent)'
            WRITE (NOUT,99999) (BERR(J),J=1,NRHS)
            WRITE (NOUT,*)
     +        'Estimated forward error bounds (machine-dependent)'
            WRITE (NOUT,99999) (FERR(J),J=1,NRHS)
          ELSE
            WRITE (NOUT,*) 'The factor D is singular'
          END IF
        END IF
        STOP
*
99999 FORMAT ((3X,1P,7e11.1))
      END
```

9.2. Program Data

```
F07MHF Example Program Data
  4  2                          :Values of N and NRHS
  'L'                           :Value of UPLO
  2.07
  3.87  -0.21
  4.20   1.87   1.15
 -1.15   0.63   2.06  -1.81    :End of matrix A
 -9.50  27.85
 -8.38   9.90
 -6.07  19.25
 -0.96   3.93                  :End of matrix B
```

9.3. Program Results

```
 F07MHF Example Program Results

 Solution(s)
                  1           2
  1      -4.0000      1.0000
  2      -1.0000      4.0000
  3       2.0000      3.0000
  4       5.0000      2.0000

 Backward errors (machine-dependent)
       5.1E-17    1.2E-16
 Estimated forward error bounds (machine-dependent)
       2.3E-14    3.5E-14
```

F07MJF (SSYTRI/DSYTRI) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07MJF (SSYTRI/DSYTRI) computes the inverse of a real symmetric indefinite matrix A, where A has been factorized by F07MDF (SSYTRF/DSYTRF).

2. Specification

```
SUBROUTINE F07MJF (UPLO, N, A, LDA, IPIV, WORK, INFO)
ENTRY      ssytri  (UPLO, N, A, LDA, IPIV, WORK, INFO)
INTEGER          N, LDA, IPIV(*), INFO
real             A(LDA,*), WORK(*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

To compute the inverse of a real symmetric indefinite matrix A, this routine must be preceded by a call to F07MDF (SSYTRF/DSYTRF), which computes the Bunch-Kaufman factorization of A.

If UPLO = 'U', $A = PUDU^TP^T$ and A^{-1} is computed by solving $U^TP^TXPU = D^{-1}$ for X.

If UPLO = 'L', $A = PLDL^TP^T$ and A^{-1} is computed by solving $L^TP^TXPL = D^{-1}$ for X.

4. References

[1] DU CROZ, J.J. and HIGHAM, N.J.
Stability of Methods for Matrix Inversion.
LAPACK Working Note No. 27, University of Tennessee, Knoxville, 1990.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates how A has been factorized as follows:

if UPLO = 'U', then $A = PUDU^TP^T$, where U is upper triangular;

if UPLO = 'L', then $A = PLDL^TP^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: A(LDA,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array A must be at least max(1,N).

On entry: details of the factorization of A, as returned by F07MDF (SSYTRF/DSYTRF).

On exit: the factorization is overwritten by the n by n symmetric matrix A^{-1}. If UPLO = 'U', the upper triangle of A^{-1} is stored in the upper triangular part of the array; if UPLO = 'L', the lower triangle of A^{-1} is stored in the lower triangular part of the array.

4: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07MJF (SSYTRI/DSYTRI) is called.

Constraint: LDA ≥ max(1,N).

5: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: details of the interchanges and the block structure of D, as returned by F07MDF (SSYTRF/DSYTRF).

6: WORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,N).

7: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, d_{ii} is exactly zero; D is singular and the inverse of A cannot be computed.

7. Accuracy

The computed inverse X satisfies a bound of the form

$|DU^TP^TXPU-I| \leq c(n)\varepsilon(|D||U^T|P^T|X|P|U|+|D||D^{-1}|)$ if UPLO = 'U', or

$|DL^TP^TXPL-I| \leq c(n)\varepsilon(|D||L^T|P^T|X|P|L|+|D||D^{-1}|)$ if UPLO = 'L',

where $c(n)$ is a modest linear function of n, and ε is the ***machine precision.***

8. Further Comments

The total number of floating-point operations is approximately $\frac{2}{3}n^3$.

The complex analogues of this routine are F07MWF (CHETRI/ZHETRI) for Hermitian matrices and F07NWF (CSYTRI/ZSYTRI) for symmetric matrices.

9. Example

To compute the inverse of the matrix A, where

$$A = \begin{pmatrix} 2.07 & 3.87 & 4.20 & -1.15 \\ 3.87 & -0.21 & 1.87 & 0.63 \\ 4.20 & 1.87 & 1.15 & 2.06 \\ -1.15 & 0.63 & 2.06 & -1.81 \end{pmatrix}.$$

Here A is symmetric indefinite and must first be factorized by F07MDF (SSYTRF/DSYTRF).

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07MJF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, LWORK
      PARAMETER        (NMAX=8,LDA=NMAX,LWORK=64*NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             A(LDA,NMAX), WORK(LWORK)
      INTEGER          IPIV(NMAX)
*     .. External Subroutines ..
      EXTERNAL         ssytrf, ssytri, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07MJF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
*
*        Factorize A
*
         CALL ssytrf(UPLO,N,A,LDA,IPIV,WORK,LWORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute inverse of A
*
            CALL ssytri(UPLO,N,A,LDA,IPIV,WORK,INFO)
*
*           Print inverse
*
            IFAIL = 0
            CALL X04CAF(UPLO,'Nonunit',N,N,A,LDA,'Inverse',IFAIL)
         ELSE
            WRITE (NOUT,*) 'The factor D is singular'
         END IF
      END IF
      STOP
*
      END
```

9.2. Program Data

```
F07MJF Example Program Data
  4                              :Value of N
  'L'                            :Value of UPLO
  2.07
  3.87  -0.21
  4.20   1.87   1.15
 -1.15   0.63   2.06  -1.81     :End of matrix A
```

9.3. Program Results

```
F07MJF Example Program Results

Inverse
                1           2           3           4
 1       0.7485
 2       0.5221     -0.1605
 3      -1.0058     -0.3131      1.3501
 4      -1.4386     -0.7440      2.0667      2.4547
```

F07MRF (CHETRF/ZHETRF) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07MRF (CHETRF/ZHETRF) computes the Bunch-Kaufman factorization of a complex Hermitian indefinite matrix.

2. Specification

```
SUBROUTINE F07MRF (UPLO, N, A, LDA, IPIV, WORK, LWORK, INFO)
ENTRY      chetrf  (UPLO, N, A, LDA, IPIV, WORK, LWORK, INFO)
INTEGER          N, LDA, IPIV(*), LWORK, INFO
complex          A(LDA,*), WORK(LWORK)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine factorizes a complex Hermitian matrix A, using the Bunch-Kaufman diagonal pivoting method. A is factorized as either $A = PUDU^HP^T$ if UPLO = 'U', or $A = PLDL^HP^T$ if UPLO = 'L', where P is a permutation matrix, U (or L) is a unit upper (or lower) triangular matrix and D is an Hermitian block diagonal matrix with 1 by 1 and 2 by 2 diagonal blocks; U (or L) has 2 by 2 unit diagonal blocks corresponding to the 2 by 2 blocks of D. Row and column interchanges are performed to ensure numerical stability while keeping the matrix Hermitian.

This method is suitable for Hermitian matrices which are not known to be positive-definite. If A is in fact positive-definite, no interchanges are performed and no 2 by 2 blocks occur in D.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §4.4.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether the upper or lower triangular part of A is stored and how A is factorized, as follows:

if UPLO = 'U', then the upper triangular part of A is stored and A is factorized as $PUDU^HP^T$, where U is upper triangular;

if UPLO = 'L', then the lower triangular part of A is stored and A is factorized as $PLDL^HP^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: A(LDA,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the n by n Hermitian matrix A. If UPLO = 'U', the upper triangle of A must be stored and the elements of the array below the diagonal are not referenced; if UPLO = 'L', the lower triangle of A must be stored and the elements of the array above the diagonal are not referenced.

On exit: the upper or lower triangle of A is overwritten by details of the block diagonal matrix D and the multipliers used to obtain the factor U or L as specified by UPLO.

4: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07MRF (CHETRF/ZHETRF) is called.

Constraint: LDA ≥ max(1,N).

5: IPIV(*) – INTEGER array. *Output*

Note: the dimension of the array IPIV must be at least max(1,N).

On exit: details of the interchanges and the block structure of D. More precisely, if IPIV(i) = $k > 0$, then d_{ii} is a 1 by 1 pivot block and the ith row and column of A was interchanged with the kth row and column. If UPLO = 'U' and IPIV($i-1$) = IPIV(i) = $-l < 0$, then $\begin{pmatrix} d_{i-1,i-1} & d_{i,i-1} \\ \bar{d}_{i,i-1} & d_{ii} \end{pmatrix}$ is a 2 by 2 pivot block and the $(i-1)$th row and column of A was interchanged with the lth row and column; if UPLO = 'L' and IPIV(i) = IPIV($i+1$) = $-m < 0$, then $\begin{pmatrix} d_{ii} & \bar{d}_{i+1,i} \\ d_{i+1,i} & d_{i+1,i+1} \end{pmatrix}$ is a 2 by 2 pivot block and the $(i+1)$th row and column of A was interchanged with the mth row and column.

6: WORK(LWORK) – ***complex*** array. *Workspace*

On exit: if INFO = 0, WORK(1) contains the minimum value of LWORK required for optimum performance.

7: LWORK – INTEGER. *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F07MRF (CHETRF/ZHETRF) is called.

Suggested value: for optimum performance LWORK should be at least N×nb, where nb is the ***blocksize***.

Constraint: LWORK ≥ 1.

8: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, d_{ii} is exactly zero. The factorization has been completed but the block diagonal matrix D is exactly singular, and division by zero will occur if it is subsequently used to solve a system of linear equations or to compute A^{-1}.

7. Accuracy

If UPLO = 'U', the computed factors U and D are the exact factors of a perturbed matrix $A + E$, where

$$|E| \leq c(n)\varepsilon P|U||D||U^H|P^T,$$

$c(n)$ is a modest linear function of n, and ε is the ***machine precision***. If UPLO = 'L', a similar statement holds for the computed factors L and D.

8. Further Comments

The elements of D overwrite the corresponding elements of A; if D has 2 by 2 blocks, only the upper or lower triangle is stored, as specified by UPLO.

The unit diagonal elements of U or L and the 2 by 2 unit diagonal blocks are not stored. The remaining elements of U and L are stored in the corresponding columns of the array A, but additional row interchanges must be applied to recover U or L explicitly (this is seldom necessary). If IPIV(i) = i, for i = 1,2,...,n (as is the case when A is positive-definite), then U or L are stored explicitly (except for their unit diagonal elements which are equal to 1).

The total number of real floating-point operations is approximately $\frac{4}{3}n^3$.

A call to this routine may be followed by calls to the routines:

F07MSF (CHETRS/ZHETRS) to solve $AX = B$;
F07MUF (CHECON/ZHECON) to estimate the condition number of A;
F07MWF (CHETRI/ZHETRI) to compute the inverse of A.

The real analogue of this routine is F07MDF (SSYTRF/DSYTRF).

9. Example

To compute the Bunch-Kaufman factorization of the matrix A, where

$$A = \begin{pmatrix} -1.36 + 0.00i & 1.58 + 0.90i & 2.21 - 0.21i & 3.91 + 1.50i \\ 1.58 - 0.90i & -8.87 + 0.00i & -1.84 - 0.03i & -1.78 + 1.18i \\ 2.21 + 0.21i & -1.84 + 0.03i & -4.63 + 0.00i & 0.11 + 0.11i \\ 3.91 - 1.50i & -1.78 - 1.18i & 0.11 - 0.11i & -1.84 + 0.00i \end{pmatrix}.$$

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07MRF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, LWORK
      PARAMETER        (NMAX=8,LDA=NMAX,LWORK=64*NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          A(LDA,NMAX), WORK(LWORK)
      INTEGER          IPIV(NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         chetrf, X04DBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07MRF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
*
```

```
*        Factorize A
*
         CALL chetrf(UPLO,N,A,LDA,IPIV,WORK,LWORK,INFO)
*
         WRITE (NOUT,*)
*
*        Print details of factorization
*
         IFAIL = 0
         CALL X04DBF(UPLO,'Nonunit',N,N,A,LDA,'Bracketed','F7.4',
     +               'Details of factorization','Integer',RLABS,
     +               'Integer',CLABS,80,0,IFAIL)
*
*        Print pivot indices
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'IPIV'
         WRITE (NOUT,99999) (IPIV(I),I=1,N)
*
         IF (INFO.NE.0) WRITE (NOUT,*) 'The factor D is singular'
*
      END IF
      STOP
*
99999 FORMAT ((1X,I12,3I18))
      END
```

9.2. Program Data

```
F07MRF Example Program Data
  4                                                          :Value of N
  'L'                                                        :Value of UPLO
 (-1.36, 0.00)
 ( 1.58,-0.90) (-8.87, 0.00)
 ( 2.21, 0.21) (-1.84, 0.03) (-4.63, 0.00)
 ( 3.91,-1.50) (-1.78,-1.18) ( 0.11,-0.11) (-1.84, 0.00)   :End of matrix A
```

9.3. Program Results

```
 F07MRF Example Program Results

 Details of factorization
                      1                  2                  3                  4
  1  (-1.3600, 0.0000)
  2  ( 3.9100,-1.5000) (-1.8400, 0.0000)
  3  ( 0.3100, 0.0433) ( 0.5637, 0.2850) (-5.4176, 0.0000)
  4  (-0.1518, 0.3743) ( 0.3397, 0.0303) ( 0.2997, 0.1578) (-7.1028, 0.0000)

 IPIV
           -4                 -4                  3                  4
```

F07MSF (CHETRS/ZHETRS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07MSF (CHETRS/ZHETRS) solves a complex Hermitian indefinite system of linear equations with multiple right-hand sides, $AX = B$, where A has been factorized by F07MRF (CHETRF/ZHETRF).

2. Specification

```
SUBROUTINE F07MSF (UPLO, N, NRHS, A, LDA, IPIV, B, LDB, INFO)
ENTRY      chetrs  (UPLO, N, NRHS, A, LDA, IPIV, B, LDB, INFO)
INTEGER          N, NRHS, LDA, IPIV(*), LDB, INFO
complex          A(LDA,*), B(LDB,*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

To solve a complex Hermitian indefinite system of linear equations $AX = B$, this routine must be preceded by a call to F07MRF (CHETRF/ZHETRF) which computes the Bunch-Kaufman factorization of A.

If UPLO = 'U', $A = PUDU^HP^T$, where P is a permutation matrix, U is an upper triangular matrix and D is an Hermitian block diagonal matrix with 1 by 1 and 2 by 2 blocks; the solution X is computed by solving $PUDY = B$ and then $U^HP^TX = Y$.

If UPLO = 'L', $A = PLDL^HP^T$, where L is a lower triangular matrix; the solution X is computed by solving $PLDY = B$ and then $L^HP^TX = Y$.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §4.4.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates how A has been factorized as follows:

if UPLO = 'U', then $A = PUDU^HP^T$, where U is upper triangular;

if UPLO = 'L', then $A = PLDL^HP^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS $\geq$ 0.

4: A(LDA,*) – ***complex*** array. *Input*

Note: the second dimension of the array A must be at least max(1,N).

On entry: details of the factorization of A, as returned by F07MRF (CHETRF/ZHETRF).

5: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07MSF (CHETRS/ZHETRS) is called.

Constraint: LDA ≥ max(1,N).

6: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: details of the interchanges and the block structure of D, as returned by F07MRF (CHETRF/ZHETRF).

7: B(LDB,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

On exit: the n by r solution matrix X.

8: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07MSF (CHETRS/ZHETRS) is called.

Constraint: LDB ≥ max(1,N).

9: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

For each right-hand side vector b, the computed solution x is the exact solution of a perturbed system of equations $(A+E)x = b$, where

$|E| \leq c(n)\varepsilon P|U||D||U^H|P^T$ if UPLO = 'U',

$|E| \leq c(n)\varepsilon P|L||D||L^H|P^T$ if UPLO = 'L',

$c(n)$ is a modest linear function of n, and ε is the ***machine precision***.

If $\hat{x}$ is the true solution, then the computed solution x satisfies a forward error bound of the form

$$\frac{\|x-\hat{x}\|_\infty}{\|x\|_\infty} \leq c(n)\mathrm{cond}(A,x)\varepsilon$$

where $\mathrm{cond}(A,x) = \||A^{-1}||A||x|\|_\infty / \|x\|_\infty \leq \mathrm{cond}(A) = \||A^{-1}||A|\|_\infty \leq \kappa_\infty(A)$. Note that $\mathrm{cond}(A,x)$ can be much smaller than $\mathrm{cond}(A)$.

Forward and backward error bounds can be computed by calling F07MVF (CHERFS/ZHERFS), and an estimate for $\kappa_\infty(A)$ $(= \kappa_1(A))$ can be obtained by calling F07MUF (CHECON/ZHECON).

8. Further Comments

The total number of real floating-point operations is approximately $8n^2r$.

This routine may be followed by a call to F07MVF (CHERFS/ZHERFS) to refine the solution and return an error estimate.

The real analogue of this routine is F07MEF (SSYTRS/DSYTRS).

9. Example

To solve the system of equations $AX = B$, where

$$A = \begin{pmatrix} -1.36 + 0.00i & 1.58 + 0.90i & 2.21 - 0.21i & 3.91 + 1.50i \\ 1.58 - 0.90i & -8.87 + 0.00i & -1.84 - 0.03i & -1.78 + 1.18i \\ 2.21 + 0.21i & -1.84 + 0.03i & -4.63 + 0.00i & 0.11 + 0.11i \\ 3.91 - 1.50i & -1.78 - 1.18i & 0.11 - 0.11i & -1.84 + 0.00i \end{pmatrix} \quad \text{and}$$

$$B = \begin{pmatrix} 7.79 + 5.48i & -35.39 + 18.01i \\ -0.77 - 16.05i & 4.23 - 70.02i \\ -9.58 + 3.88i & -24.79 - 8.40i \\ 2.98 - 10.18i & 28.68 - 39.89i \end{pmatrix}.$$

Here A is Hermitian indefinite and must first be factorized by F07MRF (CHETRF/ZHETRF).

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07MSF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, LWORK, NRHMAX, LDB
      PARAMETER        (NMAX=8,LDA=NMAX,LWORK=64*NMAX,NRHMAX=NMAX,
     +                 LDB=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          A(LDA,NMAX), B(LDB,NRHMAX), WORK(LWORK)
      INTEGER          IPIV(NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         chetrf, chetrs, X04DBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07MSF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
*
```

```
*        Factorize A
*
         CALL chetrf(UPLO,N,A,LDA,IPIV,WORK,LWORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute solution
*
            CALL chetrs(UPLO,N,NRHS,A,LDA,IPIV,B,LDB,INFO)
*
*           Print solution
*
            IFAIL = 0
            CALL X04DBF('General',' ',N,NRHS,B,LDB,'Bracketed','F7.4',
     +                  'Solution(s)','Integer',RLABS,'Integer',CLABS,
     +                  80,0,IFAIL)
         ELSE
            WRITE (NOUT,*) 'The factor D is singular'
         END IF
      END IF
      STOP
*
      END
```

9.2. Program Data

```
F07MSF Example Program Data
  4  2                                                    :Values of N and NRHS
  'L'                                                        :Value of UPLO
 (-1.36, 0.00)
 ( 1.58,-0.90) (-8.87, 0.00)
 ( 2.21, 0.21) (-1.84, 0.03) (-4.63, 0.00)
 ( 3.91,-1.50) (-1.78,-1.18) ( 0.11,-0.11) (-1.84, 0.00)   :End of matrix A
 ( 7.79,  5.48) (-35.39, 18.01)
 (-0.77,-16.05) (  4.23,-70.02)
 (-9.58,  3.88) (-24.79, -8.40)
 ( 2.98,-10.18) ( 28.68,-39.89)                            :End of matrix B
```

9.3. Program Results

```
F07MSF Example Program Results

Solution(s)
                        1                   2
 1   ( 1.0000,-1.0000) ( 3.0000,-4.0000)
 2   (-1.0000, 2.0000) (-1.0000, 5.0000)
 3   ( 3.0000,-2.0000) ( 7.0000,-2.0000)
 4   ( 2.0000, 1.0000) (-8.0000, 6.0000)
```

F07MUF (CHECON/ZHECON) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07MUF (CHECON/ZHECON) estimates the condition number of a complex Hermitian indefinite matrix A, where A has been factorized by F07MRF (CHETRF/ZHETRF).

2. Specification

```
SUBROUTINE F07MUF (UPLO, N, A, LDA, IPIV, ANORM, RCOND, WORK, INFO)
ENTRY      checon (UPLO, N, A, LDA, IPIV, ANORM, RCOND, WORK, INFO)
INTEGER          N, LDA, IPIV(*), INFO
real             ANORM, RCOND
complex          A(LDA,*), WORK(*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine estimates the condition number (in the 1-norm) of a complex Hermitian indefinite matrix A:

$$\kappa_1(A) = \|A\|_1 \|A^{-1}\|_1.$$

Since A is Hermitian, $\kappa_1(A) = \kappa_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty$.

Because $\kappa_1(A)$ is infinite if A is singular, the routine actually returns an estimate of the **reciprocal** of $\kappa_1(A)$.

The routine should be preceded by a call to F06UCF to compute $\|A\|_1$ and a call to F07MRF (CHETRF/ZHETRF) to compute the Bunch-Kaufman factorization of A. The routine then uses Higham's implementation of Hager's method [1] to estimate $\|A^{-1}\|_1$.

4. References

[1] HIGHAM, N.J.
FORTRAN Codes for Estimating the One-norm of a Real or Complex Matrix, with Applications to Condition Estimation.
ACM Trans. Math. Softw., 14, pp. 381-396, 1988.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates how A has been factorized as follows:

if UPLO = 'U', then $A = PUDU^HP^T$, where U is upper triangular;

if UPLO = 'L', then $A = PLDL^HP^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: A(LDA,*) – ***complex*** array. *Input*

Note: the second dimension of the array A must be at least max(1,N).

On entry: details of the factorization of A, as returned by F07MRF (CHETRF/ZHETRF).

4: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07MUF (CHECON/ZHECON) is called.

Constraint: LDA ≥ max(1,N).

5: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: details of the interchanges and the block structure of D, as returned by F07MRF (CHETRF/ZHETRF).

6: ANORM – ***real***. *Input*

On entry: the 1-norm of the **original** matrix A, which may be computed by calling F06UCF. ANORM must be computed either **before** calling F07MRF (CHETRF/ZHETRF) or else from a copy of the original matrix A.

Constraint: ANORM ≥ 0.0.

7: RCOND – ***real***. *Output*

On exit: an estimate of the reciprocal of the condition number of A. RCOND is set to zero if exact singularity is detected or the estimate underflows. If RCOND is less than ***machine precision***, then A is singular to working precision.

8: WORK(*) – ***complex*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,2*N).

9: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed estimate RCOND is never less than the true value ρ, and in practice is nearly always less than 10ρ, although examples can be constructed where RCOND is much larger.

8. Further Comments

A call to this routine involves solving a number of systems of linear equations of the form $Ax = b$; the number is usually 5 and never more than 11. Each solution involves approximately $8n^2$ real floating-point operations but takes considerably longer than a call to F07MSF (CHETRS/ZHETRS) with 1 right-hand side, because extra care is taken to avoid overflow when A is approximately singular.

The real analogue of this routine is F07MGF (SSYCON/DSYCON).

9. Example

To estimate the condition number in the 1-norm (or infinity-norm) of the matrix A, where

$$A = \begin{pmatrix} -1.36 + 0.00i & 1.58 + 0.90i & 2.21 - 0.21i & 3.91 + 1.50i \\ 1.58 - 0.90i & -8.87 + 0.00i & -1.84 - 0.03i & -1.78 + 1.18i \\ 2.21 + 0.21i & -1.84 + 0.03i & -4.63 + 0.00i & 0.11 + 0.11i \\ 3.91 - 1.50i & -1.78 - 1.18i & 0.11 - 0.11i & -1.84 + 0.00i \end{pmatrix}.$$

Here A is Hermitian indefinite and must first be factorized by F07MRF (CHETRF/ZHETRF). The true condition number in the 1-norm is 9.10.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07MUF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, LWORK
      PARAMETER        (NMAX=8,LDA=NMAX,LWORK=64*NMAX)
*     .. Local Scalars ..
      real             ANORM, RCOND
      INTEGER          I, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          A(LDA,NMAX), WORK(LWORK)
      real             RWORK(NMAX)
      INTEGER          IPIV(NMAX)
*     .. External Functions ..
      real             F06UCF, X02AJF
      EXTERNAL         F06UCF, X02AJF
*     .. External Subroutines ..
      EXTERNAL         checon, chetrf
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07MUF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
*
*        Compute norm of A
*
         ANORM = F06UCF('1-norm',UPLO,N,A,LDA,RWORK)
*
*        Factorize A
*
         CALL chetrf(UPLO,N,A,LDA,IPIV,WORK,LWORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Estimate condition number
*
            CALL checon(UPLO,N,A,LDA,IPIV,ANORM,RCOND,WORK,INFO)
*
            IF (RCOND.GE.X02AJF()) THEN
               WRITE (NOUT,99999) 'Estimate of condition number =',
     +           1.0e0/RCOND
            ELSE
               WRITE (NOUT,*) 'A is singular to working precision'
            END IF
         ELSE
```

```
            WRITE (NOUT,*) 'The factor D is singular'
         END IF
      END IF
      STOP
*
99999 FORMAT (1X,A,1P,e10.2)
      END
```

9.2. Program Data

```
F07MUF Example Program Data
  4                                                                  :Value of N
  'L'                                                                :Value of UPLO
 (-1.36, 0.00)
 ( 1.58,-0.90) (-8.87, 0.00)
 ( 2.21, 0.21) (-1.84, 0.03) (-4.63, 0.00)
 ( 3.91,-1.50) (-1.78,-1.18) ( 0.11,-0.11) (-1.84, 0.00)  :End of matrix A
```

9.3. Program Results

```
F07MUF Example Program Results

Estimate of condition number =  6.68E+00
```

F07MVF (CHERFS/ZHERFS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07MVF (CHERFS/ZHERFS) returns error bounds for the solution of a complex Hermitian indefinite system of linear equations with multiple right-hand sides, $AX = B$. It improves the solution by iterative refinement, in order to reduce the backward error as much as possible.

2. Specification

```
SUBROUTINE F07MVF (UPLO, N, NRHS, A, LDA, AF, LDAF, IPIV, B, LDB, X, LDX,
1                  FERR, BERR, WORK, RWORK, INFO)
ENTRY      cherfs  (UPLO, N, NRHS, A, LDA, AF, LDAF, IPIV, B, LDB, X, LDX,
1                  FERR, BERR, WORK, RWORK, INFO)
INTEGER            N, NRHS, LDA, LDAF, IPIV(*), LDB, LDX, INFO
real               FERR(*), BERR(*), RWORK(*)
complex            A(LDA,*), AF(LDAF,*), B(LDB,*), X(LDX,*), WORK(*)
CHARACTER*1        UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine returns the backward errors and estimated bounds on the forward errors for the solution of a complex Hermitian indefinite system of linear equations with multiple right-hand sides $AX = B$. The routine handles each right-hand side vector (stored as a column of the matrix B) independently, so we describe the function of the routine in terms of a single right-hand side b and solution x.

Given a computed solution x, the routine computes the *component-wise backward error* β. This is the size of the smallest relative perturbation in each element of A and b such that x is the exact solution of a perturbed system

$$(A+\delta A)x = b + \delta b$$

$$|\delta a_{ij}| \leq \beta |a_{ij}| \quad \text{and} \quad |\delta b_i| \leq \beta |b_i|.$$

Then the routine estimates a bound for the *component-wise forward error* in the computed solution, defined by:

$$\max_i |x_i - \hat{x}_i| / \max_i |x_i|$$

where $\hat{x}$ is the true solution.

For details of the method, see the Chapter Introduction.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether the upper or lower triangular part of A is stored and how A has been factorized, as follows:

if UPLO = 'U', then the upper triangular part of A is stored and A is factorized as $PUDU^HP^T$, where U is upper triangular;

if UPLO = 'L', then the lower triangular part of A is stored and A is factorized as $PLDL^HP^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS ≥ 0.

4: A(LDA,*) – ***complex*** array. *Input*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the n by n original Hermitian matrix A as supplied to F07MRF (CHETRF/ZHETRF).

5: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07MVF (CHERFS/ZHERFS) is called.

Constraint: LDA ≥ max(1,N).

6: AF(LDAF,*) – ***complex*** array. *Input*

Note: the second dimension of the array AF must be at least max(1,N).

On entry: details of the factorization of A, as returned by F07MRF (CHETRF/ZHETRF).

7: LDAF – INTEGER. *Input*

On entry: the first dimension of the array AF as declared in the (sub)program from which F07MVF (CHERFS/ZHERFS) is called.

Constraint: LDAF ≥ max(1,N).

8: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: details of the interchanges and the block structure of D, as returned by F07MRF (CHETRF/ZHETRF).

9: B(LDB,*) – ***complex*** array. *Input*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

10: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07MVF (CHERFS/ZHERFS) is called.

Constraint: LDB ≥ max(1,N).

11: X(LDX,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array X must be at least max(1,NRHS).

On entry: the n by r solution matrix X, as returned by F07MSF (CHETRS/ZHETRS).

On exit: the improved solution matrix X.

12: LDX – INTEGER. *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which F07MVF (CHERFS/ZHERFS) is called.

Constraint: LDX ≥ max(1,N).

13: FERR(*) – ***real*** array. *Output*

Note: the dimension of the array FERR must be at least max(1,NRHS).

On exit: FERR(j) contains an estimated error bound for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

14: BERR(*) – ***real*** array. *Output*

Note: the dimension of the array BERR must be at least max(1,NRHS).

On exit: BERR(j) contains the component-wise backward error bound β for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

15: WORK(*) – ***complex*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,2*N).

16: RWORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array RWORK must be at least max(1,N).

17: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The bounds returned in FERR are not rigorous, because they are estimated, not computed exactly; but in practice they almost always overestimate the actual error.

8. Further Comments

For each right-hand side, computation of the backward error involves a minimum of $16n^2$ real floating-point operations. Each step of iterative refinement involves an additional $24n^2$ real operations. At most 5 steps of iterative refinement are performed, but usually only 1 or 2 steps are required.

Estimating the forward error involves solving a number of systems of linear equations of the form $Ax = b$; the number is usually 5 and never more than 11. Each solution involves approximately $8n^2$ real operations.

The real analogue of this routine is F07MHF (SSYRFS/DSYRFS).

9. Example

To solve the system of equations $AX = B$ using iterative refinement and to compute the forward and backward error bounds, where

$$A = \begin{pmatrix} -1.36 + 0.00i & 1.58 + 0.90i & 2.21 - 0.21i & 3.91 + 1.50i \\ 1.58 - 0.90i & -8.87 + 0.00i & -1.84 - 0.03i & -1.78 + 1.18i \\ 2.21 + 0.21i & -1.84 + 0.03i & -4.63 + 0.00i & 0.11 + 0.11i \\ 3.91 - 1.50i & -1.78 - 1.18i & 0.11 - 0.11i & -1.84 + 0.00i \end{pmatrix} \text{ and}$$

$$B = \begin{pmatrix} 7.79 + 5.48i & -35.39 + 18.01i \\ -0.77 - 16.05i & 4.23 - 70.02i \\ -9.58 + 3.88i & -24.79 - 8.40i \\ 2.98 - 10.18i & 28.68 - 39.89i \end{pmatrix}.$$

Here A is Hermitian indefinite and must first be factorized by F07MRF (CHETRF/ZHETRF).

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07MVF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, NRHMAX, LDA, LWORK, LDAF, LDB, LDX
      PARAMETER        (NMAX=8,NRHMAX=NMAX,LDA=NMAX,LWORK=64*NMAX,
     +                 LDAF=NMAX,LDB=NMAX,LDX=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          A(LDA,NMAX), AF(LDAF,NMAX), B(LDB,NRHMAX),
     +                 WORK(LWORK), X(LDX,NMAX)
      real             BERR(NRHMAX), FERR(NRHMAX), RWORK(NMAX)
      INTEGER          IPIV(NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         cherfs, chetrf, chetrs, F06TFF, X04DBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07MVF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file, and copy A to AF and B to X
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
         CALL F06TFF(UPLO,N,N,A,LDA,AF,LDAF)
         CALL F06TFF('General',N,NRHS,B,LDB,X,LDX)
*
*        Factorize A in the array AF
*
         CALL chetrf(UPLO,N,AF,LDAF,IPIV,WORK,LWORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
```

```
*           Compute solution in the array X
*
            CALL chetrs(UPLO,N,NRHS,AF,LDAF,IPIV,X,LDX,INFO)
*
*           Improve solution, and compute backward errors and
*           estimated bounds on the forward errors
*
            CALL cherfs(UPLO,N,NRHS,A,LDA,AF,LDAF,IPIV,B,LDB,X,LDX,
     +                  FERR,BERR,WORK,RWORK,INFO)
*
*           Print solution
*
            IFAIL = 0
            CALL X04DBF('General',' ',N,NRHS,X,LDX,'Bracketed','F7.4',
     +                  'Solution(s)','Integer',RLABS,'Integer',CLABS,
     +                  80,0,IFAIL)
            WRITE (NOUT,*)
            WRITE (NOUT,*) 'Backward errors (machine-dependent)'
            WRITE (NOUT,99999) (BERR(J),J=1,NRHS)
            WRITE (NOUT,*)
     +        'Estimated forward error bounds (machine-dependent)'
            WRITE (NOUT,99999) (FERR(J),J=1,NRHS)
          ELSE
            WRITE (NOUT,*) 'The factor D is singular'
          END IF
        END IF
        STOP
*
99999 FORMAT ((5X,1P,4(e11.1,7X)))
        END
```

9.2. Program Data

```
F07MVF Example Program Data
  4  2                                                    :Values of N and NRHS
  'L'                                                          :Value of UPLO
 (-1.36, 0.00)
 ( 1.58,-0.90) (-8.87, 0.00)
 ( 2.21, 0.21) (-1.84, 0.03) (-4.63, 0.00)
 ( 3.91,-1.50) (-1.78,-1.18) ( 0.11,-0.11) (-1.84, 0.00)   :End of matrix A
 ( 7.79,  5.48) (-35.39, 18.01)
 (-0.77,-16.05) (  4.23,-70.02)
 (-9.58,  3.88) (-24.79, -8.40)
 ( 2.98,-10.18) ( 28.68,-39.89)                            :End of matrix B
```

9.3. Program Results

```
 F07MVF Example Program Results

 Solution(s)
                      1                     2
  1  ( 1.0000,-1.0000) ( 3.0000,-4.0000)
  2  (-1.0000, 2.0000) (-1.0000, 5.0000)
  3  ( 3.0000,-2.0000) ( 7.0000,-2.0000)
  4  ( 2.0000, 1.0000) (-8.0000, 6.0000)

 Backward errors (machine-dependent)
         8.0E-17           8.9E-17
 Estimated forward error bounds (machine-dependent)
         2.6E-15           3.0E-15
```

F07MWF (CHETRI/ZHETRI) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07MWF (CHETRI/ZHETRI) computes the inverse of a complex Hermitian indefinite matrix A, where A has been factorized by F07MRF (CHETRF/ZHETRF).

2. Specification

```
SUBROUTINE F07MWF (UPLO, N, A, LDA, IPIV, WORK, INFO)
ENTRY      chetri  (UPLO, N, A, LDA, IPIV, WORK, INFO)
INTEGER          N, LDA, IPIV(*), INFO
complex          A(LDA,*), WORK(*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

To compute the inverse of a complex Hermitian indefinite matrix A, this routine must be preceded by a call to F07MRF (CHETRF/ZHETRF), which computes the Bunch-Kaufman factorization of A.

If UPLO = 'U', $A = PUDU^HP^T$ and A^{-1} is computed by solving $U^HP^TXPU = D^{-1}$ for X.

If UPLO = 'L', $A = PLDL^HP^T$ and A^{-1} is computed by solving $L^HP^TXPL = D^{-1}$ for X.

4. References

[1] DU CROZ, J.J. and HIGHAM, N.J.
Stability of Methods for Matrix Inversion.
LAPACK Working Note No. 27, University of Tennessee, Knoxville, 1990.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates how A has been factorized as follows:

if UPLO = 'U', then $A = PUDU^HP^T$, where U is upper triangular;

if UPLO = 'L', then $A = PLDL^HP^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: A(LDA,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array A must be at least max(1,N).

On entry: details of the factorization of A, as returned by F07MRF (CHETRF/ZHETRF).

On exit: the factorization is overwritten by the n by n Hermitian matrix A^{-1}. If UPLO = 'U', the upper triangle of A^{-1} is stored in the upper triangular part of the array; if UPLO = 'L', the lower triangle of A^{-1} is stored in the lower triangular part of the array.

4: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07MWF (CHETRI/ZHETRI) is called.

Constraint: LDA ≥ max(1,N).

5: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: details of the interchanges and the block structure of D, as returned by F07MRF (CHETRF/ZHETRF).

6: WORK(*) – ***complex*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,N).

7: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, d_{ii} is exactly zero; D is singular and the inverse of A cannot be computed.

7. Accuracy

The computed inverse X satisfies a bound of the form

$|DU^HP^TXPU-I| \leq c(n)\varepsilon(|D||U^H|P^T|X|P|U|+|D||D^{-1}|)$ if UPLO = 'U', or

$|DL^HP^TXPL-I| \leq c(n)\varepsilon(|D||L^H|P^T|X|P|L|+|D||D^{-1}|)$ if UPLO = 'L',

where $c(n)$ is a modest linear function of n, and ε is the ***machine precision***.

8. Further Comments

The total number of real floating-point operations is approximately $\frac{8}{3}n^3$.

The real analogue of this routine is F07MJF (SSYTRI/DSYTRI).

9. Example

To compute the inverse of the matrix A, where

$$A = \begin{pmatrix} -1.36 + 0.00i & 1.58 + 0.90i & 2.21 - 0.21i & 3.91 + 1.50i \\ 1.58 - 0.90i & -8.87 + 0.00i & -1.84 - 0.03i & -1.78 + 1.18i \\ 2.21 + 0.21i & -1.84 + 0.03i & -4.63 + 0.00i & 0.11 + 0.11i \\ 3.91 - 1.50i & -1.78 - 1.18i & 0.11 - 0.11i & -1.84 + 0.00i \end{pmatrix}.$$

Here A is Hermitian indefinite and must first be factorized by F07MRF (CHETRF/ZHETRF).

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07MWF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, LWORK
      PARAMETER        (NMAX=8,LDA=NMAX,LWORK=64*NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          A(LDA,NMAX), WORK(LWORK)
      INTEGER          IPIV(NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         chetrf, chetri, X04DBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07MWF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
*
*        Factorize A
*
         CALL chetrf(UPLO,N,A,LDA,IPIV,WORK,LWORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute inverse of A
*
            CALL chetri(UPLO,N,A,LDA,IPIV,WORK,INFO)
*
*           Print inverse
*
            IFAIL = 0
            CALL X04DBF(UPLO,'Nonunit',N,N,A,LDA,'Bracketed','F7.4',
     +                  'Inverse','Integer',RLABS,'Integer',CLABS,80,0,
     +                  IFAIL)
         ELSE
            WRITE (NOUT,*) 'The factor D is singular'
         END IF
      END IF
      STOP
*
      END
```

9.2. Program Data

```
F07MWF Example Program Data
  4                                                          :Value of N
  'L'                                                        :Value of UPLO
 (-1.36, 0.00)
 ( 1.58,-0.90) (-8.87, 0.00)
 ( 2.21, 0.21) (-1.84, 0.03) (-4.63, 0.00)
 ( 3.91,-1.50) (-1.78,-1.18) ( 0.11,-0.11) (-1.84, 0.00)   :End of matrix A
```

9.3. Program Results

```
F07MWF Example Program Results

Inverse
                      1                  2                  3                  4
 1  ( 0.0826, 0.0000)
 2  (-0.0335, 0.0440) (-0.1408, 0.0000)
 3  ( 0.0603,-0.0105) ( 0.0422,-0.0222) (-0.2007, 0.0000)
 4  ( 0.2391,-0.0926) ( 0.0304, 0.0203) ( 0.0982,-0.0635) ( 0.0073, 0.0000)
```

F07NRF (CSYTRF/ZSYTRF) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07NRF (CSYTRF/ZSYTRF) computes the Bunch-Kaufman factorization of a complex symmetric matrix.

2. Specification

```
SUBROUTINE F07NRF (UPLO, N, A, LDA, IPIV, WORK, LWORK, INFO)
ENTRY      csytrf  (UPLO, N, A, LDA, IPIV, WORK, LWORK, INFO)
INTEGER          N, LDA, IPIV(*), LWORK, INFO
complex          A(LDA,*), WORK(LWORK)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine factorizes a complex symmetric matrix A, using the Bunch-Kaufman diagonal pivoting method. A is factorized as either $A = PUDU^TP^T$ if UPLO = 'U', or $A = PLDL^TP^T$ if UPLO = 'L', where P is a permutation matrix, U (or L) is a unit upper (or lower) triangular matrix and D is a symmetric block diagonal matrix with 1 by 1 and 2 by 2 diagonal blocks; U (or L) has 2 by 2 unit diagonal blocks corresponding to the 2 by 2 blocks of D. Row and column interchanges are performed to ensure numerical stability while preserving symmetry.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §4.4.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether the upper or lower triangular part of A is stored and how A is factorized, as follows:

if UPLO = 'U', then the upper triangular part of A is stored and A is factorized as $PUDU^TP^T$, where U is upper triangular;

if UPLO = 'L', then the lower triangular part of A is stored and A is factorized as $PLDL^TP^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: A(LDA,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the n by n symmetric matrix A. If UPLO = 'U', the upper triangle of A must be stored and the elements of the array below the diagonal are not referenced; if UPLO = 'L', the lower triangle of A must be stored and the elements of the array above the diagonal are not referenced.

On exit: the upper or lower triangle of A is overwritten by details of the block diagonal matrix D and the multipliers used to obtain the factor U or L as specified by UPLO.

4: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07NRF (CSYTRF/ZSYTRF) is called.

Constraint: LDA ≥ max(1,N).

5: IPIV(*) – INTEGER array. *Output*

Note: the dimension of the array IPIV must be at least max(1,N).

On exit: details of the interchanges and the block structure of D. More precisely, if IPIV(i) = $k > 0$, then d_{ii} is a 1 by 1 pivot block and the ith row and column of A was interchanged with the kth row and column. If UPLO = 'U' and IPIV($i-1$) = IPIV(i) = $-l < 0$, then $\begin{pmatrix} d_{i-1,i-1} & d_{i,i-1} \\ d_{i,i-1} & d_{ii} \end{pmatrix}$ is a 2 by 2 pivot block and the $(i-1)$th row and column of A was interchanged with the lth row and column; if UPLO = 'L' and IPIV(i) = IPIV($i+1$) = $-m < 0$, then $\begin{pmatrix} d_{ii} & d_{i+1,i} \\ d_{i+1,i} & d_{i+1,i+1} \end{pmatrix}$ is a 2 by 2 pivot block and the $(i+1)$th row and column of A was interchanged with the mth row and column.

6: WORK(LWORK) – ***complex*** array. *Workspace*

On exit: if INFO = 0, WORK(1) contains the minimum value of LWORK required for optimum performance.

7: LWORK – INTEGER. *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F07NRF (CSYTRF/ZSYTRF) is called.

Suggested value: for optimum performance LWORK should be at least N×*nb*, where *nb* is the ***blocksize***.

Constraint: LWORK ≥ 1.

8: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, d_{ii} is exactly zero. The factorization has been completed but the block diagonal matrix D is exactly singular, and division by zero will occur if it is subsequently used to solve a system of linear equations or to compute A^{-1}.

7. Accuracy

If UPLO = 'U', the computed factors U and D are the exact factors of a perturbed matrix $A + E$, where

$$|E| \leq c(n)\varepsilon P|U||D||U^T|P^T,$$

$c(n)$ is a modest linear function of n, and ε is the ***machine precision***. If UPLO = 'L', a similar statement holds for the computed factors L and D.

8. Further Comments

The elements of D overwrite the corresponding elements of A; if D has 2 by 2 blocks, only the upper or lower triangle is stored, as specified by UPLO.

The unit diagonal elements of U or L and the 2 by 2 unit diagonal blocks are not stored. The remaining elements of U and L are stored in the corresponding columns of the array A, but additional row interchanges must be applied to recover U or L explicitly (this is seldom necessary). If IPIV(i) = i, for i = 1,2,...,n, then U or L are stored explicitly (except for their unit diagonal elements which are equal to 1).

The total number of real floating-point operations is approximately $\frac{4}{3}n^3$.

A call to this routine may be followed by calls to the routines:

F07NSF (CSYTRS/ZSYTRS) to solve $AX = B$;
F07NUF (CSYCON/ZSYCON) to estimate the condition number of A;
F07NWF (CSYTRI/ZSYTRI) to compute the inverse of A.

The real analogue of this routine is F07MDF (SSYTRF/DSYTRF).

9. Example

To compute the Bunch-Kaufman factorization of the matrix A, where

$$A = \begin{pmatrix} -0.39 - 0.71i & 5.14 - 0.64i & -7.86 - 2.96i & 3.80 + 0.92i \\ 5.14 - 0.64i & 8.86 + 1.81i & -3.52 + 0.58i & 5.32 - 1.59i \\ -7.86 - 2.96i & -3.52 + 0.58i & -2.83 - 0.03i & -1.54 - 2.86i \\ 3.80 + 0.92i & 5.32 - 1.59i & -1.54 - 2.86i & -0.56 + 0.12i \end{pmatrix}.$$

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07NRF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, LWORK
      PARAMETER        (NMAX=8,LDA=NMAX,LWORK=64*NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          A(LDA,NMAX), WORK(LWORK)
      INTEGER          IPIV(NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         csytrf, X04DBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07NRF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
*
```

```
*        Factorize A
*
         CALL csytrf(UPLO,N,A,LDA,IPIV,WORK,LWORK,INFO)
*
         WRITE (NOUT,*)
*
*        Print details of factorization
*
         IFAIL = 0
         CALL X04DBF(UPLO,'Nonunit',N,N,A,LDA,'Bracketed','F7.4',
     +               'Details of factorization','Integer',RLABS,
     +               'Integer',CLABS,80,0,IFAIL)
*
*        Print pivot indices
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'IPIV'
         WRITE (NOUT,99999) (IPIV(I),I=1,N)
*
         IF (INFO.NE.0) WRITE (NOUT,*) 'The factor D is singular'
*
      END IF
      STOP
*
99999 FORMAT ((1X,I12,3I18))
      END
```

9.2. Program Data

```
F07NRF Example Program Data
  4                                                          :Value of N
  'L'                                                        :Value of UPLO
 (-0.39,-0.71)
 ( 5.14,-0.64) ( 8.86, 1.81)
 (-7.86,-2.96) (-3.52, 0.58) (-2.83,-0.03)
 ( 3.80, 0.92) ( 5.32,-1.59) (-1.54,-2.86) (-0.56, 0.12)   :End of matrix A
```

9.3. Program Results

```
 F07NRF Example Program Results

 Details of factorization
                      1                   2                   3                   4
  1  (-0.3900,-0.7100)
  2  (-7.8600,-2.9600) (-2.8300,-0.0300)
  3  ( 0.5279,-0.3715) (-0.6078, 0.2811) ( 4.4079, 5.3991)
  4  ( 0.4426, 0.1936) (-0.4823, 0.0150) (-0.1071,-0.3157) (-2.0954,-2.2011)

 IPIV
           -3                  -3                   3                   4
```

F07NSF (CSYTRS/ZSYTRS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07NSF (CSYTRS/ZSYTRS) solves a complex symmetric system of linear equations with multiple right-hand sides, $AX = B$, where A has been factorized by F07NRF (CSYTRF/ZSYTRF).

2. Specification

```
SUBROUTINE F07NSF (UPLO, N, NRHS, A, LDA, IPIV, B, LDB, INFO)
ENTRY      csytrs  (UPLO, N, NRHS, A, LDA, IPIV, B, LDB, INFO)
INTEGER          N, NRHS, LDA, IPIV(*), LDB, INFO
complex          A(LDA,*), B(LDB,*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

To solve a complex symmetric system of linear equations $AX = B$, this routine must be preceded by a call to F07NRF (CSYTRF/ZSYTRF) which computes the Bunch-Kaufman factorization of A.

If UPLO = 'U', $A = PUDU^TP^T$, where P is a permutation matrix, U is an upper triangular matrix and D is a symmetric block diagonal matrix with 1 by 1 and 2 by 2 blocks; the solution X is computed by solving $PUDY = B$ and then $U^TP^TX = Y$.

If UPLO = 'L', $A = PLDL^TP^T$, where L is a lower triangular matrix; the solution X is computed by solving $PLDY = B$ and then $L^TP^TX = Y$.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §4.4.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates how A has been factorized as follows:

if UPLO = 'U', then $A = PUDU^TP^T$, where U is upper triangular;

if UPLO = 'L', then $A = PLDL^TP^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS $\geq$ 0.

4: A(LDA,*) – ***complex*** array. *Input*

Note: the second dimension of the array A must be at least max(1,N).

On entry: details of the factorization of A, as returned by F07NRF (CSYTRF/ZSYTRF).

5: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07NSF (CSYTRS/ZSYTRS) is called.

Constraint: LDA ≥ max(1,N).

6: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: details of the interchanges and the block structure of D, as returned by F07NRF (CSYTRF/ZSYTRF).

7: B(LDB,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

On exit: the n by r solution matrix X.

8: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07NSF (CSYTRS/ZSYTRS) is called.

Constraint: LDB ≥ max(1,N).

9: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

For each right-hand side vector b, the computed solution x is the exact solution of a perturbed system of equations $(A+E)x = b$, where

$|E| \leq c(n)\varepsilon P|U||D||U^T|P^T$ if UPLO = 'U',

$|E| \leq c(n)\varepsilon P|L||D||L^T|P^T$ if UPLO = 'L',

$c(n)$ is a modest linear function of n, and ε is the ***machine precision***.

If $\hat{x}$ is the true solution, then the computed solution x satisfies a forward error bound of the form

$$\frac{\|x-\hat{x}\|_\infty}{\|x\|_\infty} \leq c(n)\text{cond}(A,x)\varepsilon$$

where $\text{cond}(A,x) = \||A^{-1}||A||x|\|_\infty / \|x\|_\infty \leq \text{cond}(A) = \||A^{-1}||A|\|_\infty \leq \kappa_\infty(A)$. Note that $\text{cond}(A,x)$ can be much smaller than $\text{cond}(A)$.

Forward and backward error bounds can be computed by calling F07NVF (CSYRFS/ZSYRFS), and an estimate for $\kappa_\infty(A)$ $(= \kappa_1(A))$ can be obtained by calling F07NUF (CSYCON/ZSYCON).

8. Further Comments

The total number of real floating-point operations is approximately $8n^2r$.

This routine may be followed by a call to F07NVF (CSYRFS/ZSYRFS) to refine the solution and return an error estimate.

The real analogue of this routine is F07MEF (SSYTRS/DSYTRS).

9. Example

To solve the system of equations $AX = B$, where

$$A = \begin{pmatrix} -0.39 - 0.71i & 5.14 - 0.64i & -7.86 - 2.96i & 3.80 + 0.92i \\ 5.14 - 0.64i & 8.86 + 1.81i & -3.52 + 0.58i & 5.32 - 1.59i \\ -7.86 - 2.96i & -3.52 + 0.58i & -2.83 - 0.03i & -1.54 - 2.86i \\ 3.80 + 0.92i & 5.32 - 1.59i & -1.54 - 2.86i & -0.56 + 0.12i \end{pmatrix} \quad \text{and}$$

$$B = \begin{pmatrix} -55.64 + 41.22i & -19.09 - 35.97i \\ -48.18 + 66.00i & -12.08 - 27.02i \\ -0.49 - 1.47i & 6.95 + 20.49i \\ -6.43 + 19.24i & -4.59 - 35.53i \end{pmatrix}.$$

Here A is symmetric and must first be factorized by F07NRF (CSYTRF/ZSYTRF).

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07NSF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, LWORK, NRHMAX, LDB
      PARAMETER        (NMAX=8,LDA=NMAX,LWORK=64*NMAX,NRHMAX=NMAX,
     +                 LDB=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          A(LDA,NMAX), B(LDB,NRHMAX), WORK(LWORK)
      INTEGER          IPIV(NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         csytrf, csytrs, X04DBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07NSF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
*
```

```
*        Factorize A
*
         CALL csytrf(UPLO,N,A,LDA,IPIV,WORK,LWORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute solution
*
            CALL csytrs(UPLO,N,NRHS,A,LDA,IPIV,B,LDB,INFO)
*
*           Print solution
*
            IFAIL = 0
            CALL X04DBF('General',' ',N,NRHS,B,LDB,'Bracketed','F7.4',
     +                  'Solution(s)','Integer',RLABS,'Integer',CLABS,
     +                  80,0,IFAIL)
         ELSE
            WRITE (NOUT,*) 'The factor D is singular'
         END IF
       END IF
       STOP
*
       END
```

9.2. Program Data

```
F07NSF Example Program Data
  4  2                                                        :Values of N and NRHS
  'L'                                                             :Value of UPLO
 (-0.39,-0.71)
 ( 5.14,-0.64) ( 8.86, 1.81)
 (-7.86,-2.96) (-3.52, 0.58) (-2.83,-0.03)
 ( 3.80, 0.92) ( 5.32,-1.59) (-1.54,-2.86) (-0.56, 0.12)  :End of matrix A
 (-55.64, 41.22) (-19.09,-35.97)
 (-48.18, 66.00) (-12.08,-27.02)
 ( -0.49, -1.47) (  6.95, 20.49)
 ( -6.43, 19.24) ( -4.59,-35.53)                          :End of matrix B
```

9.3. Program Results

```
F07NSF Example Program Results

Solution(s)
                        1                     2
 1   ( 1.0000,-1.0000) (-2.0000,-1.0000)
 2   (-2.0000, 5.0000) ( 1.0000,-3.0000)
 3   ( 3.0000,-2.0000) ( 3.0000, 2.0000)
 4   (-4.0000, 3.0000) (-1.0000, 1.0000)
```

F07NUF (CSYCON/ZSYCON) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07NUF (CSYCON/ZSYCON) estimates the condition number of a complex symmetric matrix A, where A has been factorized by F07NRF (CSYTRF/ZSYTRF).

2. Specification

```
SUBROUTINE F07NUF (UPLO, N, A, LDA, IPIV, ANORM, RCOND, WORK, INFO)
ENTRY      csycon  (UPLO, N, A, LDA, IPIV, ANORM, RCOND, WORK, INFO)
INTEGER          N, LDA, IPIV(*), INFO
real             ANORM, RCOND
complex          A(LDA,*), WORK(*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine estimates the condition number (in the 1-norm) of a complex symmetric matrix A:

$$\kappa_1(A) = \|A\|_1 \|A^{-1}\|_1.$$

Since A is symmetric, $\kappa_1(A) = \kappa_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty$.

Because $\kappa_1(A)$ is infinite if A is singular, the routine actually returns an estimate of the **reciprocal** of $\kappa_1(A)$.

The routine should be preceded by a call to F06UFF to compute $\|A\|_1$ and a call to F07NRF (CSYTRF/ZSYTRF) to compute the Bunch-Kaufman factorization of A. The routine then uses Higham's implementation of Hager's method [1] to estimate $\|A^{-1}\|_1$.

4. References

[1] HIGHAM, N.J.
FORTRAN Codes for Estimating the One-norm of a Real or Complex Matrix, with Applications to Condition Estimation.
ACM Trans. Math. Softw., 14, pp. 381-396, 1988.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates how A has been factorized as follows:

if UPLO = 'U', then $A = PUDU^TP^T$, where U is upper triangular;

if UPLO = 'L', then $A = PLDL^TP^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: A(LDA,*) – ***complex*** array. *Input*

Note: the second dimension of the array A must be at least max(1,N).

On entry: details of the factorization of A, as returned by F07NRF (CSYTRF/ZSYTRF).

4: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07NUF (CSYCON/ZSYCON) is called.

Constraint: LDA ≥ max(1,N).

5: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: details of the interchanges and the block structure of D, as returned by F07NRF (CSYTRF/ZSYTRF).

6: ANORM – ***real***. *Input*

On entry: the 1-norm of the **original** matrix A, which may be computed by calling F06UFF. ANORM must be computed either **before** calling F07NRF (CSYTRF/ZSYTRF) or else from a copy of the original matrix A.

Constraint: ANORM ≥ 0.0.

7: RCOND – ***real***. *Output*

On exit: an estimate of the reciprocal of the condition number of A. RCOND is set to zero if exact singularity is detected or the estimate underflows. If RCOND is less than ***machine precision***, then A is singular to working precision.

8: WORK(*) – ***complex*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,2*N).

9: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed estimate RCOND is never less than the true value ρ, and in practice is nearly always less than 10ρ, although examples can be constructed where RCOND is much larger.

8. Further Comments

A call to this routine involves solving a number of systems of linear equations of the form $Ax = b$; the number is usually 5 and never more than 11. Each solution involves approximately $8n^2$ real floating-point operations but takes considerably longer than a call to F07NSF (CSYTRS/ZSYTRS) with 1 right-hand side, because extra care is taken to avoid overflow when A is approximately singular.

The real analogue of this routine is F07MGF (SSYCON/DSYCON).

9. Example

To estimate the condition number in the 1-norm (or infinity-norm) of the matrix A, where

$$A = \begin{pmatrix} -0.39 - 0.71i & 5.14 - 0.64i & -7.86 - 2.96i & 3.80 + 0.92i \\ 5.14 - 0.64i & 8.86 + 1.81i & -3.52 + 0.58i & 5.32 - 1.59i \\ -7.86 - 2.96i & -3.52 + 0.58i & -2.83 - 0.03i & -1.54 - 2.86i \\ 3.80 + 0.92i & 5.32 - 1.59i & -1.54 - 2.86i & -0.56 + 0.12i \end{pmatrix}.$$

Here A is symmetric and must first be factorized by F07NRF (CSYTRF/ZSYTRF). The true condition number in the 1-norm is 32.92.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07NUF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, LWORK
      PARAMETER        (NMAX=8,LDA=NMAX,LWORK=64*NMAX)
*     .. Local Scalars ..
      real             ANORM, RCOND
      INTEGER          I, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          A(LDA,NMAX), WORK(LWORK)
      real             RWORK(NMAX)
      INTEGER          IPIV(NMAX)
*     .. External Functions ..
      real             F06UFF, X02AJF
      EXTERNAL         F06UFF, X02AJF
*     .. External Subroutines ..
      EXTERNAL         csycon, csytrf
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07NUF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
*
*        Compute norm of A
*
         ANORM = F06UFF('1-norm',UPLO,N,A,LDA,RWORK)
*
*        Factorize A
*
         CALL csytrf(UPLO,N,A,LDA,IPIV,WORK,LWORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Estimate condition number
*
            CALL csycon(UPLO,N,A,LDA,IPIV,ANORM,RCOND,WORK,INFO)
*
            IF (RCOND.GE.X02AJF()) THEN
               WRITE (NOUT,99999) 'Estimate of condition number =',
     +           1.0e0/RCOND
            ELSE
               WRITE (NOUT,*) 'A is singular to working precision'
            END IF
         ELSE
```

```
            WRITE (NOUT,*) 'The factor D is singular'
         END IF
      END IF
      STOP
*
99999 FORMAT (1X,A,1P,e10.2)
      END
```

9.2. Program Data

```
F07NUF Example Program Data
  4                                                                  :Value of N
  'L'                                                                :Value of UPLO
 (-0.39,-0.71)
 ( 5.14,-0.64) ( 8.86, 1.81)
 (-7.86,-2.96) (-3.52, 0.58) (-2.83,-0.03)
 ( 3.80, 0.92) ( 5.32,-1.59) (-1.54,-2.86) (-0.56, 0.12)  :End of matrix A
```

9.3. Program Results

```
F07NUF Example Program Results

Estimate of condition number =  1.57E+01
```

F07NVF (CSYRFS/ZSYRFS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07NVF (CSYRFS/ZSYRFS) returns error bounds for the solution of a complex symmetric system of linear equations with multiple right-hand sides, $AX = B$. It improves the solution by iterative refinement, in order to reduce the backward error as much as possible.

2. Specification

```
SUBROUTINE F07NVF (UPLO, N, NRHS, A, LDA, AF, LDAF, IPIV, B, LDB, X, LDX,
1                  FERR, BERR, WORK, RWORK, INFO)
ENTRY       csyrfs  (UPLO, N, NRHS, A, LDA, AF, LDAF, IPIV, B, LDB, X, LDX,
1                  FERR, BERR, WORK, RWORK, INFO)
INTEGER          N, NRHS, LDA, LDAF, IPIV(*), LDB, LDX, INFO
real             FERR(*), BERR(*), RWORK(*)
complex          A(LDA,*), AF(LDAF,*), B(LDB,*), X(LDX,*), WORK(*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine returns the backward errors and estimated bounds on the forward errors for the solution of a complex symmetric system of linear equations with multiple right-hand sides $AX = B$. The routine handles each right-hand side vector (stored as a column of the matrix B) independently, so we describe the function of the routine in terms of a single right-hand side b and solution x.

Given a computed solution x, the routine computes the *component-wise backward error* β. This is the size of the smallest relative perturbation in each element of A and b such that x is the exact solution of a perturbed system

$$(A+\delta A)x = b + \delta b$$

$$|\delta a_{ij}| \leq \beta |a_{ij}| \quad \text{and} \quad |\delta b_i| \leq \beta |b_i|.$$

Then the routine estimates a bound for the *component-wise forward error* in the computed solution, defined by:

$$\max_i |x_i - \hat{x}_i| / \max_i |x_i|$$

where $\hat{x}$ is the true solution.

For details of the method, see the Chapter Introduction.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether the upper or lower triangular part of A is stored and how A has been factorized, as follows:

if UPLO = 'U', then the upper triangular part of A is stored and A is factorized as $PUDU^TP^T$, where U is upper triangular;

if UPLO = 'L', then the lower triangular part of A is stored and A is factorized as $PLDL^TP^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS ≥ 0.

4: A(LDA,*) – ***complex*** array. *Input*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the n by n original symmetric matrix A as supplied to F07NRF (CSYTRF/ZSYTRF).

5: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07NVF (CSYRFS/ZSYRFS) is called.

Constraint: LDA ≥ max(1,N).

6: AF(LDAF,*) – ***complex*** array. *Input*

Note: the second dimension of the array AF must be at least max(1,N).

On entry: details of the factorization of A, as returned by F07NRF (CSYTRF/ZSYTRF).

7: LDAF – INTEGER. *Input*

On entry: the first dimension of the array AF as declared in the (sub)program from which F07NVF (CSYRFS/ZSYRFS) is called.

Constraint: LDAF ≥ max(1,N).

8: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: details of the interchanges and the block structure of D, as returned by F07NRF (CSYTRF/ZSYTRF).

9: B(LDB,*) – ***complex*** array. *Input*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

10: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07NVF (CSYRFS/ZSYRFS) is called.

Constraint: LDB ≥ max(1,N).

11: X(LDX,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array X must be at least max(1,NRHS).

On entry: the n by r solution matrix X, as returned by F07NSF (CSYTRS/ZSYTRS).

On exit: the improved solution matrix X.

12: LDX – INTEGER. *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which F07NVF (CSYRFS/ZSYRFS) is called.

Constraint: LDX ≥ max(1,N).

13: FERR(*) – ***real*** array. *Output*

Note: the dimension of the array FERR must be at least max(1,NRHS).

On exit: FERR(j) contains an estimated error bound for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

14: BERR(*) – ***real*** array. *Output*

Note: the dimension of the array BERR must be at least max(1,NRHS).

On exit: BERR(j) contains the component-wise backward error bound β for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

15: WORK(*) – ***complex*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,2*N).

16: RWORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array RWORK must be at least max(1,N).

17: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The bounds returned in FERR are not rigorous, because they are estimated, not computed exactly; but in practice they almost always overestimate the actual error.

8. Further Comments

For each right-hand side, computation of the backward error involves a minimum of $16n^2$ real floating-point operations. Each step of iterative refinement involves an additional $24n^2$ real operations. At most 5 steps of iterative refinement are performed, but usually only 1 or 2 steps are required.

Estimating the forward error involves solving a number of systems of linear equations of the form $Ax = b$; the number is usually 5 and never more than 11. Each solution involves approximately $8n^2$ real operations.

The real analogue of this routine is F07MHF (SSYRFS/DSYRFS).

9. Example

To solve the system of equations $AX = B$ using iterative refinement and to compute the forward and backward error bounds, where

$$A = \begin{pmatrix} -0.39 - 0.71i & 5.14 - 0.64i & -7.86 - 2.96i & 3.80 + 0.92i \\ 5.14 - 0.64i & 8.86 + 1.81i & -3.52 + 0.58i & 5.32 - 1.59i \\ -7.86 - 2.96i & -3.52 + 0.58i & -2.83 - 0.03i & -1.54 - 2.86i \\ 3.80 + 0.92i & 5.32 - 1.59i & -1.54 - 2.86i & -0.56 + 0.12i \end{pmatrix} \text{ and}$$

$$B = \begin{pmatrix} -55.64 + 41.22i & -19.09 - 35.97i \\ -48.18 + 66.00i & -12.08 - 27.02i \\ -0.49 - 1.47i & 6.95 + 20.49i \\ -6.43 + 19.24i & -4.59 - 35.53i \end{pmatrix}.$$

Here A is symmetric and must first be factorized by F07NRF (CSYTRF/ZSYTRF).

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07NVF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, NRHMAX, LDA, LWORK, LDAF, LDB, LDX
      PARAMETER        (NMAX=8,NRHMAX=NMAX,LDA=NMAX,LWORK=64*NMAX,
     +                 LDAF=NMAX,LDB=NMAX,LDX=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          A(LDA,NMAX), AF(LDAF,NMAX), B(LDB,NRHMAX),
     +                 WORK(LWORK), X(LDX,NMAX)
      real             BERR(NRHMAX), FERR(NRHMAX), RWORK(NMAX)
      INTEGER          IPIV(NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         csyrfs, csytrf, csytrs, F06TFF, X04DBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07NVF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file, and copy A to AF and B to X
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
         CALL F06TFF(UPLO,N,N,A,LDA,AF,LDAF)
         CALL F06TFF('General',N,NRHS,B,LDB,X,LDX)
*
*        Factorize A in the array AF
*
         CALL csytrf(UPLO,N,AF,LDAF,IPIV,WORK,LWORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
```

```
*           Compute solution in the array X
*
            CALL csytrs(UPLO,N,NRHS,AF,LDAF,IPIV,X,LDX,INFO)
*
*           Improve solution, and compute backward errors and
*           estimated bounds on the forward errors
*
            CALL csyrfs(UPLO,N,NRHS,A,LDA,AF,LDAF,IPIV,B,LDB,X,LDX,
     +                  FERR,BERR,WORK,RWORK,INFO)
*
*           Print solution
*
            IFAIL = 0
            CALL X04DBF('General',' ',N,NRHS,X,LDX,'Bracketed','F7.4',
     +                  'Solution(s)','Integer',RLABS,'Integer',CLABS,
     +                  80,0,IFAIL)
            WRITE (NOUT,*)
            WRITE (NOUT,*) 'Backward errors (machine-dependent)'
            WRITE (NOUT,99999) (BERR(J),J=1,NRHS)
            WRITE (NOUT,*)
     +        'Estimated forward error bounds (machine-dependent)'
            WRITE (NOUT,99999) (FERR(J),J=1,NRHS)
         ELSE
            WRITE (NOUT,*) 'The factor D is singular'
         END IF
      END IF
      STOP
*
99999 FORMAT ((5X,1P,4(e11.1,7X)))
      END
```

9.2. Program Data

```
F07NVF Example Program Data
  4  2                                                        :Values of N and NRHS
  'L'                                                             :Value of UPLO
 (-0.39,-0.71)
 ( 5.14,-0.64) ( 8.86, 1.81)
 (-7.86,-2.96) (-3.52, 0.58) (-2.83,-0.03)
 ( 3.80, 0.92) ( 5.32,-1.59) (-1.54,-2.86) (-0.56, 0.12)   :End of matrix A
 (-55.64, 41.22) (-19.09,-35.97)
 (-48.18, 66.00) (-12.08,-27.02)
 ( -0.49, -1.47) (  6.95, 20.49)
 ( -6.43, 19.24) ( -4.59,-35.53)                           :End of matrix B
```

9.3. Program Results

```
 F07NVF Example Program Results

 Solution(s)
                      1                   2
  1   ( 1.0000,-1.0000) (-2.0000,-1.0000)
  2   (-2.0000, 5.0000) ( 1.0000,-3.0000)
  3   ( 3.0000,-2.0000) ( 3.0000, 2.0000)
  4   (-4.0000, 3.0000) (-1.0000, 1.0000)

 Backward errors (machine-dependent)
         6.3E-17           7.3E-17
 Estimated forward error bounds (machine-dependent)
         1.2E-14           1.2E-14
```

F07NWF (CSYTRI/ZSYTRI) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07NWF (CSYTRI/ZSYTRI) computes the inverse of a complex symmetric matrix A, where A has been factorized by F07NRF (CSYTRF/ZSYTRF).

2. Specification

```
SUBROUTINE F07NWF (UPLO, N, A, LDA, IPIV, WORK, INFO)
ENTRY      csytri  (UPLO, N, A, LDA, IPIV, WORK, INFO)
INTEGER          N, LDA, IPIV(*), INFO
complex          A(LDA,*), WORK(*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

To compute the inverse of a complex symmetric matrix A, this routine must be preceded by a call to F07NRF (CSYTRF/ZSYTRF), which computes the Bunch-Kaufman factorization of A.

If UPLO = 'U', $A = PUDU^TP^T$ and A^{-1} is computed by solving $U^TP^TXPU = D^{-1}$ for X.

If UPLO = 'L', $A = PLDL^TP^T$ and A^{-1} is computed by solving $L^TP^TXPL = D^{-1}$ for X.

4. References

[1] DU CROZ, J.J. and HIGHAM, N.J.
Stability of Methods for Matrix Inversion.
LAPACK Working Note No. 27, University of Tennessee, Knoxville, 1990.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates how A has been factorized as follows:

if UPLO = 'U', then $A = PUDU^TP^T$, where U is upper triangular;

if UPLO = 'L', then $A = PLDL^TP^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: A(LDA,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array A must be at least max(1,N).

On entry: details of the factorization of A, as returned by F07NRF (CSYTRF/ZSYTRF).

On exit: the factorization is overwritten by the n by n symmetric matrix A^{-1}. If UPLO = 'U', the upper triangle of A^{-1} is stored in the upper triangular part of the array; if UPLO = 'L', the lower triangle of A^{-1} is stored in the lower triangular part of the array.

4: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07NWF (CSYTRI/ZSYTRI) is called.

Constraint: LDA ≥ max(1,N).

5: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: details of the interchanges and the block structure of D, as returned by F07NRF (CSYTRF/ZSYTRF).

6: WORK(*) – ***complex*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,2*N).

7: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, d_{ii} is exactly zero; D is singular and the inverse of A cannot be computed.

7. Accuracy

The computed inverse X satisfies a bound of the form

$|DU^TP^TXPU-I| \leq c(n)\varepsilon(|D||U^T|P^T|X|P|U|+|D||D^{-1}|)$ if UPLO = 'U', or

$|DL^TP^TXPL-I| \leq c(n)\varepsilon(|D||L^T|P^T|X|P|L|+|D||D^{-1}|)$ if UPLO = 'L',

where $c(n)$ is a modest linear function of n, and ε is the ***machine precision***.

8. Further Comments

The total number of real floating-point operations is approximately $\frac{8}{3}n^3$.

The real analogue of this routine is F07MJF (SSYTRI/DSYTRI).

9. Example

To compute the inverse of the matrix A, where

$$A = \begin{pmatrix} -0.39 - 0.71i & 5.14 - 0.64i & -7.86 - 2.96i & 3.80 + 0.92i \\ 5.14 - 0.64i & 8.86 + 1.81i & -3.52 + 0.58i & 5.32 - 1.59i \\ -7.86 - 2.96i & -3.52 + 0.58i & -2.83 - 0.03i & -1.54 - 2.86i \\ 3.80 + 0.92i & 5.32 - 1.59i & -1.54 - 2.86i & -0.56 + 0.12i \end{pmatrix}.$$

Here A is symmetric and must first be factorized by F07NRF (CSYTRF/ZSYTRF).

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07NWF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, LWORK
      PARAMETER        (NMAX=8,LDA=NMAX,LWORK=64*NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
      CHARACTER        UPLO
```

```
*     .. Local Arrays ..
      complex           A(LDA,NMAX), WORK(LWORK)
      INTEGER           IPIV(NMAX)
      CHARACTER         CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL          csytrf, csytri, X04DBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07NWF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
*
*        Factorize A
*
         CALL csytrf(UPLO,N,A,LDA,IPIV,WORK,LWORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute inverse of A
*
            CALL csytri(UPLO,N,A,LDA,IPIV,WORK,INFO)
*
*           Print inverse
*
            IFAIL = 0
            CALL X04DBF(UPLO,'Nonunit',N,N,A,LDA,'Bracketed','F7.4',
     +                  'Inverse','Integer',RLABS,'Integer',CLABS,80,0,
     +                  IFAIL)
         ELSE
            WRITE (NOUT,*) 'The factor D is singular'
         END IF
       END IF
       STOP
*
       END
```

9.2. Program Data

```
F07NWF Example Program Data
  4                                                          :Value of N
  'L'                                                        :Value of UPLO
 (-0.39,-0.71)
 ( 5.14,-0.64) ( 8.86, 1.81)
 (-7.86,-2.96) (-3.52, 0.58) (-2.83,-0.03)
 ( 3.80, 0.92) ( 5.32,-1.59) (-1.54,-2.86) (-0.56, 0.12)   :End of matrix A
```

9.3. Program Results

```
 F07NWF Example Program Results

 Inverse
                      1                  2                  3                  4
  1  (-0.1562,-0.1014)
  2  ( 0.0400, 0.1527) ( 0.0946,-0.1475)
  3  ( 0.0550, 0.0845) (-0.0326,-0.1370) (-0.1320,-0.0102)
  4  ( 0.2162,-0.0742) (-0.0995,-0.0461) (-0.1793, 0.1183) (-0.2269, 0.2383)
```

F07PDF (SSPTRF/DSPTRF) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07PDF (SSPTRF/DSPTRF) computes the Bunch-Kaufman factorization of a real symmetric indefinite matrix, using packed storage.

2. Specification

```
SUBROUTINE F07PDF (UPLO, N, AP, IPIV, INFO)
ENTRY      ssptrf  (UPLO, N, AP, IPIV, INFO)
INTEGER          N, IPIV(*), INFO
real             AP(*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine factorizes a real symmetric matrix A, using the Bunch-Kaufman diagonal pivoting method and packed storage. A is factorized as either $A = PUDU^TP^T$ if UPLO = 'U', or $A = PLDL^TP^T$ if UPLO = 'L', where P is a permutation matrix, U (or L) is a unit upper (or lower) triangular matrix and D is a symmetric block diagonal matrix with 1 by 1 and 2 by 2 diagonal blocks; U (or L) has 2 by 2 unit diagonal blocks corresponding to the 2 by 2 blocks of D. Row and column interchanges are performed to ensure numerical stability while preserving symmetry.

This method is suitable for symmetric matrices which are not known to be positive-definite. If A is in fact positive-definite, no interchanges are performed and no 2 by 2 blocks occur in D.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §4.4.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether the upper or lower triangular part of A is stored and how A is factorized, as follows:

if UPLO = 'U', then the upper triangular part of A is stored and A is factorized as $PUDU^TP^T$, where U is upper triangular;

if UPLO = 'L', then the lower triangular part of A is stored and A is factorized as $PLDL^TP^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: AP(*) – ***real*** array. *Input/Output*

Note: the dimension of the array AP must be at least max(1,N*(N+1)/2).

On entry: the n by n symmetric matrix A, packed by columns. More precisely, if UPLO = 'U', the upper triangle of A must be stored with element a_{ij} in AP(i+j(j−1)/2) for $i \leq j$; if UPLO = 'L', the lower triangle of A must be stored with element a_{ij} in AP(i+(2n−j)(j−1)/2) for $i \geq j$.

On exit: A is overwritten by details of the block diagonal matrix D and the multipliers used to obtain the factor U or L as specified by UPLO.

4: IPIV(*) – INTEGER array. *Output*

Note: the dimension of the array IPIV must be at least max(1,N).

On exit: details of the interchanges and the block structure of D. More precisely, if IPIV(i) = $k > 0$, then d_{ii} is a 1 by 1 pivot block and the ith row and column of A was interchanged with the kth row and column. If UPLO = 'U' and IPIV(i−1) = IPIV(i) = $-l < 0$, then $\begin{pmatrix} d_{i-1,i-1} & d_{i,i-1} \\ d_{i,i-1} & d_{ii} \end{pmatrix}$ is a 2 by 2 pivot block and the (i−1)th row and column of A was interchanged with the lth row and column; if UPLO = 'L' and IPIV(i) = IPIV(i+1) = $-m < 0$, then $\begin{pmatrix} d_{ii} & d_{i+1,i} \\ d_{i+1,i} & d_{i+1,i+1} \end{pmatrix}$ is a 2 by 2 pivot block and the (i+1)th row and column of A was interchanged with the mth row and column.

5: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, d_{ii} is exactly zero. The factorization has been completed but the block diagonal matrix D is exactly singular, and division by zero will occur if it is subsequently used to solve a system of linear equations or to compute A^{-1}.

7. Accuracy

If UPLO = 'U', the computed factors U and D are the exact factors of a perturbed matrix $A + E$, where

$$|E| \leq c(n)\varepsilon P|U||D||U^T|P^T,$$

$c(n)$ is a modest linear function of n, and ε is the ***machine precision***. If UPLO = 'L', a similar statement holds for the computed factors L and D.

8. Further Comments

The elements of D overwrite the corresponding elements of A; if D has 2 by 2 blocks, only the upper or lower triangle is stored, as specified by UPLO.

The unit diagonal elements of U or L and the 2 by 2 unit diagonal blocks are not stored. The remaining elements of U or L overwrite elements in the corresponding columns of A, but additional row interchanges must be applied to recover U or L explicitly (this is seldom necessary). If IPIV(i) = i, for i = 1,2,...,n (as is the case when A is positive-definite), then U or L are stored explicitly in packed form (except for their unit diagonal elements which are equal to 1).

The total number of floating-point operations is approximately $\frac{1}{3}n^3$.

A call to this routine may be followed by calls to the routines:

F07PEF (SSPTRS/DSPTRS) to solve $AX = B$;
F07PGF (SSPCON/DSPCON) to estimate the condition number of A;
F07PJF (SSPTRI/DSPTRI) to compute the inverse of A.

The complex analogues of this routine are F07PRF (CHPTRF/ZHPTRF) for Hermitian matrices and F07QRF (CSPTRF/ZSPTRF) for symmetric matrices.

9. Example

To compute the Bunch-Kaufman factorization of the matrix A, where

$$A = \begin{pmatrix} 2.07 & 3.87 & 4.20 & -1.15 \\ 3.87 & -0.21 & 1.87 & 0.63 \\ 4.20 & 1.87 & 1.15 & 2.06 \\ -1.15 & 0.63 & 2.06 & -1.81 \end{pmatrix},$$

using packed storage.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07PDF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=8)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             AP(NMAX*(NMAX+1)/2)
      INTEGER          IPIV(NMAX)
*     .. External Subroutines ..
      EXTERNAL         ssptrf, X04CCF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07PDF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
*
*        Factorize A
*
         CALL ssptrf(UPLO,N,AP,IPIV,INFO)
*
         WRITE (NOUT,*)
*
*        Print details of factorization
*
         IFAIL = 0
*
         CALL X04CCF(UPLO,'Nonunit',N,AP,'Details of factorization',
     +               IFAIL)
*
```

```
*        Print pivot indices
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'IPIV'
         WRITE (NOUT,99999) (IPIV(I),I=1,N)
*
         IF (INFO.NE.0) WRITE (NOUT,*) 'The factor D is singular'
*
      END IF
      STOP
*
99999 FORMAT ((3X,7I11))
      END
```

9.2. Program Data

```
F07PDF Example Program Data
  4                                :Value of N
  'L'                              :Value of UPLO
  2.07
  3.87  -0.21
  4.20   1.87   1.15
 -1.15   0.63   2.06  -1.81    :End of matrix A
```

9.3. Program Results

```
 F07PDF Example Program Results

 Details of factorization
             1           2           3           4
  1      2.0700
  2      4.2000      1.1500
  3      0.2230      0.8115     -2.5907
  4      0.6537     -0.5960      0.3031      0.4074

 IPIV
            -3          -3           3           4
```

F07PEF (SSPTRS/DSPTRS) - NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07PEF (SSPTRS/DSPTRS) solves a real symmetric indefinite system of linear equations with multiple right-hand sides, $AX = B$, where A has been factorized by F07PDF (SSPTRF/DSPTRF), using packed storage.

2. Specification

```
SUBROUTINE F07PEF (UPLO, N, NRHS, AP, IPIV, B, LDB, INFO)
ENTRY      ssptrs  (UPLO, N, NRHS, AP, IPIV, B, LDB, INFO)
INTEGER          N, NRHS, IPIV(*), LDB, INFO
real             AP(*), B(LDB,*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

To solve a real symmetric indefinite system of linear equations $AX = B$, this routine must be preceded by a call to F07PDF (SSPTRF/DSPTRF) which computes the Bunch-Kaufman factorization of A using packed storage.

If UPLO = 'U', $A = PUDU^TP^T$, where P is a permutation matrix, U is an upper triangular matrix and D is a symmetric block diagonal matrix with 1 by 1 and 2 by 2 blocks; the solution X is computed by solving $PUDY = B$ and then $U^TP^TX = Y$.

If UPLO = 'L', $A = PLDL^TP^T$, where L is a lower triangular matrix; the solution X is computed by solving $PLDY = B$ and then $L^TP^TX = Y$.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §4.4.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates how A has been factorized as follows:

if UPLO = 'U', then $A = PUDU^TP^T$, where U is upper triangular;

if UPLO = 'L', then $A = PLDL^TP^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS $\geq$ 0.

```
            CALL X04CAF('General',' ',N,NRHS,B,LDB,'Solution(s)',IFAIL)
*
         ELSE
            WRITE (NOUT,*) 'The factor D is singular'
         END IF
      END IF
      STOP
*
      END
```

9.2. Program Data

```
F07PEF Example Program Data
  4  2                              :Values of N and NRHS
  'L'                               :Value of UPLO
  2.07
  3.87  -0.21
  4.20   1.87   1.15
 -1.15   0.63   2.06  -1.81        :End of matrix A
 -9.50  27.85
 -8.38   9.90
 -6.07  19.25
 -0.96   3.93                       :End of matrix B
```

9.3. Program Results

```
F07PEF Example Program Results

Solution(s)
                  1           2
  1       -4.0000      1.0000
  2       -1.0000      4.0000
  3        2.0000      3.0000
  4        5.0000      2.0000
```

F07PEF (SSPTRS/DSPTRS) - NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07PEF (SSPTRS/DSPTRS) solves a real symmetric indefinite system of linear equations with multiple right-hand sides, $AX = B$, where A has been factorized by F07PDF (SSPTRF/DSPTRF), using packed storage.

2. Specification

```
SUBROUTINE F07PEF (UPLO, N, NRHS, AP, IPIV, B, LDB, INFO)
ENTRY      ssptrs  (UPLO, N, NRHS, AP, IPIV, B, LDB, INFO)
INTEGER          N, NRHS, IPIV(*), LDB, INFO
real             AP(*), B(LDB,*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

To solve a real symmetric indefinite system of linear equations $AX = B$, this routine must be preceded by a call to F07PDF (SSPTRF/DSPTRF) which computes the Bunch-Kaufman factorization of A using packed storage.

If UPLO = 'U', $A = PUDU^TP^T$, where P is a permutation matrix, U is an upper triangular matrix and D is a symmetric block diagonal matrix with 1 by 1 and 2 by 2 blocks; the solution X is computed by solving $PUDY = B$ and then $U^TP^TX = Y$.

If UPLO = 'L', $A = PLDL^TP^T$, where L is a lower triangular matrix; the solution X is computed by solving $PLDY = B$ and then $L^TP^TX = Y$.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §4.4.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates how A has been factorized as follows:

if UPLO = 'U', then $A = PUDU^TP^T$, where U is upper triangular;

if UPLO = 'L', then $A = PLDL^TP^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS $\geq$ 0.

4: AP(*) – ***real*** array. *Input*

Note: the dimension of the array AP must be at least max(1,N*(N+1)/2).

On entry: details of the factorization of A stored in packed form, as returned by F07PDF (SSPTRF/DSPTRF).

5: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: details of the interchanges and the block structure of D, as returned by F07PDF (SSPTRF/DSPTRF).

6: B(LDB,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

On exit: the n by r solution matrix X.

7: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07PEF (SSPTRS/DSPTRS) is called.

Constraint: LDB $\geq$ max(1,N).

8: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

For each right-hand side vector b, the computed solution x is the exact solution of a perturbed system of equations $(A+E)x = b$, where

$|E| \leq c(n)\varepsilon P|U||D||U^T|P^T$ if UPLO = 'U',

$|E| \leq c(n)\varepsilon P|L||D||L^T|P^T$ if UPLO = 'L',

$c(n)$ is a modest linear function of n, and ε is the ***machine precision***.

If $\hat{x}$ is the true solution, then the computed solution x satisfies a forward error bound of the form

$$\frac{\|x-\hat{x}\|_\infty}{\|x\|_\infty} \leq c(n)\mathrm{cond}(A,x)\varepsilon$$

where $\mathrm{cond}(A,x) = \||A^{-1}||A||x|\|_\infty / \|x\|_\infty \leq \mathrm{cond}(A) = \||A^{-1}||A|\|_\infty \leq \kappa_\infty(A)$. Note that $\mathrm{cond}(A,x)$ can be much smaller than $\mathrm{cond}(A)$.

Forward and backward error bounds can be computed by calling F07PHF (SSPRFS/DSPRFS), and an estimate for $\kappa_\infty(A)$ $(= \kappa_1(A))$ can be obtained by calling F07PGF (SSPCON/DSPCON).

8. Further Comments

The total number of floating-point operations is approximately $2n^2r$.

This routine may be followed by a call to F07PHF (SSPRFS/DSPRFS) to refine the solution and return an error estimate.

The complex analogues of this routine are F07PSF (CHPTRS/ZHPTRS) for Hermitian matrices and F07QSF (CSPTRS/ZSPTRS) for symmetric matrices.

9. Example

To solve the system of equations $AX = B$, where

$$A = \begin{pmatrix} 2.07 & 3.87 & 4.20 & -1.15 \\ 3.87 & -0.21 & 1.87 & 0.63 \\ 4.20 & 1.87 & 1.15 & 2.06 \\ -1.15 & 0.63 & 2.06 & -1.81 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} -9.50 & 27.85 \\ -8.38 & 9.90 \\ -6.07 & 19.25 \\ -0.96 & 3.93 \end{pmatrix}.$$

Here A is symmetric indefinite, stored in packed form, and must first be factorized by F07PDF (SSPTRF/DSPTRF).

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07PEF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, NRHMAX, LDB
      PARAMETER        (NMAX=8,NRHMAX=NMAX,LDB=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             AP(NMAX*(NMAX+1)/2), B(LDB,NRHMAX)
      INTEGER          IPIV(NMAX)
*     .. External Subroutines ..
      EXTERNAL         ssptrf, ssptrs, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07PEF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
*
*        Factorize A
*
         CALL ssptrf(UPLO,N,AP,IPIV,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute solution
*
            CALL ssptrs(UPLO,N,NRHS,AP,IPIV,B,LDB,INFO)
*
*           Print solution
*
            IFAIL = 0
*
```

```
            CALL X04CAF('General',' ',N,NRHS,B,LDB,'Solution(s)',IFAIL)
*
         ELSE
            WRITE (NOUT,*) 'The factor D is singular'
         END IF
      END IF
      STOP
*
      END
```

9.2. Program Data

```
F07PEF Example Program Data
  4  2                              :Values of N and NRHS
  'L'                               :Value of UPLO
  2.07
  3.87  -0.21
  4.20   1.87   1.15
 -1.15   0.63   2.06  -1.81        :End of matrix A
 -9.50  27.85
 -8.38   9.90
 -6.07  19.25
 -0.96   3.93                       :End of matrix B
```

9.3. Program Results

```
 F07PEF Example Program Results

 Solution(s)
                  1           2
  1       -4.0000      1.0000
  2       -1.0000      4.0000
  3        2.0000      3.0000
  4        5.0000      2.0000
```

F07PGF (SSPCON/DSPCON) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07PGF (SSPCON/DSPCON) estimates the condition number of a real symmetric indefinite matrix A, where A has been factorized by F07PDF (SSPTRF/DSPTRF), using packed storage.

2. Specification

```
      SUBROUTINE F07PGF (UPLO, N, AP, IPIV, ANORM, RCOND, WORK, IWORK,
     1                   INFO)
      ENTRY       sspcon (UPLO, N, AP, IPIV, ANORM, RCOND, WORK, IWORK,
     1                   INFO)
      INTEGER          N, IPIV(*), IWORK(*), INFO
      real             AP(*), ANORM, RCOND, WORK(*)
      CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine estimates the condition number (in the 1-norm) of a real symmetric indefinite matrix A:

$$\kappa_1(A) = \|A\|_1 \|A^{-1}\|_1.$$

Since A is symmetric, $\kappa_1(A) = \kappa_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty$.

Because $\kappa_1(A)$ is infinite if A is singular, the routine actually returns an estimate of the **reciprocal** of $\kappa_1(A)$.

The routine should be preceded by a call to F06RDF to compute $\|A\|_1$ and a call to F07PDF (SSPTRF/DSPTRF) to compute the Bunch-Kaufman factorization of A. The routine then uses Higham's implementation of Hager's method [1] to estimate $\|A^{-1}\|_1$.

4. References

[1] HIGHAM, N.J.
FORTRAN Codes for Estimating the One-norm of a Real or Complex Matrix, with Applications to Condition Estimation.
ACM Trans. Math. Softw., 14, pp. 381-396, 1988.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates how A has been factorized as follows:

if UPLO = 'U', then $A = PUDU^TP^T$, where U is upper triangular;

if UPLO = 'L', then $A = PLDL^TP^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: AP(*) – ***real*** array. *Input*

Note: the dimension of the array AP must be at least max(1,N*(N+1)/2).

On entry: details of the factorization of A stored in packed form, as returned by F07PDF (SSPTRF/DSPTRF).

4: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: details of the interchanges and the block structure of D, as returned by F07PDF (SSPTRF/DSPTRF).

5: ANORM – ***real***. *Input*

On entry: the 1-norm of the **original** matrix A, which may be computed by calling F06RDF. ANORM must be computed either **before** calling F07PDF (SSPTRF/DSPTRF) or else from a copy of the original matrix A.

Constraint: ANORM $\geq$ 0.0.

6: RCOND – ***real***. *Output*

On exit: an estimate of the reciprocal of the condition number of A. RCOND is set to zero if exact singularity is detected or the estimate underflows. If RCOND is less than ***machine precision***, then A is singular to working precision.

7: WORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,2*N).

8: IWORK(*) – INTEGER array. *Workspace*

Note: the dimension of the array IWORK must be at least max(1,N).

9: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed estimate RCOND is never less than the true value ρ, and in practice is nearly always less than 10ρ, although examples can be constructed where RCOND is much larger.

8. Further Comments

A call to this routine involves solving a number of systems of linear equations of the form $Ax = b$; the number is usually 4 or 5 and never more than 11. Each solution involves approximately $2n^2$ floating-point operations but takes considerably longer than a call to F07PEF (SSPTRS/DSPTRS) with 1 right-hand side, because extra care is taken to avoid overflow when A is approximately singular.

The complex analogues of this routine are F07PUF (CHPCON/ZHPCON) for Hermitian matrices and F07QUF (CSPCON/ZSPCON) for symmetric matrices.

9. Example

To estimate the condition number in the 1-norm (or infinity-norm) of the matrix A, where

$$A = \begin{pmatrix} 2.07 & 3.87 & 4.20 & -1.15 \\ 3.87 & -0.21 & 1.87 & 0.63 \\ 4.20 & 1.87 & 1.15 & 2.06 \\ -1.15 & 0.63 & 2.06 & -1.81 \end{pmatrix}.$$

Here A is symmetric indefinite, stored in packed form, and must first be factorized by F07PDF (SSPTRF/DSPTRF). The true condition number in the 1-norm is 75.68.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07PGF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=8)
*     .. Local Scalars ..
      real             ANORM, RCOND
      INTEGER          I, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             AP(NMAX*(NMAX+1)/2), WORK(2*NMAX)
      INTEGER          IPIV(NMAX), IWORK(NMAX)
*     .. External Functions ..
      real             F06RDF, X02AJF
      EXTERNAL         F06RDF, X02AJF
*     .. External Subroutines ..
      EXTERNAL         sspcon, ssptrf
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07PGF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
*
*        Compute norm of A
*
         ANORM = F06RDF('1-norm',UPLO,N,AP,WORK)
*
*        Factorize A
*
         CALL ssptrf(UPLO,N,AP,IPIV,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Estimate condition number
*
            CALL sspcon(UPLO,N,AP,IPIV,ANORM,RCOND,WORK,IWORK,INFO)
*
            IF (RCOND.GE.X02AJF()) THEN
               WRITE (NOUT,99999) 'Estimate of condition number =',
     +           1.0e0/RCOND
            ELSE
               WRITE (NOUT,*) 'A is singular to working precision'
            END IF
         ELSE
            WRITE (NOUT,*) 'The factor D is singular'
         END IF
      END IF
      STOP
*
99999 FORMAT (1X,A,1P,e10.2)
      END
```

9.2. Program Data

```
F07PGF Example Program Data
  4                                :Value of N
  'L'                              :Value of UPLO
  2.07
  3.87  -0.21
  4.20   1.87   1.15
 -1.15   0.63   2.06  -1.81    :End of matrix A
```

9.3. Program Results

```
 F07PGF Example Program Results

 Estimate of condition number =  7.57E+01
```

F07PHF (SSPRFS/DSPRFS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07PHF (SSPRFS/DSPRFS) returns error bounds for the solution of a real symmetric indefinite system of linear equations with multiple right-hand sides, $AX = B$ using packed storage. It improves the solution by iterative refinement, in order to reduce the backward error as much as possible.

2. Specification

```
      SUBROUTINE F07PHF (UPLO, N, NRHS, AP, AFP, IPIV, B, LDB, X, LDX, FERR,
     1                   BERR, WORK, IWORK, INFO)
      ENTRY       ssprfs  (UPLO, N, NRHS, AP, AFP, IPIV, B, LDB, X, LDX, FERR,
     1                   BERR, WORK, IWORK, INFO)
      INTEGER           N, NRHS, IPIV(*), LDB, LDX, IWORK(*), INFO
      real              AP(*), AFP(*), B(LDB,*), X(LDX,*), FERR(*), BERR(*),
     1                  WORK(*)
      CHARACTER*1       UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine returns the backward errors and estimated bounds on the forward errors for the solution of a real symmetric indefinite system of linear equations with multiple right-hand sides $AX = B$, using packed storage. The routine handles each right-hand side vector (stored as a column of the matrix B) independently, so we describe the function of the routine in terms of a single right-hand side b and solution x.

Given a computed solution x, the routine computes the *component-wise backward error* β. This is the size of the smallest relative perturbation in each element of A and b such that x is the exact solution of a perturbed system

$$(A+\delta A)x = b + \delta b$$

$$|\delta a_{ij}| \leq \beta |a_{ij}| \text{ and } |\delta b_i| \leq \beta |b_i|.$$

Then the routine estimates a bound for the *component-wise forward error* in the computed solution, defined by:

$$\max_i |x_i - \hat{x}_i| / \max_i |x_i|$$

where $\hat{x}$ is the true solution.

For details of the method, see the Chapter Introduction.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether the upper or lower triangular part of A is stored and how A has been factorized, as follows:

if UPLO = 'U', then the upper triangular part of A is stored and A is factorized as $PUDU^TP^T$, where U is upper triangular;

if UPLO = 'L', then the lower triangular part of A is stored and A is factorized as $PLDL^TP^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS ≥ 0.

4: AP(*) – ***real*** array. *Input*

Note: the dimension of the array AP must be at least max(1,N*(N+1)/2).

On entry: the n by n original symmetric matrix A as supplied to F07PDF (SSPTRF/DSPTRF).

5: AFP(*) – ***real*** array. *Input*

Note: the dimension of the array AFP must be at least max(1,N*(N+1)/2).

On entry: details of the factorization of A stored in packed form, as returned by F07PDF (SSPTRF/DSPTRF).

6: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: details of the interchanges and the block structure of D, as returned by F07PDF (SSPTRF/DSPTRF).

7: B(LDB,*) – ***real*** array. *Input*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

8: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07PHF (SSPRFS/DSPRFS) is called.

Constraint: LDB ≥ max(1,N).

9: X(LDX,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array X must be at least max(1,NRHS).

On entry: the n by r solution matrix X, as returned by F07PEF (SSPTRS/DSPTRS).

On exit: the improved solution matrix X.

10: LDX – INTEGER. *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which F07PHF (SSPRFS/DSPRFS) is called.

Constraint: LDX ≥ max(1,N).

11: FERR(*) – ***real*** array. *Output*

Note: the dimension of the array FERR must be at least max(1,NRHS).

On exit: FERR(j) contains an estimated error bound for the jth solution vector, that is, the jth column of X, for j = 1,2,...,r.

12: BERR(*) – ***real*** array. *Output*

Note: the dimension of the array BERR must be at least max(1,NRHS).

On exit: BERR(j) contains the component-wise backward error bound β for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

13: WORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,3*N).

14: IWORK(*) – INTEGER array. *Workspace*

Note: the dimension of the array IWORK must be at least max(1,N).

15: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The bounds returned in FERR are not rigorous, because they are estimated, not computed exactly; but in practice they almost always overestimate the actual error.

8. Further Comments

For each right-hand side, computation of the backward error involves a minimum of $4n^2$ floating-point operations. Each step of iterative refinement involves an additional $6n^2$ operations. At most 5 steps of iterative refinement are performed, but usually only 1 or 2 steps are required.

Estimating the forward error involves solving a number of systems of linear equations of the form $Ax = b$; the number is usually 4 or 5 and never more than 11. Each solution involves approximately $2n^2$ operations.

The complex analogues of this routine are F07PVF (CHPRFS/ZHPRFS) for Hermitian matrices and F07QVF (CSPRFS/ZSPRFS) for symmetric matrices.

9. Example

To solve the system of equations $AX = B$ using iterative refinement and to compute the forward and backward error bounds, where

$$A = \begin{pmatrix} 2.07 & 3.87 & 4.20 & -1.15 \\ 3.87 & -0.21 & 1.87 & 0.63 \\ 4.20 & 1.87 & 1.15 & 2.06 \\ -1.15 & 0.63 & 2.06 & -1.81 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} -9.50 & 27.85 \\ -8.38 & 9.90 \\ -6.07 & 19.25 \\ -0.96 & 3.93 \end{pmatrix}.$$

Here A is symmetric indefinite, stored in packed form, and must first be factorized by F07PDF (SSPTRF/DSPTRF).

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07PHF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, NRHMAX, LDB, LDX
      PARAMETER        (NMAX=8,NRHMAX=NMAX,LDB=NMAX,LDX=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             AFP(NMAX*(NMAX+1)/2), AP(NMAX*(NMAX+1)/2),
     +                 B(LDB,NRHMAX), BERR(NRHMAX), FERR(NRHMAX),
     +                 WORK(3*NMAX), X(LDX,NMAX)
      INTEGER          IPIV(NMAX), IWORK(NMAX)
*     .. External Subroutines ..
      EXTERNAL         F06QFF, ssprfs, ssptrf, ssptrs, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07PHF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file, and copy A to AFP and B to X
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
         DO 20 I = 1, N*(N+1)/2
            AFP(I) = AP(I)
   20    CONTINUE
*
         CALL F06QFF('General',N,NRHS,B,LDB,X,LDX)
*
*        Factorize A in the array AFP
*
         CALL ssptrf(UPLO,N,AFP,IPIV,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute solution in the array X
*
            CALL ssptrs(UPLO,N,NRHS,AFP,IPIV,X,LDX,INFO)
*
*           Improve solution, and compute backward errors and
*           estimated bounds on the forward errors
*
            CALL ssprfs(UPLO,N,NRHS,AP,AFP,IPIV,B,LDB,X,LDX,FERR,
     +                  BERR,WORK,IWORK,INFO)
*
*           Print solution
*
            IFAIL = 0
*
```

```
            CALL X04CAF('General',' ',N,NRHS,X,LDX,'Solution(s)',IFAIL)
*
            WRITE (NOUT,*)
            WRITE (NOUT,*) 'Backward errors (machine-dependent)'
            WRITE (NOUT,99999) (BERR(J),J=1,NRHS)
            WRITE (NOUT,*)
     +        'Estimated forward error bounds (machine-dependent)'
            WRITE (NOUT,99999) (FERR(J),J=1,NRHS)
          ELSE
            WRITE (NOUT,*) 'The factor D is singular'
          END IF
        END IF
        STOP
*
99999 FORMAT ((3X,1P,7e11.1))
      END
```

9.2. Program Data

```
F07PHF Example Program Data
  4  2                              :Values of N and NRHS
  'L'                               :Value of UPLO
  2.07
  3.87  -0.21
  4.20   1.87   1.15
 -1.15   0.63   2.06  -1.81        :End of matrix A
 -9.50  27.85
 -8.38   9.90
 -6.07  19.25
 -0.96   3.93                       :End of matrix B
```

9.3. Program Results

```
F07PHF Example Program Results

Solution(s)
                1           2
 1      -4.0000      1.0000
 2      -1.0000      4.0000
 3       2.0000      3.0000
 4       5.0000      2.0000

Backward errors (machine-dependent)
       5.1E-17     1.2E-16
Estimated forward error bounds (machine-dependent)
       2.3E-14     3.5E-14
```

F07PJF (SSPTRI/DSPTRI) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07PJF (SSPTRI/DSPTRI) computes the inverse of a real symmetric indefinite matrix A, where A has been factorized by F07PDF (SSPTRF/DSPTRF), using packed storage.

2. Specification

```
SUBROUTINE F07PJF (UPLO, N, AP, IPIV, WORK, INFO)
ENTRY      ssptri  (UPLO, N, AP, IPIV, WORK, INFO)
INTEGER          N, IPIV(*), INFO
real             AP(*), WORK(*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

To compute the inverse of a real symmetric indefinite matrix A, this routine must be preceded by a call to F07PDF (SSPTRF/DSPTRF), which computes the Bunch-Kaufman factorization of A using packed storage.

If UPLO = 'U', $A = PUDU^TP^T$ and A^{-1} is computed by solving $U^TP^TXPU = D^{-1}$.

If UPLO = 'L', $A = PLDL^TP^T$ and A^{-1} is computed by solving $L^TP^TXPL = D^{-1}$.

4. References

[1] DU CROZ, J.J. and HIGHAM, N.J.
Stability of Methods for Matrix Inversion.
LAPACK Working Note No. 27, University of Tennessee, Knoxville, 1990.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates how A has been factorized as follows:

if UPLO = 'U', then $A = PUDU^TP^T$, where U is upper triangular;

if UPLO = 'L', then $A = PLDL^TP^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: AP(*) – ***real*** array. *Input/Output*

Note: the dimension of the array AP must be at least max(1,N*(N+1)/2).

On entry: details of the factorization of A stored in packed form, as returned by F07PDF (SSPTRF/DSPTRF).

On exit: the factorization is overwritten by the n by n symmetric matrix A^{-1} stored in packed form. More precisely, the (i,j)th element of A^{-1} is stored in AP(i+j(j−1)/2) for $i \le j$ if UPLO = 'U', and in AP(i+(2n−j)(j−1)/2) for $i \ge j$ if UPLO = 'L'.

4: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: details of the interchanges and the block structure of D, as returned by F07PDF (SSPTRF/DSPTRF).

5: WORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,N).

6: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, d_{ii} is exactly zero; D is singular and the inverse of A cannot be computed.

7. Accuracy

The computed inverse X satisfies a bound of the form

$|DU^TP^TXPU-I| \leq c(n)\varepsilon(|D||U^T|P^T|X|P|U|+|D||D^{-1}|)$ if UPLO = 'U', or

$|DL^TP^TXPL-I| \leq c(n)\varepsilon(|D||L^T|P^T|X|P|L|+|D||D^{-1}|)$ if UPLO = 'L',

where $c(n)$ is a modest linear function of n, and ε is the ***machine precision***.

8. Further Comments

The total number of floating-point operations is approximately $\frac{2}{3}n^3$.

The complex analogues of this routine are F07PWF (CHPTRI/ZHPTRI) for Hermitian matrices and F07QWF (CSPTRI/ZSPTRI) for symmetric matrices.

9. Example

To compute the inverse of the matrix A, where

$$A = \begin{pmatrix} 2.07 & 3.87 & 4.20 & -1.15 \\ 3.87 & -0.21 & 1.87 & 0.63 \\ 4.20 & 1.87 & 1.15 & 2.06 \\ -1.15 & 0.63 & 2.06 & -1.81 \end{pmatrix}.$$

Here A is symmetric indefinite, stored in packed form, and must first be factorized by F07PDF (SSPTRF/DSPTRF).

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07PJF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=8)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
      CHARACTER        UPLO
```

```
*     .. Local Arrays ..
      real               AP(NMAX*(NMAX+1)/2), WORK(NMAX)
      INTEGER            IPIV(NMAX)
*     .. External Subroutines ..
      EXTERNAL           ssptrf, ssptri, X04CCF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07PJF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
*
*        Factorize A
*
         CALL ssptrf(UPLO,N,AP,IPIV,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute inverse of A
*
            CALL ssptri(UPLO,N,AP,IPIV,WORK,INFO)
*
*           Print inverse
*
            IFAIL = 0
*
            CALL X04CCF(UPLO,'Nonunit',N,AP,'Inverse',IFAIL)
*
         ELSE
            WRITE (NOUT,*) 'The factor D is singular'
         END IF
      END IF
      STOP
*
      END
```

9.2. Program Data

```
F07PJF Example Program Data
  4                              :Value of N
  'L'                            :Value of UPLO
  2.07
  3.87  -0.21
  4.20   1.87   1.15
 -1.15   0.63   2.06  -1.81     :End of matrix A
```

9.3. Program Results

```
 F07PJF Example Program Results

 Inverse
                1           2           3           4
  1        0.7485
  2        0.5221     -0.1605
  3       -1.0058     -0.3131      1.3501
  4       -1.4386     -0.7440      2.0667      2.4547
```

F07PRF (CHPTRF/ZHPTRF) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07PRF (CHPTRF/ZHPTRF) computes the Bunch-Kaufman factorization of a complex Hermitian indefinite matrix, using packed storage.

2. Specification

```
SUBROUTINE F07PRF (UPLO, N, AP, IPIV, INFO)
ENTRY      chptrf  (UPLO, N, AP, IPIV, INFO)
INTEGER          N, IPIV(*), INFO
complex          AP(*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine factorizes a complex Hermitian matrix A, using the Bunch-Kaufman diagonal pivoting method and packed storage. A is factorized as either $A = PUDU^HP^T$ if UPLO = 'U', or $A = PLDL^HP^T$ if UPLO = 'L', where P is a permutation matrix, U (or L) is a unit upper (or lower) triangular matrix and D is an Hermitian block diagonal matrix with 1 by 1 and 2 by 2 diagonal blocks; U (or L) has 2 by 2 unit diagonal blocks corresponding to the 2 by 2 blocks of D. Row and column interchanges are performed to ensure numerical stability while keeping the matrix Hermitian.

This method is suitable for Hermitian matrices which are not known to be positive-definite. If A is in fact positive-definite, no interchanges are performed and no 2 by 2 blocks occur in D.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §4.4.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether the upper or lower triangular part of A is stored and how A is factorized, as follows:

if UPLO = 'U', then the upper triangular part of A is stored and A is factorized as $PUDU^HP^T$, where U is upper triangular;

if UPLO = 'L', then the lower triangular part of A is stored and A is factorized as $PLDL^HP^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: AP(*) – ***complex*** array. *Input/Output*

Note: the dimension of the array AP must be at least max(1,N*(N+1)/2).

On entry: the n by n Hermitian matrix A, packed by columns. More precisely, if UPLO = 'U', the upper triangle of A must be stored with element a_{ij} in AP(i+j(j−1)/2) for $i \leq j$; if UPLO = 'L', the lower triangle of A must be stored with element a_{ij} in AP(i+(2n−j)(j−1)/2) for $i \geq j$.

On exit: A is overwritten by details of the block diagonal matrix D and the multipliers used to obtain the factor U or L as specified by UPLO.

4: IPIV(*) – INTEGER array. *Output*

Note: the dimension of the array IPIV must be at least max(1,N).

On exit: details of the interchanges and the block structure of D. More precisely, if IPIV(i) = $k > 0$, then d_{ii} is a 1 by 1 pivot block and the ith row and column of A was interchanged with the kth row and column. If UPLO = 'U' and IPIV(i−1) = IPIV(i) = $-l < 0$, then $\begin{pmatrix} d_{i-1,i-1} & d_{i,i-1} \\ \bar{d}_{i,i-1} & d_{ii} \end{pmatrix}$ is a 2 by 2 pivot block and the (i−1)th row and column of A was interchanged with the lth row and column; if UPLO = 'L' and IPIV(i) = IPIV(i+1) = $-m < 0$, then $\begin{pmatrix} d_{ii} & \bar{d}_{i+1,i} \\ d_{i+1,i} & d_{i+1,i+1} \end{pmatrix}$ is a 2 by 2 pivot block and the (i+1)th row and column of A was interchanged with the mth row and column.

5: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, d_{ii} is exactly zero. The factorization has been completed but the block diagonal matrix D is exactly singular, and division by zero will occur if it is subsequently used to solve a system of linear equations or to compute A^{-1}.

7. Accuracy

If UPLO = 'U', the computed factors U and D are the exact factors of a perturbed matrix $A + E$, where

$$|E| \leq c(n)\varepsilon P|U||D||U^H|P^T,$$

$c(n)$ is a modest linear function of n, and ε is the ***machine precision***. If UPLO = 'L', a similar statement holds for the computed factors L and D.

8. Further Comments

The elements of D overwrite the corresponding elements of A; if D has 2 by 2 blocks, only the upper or lower triangle is stored, as specified by UPLO.

The unit diagonal elements of U or L and the 2 by 2 unit diagonal blocks are not stored. The remaining elements of U and L are stored in the corresponding columns of the array A, but additional row interchanges must be applied to recover U or L explicitly (this is seldom necessary). If IPIV(i) = i, for i = 1,2,...,n (as is the case when A is positive-definite), then U or L are stored explicitly in packed form (except for their unit diagonal elements which are equal to 1).

The total number of real floating-point operations is approximately $\frac{4}{3}n^3$.

A call to this routine may be followed by calls to the routines:

F07PSF (CHPTRS/ZHPTRS) to solve $AX = B$;
F07PUF (CHPCON/ZHPCON) to estimate the condition number of A;
F07PWF (CHPTRI/ZHPTRI) to compute the inverse of A.

The real analogue of this routine is F07PDF (SSPTRF/DSPTRF).

9. Example

To compute the Bunch-Kaufman factorization of the matrix A, where

$$A = \begin{pmatrix} -1.36 + 0.00i & 1.58 + 0.90i & 2.21 - 0.21i & 3.91 + 1.50i \\ 1.58 - 0.90i & -8.87 + 0.00i & -1.84 - 0.03i & -1.78 + 1.18i \\ 2.21 + 0.21i & -1.84 + 0.03i & -4.63 + 0.00i & 0.11 + 0.11i \\ 3.91 - 1.50i & -1.78 - 1.18i & 0.11 - 0.11i & -1.84 + 0.00i \end{pmatrix},$$

using packed storage.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07PRF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=8)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          AP(NMAX*(NMAX+1)/2)
      INTEGER          IPIV(NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         chptrf, X04DDF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07PRF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
*
*        Factorize A
*
         CALL chptrf(UPLO,N,AP,IPIV,INFO)
*
         WRITE (NOUT,*)
*
*        Print details of factorization
*
         IFAIL = 0
         CALL X04DDF(UPLO,'Nonunit',N,AP,'Bracketed','F7.4',
     +              'Details of factorization','Integer',RLABS,
     +              'Integer',CLABS,80,0,IFAIL)
*
```

```
*        Print pivot indices
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'IPIV'
         WRITE (NOUT,99999) (IPIV(I),I=1,N)
*
         IF (INFO.NE.0) WRITE (NOUT,*) 'The factor D is singular'
*
      END IF
      STOP
*
99999 FORMAT ((1X,I12,3I18))
      END
```

9.2. Program Data

```
F07PRF Example Program Data
  4                                                          :Value of N
  'L'                                                        :Value of UPLO
 (-1.36, 0.00)
 ( 1.58,-0.90) (-8.87, 0.00)
 ( 2.21, 0.21) (-1.84, 0.03) (-4.63, 0.00)
 ( 3.91,-1.50) (-1.78,-1.18) ( 0.11,-0.11) (-1.84, 0.00)   :End of matrix A
```

9.3. Program Results

```
 F07PRF Example Program Results

 Details of factorization
                      1                   2                   3                   4
  1  (-1.3600, 0.0000)
  2  ( 3.9100,-1.5000) (-1.8400, 0.0000)
  3  ( 0.3100, 0.0433) ( 0.5637, 0.2850) (-5.4176, 0.0000)
  4  (-0.1518, 0.3743) ( 0.3397, 0.0303) ( 0.2997, 0.1578) (-7.1028, 0.0000)

 IPIV
           -4                  -4                   3                   4
```

F07PSF (CHPTRS/ZHPTRS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07PSF (CHPTRS/ZHPTRS) solves a complex Hermitian indefinite system of linear equations with multiple right-hand sides, $AX = B$, where A has been factorized by F07PRF (CHPTRF/ZHPTRF), using packed storage.

2. Specification

```
SUBROUTINE F07PSF (UPLO, N, NRHS, AP, IPIV, B, LDB, INFO)
ENTRY      chptrs  (UPLO, N, NRHS, AP, IPIV, B, LDB, INFO)
INTEGER          N, NRHS, IPIV(*), LDB, INFO
complex          AP(*), B(LDB,*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

To solve a complex Hermitian indefinite system of linear equations $AX = B$, this routine must be preceded by a call to F07PRF (CHPTRF/ZHPTRF) which computes the Bunch-Kaufman factorization of A using packed storage.

If UPLO = 'U', $A = PUDU^HP^T$, where P is a permutation matrix, U is an upper triangular matrix and D is an Hermitian block diagonal matrix with 1 by 1 and 2 by 2 blocks; the solution X is computed by solving $PUDY = B$ and then $U^HP^TX = Y$.

If UPLO = 'L', $A = PLDL^HP^T$, where L is a lower triangular matrix; the solution X is computed by solving $PLDY = B$ and then $L^HP^TX = Y$.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §4.4.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates how A has been factorized as follows:

if UPLO = 'U', then $A = PUDU^HP^T$, where U is upper triangular;

if UPLO = 'L', then $A = PLDL^HP^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS $\geq$ 0.

4: AP(*) – ***complex*** array. *Input*

Note: the dimension of the array AP must be at least max(1,N*(N+1)/2).

On entry: details of the factorization of A stored in packed form, as returned by F07PRF (CHPTRF/ZHPTRF).

5: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: details of the interchanges and the block structure of D, as returned by F07PRF (CHPTRF/ZHPTRF).

6: B(LDB,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

On exit: the n by r solution matrix X.

7: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07PSF (CHPTRS/ZHPTRS) is called.

Constraint: LDB $\geq$ max(1,N).

8: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

For each right-hand side vector b, the computed solution x is the exact solution of a perturbed system of equations $(A+E)x = b$, where

$|E| \leq c(n)\varepsilon P|U||D||U^H|P^T$ if UPLO = 'U',

$|E| \leq c(n)\varepsilon P|L||D||L^H|P^T$ if UPLO = 'L',

$c(n)$ is a modest linear function of n, and ε is the ***machine precision***.

If $\hat{x}$ is the true solution, then the computed solution x satisfies a forward error bound of the form

$$\frac{\|x-\hat{x}\|_\infty}{\|x\|_\infty} \leq c(n)\text{cond}(A,x)\varepsilon$$

where $\text{cond}(A,x) = \||A^{-1}||A||x|\|_\infty / \|x\|_\infty \leq \text{cond}(A) = \||A^{-1}||A|\|_\infty \leq \kappa_\infty(A)$. Note that $\text{cond}(A,x)$ can be much smaller than $\text{cond}(A)$.

Forward and backward error bounds can be computed by calling F07PVF (CHPRFS/ZHPRFS), and an estimate for $\kappa_\infty(A)$ $(= \kappa_1(A))$ can be obtained by calling F07PUF (CHPCON/ZHPCON).

8. Further Comments

The total number of real floating-point operations is approximately $8n^2r$.

This routine may be followed by a call to F07PVF (CHPRFS/ZHPRFS) to refine the solution and return an error estimate.

The real analogue of this routine is F07PEF (SSPTRS/DSPTRS).

9. Example

To solve the system of equations $AX = B$, where

$$A = \begin{pmatrix} -1.36 + 0.00i & 1.58 + 0.90i & 2.21 - 0.21i & 3.91 + 1.50i \\ 1.58 - 0.90i & -8.87 + 0.00i & -1.84 - 0.03i & -1.78 + 1.18i \\ 2.21 + 0.21i & -1.84 + 0.03i & -4.63 + 0.00i & 0.11 + 0.11i \\ 3.91 - 1.50i & -1.78 - 1.18i & 0.11 - 0.11i & -1.84 + 0.00i \end{pmatrix} \text{ and}$$

$$B = \begin{pmatrix} 7.79 + 5.48i & -35.39 + 18.01i \\ -0.77 - 16.05i & 4.23 - 70.02i \\ -9.58 + 3.88i & -24.79 - 8.40i \\ 2.98 - 10.18i & 28.68 - 39.89i \end{pmatrix}.$$

Here A is Hermitian indefinite, stored in packed form, and must first be factorized by F07PRF (CHPTRF/ZHPTRF).

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07PSF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, NRHMAX, LDB
      PARAMETER        (NMAX=8,NRHMAX=NMAX,LDB=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          AP(NMAX*(NMAX+1)/2), B(LDB,NRHMAX)
      INTEGER          IPIV(NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         chptrf, chptrs, X04DBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07PSF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
*
*        Factorize A
*
         CALL chptrf(UPLO,N,AP,IPIV,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute solution
*
            CALL chptrs(UPLO,N,NRHS,AP,IPIV,B,LDB,INFO)
*
```

```
*           Print solution
*
            IFAIL = 0
            CALL X04DBF('General',' ',N,NRHS,B,LDB,'Bracketed','F7.4',
     +                  'Solution(s)','Integer',RLABS,'Integer',CLABS,
     +                  80,0,IFAIL)
         ELSE
            WRITE (NOUT,*) 'The factor D is singular'
         END IF
       END IF
       STOP
*
       END
```

9.2. Program Data

```
F07PSF Example Program Data
  4  2                                                        :Values of N and NRHS
  'L'                                                             :Value of UPLO
 (-1.36, 0.00)
 ( 1.58,-0.90) (-8.87, 0.00)
 ( 2.21, 0.21) (-1.84, 0.03) (-4.63, 0.00)
 ( 3.91,-1.50) (-1.78,-1.18) ( 0.11,-0.11) (-1.84, 0.00)  :End of matrix A
 ( 7.79,  5.48) (-35.39, 18.01)
 (-0.77,-16.05) (  4.23,-70.02)
 (-9.58,  3.88) (-24.79, -8.40)
 ( 2.98,-10.18) ( 28.68,-39.89)                           :End of matrix B
```

9.3. Program Results

```
 F07PSF Example Program Results

 Solution(s)
                        1                   2
  1   ( 1.0000,-1.0000) ( 3.0000,-4.0000)
  2   (-1.0000, 2.0000) (-1.0000, 5.0000)
  3   ( 3.0000,-2.0000) ( 7.0000,-2.0000)
  4   ( 2.0000, 1.0000) (-8.0000, 6.0000)
```

F07PUF (CHPCON/ZHPCON) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07PUF (CHPCON/ZHPCON) estimates the condition number of a complex Hermitian indefinite matrix A, where A has been factorized by F07PRF (CHPTRF/ZHPTRF), using packed storage.

2. Specification

```
SUBROUTINE F07PUF (UPLO, N, AP, IPIV, ANORM, RCOND, WORK, INFO)
ENTRY      chpcon (UPLO, N, AP, IPIV, ANORM, RCOND, WORK, INFO)
INTEGER          N, IPIV(*), INFO
real             ANORM, RCOND
complex          AP(*), WORK(*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine estimates the condition number (in the 1-norm) of a complex Hermitian indefinite matrix A:

$$\kappa_1(A) = \|A\|_1 \|A^{-1}\|_1.$$

Since A is Hermitian, $\kappa_1(A) = \kappa_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty$.

Because $\kappa_1(A)$ is infinite if A is singular, the routine actually returns an estimate of the **reciprocal** of $\kappa_1(A)$.

The routine should be preceded by a call to F06UDF to compute $\|A\|_1$ and a call to F07PRF (CHPTRF/ZHPTRF) to compute the Bunch-Kaufman factorization of A. The routine then uses Higham's implementation of Hager's method [1] to estimate $\|A^{-1}\|_1$.

4. References

[1] HIGHAM, N.J.
FORTRAN Codes for Estimating the One-norm of a Real or Complex Matrix, with Applications to Condition Estimation.
ACM Trans. Math. Softw., 14, pp. 381-396, 1988.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates how A has been factorized as follows:

if UPLO = 'U', then $A = PUDU^HP^T$, where U is upper triangular;

if UPLO = 'L', then $A = PLDL^HP^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: AP(*) – ***complex*** array. *Input*

Note: the dimension of the array AP must be at least max(1,N*(N+1)/2).

On entry: details of the factorization of A stored in packed form, as returned by F07PRF (CHPTRF/ZHPTRF).

4: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: details of the interchanges and the block structure of D, as returned by F07PRF (CHPTRF/ZHPTRF).

5: ANORM – ***real***. *Input*

On entry: the 1-norm of the **original** matrix A, which may be computed by calling F06UDF. ANORM must be computed either **before** calling F07PRF (CHPTRF/ZHPTRF) or else from a copy of the original matrix A.

Constraint: ANORM ≥ 0.0.

6: RCOND – ***real***. *Output*

On exit: an estimate of the reciprocal of the condition number of A. RCOND is set to zero if exact singularity is detected or the estimate underflows. If RCOND is less than ***machine precision***, then A is singular to working precision.

7: WORK(*) – ***complex*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,2*N).

8: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed estimate RCOND is never less than the true value ρ, and in practice is nearly always less than 10ρ, although examples can be constructed where RCOND is much larger.

8. Further Comments

A call to this routine involves solving a number of systems of linear equations of the form $Ax = b$; the number is usually 5 and never more than 11. Each solution involves approximately $8n^2$ real floating-point operations but takes considerably longer than a call to F07PSF (CHPTRS/ZHPTRS) with 1 right-hand side, because extra care is taken to avoid overflow when A is approximately singular.

The real analogue of this routine is F07PGF (SSPCON/DSPCON).

9. Example

To estimate the condition number in the 1-norm (or infinity-norm) of the matrix A, where

$$A = \begin{pmatrix} -1.36 + 0.00i & 1.58 + 0.90i & 2.21 - 0.21i & 3.91 + 1.50i \\ 1.58 - 0.90i & -8.87 + 0.00i & -1.84 - 0.03i & -1.78 + 1.18i \\ 2.21 + 0.21i & -1.84 + 0.03i & -4.63 + 0.00i & 0.11 + 0.11i \\ 3.91 - 1.50i & -1.78 - 1.18i & 0.11 - 0.11i & -1.84 + 0.00i \end{pmatrix}.$$

Here A is Hermitian indefinite, stored in packed form, and must first be factorized by F07PRF (CHPTRF/ZHPTRF). The true condition number in the 1-norm is 9.10.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07PUF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=8)
*     .. Local Scalars ..
      real             ANORM, RCOND
      INTEGER          I, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          AP(NMAX*(NMAX+1)/2), WORK(2*NMAX)
      real             RWORK(NMAX)
      INTEGER          IPIV(NMAX)
*     .. External Functions ..
      real             F06UDF, X02AJF
      EXTERNAL         F06UDF, X02AJF
*     .. External Subroutines ..
      EXTERNAL         chpcon, chptrf
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07PUF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
*
*        Compute norm of A
*
         ANORM = F06UDF('1-norm',UPLO,N,AP,RWORK)
*
*        Factorize A
*
         CALL chptrf(UPLO,N,AP,IPIV,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Estimate condition number
*
            CALL chpcon(UPLO,N,AP,IPIV,ANORM,RCOND,WORK,INFO)
*
            IF (RCOND.GE.X02AJF()) THEN
               WRITE (NOUT,99999) 'Estimate of condition number =',
     +           1.0e0/RCOND
            ELSE
               WRITE (NOUT,*) 'A is singular to working precision'
            END IF
         ELSE
```

```
            WRITE (NOUT,*) 'The factor D is singular'
         END IF
      END IF
      STOP
*
99999 FORMAT (1X,A,1P,e10.2)
      END
```

9.2. Program Data

```
F07PUF Example Program Data
  4                                                                  :Value of N
  'L'                                                                :Value of UPLO
 (-1.36, 0.00)
 ( 1.58,-0.90) (-8.87, 0.00)
 ( 2.21, 0.21) (-1.84, 0.03) (-4.63, 0.00)
 ( 3.91,-1.50) (-1.78,-1.18) ( 0.11,-0.11) (-1.84, 0.00)  :End of matrix A
```

9.3. Program Results

```
F07PUF Example Program Results

Estimate of condition number =  6.68E+00
```

F07PVF (CHPRFS/ZHPRFS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07PVF (CHPRFS/ZHPRFS) returns error bounds for the solution of a complex Hermitian indefinite system of linear equations with multiple right-hand sides, $AX = B$, using packed storage. It improves the solution by iterative refinement, in order to reduce the backward error as much as possible.

2. Specification

```
      SUBROUTINE F07PVF (UPLO, N, NRHS, AP, AFP, IPIV, B, LDB, X, LDX, FERR,
     1                   BERR, WORK, RWORK, INFO)
      ENTRY       chprfs  (UPLO, N, NRHS, AP, AFP, IPIV, B, LDB, X, LDX, FERR,
     1                   BERR, WORK, RWORK, INFO)
      INTEGER           N, NRHS, IPIV(*), LDB, LDX, INFO
      real              FERR(*), BERR(*), RWORK(*)
      complex           AP(*), AFP(*), B(LDB,*), X(LDX,*), WORK(*)
      CHARACTER*1       UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine returns the backward errors and estimated bounds on the forward errors for the solution of a complex Hermitian indefinite system of linear equations with multiple right-hand sides, $AX = B$, using packed storage. The routine handles each right-hand side vector (stored as a column of the matrix B) independently, so we describe the function of the routine in terms of a single right-hand side b and solution x.

Given a computed solution x, the routine computes the *component-wise backward error* β. This is the size of the smallest relative perturbation in each element of A and b such that x is the exact solution of a perturbed system

$$(A+\delta A)x = b + \delta b$$

$$|\delta a_{ij}| \leq \beta |a_{ij}| \quad \text{and} \quad |\delta b_i| \leq \beta |b_i|.$$

Then the routine estimates a bound for the *component-wise forward error* in the computed solution, defined by:

$$\max_i |x_i - \hat{x}_i| / \max_i |x_i|$$

where $\hat{x}$ is the true solution.

For details of the method, see the Chapter Introduction.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether the upper or lower triangular part of A is stored and how A has been factorized, as follows:

if UPLO = 'U', then the upper triangular part of A is stored and A is factorized as $PUDU^HP^T$, where U is upper triangular;

if UPLO = 'L', then the lower triangular part of A is stored and A is factorized as $PLDL^HP^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS ≥ 0.

4: AP(*) – ***complex*** array. *Input*

Note: the dimension of the array AP must be at least max(1,N*(N+1)/2).

On entry: the n by n original Hermitian matrix A as supplied to F07PRF (CHPTRF/ZHPTRF).

5: AFP(*) – ***complex*** array. *Input*

Note: the dimension of the array AFP must be at least max(1,N*(N+1)/2).

On entry: details of the factorization of A stored in packed form, as returned by F07PRF (CHPTRF/ZHPTRF).

6: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: details of the interchanges and the block structure of D, as returned by F07PRF (CHPTRF/ZHPTRF).

7: B(LDB,*) – ***complex*** array. *Input*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

8: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07PVF (CHPRFS/ZHPRFS) is called.

Constraint: LDB ≥ max(1,N).

9: X(LDX,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array X must be at least max(1,NRHS).

On entry: the n by r solution matrix X, as returned by F07PSF (CHPTRS/ZHPTRS).

On exit: the improved solution matrix X.

10: LDX – INTEGER. *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which F07PVF (CHPRFS/ZHPRFS) is called.

Constraint: LDX ≥ max(1,N).

11: FERR(*) – ***real*** array. *Output*

Note: the dimension of the array FERR must be at least max(1,NRHS).

On exit: FERR(j) contains an estimated error bound for the jth solution vector, that is, the jth column of X, for j = 1,2,...,r.

12: BERR(*) – ***real*** array. *Output*

Note: the dimension of the array BERR must be at least max(1,NRHS).

On exit: BERR(j) contains the component-wise backward error bound β for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

13: WORK(*) – ***complex*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,2*N).

14: RWORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array RWORK must be at least max(1,N).

15: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The bounds returned in FERR are not rigorous, because they are estimated, not computed exactly; but in practice they almost always overestimate the actual error.

8. Further Comments

For each right-hand side, computation of the backward error involves a minimum of $16n^2$ real floating-point operations. Each step of iterative refinement involves an additional $24n^2$ real operations. At most 5 steps of iterative refinement are performed, but usually only 1 or 2 steps are required.

Estimating the forward error involves solving a number of systems of linear equations of the form $Ax = b$; the number is usually 5 and never more than 11. Each solution involves approximately $8n^2$ real operations.

The real analogue of this routine is F07PHF (SSPRFS/DSPRFS).

9. Example

To solve the system of equations $AX = B$ using iterative refinement and to compute the forward and backward error bounds, where

$$A = \begin{pmatrix} -1.36 + 0.00i & 1.58 + 0.90i & 2.21 - 0.21i & 3.91 + 1.50i \\ 1.58 - 0.90i & -8.87 + 0.00i & -1.84 - 0.03i & -1.78 + 1.18i \\ 2.21 + 0.21i & -1.84 + 0.03i & -4.63 + 0.00i & 0.11 + 0.11i \\ 3.91 - 1.50i & -1.78 - 1.18i & 0.11 - 0.11i & -1.84 + 0.00i \end{pmatrix} \text{ and}$$

$$B = \begin{pmatrix} 7.79 + 5.48i & -35.39 + 18.01i \\ -0.77 - 16.05i & 4.23 - 70.02i \\ -9.58 + 3.88i & -24.79 - 8.40i \\ 2.98 - 10.18i & 28.68 - 39.89i \end{pmatrix}.$$

Here A is Hermitian indefinite, stored in packed form, and must first be factorized by F07PRF (CHPTRF/ZHPTRF).

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07PVF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, NRHMAX, LDB, LDX
      PARAMETER        (NMAX=8,NRHMAX=NMAX,LDB=NMAX,LDX=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          AFP(NMAX*(NMAX+1)/2), AP(NMAX*(NMAX+1)/2),
     +                 B(LDB,NRHMAX), WORK(2*NMAX), X(LDX,NMAX)
      real             BERR(NRHMAX), FERR(NRHMAX), RWORK(NMAX)
      INTEGER          IPIV(NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         chprfs, chptrf, chptrs, F06TFF, X04DBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07PVF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file, and copy A to AFP and B to X
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
         DO 20 I = 1, N*(N+1)/2
            AFP(I) = AP(I)
   20    CONTINUE
         CALL F06TFF('General',N,NRHS,B,LDB,X,LDX)
*
*        Factorize A in the array AFP
*
         CALL chptrf(UPLO,N,AFP,IPIV,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute solution in the array X
*
            CALL chptrs(UPLO,N,NRHS,AFP,IPIV,X,LDX,INFO)
*
*           Improve solution, and compute backward errors and
*           estimated bounds on the forward errors
*
            CALL chprfs(UPLO,N,NRHS,AP,AFP,IPIV,B,LDB,X,LDX,FERR,
     +                  BERR,WORK,RWORK,INFO)
*
*           Print solution
*
            IFAIL = 0
            CALL X04DBF('General',' ',N,NRHS,X,LDX,'Bracketed','F7.4',
     +                  'Solution(s)','Integer',RLABS,'Integer',CLABS,
     +                  80,0,IFAIL)
            WRITE (NOUT,*)
            WRITE (NOUT,*) 'Backward errors (machine-dependent)'
```

```
            WRITE (NOUT,99999) (BERR(J),J=1,NRHS)
            WRITE (NOUT,*)
     +        'Estimated forward error bounds (machine-dependent)'
            WRITE (NOUT,99999) (FERR(J),J=1,NRHS)
          ELSE
            WRITE (NOUT,*) 'The factor D is singular'
          END IF
        END IF
        STOP
*
99999 FORMAT ((5X,1P,4(e11.1,7X)))
      END
```

9.2. Program Data

```
F07PVF Example Program Data
  4  2                                                          :Values of N and NRHS
  'L'                                                              :Value of UPLO
 (-1.36, 0.00)
 ( 1.58,-0.90) (-8.87, 0.00)
 ( 2.21, 0.21) (-1.84, 0.03) (-4.63, 0.00)
 ( 3.91,-1.50) (-1.78,-1.18) ( 0.11,-0.11) (-1.84, 0.00)   :End of matrix A
 ( 7.79,  5.48) (-35.39, 18.01)
 (-0.77,-16.05) (  4.23,-70.02)
 (-9.58,  3.88) (-24.79, -8.40)
 ( 2.98,-10.18) ( 28.68,-39.89)                                   :End of matrix B
```

9.3. Program Results

```
F07PVF Example Program Results

Solution(s)
                      1                     2
 1   ( 1.0000,-1.0000) ( 3.0000,-4.0000)
 2   (-1.0000, 2.0000) (-1.0000, 5.0000)
 3   ( 3.0000,-2.0000) ( 7.0000,-2.0000)
 4   ( 2.0000, 1.0000) (-8.0000, 6.0000)

Backward errors (machine-dependent)
        8.0E-17           8.9E-17
Estimated forward error bounds (machine-dependent)
        2.6E-15           3.0E-15
```

F07PWF (CHPTRI/ZHPTRI) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07PWF (CHPTRI/ZHPTRI) computes the inverse of a complex Hermitian indefinite matrix A, where A has been factorized by F07PRF (CHPTRF/ZHPTRF), using packed storage.

2. Specification

```
SUBROUTINE F07PWF (UPLO, N, AP, IPIV, WORK, INFO)
ENTRY      chptri  (UPLO, N, AP, IPIV, WORK, INFO)
INTEGER          N, IPIV(*), INFO
complex          AP(*), WORK(*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

To compute the inverse of a complex Hermitian indefinite matrix A, this routine must be preceded by a call to F07PRF (CHPTRF/ZHPTRF), which computes the Bunch-Kaufman factorization of A using packed storage.

If UPLO = 'U', $A = PUDU^HP^T$ and A^{-1} is computed by solving $U^HP^TXPU = D^{-1}$ for X.

If UPLO = 'L', $A = PLDL^HP^T$ and A^{-1} is computed by solving $L^HP^TXPL = D^{-1}$ for X.

4. References

[1] DU CROZ, J.J. and HIGHAM, N.J.
Stability of Methods for Matrix Inversion.
LAPACK Working Note No. 27, University of Tennessee, Knoxville, 1990.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates how A has been factorized as follows:

if UPLO = 'U', then $A = PUDU^HP^T$, where U is upper triangular;

if UPLO = 'L', then $A = PLDL^HP^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: AP(*) – ***complex*** array. *Input/Output*

Note: the dimension of the array AP must be at least max(1,N*(N+1)/2).

On entry: details of the factorization of A stored in packed form, as returned by F07PRF (CHPTRF/ZHPTRF).

On exit: the factorization is overwritten by the n by n Hermitian matrix A^{-1} stored in packed form. More precisely, the (i,j)th element of A^{-1} is stored in AP(i+j(j−1)/2) for $i \le j$ if UPLO = 'U', and in AP(i+(2n−j)(j−1)/2) for $i \ge j$ if UPLO = 'L'.

4: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: details of the interchanges and the block structure of D, as returned by F07PRF (CHPTRF/ZHPTRF).

5: WORK(*) – ***complex*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,N).

6: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, d_{ii} is exactly zero; D is singular and the inverse of A cannot be computed.

7. Accuracy

The computed inverse X satisfies a bound of the form

$|DU^TP^TXPU-I| \leq c(n)\varepsilon(|D||U^T|P^T|X|P|U|+|D||D^{-1}|)$ if UPLO = 'U', or

$|DL^TP^TXPL-I| \leq c(n)\varepsilon(|D||L^T|P^T|X|P|L|+|D||D^{-1}|)$ if UPLO = 'L',

where $c(n)$ is a modest linear function of n, and ε is the ***machine precision.***

8. Further Comments

The total number of real floating-point operations is approximately $\frac{8}{3}n^3$.

The real analogue of this routine is F07PJF (SSPTRI/DSPTRI).

9. Example

To compute the inverse of the matrix A, where

$$A = \begin{pmatrix} -1.36 + 0.00i & 1.58 + 0.90i & 2.21 - 0.21i & 3.91 + 1.50i \\ 1.58 - 0.90i & -8.87 + 0.00i & -1.84 - 0.03i & -1.78 + 1.18i \\ 2.21 + 0.21i & -1.84 + 0.03i & -4.63 + 0.00i & 0.11 + 0.11i \\ 3.91 - 1.50i & -1.78 - 1.18i & 0.11 - 0.11i & -1.84 + 0.00i \end{pmatrix}.$$

Here A is Hermitian indefinite, stored in packed form, and must first be factorized by F07PRF (CHPTRF/ZHPTRF).

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07PWF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=8)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
      CHARACTER        UPLO
```

```
*     .. Local Arrays ..
      complex           AP(NMAX*(NMAX+1)/2), WORK(NMAX)
      INTEGER           IPIV(NMAX)
      CHARACTER         CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL          chptrf, chptri, X04DDF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07PWF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
*
*        Factorize A
*
         CALL chptrf(UPLO,N,AP,IPIV,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute inverse of A
*
            CALL chptri(UPLO,N,AP,IPIV,WORK,INFO)
*
*           Print inverse
*
            IFAIL = 0
            CALL X04DDF(UPLO,'Nonunit',N,AP,'Bracketed','F7.4',
     +                  'Inverse','Integer',RLABS,'Integer',CLABS,80,0,
     +                  IFAIL)
         ELSE
            WRITE (NOUT,*) 'The factor D is singular'
         END IF
      END IF
      STOP
*
      END
```

9.2. Program Data

```
F07PWF Example Program Data
  4                                                          :Value of N
  'L'                                                        :Value of UPLO
 (-1.36, 0.00)
 ( 1.58,-0.90) (-8.87, 0.00)
 ( 2.21, 0.21) (-1.84, 0.03) (-4.63, 0.00)
 ( 3.91,-1.50) (-1.78,-1.18) ( 0.11,-0.11) (-1.84, 0.00)   :End of matrix A
```

9.3. Program Results

```
 F07PWF Example Program Results

 Inverse
                      1                  2                  3                  4
  1  ( 0.0826, 0.0000)
  2  (-0.0335, 0.0440) (-0.1408, 0.0000)
  3  ( 0.0603,-0.0105) ( 0.0422,-0.0222) (-0.2007, 0.0000)
  4  ( 0.2391,-0.0926) ( 0.0304, 0.0203) ( 0.0982,-0.0635) ( 0.0073, 0.0000)
```

F07QRF (CSPTRF/ZSPTRF) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07QRF (CSPTRF/ZSPTRF) computes the Bunch-Kaufman factorization of a complex symmetric matrix, using packed storage.

2. Specification

```
SUBROUTINE F07QRF (UPLO, N, AP, IPIV, INFO)
ENTRY      csptrf  (UPLO, N, AP, IPIV, INFO)
INTEGER          N, IPIV(*), INFO
complex          AP(*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine factorizes a complex symmetric matrix A, using the Bunch-Kaufman diagonal pivoting method and packed storage. A is factorized as either $A = PUDU^TP^T$ if UPLO = 'U', or $A = PLDL^TP^T$ if UPLO = 'L', where P is a permutation matrix, U (or L) is a unit upper (or lower) triangular matrix and D is a symmetric block diagonal matrix with 1 by 1 and 2 by 2 diagonal blocks; U (or L) has 2 by 2 unit diagonal blocks corresponding to the 2 by 2 blocks of D. Row and column interchanges are performed to ensure numerical stability while preserving symmetry.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §4.4.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether the upper or lower triangular part of A is stored and how A is factorized, as follows:

if UPLO = 'U', then the upper triangular part of A is stored and A is factorized as $PUDU^TP^T$, where U is upper triangular;

if UPLO = 'L', then the lower triangular part of A is stored and A is factorized as $PLDL^TP^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: AP(*) – ***complex*** array. *Input/Output*

Note: the dimension of the array AP must be at least max(1,N*(N+1)/2).

On entry: the n by n symmetric matrix A, packed by columns. More precisely, if UPLO = 'U', the upper triangle of A must be stored with element a_{ij} in AP(i+j(j−1)/2) for $i \le j$; if UPLO = 'L', the lower triangle of A must be stored with element a_{ij} in AP(i+(2n−j)(j−1)/2) for $i \ge j$.

On exit: A is overwritten by details of the block diagonal matrix D and the multipliers used to obtain the factor U or L as specified by UPLO.

4: IPIV(*) – INTEGER array. *Output*

Note: the dimension of the array IPIV must be at least max(1,N).

On exit: details of the interchanges and the block structure of D. More precisely, if IPIV(i) = $k > 0$, then d_{ii} is a 1 by 1 pivot block and the ith row and column of A was interchanged with the kth row and column. If UPLO = 'U' and IPIV($i-1$) = IPIV(i) = $-l < 0$, then $\begin{pmatrix} d_{i-1,i-1} & d_{i,i-1} \\ d_{i,i-1} & d_{ii} \end{pmatrix}$ is a 2 by 2 pivot block and the $(i-1)$th row and column of A was interchanged with the lth row and column; if UPLO = 'L' and IPIV(i) = IPIV($i+1$) = $-m < 0$, then $\begin{pmatrix} d_{ii} & d_{i+1,i} \\ d_{i+1,i} & d_{i+1,i+1} \end{pmatrix}$ is a 2 by 2 pivot block and the $(i+1)$th row and column of A was interchanged with the mth row and column.

5: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, d_{ii} is exactly zero. The factorization has been completed but the block diagonal matrix D is exactly singular, and division by zero will occur if it is subsequently used to solve a system of linear equations or to compute A^{-1}.

7. Accuracy

If UPLO = 'U', the computed factors U and D are the exact factors of a perturbed matrix $A + E$, where

$$|E| \leq c(n)\varepsilon P|U||D||U^T|P^T,$$

$c(n)$ is a modest linear function of n, and ε is the ***machine precision***. If UPLO = 'L', a similar statement holds for the computed factors L and D.

8. Further Comments

The elements of D overwrite the corresponding elements of A; if D has 2 by 2 blocks, only the upper or lower triangle is stored, as specified by UPLO.

The unit diagonal elements of U or L and the 2 by 2 unit diagonal blocks are not stored. The remaining elements of U or L overwrite elements in the corresponding columns of A, but additional row interchanges must be applied to recover U or L explicitly (this is seldom necessary). If IPIV(i) = i, for i = 1,2,...,n, then U or L are stored explicitly in packed form (except for their unit diagonal elements which are equal to 1).

The total number of real floating-point operations is approximately $\frac{4}{3}n^3$.

A call to this routine may be followed by calls to the routines:

F07QSF (CSPTRS/ZSPTRS) to solve $AX = B$;
F07QUF (CSPCON/ZSPCON) to estimate the condition number of A;
F07QWF (CSPTRI/ZSPTRI) to compute the inverse of A.

The real analogue of this routine is F07PDF (SSPTRF/DSPTRF).

9. Example

To compute the Bunch-Kaufman factorization of the matrix A, where

$$A = \begin{pmatrix} -0.39 - 0.71i & 5.14 - 0.64i & -7.86 - 2.96i & 3.80 + 0.92i \\ 5.14 - 0.64i & 8.86 + 1.81i & -3.52 + 0.58i & 5.32 - 1.59i \\ -7.86 - 2.96i & -3.52 + 0.58i & -2.83 - 0.03i & -1.54 - 2.86i \\ 3.80 + 0.92i & 5.32 - 1.59i & -1.54 - 2.86i & -0.56 + 0.12i \end{pmatrix},$$

using packed storage.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07QRF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=8)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          AP(NMAX*(NMAX+1)/2)
      INTEGER          IPIV(NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         csptrf, X04DDF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07QRF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
*
*        Factorize A
*
         CALL csptrf(UPLO,N,AP,IPIV,INFO)
*
         WRITE (NOUT,*)
*
*        Print details of factorization
*
         IFAIL = 0
         CALL X04DDF(UPLO,'Nonunit',N,AP,'Bracketed','F7.4',
     +              'Details of factorization','Integer',RLABS,
     +              'Integer',CLABS,80,0,IFAIL)
*
*        Print pivot indices
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'IPIV'
         WRITE (NOUT,99999) (IPIV(I),I=1,N)
*
```

```
         IF (INFO.NE.0) WRITE (NOUT,*) 'The factor D is singular'
*
      END IF
      STOP
*
99999 FORMAT ((1X,I12,3I18))
      END
```

9.2. Program Data

```
F07QRF Example Program Data
  4                                                              :Value of N
  'L'                                                            :Value of UPLO
 (-0.39,-0.71)
 ( 5.14,-0.64) ( 8.86, 1.81)
 (-7.86,-2.96) (-3.52, 0.58) (-2.83,-0.03)
 ( 3.80, 0.92) ( 5.32,-1.59) (-1.54,-2.86) (-0.56, 0.12)  :End of matrix A
```

9.3. Program Results

```
 F07QRF Example Program Results

 Details of factorization
                    1                  2                  3                  4
  1  (-0.3900,-0.7100)
  2  (-7.8600,-2.9600) (-2.8300,-0.0300)
  3  ( 0.5279,-0.3715) (-0.6078, 0.2811) ( 4.4079, 5.3991)
  4  ( 0.4426, 0.1936) (-0.4823, 0.0150) (-0.1071,-0.3157) (-2.0954,-2.2011)

 IPIV
           -3                 -3                  3                  4
```

F07QSF (CSPTRS/ZSPTRS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07QSF (CSPTRS/ZSPTRS) solves a complex symmetric system of linear equations with multiple right-hand sides, $AX = B$, where A has been factorized by F07QRF (CSPTRF/ZSPTRF), using packed storage.

2. Specification

```
SUBROUTINE F07QSF (UPLO, N, NRHS, AP, IPIV, B, LDB, INFO)
ENTRY      csptrs  (UPLO, N, NRHS, AP, IPIV, B, LDB, INFO)
INTEGER        N, NRHS, IPIV(*), LDB, INFO
complex        AP(*), B(LDB,*)
CHARACTER*1    UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

To solve a complex symmetric system of linear equations $AX = B$, this routine must be preceded by a call to F07QRF (CSPTRF/ZSPTRF) which computes the Bunch-Kaufman factorization of A using packed storage.

If UPLO = 'U', $A = PUDU^TP^T$, where P is a permutation matrix, U is an upper triangular matrix and D is a symmetric block diagonal matrix with 1 by 1 and 2 by 2 blocks; the solution X is computed by solving $PUDY = B$ and then $U^TP^TX = Y$.

If UPLO = 'L', $A = PLDL^TP^T$, where L is a lower triangular matrix; the solution X is computed by solving $PLDY = B$ and then $L^TP^TX = Y$.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §4.4.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates how A has been factorized as follows:

if UPLO = 'U', then $A = PUDU^TP^T$, where U is upper triangular;

if UPLO = 'L', then $A = PLDL^TP^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS $\geq$ 0.

4: AP(*) – ***complex*** array. *Input*

Note: the dimension of the array AP must be at least max(1,N*(N+1)/2).

On entry: details of the factorization of A stored in packed form, as returned by F07QRF (CSPTRF/ZSPTRF).

5: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: details of the interchanges and the block structure of D, as returned by F07QRF (CSPTRF/ZSPTRF).

6: B(LDB,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

On exit: the n by r solution matrix X.

7: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07QSF (CSPTRS/ZSPTRS) is called.

Constraint: LDB ≥ max(1,N).

8: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

For each right-hand side vector b, the computed solution x is the exact solution of a perturbed system of equations $(A+E)x = b$, where

$|E| \leq c(n)\varepsilon P|U||D||U^T|P^T$ if UPLO = 'U',

$|E| \leq c(n)\varepsilon P|L||D||L^T|P^T$ if UPLO = 'L',

$c(n)$ is a modest linear function of n, and ε is the ***machine precision***.

If $\hat{x}$ is the true solution, then the computed solution x satisfies a forward error bound of the form

$$\frac{\|x-\hat{x}\|_\infty}{\|x\|_\infty} \leq c(n)\text{cond}(A,x)\varepsilon$$

where $\text{cond}(A,x) = \||A^{-1}||A||x|\|_\infty / \|x\|_\infty \leq \text{cond}(A) = \||A^{-1}||A|\|_\infty \leq \kappa_\infty(A)$. Note that $\text{cond}(A,x)$ can be much smaller than $\text{cond}(A)$.

Forward and backward error bounds can be computed by calling F07QVF (CSPRFS/ZSPRFS), and an estimate for $\kappa_\infty(A)$ $(= \kappa_1(A))$ can be obtained by calling F07QUF (CSPCON/ZSPCON).

8. Further Comments

The total number of real floating-point operations is approximately $8n^2r$.

This routine may be followed by a call to F07QVF (CSPRFS/ZSPRFS) to refine the solution and return an error estimate.

The real analogue of this routine is F07PEF (SSPTRS/DSPTRS).

9. Example

To solve the system of equations $AX = B$, where

$$A = \begin{pmatrix} -0.39 - 0.71i & 5.14 - 0.64i & -7.86 - 2.96i & 3.80 + 0.92i \\ 5.14 - 0.64i & 8.86 + 1.81i & -3.52 + 0.58i & 5.32 - 1.59i \\ -7.86 - 2.96i & -3.52 + 0.58i & -2.83 - 0.03i & -1.54 - 2.86i \\ 3.80 + 0.92i & 5.32 - 1.59i & -1.54 - 2.86i & -0.56 + 0.12i \end{pmatrix} \quad \text{and}$$

$$B = \begin{pmatrix} -55.64 + 41.22i & -19.09 - 35.97i \\ -48.18 + 66.00i & -12.08 - 27.02i \\ -0.49 - 1.47i & 6.95 + 20.49i \\ -6.43 + 19.24i & -4.59 - 35.53i \end{pmatrix}.$$

Here A is symmetric, stored in packed form, and must first be factorized by F07QRF (CSPTRF/ZSPTRF).

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07QSF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, NRHMAX, LDB
      PARAMETER        (NMAX=8,NRHMAX=NMAX,LDB=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          AP(NMAX*(NMAX+1)/2), B(LDB,NRHMAX)
      INTEGER          IPIV(NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         csptrf, csptrs, X04DBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07QSF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
*
*        Factorize A
*
         CALL csptrf(UPLO,N,AP,IPIV,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute solution
*
            CALL csptrs(UPLO,N,NRHS,AP,IPIV,B,LDB,INFO)
*
```

```
*           Print solution
*
            IFAIL = 0
            CALL X04DBF('General',' ',N,NRHS,B,LDB,'Bracketed','F7.4',
     +                  'Solution(s)','Integer',RLABS,'Integer',CLABS,
     +                  80,0,IFAIL)
         ELSE
            WRITE (NOUT,*) 'The factor D is singular'
         END IF
       END IF
       STOP
*
       END
```

9.2. Program Data

```
F07QSF Example Program Data
  4  2                                                          :Values of N and NRHS
  'L'                                                              :Value of UPLO
 (-0.39,-0.71)
 ( 5.14,-0.64) ( 8.86, 1.81)
 (-7.86,-2.96) (-3.52, 0.58) (-2.83,-0.03)
 ( 3.80, 0.92) ( 5.32,-1.59) (-1.54,-2.86) (-0.56, 0.12)   :End of matrix A
 (-55.64, 41.22) (-19.09,-35.97)
 (-48.18, 66.00) (-12.08,-27.02)
 ( -0.49, -1.47) (  6.95, 20.49)
 ( -6.43, 19.24) ( -4.59,-35.53)                           :End of matrix B
```

9.3. Program Results

```
F07QSF Example Program Results

Solution(s)
                        1                     2
  1   ( 1.0000,-1.0000) (-2.0000,-1.0000)
  2   (-2.0000, 5.0000) ( 1.0000,-3.0000)
  3   ( 3.0000,-2.0000) ( 3.0000, 2.0000)
  4   (-4.0000, 3.0000) (-1.0000, 1.0000)
```

F07QUF (CSPCON/ZSPCON) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07QUF (CSPCON/ZSPCON) estimates the condition number of a complex symmetric matrix A, where A has been factorized by F07QRF (CSPTRF/ZSPTRF), using packed storage.

2. Specification

```
SUBROUTINE F07QUF (UPLO, N, AP, IPIV, ANORM, RCOND, WORK, INFO)
ENTRY      cspcon (UPLO, N, AP, IPIV, ANORM, RCOND, WORK, INFO)
INTEGER          N, IPIV(*), INFO
real             ANORM, RCOND
complex          AP(*), WORK(*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine estimates the condition number (in the 1-norm) of a complex symmetric matrix A:

$$\kappa_1(A) = \|A\|_1 \|A^{-1}\|_1.$$

Since A is symmetric, $\kappa_1(A) = \kappa_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty$.

Because $\kappa_1(A)$ is infinite if A is singular, the routine actually returns an estimate of the **reciprocal** of $\kappa_1(A)$.

The routine should be preceded by a call to F06UGF to compute $\|A\|_1$ and a call to F07QRF (CSPTRF/ZSPTRF) to compute the Bunch-Kaufman factorization of A. The routine then uses Higham's implementation of Hager's method [1] to estimate $\|A^{-1}\|_1$.

4. References

[1] HIGHAM, N.J.
FORTRAN Codes for Estimating the One-norm of a Real or Complex Matrix, with Applications to Condition Estimation.
ACM Trans. Math. Softw., 14, pp. 381-396, 1988.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates how A has been factorized as follows:

if UPLO = 'U', then $A = PUDU^TP^T$, where U is upper triangular;

if UPLO = 'L', then $A = PLDL^TP^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: AP(*) – ***complex*** array. *Input*

Note: the dimension of the array AP must be at least max(1,N*(N+1)/2).

On entry: details of the factorization of A stored in packed form, as returned by F07QRF (CSPTRF/ZSPTRF).

4: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: details of the interchanges and the block structure of D, as returned by F07QRF (CSPTRF/ZSPTRF).

5: ANORM – ***real***. *Input*

On entry: the 1-norm of the **original** matrix A, which may be computed by calling F06UGF. ANORM must be computed either **before** calling F07QRF (CSPTRF/ZSPTRF) or else from a copy of the original matrix A.

Constraint: ANORM $\geq$ 0.0.

6: RCOND – ***real***. *Output*

On exit: an estimate of the reciprocal of the condition number of A. RCOND is set to zero if exact singularity is detected or the estimate underflows. If RCOND is less than ***machine precision***, then A is singular to working precision.

7: WORK(*) – ***complex*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,2*N).

8: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed estimate RCOND is never less than the true value ρ, and in practice is nearly always less than 10ρ, although examples can be constructed where RCOND is much larger.

8. Further Comments

A call to this routine involves solving a number of systems of linear equations of the form $Ax = b$; the number is usually 5 and never more than 11. Each solution involves approximately $8n^2$ real floating-point operations but takes considerably longer than a call to F07QSF (CSPTRS/ZSPTRS) with 1 right-hand side, because extra care is taken to avoid overflow when A is approximately singular.

The real analogue of this routine is F07PGF (SSPCON/DSPCON).

9. Example

To estimate the condition number in the 1-norm (or infinity-norm) of the matrix A, where

$$A = \begin{pmatrix} -0.39 - 0.71i & 5.14 - 0.64i & -7.86 - 2.96i & 3.80 + 0.92i \\ 5.14 - 0.64i & 8.86 + 1.81i & -3.52 + 0.58i & 5.32 - 1.59i \\ -7.86 - 2.96i & -3.52 + 0.58i & -2.83 - 0.03i & -1.54 - 2.86i \\ 3.80 + 0.92i & 5.32 - 1.59i & -1.54 - 2.86i & -0.56 + 0.12i \end{pmatrix}.$$

Here A is symmetric, stored in packed form, and must first be factorized by F07QRF (CSPTRF/ZSPTRF). The true condition number in the 1-norm is 32.92.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07QUF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=8)
*     .. Local Scalars ..
      real             ANORM, RCOND
      INTEGER          I, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          AP(NMAX*(NMAX+1)/2), WORK(2*NMAX)
      real             RWORK(NMAX)
      INTEGER          IPIV(NMAX)
*     .. External Functions ..
      real             F06UGF, X02AJF
      EXTERNAL         F06UGF, X02AJF
*     .. External Subroutines ..
      EXTERNAL         cspcon, csptrf
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07QUF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
*
*        Compute norm of A
*
         ANORM = F06UGF('1-norm',UPLO,N,AP,RWORK)
*
*        Factorize A
*
         CALL csptrf(UPLO,N,AP,IPIV,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Estimate condition number
*
            CALL cspcon(UPLO,N,AP,IPIV,ANORM,RCOND,WORK,INFO)
*
            IF (RCOND.GE.X02AJF()) THEN
               WRITE (NOUT,99999) 'Estimate of condition number =',
     +           1.0e0/RCOND
            ELSE
               WRITE (NOUT,*) 'A is singular to working precision'
            END IF
         ELSE
```

```
            WRITE (NOUT,*) 'The factor D is singular'
          END IF
        END IF
        STOP
*
99999 FORMAT (1X,A,1P,e10.2)
      END
```

9.2. Program Data

```
F07QUF Example Program Data
  4                                                                  :Value of N
  'L'                                                                :Value of UPLO
 (-0.39,-0.71)
 ( 5.14,-0.64) ( 8.86, 1.81)
 (-7.86,-2.96) (-3.52, 0.58) (-2.83,-0.03)
 ( 3.80, 0.92) ( 5.32,-1.59) (-1.54,-2.86) (-0.56, 0.12)   :End of matrix A
```

9.3. Program Results

```
F07QUF Example Program Results

Estimate of condition number =  1.57E+01
```

F07QVF (CSPRFS/ZSPRFS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07QVF (CSPRFS/ZSPRFS) returns error bounds for the solution of a complex symmetric system of linear equations with multiple right-hand sides, $AX = B$ using packed storage. It improves the solution by iterative refinement, in order to reduce the backward error as much as possible.

2. Specification

```
 SUBROUTINE F07QVF (UPLO, N, NRHS, AP, AFP, IPIV, B, LDB, X, LDX, FERR,
1                   BERR, WORK, RWORK, INFO)
 ENTRY       csprfs  (UPLO, N, NRHS, AP, AFP, IPIV, B, LDB, X, LDX, FERR,
1                   BERR, WORK, RWORK, INFO)
 INTEGER          N, NRHS, IPIV(*), LDB, LDX, INFO
 real             FERR(*), BERR(*), RWORK(*)
 complex          AP(*), AFP(*), B(LDB,*), X(LDX,*), WORK(*)
 CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine returns the backward errors and estimated bounds on the forward errors for the solution of a complex symmetric system of linear equations with multiple right-hand sides $AX = B$, using packed storage. The routine handles each right-hand side vector (stored as a column of the matrix B) independently, so we describe the function of the routine in terms of a single right-hand side b and solution x.

Given a computed solution x, the routine computes the *component-wise backward error* β. This is the size of the smallest relative perturbation in each element of A and b such that x is the exact solution of a perturbed system

$$(A+\delta A)x = b + \delta b$$

$$|\delta a_{ij}| \leq \beta |a_{ij}| \quad \text{and} \quad |\delta b_i| \leq \beta |b_i|.$$

Then the routine estimates a bound for the *component-wise forward error* in the computed solution, defined by:

$$\max_i |x_i - \hat{x}_i| / \max_i |x_i|$$

where $\hat{x}$ is the true solution.

For details of the method, see the Chapter Introduction.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether the upper or lower triangular part of A is stored and how A has been factorized, as follows:

if UPLO = 'U', then the upper triangular part of A is stored and A is factorized as $PUDU^TP^T$, where U is upper triangular;

if UPLO = 'L', then the lower triangular part of A is stored and A is factorized as $PLDL^TP^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS ≥ 0.

4: AP(*) – ***complex*** array. *Input*

Note: the dimension of the array AP must be at least max(1,N*(N+1)/2).

On entry: the n by n original symmetric matrix A as supplied to F07QRF (CSPTRF/ZSPTRF).

5: AFP(*) – ***complex*** array. *Input*

Note: the dimension of the array AFP must be at least max(1,N*(N+1)/2).

On entry: details of the factorization of A stored in packed form, as returned by F07QRF (CSPTRF/ZSPTRF).

6: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: details of the interchanges and the block structure of D, as returned by F07QRF (CSPTRF/ZSPTRF).

7: B(LDB,*) – ***complex*** array. *Input*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

8: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07QVF (CSPRFS/ZSPRFS) is called.

Constraint: LDB ≥ max(1,N).

9: X(LDX,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array X must be at least max(1,NRHS).

On entry: the n by r solution matrix X, as returned by F07QSF (CSPTRS/ZSPTRS).

On exit: the improved solution matrix X.

10: LDX – INTEGER. *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which F07QVF (CSPRFS/ZSPRFS) is called.

Constraint: LDX ≥ max(1,N).

11: FERR(*) – ***real*** array. *Output*

Note: the dimension of the array FERR must be at least max(1,NRHS).

On exit: FERR(j) contains an estimated error bound for the jth solution vector, that is, the jth column of X, for j = 1,2,...,r.

12: BERR(*) – ***real*** array. *Output*

Note: the dimension of the array BERR must be at least max(1,NRHS).

On exit: BERR(j) contains the component-wise backward error bound β for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

13: WORK(*) – ***complex*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,2*N).

14: RWORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array RWORK must be at least max(1,N).

15: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The bounds returned in FERR are not rigorous, because they are estimated, not computed exactly; but in practice they almost always overestimate the actual error.

8. Further Comments

For each right-hand side, computation of the backward error involves a minimum of $16n^2$ real floating-point operations. Each step of iterative refinement involves an additional $24n^2$ real operations. At most 5 steps of iterative refinement are performed, but usually only 1 or 2 steps are required.

Estimating the forward error involves solving a number of systems of linear equations of the form $Ax = b$; the number is usually 5 and never more than 11. Each solution involves approximately $8n^2$ real operations.

The real analogue of this routine is F07PHF (SSPRFS/DSPRFS).

9. Example

To solve the system of equations $AX = B$ using iterative refinement and to compute the forward and backward error bounds, where

$$A = \begin{pmatrix} -0.39 - 0.71i & 5.14 - 0.64i & -7.86 - 2.96i & 3.80 + 0.92i \\ 5.14 - 0.64i & 8.86 + 1.81i & -3.52 + 0.58i & 5.32 - 1.59i \\ -7.86 - 2.96i & -3.52 + 0.58i & -2.83 - 0.03i & -1.54 - 2.86i \\ 3.80 + 0.92i & 5.32 - 1.59i & -1.54 - 2.86i & -0.56 + 0.12i \end{pmatrix} \quad \text{and}$$

$$B = \begin{pmatrix} -55.64 + 41.22i & -19.09 - 35.97i \\ -48.18 + 66.00i & -12.08 - 27.02i \\ -0.49 - 1.47i & 6.95 + 20.49i \\ -6.43 + 19.24i & -4.59 - 35.53i \end{pmatrix}.$$

Here A is symmetric, stored in packed form, and must first be factorized by F07QRF (CSPTRF/ZSPTRF).

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07QVF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, NRHMAX, LDB, LDX
      PARAMETER        (NMAX=8,NRHMAX=NMAX,LDB=NMAX,LDX=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          AFP(NMAX*(NMAX+1)/2), AP(NMAX*(NMAX+1)/2),
     +                 B(LDB,NRHMAX), WORK(2*NMAX), X(LDX,NMAX)
      real             BERR(NRHMAX), FERR(NRHMAX), RWORK(NMAX)
      INTEGER          IPIV(NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         csprfs, csptrf, csptrs, F06TFF, X04DBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07QVF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file, and copy A to AFP and B to X
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
         DO 20 I = 1, N*(N+1)/2
            AFP(I) = AP(I)
   20    CONTINUE
         CALL F06TFF('General',N,NRHS,B,LDB,X,LDX)
*
*        Factorize A in the array AFP
*
         CALL csptrf(UPLO,N,AFP,IPIV,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute solution in the array X
*
            CALL csptrs(UPLO,N,NRHS,AFP,IPIV,X,LDX,INFO)
*
*           Improve solution, and compute backward errors and
*           estimated bounds on the forward errors
*
            CALL csprfs(UPLO,N,NRHS,AP,AFP,IPIV,B,LDB,X,LDX,FERR,
     +                  BERR,WORK,RWORK,INFO)
*
*           Print solution
*
            IFAIL = 0
            CALL X04DBF('General',' ',N,NRHS,X,LDX,'Bracketed','F7.4',
     +                  'Solution(s)','Integer',RLABS,'Integer',CLABS,
     +                  80,0,IFAIL)
            WRITE (NOUT,*)
            WRITE (NOUT,*) 'Backward errors (machine-dependent)'
```

```
            WRITE (NOUT,99999) (BERR(J),J=1,NRHS)
            WRITE (NOUT,*)
     +        'Estimated forward error bounds (machine-dependent)'
            WRITE (NOUT,99999) (FERR(J),J=1,NRHS)
         ELSE
            WRITE (NOUT,*) 'The factor D is singular'
         END IF
       END IF
       STOP
*
99999 FORMAT ((5X,1P,4(e11.1,7X)))
      END
```

9.2. Program Data

```
F07QVF Example Program Data
  4  2                                                          :Values of N and NRHS
  'L'                                                              :Value of UPLO
 (-0.39,-0.71)
 ( 5.14,-0.64) ( 8.86, 1.81)
 (-7.86,-2.96) (-3.52, 0.58) (-2.83,-0.03)
 ( 3.80, 0.92) ( 5.32,-1.59) (-1.54,-2.86) (-0.56, 0.12)   :End of matrix A
 (-55.64, 41.22) (-19.09,-35.97)
 (-48.18, 66.00) (-12.08,-27.02)
 ( -0.49, -1.47) (  6.95, 20.49)
 ( -6.43, 19.24) ( -4.59,-35.53)                                  :End of matrix B
```

9.3. Program Results

```
F07QVF Example Program Results

Solution(s)
                     1                     2
 1   ( 1.0000,-1.0000) (-2.0000,-1.0000)
 2   (-2.0000, 5.0000) ( 1.0000,-3.0000)
 3   ( 3.0000,-2.0000) ( 3.0000, 2.0000)
 4   (-4.0000, 3.0000) (-1.0000, 1.0000)

Backward errors (machine-dependent)
        6.3E-17           7.3E-17
Estimated forward error bounds (machine-dependent)
        1.2E-14           1.2E-14
```

F07QWF (CSPTRI/ZSPTRI) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07QWF (CSPTRI/ZSPTRI) computes the inverse of a complex symmetric matrix A, where A has been factorized by F07QRF (CSPTRF/ZSPTRF), using packed storage.

2. Specification

```
SUBROUTINE F07QWF (UPLO, N, AP, IPIV, WORK, INFO)
ENTRY      csptri  (UPLO, N, AP, IPIV, WORK, INFO)
INTEGER          N, IPIV(*), INFO
complex          AP(*), WORK(*)
CHARACTER*1      UPLO
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

To compute the inverse of a complex symmetric matrix A, this routine must be preceded by a call to F07QRF (CSPTRF/ZSPTRF), which computes the Bunch-Kaufman factorization of A using packed storage.

If UPLO = 'U', $A = PUDU^TP^T$ and A^{-1} is computed by solving $U^TP^TXPU = D^{-1}$.

If UPLO = 'L', $A = PLDL^TP^T$ and A^{-1} is computed by solving $L^TP^TXPL = D^{-1}$.

4. References

[1] DU CROZ, J.J. and HIGHAM, N.J.
Stability of Methods for Matrix Inversion.
LAPACK Working Note No. 27, University of Tennessee, Knoxville, 1990.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates how A has been factorized as follows:

if UPLO = 'U', then $A = PUDU^TP^T$, where U is upper triangular;

if UPLO = 'L', then $A = PLDL^TP^T$, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

3: AP(*) – ***complex*** array. *Input/Output*

Note: the dimension of the array AP must be at least max(1,N*(N+1)/2).

On entry: details of the factorization of A stored in packed form, as returned by F07QRF (CSPTRF/ZSPTRF).

On exit: the factorization is overwritten by the n by n symmetric matrix A^{-1} stored in packed form. More precisely, the (i,j)th element of A^{-1} is stored in AP($i+j(j-1)/2$) for $i \leq j$ if UPLO = 'U', and in AP($i+(2n-j)(j-1)/2$) for $i \geq j$ if UPLO = 'L'.

4: IPIV(*) – INTEGER array. *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: details of the interchanges and the block structure of D, as returned by F07QRF (CSPTRF/ZSPTRF).

5: WORK(*) – ***complex*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,N).

6: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, d_{ii} is exactly zero; D is singular and the inverse of A cannot be computed.

7. Accuracy

The computed inverse X satisfies a bound of the form

$|DU^TP^TXPU-I| \leq c(n)\varepsilon(|D||U^T|P^T|X|P|U|+|D||D^{-1}|)$ if UPLO = 'U', or

$|DL^TP^TXPL-I| \leq c(n)\varepsilon(|D||L^T|P^T|X|P|L|+|D||D^{-1}|)$ if UPLO = 'L',

where $c(n)$ is a modest linear function of n, and ε is the ***machine precision***.

8. Further Comments

The total number of real floating-point operations is approximately $\frac{8}{3}n^3$.

The real analogue of this routine is F07PJF (SSPTRI/DSPTRI).

9. Example

To compute the inverse of the matrix A, where

$$A = \begin{pmatrix} -0.39 - 0.71i & 5.14 - 0.64i & -7.86 - 2.96i & 3.80 + 0.92i \\ 5.14 - 0.64i & 8.86 + 1.81i & -3.52 + 0.58i & 5.32 - 1.59i \\ -7.86 - 2.96i & -3.52 + 0.58i & -2.83 - 0.03i & -1.54 - 2.86i \\ 3.80 + 0.92i & 5.32 - 1.59i & -1.54 - 2.86i & -0.56 + 0.12i \end{pmatrix}.$$

Here A is symmetric, stored in packed form, and must first be factorized by F07QRF (CSPTRF/ZSPTRF).

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07QWF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=8)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
      CHARACTER        UPLO
```

```
*     .. Local Arrays ..
      complex           AP(NMAX*(NMAX+1)/2), WORK(NMAX)
      INTEGER           IPIV(NMAX)
      CHARACTER         CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL          csptrf, csptri, X04DDF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07QWF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
*
*        Factorize A
*
         CALL csptrf(UPLO,N,AP,IPIV,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute inverse of A
*
            CALL csptri(UPLO,N,AP,IPIV,WORK,INFO)
*
*           Print inverse
*
            IFAIL = 0
            CALL X04DDF(UPLO,'Nonunit',N,AP,'Bracketed','F7.4',
     +                  'Inverse','Integer',RLABS,'Integer',CLABS,80,0,
     +                  IFAIL)
         ELSE
            WRITE (NOUT,*) 'The factor D is singular'
         END IF
      END IF
      STOP
*
      END
```

9.2. Program Data

```
F07QWF Example Program Data
  4                                                              :Value of N
  'L'                                                            :Value of UPLO
 (-0.39,-0.71)
 ( 5.14,-0.64) ( 8.86, 1.81)
 (-7.86,-2.96) (-3.52, 0.58) (-2.83,-0.03)
 ( 3.80, 0.92) ( 5.32,-1.59) (-1.54,-2.86) (-0.56, 0.12)  :End of matrix A
```

9.3. Program Results

```
F07QWF Example Program Results

Inverse
                      1                   2                   3                   4
 1  (-0.1562,-0.1014)
 2  ( 0.0400, 0.1527) ( 0.0946,-0.1475)
 3  ( 0.0550, 0.0845) (-0.0326,-0.1370) (-0.1320,-0.0102)
 4  ( 0.2162,-0.0742) (-0.0995,-0.0461) (-0.1793, 0.1183) (-0.2269, 0.2383)
```

F07TEF (STRTRS/DTRTRS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07TEF (STRTRS/DTRTRS) solves a real triangular system of linear equations with multiple right-hand sides, $AX = B$ or $A^TX = B$.

2. Specification

```
SUBROUTINE F07TEF (UPLO, TRANS, DIAG, N, NRHS, A, LDA, B, LDB, INFO)
ENTRY      strtrs  (UPLO, TRANS, DIAG, N, NRHS, A, LDA, B, LDB, INFO)
INTEGER          N, NRHS, LDA, LDB, INFO
real             A(LDA,*), B(LDB,*)
CHARACTER*1      UPLO, TRANS, DIAG
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine solves a real triangular system of linear equations $AX = B$ or $A^TX = B$.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §3.1.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

[2] HIGHAM, N.J.
The Accuracy of Solutions to Triangular Systems.
SIAM J. Numer. Anal., 5, pp. 1252-1265, 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A is upper or lower triangular as follows:

if UPLO = 'U', then A is upper triangular;

if UPLO = 'L', then A is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: TRANS – CHARACTER*1. *Input*

On entry: indicates the form of the equations as follows:

if TRANS = 'N', then the equations are of the form $AX = B$;

if TRANS = 'T' or 'C', then the equations are of the form $A^TX = B$.

Constraint: TRANS = 'N', 'T' or 'C'.

3: DIAG – CHARACTER*1. *Input*

On entry: indicates whether A is a non-unit or unit triangular matrix as follows:

if DIAG = 'N', then A is a non-unit triangular matrix;

if DIAG = 'U', then A is a unit triangular matrix; the diagonal elements are not referenced and are assumed to be 1.

Constraint: DIAG = 'N' or 'U'.

4: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

5: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS $\geq$ 0.

6: A(LDA,*) – ***real*** array. *Input*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the n by n triangular matrix A. If UPLO = 'U', A is upper triangular and the elements of the array below the diagonal are not referenced; if UPLO = 'L', A is lower triangular and the elements of the array above the diagonal are not referenced. If DIAG = 'U', the diagonal elements of A are not referenced, but are assumed to be 1.

7: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07TEF (STRTRS/DTRTRS) is called.

Constraint: LDA $\geq$ max(1,N).

8: B(LDB,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

On exit: the n by r solution matrix X.

9: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07TEF (STRTRS/DTRTRS) is called.

Constraint: LDB $\geq$ max(1,N).

10: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, a_{ii} is zero and the matrix A is singular.

7. Accuracy

The solutions of triangular systems of equations are usually computed to high accuracy. See Higham [2].

For each right-hand side vector b, the computed solution x is the exact solution of a perturbed system of equations $(A+E)x = b$, where

$$|E| \leq c(n)\varepsilon|A|,$$

$c(n)$ is a modest linear function of n, and ε is the ***machine precision***.

If $\hat{x}$ is the true solution, then the computed solution x satisfies a forward error bound of the form

$$\frac{\|x-\hat{x}\|_\infty}{\|x\|_\infty} \leq c(n)\text{cond}(A,x)\varepsilon, \text{ provided } c(n)\text{cond}(A,x)\varepsilon < 1,$$

where $\text{cond}(A,x) = \||A^{-1}||A||x|\|_\infty / \|x\|_\infty$.

Note that $\text{cond}(A,x) \leq \text{cond}(A) = \||A^{-1}||A|\|_\infty \leq \kappa_\infty(A)$; $\text{cond}(A,x)$ can be much smaller than $\text{cond}(A)$ and it is also possible for $\text{cond}(A^T)$ to be much larger (or smaller) than $\text{cond}(A)$.

Forward and backward error bounds can be computed by calling F07THF (STRRFS/DTRRFS), and an estimate for $\kappa_\infty(A)$ can be obtained by calling F07TGF (STRCON/DTRCON) with NORM = 'I'.

8. Further Comments

The total number of floating-point operations is approximately n^2r.

The complex analogue of this routine is F07TSF (CTRTRS/ZTRTRS).

9. Example

To solve the system of equations $AX = B$, where

$$A = \begin{pmatrix} 4.30 & 0.00 & 0.00 & 0.00 \\ -3.96 & -4.87 & 0.00 & 0.00 \\ 0.40 & 0.31 & -8.02 & 0.00 \\ -0.27 & 0.07 & -5.95 & 0.12 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} -12.90 & -21.50 \\ 16.75 & 14.93 \\ -17.55 & 6.33 \\ -11.04 & 8.09 \end{pmatrix}.$$

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07TEF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, NRHMAX, LDB
      PARAMETER        (NMAX=8,LDA=NMAX,NRHMAX=NMAX,LDB=NMAX)
      CHARACTER        TRANS, DIAG
      PARAMETER        (TRANS='N',DIAG='N')
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             A(LDA,NMAX), B(LDB,NRHMAX)
*     .. External Subroutines ..
      EXTERNAL         strtrs, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07TEF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
*
```

```
*        Compute solution
*
         CALL strtrs(UPLO,TRANS,DIAG,N,NRHS,A,LDA,B,LDB,INFO)
*
*        Print solution
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
            IFAIL = 0
            CALL X04CAF('General',' ',N,NRHS,B,LDB,'Solution(s)',IFAIL)
         ELSE
            WRITE (NOUT,*) 'A is singular'
         END IF
      END IF
      STOP
*
      END
```

9.2. Program Data

```
F07TEF Example Program Data
  4  2                          :Values of N and NRHS
  'L'                           :Value of UPLO
  4.30
 -3.96  -4.87
  0.40   0.31  -8.02
 -0.27   0.07  -5.95   0.12    :End of matrix A
-12.90 -21.50
 16.75  14.93
-17.55   6.33
-11.04   8.09                  :End of matrix B
```

9.3. Program Results

```
 F07TEF Example Program Results

 Solution(s)
                  1           2
  1        -3.0000     -5.0000
  2        -1.0000      1.0000
  3         2.0000     -1.0000
  4         1.0000      6.0000
```

F07TGF (STRCON/DTRCON) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07TGF (STRCON/DTRCON) estimates the condition number of a real triangular matrix.

2. Specification

```
SUBROUTINE F07TGF (NORM, UPLO, DIAG, N, A, LDA, RCOND, WORK, IWORK,
1                  INFO)
ENTRY       strcon (NORM, UPLO, DIAG, N, A, LDA, RCOND, WORK, IWORK,
1                  INFO)
INTEGER         N, LDA, IWORK(*), INFO
real            A(LDA,*), RCOND, WORK(*)
CHARACTER*1     NORM, UPLO, DIAG
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine estimates the condition number of a real triangular matrix A, in either the 1-norm or the infinity-norm:

$$\kappa_1(A) = \|A\|_1\|A^{-1}\|_1 \quad \text{or} \quad \kappa_\infty(A) = \|A\|_\infty\|A^{-1}\|_\infty.$$

Note that $\kappa_\infty(A) = \kappa_1(A^T)$.

Because the condition number is infinite if A is singular, the routine actually returns an estimate of the ***reciprocal*** of the condition number.

The routine computes $\|A\|_1$ or $\|A\|_\infty$ exactly, and uses Higham's implementation of Hager's method [1] to estimate $\|A^{-1}\|_1$ or $\|A^{-1}\|_\infty$.

4. References

[1] HIGHAM, N.J.
FORTRAN Codes for Estimating the One-norm of a Real or Complex Matrix, with Applications to Condition Estimation.
ACM Trans. Math. Softw., 14, pp. 381-396, 1988.

5. Parameters

1: NORM – CHARACTER*1. *Input*

On entry: indicates whether $\kappa_1(A)$ or $\kappa_\infty(A)$ is estimated as follows:

if NORM = '1' or 'O', then $\kappa_1(A)$ is estimated;

if NORM = 'I', then $\kappa_\infty(A)$ is estimated.

Constraint: NORM = '1', 'O' or 'I'.

2: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A is upper or lower triangular as follows:

if UPLO = 'U', then A is upper triangular;

if UPLO = 'L', then A is lower triangular.

Constraint: UPLO = 'U' or 'L'.

3: DIAG – CHARACTER*1. *Input*

On entry: indicates whether A is a non-unit or unit triangular matrix as follows:

if DIAG = 'N', then A is a non-unit triangular matrix;

if DIAG = 'U', then A is a unit triangular matrix; the diagonal elements are not referenced and are assumed to be 1.

Constraint: DIAG = 'N' or 'U'.

4: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

5: A(LDA,*) – ***real*** array. *Input*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the n by n triangular matrix A. If UPLO = 'U', A is upper triangular and the elements of the array below the diagonal are not referenced; if UPLO = 'L', A is lower triangular and the elements of the array above the diagonal are not referenced. If DIAG = 'U', the diagonal elements of A are not referenced, but are assumed to be 1.

6: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07TGF (STRCON/DTRCON) is called.

Constraint: LDA $\geq$ max(1,N).

7: RCOND – ***real***. *Output*

On exit: an estimate of the reciprocal of the condition number of A. RCOND is set to zero if exact singularity is detected or if the estimate underflows. If RCOND is less than ***machine precision***, then A is singular to working precision.

8: WORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,3*N).

9: IWORK(*) – INTEGER array. *Workspace*

Note: the dimension of the array IWORK must be at least max(1,N).

10: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed estimate RCOND is never less than the true value ρ, and in practice is nearly always less than 10ρ, although examples can be constructed where RCOND is much larger.

8. Further Comments

A call to this routine involves solving a number of systems of linear equations of the form $Ax = b$ or $A^Tx = b$; the number is usually 4 or 5 and never more than 11. Each solution involves approximately n^2 floating-point operations but takes considerably longer than a call to F07TEF (STRTRS/DTRTRS) with 1 right-hand side, because extra care is taken to avoid overflow when A is approximately singular.

The complex analogue of this routine is F07TUF (CTRCON/ZTRCON).

9. Example

To estimate the condition number in the 1-norm of the matrix A, where

$$A = \begin{pmatrix} 4.30 & 0.00 & 0.00 & 0.00 \\ -3.96 & -4.87 & 0.00 & 0.00 \\ 0.40 & 0.31 & -8.02 & 0.00 \\ -0.27 & 0.07 & -5.95 & 0.12 \end{pmatrix}.$$

The true condition number in the 1-norm is 116.41.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07TGF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA
      PARAMETER        (NMAX=8,LDA=NMAX)
      CHARACTER        NORM, DIAG
      PARAMETER        (NORM='1',DIAG='N')
*     .. Local Scalars ..
      real             RCOND
      INTEGER          I, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             A(LDA,NMAX), WORK(3*NMAX)
      INTEGER          IWORK(NMAX)
*     .. External Functions ..
      real             X02AJF
      EXTERNAL         X02AJF
*     .. External Subroutines ..
      EXTERNAL         strcon
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07TGF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
*
*        Estimate condition number
*
         CALL strcon(NORM,UPLO,DIAG,N,A,LDA,RCOND,WORK,IWORK,INFO)
*
         WRITE (NOUT,*)
         IF (RCOND.GE.X02AJF()) THEN
            WRITE (NOUT,99999) 'Estimate of condition number =',
     +        1.0e0/RCOND
         ELSE
```

```
            WRITE (NOUT,*) 'A is singular to working precision'
          END IF
        END IF
        STOP
*
99999 FORMAT (1X,A,1P,e10.2)
      END
```

9.2. Program Data

```
F07TGF Example Program Data
  4                                :Value of N
  'L'                              :Value of UPLO
  4.30
 -3.96  -4.87
  0.40   0.31  -8.02
 -0.27   0.07  -5.95   0.12    :End of matrix A
```

9.3. Program Results

```
 F07TGF Example Program Results

 Estimate of condition number =  1.16E+02
```

F07THF (STRRFS/DTRRFS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07THF (STRRFS/DTRRFS) returns error bounds for the solution of a real triangular system of linear equations with multiple right-hand sides, $AX = B$ or $A^TX = B$.

2. Specification

```
SUBROUTINE F07THF (UPLO, TRANS, DIAG, N, NRHS, A, LDA, B, LDB, X, LDX,
1                  FERR, BERR, WORK, IWORK, INFO)
ENTRY       strrfs  (UPLO, TRANS, DIAG, N, NRHS, A, LDA, B, LDB, X, LDX,
1                  FERR, BERR, WORK, IWORK, INFO)
INTEGER          N, NRHS, LDA, LDB, LDX, IWORK(*), INFO
real             A(LDA,*), B(LDB,*), X(LDX,*), FERR(*), BERR(*), WORK(*)
CHARACTER*1      UPLO, TRANS, DIAG
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine returns the backward errors and estimated bounds on the forward errors for the solution of a real triangular system of linear equations with multiple right-hand sides $AX = B$ or $A^TX = B$. The routine handles each right-hand side vector (stored as a column of the matrix B) independently, so we describe the function of the routine in terms of a single right-hand side b and solution x.

Given a computed solution x, the routine computes the *component-wise backward error* β. This is the size of the smallest relative perturbation in each element of A and b such that x is the exact solution of a perturbed system

$$(A+\delta A)x = b + \delta b$$

$$|\delta a_{ij}| \leq \beta |a_{ij}| \text{ and } |\delta b_i| \leq \beta |b_i|.$$

Then the routine estimates a bound for the *component-wise forward error* in the computed solution, defined by:

$$\max_i |x_i - \hat{x}_i| / \max_i |x_i|$$

where $\hat{x}$ is the true solution.

For details of the method, see the Chapter Introduction.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A is upper or lower triangular as follows:

if UPLO = 'U', then A is upper triangular;

if UPLO = 'L', then A is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: TRANS – CHARACTER*1. *Input*

On entry: indicates the form of the equations as follows:

if TRANS = 'N', then the equations are of the form $AX = B$;

if TRANS = 'T' or 'C', then the equations are of the form $A^TX = B$.

Constraint: TRANS = 'N', 'T' or 'C'.

3: DIAG – CHARACTER*1. *Input*

On entry: indicates whether A is a non-unit or unit triangular matrix as follows:

if DIAG = 'N', then A is a non-unit triangular matrix;

if DIAG = 'U', then A is a unit triangular matrix; the diagonal elements are not referenced and are assumed to be 1.

Constraint: DIAG = 'N' or 'U'.

4: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

5: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS $\geq$ 0.

6: A(LDA,*) – ***real*** array. *Input*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the n by n triangular matrix A. If UPLO = 'U', A is upper triangular and the elements of the array below the diagonal are not referenced; if UPLO = 'L', A is lower triangular and the elements of the array above the diagonal are not referenced. If DIAG = 'U', the diagonal elements of A are not referenced, but are assumed to be 1.

7: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07THF (STRRFS/DTRRFS) is called.

Constraint: LDA $\geq$ max(1,N).

8: B(LDB,*) – ***real*** array. *Input*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

9: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07THF (STRRFS/DTRRFS) is called.

Constraint: LDB $\geq$ max(1,N).

10: X(LDX,*) – ***real*** array. *Input*

Note: the second dimension of the array X must be at least max(1,NRHS).

On entry: the n by r solution matrix X, as returned by F07TEF (STRTRS/DTRTRS).

11: LDX – INTEGER. *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which F07THF (STRRFS/DTRRFS) is called.

Constraint: LDX $\geq$ max(1,N).

12: FERR(*) – ***real*** array. *Output*

Note: the dimension of the array FERR must be at least max(1,NRHS).

On exit: FERR(j) contains an estimated error bound for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

13: BERR(*) – ***real*** array. *Output*

Note: the dimension of the array BERR must be at least max(1,NRHS).

On exit: BERR(j) contains the component-wise backward error bound β for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

14: WORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,3*N).

15: IWORK(*) – INTEGER array. *Workspace*

Note: the dimension of the array IWORK must be at least max(1,N).

16: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The bounds returned in FERR are not rigorous, because they are estimated, not computed exactly; but in practice they almost always overestimate the actual error.

8. Further Comments

A call to this routine involves, for each right-hand side, solving a number of systems of linear equations of the form $Ax = b$ or $A^Tx = b$; the number is usually 4 or 5 and never more than 11. Each solution involves approximately n^2 floating-point operations.

The complex analogue of this routine is F07TVF (CTRRFS/ZTRRFS).

9. Example

To solve the system of equations $AX = B$ and to compute forward and backward error bounds, where

$$A = \begin{pmatrix} 4.30 & 0.00 & 0.00 & 0.00 \\ -3.96 & -4.87 & 0.00 & 0.00 \\ 0.40 & 0.31 & -8.02 & 0.00 \\ -0.27 & 0.07 & -5.95 & 0.12 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} -12.90 & -21.50 \\ 16.75 & 14.93 \\ -17.55 & 6.33 \\ -11.04 & 8.09 \end{pmatrix}.$$

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07THF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, NRHMAX, LDA, LDB, LDX
      PARAMETER        (NMAX=8,NRHMAX=NMAX,LDA=NMAX,LDB=NMAX,LDX=NMAX)
      CHARACTER        TRANS, DIAG
      PARAMETER        (TRANS='N',DIAG='N')
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             A(LDA,NMAX), B(LDB,NRHMAX), BERR(NRHMAX),
     +                 FERR(NRHMAX), WORK(3*NMAX), X(LDX,NMAX)
      INTEGER          IWORK(NMAX)
*     .. External Subroutines ..
      EXTERNAL         strrfs, strtrs, F06QFF, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07THF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file, and copy B to X
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
*
         CALL F06QFF('General',N,NRHS,B,LDB,X,LDX)
*
*        Compute solution in the array X
*
         CALL strtrs(UPLO,TRANS,DIAG,N,NRHS,A,LDA,X,LDX,INFO)
*
*        Compute backward errors and estimated bounds on the
*        forward errors
*
         CALL strrfs(UPLO,TRANS,DIAG,N,NRHS,A,LDA,B,LDB,X,LDX,FERR,BERR,
     +               WORK,IWORK,INFO)
*
*        Print solution
*
         WRITE (NOUT,*)
         IFAIL = 0
         CALL X04CAF('General',' ',N,NRHS,X,LDX,'Solution(s)',IFAIL)
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Backward errors (machine-dependent)'
         WRITE (NOUT,99999) (BERR(J),J=1,NRHS)
         WRITE (NOUT,*)
     +     'Estimated forward error bounds (machine-dependent)'
         WRITE (NOUT,99999) (FERR(J),J=1,NRHS)
      END IF
      STOP
*
99999 FORMAT ((3X,1P,7e11.1))
      END
```

9.2. Program Data

```
F07THF Example Program Data
  4  2                          :Values of N and NRHS
  'L'                           :Value of UPLO
  4.30
 -3.96  -4.87
  0.40   0.31  -8.02
 -0.27   0.07  -5.95   0.12    :End of matrix A
-12.90 -21.50
 16.75  14.93
-17.55   6.33
-11.04   8.09                   :End of matrix B
```

9.3. Program Results

```
F07THF Example Program Results

Solution(s)
                1           2
 1      -3.0000     -5.0000
 2      -1.0000      1.0000
 3       2.0000     -1.0000
 4       1.0000      6.0000

Backward errors (machine-dependent)
       6.9E-17     0.0E+00
Estimated forward error bounds (machine-dependent)
       8.3E-14     2.6E-14
```

F07TJF (STRTRI/DTRTRI) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07TJF (STRTRI/DTRTRI) computes the inverse of a real triangular matrix.

2. Specification

```
SUBROUTINE F07TJF (UPLO, DIAG, N, A, LDA, INFO)
ENTRY      strtri  (UPLO, DIAG, N, A, LDA, INFO)
INTEGER            N, LDA, INFO
real               A(LDA,*)
CHARACTER*1        UPLO, DIAG
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine forms the inverse of a real triangular matrix A. Note that the inverse of an upper (lower) triangular matrix is also upper (lower) triangular.

4. References

[1] DU CROZ, J.J. and HIGHAM, N.J.
Stability of Methods for Matrix Inversion.
LAPACK Working Note No. 27, University of Tennessee, Knoxville, 1990.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A is upper or lower triangular as follows:

if UPLO = 'U', then A is upper triangular;

if UPLO = 'L', then A is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: DIAG – CHARACTER*1. *Input*

On entry: indicates whether A is a non-unit or unit triangular matrix as follows:

if DIAG = 'N', then A is a non-unit triangular matrix;

if DIAG = 'U', then A is a unit triangular matrix; the diagonal elements are not referenced and are assumed to be 1.

Constraint: DIAG = 'N' or 'U'.

3: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: $N \geq 0$.

4: A(LDA,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the n by n triangular matrix A. If UPLO = 'U', A is upper triangular and the elements of the array below the diagonal are not referenced; if UPLO = 'L', A is lower triangular and the elements of the array above the diagonal are not referenced. If DIAG = 'U', the diagonal elements of A are not referenced, but are assumed to be 1.

On exit: A is overwritten by A^{-1}, using the same storage format as described above.

5: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07TJF (STRTRI/DTRTRI) is called.

Constraint: LDA ≥ max(1,N).

6: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, a_{ii} is zero and the matrix A is singular.

7. Accuracy

The computed inverse X satisfies

$$|XA-I| \leq c(n)\varepsilon|X||A|,$$

where $c(n)$ is a modest linear function of n, and ε is the ***machine precision***.

Note that a similar bound for $|AX-I|$ cannot be guaranteed, although it is almost always satisfied.

The computed inverse satisfies the forward error bound

$$|X-A^{-1}| \leq c(n)\varepsilon|A^{-1}||A||X|.$$

See Du Croz and Higham [1].

8. Further Comments

The total number of floating-point operations is approximately $\frac{1}{3}n^3$.

The complex analogue of this routine is F07TWF (CTRTRI/ZTRTRI).

9. Example

To compute the inverse of the matrix A, where

$$A = \begin{pmatrix} 4.30 & 0.00 & 0.00 & 0.00 \\ -3.96 & -4.87 & 0.00 & 0.00 \\ 0.40 & 0.31 & -8.02 & 0.00 \\ -0.27 & 0.07 & -5.95 & 0.12 \end{pmatrix}.$$

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07TJF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA
      PARAMETER        (NMAX=8,LDA=NMAX)
      CHARACTER        DIAG
      PARAMETER        (DIAG='N')
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
      CHARACTER        UPLO
```

```
*     .. Local Arrays ..
      real               A(LDA,NMAX)
*     .. External Subroutines ..
      EXTERNAL           strtri, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07TJF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
*
*        Compute inverse of A
*
         CALL strtri(UPLO,DIAG,N,A,LDA,INFO)
*
*        Print inverse
*
         WRITE (NOUT,*)
         IFAIL = 0
         CALL X04CAF(UPLO,DIAG,N,N,A,LDA,'Inverse',IFAIL)
      END IF
      STOP
*
      END
```

9.2. Program Data

```
F07TJF Example Program Data
  4                              :Value of N
  'L'                            :Value of UPLO
  4.30
 -3.96  -4.87
  0.40   0.31  -8.02
 -0.27   0.07  -5.95   0.12     :End of matrix A
```

9.3. Program Results

```
 F07TJF Example Program Results

 Inverse
              1          2          3          4
  1      0.2326
  2     -0.1891    -0.2053
  3      0.0043    -0.0079    -0.1247
  4      0.8463    -0.2738    -6.1825     8.3333
```

F07TSF (CTRTRS/ZTRTRS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07TSF (CTRTRS/ZTRTRS) solves a complex triangular system of linear equations with multiple right-hand sides, $AX = B$, $A^TX = B$ or $A^HX = B$.

2. Specification

```
SUBROUTINE F07TSF (UPLO, TRANS, DIAG, N, NRHS, A, LDA, B, LDB, INFO)
ENTRY      ctrtrs  (UPLO, TRANS, DIAG, N, NRHS, A, LDA, B, LDB, INFO)
INTEGER          N, NRHS, LDA, LDB, INFO
complex          A(LDA,*), B(LDB,*)
CHARACTER*1      UPLO, TRANS, DIAG
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine solves a complex triangular system of linear equations $AX = B$, $A^TX = B$ or $A^HX = B$.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §3.1.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

[2] HIGHAM, N.J.
The Accuracy of Solutions to Triangular Systems.
SIAM J. Numer. Anal. 5, pp. 1252-1265, 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A is upper or lower triangular as follows:

if UPLO = 'U', then A is upper triangular;

if UPLO = 'L', then A is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: TRANS – CHARACTER*1. *Input*

On entry: indicates the form of the equations as follows:

if TRANS = 'N', then the equations are of the form $AX = B$;

if TRANS = 'T', then the equations are of the form $A^TX = B$;

if TRANS = 'C', then the equations are of the form $A^HX = B$.

Constraint: TRANS = 'N', 'T' or 'C'.

3: DIAG – CHARACTER*1. *Input*

On entry: indicates whether A is a non-unit or unit triangular matrix as follows:

if DIAG = 'N', then A is a non-unit triangular matrix;

if DIAG = 'U', then A is a unit triangular matrix; the diagonal elements are not referenced and are assumed to be 1.

Constraint: DIAG = 'N' or 'U'.

4: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

5: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS $\geq$ 0.

6: A(LDA,*) – ***complex*** array. *Input*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the n by n triangular matrix A. If UPLO = 'U', A is upper triangular and the elements of the array below the diagonal are not referenced; if UPLO = 'L', A is lower triangular and the elements of the array above the diagonal are not referenced. If DIAG = 'U', the diagonal elements of A are not referenced, but are assumed to be 1.

7: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07TSF (CTRTRS/ZTRTRS) is called.

Constraint: LDA $\geq$ max(1,N).

8: B(LDB,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

On exit: the n by r solution matrix X.

9: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07TSF (CTRTRS/ZTRTRS) is called.

Constraint: LDB $\geq$ max(1,N).

10: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, a_{ii} is zero and the matrix A is singular.

7. Accuracy

The solutions of triangular systems of equations are usually computed to high accuracy. See Higham [2].

For each right-hand side vector b, the computed solution x is the exact solution of a perturbed system of equations $(A+E)x = b$, where

$$|E| \leq c(n)\varepsilon|A|,$$

$c(n)$ is a modest linear function of n, and ε is the ***machine precision***.

If $\hat{x}$ is the true solution, then the computed solution x satisfies a forward error bound of the form

$$\frac{\|x-\hat{x}\|_\infty}{\|x\|_\infty} \leq c(n)\text{cond}(A,x)\varepsilon, \text{ provided } c(n)\text{cond}(A,x)\varepsilon < 1,$$

where $\text{cond}(A,x) = \||A^{-1}||A||x|\|_\infty / \|x\|_\infty$.

Note that $\text{cond}(A,x) \leq \text{cond}(A) = \||A^{-1}||A|\|_\infty \leq \kappa_\infty(A)$; $\text{cond}(A,x)$ can be much smaller than $\text{cond}(A)$ and it is also possible for $\text{cond}(A^H)$, which is the same as $\text{cond}(A^T)$, to be much larger (or smaller) than $\text{cond}(A)$.

Forward and backward error bounds can be computed by calling F07TVF (CTRRFS/ZTRRFS), and an estimate for $\kappa_\infty(A)$ can be obtained by calling F07TUF (CTRCON/ZTRCON) with NORM = 'I'.

8. Further Comments

The total number of real floating-point operations is approximately $4n^2r$.

The real analogue of this routine is F07TEF (STRTRS/DTRTRS).

9. Example

To solve the system of equations $AX = B$, where

$$A = \begin{pmatrix} 4.78 + 4.56i & 0.00 + 0.00i & 0.00 + 0.00i & 0.00 + 0.00i \\ 2.00 - 0.30i & -4.11 + 1.25i & 0.00 + 0.00i & 0.00 + 0.00i \\ 2.89 - 1.34i & 2.36 - 4.25i & 4.15 + 0.80i & 0.00 + 0.00i \\ -1.89 + 1.15i & 0.04 - 3.69i & -0.02 + 0.46i & 0.33 - 0.26i \end{pmatrix} \text{ and}$$

$$B = \begin{pmatrix} -14.78 - 32.36i & -18.02 + 28.46i \\ 2.98 - 2.14i & 14.22 + 15.42i \\ -20.96 + 17.06i & 5.62 + 35.89i \\ 9.54 + 9.91i & -16.46 - 1.73i \end{pmatrix}.$$

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07TSF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, NRHMAX, LDB
      PARAMETER        (NMAX=8,LDA=NMAX,NRHMAX=NMAX,LDB=NMAX)
      CHARACTER        TRANS, DIAG
      PARAMETER        (TRANS='N',DIAG='N')
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          A(LDA,NMAX), B(LDB,NRHMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         X04DBF, ctrtrs
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07TSF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX) THEN
*
```

```
*        Read A and B from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
*
*        Compute solution
*
         CALL ctrtrs(UPLO,TRANS,DIAG,N,NRHS,A,LDA,B,LDB,INFO)
*
*        Print solution
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
            IFAIL = 0
            CALL X04DBF('General',' ',N,NRHS,B,LDB,'Bracketed','F7.4',
     +                  'Solution(s)','Integer',RLABS,'Integer',CLABS,
     +                  80,0,IFAIL)
         ELSE
            WRITE (NOUT,*) 'A is singular'
         END IF
      END IF
      STOP
*
      END
```

9.2. Program Data

```
F07TSF Example Program Data
  4  2                                                        :Values of N and NRHS
  'L'                                                            :Value of UPLO
 ( 4.78, 4.56)
 ( 2.00,-0.30) (-4.11, 1.25)
 ( 2.89,-1.34) ( 2.36,-4.25) ( 4.15, 0.80)
 (-1.89, 1.15) ( 0.04,-3.69) (-0.02, 0.46) ( 0.33,-0.26)   :End of matrix A
 (-14.78,-32.36) (-18.02, 28.46)
 (  2.98, -2.14) ( 14.22, 15.42)
 (-20.96, 17.06) (  5.62, 35.89)
 (  9.54,  9.91) (-16.46, -1.73)                                :End of matrix B
```

9.3. Program Results

```
F07TSF Example Program Results

Solution(s)
                        1                       2
 1  (-5.0000,-2.0000) ( 1.0000, 5.0000)
 2  (-3.0000,-1.0000) (-2.0000,-2.0000)
 3  ( 2.0000, 1.0000) ( 3.0000, 4.0000)
 4  ( 4.0000, 3.0000) ( 4.0000,-3.0000)
```

F07TUF (CTRCON/ZTRCON) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07TUF (CTRCON/ZTRCON) estimates the condition number of a complex triangular matrix.

2. Specification

```
      SUBROUTINE F07TUF (NORM, UPLO, DIAG, N, A, LDA, RCOND, WORK, RWORK,
     1                   INFO)
      ENTRY       ctrcon  (NORM, UPLO, DIAG, N, A, LDA, RCOND, WORK, RWORK,
     1                   INFO)
      INTEGER          N, LDA, INFO
      real             RCOND, RWORK(*)
      complex          A(LDA,*), WORK(*)
      CHARACTER*1      NORM, UPLO, DIAG
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine estimates the condition number of a complex triangular matrix A, in either the 1-norm or the infinity-norm:

$$\kappa_1(A) = \|A\|_1\|A^{-1}\|_1 \quad \text{or} \quad \kappa_\infty(A) = \|A\|_\infty\|A^{-1}\|_\infty.$$

Note that $\kappa_\infty(A) = \kappa_1(A^T)$.

Because the condition number is infinite if A is singular, the routine actually returns an estimate of the **reciprocal** of the condition number.

The routine computes $\|A\|_1$ or $\|A\|_\infty$ exactly, and uses Higham's implementation of Hager's method [1] to estimate $\|A^{-1}\|_1$ or $\|A^{-1}\|_\infty$.

4. References

[1] HIGHAM, N.J.
FORTRAN Codes for Estimating the One-norm of a Real or Complex Matrix, with Applications to Condition Estimation.
ACM Trans. Math. Softw., 14, pp. 381-396, 1988.

5. Parameters

1: NORM – CHARACTER*1. *Input*

On entry: indicates whether $\kappa_1(A)$ or $\kappa_\infty(A)$ is estimated as follows:

if NORM = '1' or 'O', then $\kappa_1(A)$ is estimated;

if NORM = 'I', then $\kappa_\infty(A)$ is estimated.

Constraint: NORM = '1', 'O' or 'I'.

2: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A is upper or lower triangular as follows:

if UPLO = 'U', then A is upper triangular;

if UPLO = 'L', then A is lower triangular.

Constraint: UPLO = 'U' or 'L'.

3: DIAG – CHARACTER*1. *Input*

On entry: indicates whether A is a non-unit or unit triangular matrix as follows:

if DIAG = 'N', then A is a non-unit triangular matrix;

if DIAG = 'U', then A is a unit triangular matrix; the diagonal elements are not referenced and are assumed to be 1.

Constraint: DIAG = 'N' or 'U'.

4: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

5: A(LDA,*) – ***complex*** array. *Input*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the n by n triangular matrix A. If UPLO = 'U', A is upper triangular and the elements of the array below the diagonal are not referenced; if UPLO = 'L', A is lower triangular and the elements of the array above the diagonal are not referenced. If DIAG = 'U', the diagonal elements of A are not referenced, but are assumed to be 1.

6: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07TUF (CTRCON/ZTRCON) is called.

Constraint: LDA $\geq$ max(1,N).

7: RCOND – ***real***. *Output*

On exit: an estimate of the reciprocal of the condition number of A. RCOND is set to zero if exact singularity is detected or the estimate underflows. If RCOND is less than ***machine precision***, then A is singular to working precision.

8: WORK(*) – ***complex*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,2*N).

9: RWORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array RWORK must be at least max(1,N).

10: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed estimate RCOND is never less than the true value ρ, and in practice is nearly always less than 10ρ, although examples can be constructed where RCOND is much larger.

8. Further Comments

A call to this routine involves solving a number of systems of linear equations of the form $Ax = b$ or $A^H x = b$; the number is usually 5 and never more than 11. Each solution involves approximately $4n^2$ real floating-point operations but takes considerably longer than a call to F07TSF (CTRTRS/ZTRTRS) with 1 right-hand side, because extra care is taken to avoid overflow when A is approximately singular.

The real analogue of this routine is F07TGF (STRCON/DTRCON).

9. Example

To estimate the condition number in the 1-norm of the matrix A, where

$$A = \begin{pmatrix} 4.78 + 4.56i & 0.00 + 0.00i & 0.00 + 0.00i & 0.00 + 0.00i \\ 2.00 - 0.30i & -4.11 + 1.25i & 0.00 + 0.00i & 0.00 + 0.00i \\ 2.89 - 1.34i & 2.36 - 4.25i & 4.15 + 0.80i & 0.00 + 0.00i \\ -1.89 + 1.15i & 0.04 - 3.69i & -0.02 + 0.46i & 0.33 - 0.26i \end{pmatrix}.$$

The true condition number in the 1-norm is 70.27.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07TUF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA
      PARAMETER        (NMAX=8,LDA=NMAX)
      CHARACTER        NORM, DIAG
      PARAMETER        (NORM='1',DIAG='N')
*     .. Local Scalars ..
      real             RCOND
      INTEGER          I, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          A(LDA,NMAX), WORK(2*NMAX)
      real             RWORK(NMAX)
*     .. External Functions ..
      real             X02AJF
      EXTERNAL         X02AJF
*     .. External Subroutines ..
      EXTERNAL         ctrcon
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07TUF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
*
*        Estimate condition number
*
         CALL ctrcon(NORM,UPLO,DIAG,N,A,LDA,RCOND,WORK,RWORK,INFO)
*
         WRITE (NOUT,*)
         IF (RCOND.GE.X02AJF()) THEN
            WRITE (NOUT,99999) 'Estimate of condition number =',
     +        1.0e0/RCOND
         ELSE
```

```
            WRITE (NOUT,*) 'A is singular to working precision'
         END IF
      END IF
      STOP
*
99999 FORMAT (1X,A,1P,e10.2)
      END
```

9.2. Program Data

```
F07TUF Example Program Data
  4                                                                  :Value of N
  'L'                                                                :Value of UPLO
 ( 4.78, 4.56)
 ( 2.00,-0.30) (-4.11, 1.25)
 ( 2.89,-1.34) ( 2.36,-4.25) ( 4.15, 0.80)
 (-1.89, 1.15) ( 0.04,-3.69) (-0.02, 0.46) ( 0.33,-0.26)  :End of matrix A
```

9.3. Program Results

```
 F07TUF Example Program Results

 Estimate of condition number =  3.74E+01
```

F07TVF (CTRRFS/ZTRRFS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07TVF (CTRRFS/ZTRRFS) returns error bounds for the solution of a complex triangular system of linear equations with multiple right-hand sides, $AX = B$, $A^TX = B$ or $A^HX = B$.

2. Specification

```
 SUBROUTINE F07TVF (UPLO, TRANS, DIAG, N, NRHS, A, LDA, B, LDB, X, LDX,
1                   FERR, BERR, WORK, RWORK, INFO)
 ENTRY       ctrrfs  (UPLO, TRANS, DIAG, N, NRHS, A, LDA, B, LDB, X, LDX,
1                   FERR, BERR, WORK, RWORK, INFO)
 INTEGER          N, NRHS, LDA, LDB, LDX, INFO
 real             FERR(*), BERR(*), RWORK(*)
 complex          A(LDA,*), B(LDB,*), X(LDX,*), WORK(*)
 CHARACTER*1      UPLO, TRANS, DIAG
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine returns the backward errors and estimated bounds on the forward errors for the solution of a complex triangular system of linear equations with multiple right-hand sides $AX = B$, $A^TX = B$ or $A^HX = B$. The routine handles each right-hand side vector (stored as a column of the matrix B) independently, so we describe the function of the routine in terms of a single right-hand side b and solution x.

Given a computed solution x, the routine computes the *component-wise backward error* β. This is the size of the smallest relative perturbation in each element of A and b such that x is the exact solution of a perturbed system

$$(A+\delta A)x = b + \delta b$$

$$|\delta a_{ij}| \leq \beta |a_{ij}| \quad \text{and} \quad |\delta b_i| \leq \beta |b_i|.$$

Then the routine estimates a bound for the *component-wise forward error* in the computed solution, defined by:

$$\max_i |x_i - \hat{x}_i| / \max_i |x_i|$$

where $\hat{x}$ is the true solution.

For details of the method, see the Chapter Introduction.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A is upper or lower triangular as follows:

if UPLO = 'U', then A is upper triangular;

if UPLO = 'L', then A is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: TRANS – CHARACTER*1. *Input*

On entry: indicates the form of the equations as follows:

if TRANS = 'N', then the equations are of the form $AX = B$;

if TRANS = 'T', then the equations are of the form $A^TX = B$;

if TRANS = 'C', then the equations are of the form $A^HX = B$.

Constraint: TRANS = 'N', 'T' or 'C'.

3: DIAG – CHARACTER*1. *Input*

On entry: indicates whether A is a non-unit or unit triangular matrix as follows:

if DIAG = 'N', then A is a non-unit triangular matrix;

if DIAG = 'U', then A is a unit triangular matrix; the diagonal elements are not referenced and are assumed to be 1.

Constraint: DIAG = 'N' or 'U'.

4: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

5: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS $\geq$ 0.

6: A(LDA,*) – ***complex*** array. *Input*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the n by n triangular matrix A. If UPLO = 'U', A is upper triangular and the elements of the array below the diagonal are not referenced; if UPLO = 'L', A is lower triangular and the elements of the array above the diagonal are not referenced. If DIAG = 'U', the diagonal elements of A are not referenced, but are assumed to be 1.

7: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07TVF (CTRRFS/ZTRRFS) is called.

Constraint: LDA $\geq$ max(1,N).

8: B(LDB,*) – ***complex*** array. *Input*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

9: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07TVF (CTRRFS/ZTRRFS) is called.

Constraint: LDB $\geq$ max(1,N).

10: X(LDX,*) – ***complex*** array. *Input*

Note: the second dimension of the array X must be at least max(1,NRHS).

On entry: the n by r solution matrix X, as returned by F07TSF (CTRTRS/ZTRTRS).

11: LDX – INTEGER. *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which F07TVF (CTRRFS/ZTRRFS) is called.

Constraint: LDX ≥ max(1,N).

12: FERR(*) – ***real*** array. *Output*

Note: the dimension of the array FERR must be at least max(1,NRHS).

On exit: FERR(j) contains an estimated error bound for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

13: BERR(*) – ***real*** array. *Output*

Note: the dimension of the array BERR must be at least max(1,NRHS).

On exit: BERR(j) contains the component-wise backward error bound β for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

14: WORK(*) – ***complex*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,2*N).

15: RWORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array RWORK must be at least max(1,N).

16: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The bounds returned in FERR are not rigorous, because they are estimated, not computed exactly; but in practice they almost always overestimate the actual error.

8. Further Comments

A call to this routine, for each right-hand side, involves solving a number of systems of linear equations of the form $Ax = b$ or $A^H x = b$; the number is usually 5 and never more than 11. Each solution involves approximately $4n^2$ real floating-point operations.

The real analogue of this routine is F07THF (STRRFS/DTRRFS).

9. Example

To solve the system of equations $AX = B$ and to compute forward and backward error bounds, where

$$A = \begin{pmatrix} 4.78 + 4.56i & 0.00 + 0.00i & 0.00 + 0.00i & 0.00 + 0.00i \\ 2.00 - 0.30i & -4.11 + 1.25i & 0.00 + 0.00i & 0.00 + 0.00i \\ 2.89 - 1.34i & 2.36 - 4.25i & 4.15 + 0.80i & 0.00 + 0.00i \\ -1.89 + 1.15i & 0.04 - 3.69i & -0.02 + 0.46i & 0.33 - 0.26i \end{pmatrix} \text{ and}$$

$$B = \begin{pmatrix} -14.78 - 32.36i & -18.02 + 28.46i \\ 2.98 - 2.14i & 14.22 + 15.42i \\ -20.96 + 17.06i & 5.62 + 35.89i \\ 9.54 + 9.91i & -16.46 - 1.73i \end{pmatrix}.$$

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07TVF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, NRHMAX, LDA, LDB, LDX
      PARAMETER        (NMAX=8,NRHMAX=NMAX,LDA=NMAX,LDB=NMAX,LDX=NMAX)
      CHARACTER        TRANS, DIAG
      PARAMETER        (TRANS='N',DIAG='N')
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          A(LDA,NMAX), B(LDB,NRHMAX), WORK(2*NMAX),
     +                 X(LDX,NMAX)
      real             BERR(NRHMAX), FERR(NRHMAX), RWORK(NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         F06TFF, X04DBF, ctrrfs, ctrtrs
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07TVF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file, and copy B to X
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
         CALL F06TFF('General',N,NRHS,B,LDB,X,LDX)
*
*        Compute solution in the array X
*
         CALL ctrtrs(UPLO,TRANS,DIAG,N,NRHS,A,LDA,X,LDX,INFO)
*
*        Compute backward errors and estimated bounds on the
*        forward errors
*
         CALL ctrrfs(UPLO,TRANS,DIAG,N,NRHS,A,LDA,B,LDB,X,LDX,FERR,BERR,
     +               WORK,RWORK,INFO)
*
*        Print solution
*
         WRITE (NOUT,*)
         IFAIL = 0
         CALL X04DBF('General',' ',N,NRHS,X,LDX,'Bracketed','F7.4',
     +               'Solution(s)','Integer',RLABS,'Integer',CLABS,80,0,
     +               IFAIL)
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Backward errors (machine-dependent)'
         WRITE (NOUT,99999) (BERR(J),J=1,NRHS)
         WRITE (NOUT,*)
     +     'Estimated forward error bounds (machine-dependent)'
```

```
         WRITE (NOUT,99999) (FERR(J),J=1,NRHS)
      END IF
      STOP
*
99999 FORMAT ((5X,1P,4(e11.1,7X)))
      END
```

9.2. Program Data

```
F07TVF Example Program Data
  4  2                                                        :Values of N and NRHS
  'L'                                                            :Value of UPLO
 ( 4.78, 4.56)
 ( 2.00,-0.30) (-4.11, 1.25)
 ( 2.89,-1.34) ( 2.36,-4.25) ( 4.15, 0.80)
 (-1.89, 1.15) ( 0.04,-3.69) (-0.02, 0.46) ( 0.33,-0.26)   :End of matrix A
 (-14.78,-32.36) (-18.02, 28.46)
 (  2.98, -2.14) ( 14.22, 15.42)
 (-20.96, 17.06) (  5.62, 35.89)
 (  9.54,  9.91) (-16.46, -1.73)                                :End of matrix B
```

9.3. Program Results

```
F07TVF Example Program Results

Solution(s)
                     1                    2
 1  (-5.0000,-2.0000) ( 1.0000, 5.0000)
 2  (-3.0000,-1.0000) (-2.0000,-2.0000)
 3  ( 2.0000, 1.0000) ( 3.0000, 4.0000)
 4  ( 4.0000, 3.0000) ( 4.0000,-3.0000)

Backward errors (machine-dependent)
        4.2E-17           7.5E-18
Estimated forward error bounds (machine-dependent)
        2.9E-14           3.2E-14
```

F07TWF (CTRTRI/ZTRTRI) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07TWF (CTRTRI/ZTRTRI) computes the inverse of a complex triangular matrix.

2. Specification

```
SUBROUTINE F07TWF (UPLO, DIAG, N, A, LDA, INFO)
ENTRY      ctrtri  (UPLO, DIAG, N, A, LDA, INFO)
INTEGER          N, LDA, INFO
complex          A(LDA,*)
CHARACTER*1      UPLO, DIAG
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine forms the inverse of a complex triangular matrix A. Note that the inverse of an upper (lower) triangular matrix is also upper (lower) triangular.

4. References

[1] DU CROZ, J.J. and HIGHAM, N.J.
Stability of Methods for Matrix Inversion.
LAPACK Working Note No. 27, University of Tennessee, Knoxville, 1990.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A is upper or lower triangular as follows:

if UPLO = 'U', then A is upper triangular;

if UPLO = 'L', then A is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: DIAG – CHARACTER*1. *Input*

On entry: indicates whether A is a non-unit or unit triangular matrix as follows:

if DIAG = 'N', then A is a non-unit triangular matrix;

if DIAG = 'U', then A is a unit triangular matrix; the diagonal elements are not referenced and are assumed to be 1.

Constraint: DIAG = 'N' or 'U'.

3: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: $N \geq 0$.

4: A(LDA,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the n by n triangular matrix A. If UPLO = 'U', A is upper triangular and the elements of the array below the diagonal are not referenced; if UPLO = 'L', A is lower triangular and the elements of the array above the diagonal are not referenced. If DIAG = 'U', the diagonal elements of A are not referenced, but are assumed to be 1.

On exit: A is overwritten by A^{-1}, using the same storage format as described above.

5: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F07TWF (CTRTRI/ZTRTRI) is called.

Constraint: LDA $\geq$ max(1,N).

6: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, a_{ii} is zero and the matrix A is singular.

7. Accuracy

The computed inverse X satisfies

$$|XA-I| \leq c(n)\varepsilon|X||A|,$$

where $c(n)$ is a modest linear function of n, and ε is the ***machine precision***.

Note that a similar bound for $|AX-I|$ cannot be guaranteed, although it is almost always satisfied.

The computed inverse satisfies the forward error bound

$$|X-A^{-1}| \leq c(n)\varepsilon|A^{-1}||A||X|.$$

See Du Croz and Higham [1].

8. Further Comments

The total number of real floating-point operations is approximately $\frac{4}{3}n^3$.

The real analogue of this routine is F07TJF (STRTRI/DTRTRI).

9. Example

To compute the inverse of the matrix A, where

$$A = \begin{pmatrix} 4.78 + 4.56i & 0.00 + 0.00i & 0.00 + 0.00i & 0.00 + 0.00i \\ 2.00 - 0.30i & -4.11 + 1.25i & 0.00 + 0.00i & 0.00 + 0.00i \\ 2.89 - 1.34i & 2.36 - 4.25i & 4.15 + 0.80i & 0.00 + 0.00i \\ -1.89 + 1.15i & 0.04 - 3.69i & -0.02 + 0.46i & 0.33 - 0.26i \end{pmatrix}.$$

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07TWF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA
      PARAMETER        (NMAX=8,LDA=NMAX)
      CHARACTER        DIAG
      PARAMETER        (DIAG='N')
```

```
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          A(LDA,NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         X04DBF, ctrtri
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07TWF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
*
*        Compute inverse of A
*
         CALL ctrtri(UPLO,DIAG,N,A,LDA,INFO)
*
*        Print inverse
*
         WRITE (NOUT,*)
         IFAIL = 0
         CALL X04DBF(UPLO,DIAG,N,N,A,LDA,'Bracketed','F7.4','Inverse',
     +               'Integer',RLABS,'Integer',CLABS,80,0,IFAIL)
      END IF
      STOP
*
      END
```

9.2. Program Data

```
F07TWF Example Program Data
  4                                                              :Value of N
  'L'                                                            :Value of UPLO
 ( 4.78, 4.56)
 ( 2.00,-0.30) (-4.11, 1.25)
 ( 2.89,-1.34) ( 2.36,-4.25) ( 4.15, 0.80)
 (-1.89, 1.15) ( 0.04,-3.69) (-0.02, 0.46) ( 0.33,-0.26)  :End of matrix A
```

9.3. Program Results

```
 F07TWF Example Program Results

 Inverse
                      1                  2                  3                  4
  1  ( 0.1095,-0.1045)
  2  ( 0.0582,-0.0411) (-0.2227,-0.0677)
  3  ( 0.0032, 0.1905) ( 0.1538,-0.2192) ( 0.2323,-0.0448)
  4  ( 0.7602, 0.2814) ( 1.6184,-1.4346) ( 0.1289,-0.2250) ( 1.8697, 1.4731)
```

F07UEF (STPTRS/DTPTRS) - NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07UEF (STPTRS/DTPTRS) solves a real triangular system of linear equations with multiple right-hand sides, $AX = B$ or $A^TX = B$, using packed storage.

2. Specification

```
SUBROUTINE F07UEF (UPLO, TRANS, DIAG, N, NRHS, AP, B, LDB, INFO)
ENTRY      stptrs  (UPLO, TRANS, DIAG, N, NRHS, AP, B, LDB, INFO)
INTEGER          N, NRHS, LDB, INFO
real             AP(*), B(LDB,*)
CHARACTER*1      UPLO, TRANS, DIAG
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine solves a real triangular system of linear equations $AX = B$ or $A^TX = B$ using packed storage.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §3.1.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

[2] HIGHAM, N.J.
The Accuracy of Solutions to Triangular Systems.
SIAM J. Numer. Anal., 5, pp. 1252-1265, 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A is upper or lower triangular as follows:

if UPLO = 'U', then A is upper triangular;

if UPLO = 'L', then A is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: TRANS – CHARACTER*1. *Input*

On entry: indicates the form of the equations as follows:

if TRANS = 'N', then the equations are of the form $AX = B$;

if TRANS = 'T' or 'C', then the equations are of the form $A^TX = B$.

Constraint: TRANS = 'N', 'T' or 'C'.

3: DIAG – CHARACTER*1. *Input*

On entry: indicates whether A is a non-unit or unit triangular matrix as follows:

if DIAG = 'N', then A is a non-unit triangular matrix;

if DIAG = 'U', then A is a unit triangular matrix; the diagonal elements are not referenced and are assumed to be 1.

Constraint: DIAG = 'N' or 'U'.

4: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

5: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS $\geq$ 0.

6: AP(*) – ***real*** array. *Input*

Note: the dimension of the array AP must be at least max(1,N*(N+1)/2).

On entry: the n by n triangular matrix A, packed by columns. More precisely, if UPLO = 'U', the upper triangle of A must be stored with element a_{ij} in AP(i+j(j−1)/2) for $i \leq j$; if UPLO = 'L', the lower triangle of A must be stored with element a_{ij} in AP(i+(2n−j)(j−1)/2) for $i \geq j$. If DIAG = 'U', the diagonal elements of the matrix are not referenced and are assumed to be 1; the same storage scheme is used whether DIAG = 'N' or 'U'.

7: B(LDB,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

On exit: the n by r solution matrix X.

8: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07UEF (STPTRS/DTPTRS) is called.

Constraint: LDB $\geq$ max(1,N).

9: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, a_{ii} is zero and the matrix A is singular.

7. Accuracy

The solutions of triangular systems of equations are usually computed to high accuracy. See Higham [2].

For each right-hand side vector b, the computed solution x is the exact solution of a perturbed system of equations $(A+E)x = b$, where

$$|E| \leq c(n)\varepsilon|A|,$$

$c(n)$ is a modest linear function of n, and ε is the ***machine precision***.

If $\hat{x}$ is the true solution, then the computed solution x satisfies a forward error bound of the form

$$\frac{\|x-\hat{x}\|_\infty}{\|x\|_\infty} \leq c(n)\text{cond}(A,x)\varepsilon, \text{ provided } c(n)\text{cond}(A,x)\varepsilon < 1,$$

where $\text{cond}(A,x) = \||A^{-1}||A||x|\|_\infty / \|x\|_\infty$.

Note that $\mathrm{cond}(A,x) \leq \mathrm{cond}(A) = \||A^{-1}||A|\|_\infty \leq \kappa_\infty(A)$; $\mathrm{cond}(A,x)$ can be much smaller than $\mathrm{cond}(A)$ and it is also possible for $\mathrm{cond}(A^T)$ to be much larger (or smaller) than $\mathrm{cond}(A)$.

Forward and backward error bounds can be computed by calling F07UHF (STPRFS/DTPRFS), and an estimate for $\kappa_\infty(A)$ can be obtained by calling F07UGF (STPCON/DTPCON) with NORM = 'I'.

8. Further Comments

The total number of floating-point operations is approximately $n^2 r$.

The complex analogue of this routine is F07USF (CTPTRS/ZTPTRS).

9. Example

To solve the system of equations $AX = B$, where

$$A = \begin{pmatrix} 4.30 & 0.00 & 0.00 & 0.00 \\ -3.96 & -4.87 & 0.00 & 0.00 \\ 0.40 & 0.31 & -8.02 & 0.00 \\ -0.27 & 0.07 & -5.95 & 0.12 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} -12.90 & -21.50 \\ 16.75 & 14.93 \\ -17.55 & 6.33 \\ -11.04 & 8.09 \end{pmatrix},$$

using packed storage for A.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07UEF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, NRHMAX, LDB
      PARAMETER        (NMAX=8,NRHMAX=NMAX,LDB=NMAX)
      CHARACTER        TRANS, DIAG
      PARAMETER        (TRANS='N',DIAG='N')
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             AP(NMAX*(NMAX+1)/2), B(LDB,NRHMAX)
*     .. External Subroutines ..
      EXTERNAL         stptrs, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07UEF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
*
*        Compute solution
*
         CALL stptrs(UPLO,TRANS,DIAG,N,NRHS,AP,B,LDB,INFO)
*
```

```
*        Print solution
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
            IFAIL = 0
            CALL X04CAF('General',' ',N,NRHS,B,LDB,'Solution(s)',IFAIL)
         ELSE
            WRITE (NOUT,*) 'A is singular'
         END IF
      END IF
      STOP
*
      END
```

9.2. Program Data

```
F07UEF Example Program Data
  4  2                          :Values of N and NRHS
  'L'                           :Value of UPLO
  4.30
 -3.96  -4.87
  0.40   0.31  -8.02
 -0.27   0.07  -5.95   0.12    :End of matrix A
-12.90 -21.50
 16.75  14.93
-17.55   6.33
-11.04   8.09                   :End of matrix B
```

9.3. Program Results

```
 F07UEF Example Program Results

 Solution(s)
                  1           2
  1       -3.0000     -5.0000
  2       -1.0000      1.0000
  3        2.0000     -1.0000
  4        1.0000      6.0000
```

F07UGF (STPCON/DTPCON) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07UGF (STPCON/DTPCON) estimates the condition number of a real triangular matrix, using packed storage.

2. Specification

```
SUBROUTINE F07UGF (NORM, UPLO, DIAG, N, AP, RCOND, WORK, IWORK, INFO)
ENTRY      stpcon  (NORM, UPLO, DIAG, N, AP, RCOND, WORK, IWORK, INFO)
INTEGER          N, IWORK(*), INFO
real             AP(*), RCOND, WORK(*)
CHARACTER*1      NORM, UPLO, DIAG
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine estimates the condition number of a real triangular matrix A, in either the 1-norm or the infinity-norm, using packed storage:

$$\kappa_1(A) = \|A\|_1\|A^{-1}\|_1 \quad \text{or} \quad \kappa_\infty(A) = \|A\|_\infty\|A^{-1}\|_\infty.$$

Note that $\kappa_\infty(A) = \kappa_1(A^T)$.

Because the condition number is infinite if A is singular, the routine actually returns an estimate of the ***reciprocal*** of the condition number.

The routine computes $\|A\|_1$ or $\|A\|_\infty$ exactly, and uses Higham's implementation of Hager's method [1] to estimate $\|A^{-1}\|_1$ or $\|A^{-1}\|_\infty$.

4. References

[1] HIGHAM, N.J.
FORTRAN Codes for Estimating the One-norm of a Real or Complex Matrix, with Applications to Condition Estimation.
ACM Trans. Math. Softw., 14, pp. 381-396, 1988.

5. Parameters

1: NORM – CHARACTER*1. *Input*

On entry: indicates whether $\kappa_1(A)$ or $\kappa_\infty(A)$ is estimated as follows:

if NORM = '1' or 'O', then $\kappa_1(A)$ is estimated;

if NORM = 'I', then $\kappa_\infty(A)$ is estimated.

Constraint: NORM = '1', 'O' or 'I'.

2: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A is upper or lower triangular as follows:

if UPLO = 'U', then A is upper triangular;

if UPLO = 'L', then A is lower triangular.

Constraint: UPLO = 'U' or 'L'.

3: DIAG – CHARACTER*1. *Input*

On entry: indicates whether A is a non-unit or unit triangular matrix as follows:

if DIAG = 'N', then A is a non-unit triangular matrix;

if DIAG = 'U', then A is a unit triangular matrix; the diagonal elements are not referenced and are assumed to be 1.

Constraint: DIAG = 'N' or 'U'.

4: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

5: AP(*) – ***real*** array. *Input*

Note: the dimension of the array AP must be at least max(1,N*(N+1)/2).

On entry: the n by n triangular matrix A, packed by columns. More precisely, if UPLO = 'U', the upper triangle of A must be stored with element a_{ij} in AP(i+j(j−1)/2) for $i \leq j$; if UPLO = 'L', the lower triangle of A must be stored with element a_{ij} in AP(i+(2n−j)(j−1)/2) for $i \geq j$. If DIAG = 'U', the diagonal elements of the matrix are not referenced and are assumed to be 1; the same storage scheme is used whether DIAG = 'N' or 'U'.

6: RCOND – ***real***. *Output*

On exit: an estimate of the reciprocal of the condition number of A. RCOND is set to zero if exact singularity is detected or if the estimate underflows. If RCOND is less than ***machine precision***, then A is singular to working precision.

7: WORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,3*N).

8: IWORK(*) – INTEGER array. *Workspace*

Note: the dimension of the array IWORK must be at least max(1,N).

9: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed estimate RCOND is never less than the true value ρ, and in practice is nearly always less than 10ρ, although examples can be constructed where RCOND is much larger.

8. Further Comments

A call to this routine involves solving a number of systems of linear equations of the form $Ax = b$ or $A^Tx = b$; the number is usually 4 or 5 and never more than 11. Each solution involves approximately n^2 floating-point operations but takes considerably longer than a call to F07UEF (STPTRS/DTPTRS) with one right-hand side, because extra care is taken to avoid overflow when A is approximately singular.

The complex analogue of this routine is F07UUF (CTPCON/ZTPCON).

9. Example

To estimate the condition number in the 1-norm of the matrix A, where

$$A = \begin{pmatrix} 4.30 & 0.00 & 0.00 & 0.00 \\ -3.96 & -4.87 & 0.00 & 0.00 \\ 0.40 & 0.31 & -8.02 & 0.00 \\ -0.27 & 0.07 & -5.95 & 0.12 \end{pmatrix},$$

using packed storage. The true condition number in the 1-norm is 116.41.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07UGF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=8)
      CHARACTER        NORM, DIAG
      PARAMETER        (NORM='1',DIAG='N')
*     .. Local Scalars ..
      real             RCOND
      INTEGER          I, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             AP(NMAX*(NMAX+1)/2), WORK(3*NMAX)
      INTEGER          IWORK(NMAX)
*     .. External Functions ..
      real             X02AJF
      EXTERNAL         X02AJF
*     .. External Subroutines ..
      EXTERNAL         stpcon
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07UGF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
*
*        Estimate condition number
*
         CALL stpcon(NORM,UPLO,DIAG,N,AP,RCOND,WORK,IWORK,INFO)
*
         WRITE (NOUT,*)
         IF (RCOND.GE.X02AJF()) THEN
            WRITE (NOUT,99999) 'Estimate of condition number =',
     +        1.0e0/RCOND
         ELSE
            WRITE (NOUT,*) 'A is singular to working precision'
         END IF
      END IF
      STOP
*
99999 FORMAT (1X,A,1P,e10.2)
      END
```

9.2. Program Data

```
F07UGF Example Program Data
  4                                 :Value of N
  'L'                               :Value of UPLO
  4.30
 -3.96  -4.87
  0.40   0.31  -8.02
 -0.27   0.07  -5.95   0.12        :End of matrix A
```

9.3. Program Results

```
F07UGF Example Program Results

Estimate of condition number =  1.16E+02
```

F07UHF (STPRFS/DTPRFS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07UHF (STPRFS/DTPRFS) returns error bounds for the solution of a real triangular system of linear equations with multiple right-hand sides, $AX = B$ or $A^TX = B$, using packed storage.

2. Specification

```
 SUBROUTINE F07UHF (UPLO, TRANS, DIAG, N, NRHS, AP, B, LDB, X, LDX, FERR,
1                   BERR, WORK, IWORK, INFO)
 ENTRY       stprfs  (UPLO, TRANS, DIAG, N, NRHS, AP, B, LDB, X, LDX, FERR,
1                   BERR, WORK, IWORK, INFO)
 INTEGER          N, NRHS, LDB, LDX, IWORK(*), INFO
 real             AP(*), B(LDB,*), X(LDX,*), FERR(*), BERR(*), WORK(*)
 CHARACTER*1      UPLO, TRANS, DIAG
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine returns the backward errors and estimated bounds on the forward errors for the solution of a real triangular system of linear equations with multiple right-hand sides $AX = B$ or $A^TX = B$, using packed storage. The routine handles each right-hand side vector (stored as a column of the matrix B) independently, so we describe the function of the routine in terms of a single right-hand side b and solution x.

Given a computed solution x, the routine computes the *component-wise backward error* β. This is the size of the smallest relative perturbation in each element of A and b such that x is the exact solution of a perturbed system

$$(A+\delta A)x = b + \delta b$$

$$|\delta a_{ij}| \leq \beta |a_{ij}| \text{ and } |\delta b_i| \leq \beta |b_i|.$$

Then the routine estimates a bound for the *component-wise forward error* in the computed solution, defined by:

$$\max_i |x_i - \hat{x}_i| / \max_i |x_i|$$

where $\hat{x}$ is the true solution.

For details of the method, see the Chapter Introduction.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A is upper or lower triangular as follows:

if UPLO = 'U', then A is upper triangular;

if UPLO = 'L', then A is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: TRANS – CHARACTER*1. *Input*

On entry: indicates the form of the equations as follows:

if TRANS = 'N', then the equations are of the form $AX = B$;

if TRANS = 'T' or 'C', then the equations are of the form $A^TX = B$.

Constraint: TRANS = 'N', 'T' or 'C'.

3: DIAG – CHARACTER*1. *Input*

On entry: indicates whether A is a non-unit or unit triangular matrix as follows:

if DIAG = 'N', then A is a non-unit triangular matrix;

if DIAG = 'U', then A is a unit triangular matrix; the diagonal elements are not referenced and are assumed to be 1.

Constraint: DIAG = 'N' or 'U'.

4: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

5: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS $\geq$ 0.

6: AP(*) – ***real*** array. *Input*

Note: the dimension of the array AP must be at least max(1,N*(N+1)/2).

On entry: the n by n triangular matrix A, packed by columns. More precisely, if UPLO = 'U', the upper triangle of A must be stored with element a_{ij} in AP(i+j(j−1)/2) for $i \leq j$; if UPLO = 'L', the lower triangle of A must be stored with element a_{ij} in AP(i+(2n−j)(j−1)/2) for $i \geq j$. If DIAG = 'U', the diagonal elements of the matrix are not referenced and are assumed to be 1; the same storage scheme is used whether DIAG = 'N' or 'U'.

7: B(LDB,*) – ***real*** array. *Input*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

8: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07UHF (STPRFS/DTPRFS) is called.

Constraint: LDB $\geq$ max(1,N).

9: X(LDX,*) – ***real*** array. *Input*

Note: the second dimension of the array X must be at least max(1,NRHS).

On entry: the n by r solution matrix X, as returned by F07UEF (STPTRS/DTPTRS).

10: LDX – INTEGER. *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which F07UHF (STPRFS/DTPRFS) is called.

Constraint: LDX $\geq$ max(1,N).

11: FERR(*) – ***real*** array. *Output*

Note: the dimension of the array FERR must be at least max(1,NRHS).

On exit: FERR(j) contains an estimated error bound for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

12: BERR(*) – ***real*** array. *Output*

Note: the dimension of the array BERR must be at least max(1,NRHS).

On exit: BERR(j) contains the component-wise backward error bound β for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

13: WORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,3*N).

14: IWORK(*) – INTEGER array. *Workspace*

Note: the dimension of the array IWORK must be at least max(1,N).

15: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The bounds returned in FERR are not rigorous, because they are estimated, not computed exactly; but in practice they almost always overestimate the actual error.

8. Further Comments

A call to this routine involves, for each right-hand side, solving a number of systems of linear equations of the form $Ax = b$ or $A^Tx = b$; the number is usually 4 or 5 and never more than 11. Each solution involves approximately n^2 floating-point operations.

The complex analogue of this routine is F07UVF (CTPRFS/ZTPRFS).

9. Example

To solve the system of equations $AX = B$ and to compute forward and backward error bounds, where

$$A = \begin{pmatrix} 4.30 & 0.00 & 0.00 & 0.00 \\ -3.96 & -4.87 & 0.00 & 0.00 \\ 0.40 & 0.31 & -8.02 & 0.00 \\ -0.27 & 0.07 & -5.95 & 0.12 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} -12.90 & -21.50 \\ 16.75 & 14.93 \\ -17.55 & 6.33 \\ -11.04 & 8.09 \end{pmatrix},$$

using packed storage for A.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07UHF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, NRHMAX, LDB, LDX
      PARAMETER        (NMAX=8,NRHMAX=NMAX,LDB=NMAX,LDX=NMAX)
      CHARACTER        TRANS, DIAG
      PARAMETER        (TRANS='N',DIAG='N')
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             AP(NMAX*(NMAX+1)/2), B(LDB,NRHMAX), BERR(NRHMAX),
     +                 FERR(NRHMAX), WORK(3*NMAX), X(LDX,NMAX)
      INTEGER          IWORK(NMAX)
*     .. External Subroutines ..
      EXTERNAL         F06QFF, stprfs, stptrs, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07UHF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file, and copy B to X
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
*
         CALL F06QFF('General',N,NRHS,B,LDB,X,LDX)
*
*        Compute solution in the array X
*
         CALL stptrs(UPLO,TRANS,DIAG,N,NRHS,AP,X,LDX,INFO)
*
*        Compute backward errors and estimated bounds on the
*        forward errors
*
         CALL stprfs(UPLO,TRANS,DIAG,N,NRHS,AP,B,LDB,X,LDX,FERR,BERR,
     +               WORK,IWORK,INFO)
*
*        Print solution
*
         WRITE (NOUT,*)
         IFAIL = 0
         CALL X04CAF('General',' ',N,NRHS,X,LDX,'Solution(s)',IFAIL)
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Backward errors (machine-dependent)'
         WRITE (NOUT,99999) (BERR(J),J=1,NRHS)
         WRITE (NOUT,*)
     +     'Estimated forward error bounds (machine-dependent)'
         WRITE (NOUT,99999) (FERR(J),J=1,NRHS)
      END IF
      STOP
*
99999 FORMAT ((3X,1P,7e11.1))
      END
```

9.2. Program Data

```
F07UHF Example Program Data
  4  2                            :Values of N and NRHS
  'L'                             :Value of UPLO
  4.30
 -3.96  -4.87
  0.40   0.31  -8.02
 -0.27   0.07  -5.95   0.12      :End of matrix A
-12.90 -21.50
 16.75  14.93
-17.55   6.33
-11.04   8.09                     :End of matrix B
```

9.3. Program Results

```
 F07UHF Example Program Results

 Solution(s)
                 1           2
  1      -3.0000     -5.0000
  2      -1.0000      1.0000
  3       2.0000     -1.0000
  4       1.0000      6.0000

 Backward errors (machine-dependent)
       6.9E-17     0.0E+00
 Estimated forward error bounds (machine-dependent)
       8.3E-14     2.6E-14
```

F07UJF (STPTRI/DTPTRI) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07UJF (STPTRI/DTPTRI) computes the inverse of a real triangular matrix, using packed storage.

2. Specification

```
SUBROUTINE F07UJF (UPLO, DIAG, N, AP, INFO)
ENTRY      stptri  (UPLO, DIAG, N, AP, INFO)
INTEGER            N, INFO
real               AP(*)
CHARACTER*1        UPLO, DIAG
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine forms the inverse of a real triangular matrix A using packed storage. Note that the inverse of an upper (lower) triangular matrix is also upper (lower) triangular.

4. References

[1] DU CROZ, J.J. and HIGHAM, N.J.
Stability of Methods for Matrix Inversion.
LAPACK Working Note No. 27, University of Tennessee, Knoxville, 1990.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A is upper or lower triangular as follows:

if UPLO = 'U', then A is upper triangular;

if UPLO = 'L', then A is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: DIAG – CHARACTER*1. *Input*

On entry: indicates whether A is a non-unit or unit triangular matrix as follows:

if DIAG = 'N', then A is a non-unit triangular matrix;

if DIAG = 'U', then A is a unit triangular matrix; the diagonal elements are not referenced and are assumed to be 1.

Constraint: DIAG = 'N' or 'U'.

3: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

4: AP(*) – ***real*** array. *Input/Output*

Note: the dimension of the array AP must be at least max(1,N*(N+1)/2).

On entry: the n by n triangular matrix A, packed by columns. More precisely, if UPLO = 'U', the upper triangle of A must be stored with element a_{ij} in AP(i+j(j−1)/2) for $i \leq j$; if UPLO = 'L', the lower triangle of A must be stored with element a_{ij} in AP(i+(2n−j)(j−1)/2) for $i \geq j$. If DIAG = 'U', the diagonal elements of the matrix are not referenced and are assumed to be 1; the same storage scheme is used whether DIAG = 'N' or 'U'.

On exit: A is overwritten by A^{-1}, using the same storage format as described above.

5: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, a_{ii} is zero and the matrix A is singular.

7. Accuracy

The computed inverse X satisfies

$$|XA-I| \leq c(n)\varepsilon|X||A|,$$

where $c(n)$ is a modest linear function of n, and ε is the ***machine precision.***

Note that a similar bound for $|AX-I|$ cannot be guaranteed, although it is almost always satisfied.

The computed inverse satisfies the forward error bound

$$|X-A^{-1}| \leq c(n)\varepsilon|A^{-1}||A||X|.$$

See Du Croz and Higham [1].

8. Further Comments

The total number of floating-point operations is approximately $\frac{1}{3}n^3$.

The complex analogue of this routine is F07UWF (CTPTRI/ZTPTRI).

9. Example

To compute the inverse of the matrix A, where

$$A = \begin{pmatrix} 4.30 & 0.00 & 0.00 & 0.00 \\ -3.96 & -4.87 & 0.00 & 0.00 \\ 0.40 & 0.31 & -8.02 & 0.00 \\ -0.27 & 0.07 & -5.95 & 0.12 \end{pmatrix},$$

using packed storage.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07UJF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=8)
      CHARACTER        DIAG
      PARAMETER        (DIAG='N')
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             AP(NMAX*(NMAX+1)/2)
*     .. External Subroutines ..
      EXTERNAL         stptri, X04CCF
```

```
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07UJF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
*
*        Compute inverse of A
*
         CALL stptri(UPLO,DIAG,N,AP,INFO)
*
*        Print inverse
*
         WRITE (NOUT,*)
         IFAIL = 0
         CALL X04CCF(UPLO,DIAG,N,AP,'Inverse',IFAIL)
      END IF
      STOP
*
      END
```

9.2. Program Data

```
F07UJF Example Program Data
  4                              :Value of N
  'L'                            :Value of UPLO
  4.30
 -3.96  -4.87
  0.40   0.31  -8.02
 -0.27   0.07  -5.95   0.12     :End of matrix A
```

9.3. Program Results

```
 F07UJF Example Program Results

 Inverse
                1          2          3          4
  1      0.2326
  2     -0.1891    -0.2053
  3      0.0043    -0.0079    -0.1247
  4      0.8463    -0.2738    -6.1825     8.3333
```

F07USF (CTPTRS/ZTPTRS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07USF (CTPTRS/ZTPTRS) solves a complex triangular system of linear equations with multiple right-hand sides, $AX = B$, $A^TX = B$ or $A^HX = B$, using packed storage.

2. Specification

```
SUBROUTINE F07USF (UPLO, TRANS, DIAG, N, NRHS, AP, B, LDB, INFO)
ENTRY      ctptrs  (UPLO, TRANS, DIAG, N, NRHS, AP, B, LDB, INFO)
INTEGER          N, NRHS, LDB, INFO
complex          AP(*), B(LDB,*)
CHARACTER*1      UPLO, TRANS, DIAG
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine solves a complex triangular system of linear equations $AX = B$, $A^TX = B$ or $A^HX = B$ using packed storage.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §3.1.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

[2] HIGHAM, N.J.
The Accuracy of Solutions to Triangular Systems.
SIAM J. Numer. Anal. 5, pp. 1252-1265, 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A is upper or lower triangular as follows:

if UPLO = 'U', then A is upper triangular;

if UPLO = 'L', then A is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: TRANS – CHARACTER*1. *Input*

On entry: indicates the form of the equations as follows:

if TRANS = 'N', then the equations are of the form $AX = B$;

if TRANS = 'T', then the equations are of the form $A^TX = B$;

if TRANS = 'C', then the equations are of the form $A^HX = B$.

Constraint: TRANS = 'N', 'T' or 'C'.

3: DIAG – CHARACTER*1. *Input*

On entry: indicates whether A is a non-unit or unit triangular matrix as follows:

if DIAG = 'N', then A is a non-unit triangular matrix;

if DIAG = 'U', then A is a unit triangular matrix; the diagonal elements are not referenced and are assumed to be 1.

Constraint: DIAG = 'N' or 'U'.

4: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

5: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS $\geq$ 0.

6: AP(*) – ***complex*** array. *Input*

Note: the dimension of the array AP must be at least max(1,N*(N+1)/2).

On entry: the n by n triangular matrix A, packed by columns. More precisely, if UPLO = 'U', the upper triangle of A must be stored with element a_{ij} in AP(i+j(j−1)/2) for $i \leq j$; if UPLO = 'L', the lower triangle of A must be stored with element a_{ij} in AP(i+(2n−j)(j−1)/2) for $i \geq j$. If DIAG = 'U', the diagonal elements of the matrix are not referenced and are assumed to be 1; the same storage scheme is used whether DIAG = 'N' or 'U'.

7: B(LDB,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

On exit: the n by r solution matrix X.

8: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07USF (CTPTRS/ZTPTRS) is called.

Constraint: LDB $\geq$ max(1,N).

9: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, a_{ii} is zero and the matrix A is singular.

7. Accuracy

The solutions of triangular systems of equations are usually computed to high accuracy. See Higham [2].

For each right-hand side vector b, the computed solution x is the exact solution of a perturbed system of equations $(A+E)x = b$, where

$$|E| \leq c(n)\varepsilon|A|,$$

$c(n)$ is a modest linear function of n, and ε is the ***machine precision***.

If $\hat{x}$ is the true solution, then the computed solution x satisfies a forward error bound of the form

$$\frac{\|x-\hat{x}\|_\infty}{\|x\|_\infty} \leq c(n)\text{cond}(A,x)\varepsilon, \text{ provided } c(n)\text{cond}(A,x)\varepsilon < 1,$$

where $\text{cond}(A,x) = \||A^{-1}||A||x|\|_\infty / \|x\|_\infty$.

Note that $\mathrm{cond}(A,x) \leq \mathrm{cond}(A) = \||A^{-1}||A|\|_\infty \leq \kappa_\infty(A)$; $\mathrm{cond}(A,x)$ can be much smaller than $\mathrm{cond}(A)$ and it is also possible for $\mathrm{cond}(A^H)$, which is the same as $\mathrm{cond}(A^T)$, to be much larger (or smaller) than $\mathrm{cond}(A)$.

Forward and backward error bounds can be computed by calling F07UVF (CTPRFS/ZTPRFS), and an estimate for $\kappa_\infty(A)$ can be obtained by calling F07UUF (CTPCON/ZTPCON) with NORM = 'I'.

8. Further Comments

The total number of real floating-point operations is approximately $4n^2r$.

The real analogue of this routine is F07UEF (STPTRS/DTPTRS).

9. Example

To solve the system of equations $AX = B$, where

$$A = \begin{pmatrix} 4.78 + 4.56i & 0.00 + 0.00i & 0.00 + 0.00i & 0.00 + 0.00i \\ 2.00 - 0.30i & -4.11 + 1.25i & 0.00 + 0.00i & 0.00 + 0.00i \\ 2.89 - 1.34i & 2.36 - 4.25i & 4.15 + 0.80i & 0.00 + 0.00i \\ -1.89 + 1.15i & 0.04 - 3.69i & -0.02 + 0.46i & 0.33 - 0.26i \end{pmatrix} \quad \text{and}$$

$$B = \begin{pmatrix} -14.78 - 32.36i & -18.02 + 28.46i \\ 2.98 - 2.14i & 14.22 + 15.42i \\ -20.96 + 17.06i & 5.62 + 35.89i \\ 9.54 + 9.91i & -16.46 - 1.73i \end{pmatrix},$$

using packed storage for A.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07USF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, NRHMAX, LDB
      PARAMETER        (NMAX=8,NRHMAX=NMAX,LDB=NMAX)
      CHARACTER        TRANS, DIAG
      PARAMETER        (TRANS='N',DIAG='N')
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          AP(NMAX*(NMAX+1)/2), B(LDB,NRHMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         ctptrs, X04DBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07USF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
```

3: DIAG – CHARACTER*1. *Input*

On entry: indicates whether A is a non-unit or unit triangular matrix as follows:

if DIAG = 'N', then A is a non-unit triangular matrix;

if DIAG = 'U', then A is a unit triangular matrix; the diagonal elements are not referenced and are assumed to be 1.

Constraint: DIAG = 'N' or 'U'.

4: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

5: AP(*) – ***complex*** array. *Input*

Note: the dimension of the array AP must be at least max(1,N*(N+1)/2).

On entry: the n by n triangular matrix A, packed by columns. More precisely, if UPLO = 'U', the upper triangle of A must be stored with element a_{ij} in AP(i+j(j–1)/2) for $i \leq j$; if UPLO = 'L', the lower triangle of A must be stored with element a_{ij} in AP(i+(2n–j)(j–1)/2) for $i \geq j$. If DIAG = 'U', the diagonal elements of the matrix are not referenced and are assumed to be 1; the same storage scheme is used whether DIAG = 'N' or 'U'.

6: RCOND – ***real***. *Output*

On exit: an estimate of the reciprocal of the condition number of A. RCOND is set to zero if exact singularity is detected or the estimate underflows. If RCOND is less than ***machine precision***, then A is singular to working precision.

7: WORK(*) – ***complex*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,2*N).

8: RWORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array RWORK must be at least max(1,N).

9: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed estimate RCOND is never less than the true value ρ, and in practice is nearly always less than 10ρ, although examples can be constructed where RCOND is much larger.

8. Further Comments

A call to this routine involves solving a number of systems of linear equations of the form $Ax = b$ or $A^H x = b$; the number is usually 5 and never more than 11. Each solution involves approximately $4n^2$ real floating-point operations but takes considerably longer than a call to F07USF (CTPTRS/ZTPTRS) with one right-hand side, because extra care is taken to avoid overflow when A is approximately singular.

The real analogue of this routine is F07UGF (STPCON/DTPCON).

9. Example

To estimate the condition number in the 1-norm of the matrix A, where

$$A = \begin{pmatrix} 4.78 + 4.56i & 0.00 + 0.00i & 0.00 + 0.00i & 0.00 + 0.00i \\ 2.00 - 0.30i & -4.11 + 1.25i & 0.00 + 0.00i & 0.00 + 0.00i \\ 2.89 - 1.34i & 2.36 - 4.25i & 4.15 + 0.80i & 0.00 + 0.00i \\ -1.89 + 1.15i & 0.04 - 3.69i & -0.02 + 0.46i & 0.33 - 0.26i \end{pmatrix},$$

using packed storage. The true condition number in the 1-norm is 70.27.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07UUF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=8)
      CHARACTER        NORM, DIAG
      PARAMETER        (NORM='1',DIAG='N')
*     .. Local Scalars ..
      real             RCOND
      INTEGER          I, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          AP(NMAX*(NMAX+1)/2), WORK(2*NMAX)
      real             RWORK(NMAX)
*     .. External Functions ..
      real             X02AJF
      EXTERNAL         X02AJF
*     .. External Subroutines ..
      EXTERNAL         ctpcon
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07UUF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
*
*        Estimate condition number
*
         CALL ctpcon(NORM,UPLO,DIAG,N,AP,RCOND,WORK,RWORK,INFO)
*
         WRITE (NOUT,*)
         IF (RCOND.GE.X02AJF()) THEN
            WRITE (NOUT,99999) 'Estimate of condition number =',
     +        1.0e0/RCOND
         ELSE
            WRITE (NOUT,*) 'A is singular to working precision'
         END IF
      END IF
      STOP
*
99999 FORMAT (1X,A,1P,e10.2)
      END
```

9.2. Program Data

```
F07UUF Example Program Data
  4                                                                  :Value of N
  'L'                                                                :Value of UPLO
 ( 4.78, 4.56)
 ( 2.00,-0.30) (-4.11, 1.25)
 ( 2.89,-1.34) ( 2.36,-4.25) ( 4.15, 0.80)
 (-1.89, 1.15) ( 0.04,-3.69) (-0.02, 0.46) ( 0.33,-0.26)  :End of matrix A
```

9.3. Program Results

```
F07UUF Example Program Results

Estimate of condition number =  3.74E+01
```

F07UVF (CTPRFS/ZTPRFS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07UVF (CTPRFS/ZTPRFS) returns error bounds for the solution of a complex triangular system of linear equations with multiple right-hand sides, $AX = B$, $A^TX = B$ or $A^HX = B$, using packed storage.

2. Specification

```
SUBROUTINE F07UVF (UPLO, TRANS, DIAG, N, NRHS, AP, B, LDB, X, LDX, FERR,
1                  BERR, WORK, RWORK, INFO)
ENTRY       ctprfs  (UPLO, TRANS, DIAG, N, NRHS, AP, B, LDB, X, LDX, FERR,
1                  BERR, WORK, RWORK, INFO)
INTEGER          N, NRHS, LDB, LDX, INFO
real             FERR(*), BERR(*), RWORK(*)
complex          AP(*), B(LDB,*), X(LDX,*), WORK(*)
CHARACTER*1      UPLO, TRANS, DIAG
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine returns the backward errors and estimated bounds on the forward errors for the solution of a complex triangular system of linear equations with multiple right-hand sides $AX = B$, $A^TX = B$ or $A^HX = B$, using packed storage. The routine handles each right-hand side vector (stored as a column of the matrix B) independently, so we describe the function of the routine in terms of a single right-hand side b and solution x.

Given a computed solution x, the routine computes the *component-wise backward error* β. This is the size of the smallest relative perturbation in each element of A and b such that x is the exact solution of a perturbed system

$$(A+\delta A)x = b + \delta b$$

$$|\delta a_{ij}| \leq \beta |a_{ij}| \quad \text{and} \quad |\delta b_i| \leq \beta |b_i|.$$

Then the routine estimates a bound for the *component-wise forward error* in the computed solution, defined by:

$$\max_i |x_i - \hat{x}_i| / \max_i |x_i|$$

where $\hat{x}$ is the true solution.

For details of the method, see the Chapter Introduction.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A is upper or lower triangular as follows:

if UPLO = 'U', then A is upper triangular;

if UPLO = 'L', then A is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: TRANS – CHARACTER*1. *Input*

On entry: indicates the form of the equations as follows:

if TRANS = 'N', then the equations are of the form $AX = B$;

if TRANS = 'T', then the equations are of the form $A^TX = B$;

if TRANS = 'C', then the equations are of the form $A^HX = B$.

Constraint: TRANS = 'N', 'T' or 'C'.

3: DIAG – CHARACTER*1. *Input*

On entry: indicates whether A is a non-unit or unit triangular matrix as follows:

if DIAG = 'N', then A is a non-unit triangular matrix;

if DIAG = 'U', then A is a unit triangular matrix; the diagonal elements are not referenced and are assumed to be 1.

Constraint: DIAG = 'N' or 'U'.

4: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

5: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS $\geq$ 0.

6: AP(*) – ***complex*** array. *Input*

Note: the dimension of the array AP must be at least max(1,N*(N+1)/2).

On entry: the n by n triangular matrix A, packed by columns. More precisely, if UPLO = 'U', the upper triangle of A must be stored with element a_{ij} in AP(i+j(j−1)/2) for $i \leq j$; if UPLO = 'L', the lower triangle of A must be stored with element a_{ij} in AP(i+(2n−j)(j−1)/2) for $i \geq j$. If DIAG = 'U', the diagonal elements of the matrix are not referenced and are assumed to be 1; the same storage scheme is used whether DIAG = 'N' or 'U'.

7: B(LDB,*) – ***complex*** array. *Input*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

8: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07UVF (CTPRFS/ZTPRFS) is called.

Constraint: LDB $\geq$ max(1,N).

9: X(LDX,*) – ***complex*** array. *Input*

Note: the second dimension of the array X must be at least max(1,NRHS).

On entry: the n by r solution matrix X, as returned by F07USF (CTPTRS/ZTPTRS).

10: LDX – INTEGER. *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which F07UVF (CTPRFS/ZTPRFS) is called.

Constraint: LDX $\geq$ max(1,N).

11: FERR(*) – ***real*** array. *Output*

Note: the dimension of the array FERR must be at least max(1,NRHS).

On exit: FERR(j) contains an estimated error bound for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

12: BERR(*) – ***real*** array. *Output*

Note: the dimension of the array BERR must be at least max(1,NRHS).

On exit: BERR(j) contains the component-wise backward error bound β for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

13: WORK(*) – ***complex*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,2*N).

14: RWORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array RWORK must be at least max(1,N).

15: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The bounds returned in FERR are not rigorous, because they are estimated, not computed exactly; but in practice they almost always overestimate the actual error.

8. Further Comments

A call to this routine involves, for each right-hand side, solving a number of systems of linear equations of the form $Ax = b$ or $A^H x = b$; the number is usually 5 and never more than 11. Each solution involves approximately $4n^2$ real floating-point operations.

The real analogue of this routine is F07UHF (STPRFS/DTPRFS).

9. Example

To solve the system of equations $AX = B$ and to compute forward and backward error bounds, where

$$A = \begin{pmatrix} 4.78 + 4.56i & 0.00 + 0.00i & 0.00 + 0.00i & 0.00 + 0.00i \\ 2.00 - 0.30i & -4.11 + 1.25i & 0.00 + 0.00i & 0.00 + 0.00i \\ 2.89 - 1.34i & 2.36 - 4.25i & 4.15 + 0.80i & 0.00 + 0.00i \\ -1.89 + 1.15i & 0.04 - 3.69i & -0.02 + 0.46i & 0.33 - 0.26i \end{pmatrix} \quad \text{and}$$

$$B = \begin{pmatrix} -14.78 - 32.36i & -18.02 + 28.46i \\ 2.98 - 2.14i & 14.22 + 15.42i \\ -20.96 + 17.06i & 5.62 + 35.89i \\ 9.54 + 9.91i & -16.46 - 1.73i \end{pmatrix},$$

using packed storage for A.

AP$(i+(2n-j)(j-1)/2)$ for $i \geq j$. If DIAG = 'U', the diagonal elements of the matrix are not referenced and are assumed to be 1; the same storage scheme is used whether DIAG = 'N' or 'U'.

On exit: A is overwritten by A^{-1}, using the same storage format as described above.

5: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, a_{ii} is zero and the matrix A is singular.

7. Accuracy

The computed inverse X satisfies

$$|XA-I| \leq c(n)\varepsilon|X||A|,$$

where $c(n)$ is a modest linear function of n, and ε is the ***machine precision***.

Note that a similar bound for $|AX-I|$ cannot be guaranteed, although it is almost always satisfied.

The computed inverse satisfies the forward error bound

$$|X-A^{-1}| \leq c(n)\varepsilon|A^{-1}||A||X|.$$

See Du Croz and Higham [1].

8. Further Comments

The total number of real floating-point operations is approximately $\frac{4}{3}n^3$.

The real analogue of this routine is F07UJF (STPTRI/DTPTRI).

9. Example

To compute the inverse of the matrix A, where

$$A = \begin{pmatrix} 4.78 + 4.56i & 0.00 + 0.00i & 0.00 + 0.00i & 0.00 + 0.00i \\ 2.00 - 0.30i & -4.11 + 1.25i & 0.00 + 0.00i & 0.00 + 0.00i \\ 2.89 - 1.34i & 2.36 - 4.25i & 4.15 + 0.80i & 0.00 + 0.00i \\ -1.89 + 1.15i & 0.04 - 3.69i & -0.02 + 0.46i & 0.33 - 0.26i \end{pmatrix},$$

using packed storage.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07UWF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=8)
      CHARACTER        DIAG
      PARAMETER        (DIAG='N')
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
      CHARACTER        UPLO
```

```
*     .. Local Arrays ..
      complex           AP(NMAX*(NMAX+1)/2)
      CHARACTER         CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL          ctptri, X04DDF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07UWF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
*
*        Compute inverse of A
*
         CALL ctptri(UPLO,DIAG,N,AP,INFO)
*
*        Print inverse
*
         WRITE (NOUT,*)
         IFAIL = 0
         CALL X04DDF(UPLO,DIAG,N,AP,'Bracketed','F7.4','Inverse',
     +               'Integer',RLABS,'Integer',CLABS,80,0,IFAIL)
      END IF
      STOP
*
      END
```

9.2. Program Data

```
F07UWF Example Program Data
  4                                                              :Value of N
  'L'                                                            :Value of UPLO
 ( 4.78, 4.56)
 ( 2.00,-0.30) (-4.11, 1.25)
 ( 2.89,-1.34) ( 2.36,-4.25) ( 4.15, 0.80)
 (-1.89, 1.15) ( 0.04,-3.69) (-0.02, 0.46) ( 0.33,-0.26)  :End of matrix A
```

9.3. Program Results

```
 F07UWF Example Program Results

 Inverse
                      1                  2                  3                  4
  1  ( 0.1095,-0.1045)
  2  ( 0.0582,-0.0411) (-0.2227,-0.0677)
  3  ( 0.0032, 0.1905) ( 0.1538,-0.2192) ( 0.2323,-0.0448)
  4  ( 0.7602, 0.2814) ( 1.6184,-1.4346) ( 0.1289,-0.2250) ( 1.8697, 1.4731)
```

F07VEF (STBTRS/DTBTRS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07VEF (STBTRS/DTBTRS) solves a real triangular band system of linear equations with multiple right-hand sides, $AX = B$ or $A^TX = B$.

2. Specification

```
SUBROUTINE F07VEF (UPLO, TRANS, DIAG, N, KD, NRHS, AB, LDAB, B, LDB,
1                  INFO)
ENTRY       stbtrs  (UPLO, TRANS, DIAG, N, KD, NRHS, AB, LDAB, B, LDB,
1                  INFO)
INTEGER          N, KD, NRHS, LDAB, LDB, INFO
real             AB(LDAB,*), B(LDB,*)
CHARACTER*1      UPLO, TRANS, DIAG
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine solves a real triangular band system of linear equations $AX = B$ or $A^TX = B$.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §3.1.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

[2] HIGHAM, N.J.
The Accuracy of Solutions to Triangular Systems.
SIAM J. Numer. Anal., 5, pp. 1252-1265, 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A is upper or lower triangular as follows:

if UPLO = 'U', then A is upper triangular;

if UPLO = 'L', then A is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: TRANS – CHARACTER*1. *Input*

On entry: indicates the form of the equations as follows:

if TRANS = 'N', then the equations are of the form $AX = B$;

if TRANS = 'T' or 'C', then the equations are of the form $A^TX = B$.

Constraint: TRANS = 'N', 'T' or 'C'.

3: DIAG – CHARACTER*1. *Input*

On entry: indicates whether A is a non-unit or unit triangular matrix as follows:

if DIAG = 'N', then A is a non-unit triangular matrix;

if DIAG = 'U', then A is a unit triangular matrix; the diagonal elements are not referenced and are assumed to be 1.

Constraint: DIAG = 'N' or 'U'.

4: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

5: KD – INTEGER. *Input*

On entry: k, the number of super-diagonals of the matrix A if UPLO = 'U' or the number of sub-diagonals if UPLO = 'L'.

Constraint: KD $\geq$ 0.

6: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS $\geq$ 0.

7: AB(LDAB,*) – ***real*** array. *Input*

Note: the second dimension of the array AB must be at least max(1,N).

On entry: the n by n triangular band matrix A, stored in rows 1 to $(k+1)$. More precisely, if UPLO = 'U', the elements of the upper triangle of A within the band must be stored with element a_{ij} in AB$(k+1+i-j,j)$ for max$(1,j-k) \leq i \leq j$; if UPLO = 'L', the elements of the lower triangle of A within the band must be stored with element a_{ij} in AB$(1+i-j,j)$ for $j \leq i \leq$ min$(n,j+k)$. If DIAG = 'U', the diagonal elements are not referenced and are assumed to be 1.

8: LDAB – INTEGER. *Input*

On entry: the first dimension of the array AB as declared in the (sub)program from which F07VEF (STBTRS/DTBTRS) is called.

Constraint: LDAB $\geq$ KD + 1.

9: B(LDB,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

On exit: the n by r solution matrix X.

10: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07VEF (STBTRS/DTBTRS) is called.

Constraint: LDB $\geq$ max(1,N).

11: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, a_{ii} is zero and the matrix A is singular.

7. Accuracy

The solutions of triangular systems of equations are usually computed to high accuracy. See Higham [2].

For each right-hand side vector b, the computed solution x is the exact solution of a perturbed system of equations $(A+E)x = b$, where

$$|E| \leq c(k)\varepsilon|A|,$$

$c(k)$ is a modest linear function of k, and ε is the ***machine precision***.

If $\hat{x}$ is the true solution, then the computed solution x satisfies a forward error bound of the form

$$\frac{\|x-\hat{x}\|_\infty}{\|x\|_\infty} \leq c(k)\text{cond}(A,x)\varepsilon, \text{ provided } c(k)\text{cond}(A,x)\varepsilon < 1,$$

where $\text{cond}(A,x) = \||A^{-1}||A||x|\|_\infty / \|x\|_\infty$.

Note that $\text{cond}(A,x) \leq \text{cond}(A) = \||A^{-1}||A|\|_\infty \leq \kappa_\infty(A)$; $\text{cond}(A,x)$ can be much smaller than $\text{cond}(A)$ and it is also possible for $\text{cond}(A^T)$ to be much larger (or smaller) than $\text{cond}(A)$.

Forward and backward error bounds can be computed by calling F07VHF (STBRFS/DTBRFS), and an estimate for $\kappa_\infty(A)$ can be obtained by calling F07VGF (STBCON/DTBCON) with NORM = 'I'.

8. Further Comments

The total number of floating-point operations is approximately $2nkr$ if $k \ll n$.

The complex analogue of this routine is F07VSF (CTBTRS/ZTBTRS).

9. Example

To solve the system of equations $AX = B$, where

$$A = \begin{pmatrix} -4.16 & 0.00 & 0.00 & 0.00 \\ -2.25 & 4.78 & 0.00 & 0.00 \\ 0.00 & 5.86 & 6.32 & 0.00 \\ 0.00 & 0.00 & -4.82 & 0.16 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} -16.64 & -4.16 \\ -13.78 & -16.59 \\ 13.10 & -4.94 \\ -14.14 & -9.96 \end{pmatrix}.$$

Here A is treated as a lower triangular band matrix with 1 sub-diagonal.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07VEF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, KDMAX, LDAB, NRHMAX, LDB
      PARAMETER        (NMAX=8,KDMAX=NMAX,LDAB=KDMAX+1,NRHMAX=NMAX,
     +                 LDB=NMAX)
      CHARACTER        TRANS, DIAG
      PARAMETER        (TRANS='N',DIAG='N')
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, KD, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             AB(LDAB,NMAX), B(LDB,NRHMAX)
*     .. External Subroutines ..
      EXTERNAL         stbtrs, X04CAF
*     .. Intrinsic Functions ..
      INTRINSIC        MAX, MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07VEF Example Program Results'
```

```
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, KD, NRHS
      IF (N.LE.NMAX .AND. KD.LE.KDMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            DO 20 I = 1, N
               READ (NIN,*) (AB(KD+1+I-J,J),J=I,MIN(N,I+KD))
   20       CONTINUE
         ELSE IF (UPLO.EQ.'L') THEN
            DO 40 I = 1, N
               READ (NIN,*) (AB(1+I-J,J),J=MAX(1,I-KD),I)
   40       CONTINUE
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
*
*        Compute solution
*
         CALL stbtrs(UPLO,TRANS,DIAG,N,KD,NRHS,AB,LDAB,B,LDB,INFO)
*
*        Print solution
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
            IFAIL = 0
            CALL X04CAF('General',' ',N,NRHS,B,LDB,'Solution(s)',IFAIL)
         ELSE
            WRITE (NOUT,*) 'A is singular'
         END IF
      END IF
      STOP
*
      END
```

9.2. Program Data

```
F07VEF Example Program Data
  4  1  2                        :Values of N, KD and NRHS
  'L'                             :Value of UPLO
 -4.16
 -2.25   4.78
         5.86   6.32
                -4.82   0.16     :End of matrix A
-16.64  -4.16
-13.78 -16.59
 13.10  -4.94
-14.14  -9.96                    :End of matrix B
```

9.3. Program Results

```
 F07VEF Example Program Results

 Solution(s)
                  1           2
  1       4.0000      1.0000
  2      -1.0000     -3.0000
  3       3.0000      2.0000
  4       2.0000     -2.0000
```

F07VGF (STBCON/DTBCON) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07VGF (STBCON/DTBCON) estimates the condition number of a real triangular band matrix.

2. Specification

```
 SUBROUTINE F07VGF (NORM, UPLO, DIAG, N, KD, AB, LDAB, RCOND, WORK,
1                   IWORK, INFO)
 ENTRY       stbcon (NORM, UPLO, DIAG, N, KD, AB, LDAB, RCOND, WORK,
1                   IWORK, INFO)
 INTEGER          N, KD, LDAB, IWORK(*), INFO
 real             AB(LDAB,*), RCOND, WORK(*)
 CHARACTER*1      NORM, UPLO, DIAG
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine estimates the condition number of a real triangular band matrix A, in either the 1-norm or the infinity-norm:

$$\kappa_1(A) = \|A\|_1\|A^{-1}\|_1 \quad \text{or} \quad \kappa_\infty(A) = \|A\|_\infty\|A^{-1}\|_\infty.$$

Note that $\kappa_\infty(A) = \kappa_1(A^T)$.

Because the condition number is infinite if A is singular, the routine actually returns an estimate of the ***reciprocal*** of the condition number.

The routine computes $\|A\|_1$ or $\|A\|_\infty$ exactly, and uses Higham's implementation of Hager's method [1] to estimate $\|A^{-1}\|_1$ or $\|A^{-1}\|_\infty$.

4. References

[1] HIGHAM, N.J.
FORTRAN Codes for Estimating the One-norm of a Real or Complex Matrix, with Applications to Condition Estimation.
ACM Trans. Math. Softw., 14, pp. 381-396, 1988.

5. Parameters

1: NORM – CHARACTER*1. *Input*

On entry: indicates whether $\kappa_1(A)$ or $\kappa_\infty(A)$ is estimated as follows:

if NORM = '1' or 'O', then $\kappa_1(A)$ is estimated;

if NORM = 'I', then $\kappa_\infty(A)$ is estimated.

Constraint: NORM = '1', 'O' or 'I'.

2: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A is upper or lower triangular as follows:

if UPLO = 'U', then A is upper triangular;

if UPLO = 'L', then A is lower triangular.

Constraint: UPLO = 'U' or 'L'.

3: DIAG – CHARACTER*1. *Input*

On entry: indicates whether A is a non-unit or unit triangular matrix as follows:

if DIAG = 'N', then A is a non-unit triangular matrix;

if DIAG = 'U', then A is a unit triangular matrix; the diagonal elements are not referenced and are assumed to be 1.

Constraint: DIAG = 'N' or 'U'.

4: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

5: KD – INTEGER. *Input*

On entry: k, the number of super-diagonals of the matrix A if UPLO = 'U' or the number of sub-diagonals if UPLO = 'L'.

Constraint: KD $\geq$ 0.

6: AB(LDAB,*) – ***real*** array. *Input*

Note: the second dimension of the array AB must be at least max(1,N).

On entry: the n by n triangular band matrix A, stored in rows 1 to $(k+1)$. More precisely, if UPLO = 'U', the elements of the upper triangle of A within the band must be stored with element a_{ij} in AB$(k+1+i-j,j)$ for max$(1,j-k) \leq i \leq j$; if UPLO = 'L', the elements of the lower triangle of A within the band must be stored with element a_{ij} in AB$(1+i-j,j)$ for $j \leq i \leq \min(n,j+k)$. If DIAG = 'U', the diagonal elements are not referenced and are assumed to be 1.

7: LDAB – INTEGER. *Input*

On entry: the first dimension of the array AB as declared in the (sub)program from which F07VGF (STBCON/DTBCON) is called.

Constraint: LDAB $\geq$ KD + 1.

8: RCOND – ***real***. *Output*

On exit: an estimate of the reciprocal of the condition number of A. RCOND is set to zero if exact singularity is detected or if the estimate underflows. If RCOND is less than ***machine precision***, then A is singular to working precision.

9: WORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,3*N).

10: IWORK(*) – INTEGER array. *Workspace*

Note: the dimension of the array IWORK must be at least max(1,N).

11: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed estimate RCOND is never less than the true value ρ, and in practice is nearly always less than 10ρ, although examples can be constructed where RCOND is much larger.

8. Further Comments

A call to this routine involves solving a number of systems of linear equations of the form $Ax = b$ or $A^T x = b$; the number is usually 4 or 5 and never more than 11. Each solution involves approximately $2nk$ floating-point operations (assuming $n \gg k$) but takes considerably longer than a call to F07VEF (STBTRS/DTBTRS) with one right-hand side, because extra care is taken to avoid overflow when A is approximately singular.

The complex analogue of this routine is F07VUF (CTBCON/ZTBCON).

9. Example

To estimate the condition number in the 1-norm of the matrix A, where

$$A = \begin{pmatrix} -4.16 & 0.00 & 0.00 & 0.00 \\ -2.25 & 4.78 & 0.00 & 0.00 \\ 0.00 & 5.86 & 6.32 & 0.00 \\ 0.00 & 0.00 & -4.82 & 0.16 \end{pmatrix}.$$

Here A is treated as a lower triangular band matrix with 1 sub-diagonal. The true condition number in the 1-norm is 69.62.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07VGF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, KDMAX, LDAB
      PARAMETER        (NMAX=8,KDMAX=NMAX,LDAB=KDMAX+1)
      CHARACTER        NORM, DIAG
      PARAMETER        (NORM='1',DIAG='N')
*     .. Local Scalars ..
      real             RCOND
      INTEGER          I, INFO, J, KD, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             AB(LDAB,NMAX), WORK(3*NMAX)
      INTEGER          IWORK(NMAX)
*     .. External Functions ..
      real             X02AJF
      EXTERNAL         X02AJF
*     .. External Subroutines ..
      EXTERNAL         stbcon
*     .. Intrinsic Functions ..
      INTRINSIC        MAX, MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07VGF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, KD
      IF (N.LE.NMAX .AND. KD.LE.KDMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            DO 20 I = 1, N
               READ (NIN,*) (AB(KD+1+I-J,J),J=I,MIN(N,I+KD))
   20       CONTINUE
```

```
         ELSE IF (UPLO.EQ.'L') THEN
            DO 40 I = 1, N
               READ (NIN,*) (AB(1+I-J,J),J=MAX(1,I-KD),I)
   40       CONTINUE
         END IF
*
*        Estimate condition number
*
         CALL stbcon(NORM,UPLO,DIAG,N,KD,AB,LDAB,RCOND,WORK,IWORK,INFO)
*
         WRITE (NOUT,*)
         IF (RCOND.GE.X02AJF()) THEN
            WRITE (NOUT,99999) 'Estimate of condition number =',
     +        1.0e0/RCOND
         ELSE
            WRITE (NOUT,*) 'A is singular to working precision'
         END IF
       END IF
       STOP
*
99999 FORMAT (1X,A,1P,e10.2)
      END
```

9.2. Program Data

```
F07VGF Example Program Data
  4  1                          :Values of N and KD
  'L'                           :Value of UPLO
 -4.16
 -2.25   4.78
         5.86   6.32
                -4.82   0.16    :End of matrix A
```

9.3. Program Results

```
F07VGF Example Program Results

Estimate of condition number =  6.96E+01
```

F07VHF (STBRFS/DTBRFS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07VHF (STBRFS/DTBRFS) returns error bounds for the solution of a real triangular band system of linear equations with multiple right-hand sides, $AX = B$ or $A^TX = B$.

2. Specification

```
      SUBROUTINE F07VHF (UPLO, TRANS, DIAG, N, KD, NRHS, AB, LDAB, B, LDB, X,
     1                   LDX, FERR, BERR, WORK, IWORK, INFO)
      ENTRY       stbrfs  (UPLO, TRANS, DIAG, N, KD, NRHS, AB, LDAB, B, LDB, X,
     1                   LDX, FERR, BERR, WORK, IWORK, INFO)
      INTEGER           N, KD, NRHS, LDAB, LDB, LDX, IWORK(*), INFO
      real              AB(LDAB,*), B(LDB,*), X(LDX,*), FERR(*), BERR(*),
     1                  WORK(*)
      CHARACTER*1       UPLO, TRANS, DIAG
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine returns the backward errors and estimated bounds on the forward errors for the solution of a real triangular band system of linear equations with multiple right-hand sides $AX = B$ or $A^TX = B$. The routine handles each right-hand side vector (stored as a column of the matrix B) independently, so we describe the function of the routine in terms of a single right-hand side b and solution x.

Given a computed solution x, the routine computes the *component-wise backward error* β. This is the size of the smallest relative perturbation in each element of A and b such that x is the exact solution of a perturbed system

$$(A+\delta A)x = b + \delta b$$

$$|\delta a_{ij}| \leq \beta |a_{ij}| \text{ and } |\delta b_i| \leq \beta |b_i|.$$

Then the routine estimates a bound for the *component-wise forward error* in the computed solution, defined by:

$$\max_i |x_i - \hat{x}_i| / \max_i |x_i|$$

where $\hat{x}$ is the true solution.

For details of the method, see the Chapter Introduction.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A is upper or lower triangular as follows:

if UPLO = 'U', then A is upper triangular;

if UPLO = 'L', then A is lower triangular.

Constraint: UPLO = 'U' or 'L'.

F07VSF (CTBTRS/ZTBTRS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07VSF (CTBTRS/ZTBTRS) solves a complex triangular band system of linear equations with multiple right-hand sides, $AX = B$, $A^TX = B$ or $A^HX = B$.

2. Specification

```
 SUBROUTINE F07VSF (UPLO, TRANS, DIAG, N, KD, NRHS, AB, LDAB, B, LDB,
1                   INFO)
 ENTRY       ctbtrs  (UPLO, TRANS, DIAG, N, KD, NRHS, AB, LDAB, B, LDB,
1                   INFO)
 INTEGER          N, KD, NRHS, LDAB, LDB, INFO
 complex          AB(LDAB,*), B(LDB,*)
 CHARACTER*1      UPLO, TRANS, DIAG
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine solves a complex triangular band system of linear equations $AX = B$, $A^TX = B$ or $A^HX = B$.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §3.1.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

[2] HIGHAM, N.J.
The Accuracy of Solutions to Triangular Systems.
SIAM J. Numer. Anal. 5, pp. 1252-1265, 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A is upper or lower triangular as follows:

if UPLO = 'U', then A is upper triangular;

if UPLO = 'L', then A is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: TRANS – CHARACTER*1. *Input*

On entry: indicates the form of the equations as follows:

if TRANS = 'N', then the equations are of the form $AX = B$;

if TRANS = 'T', then the equations are of the form $A^TX = B$;

if TRANS = 'C', then the equations are of the form $A^HX = B$.

Constraint: TRANS = 'N', 'T' or 'C'.

3: DIAG – CHARACTER*1. *Input*

On entry: indicates whether A is a non-unit or unit triangular matrix as follows:

if DIAG = 'N', then A is a non-unit triangular matrix;

if DIAG = 'U', then A is a unit triangular matrix; the diagonal elements are not referenced and are assumed to be 1.

Constraint: DIAG = 'N' or 'U'.

4: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 0.

5: KD – INTEGER. *Input*

On entry: k, the number of super-diagonals of the matrix A if UPLO = 'U' or the number of sub-diagonals if UPLO = 'L'.

Constraint: KD ≥ 0.

6: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS ≥ 0.

7: AB(LDAB,*) – ***complex*** array. *Input*

Note: the second dimension of the array AB must be at least max(1,N).

On entry: the n by n triangular band matrix A, stored in rows 1 to $(k+1)$. More precisely, if UPLO = 'U', the elements of the upper triangle of A within the band must be stored with element a_{ij} in AB$(k+1+i-j,j)$ for $\max(1,j-k) \leq i \leq j$; if UPLO = 'L', the elements of the lower triangle of A within the band must be stored with element a_{ij} in AB$(1+i-j,j)$ for $j \leq i \leq \min(n,j+k)$. If DIAG = 'U', the diagonal elements of A are not referenced and are assumed to be 1.

8: LDAB – INTEGER. *Input*

On entry: the first dimension of the array AB as declared in the (sub)program from which F07VSF (CTBTRS/ZTBTRS) is called.

Constraint: LDAB ≥ KD + 1.

9: B(LDB,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

On exit: the n by r solution matrix X.

10: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07VSF (CTBTRS/ZTBTRS) is called.

Constraint: LDB ≥ max(1,N).

11: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, a_{ii} is zero and the matrix A is singular.

7. Accuracy

The solutions of triangular systems of equations are usually computed to high accuracy. See Higham [2].

For each right-hand side vector b, the computed solution x is the exact solution of a perturbed system of equations $(A+E)x = b$, where

$$|E| \leq c(k)\varepsilon|A|,$$

$c(k)$ is a modest linear function of k, and ε is the ***machine precision***.

If $\hat{x}$ is the true solution, then the computed solution x satisfies a forward error bound of the form

$$\frac{\|x-\hat{x}\|_\infty}{\|x\|_\infty} \leq c(k)\text{cond}(A,x)\varepsilon, \text{ provided } c(k)\text{cond}(A,x)\varepsilon < 1,$$

where $\text{cond}(A,x) = \||A^{-1}||A||x|\|_\infty / \|x\|_\infty$.

Note that $\text{cond}(A,x) \leq \text{cond}(A) = \||A^{-1}||A|\|_\infty \leq \kappa_\infty(A)$; $\text{cond}(A,x)$ can be much smaller than $\text{cond}(A)$ and it is also possible for $\text{cond}(A^H)$, which is the same as $\text{cond}(A^T)$, to be much larger (or smaller) than $\text{cond}(A)$.

Forward and backward error bounds can be computed by calling F07VVF (CTBRFS/ZTBRFS), and an estimate for $\kappa_\infty(A)$ can be obtained by calling F07VUF (CTBCON/ZTBCON) with NORM = 'I'.

8. Further Comments

The total number of real floating-point operations is approximately $8nkr$ if $k \ll n$.

The real analogue of this routine is F07VEF (STBTRS/DTBTRS).

9. Example

To solve the system of equations $AX = B$, where

$$A = \begin{pmatrix} -1.94 + 4.43i & 0.00 + 0.00i & 0.00 + 0.00i & 0.00 + 0.00i \\ -3.39 + 3.44i & 4.12 - 4.27i & 0.00 + 0.00i & 0.00 + 0.00i \\ 1.62 + 3.68i & -1.84 + 5.53i & 0.43 - 2.66i & 0.00 + 0.00i \\ 0.00 + 0.00i & -2.77 - 1.93i & 1.74 - 0.04i & 0.44 + 0.10i \end{pmatrix} \text{ and}$$

$$B = \begin{pmatrix} -8.86 - 3.88i & -24.09 - 5.27i \\ -15.57 - 23.41i & -57.97 + 8.14i \\ -7.63 + 22.78i & 19.09 - 29.51i \\ -14.74 - 2.40i & 19.17 + 21.33i \end{pmatrix}.$$

Here A is treated as a lower triangular band matrix with 2 sub-diagonals.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07VSF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, KDMAX, LDAB, NRHMAX, LDB
      PARAMETER        (NMAX=8,KDMAX=NMAX,LDAB=KDMAX+1,NRHMAX=NMAX,
     +                 LDB=NMAX)
      CHARACTER        TRANS, DIAG
      PARAMETER        (TRANS='N',DIAG='N')
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, KD, N, NRHS
      CHARACTER        UPLO
```

```
*     .. Local Arrays ..
      complex           AB(LDAB,NMAX), B(LDB,NRHMAX)
      CHARACTER         CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL          ctbtrs, X04DBF
*     .. Intrinsic Functions ..
      INTRINSIC         MAX, MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07VSF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, KD, NRHS
      IF (N.LE.NMAX .AND. KD.LE.KDMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            DO 20 I = 1, N
               READ (NIN,*) (AB(KD+1+I-J,J),J=I,MIN(N,I+KD))
   20       CONTINUE
         ELSE IF (UPLO.EQ.'L') THEN
            DO 40 I = 1, N
               READ (NIN,*) (AB(1+I-J,J),J=MAX(1,I-KD),I)
   40       CONTINUE
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
*
*        Compute solution
*
         CALL ctbtrs(UPLO,TRANS,DIAG,N,KD,NRHS,AB,LDAB,B,LDB,INFO)
*
*        Print solution
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
            IFAIL = 0
            CALL X04DBF('General',' ',N,NRHS,B,LDB,'Bracketed','F7.4',
     +                  'Solution(s)','Integer',RLABS,'Integer',CLABS,
     +                  80,0,IFAIL)
         ELSE
            WRITE (NOUT,*) 'A is singular'
         END IF
      END IF
      STOP
*
      END
```

9.2. Program Data

```
F07VSF Example Program Data
  4  2  2                                              :Values of N, KD and NRHS
  'L'                                                    :Value of UPLO
 (-1.94, 4.43)
 (-3.39, 3.44) ( 4.12,-4.27)
 ( 1.62, 3.68) (-1.84, 5.53) ( 0.43,-2.66)
               (-2.77,-1.93) ( 1.74,-0.04) ( 0.44, 0.10)  :End of matrix A
 ( -8.86, -3.88) (-24.09, -5.27)
 (-15.57,-23.41) (-57.97,  8.14)
 ( -7.63, 22.78) ( 19.09,-29.51)
 (-14.74, -2.40) ( 19.17, 21.33)                         :End of matrix B
```

9.3. Program Results

```
F07VSF Example Program Results

Solution(s)
                         1                   2
 1   ( 0.0000, 2.0000) ( 1.0000, 5.0000)
 2   ( 1.0000,-3.0000) (-7.0000,-2.0000)
 3   (-4.0000,-5.0000) ( 3.0000, 4.0000)
 4   ( 2.0000,-1.0000) (-6.0000,-9.0000)
```

F07VUF (CTBCON/ZTBCON) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07VUF (CTBCON/ZTBCON) estimates the condition number of a complex triangular band matrix.

2. Specification

```
 SUBROUTINE F07VUF (NORM, UPLO, DIAG, N, KD, AB, LDAB, RCOND, WORK,
1                   RWORK, INFO)
 ENTRY       ctbcon  (NORM, UPLO, DIAG, N, KD, AB, LDAB, RCOND, WORK,
1                   RWORK, INFO)
 INTEGER          N, KD, LDAB, INFO
 real             RCOND, RWORK(*)
 complex          AB(LDAB,*), WORK(*)
 CHARACTER*1      NORM, UPLO, DIAG
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine estimates the condition number of a complex triangular band matrix A, in either the 1-norm or the infinity-norm:

$$\kappa_1(A) = \|A\|_1\|A^{-1}\|_1 \quad \text{or} \quad \kappa_\infty(A) = \|A\|_\infty\|A^{-1}\|_\infty.$$

Note that $\kappa_\infty(A) = \kappa_1(A^T)$.

Because the condition number is infinite if A is singular, the routine actually returns an estimate of the **reciprocal** of the condition number.

The routine computes $\|A\|_1$ or $\|A\|_\infty$ exactly, and uses Higham's implementation of Hager's method [1] to estimate $\|A^{-1}\|_1$ or $\|A^{-1}\|_\infty$.

4. References

[1] HIGHAM, N.J.
FORTRAN Codes for Estimating the One-norm of a Real or Complex Matrix, with Applications to Condition Estimation.
ACM Trans. Math. Softw., 14, pp. 381-396, 1988.

5. Parameters

1: NORM – CHARACTER*1. *Input*

On entry: indicates whether $\kappa_1(A)$ or $\kappa_\infty(A)$ is estimated as follows:

if NORM = '1' or 'O', then $\kappa_1(A)$ is estimated;

if NORM = 'I', then $\kappa_\infty(A)$ is estimated.

Constraint: NORM = '1', 'O' or 'I'.

2: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A is upper or lower triangular as follows:

if UPLO = 'U', then A is upper triangular;

if UPLO = 'L', then A is lower triangular.

Constraint: UPLO = 'U' or 'L'.

3: DIAG – CHARACTER*1. *Input*

On entry: indicates whether A is a non-unit or unit triangular matrix as follows:

if DIAG = 'N', then A is a non-unit triangular matrix;

if DIAG = 'U', then A is a unit triangular matrix; the diagonal elements are not referenced and are assumed to be 1.

Constraint: DIAG = 'N' or 'U'.

4: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

5: KD – INTEGER. *Input*

On entry: k, the number of super-diagonals of the matrix A if UPLO = 'U' or the number of sub-diagonals if UPLO = 'L'.

Constraint: KD $\geq$ 0.

6: AB(LDAB,*) – ***complex*** array. *Input*

Note: the second dimension of the array AB must be at least max(1,N).

On entry: the n by n triangular band matrix A, stored in rows 1 to $(k+1)$. More precisely, if UPLO = 'U', the elements of the upper triangle of A within the band must be stored with element a_{ij} in AB$(k+1+i-j,j)$ for $\max(1,j-k) \leq i \leq j$; if UPLO = 'L', the elements of the lower triangle of A within the band must be stored with element a_{ij} in AB$(1+i-j,j)$ for $j \leq i \leq \min(n,j+k)$. If DIAG = 'U', the diagonal elements of A are not referenced and are assumed to be 1.

7: LDAB – INTEGER. *Input*

On entry: the first dimension of the array AB as declared in the (sub)program from which F07VUF (CTBCON/ZTBCON) is called.

Constraint: LDAB $\geq$ KD + 1.

8: RCOND – ***real***. *Output*

On exit: an estimate of the reciprocal of the condition number of A. RCOND is set to zero if exact singularity is detected or the estimate underflows. If RCOND is less than ***machine precision***, then A is singular to working precision.

9: WORK(*) – ***complex*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,2*N).

10: RWORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array RWORK must be at least max(1,N).

11: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed estimate RCOND is never less than the true value ρ, and in practice is nearly always less than 10ρ, although examples can be constructed where RCOND is much larger.

8. Further Comments

A call to this routine involves solving a number of systems of linear equations of the form $Ax = b$ or $A^H x = b$; the number is usually 5 and never more than 11. Each solution involves approximately $8nk$ real floating-point operations (assuming $n \gg k$) but takes considerably longer than a call to F07VSF (CTBTRS/ZTBTRS) with one right-hand side, because extra care is taken to avoid overflow when A is approximately singular.

The real analogue of this routine is F07VGF (STBCON/DTBCON).

9. Example

To estimate the condition number in the 1-norm of the matrix A, where

$$A = \begin{pmatrix} -1.94 + 4.43i & 0.00 + 0.00i & 0.00 + 0.00i & 0.00 + 0.00i \\ -3.39 + 3.44i & 4.12 - 4.27i & 0.00 + 0.00i & 0.00 + 0.00i \\ 1.62 + 3.68i & -1.84 + 5.53i & 0.43 - 2.66i & 0.00 + 0.00i \\ 0.00 + 0.00i & -2.77 - 1.93i & 1.74 - 0.04i & 0.44 + 0.10i \end{pmatrix}.$$

Here A is treated as a lower triangular band matrix with 2 sub-diagonals. The true condition number in the 1-norm is 71.51.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07VUF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, KDMAX, LDAB
      PARAMETER        (NMAX=8,KDMAX=NMAX,LDAB=KDMAX+1)
      CHARACTER        NORM, DIAG
      PARAMETER        (NORM='1',DIAG='N')
*     .. Local Scalars ..
      real             RCOND
      INTEGER          I, INFO, J, KD, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          AB(LDAB,NMAX), WORK(2*NMAX)
      real             RWORK(NMAX)
*     .. External Functions ..
      real             X02AJF
      EXTERNAL         X02AJF
*     .. External Subroutines ..
      EXTERNAL         ctbcon
*     .. Intrinsic Functions ..
      INTRINSIC        MAX, MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07VUF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, KD
      IF (N.LE.NMAX .AND. KD.LE.KDMAX) THEN
*
*        Read A from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            DO 20 I = 1, N
               READ (NIN,*) (AB(KD+1+I-J,J),J=I,MIN(N,I+KD))
20          CONTINUE
```

```
            ELSE IF (UPLO.EQ.'L') THEN
               DO 40 I = 1, N
                  READ (NIN,*) (AB(1+I-J,J),J=MAX(1,I-KD),I)
   40          CONTINUE
            END IF
*
*           Estimate condition number
*
            CALL ctbcon(NORM,UPLO,DIAG,N,KD,AB,LDAB,RCOND,WORK,RWORK,INFO)
*
            WRITE (NOUT,*)
            IF (RCOND.GE.X02AJF()) THEN
               WRITE (NOUT,99999) 'Estimate of condition number =',
     +           1.0e0/RCOND
            ELSE
               WRITE (NOUT,*) 'A is singular to working precision'
            END IF
         END IF
         STOP
*
99999 FORMAT (1X,A,1P,e10.2)
      END
```

9.2. Program Data

```
F07VUF Example Program Data
  4  2                                                          :Values of N and KD
  'L'                                                           :Value of UPLO
 (-1.94,  4.43)
 (-3.39,  3.44) ( 4.12,-4.27)
 ( 1.62,  3.68) (-1.84,  5.53) ( 0.43,-2.66)
                (-2.77,-1.93) ( 1.74,-0.04) ( 0.44,  0.10)  :End of matrix A
```

9.3. Program Results

```
F07VUF Example Program Results

Estimate of condition number =  3.35E+01
```

F07VVF (CTBRFS/ZTBRFS) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F07VVF (CTBRFS/ZTBRFS) returns error bounds for the solution of a complex triangular band system of linear equations with multiple right-hand sides, $AX = B$, $A^TX = B$ or $A^HX = B$.

2. Specification

```
 SUBROUTINE F07VVF (UPLO, TRANS, DIAG, N, KD, NRHS, AB, LDAB, B, LDB, X,
1                   LDX, FERR, BERR, WORK, RWORK, INFO)
 ENTRY       ctbrfs  (UPLO, TRANS, DIAG, N, KD, NRHS, AB, LDAB, B, LDB, X,
1                   LDX, FERR, BERR, WORK, RWORK, INFO)
 INTEGER          N, KD, NRHS, LDAB, LDB, LDX, INFO
 real             FERR(*), BERR(*), RWORK(*)
 complex          AB(LDAB,*), B(LDB,*), X(LDX,*), WORK(*)
 CHARACTER*1      UPLO, TRANS, DIAG
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine returns the backward errors and estimated bounds on the forward errors for the solution of a complex triangular band system of linear equations with multiple right-hand sides $AX = B$, $A^TX = B$ or $A^HX = B$. The routine handles each right-hand side vector (stored as a column of the matrix B) independently, so we describe the function of the routine in terms of a single right-hand side b and solution x.

Given a computed solution x, the routine computes the *component-wise backward error* β. This is the size of the smallest relative perturbation in each element of A and b such that x is the exact solution of a perturbed system

$$(A+\delta A)x = b + \delta b$$

$$|\delta a_{ij}| \leq \beta |a_{ij}| \quad \text{and} \quad |\delta b_i| \leq \beta |b_i|.$$

Then the routine estimates a bound for the *component-wise forward error* in the computed solution, defined by:

$$\max_i |x_i - \hat{x}_i| / \max_i |x_i|$$

where $\hat{x}$ is the true solution.

For details of the method, see the Chapter Introduction.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: UPLO – CHARACTER*1. *Input*

On entry: indicates whether A is upper or lower triangular as follows:

if UPLO = 'U', then A is upper triangular;

if UPLO = 'L', then A is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: TRANS – CHARACTER*1. *Input*

On entry: indicates the form of the equations as follows:

if TRANS = 'N', then the equations are of the form $AX = B$;

if TRANS = 'T', then the equations are of the form $A^TX = B$;

if TRANS = 'C', then the equations are of the form $A^HX = B$.

Constraint: TRANS = 'N', 'T' or 'C'.

3: DIAG – CHARACTER*1. *Input*

On entry: indicates whether A is a non-unit or unit triangular matrix as follows:

if DIAG = 'N', then A is a non-unit triangular matrix;

if DIAG = 'U', then A is a unit triangular matrix; the diagonal elements are not referenced and are assumed to be 1.

Constraint: DIAG = 'N' or 'U'.

4: N – INTEGER. *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

5: KD – INTEGER. *Input*

On entry: k, the number of super-diagonals of the matrix A if UPLO = 'U' or the number of sub-diagonals if UPLO = 'L'.

Constraint: KD $\geq$ 0.

6: NRHS – INTEGER. *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS $\geq$ 0.

7: AB(LDAB,*) – ***complex*** array. *Input*

Note: the second dimension of the array AB must be at least max(1,N).

On entry: the n by n triangular band matrix A, stored in rows 1 to $(k+1)$. More precisely, if UPLO = 'U', the elements of the upper triangle of A within the band must be stored with element a_{ij} in AB$(k+1+i-j,j)$ for $\max(1,j-k) \leq i \leq j$; if UPLO = 'L', the elements of the lower triangle of A within the band must be stored with element a_{ij} in AB$(1+i-j,j)$ for $j \leq i \leq \min(n,j+k)$. If DIAG = 'U', the diagonal elements of A are not referenced and are assumed to be 1.

8: LDAB – INTEGER. *Input*

On entry: the first dimension of the array AB as declared in the (sub)program from which F07VVF (CTBRFS/ZTBRFS) is called.

Constraint: LDAB $\geq$ KD + 1.

9: B(LDB,*) – ***complex*** array. *Input*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

10: LDB – INTEGER. *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07VVF (CTBRFS/ZTBRFS) is called.

Constraint: LDB $\geq$ max(1,N).

11: X(LDX,*) – ***complex*** array. *Input*

Note: the second dimension of the array X must be at least max(1,NRHS).

On entry: the n by r solution matrix X, as returned by F07VSF (CTBTRS/ZTBTRS).

12: LDX – INTEGER. *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which F07VVF (CTBRFS/ZTBRFS) is called.

Constraint: LDX ≥ max(1,N).

13: FERR(*) – ***real*** array. *Output*

Note: the dimension of the array FERR must be at least max(1,NRHS).

On exit: FERR(j) contains an estimated error bound for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

14: BERR(*) – ***real*** array. *Output*

Note: the dimension of the array BERR must be at least max(1,NRHS).

On exit: BERR(j) contains the component-wise backward error bound β for the jth solution vector, that is, the jth column of X, for $j = 1,2,...,r$.

15: WORK(*) – ***complex*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,2*N).

16: RWORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array RWORK must be at least max(1,N).

17: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The bounds returned in FERR are not rigorous, because they are estimated, not computed exactly; but in practice they almost always overestimate the actual error.

8. Further Comments

A call to this routine, for each right-hand side, involves solving a number of systems of linear equations of the form $Ax = b$ or $A^H x = b$; the number is usually 5 and never more than 11. Each solution involves approximately $8nk$ real floating-point operations (assuming $n \gg k$).

The real analogue of this routine is F07VHF (STBRFS/DTBRFS).

9. Example

To solve the system of equations $AX = B$ and to compute forward and backward error bounds, where

$$A = \begin{pmatrix} -1.94 + 4.43i & 0.00 + 0.00i & 0.00 + 0.00i & 0.00 + 0.00i \\ -3.39 + 3.44i & 4.12 - 4.27i & 0.00 + 0.00i & 0.00 + 0.00i \\ 1.62 + 3.68i & -1.84 + 5.53i & 0.43 - 2.66i & 0.00 + 0.00i \\ 0.00 + 0.00i & -2.77 - 1.93i & 1.74 - 0.04i & 0.44 + 0.10i \end{pmatrix} \text{ and}$$

$$B = \begin{pmatrix} -8.86 - 3.88i & -24.09 - 5.27i \\ -15.57 - 23.41i & -57.97 + 8.14i \\ -7.63 + 22.78i & 19.09 - 29.51i \\ -14.74 - 2.40i & 19.17 + 21.33i \end{pmatrix}.$$

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F07VVF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, KDMAX, LDAB, NRHMAX, LDB, LDX
      PARAMETER        (NMAX=8,KDMAX=NMAX,LDAB=KDMAX+1,NRHMAX=NMAX,
     +                 LDB=NMAX,LDX=NMAX)
      CHARACTER        TRANS, DIAG
      PARAMETER        (TRANS='N',DIAG='N')
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, KD, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          AB(LDAB,NMAX), B(LDB,NRHMAX), WORK(2*NMAX),
     +                 X(LDX,NMAX)
      real             BERR(NRHMAX), FERR(NRHMAX), RWORK(NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         ctbrfs, ctbtrs, F06TFF, X04DBF
*     .. Intrinsic Functions ..
      INTRINSIC        MAX, MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07VVF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, KD, NRHS
      IF (N.LE.NMAX .AND. KD.LE.KDMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file, and copy B to X
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            DO 20 I = 1, N
               READ (NIN,*) (AB(KD+1+I-J,J),J=I,MIN(N,I+KD))
   20       CONTINUE
         ELSE IF (UPLO.EQ.'L') THEN
            DO 40 I = 1, N
               READ (NIN,*) (AB(1+I-J,J),J=MAX(1,I-KD),I)
   40       CONTINUE
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
         CALL F06TFF('General',N,NRHS,B,LDB,X,LDX)
*
*        Compute solution in the array X
*
         CALL ctbtrs(UPLO,TRANS,DIAG,N,KD,NRHS,AB,LDAB,X,LDX,INFO)
*
*        Compute backward errors and estimated bounds on the
*        forward errors
*
         CALL ctbrfs(UPLO,TRANS,DIAG,N,KD,NRHS,AB,LDAB,B,LDB,X,LDX,
     +               FERR,BERR,WORK,RWORK,INFO)
*
```

```
*        Print solution
*
         WRITE (NOUT,*)
         IFAIL = 0
         CALL X04DBF('General',' ',N,NRHS,X,LDX,'Bracketed','F7.4',
     +               'Solution(s)','Integer',RLABS,'Integer',CLABS,80,0,
     +               IFAIL)
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Backward errors (machine-dependent)'
         WRITE (NOUT,99999) (BERR(J),J=1,NRHS)
         WRITE (NOUT,*)
     +     'Estimated forward error bounds (machine-dependent)'
         WRITE (NOUT,99999) (FERR(J),J=1,NRHS)
       END IF
       STOP
*
99999 FORMAT ((5X,1P,4(e11.1,7X)))
      END
```

9.2. Program Data

```
F07VVF Example Program Data
  4  2  2                                                :Values of N, KD and NRHS
  'L'                                                      :Value of UPLO
 (-1.94, 4.43)
 (-3.39, 3.44) ( 4.12,-4.27)
 ( 1.62, 3.68) (-1.84, 5.53) ( 0.43,-2.66)
               (-2.77,-1.93) ( 1.74,-0.04) ( 0.44, 0.10)  :End of matrix A
 ( -8.86, -3.88) (-24.09, -5.27)
 (-15.57,-23.41) (-57.97,  8.14)
 ( -7.63, 22.78) ( 19.09,-29.51)
 (-14.74, -2.40) ( 19.17, 21.33)                          :End of matrix B
```

9.3. Program Results

```
 F07VVF Example Program Results

 Solution(s)
                        1                   2
  1   ( 0.0000, 2.0000) ( 1.0000, 5.0000)
  2   ( 1.0000,-3.0000) (-7.0000,-2.0000)
  3   (-4.0000,-5.0000) ( 3.0000, 4.0000)
  4   ( 2.0000,-1.0000) (-6.0000,-9.0000)

 Backward errors (machine-dependent)
         8.3E-18           4.2E-17
 Estimated forward error bounds (machine-dependent)
         1.8E-14           2.2E-14
```

Chapter F08 – Least-squares and Eigenvalue Problems (LAPACK)

Note. Please refer to the Users' Note for your implementation to check that a routine is available.

Routine Name	Mark of Introduction	Purpose
F08AEF	16	(SGEQRF/DGEQRF) QR factorization of real general rectangular matrix
F08AFF	16	(SORGQR/DORGQR) Form all or part of orthogonal Q from QR factorization determined by F08AEF or F08BEF
F08AGF	16	(SORMQR/DORMQR) Apply orthogonal transformation determined by F08AEF or F08BEF
F08AHF	16	(SGELQF/DGELQF) LQ factorization of real general rectangular matrix
F08AJF	16	(SORGLQ/DORGLQ) Form all or part of orthogonal Q from LQ factorization determined by F08AHF
F08AKF	16	(SORMLQ/DORMLQ) Apply orthogonal transformation determined by F08AHF
F08ASF	16	(CGEQRF/ZGEQRF) QR factorization of complex general rectangular matrix
F08ATF	16	(CUNGQR/ZUNGQR) Form all or part of unitary Q from QR factorization determined by F08ASF or F08BSF
F08AUF	16	(CUNMQR/ZUNMQR) Apply unitary transformation determined by F08ASF or F08BSF
F08AVF	16	(CGELQF/ZGELQF) LQ factorization of complex general rectangular matrix
F08AWF	16	(CUNGLQ/ZUNGLQ) Form all or part of unitary Q from LQ factorization determined by F08AVF
F08AXF	16	(CUNMLQ/ZUNMLQ) Apply unitary transformation determined by F08AVF
F08BEF	16	(SGEQPF/DGEQPF) QR factorization of real general rectangular matrix with column pivoting
F08BSF	16	(CGEQPF/ZGEQPF) QR factorization of complex general rectangular matrix with column pivoting
F08FCF	19	(SSYEVD/DSYEVD) All eigenvalues and optionally all eigenvectors of real symmetric matrix, using divide and conquer
F08FEF	16	(SSYTRD/DSYTRD) Orthogonal reduction of real symmetric matrix to symmetric tridiagonal form
F08FFF	16	(SORGTR/DORGTR) Generate orthogonal transformation matrix from reduction to tridiagonal form determined by F08FEF
F08FGF	16	(SORMTR/DORMTR) Apply orthogonal transformation determined by F08FEF
F08FQF	19	(CHEEVD/ZHEEVD) All eigenvalues and optionally all eigenvectors of complex Hermitian matrix, using divide and conquer
F08FSF	16	(CHETRD/ZHETRD) Unitary reduction of complex Hermitian matrix to real symmetric tridiagonal form
F08FTF	16	(CUNGTR/ZUNGTR) Generate unitary transformation matrix from reduction to tridiagonal form determined by F08FSF
F08FUF	16	(CUNMTR/ZUNMTR) Apply unitary transformation matrix determined by F08FSF
F08GCF	19	(SSPEVD/DSPEVD) All eigenvalues and optionally all eigenvectors of real symmetric matrix, packed storage, using divide and conquer
F08GEF	16	(SSPTRD/DSPTRD) Orthogonal reduction of real symmetric matrix to symmetric tridiagonal form, packed storage
F08GFF	16	(SOPGTR/DOPGTR) Generate orthogonal transformation matrix from reduction to tridiagonal form determined by F08GEF

F08GGF	16	(SOPMTR/DOPMTR) Apply orthogonal transformation determined by F08GEF
F08GQF	19	(CHPEVD/ZHPEVD) All eigenvalues and optionally all eigenvectors of complex Hermitian matrix, packed storage, using divide and conquer
F08GSF	16	(CHPTRD/ZHPTRD) Unitary reduction of complex Hermitian matrix to real symmetric tridiagonal form, packed storage
F08GTF	16	(CUPGTR/ZUPGTR) Generate unitary transformation matrix from reduction to tridiagonal form determined by F08GSF
F08GUF	16	(CUPMTR/ZUPMTR) Apply unitary transformation matrix determined by F08GSF
F08HCF	19	(SSBEVD/DSBEVD) All eigenvalues and optionally all eigenvectors of real symmetric band matrix, using divide and conquer
F08HEF	16	(SSBTRD/DSBTRD) Orthogonal reduction of real symmetric band matrix to symmetric tridiagonal form
F08HQF	19	(CHBEVD/ZHBEVD) All eigenvalues and optionally all eigenvectors of complex Hermitian band matrix, using divide and conquer
F08HSF	16	(CHBTRD/ZHBTRD) Unitary reduction of complex Hermitian band matrix to real symmetric tridiagonal form
F08JCF	19	(SSTEVD/DSTEVD) All eigenvalues and optionally all eigenvectors of real symmetric tridiagonal matrix, using divide and conquer
F08JEF	16	(SSTEQR/DSTEQR) All eigenvalues and eigenvectors of real symmetric tridiagonal matrix, reduced from real symmetric matrix using implicit QL or QR
F08JFF	16	(SSTERF/DSTERF) All eigenvalues of real symmetric tridiagonal matrix, root-free variant of QL or QR
F08JGF	16	(SPTEQR/DPTEQR) All eigenvalues and eigenvectors of real symmetric positive-definite tridiagonal matrix, reduced from real symmetric positive-definite matrix
F08JJF	16	(SSTEBZ/DSTEBZ) Selected eigenvalues of real symmetric tridiagonal matrix by bisection
F08JKF	16	(SSTEIN/DSTEIN) Selected eigenvectors of real symmetric tridiagonal matrix by inverse iteration, storing eigenvectors in real array
F08JSF	16	(CSTEQR/ZSTEQR) All eigenvalues and eigenvectors of real symmetric tridiagonal matrix, reduced from complex Hermitian matrix, using implicit QL or QR
F08JUF	16	(CPTEQR/ZPTEQR) All eigenvalues and eigenvectors of real symmetric positive-definite tridiagonal matrix, reduced from complex Hermitian positive-definite matrix
F08JXF	16	(CSTEIN/ZSTEIN) Selected eigenvectors of real symmetric tridiagonal matrix by inverse iteration, storing eigenvectors in complex array
F08KEF	16	(SGEBRD/DGEBRD) Orthogonal reduction of real general rectangular matrix to bidiagonal form
F08KFF	16	(SORGBR/DORGBR) Generate orthogonal transformation matrices from reduction to bidiagonal form determined by F08KEF
F08KGF	16	(SORMBR/DORMBR) Apply orthogonal transformations from reduction to bidiagonal form determined by F08KEF
F08KSF	16	(CGEBRD/ZGEBRD) Unitary reduction of complex general rectangular matrix to bidiagonal form
F08KTF	16	(CUNGBR/ZUNGBR) Generate unitary transformation matrices from reduction to bidiagonal form determined by F08KSF
F08KUF	16	(CUNMBR/ZUNMBR) Apply unitary transformations from reduction to bidiagonal form determined by F08KSF
F08LEF	19	(SGBBRD/DGBBRD) Reduction of real rectangular band matrix to upper bidiagonal form
F08LSF	19	(CGBBRD/ZGBBRD) Reduction of complex rectangular band matrix to upper bidiagonal form
F08MEF	16	(SBDSQR/DBDSQR) SVD of real bidiagonal matrix reduced from real general matrix

F08MSF	16	(CBDSQR/ZBDSQR) SVD of real bidiagonal matrix reduced from complex general matrix
F08NEF	16	(SGEHRD/DGEHRD) Orthogonal reduction of real general matrix to upper Hessenberg form
F08NFF	16	(SORGHR/DORGHR) Generate orthogonal transformation matrix from reduction to Hessenberg form determined by F08NEF
F08NGF	16	(SORMHR/DORMHR) Apply orthogonal transformation matrix from reduction to Hessenberg form determined by F08NEF
F08NHF	16	(SGEBAL/DGEBAL) Balance real general matrix
F08NJF	16	(SGEBAK/DGEBAK) Transform eigenvectors of real balanced matrix to those of original matrix supplied to F08NHF
F08NSF	16	(CGEHRD/ZGEHRD) Unitary reduction of complex general matrix to upper Hessenberg form
F08NTF	16	(CUNGHR/ZUNGHR) Generate unitary transformation matrix from reduction to Hessenberg form determined by F08NSF
F08NUF	16	(CUNMHR/ZUNMHR) Apply unitary transformation matrix from reduction to Hessenberg form determined by F08NSF
F08NVF	16	(CGEBAL/ZGEBAL) Balance complex general matrix
F08NWF	16	(CGEBAK/ZGEBAK) Transform eigenvectors of complex balanced matrix to those of original matrix supplied to F08NVF
F08PEF	16	(SHSEQR/DHSEQR) Eigenvalues and Schur factorization of real upper Hessenberg matrix reduced from real general matrix
F08PKF	16	(SHSEIN/DHSEIN) Selected right and/or left eigenvectors of real upper Hessenberg matrix by inverse iteration
F08PSF	16	(CHSEQR/ZHSEQR) Eigenvalues and Schur factorization of complex upper Hessenberg matrix reduced from complex general matrix
F08PXF	16	(CHSEIN/ZHSEIN) Selected right and/or left eigenvectors of complex upper Hessenberg matrix by inverse iteration
F08QFF	16	(STREXC/DTREXC) Reorder Schur factorization of real matrix using orthogonal similarity transformation
F08QGF	16	(STRSEN/DTRSEN) Reorder Schur factorization of real matrix, form orthonormal basis of right invariant subspace for selected eigenvalues, with estimates of sensitivities
F08QHF	16	(STRSYL/DTRSYL) Solve real Sylvester matrix equation $AX + XB = C$, A and B are upper quasi-triangular or transposes
F08QKF	16	(STREVC/DTREVC) Left and right eigenvectors of real upper quasi-triangular matrix
F08QLF	16	(STRSNA/DTRSNA) Estimates of sensitivities of selected eigenvalues and eigenvectors of real upper quasi-triangular matrix
F08QTF	16	(CTREXC/ZTREXC) Reorder Schur factorization of complex matrix using unitary similarity transformation
F08QUF	16	(CTRSEN/ZTRSEN) Reorder Schur factorization of complex matrix, form orthonormal basis of right invariant subspace for selected eigenvalues, with estimates of sensitivities
F08QVF	16	(CTRSYL/ZTRSYL) Solve complex Sylvester matrix equation $AX + XB = C$, A and B are upper triangular or conjugate-transposes
F08QXF	16	(CTREVC/ZTREVC) Left and right eigenvectors of complex upper triangular matrix
F08QYF	16	(CTRSNA/ZTRSNA) Estimates of sensitivities of selected eigenvalues and eigenvectors of complex upper triangular matrix
F08SEF	16	(SSYGST/DSYGST) Reduction to standard form of real symmetric-definite generalized eigenproblem $Ax = \lambda Bx$, $ABx = \lambda x$ or $BAx = \lambda x$, B factorized by F07FDF
F08SSF	16	(CHEGST/ZHEGST) Reduction to standard form of complex Hermitian-definite generalized eigenproblem $Ax = \lambda Bx$, $ABx = \lambda x$ or $BAx = \lambda x$, B factorized by F07FRF

F08TEF	16	(SSPGST/DSPGST) Reduction to standard form of real symmetric-definite generalized eigenproblem $Ax = \lambda Bx$, $ABx = \lambda x$ or $BAx = \lambda x$, packed storage, B factorized by F07GDF
F08TSF	16	(CHPGST/ZHPGST) Reduction to standard form of complex Hermitian-definite generalized eigenproblem $Ax = \lambda Bx$, $ABx = \lambda x$ or $BAx = \lambda x$, packed storage, B factorized by F07GRF
F08UEF	19	(SSBGST/DSBGST) Reduction of real symmetric-definite banded generalized eigenproblem $Ax = \lambda Bx$ to standard form $Cy = \lambda y$, such that C has the same bandwidth as A
F08UFF	19	(SPBSTF/DPBSTF) Computes a split Cholesky factorization of real symmetric positive-definite band matrix A
F08USF	19	(CHBGST/ZHBGST) Reduction of complex Hermitian-definite banded generalized eigenproblem $Ax = \lambda Bx$ to standard form $Cy = \lambda y$, such that C has the same bandwidth as A
F08UTF	19	(CPBSTF/ZPBSTF) Computes a split Cholesky factorization of complex Hermitian positive-definite band matrix A

Chapter F08

Least-squares and Eigenvalue Problems (LAPACK)

Contents

1 Scope of the Chapter

This chapter provides routines for the solution of linear least-squares problems, eigenvalue problems and singular value problems, as well as associated computations. It provides routines for:

- solution of linear least-squares problems
- solution of symmetric eigenvalue problems
- solution of nonsymmetric eigenvalue problems
- solution of singular value problems
- solution of generalized symmetric-definite eigenvalue problems
- matrix factorizations associated with the above problems
- estimating condition numbers of eigenvalues and eigenvectors
- estimating the numerical rank of a matrix
- solution of the Sylvester matrix equation

Routines are provided for both *real* and *complex* data.

For a general introduction to the solution of linear least-squares problems, you should turn first to the the F04 Chapter Introduction. The decision trees, at the end of the the F04 Chapter Introduction, direct you to the most appropriate routines in Chapter F04 or Chapter F08. Chapter F04 contains *Black Box* routines which enable standard linear least-squares problems to be solved by a call to a single routine.

For a general introduction to eigenvalue and singular value problems, you should turn first to the the F02 Chapter Introduction. The decision trees, at the end of the the F02 Chapter Introduction, direct you to the most appropriate routines in Chapter F02. Chapter F02 contains *Black Box* routines which enable some standard types of problem to be solved by a call to a single routine. Often routines in Chapter F02 call Chapter F08 routines to perform the necessary computational tasks. However, divide and conquer algorithms for symmetric (Hermitian) eigenvalue problem are available only in this chapter and they can be considered as *Black Box* routines.

The routines in this chapter (F08) handle only *dense*, *band*, *tridiagonal* and *Hessenberg* matrices (not matrices with more specialized structures, or general sparse matrices). The decision trees in Section 4 direct you to the most appropriate routines in Chapter F08.

The routines in this chapter have all been derived from the LAPACK project (see Anderson *et al.* [1]). They have been designed to be efficient on a wide range of high-performance computers, without compromising efficiency on conventional serial machines.

It is not expected that every user will need to read all of the following sections, but rather will pick out those sections relevant to their particular problem.

2 Background to the Problems

This section is only a brief introduction to the numerical solution of linear least-squares problems, eigenvalue and singular value problems. Consult a standard textbook for a more thorough discussion, for example Golub and Van Loan [4].

2.1 Linear Least-squares Problems

The *linear least-squares problem* is

$$\underset{x}{\text{minimize}} \|b - Ax\|_2, \tag{1}$$

where A is an m by n matrix, b is a given m element vector and x is the n element solution vector.

In the most usual case $m \geq n$ and $\text{rank}(A) = n$, so that A has *full rank* and in this case the solution to problem (1) is unique; the problem is also referred to as finding a *least-squares solution* to an *overdetermined* system of linear equations.

When $m < n$ and $\text{rank}(A) = m$, there are an infinite number of solutions x which exactly satisfy $b - Ax = 0$. In this case it is often useful to find the unique solution x which minimizes $\|x\|_2$, and

the problem is referred to as finding a *minimum-norm solution* to an *underdetermined* system of linear equations.

In the general case when we may have $\text{rank}(A) < \min(m, n)$ – in other words, A may be *rank-deficient* – we seek the *minimum-norm least-squares* solution x which minimizes both $||x||_2$ and $||b - Ax||_2$.

This chapter (F08) contains computational routines that can be combined with routines in Chapter F07 to solve these linear least-squares problems. Chapter F04 contains Black Box routines to solve these linear least-squares problems in standard cases. The next two sections discuss the factorizations that can be used in the solution of linear least-squares problems.

2.2 Orthogonal Factorizations and Least-squares Problems

A number of routines are provided for factorizing a general rectangular m by n matrix A, as the product of an *orthogonal* matrix (*unitary* if complex) and a *triangular* (or possibly trapezoidal) matrix.

A real matrix Q is *orthogonal* if $Q^T Q = I$; a complex matrix Q is *unitary* if $Q^H Q = I$. Orthogonal or unitary matrices have the important property that they leave the two-norm of a vector invariant, so that

$$||x||_2 = ||Qx||_2, \text{ if } Q \text{ is orthogonal or unitary.}$$

They usually help to maintain numerical stability because they do not amplify rounding errors.

Orthogonal factorizations are used in the solution of linear least-squares problems. They may also be used to perform preliminary steps in the solution of eigenvalue or singular value problems, and are useful tools in the solution of a number of other problems.

2.2.1 QR factorization

The most common, and best known, of the factorizations is the QR *factorization* given by

$$A = Q\begin{pmatrix} R \\ 0 \end{pmatrix}, \text{ if } m \geq n,$$

where R is an n by n upper triangular matrix and Q is an m by m orthogonal (or unitary) matrix. If A is of full rank n, then R is non-singular. It is sometimes convenient to write the factorization as

$$A = (Q_1 \; Q_2)\begin{pmatrix} R \\ 0 \end{pmatrix}$$

which reduces to

$$A = Q_1 R,$$

where Q_1 consists of the first n columns of Q, and Q_2 the remaining $m - n$ columns.

If $m < n$, R is trapezoidal, and the factorization can be written

$$A = Q\,(R_1 \; R_2), \text{if } m < n,$$

where R_1 is upper triangular and R_2 is rectangular.

The QR factorization can be used to solve the linear least-squares problem (1) when $m \geq n$ and A is of full rank, since

$$||b - Ax||_2 = ||Q^T b - Q^T Ax||_2 = \left\| \begin{matrix} c_1 - Rx \\ c_2 \end{matrix} \right\|,$$

where

$$c \equiv \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} Q_1^T b \\ Q_2^T b \end{pmatrix} = Q^T b;$$

and c_1 is an n element vector. Then x is the solution of the upper triangular system

$$Rx = c_1.$$

The residual vector r is given by

$$r = b - Ax = Q\begin{pmatrix} 0 \\ c_2 \end{pmatrix}.$$

The residual sum of squares $||r||_2^2$ may be computed without forming r explicitly, since

$$||r||_2 = ||b - Ax||_2 = ||c_2||_2.$$

2.2.2 *LQ* factorization

The *LQ factorization* is given by

$$A = (L \;\; 0)Q = (L \;\; 0)\begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix} = LQ_1, \text{ if } m \leq n,$$

where L is m by m lower triangular, Q is n by n orthogonal (or unitary), Q_1 consists of the first m rows of Q, and Q_2 the remaining $n - m$ rows.

The LQ factorization of A is essentially the same as the QR factorization of A^T (A^H if A is complex), since

$$A = (L \;\; 0)\,Q \Leftrightarrow A^T = Q^T \begin{pmatrix} L^T \\ 0 \end{pmatrix}.$$

The LQ factorization may be used to find a minimum norm solution of an underdetermined system of linear equations $Ax = b$ where A is m by n with $m < n$ and has rank m. The solution is given by

$$x = Q^T \begin{pmatrix} L^{-1}b \\ 0 \end{pmatrix}.$$

2.2.3 *QR* factorization with column pivoting

To solve a linear least-squares problem (1) when A is not of full rank, or the rank of A is in doubt, we can perform either a QR factorization with column pivoting or a singular value decomposition.

The *QR factorization with column pivoting* is given by

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix} P^T, \quad m \geq n,$$

where Q and R are as before and P is a (real) permutation matrix, chosen (in general) so that

$$|r_{11}| \geq |r_{22}| \geq \cdots \geq |r_{nn}|$$

and moreover, for each k,

$$|r_{kk}| \geq \|R_{k:j,j}\|_2 \quad \text{for } j = k+1, \ldots, n.$$

If we put

$$R = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$$

where R_{11} is the leading k by k upper triangular submatrix of R then, in exact arithmetic, if $\text{rank}(A) = k$, the whole of the submatrix R_{22} in rows and columns $k+1$ to n would be zero. In numerical computation, the aim must be to determine an index k, such that the leading submatrix R_{11} is well-conditioned, and R_{22} is negligible, so that

$$R = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} \simeq \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix}.$$

Then k is the effective rank of A. See Golub and Van Loan [4] for a further discussion of numerical rank determination.

The so-called basic solution to the linear least-squares problem (1) can be obtained from this factorization as

$$x = P\begin{pmatrix} R_{11}^{-1}\hat{c}_1 \\ 0 \end{pmatrix},$$

where $\hat{c}_1$ consists of just the first k elements of $c = Q^T b$.

2.3 The Singular Value Decomposition

The *singular value decomposition* (SVD) of an m by n matrix A is given by

$$A = U\Sigma V^T, \quad (A = U\Sigma V^H \text{ in the complex case})$$

where U and V are orthogonal (unitary) and Σ is an m by n diagonal matrix with real diagonal elements, σ_i, such that

$$\sigma_1 \geq \sigma_2 \geq \cdots \sigma_{\min(m,n)} \geq 0.$$

The σ_i are the *singular values* of A and the first $\min(m,n)$ columns of U and V are the *left* and *right singular vectors* of A. The singular values and singular vectors satisfy

$$Av_i = \sigma_i u_i \text{ and } A^T u_i = \sigma_i v_i \text{ (or } A^H u_i = \sigma_i v_i)$$

where u_i and v_i are the ith columns of U and V respectively.

The computation proceeds in the following stages.

(1) The matrix A is reduced to bidiagonal form $A = U_1 B V_1^T$ if A is real ($A = U_1 B V_1^H$ if A is complex), where U_1 and V_1 are orthogonal (unitary if A is complex), and B is real and upper bidiagonal when $m \geq n$ and lower bidiagonal when $m < n$, so that B is nonzero only on the main diagonal and either on the first superdiagonal (if $m \geq n$) or the first subdiagonal (if $m < n$).

(2) The SVD of the bidiagonal matrix B is computed as $B = U_2 \Sigma V_2^T$, where U_2 and V_2 are orthogonal and Σ is diagonal as described above. The singular vectors of A are then $U = U_1 U_2$ and $V = V_1 V_2$.

If $m \gg n$, it may be more efficient to first perform a QR factorization of A, and then compute the SVD of the n by n matrix R, since if $A = QR$ and $R = U \Sigma V^T$, then the SVD of A is given by $A = (QU)\Sigma V^T$.

Similarly, if $m \ll n$, it may be more efficient to first perform an LQ factorization of A.

2.4 The Singular Value Decomposition and Least-squares Problems

The SVD may be used to find a minimum norm solution to a (possibly) rank-deficient linear least-squares problem (1). The effective rank, k, of A can be determined as the number of singular values which exceed a suitable threshold. Let $\hat{\Sigma}$ be the leading k by k submatrix of Σ, and $\hat{V}$ be the matrix consisting of the first k columns of V. Then the solution is given by

$$x = \hat{V} \hat{\Sigma}^{-1} \hat{c}_1,$$

where $\hat{c}_1$ consists of the first k elements of $c = U^T b = U_2^T U_1^T b$.

2.5 Symmetric Eigenvalue Problems

The *symmetric eigenvalue problem* is to find the *eigenvalues*, λ, and corresponding *eigenvectors*, $z \neq 0$, such that

$$Az = \lambda z, \quad A = A^T, \quad \text{where } A \text{ is real.}$$

For the *Hermitian eigenvalue problem* we have

$$Az = \lambda z, \quad A = A^H, \quad \text{where } A \text{ is complex.}$$

For both problems the eigenvalues λ are real.

When all eigenvalues and eigenvectors have been computed, we write

$$A = Z \Lambda Z^T \text{ (or } A = Z \Lambda Z^H \text{ if complex),}$$

where Λ is a diagonal matrix whose diagonal elements are the eigenvalues, and Z is an orthogonal (or unitary) matrix whose columns are the eigenvectors. This is the classical *spectral factorization* of A.

The basic task of the symmetric eigenproblem routines is to compute values of λ and, optionally, corresponding vectors z for a given matrix A. This computation proceeds in the following stages.

(1) The real symmetric or complex Hermitian matrix A is reduced to *real tridiagonal form* T. If A is real symmetric this decomposition is $A = QTQ^T$ with Q orthogonal and T symmetric tridiagonal. If A is complex Hermitian, the decomposition is $A = QTQ^H$ with Q unitary and T, as before, *real* symmetric tridiagonal.

(2) Eigenvalues and eigenvectors of the real symmetric tridiagonal matrix T are computed. If all eigenvalues and eigenvectors are computed, this is equivalent to factorizing T as $T = S \Lambda S^T$, where S is orthogonal and Λ is diagonal. The diagonal entries of Λ are the eigenvalues of T, which are also the eigenvalues of A, and the columns of S are the eigenvectors of T; the eigenvectors of A are the columns of $Z = QS$, so that $A = Z \Lambda Z^T$ ($Z \Lambda Z^H$ when A is complex Hermitian).

This chapter now supports three primary algorithms for computing eigenvalues and eigenvectors of real symmetric matrices and complex Hermitian matrices. They are:

(i) the divide and conquer algorithm;
(ii) the QR algorithm;
(iii) bisection followed by inverse iteration.

The divide and conquer algorithm is generally more efficient than the traditional QR algorithm and is recommended for computing all eigenvalues and eigenvectors. Furthermore, eigenvalues and eigenvectors can be obtained by calling one single routine in the case of the divide and conquer algorithm. In general, more than one routine has to be called if the QR algorithm or bisection followed by inverse iteration is used.

2.6 Generalized Symmetric-Definite Eigenvalue Problems

This section is concerned with the solution of the generalized eigenvalue problems $Az = \lambda Bz$, $ABz = \lambda z$, and $BAz = \lambda z$, where A and B are real symmetric or complex Hermitian and B is positive-definite. Each of these problems can be reduced to a standard symmetric eigenvalue problem, using a Cholesky factorization of B as either $B = LL^T$ or $B = U^TU$ (LL^H or U^HU in the Hermitian case).

With $B = LL^T$, we have

$$Az = \lambda Bz \Rightarrow (L^{-1}AL^{-T})(L^Tz) = \lambda(L^Tz).$$

Hence the eigenvalues of $Az = \lambda Bz$ are those of $Cy = \lambda y$, where C is the symmetric matrix $C = L^{-1}AL^{-T}$ and $y = L^Tz$. In the complex case C is Hermitian with $C = L^{-1}AL^{-H}$ and $y = L^Hz$.

Table 1 summarizes how each of the three types of problem may be reduced to standard form $Cy = \lambda y$, and how the eigenvectors z of the original problem may be recovered from the eigenvectors y of the reduced problem. The table applies to real problems; for complex problems, transposed matrices must be replaced by conjugate-transposes.

	Type of problem	**Factorization of B**	**Reduction**	**Recovery of eigenvectors**
1.	$Az = \lambda Bz$	$B = LL^T$	$C = L^{-1}AL^{-T}$	$z = L^{-T}y$
		$B = U^TU$	$C = U^{-T}AU^{-1}$	$z = U^{-1}y$
2.	$ABz = \lambda z$	$B = LL^T$	$C = L^TAL$	$z = L^{-T}y$
		$B = U^TU$	$C = UAU^T$	$z = U^{-1}y$
3.	$BAz = \lambda z$	$B = LL^T$	$C = L^TAL$	$z = Ly$
		$B = U^TU$	$C = UAU^T$	$z = U^Ty$

Table 1
Reduction of generalized symmetric-definite eigenproblems to standard problems

When the generalized symmetric-definite problem has been reduced to the corresponding standard problem $Cy = \lambda y$, this may then be solved using the routines described in the previous section. No special routines are needed to recover the eigenvectors z of the generalized problem from the eigenvectors y of the standard problem, because these computations are simple applications of Level 2 or Level 3 BLAS (see Chapter F06).

2.7 Packed Storage for Symmetric Matrices

Routines which handle symmetric matrices are usually designed so that they use either the upper or lower triangle of the matrix; it is not necessary to store the whole matrix. If either the upper or lower triangle is stored conventionally in the upper or lower triangle of a two-dimensional array, the remaining elements of the array can be used to store other useful data. However, that is not always convenient, and if it is important to economize on storage, the upper or lower triangle can be stored in a one-dimensional array of length $n(n+1)/2$; that is, the storage is almost halved.

This storage format is referred to as *packed storage*; it is described in Section 3.3.

Routines designed for packed storage are usually less efficient, especially on high-performance computers, so there is a trade-off between storage and efficiency.

2.8 Band Matrices

A *band* matrix is one whose elements are confined to a relatively small number of sub-diagonals or super-diagonals on either side of the main diagonal. Algorithms can take advantage of bandedness to reduce the amount of work and storage required. The storage scheme for band matrices is described in Section 3.3.

If the problem is the generalized symmetric definite eigenvalue problem $Az = \lambda Bz$ and the matrices A and B are additionally banded, the matrix C as defined in Section 2.6 is, in general, full. We can reduce the problem to a banded standard problem by modifying the definition of C thus:

$$C = X^T A X, \quad \text{where } X = U^{-1}Q \text{ or } L^{-T}Q,$$

where Q is an orthogonal matrix chosen to ensure that C has bandwidth no greater than that of A.

A further refinement is possible when A and B are banded, which halves the amount of work required to form C. Instead of the standard Cholesky factorization of B as $U^T U$ or LL^T, we use a *split Cholesky* factorization $B = S^T S$, where

$$S = \begin{pmatrix} U_{11} & \\ M_{21} & L_{22} \end{pmatrix}$$

with U_{11} upper triangular and L_{22} lower triangular of order approximately $n/2$; S has the same bandwidth as B.

2.9 Nonsymmetric Eigenvalue Problems

The *nonsymmetric eigenvalue problem* is to find the *eigenvalues*, λ, and corresponding *eigenvectors*, $v \neq 0$, such that

$$Av = \lambda v.$$

More precisely, a vector v as just defined is called a *right eigenvector* of A, and a vector $u \neq 0$ satisfying

$$u^T A = \lambda u^T \quad (u^H A = \lambda u^H \text{ when } u \text{ is complex})$$

is called a *left eigenvector* of A.

A real matrix A may have complex eigenvalues, occurring as complex conjugate pairs.

This problem can be solved via the *Schur factorization* of A, defined in the real case as

$$A = ZTZ^T,$$

where Z is an orthogonal matrix and T is an upper quasi-triangular matrix with 1 by 1 and 2 by 2 diagonal blocks, the 2 by 2 blocks corresponding to complex conjugate pairs of eigenvalues of A. In the complex case, the Schur factorization is

$$A = ZTZ^H,$$

where Z is unitary and T is a complex upper triangular matrix.

The columns of Z are called the *Schur vectors*. For each k ($1 \leq k \leq n$), the first k columns of Z form an orthonormal basis for the *invariant subspace* corresponding to the first k eigenvalues on the diagonal of T. Because this basis is orthonormal, it is preferable in many applications to compute Schur vectors rather than eigenvectors. It is possible to order the Schur factorization so that any desired set of k eigenvalues occupy the k leading positions on the diagonal of T.

The two basic tasks of the nonsymmetric eigenvalue routines are to compute, for a given matrix A, all n values of λ and, if desired, their associated right eigenvectors v and/or left eigenvectors u, and the Schur factorization.

These two basic tasks can be performed in the following stages.

(1) A general matrix A is reduced to *upper Hessenberg form* H which is zero below the first subdiagonal. The reduction may be written $A = QHQ^T$ with Q orthogonal if A is real, or $A = QHQ^H$ with Q unitary if A is complex.

(2) The upper Hessenberg matrix H is reduced to Schur form T, giving the Schur factorization $H = STS^T$ (for H real) or $H = STS^H$ (for H complex). The matrix S (the Schur vectors of H) may optionally be computed as well. Alternatively S may be postmultiplied into the matrix Q determined in stage 1, to give the matrix $Z = QS$, the Schur vectors of A. The eigenvalues are obtained from the diagonal elements or diagonal blocks of T.

(3) Given the eigenvalues, the eigenvectors may be computed in two different ways. Inverse iteration can be performed on H to compute the eigenvectors of H, and then the eigenvectors can be multiplied by the matrix Q in order to transform them to eigenvectors of A. Alternatively the eigenvectors of T can be computed, and optionally transformed to those of H or A if the matrix S or Z is supplied.

The accuracy with which eigenvalues can be obtained can often be improved by *balancing* a matrix. This is discussed further in Section 2.11.6 below.

2.10 The Sylvester Equation

The Sylvester equation is a matrix equation of the form

$$AX + XB = C,$$

where A, B, and C are given matrices with A being m by m, B an n by n matrix and C, and the solution matrix X, m by n matrices. The solution of a special case of this equation occurs in the computation of the condition number for an invariant subspace, but a combination of routines in this chapter allows the solution of the general Sylvester equation.

2.11 Error and Perturbation Bounds and Condition Numbers

In this section we discuss the effects of rounding errors in the solution process and the effects of uncertainties in the data, on the solution to the problem. A number of the routines in this chapter return information, such as condition numbers, that allow these effects to be assessed. First we discuss some notation used in the error bounds of later sections.

The bounds usually contain the factor $p(n)$ (or $p(m,n)$), which grows as a function of the matrix dimension n (or matrix dimensions m and n). It measures how errors can grow as a function of the matrix dimension, and represents a potentially different function for each problem. In practice, it usually grows just linearly; $p(n) \leq 10n$ is often true, although generally only much weaker bounds can be actually proved. We normally describe $p(n)$ as a 'modestly growing' function of n. For detailed derivations of various $p(n)$, see [4] and [6].

For linear equation (see Chapter F07) and least-squares solvers, we consider bounds on the relative error $||x-\hat{x}||/||x||$ in the computed solution $\hat{x}$, where x is the true solution. For eigenvalue problems we consider bounds on the error $|\lambda_i - \hat{\lambda}_i|$ in the ith computed eigenvalue $\hat{\lambda}_i$, where λ_i is the true ith eigenvalue. For singular value problems we similarly consider bounds $|\sigma_i - \hat{\sigma}_i|$.

Bounding the error in computed eigenvectors and singular vectors $\hat{v}_i$ is more subtle because these vectors are not unique: even though we restrict $||\hat{v}_i||_2 = 1$ and $||v_i||_2 = 1$, we may still multiply them by arbitrary constants of absolute value 1. So to avoid ambiguity we bound the *angular difference* between $\hat{v}_i$ and the true vector v_i, so that

$$\begin{aligned} \theta(v_i, \hat{v}_i) &= \text{acute angle between } v_i \text{ and } \hat{v}_i \\ &= \arccos |v_i^H \hat{v}_i|. \end{aligned} \tag{2}$$

When $\theta(v_i, \hat{v}_i)$ is small, we can choose a constant α with absolute value 1 so that $||\alpha v_i - \hat{v}_i||_2 \approx \theta(v_i, \hat{v}_i)$.

In addition to bounds for individual eigenvectors, bounds can be obtained for the spaces spanned by collections of eigenvectors. These may be much more accurately determined than the individual eigenvectors which span them. These spaces are called *invariant subspaces* in the case of eigenvectors, because if v is any vector in the space, Av is also in the space, where A is the matrix. Again, we will use angle to measure the difference between a computed space $\hat{S}$ and the true space S:

$$\begin{aligned} \theta(S, \hat{S}) &= \text{acute angle between } S \text{ and } \hat{S} \\ &= \max_{\substack{s \in S \\ s \neq 0}} \min_{\substack{\hat{s} \in \hat{S} \\ \hat{s} \neq 0}} \theta(s, \hat{s}) \quad \text{or} \quad \max_{\substack{\hat{s} \in \hat{S} \\ \hat{s} \neq 0}} \min_{\substack{s \in S \\ s \neq 0}} \theta(s, \hat{s}) \end{aligned} \tag{3}$$

$\theta(S, \hat{S})$ may be computed as follows. Let S be a matrix whose columns are orthonormal and span S. Similarly let $\hat{S}$ be an orthonormal matrix with columns spanning $\hat{S}$. Then

$$\theta(S, \hat{S}) = \arccos \sigma_{\min}(S^H \hat{S}).$$

Finally, we remark on the accuracy of the bounds when they are large. Relative errors like $||\hat{x} - x||/||x||$ and angular errors like $\theta(\hat{v}_i, v_i)$ are only of interest when they are much less than 1. Some stated bounds

are not strictly true when they are close to 1, but rigorous bounds are much more complicated and supply little extra information in the interesting case of small errors. These bounds are indicated by using the symbol $\lesssim$, or 'approximately less than', instead of the usual $\leq$. Thus, when these bounds are close to 1 or greater, they indicate that the computed answer may have no significant digits at all, but do not otherwise bound the error.

2.11.1 Least-squares problems

The conventional error analysis of linear least-squares problems goes as follows. The problem is to find the x minimizing $||Ax - b||_2$. Let $\hat{x}$ be the solution computed using one of the methods described above. We discuss the most common case, where A is overdetermined (i.e., has more rows than columns) and has full rank.

Then the computed solution $\hat{x}$ has a small normwise backward error. In other words $\hat{x}$ minimizes $||(A+E)\hat{x} - (b+f)||_2$, where

$$\max\left(\frac{||E||_2}{||A||_2}, \frac{||f||_2}{||b||_2}\right) \leq p(n)\epsilon$$

and $p(n)$ is a modestly growing function of n and ϵ is the machine precision. Let $\kappa_2(A) = \sigma_{\max}(A)/\sigma_{\min}(A)$, $\rho = ||Ax - b||_2$, and $\sin(\theta) = \rho/||b||_2$. Then if $p(n)\epsilon$ is small enough, the error $\hat{x} - x$ is bounded by

$$\frac{||x - \hat{x}||_2}{||x||_2} \lesssim p(n)\epsilon\left\{\frac{2\kappa_2(A)}{\cos(\theta)} + \tan(\theta)\kappa_2^2(A)\right\}.$$

If A is rank-deficient, the problem can be *regularized* by treating all singular values less than a user-specified threshold as exactly zero. See [4] for error bounds in this case, as well as for the underdetermined case.

The solution of the overdetermined, full-rank problem may also be characterized as the solution of the linear system of equations

$$\begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix}\begin{pmatrix} r \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}.$$

By solving this linear system (see Chapter F07) componentwise error bounds can also be obtained [2].

2.11.2 The singular value decomposition

The usual error analysis of the SVD algorithm is as follows [4].

The computed SVD, $\hat{U}\hat{\Sigma}\hat{V}^T$, is nearly the exact SVD of $A+E$, i.e., $A+E = (\hat{U}+\delta\hat{U})\hat{\Sigma}(\hat{V}+\delta\hat{V})$ is the true SVD, so that $\hat{U} + \delta\hat{U}$ and $\hat{V} + \delta\hat{V}$ are both orthogonal, where $||E||_2/||A||_2 \leq p(m,n)\epsilon$, $||\delta\hat{U}|| \leq p(m,n)\epsilon$, and $||\delta\hat{V}|| \leq p(m,n)\epsilon$. Here $p(m,n)$ is a modestly growing function of m and n and ϵ is the machine precision. Each computed singular value $\hat{\sigma}_i$ differs from the true σ_i by an amount satisfying the bound

$$|\hat{\sigma}_i - \sigma_i| \leq p(m,n)\epsilon\sigma_1.$$

Thus large singular values (those near σ_1) are computed to high relative accuracy and small ones may not be.

The angular difference between the computed left singular vector $\hat{u}_i$ and the true u_i satisfies the approximate bound

$$\theta(\hat{u}_i, u_i) \lesssim \frac{p(m,n)\epsilon||A||_2}{\text{gap}_i}$$

where

$$\text{gap}_i = \min_{j\neq i}|\sigma_i - \sigma_j|$$

is the *absolute gap* between σ_i and the nearest other singular value. Thus, if σ_i is close to other singular values, its corresponding singular vector u_i may be inaccurate. The same bound applies to the computed right singular vector $\hat{v}_i$ and the true vector v_i. The gaps may be easily obtained from the computed singular values.

Let $\hat{S}$ be the space spanned by a collection of computed left singular vectors $\{\hat{u}_i, i \in I\}$, where I is a subset of the integers from 1 to n. Let S be the corresponding true space. Then

$$\theta(\hat{S}, S) \lesssim \frac{p(m,n)\epsilon||A||_2}{\text{gap}_I}.$$

where

$$\text{gap}_I = \min\{|\sigma_i - \sigma_j| \text{ for } i \in I,\ j \notin I\}$$

is the absolute gap between the singular values in I and the nearest other singular value. Thus, a cluster of close singular values which is far away from any other singular value may have a well determined space $\hat{S}$ even if its individual singular vectors are ill-conditioned. The same bound applies to a set of right singular vectors $\{\hat{v}_i, i \in I\}$.

In the special case of bidiagonal matrices, the singular values and singular vectors may be computed much more accurately [3]. A bidiagonal matrix B has nonzero entries only on the main diagonal and the diagonal immediately above it (or immediately below it). Reduction of a dense matrix to bidiagonal form B can introduce additional errors, so the following bounds for the bidiagonal case do not apply to the dense case.

Using the routines in this chapter, each computed singular value of a bidiagonal matrix is accurate to nearly full relative accuracy, no matter how tiny it is, so that

$$|\hat{\sigma}_i - \sigma_i| \leq p(m,n)\epsilon\sigma_i.$$

The computed left singular vector $\hat{u}_i$ has an angular error at most about

$$\theta(\hat{u}_i, u_i) \lesssim \frac{p(m,n)\epsilon}{\text{relgap}_i}$$

where

$$\text{relgap}_i = \min_{j \neq i} |\sigma_i - \sigma_j|/(\sigma_i + \sigma_j)$$

is the *relative gap* between σ_i and the nearest other singular value. The same bound applies to the right singular vector $\hat{v}_i$ and v_i. Since the relative gap may be much larger than the absolute gap, this error bound may be much smaller than the previous one. The relative gaps may be easily obtained from the computed singular values.

2.11.3 The symmetric eigenproblem

The usual error analysis of the symmetric eigenproblem is as follows [5].

The computed eigendecomposition $\hat{Z}\hat{\Lambda}\hat{Z}^T$ is nearly the exact eigendecomposition of $A + E$, i.e., $A + E = (\hat{Z}+\delta\hat{Z})\hat{\Lambda}(\hat{Z}+\delta\hat{Z})^T$ is the true eigendecomposition so that $\hat{Z}+\delta\hat{Z}$ is orthogonal, where $\|E\|_2/\|A\|_2 \leq p(n)\epsilon$ and $\|\delta\hat{Z}\|_2 \leq p(n)\epsilon$ and $p(n)$ is a modestly growing function of n and ϵ is the machine precision. Each computed eigenvalue $\hat{\lambda}_i$ differs from the true λ_i by an amount satisfying the bound

$$|\hat{\lambda}_i - \lambda_i| \leq p(n)\epsilon\|A\|_2.$$

Thus large eigenvalues (those near $\max_i |\lambda_i| = \|A\|_2$) are computed to high relative accuracy and small ones may not be.

The angular difference between the computed unit eigenvector $\hat{z}_i$ and the true z_i satisfies the approximate bound

$$\theta(\hat{z}_i, z_i) \lesssim \frac{p(n)\epsilon\|A\|_2}{\text{gap}_i}$$

if $p(n)\epsilon$ is small enough, where

$$\text{gap}_i = \min_{j \neq i} |\lambda_i - \lambda_j|$$

is the *absolute gap* between λ_i and the nearest other eigenvalue. Thus, if λ_i is close to other eigenvalues, its corresponding eigenvector z_i may be inaccurate. The gaps may be easily obtained from the computed eigenvalues.

Let $\hat{S}$ be the invariant subspace spanned by a collection of eigenvectors $\{\hat{z}_i, i \in I\}$, where I is a subset of the integers from 1 to n. Let S be the corresponding true subspace. Then

$$\theta(\hat{S}, S) \lesssim \frac{p(n)\epsilon\|A\|_2}{\text{gap}_I}$$

where

$$\text{gap}_I = \min\{|\lambda_i - \lambda_j| \text{ for } i \in I,\ j \notin I\}$$

is the absolute gap between the eigenvalues in I and the nearest other eigenvalue. Thus, a cluster of close eigenvalues which is far away from any other eigenvalue may have a well determined invariant subspace $\hat{S}$ even if its individual eigenvectors are ill-conditioned.

In the special case of a real symmetric tridiagonal matrix T, routines in this chapter can compute the eigenvalues and eigenvectors much more accurately. See Anderson *et al.*[1] for further details.

2.11.4 The generalized symmetric-definite eigenproblem

The three types of problem to be considered are $A - \lambda B$, $AB - \lambda I$ and $BA - \lambda I$. In each case A and B are real symmetric (or complex Hermitian) and B is positive-definite. We consider each case in turn, assuming that routines in this chapter are used to transform the generalized problem to the standard symmetric problem, followed by the solution of the the symmetric problem. In all cases

$$\text{gap}_i = \min_{j \neq i} |\lambda_i - \lambda_j|$$

is the *absolute gap* between λ_i and the nearest other eigenvalue.

(1) $A - \lambda B$. The computed eigenvalues $\hat{\lambda}_i$ can differ from the true eigenvalues λ_i by an amount

$$|\hat{\lambda}_i - \lambda_i| \lesssim p(n)\epsilon \|B^{-1}\|_2 \|A\|_2.$$

The angular difference between the computed eigenvector $\hat{z}_i$ and the true eigenvector z_i is

$$\theta(\hat{z}_i, z_i) \lesssim \frac{p(n)\epsilon \|B^{-1}\|_2 \|A\|_2 (\kappa_2(B))^{1/2}}{\text{gap}_i}.$$

(2) $AB - \lambda I$ or $BA - \lambda I$. The computed eigenvalues $\hat{\lambda}_i$ can differ from the true eigenvalues λ_i by an amount

$$|\hat{\lambda}_i - \lambda_i| \lesssim p(n)\epsilon \|B\|_2 \|A\|_2.$$

The angular difference between the computed eigenvector $\hat{z}_i$ and the true eigenvector z_i is

$$\theta(\hat{z}_i, z_i) \lesssim \frac{q(n)\epsilon \|B\|_2 \|A\|_2 (\kappa_2(B))^{1/2}}{\text{gap}_i}.$$

These error bounds are large when B is ill-conditioned with respect to inversion ($\kappa_2(B)$ is large). It is often the case that the eigenvalues and eigenvectors are much better conditioned than indicated here. One way to get tighter bounds is effective when the diagonal entries of B differ widely in magnitude, as for example with a *graded matrix*.

(1) $A - \lambda B$. Let $D = \text{diag}(b_{11}^{-1/2}, \ldots, b_{nn}^{-1/2})$ be a diagonal matrix. Then replace B by DBD and A by DAD in the above bounds.
(2) $AB - \lambda I$ or $BA - \lambda I$. Let $D = \text{diag}(b_{11}^{-1/2}, \ldots, b_{nn}^{-1/2})$ be a diagonal matrix. Then replace B by DBD and A by $D^{-1}AD^{-1}$ in the above bounds.

Further details can be found in Anderson *et al.* [1].

2.11.5 The nonsymmetric eigenproblem

The nonsymmetric eigenvalue problem is more complicated than the symmetric eigenvalue problem. In this section, we just summarize the bounds. Further details can be found in Anderson *et al.* [1].

We let $\hat{\lambda}_i$ be the ith computed eigenvalue and λ_i the ith true eigenvalue. Let $\hat{v}_i$ be the corresponding computed right eigenvector, and v_i the true right eigenvector (so $Av_i = \lambda_i v_i$). If I is a subset of the integers from 1 to n, we let λ_I denote the average of the selected eigenvalues: $\lambda_I = (\sum_{i \in I} \lambda_i)/(\sum_{i \in I} 1)$, and similarly for $\hat{\lambda}_I$. We also let S_I denote the subspace spanned by $\{v_i, i \in I\}$; it is called a right invariant subspace because if v is any vector in S_I then Av is also in S_I. $\hat{S}_I$ is the corresponding computed subspace.

The algorithms for the nonsymmetric eigenproblem are normwise backward stable: they compute the exact eigenvalues, eigenvectors and invariant subspaces of slightly perturbed matrices $A + E$, where

$||E|| \leq p(n)\epsilon||A||$. Some of the bounds are stated in terms of $||E||_2$ and others in terms of $||E||_F$; one may use $p(n)\epsilon$ for either quantity.

Routines are provided so that, for each $(\hat{\lambda}_i,\ \hat{v}_i)$ pair the two values s_i and sep_i, or for a selected subset I of eigenvalues the values s_I and sep_I can be obtained, for which the error bounds in Table 2 are true for sufficiently small $||E||$, (which is why they are called asymptotic):

Simple eigenvalue	$\lvert\hat{\lambda}_i - \lambda_i\rvert \lesssim ||E||_2/s_i$
Eigenvalue cluster	$\lvert\hat{\lambda}_I - \lambda_I\rvert \lesssim ||E||_2/s_I$
Eigenvector	$\theta(\hat{\vartheta}_i, \vartheta_i) \lesssim ||E||_F/sep_i$
Invariant subspace	$\theta(\hat{S}_I, S_I) \lesssim ||E||_F/sep_I$

Table 2
Asymptotic error bounds for the nonsymmetric eigenproblem

If the problem is ill-conditioned, the asymptotic bounds may only hold for extremely small $||E||$. The global error bounds of Table 3 are guaranteed to hold for all $||E||_F < s \times sep/4$:

Simple eigenvalue	$\lvert\hat{\lambda}_i - \lambda_i\rvert \leq n\|\|E\|\|_2/s_i$	Holds for all E
Eigenvalue cluster	$\lvert\hat{\lambda}_I - \lambda_I\rvert \leq 2\|\|E\|\|_2/s_I$	Requires $\|\|E\|\|_F < s_I \times sep_I/4$
Eigenvector	$\theta(\hat{\vartheta}_i, \vartheta_i) \leq \arctan(2\|\|E\|\|_F/(sep_i - 4\|\|E\|\|_F/s_i))$	Requires $\|\|E\|\|_F < s_i \times sep_i/4$
Invariant subspace	$\theta(\hat{S}_I, S_I) \leq \arctan(2\|\|E\|\|_F/(sep_I - 4\|\|E\|\|_F/s_I))$	Requires $\|\|E\|\|_F < s_I \times sep_I/4$

Table 3
Global error bounds for the nonsymmetric eigenproblem

2.11.6 Balancing and condition

There are two preprocessing steps one may perform on a matrix A in order to make its eigenproblem easier. The first is *permutation*, or reordering the rows and columns to make A more nearly upper triangular (closer to Schur form): $A' = PAP^T$, where P is a permutation matrix. If A' is permutable to upper triangular form (or close to it), then no floating-point operations (or very few) are needed to reduce it to Schur form. The second is *scaling* by a diagonal matrix D to make the rows and columns of A' more nearly equal in norm: $A'' = DA'D^{-1}$. Scaling can make the matrix norm smaller with respect to the eigenvalues, and so possibly reduce the inaccuracy contributed by roundoff (see Chapter, II/11 of [7]). We refer to these two operations as *balancing*.

Permuting has no effect on the condition numbers or their interpretation as described previously. Scaling, however, does change their interpretation and further details can be found in Anderson *et al.* [1].

2.12 Block Algorithms

A number of the routines in this chapter use what is termed a *block algorithm*. This means that at each major step of the algorithm a *block* of rows or columns is updated, and much of the computation is performed by matrix-matrix operations on these blocks. The matrix-matrix operations are performed by calls to the Level 3 BLAS (see Chapter F06), which are the key to achieving high performance on many modern computers. In the case of the QR algorithm for reducing an upper Hessenberg matrix to Schur form, a multishift strategy is used in order to improve performance. See Golub and Van Loan [4] or Anderson *et al.* [1] for more about block algorithms and the multishift strategy.

The performance of a block algorithm varies to some extent with the ***blocksize*** - that is, the number of rows or columns per block. This is a machine-dependent parameter, which is set to a suitable value when the library is implemented on each range of machines. Users of the library do not normally need to be aware of what value is being used. Different block sizes may be used for different routines. Values in the range 16 to 64 are typical.

On more conventional machines there is often no advantage from using a block algorithm, and then the routines use an *unblocked* algorithm (effectively a block size of 1), relying solely on calls to the Level 2 BLAS (see Chapter F06 again).

The only situation in which a user needs some awareness of the block size is when it affects the amount of workspace to be supplied to a particular routine. This is discussed in Section 3.4.3.

3 Recommendations on Choice and Use of Available Routines

Note. Refer to the Users' Note for your implementation to check that a routine is available.

3.1 Available Routines

The tables in the following subsections show the routines which are provided for performing different computations on different types of matrices. Each entry in the table gives the NAG routine name, the LAPACK single precision name, and the LAPACK double precision name (see Section 3.2).

For many computations it is necessary to call two or more routines in sequence some commonly required sequences of routines are indicated below; an asterisk (*) against a routine name means that the sequence of calls is illustrated in the example program for that routine. (But remember that Black Box routines for the same computations may be provided in Chapter F02 or Chapter F04.)

3.1.1 Orthogonal factorizations

Routines are provided for QR factorization (with and without column pivoting), and for LQ factorization (without pivoting only), of a general real or complex rectangular matrix.

The factorization routines do not form the matrix Q explicitly, but represent it as a product of elementary reflectors (see Section 3.3.6). Additional routines are provided to generate all or part of Q explicitly if it is required, or to apply Q in its factored form to another matrix (specifically to compute one of the matrix products QC, Q^TC, CQ or CQ^T with Q^T replaced by Q^H if C and Q are complex.

	Factorize without pivoting	**Factorize with pivoting**	**Generate Matrix Q**	**Apply matrix Q**
QR factorization, real matrices	F08AEF SGEQRF DGEQRF	F08BEF SGEQPF DGEQPF	F08AFF SORGQR DORGQR	F08AGF SORMQR DORMQR
LQ factorization, real matrices	F08AHF SGELQF DGELQF		F08AJF SORGLQ DORGLQ	F08AKF SORMLQ DORMLQ
QR factorization, complex matrices	F08ASF CGEQRF ZGEQRF	F08BSF CGEQPF ZGEQPF	F08ATF CUNGQR ZUNMQR	F08AUF CUNMQR ZUNGQR
LQ factorization, complex matrices	F08AVF CGELQF ZGELQF		F08AWF CUNGLQ ZUNGLQ	F08AXF CUNMQL ZUNMLQ

To solve linear least-squares problems, as described in Section 2.2.1 or Section 2.2.3, routines based on the QR factorization can be used:

real data, full-rank problem	F08AEF*, F08AGF, F06YJF
complex data, full-rank problem	F08ASF*, F08AUF, F06ZJF
real data, rank-deficient problem	F08BEF*, F08AGF, F06YJF
complex data, rank-deficient problem	F08BSF*, F08AUF, F06ZJF

To find the minimum norm solution of under-determined systems of linear equations, as described in Section 2.2.2, routines based on the LQ factorization can be used:

real data, full-rank problem	F08AHF*, F06YJF, F08AKF
complex data, full-rank problem	F08AVF*, F06ZJF, F08AXF

3.1.2 Singular value problems

Routines are provided to reduce a general real or complex rectangular matrix A to real bidiagonal form B by an orthogonal transformation $A = QBP^T$ (or by a unitary transformation $A = QBP^H$ if A is complex). Different routines allow a full matrix A to be stored conventionally (see Section 3.3.1), or a band matrix to use band storage (see Section 3.3.3).

The routines for reducing full matrices do not form the matrix Q or P explicitly; additional routines are provided to generate all or part of them, or to apply them to another matrix, as with the routines for orthogonal factorizations. Explicit generation of Q or P is required before using the bidiagonal QR algorithm to compute left or right singular vectors of A.

The routines for reducing band matrices have options to generate Q or P if required.

Further routines are provided to compute all or part of the singular value decomposition of a real bidiagonal matrix; the same routines can be used to compute the singular value decomposition of a real or complex matrix that has been reduced to bidiagonal form.

	Reduce to bidiagonal form	**Generate matrix Q or P^T**	**Apply matrix Q or P**	**Reduce band matrix to bidiagonal form**	**SVD of bidiagonal form (QR algorithm)**
real matrices	F08KEF SGEBRD DGEBRD	F08KFF SORGBR DORGBR	F08KGF SORMBR DORMBR	F08LEF SGBBRD DGBBRD	F08MEF SBDSQR DBDSQR
complex matrices	F08KSF CGEBRD ZGEBRD	F08KTF CUNGBR ZUNGBR	F08KUF CUNMBR ZUNMBR	F08LSF CGBBRD ZGBBRD	F08MSF CBDSQR ZBDSQR

To compute the singular values and vectors of a rectangular matrix, as described in Section 2.3, use the following sequence of calls:

Rectangular matrix (standard storage)

real matrix, singular values and vectors `F08KEF, F08KFF*, F08MEF`
complex matrix, singular values and vectors `F08KSF, F08KTF*, F08MSF`

Rectangular matrix (banded)

real matrix, singular values and vectors `F08LEF, F08MEF`
complex matrix, singular values and vectors `F08LSF, F08MSF`

To use the singular value decomposition to solve a linear least-squares problem, as described in Section 2.4, the following routines are required:

real data: `F08KEF, F08KGF, F08KFF, F08MEF, F06YAF`
complex data: `F08KSF, F08KUF, F08KTF, F08MSF, F06ZAF`

3.1.3 Symmetric eigenvalue problems

Routines are provided to reduce a real symmetric or complex Hermitian matrix A to real tridiagonal form T by an orthogonal similarity transformation $A = QTQ^T$ (or by a unitary transformation $A = QTQ^H$ if A is complex). Different routines allow a full matrix A to be stored conventionally (see Section 3.3.1) or in packed storage (see Section 3.3.2); or a band matrix to use band storage (see Section 3.3.3).

The routines for reducing full matrices do not form the matrix Q explicitly; additional routines are provided to generate Q, or to apply it to another matrix, as with the routines for orthogonal factorizations. Explicit generation of Q is required before using the QR algorithm to find all the eigenvectors of A; application of Q to another matrix is required after eigenvectors of T have been found by inverse iteration, in order to transform them to eigenvectors of A.

The routines for reducing band matrices have an option to generate Q if required.

	Reduce to tridiagonal form	**Generate matrix Q**	**Apply matrix Q**
real symmetric matrices	F08FEF SSYTRD DSYTRD	F08FFF SORGTR DORGTR	F08FGF SORMTR DORMTR
real symmetric matrices (packed storage)	F08GEF SSPTRD DSPTRD	F08GFF SOPGTR DOPGTR	F08GGF SOPMTR DOPMTR
real symmetric band matrices	F08HEF SSBTRD DSBTRD		
complex Hermitian matrices	F08FSF CHETRD ZHETRD	F08FTF CUNGTR ZUNGTR	F08FUF CUNMTR ZUNMTR
complex Hermitian matrices (packed storage)	F08GSF CHPTRD ZHPTRD	F08GTF CUPGTR ZUPGTR	F08GUF CUPMTR ZUPMTR
complex Hermitian band matrices	F08HSF CHBTRD ZHBTRD		

A variety of routines are provided to compute eigenvalues and eigenvectors of the real symmetric tridiagonal matrix T, some computing all eigenvalues and eigenvectors, some computing selected eigenvalues and eigenvectors. The same routines can be used to compute eigenvalues and eigenvectors of a real symmetric or complex Hermitian matrix which has been reduced to tridiagonal form.

Eigenvalues and eigenvectors of real symmetric tridiagonal matrices:

The original (non-reduced) matrix is Real or Complex Hermitian

all eigenvalues (root-free QR algorithm)	F08JFF
all eigenvalues (root-free QR algorithm called by divide and conquer)	F08JCF
selected eigenvalues (bisection)	F08JJF

The original (non-reduced) matrix is Real

all eigenvalues and eigenvectors (QR algorithm)	F08JEF
all eigenvalues and eigenvectors (divide and conquer)	F08JCF
all eigenvalues and eigenvectors (positive-definite case)	F08JGF
selected eigenvectors (inverse iteration)	F08JKF

The original (non-reduced) matrix is Complex Hermitian

all eigenvalues and eigenvectors (QR algorithm)	F08JSF
all eigenvalues and eigenvectors (positive-definite case)	F08JUF
selected eigenvectors (inverse iteration)	F08JXF

The following sequences of calls may be used to compute various combinations of eigenvalues and eigenvectors, as described in Section 2.5.

Sequences for computing eigenvalues and eigenvectors

Real Symmetric matrix (standard storage)

all eigenvalues and eigenvectors (using divide and conquer)	F08FCF
all eigenvalues and eigenvectors (using QR algorithm)	F08FEF, F08FFF*, F08JEF
selected eigenvalues and eigenvectors (bisection and inverse iteration)	F08FEF, F08JJF, F08JKF, F08FGF*

Real Symmetric matrix (packed storage)

all eigenvalues and eigenvectors (using divide and conquer)	F08GCF
all eigenvalues and eigenvectors (using QR algorithm)	F08GEF, F08GFF*, F08JEF
selected eigenvalues and eigenvectors (bisection and inverse iteration)	F08GEF, F08JJF, F08JKF, F08GGF*

Real Symmetric banded matrix

all eigenvalues and eigenvectors (using divide and conquer)	F08HCF
all eigenvalues and eigenvectors (using QR algorithm)	F08HEF*, F08JEF

Complex Hermitian matrix (standard storage)

all eigenvalues and eigenvectors (using divide and conquer)	F08FQF
all eigenvalues and eigenvectors (using QR algorithm)	F08FSF, F08FTF*, F08JSF
selected eigenvalues and eigenvectors (bisection and inverse iteration)	F08FSF, F08JJF, F08JXF, F08FUF*

Complex Hermitian matrix (packed storage)

all eigenvalues and eigenvectors (using divide and conquer)	F08GQF
all eigenvalues and eigenvectors (using QR algorithm)	F08GSF, F08GTF*, F08JSF
selected eigenvalues and eigenvectors (bisection and inverse iteration)	F08GSF, F08JJF, F08JXF, F08GUF*

Complex Hermitian banded matrix

all eigenvalues and eigenvectors (using divide and conquer)	F08HQF
all eigenvalues and eigenvectors (using QR algorithm)	F08HSF*, F08JSF

3.1.4 Generalized symmetric-definite eigenvalue problems

Routines are provided for reducing each of the problems $Ax = \lambda Bx$, $ABx = \lambda x$ or $BAx = \lambda x$ to an equivalent standard eigenvalue problem $Cy = \lambda y$. Different routines allow the matrices to be stored either conventionally or in packed storage. The positive-definite matrix B must first be factorized using a routine from Chapter F07. There is also a routine which reduces the problem $Ax = \lambda Bx$ where A and B are banded, to an equivalent banded standard eigenvalue problem; this uses a split Cholesky factorization for which a routine in Chapter F08 is provided.

	Reduce to standard problem	**Reduce to standard problem (packed storage)**	**Reduce to standard problem (band matrices)**
real symmetric matrices	F08SEF SSYGST DSYGST	F08TEF SSPGST DSPGST	F08UEF SSBGST DSBGST
complex Hermitian matrices	F08SSF CHEGST ZHEGST	F08TSF CHPGST ZHPGST	F08USF CHBGST ZHBGST

The equivalent standard problem can then be solved using the routines discussed in Section 3.1.3. For example, to compute all the eigenvalues, the following routines must be called:

real symmetric-definite problem	F07FDF, F08SEF*, F08FEF, F08JFF
real symmetric-definite problem, packed storage	F07GDF, F08TEF*, F08GEF, F08JFF
real symmetric-definite banded problem	F08UFF*, F08UEF*, F08HEF, F08JFF
complex Hermitian-definite problem	F07FRF, F08SSF*, F08FSF, F08JFF
complex Hermitian-definite problem, packed storage	F07GRF, F08TSF*, F08GSF, F08JFF
complex Hermitian-definite banded problem	F08UTF*, F08USF*, F08HSF, F08JFF

If eigenvectors are computed, the eigenvectors of the equivalent standard problem must be transformed back to those of the original generalized problem, as indicated in Section 2.6; routines from Chapter F06 may be used for this.

3.1.5 Nonsymmetric eigenvalue problems

Routines are provided to reduce a general real or complex matrix A to upper Hessenberg form H by an orthogonal similarity transformation $A = QHQ^T$ (or by a unitary transformation $A = QHQ^H$ if A is complex).

These routines do not form the matrix Q explicitly; additional routines are provided to generate Q, or to apply it to another matrix, as with the routines for orthogonal factorizations. Explicit generation of Q is required before using the QR algorithm on H to compute the Schur vectors; application of Q to another matrix is needed after eigenvectors of H have been computed by inverse iteration, in order to transform them to eigenvectors of A.

Routines are also provided to balance the matrix before reducing it to Hessenberg form, as described in Section 2.11.6. Companion routines are required to transform Schur vectors or eigenvectors of the balanced matrix to those of the original matrix.

	Reduce to Hessenberg form	**Generate matrix Q**	**Apply matrix Q**	**Balance**	**Backtransform vectors after balancing**
real matrices	F08NEF SGEHRD DGEHRD	F08NFF SORGHR DORGHR	F08NGF SORMHR DORMHR	F08NHF SGEBAL DGEBAL	F08NJF SGEBAK DGEBAK
complex matrices	F08NSF CGEHRD ZGEHRD	F08NTF CUNGHR ZUNGHR	F08NUF CUNMHR ZUNMHR	F08NVF CGEBAL ZGEBAL	F08NWF CGEBAK ZGEBAK

Routines are provided to compute the eigenvalues and all or part of the Schur factorization of an upper Hessenberg matrix. Eigenvectors may be computed either from the upper Hessenberg form by inverse iteration, or from the Schur form by back-substitution; these approaches are equally satisfactory for computing individual eigenvectors, but the latter may provide a more accurate basis for a subspace spanned by several eigenvectors.

Additional routines estimate the sensitivities of computed eigenvalues and eigenvectors, as discussed in Section 2.11.5.

	Eigenvalues and Schur factorization (QR algorithm)	**Eigenvectors from Hessenberg form (inverse iteration)**	**Eigenvectors from Schur factorization**	**Sensitivities of eigenvalues and eigenvectors**
real matrices	F08PEF SHSEQR DHSEQR	F08PKF SHSEIN DHSEIN	F08QKF STREVC DTREVC	F08QLF STRSNA DTRSNA
complex matrices	F08PSF CHSEQR ZHSEQR	F08PXF CHSEIN ZHSEIN	F08QXF CTREVC ZTREVC	F08QYF CTRSNA ZTRSNA

Finally routines are provided for re-ordering the Schur factorization, so that eigenvalues appear in any desired order on the diagonal of the Schur form. The routines F08QFF and F08QTF simply swap two diagonal elements or blocks, and may need to be called repeatedly to achieve a desired order. The routines F08QGF and F08QUF perform the whole re-ordering process for the important special case where a specified cluster of eigenvalues is to appear at the top of the Schur form; if the Schur vectors are re-ordered at the same time, they yield an orthonormal basis of the invariant subspace corresponding to the specified cluster of eigenvalues. These routines can also compute the sensitivities of the cluster of eigenvalues and the invariant subspace.

	Reorder Schur factorization	Reorder Schur factorization, find basis of invariant subspace and estimate sensitivities
real matrices	F08QFF STREXC DTREXC	F08QGF STRSEN DTRSEN
complex matrices	F08QTF CTREXC ZTREXC	F08QUF CTRSEN ZTRSEN

The following sequences of calls may be used to compute various combinations of eigenvalues, Schur vectors and eigenvectors, as described in Section 2.9:

real matrix, all eigenvalues and Schur factorization	F08NEF, F08NFF*, F08PEF
real matrix, all eigenvalues and selected eigenvectors	F08NEF, F08PEF, F08PKF, F08NGF*
real matrix, all eigenvalues and eigenvectors (with balancing)	F08NHF*, F08NEF, F08NFF, F08PEF, F08PKF, F08NJF
complex matrix, all eigenvalues and Schur factorization	F08NSF, F08NTF*, F08PSF
complex matrix, all eigenvalues and selected eigenvectors	F08NSF, F08PSF, F08PXF, F08NUF*
complex matrix, all eigenvalues and eigenvectors (with balancing)	F08NVF*, F08NSF, F08NTF, F08PSF, F08PXF, F08NWF

3.1.6 Sylvester's equation

Routines are provided to solve the real or complex Sylvester equation $AX \pm XB = C$, where A and B are upper quasi-triangular if real, or upper triangular if complex. To solve the general form of Sylvester's equation in which A and B are general square matrices, A and B must be reduced to upper (quasi-) triangular form by the Schur factorization,using routines described in Section 3.1.5. For more details, see the documents for the routines listed below.

	solve Sylvester's equation
real matrices	F08QHF STRSYL DTRSYL
complex matrices	F08QVF CTRSYL ZTRSYL

3.2 NAG Names and LAPACK Names

As well as the NAG routine name (beginning F08-), the tables in Section 3.1 show the LAPACK routine names in both single and double precision.

The routines may be called either by their NAG names or by their LAPACK names. When using a single precision implementation of the NAG Library, the single precision form of the LAPACK name must be used (beginning with S- or C-); when using a double precision implementation of the NAG Library, the double precision form of the LAPACK name must be used (beginning with D- or Z-).

References to F08 routines in the Manual normally include the LAPACK single and double precision names, in that order – for example F08AEF (SGEQRF/DGEQRF). The LAPACK routine names follow a simple scheme (which is similar to that used for the BLAS in Chapter F06). Each name has the structure **XYYZZZ**, where the components have the following meanings:

– the initial letter **X** indicates the data type (real or complex) and precision:

S – real, single precision (in Fortran 77, **REAL**)

D – real, double precision (in Fortran 77, **DOUBLE PRECISION**)

C – complex, single precision (in Fortran 77, `COMPLEX`)

Z – complex, double precision (in Fortran 77, `COMPLEX*16 or DOUBLE COMPLEX`)

– the 2nd and 3rd letters **YY** indicate the type of the matrix A (and in some cases its storage scheme):

BD – bidiagonal
GB – general band
GE – general
HS – upper Hessenberg
OP – (real) orthogonal (packed storage)
UP – (complex) unitary (packed storage)
OR – (real) orthogonal
UN – (complex) unitary
PT – symmetric or Hermitian positive-definite tridiagonal
SB – (real) symmetric band
HB – (complex) Hermitian band
SP – symmetric (packed storage)
HP – Hermitian (packed storage)
ST – (real) symmetric tridiagonal
SY – symmetric
HE – Hermitian
TR – triangular (or quasi-triangular)

– the last 3 letters **ZZZ** indicate the computation performed. For example, QRF is a QR factorization.

Thus the routine SGEQRF performs a QR factorization of a real general matrix in a single precision implementation of the Library; the corresponding routine in a double precision implementation is DGEQRF.

Some sections of the routine documents – Section 2 (Specification) and Section 9.1 (Example program) – print the LAPACK name in ***bolditalics***, according to the NAG convention of using bold italics for precision-dependent terms – for example, ***sgeqrf***, which should be interpreted as either SGEQRF (in single precision) or DGEQRF (in double precision).

3.3 Matrix Storage Schemes

In this chapter the following storage schemes are used for matrices:

- conventional storage in a two-dimensional array;
- packed storage for symmetric or Hermitian matrices;
- packed storage for orthogonal or unitary matrices;
- band storage for general, symmetric or Hermitian band matrices;
- storage of bidiagonal, symmetric or Hermitian tridiagonal matrices in two one-dimensional arrays.

These storage schemes are compatible with those used in Chapter F06 and Chapter F07, but different schemes for packed, band and tridiagonal storage are used in a few older routines in Chapter F01, Chapter F02, Chapter F03 and Chapter F04.

In the examples below, $*$ indicates an array element which need not be set and is not referenced by the routines. The examples illustrate only the relevant leading rows and columns of the arrays; array arguments may of course have additional rows or columns, according to the usual rules for passing array arguments in Fortran 77.

3.3.1 Conventional storage

The default scheme for storing matrices is the obvious one: a matrix A is stored in a two-dimensional array A, with matrix element a_{ij} stored in array element $A(i,j)$.

If a matrix is *triangular* (upper or lower, as specified by the argument UPLO when present), only the elements of the relevant triangle are stored; the remaining elements of the array need not be set. Such elements are indicated by $*$ in the examples below. For example, when $n = 4$:

UPLO	Triangular matrix A	Storage in array A
'U'	$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ & a_{22} & a_{23} & a_{24} \\ & & a_{33} & a_{34} \\ & & & a_{44} \end{pmatrix}$	$\begin{matrix} a_{11} & a_{12} & a_{13} & a_{14} \\ * & a_{22} & a_{23} & a_{24} \\ * & * & a_{33} & a_{34} \\ * & * & * & a_{44} \end{matrix}$
'L'	$\begin{pmatrix} a_{11} & & & \\ a_{21} & a_{22} & & \\ a_{31} & a_{32} & a_{33} & \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$	$\begin{matrix} a_{11} & * & * & * \\ a_{21} & a_{22} & * & * \\ a_{31} & a_{32} & a_{33} & * \\ a_{41} & a_{42} & a_{43} & a_{44} \end{matrix}$

Similarly, if the matrix is upper Hessenberg, or if the matrix is quasi-upper triangular, elements below the first subdiagonal need not be set.

Routines that handle *symmetric* or *Hermitian* matrices allow for either the upper or lower triangle of the matrix (as specified by UPLO) to be stored in the corresponding elements of the array; the remaining elements of the array need not be set. For example, when $n = 4$:

UPLO	Hermitian matrix A	Storage in array A
'U'	$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ \bar{a}_{12} & a_{22} & a_{23} & a_{24} \\ \bar{a}_{13} & \bar{a}_{23} & a_{33} & a_{34} \\ \bar{a}_{14} & \bar{a}_{24} & \bar{a}_{34} & a_{44} \end{pmatrix}$	$\begin{matrix} a_{11} & a_{12} & a_{13} & a_{14} \\ * & a_{22} & a_{23} & a_{24} \\ * & * & a_{33} & a_{34} \\ * & * & * & a_{44} \end{matrix}$
'L'	$\begin{pmatrix} a_{11} & \bar{a}_{21} & \bar{a}_{31} & \bar{a}_{41} \\ a_{21} & a_{22} & \bar{a}_{32} & \bar{a}_{42} \\ a_{31} & a_{32} & a_{33} & \bar{a}_{43} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$	$\begin{matrix} a_{11} & * & * & * \\ a_{21} & a_{22} & * & * \\ a_{31} & a_{32} & a_{33} & * \\ a_{41} & a_{42} & a_{43} & a_{44} \end{matrix}$

3.3.2 Packed storage

Symmetric and Hermitian matrices may be stored more compactly, if the relevant triangle (again as specified by UPLO) is packed *by columns* in a one-dimensional array. In Chapter F07 and Chapter F08, arrays that hold matrices in packed storage, have argument names ending in 'P'. So:

if UPLO = 'U', a_{ij} is stored in $AP(i + j(j-1)/2)$ for $i \leq j$;

if UPLO = 'L', a_{ij} is stored in $AP(i + (2n-j)(j-1)/2)$ for $j \leq i$.

For example:

UPLO	Triangle of matrix A	Packed storage in array AP
'U'	$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ & a_{22} & a_{23} & a_{24} \\ & & a_{33} & a_{34} \\ & & & a_{44} \end{pmatrix}$	a_{11} $\underbrace{a_{12}a_{22}}$ $\underbrace{a_{13}a_{23}a_{33}}$ $\underbrace{a_{14}a_{24}a_{34}a_{44}}$
'L'	$\begin{pmatrix} a_{11} & & & \\ a_{21} & a_{22} & & \\ a_{31} & a_{32} & a_{33} & \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$	$\underbrace{a_{11}a_{21}a_{31}a_{41}}$ $\underbrace{a_{22}a_{32}a_{42}}$ $\underbrace{a_{33}a_{43}}$ a_{44}

Note that for symmetric matrices, packing the upper triangle by columns is equivalent to packing the lower triangle by rows; packing the lower triangle by columns is equivalent to packing the upper triangle by rows. For Hermitian matrices, packing the upper triangle by columns is equivalent to packing the conjugate of the lower triangle by rows; packing the lower triangle by columns is equivalent to packing the conjugate of the upper triangle by rows.

3.3.3 Band storage

A general m by n band matrix with k_l subdiagonals and k_u superdiagonals may be stored compactly in a two-dimensional array with $k_l + k_u + 1$ rows and n columns. Columns of the matrix are stored in corresponding columns of the array, and diagonals of the matrix are stored in rows of the array. This storage scheme should be used in practice only if $k_l, k_u \ll n$, although routines in Chapter F07 and Chapter F08 work correctly for all values of k_l and k_u. In Chapter F07 and Chapter F08, arrays that hold matrices in band storage have argument names ending in 'B'. So:

a_{ij} is stored in $\mathrm{AB}(k_u + 1 + i - j, j)$ for $\max(1, j - k_u) \le i \le \min(m, j + k_l)$.

For example, when $m = 6$, $n = 5$, $k_l = 2$ and $k_u = 1$:

general band matrix A	Band storage in array AB
$\begin{pmatrix} a_{11} & a_{12} & & & \\ a_{21} & a_{22} & a_{23} & & \\ a_{31} & a_{32} & a_{33} & a_{34} & \\ & a_{42} & a_{43} & a_{44} & a_{45} \\ & & a_{53} & a_{54} & a_{55} \\ & & & a_{64} & a_{65} \end{pmatrix}$	$\begin{pmatrix} * & a_{12} & a_{23} & a_{34} & a_{45} \\ a_{11} & a_{22} & a_{33} & a_{44} & a_{55} \\ a_{21} & a_{32} & a_{43} & a_{54} & a_{65} \\ a_{31} & a_{42} & a_{53} & a_{64} & * \end{pmatrix}$

A symmetric or Hermitian band matrix with k subdiagonals and superdiagonals may be stored more compactly in a two-dimensional array with $k + 1$ rows and n columns. Only the upper or lower triangle (as specified by UPLO) need to be stored. So:

if UPLO = 'U', a_{ij} is stored in $\mathrm{AB}(k + 1 + i - j, j)$ for $\max(1, j - k) \le i \le j$;

if UPLO = 'L', a_{ij} is stored in $\mathrm{AB}(1 + i - j, j)$ for $j \le i \le \min(n, j + k)$.

For example, when $n = 5$ and $k = 2$:

UPLO	Hermitian band matrix A	Band storage in array AB
'U'	$\begin{pmatrix} a_{11} & a_{12} & a_{13} & & \\ \bar{a}_{12} & a_{22} & a_{23} & a_{24} & \\ \bar{a}_{13} & \bar{a}_{23} & a_{33} & a_{34} & a_{35} \\ & \bar{a}_{24} & \bar{a}_{34} & a_{44} & a_{45} \\ & & \bar{a}_{35} & \bar{a}_{45} & a_{55} \end{pmatrix}$	$\begin{pmatrix} * & * & a_{13} & a_{24} & a_{35} \\ * & a_{12} & a_{23} & a_{34} & a_{45} \\ a_{11} & a_{22} & a_{33} & a_{44} & a_{55} \end{pmatrix}$
'L'	$\begin{pmatrix} a_{11} & \bar{a}_{21} & \bar{a}_{31} & & \\ a_{21} & a_{22} & \bar{a}_{32} & \bar{a}_{42} & \\ a_{31} & a_{32} & a_{33} & \bar{a}_{43} & \bar{a}_{53} \\ & a_{42} & a_{43} & a_{44} & \bar{a}_{54} \\ & & a_{53} & a_{54} & a_{55} \end{pmatrix}$	$\begin{pmatrix} a_{11} & a_{22} & a_{33} & a_{44} & a_{55} \\ a_{21} & a_{32} & a_{43} & a_{54} & * \\ a_{31} & a_{42} & a_{53} & * & * \end{pmatrix}$

3.3.4 Tridiagonal and bidiagonal matrices

A symmetric tridiagonal or bidiagonal matrix is stored in two one-dimensional arrays, one of length n containing the diagonal elements, and one of length $n-1$ containing the off-diagonal elements. (Older routines in Chapter F02 store the off-diagonal elements in elements $2:n$ of a vector of length n.)

3.3.5 Real diagonal elements of complex matrices

Complex Hermitian matrices have diagonal matrices that are by definition purely real. In addition, some complex triangular matrices computed by F08 routines are defined by the algorithm to have real diagonal elements – in QR factorization, for example.

If such matrices are supplied as input to F08 routines, the imaginary parts of the diagonal elements are not referenced, but are assumed to be zero. If such matrices are returned as output by F08 routines, the computed imaginary parts are explicitly set to zero.

3.3.6 Representation of orthogonal or unitary matrices

A real orthogonal or complex unitary matrix (usually denoted Q) is often represented in the NAG Library as a product of *elementary reflectors* – also referred to as *elementary Householder matrices* (usually denoted H_i). For example,

$$Q = H_1 H_2 \ldots H_k.$$

Most users need not be aware of the details, because routines are provided to work with this representation, either to generate all or part of Q explicitly, or to multiply a given matrix by Q or Q^T (Q^H in the complex case) without forming Q explicitly.

Nevertheless, the following further details may occasionally be useful.

An elementary reflector (or elementary Householder matrix) H of order n is a unitary matrix of the form

$$H = I - \tau v v^H \tag{4}$$

where τ is a scalar, and v is an n element vector, with $|\tau|^2 \|v\|_2^2 = 2 \times \mathrm{Re}(\tau)$; v is often referred to as the *Householder vector*. Often v has several leading or trailing zero elements, but for the purpose of this discussion assume that H has no such special structure.

There is some redundancy in the representation (4), which can be removed in various ways. The representation used in Chapter F08 and in LAPACK (which differs from those used in some of the routines in Chapter F01, Chapter F02, Chapter F04 and Chapter F06) sets $v_1 = 1$; hence v_1 need not be stored. In real arithmetic, $1 \le \tau \le 2$, except that $\tau = 0$ implies $H = I$.

In complex arithmetic, τ may be complex, and satisfies $1 \le \mathrm{Re}(\tau) \le 2$ and $|\tau - 1| \le 1$. Thus a complex H is not Hermitian (as it is in other representations), but it is unitary, which is the important property. The

advantage of allowing τ to be complex is that, given an arbitrary complex vector x, H can be computed so that

$$H^H x = \beta(1, 0, \ldots, 0)^T$$

with *real* β. This is useful, for example, when reducing a complex Hermitian matrix to real symmetric tridiagonal form, or a complex rectangular matrix to real bidiagonal form.

3.4 Parameter Conventions

3.4.1 Option parameters

Most routines in this chapter have one or more option parameters, of type CHARACTER. The descriptions in Section 5 of the routine documents refer only to upper case values (for example 'U' or 'L'); however in every case, the corresponding lower case characters may be supplied (with the same meaning). Any other value is illegal.

A longer character string can be passed as the actual parameter, making the calling program more readable, but only the first character is significant. (This is a feature of Fortran 77.) For example:

```
CALL SSYTRD ('Upper', . . . )
```

3.4.2 Problem dimensions

It is permissible for the problem dimensions (for example, M or N) to be passed as zero, in which case the computation (or part of it) is skipped. Negative dimensions are regarded as an error.

3.4.3 Length of work arrays

A number of routines implementing block algorithms require workspace sufficient to hold one block of rows or columns of the matrix if they are to achieve optimum levels of performance - for example, workspace of size $n \times nb$, where nb is the optimum block size. In such cases, the actual declared length of the work array must be passed as a separate argument LWORK, which immediately follows WORK in the argument-list.

The routine will still perform correctly when less workspace is provided: it simply uses the largest block size allowed by the amount of workspace supplied, as long as this is likely to give better performance than the unblocked algorithm. On exit, WORK(1) contains the minimum value of LWORK which would allow the routine to use the optimum block size; this value of LWORK can be used for subsequent runs.

If LWORK indicates that there is insufficient workspace to perform the unblocked algorithm, this is regarded as an illegal value of LWORK, and is treated like any other illegal parameter value (see Section 3.4.4).

If you are in doubt how much workspace to supply and are concerned to achieve optimum performance, supply a generous amount (assume a block size of 64, say), and then examine the value of WORK(1) on exit.

3.4.4 Error-handling and the diagnostic parameter INFO

Routines in this chapter do not use the usual NAG Library error-handling mechanism, involving the parameter IFAIL. Instead they have a diagnostic parameter INFO. (Thus they preserve complete compatibility with the LAPACK specification.)

Whereas IFAIL is an *Input/Output* parameter and must be set before calling a routine, INFO is purely an *Output* parameter and need not be set before entry.

INFO indicates the success or failure of the computation, as follows:

INFO = 0: successful termination

INFO < 0: failure in the course of computation, control returned to the calling program

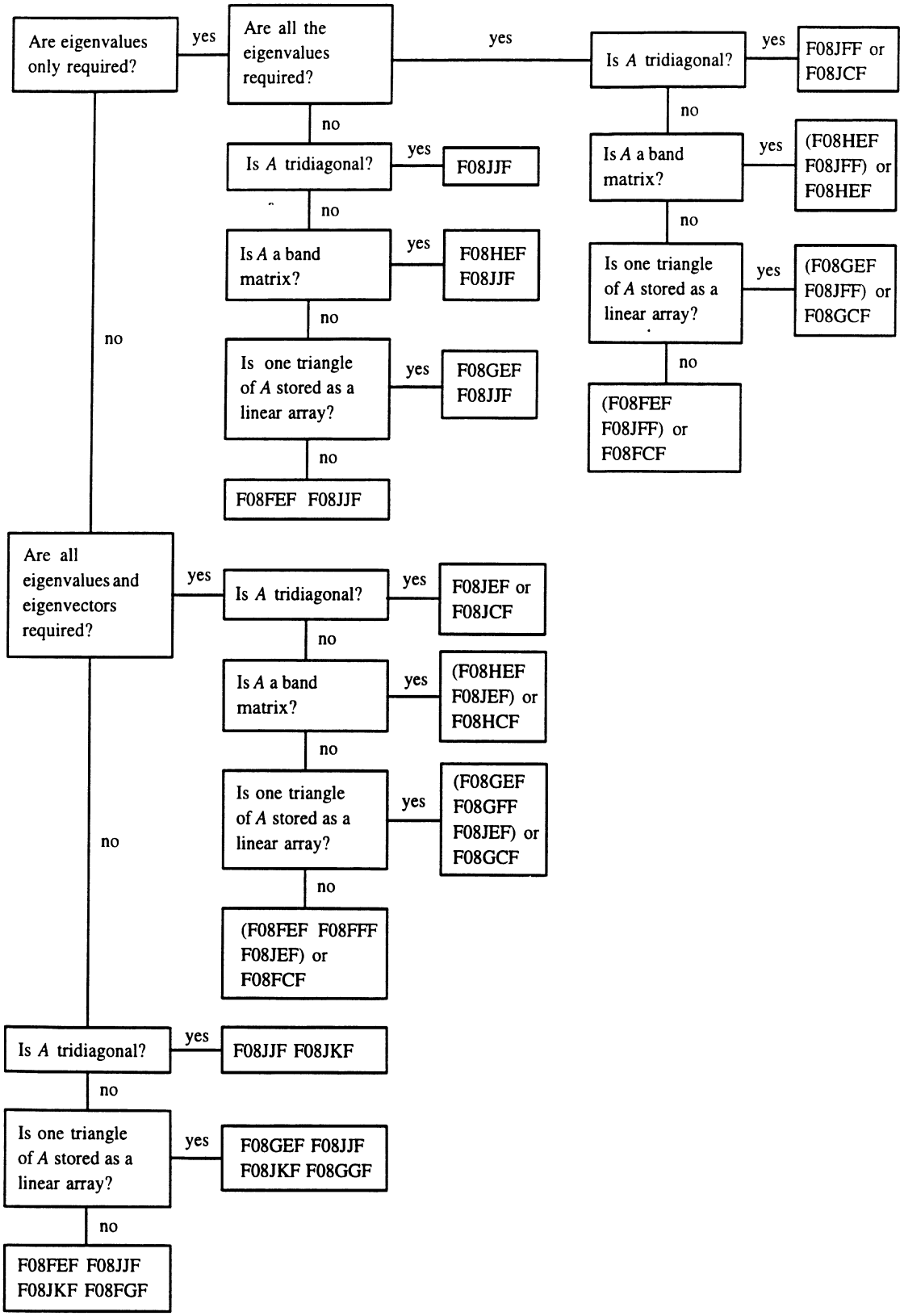
If the routine document specifies that the routine may terminate with INFO < 0, then it is **essential to test INFO on exit** from the routine. (This corresponds to a *soft failure* in terms of the usual NAG error-handling terminology.) No error message is output.

All routines check that input parameters such as N or LDA or option parameters of type CHARACTER have permitted values. If an illegal value of the ith parameter is detected, INFO is set to $-i$, a message is output, and execution of the program is terminated. (This corresponds to a *hard failure* in the usual NAG terminology.)

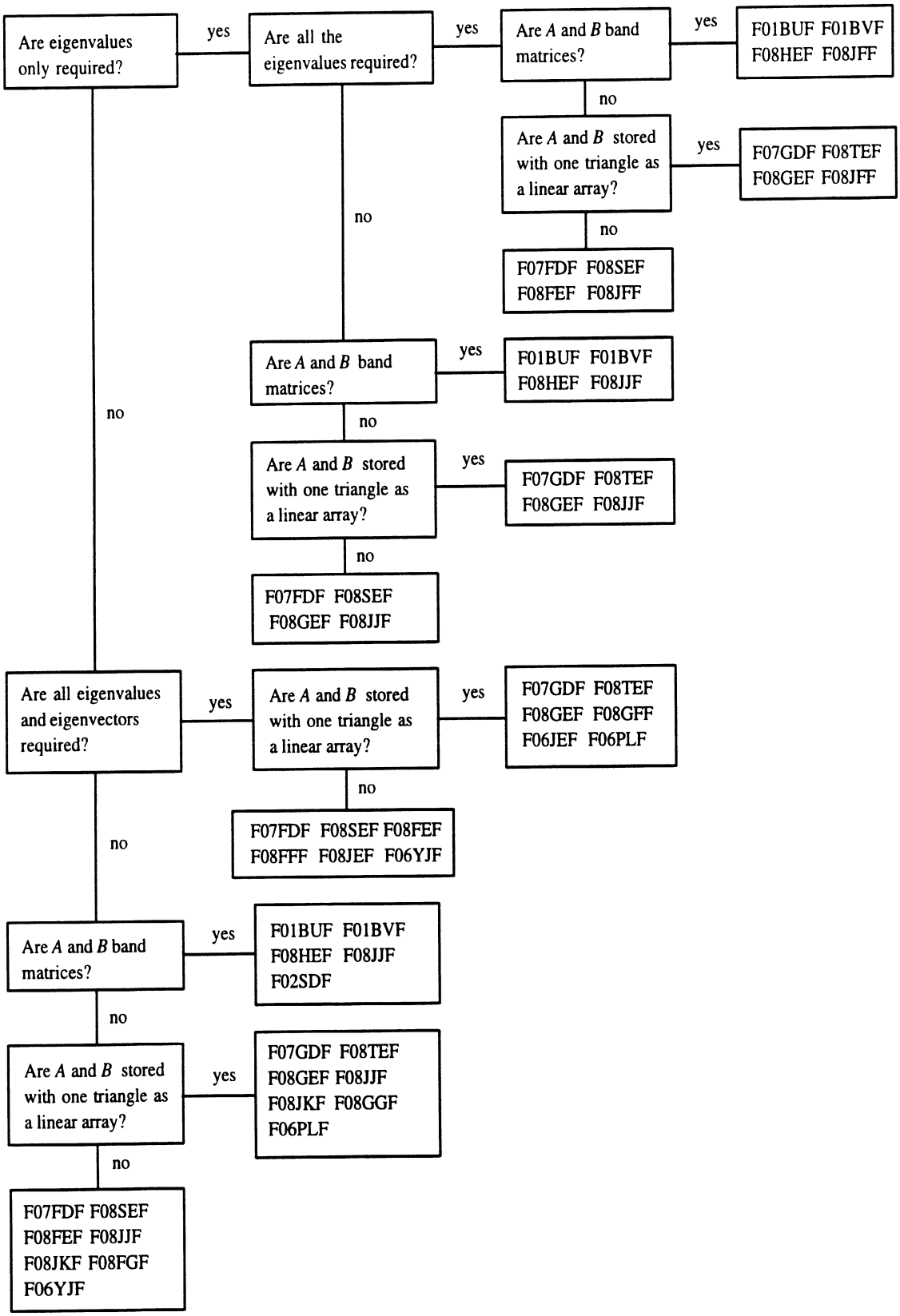
4 Decision Trees

4.1 General purpose routines (eigenvalues and eigenvectors)

Tree 1: Real Symmetric Matrices

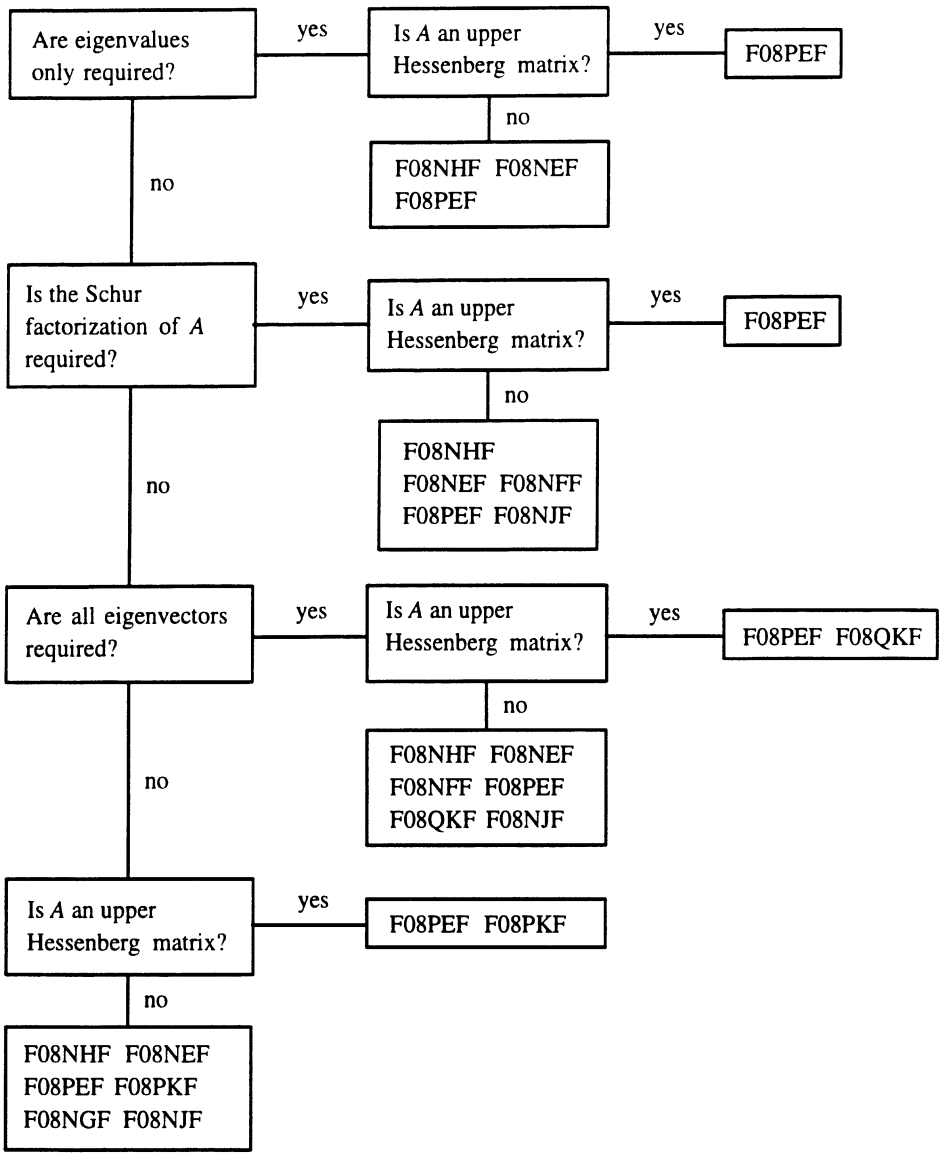


Tree 2: Real Generalized Symmetric-definite Eigenvalue Problems

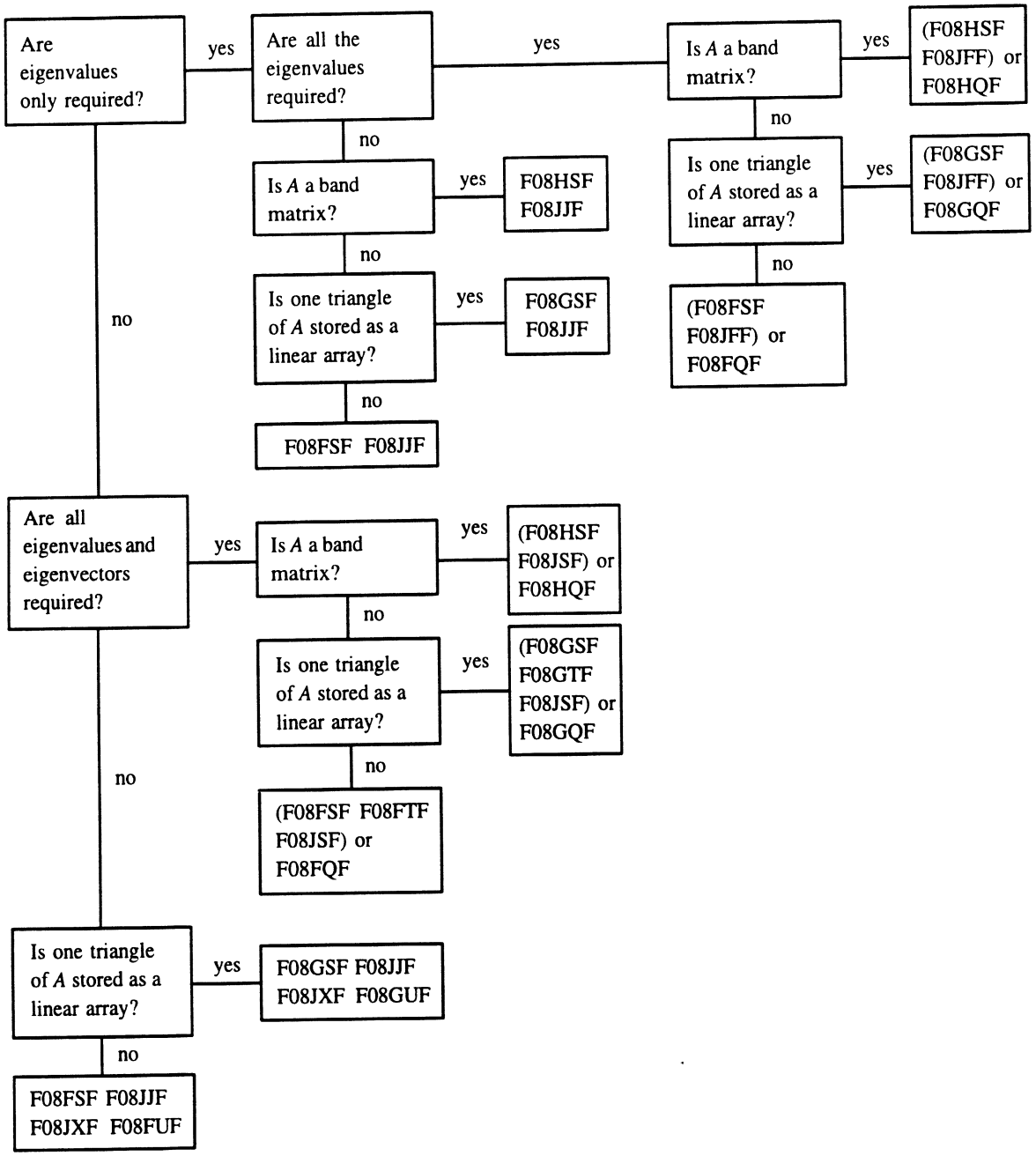


Note: the routines for band matrices only handle the problem $Ax = \lambda Bx$; the other routines handle all three types of problems ($Ax = \lambda Bx$, $ABx = \lambda x$ or $BAx = \lambda x$) except that, if the problem is $BAx = \lambda x$ and eigenvectors are required, F06PHF must be used instead of F06PLF, and F06YFF instead of F06YJF.

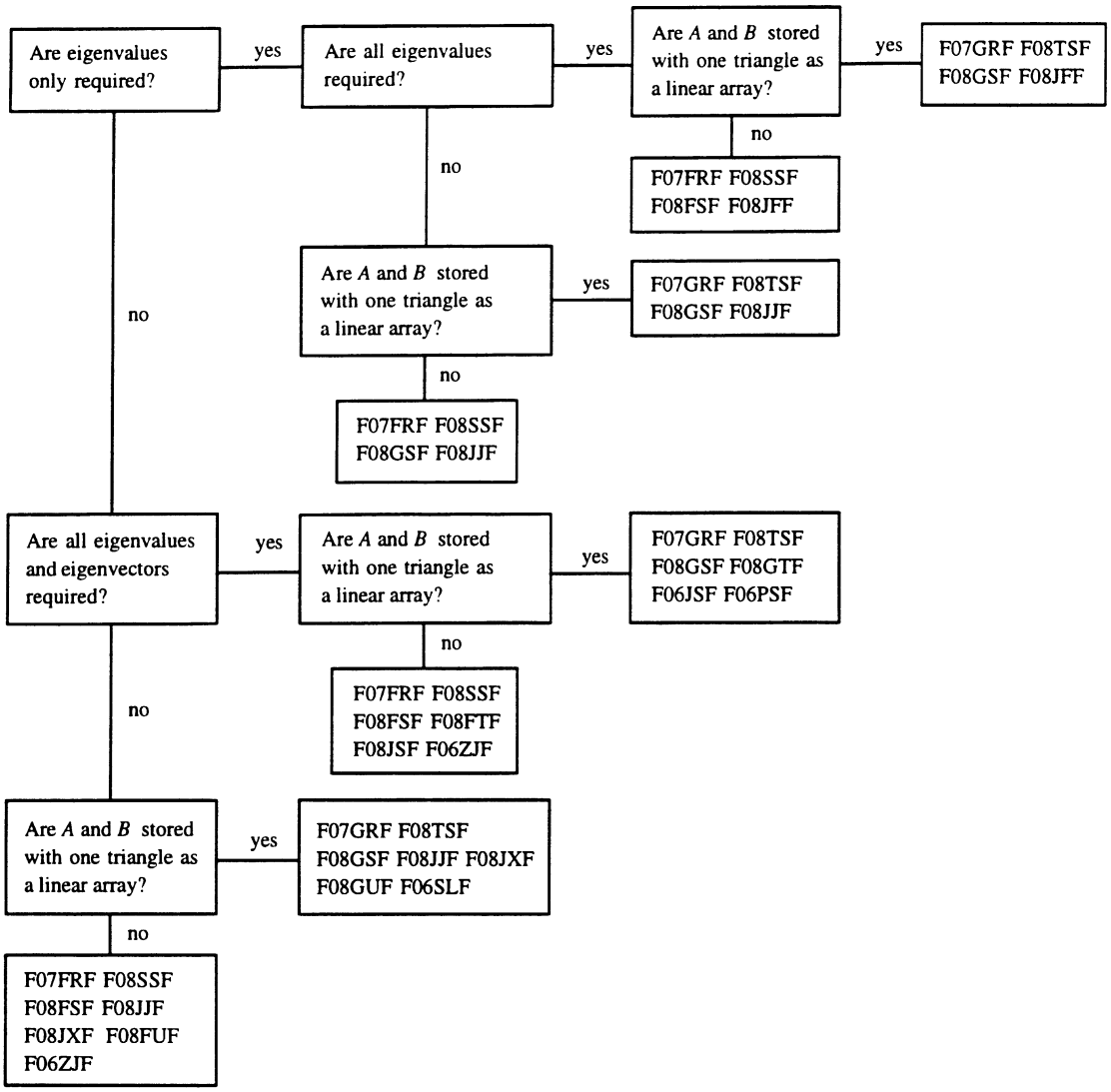
Tree 3: Real Nonsymmetric Matrices



Tree 4: Complex Hermitian Matrices

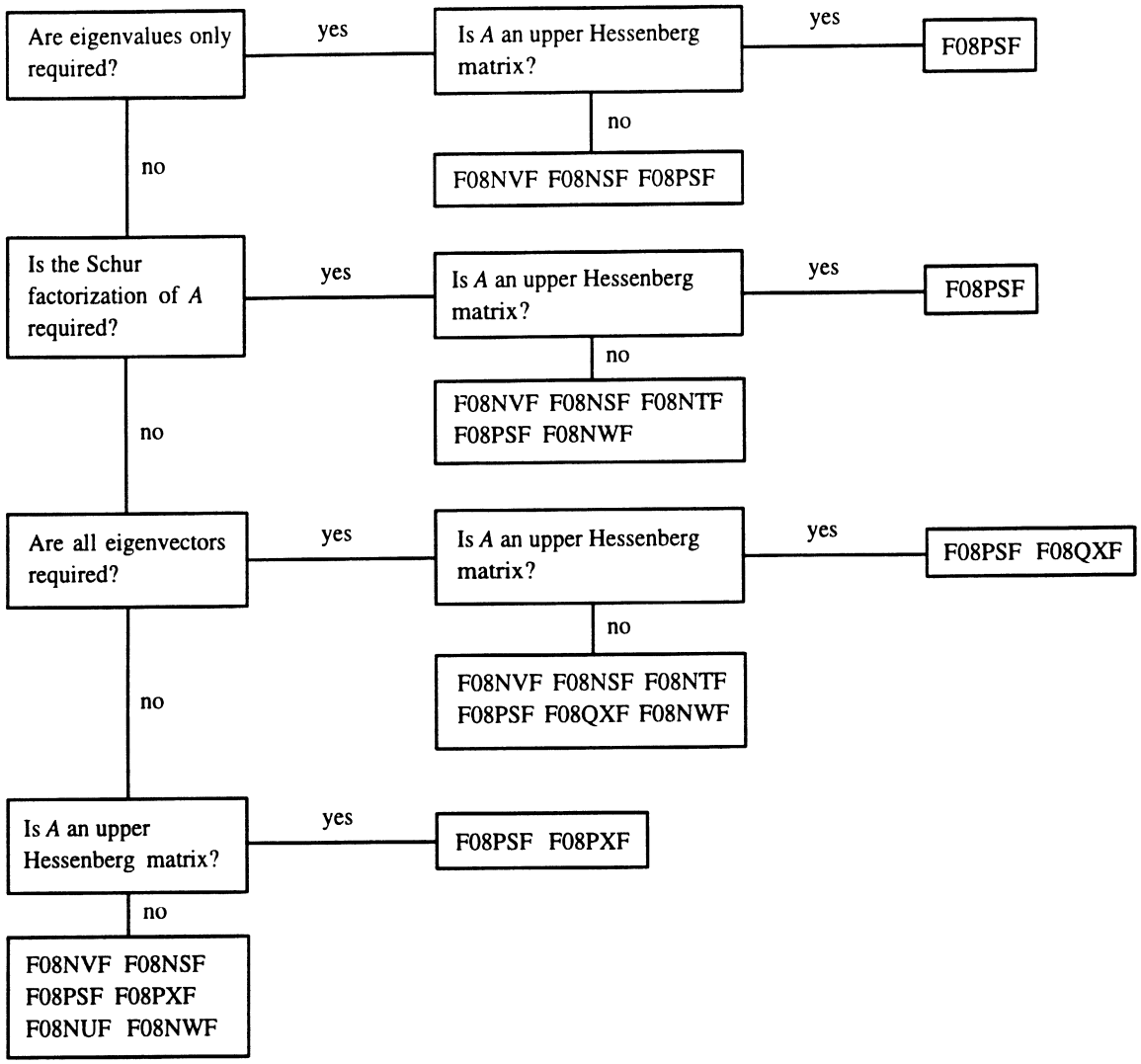


Tree 5: Complex Generalized Hermitian-definite Eigenvalue Problems

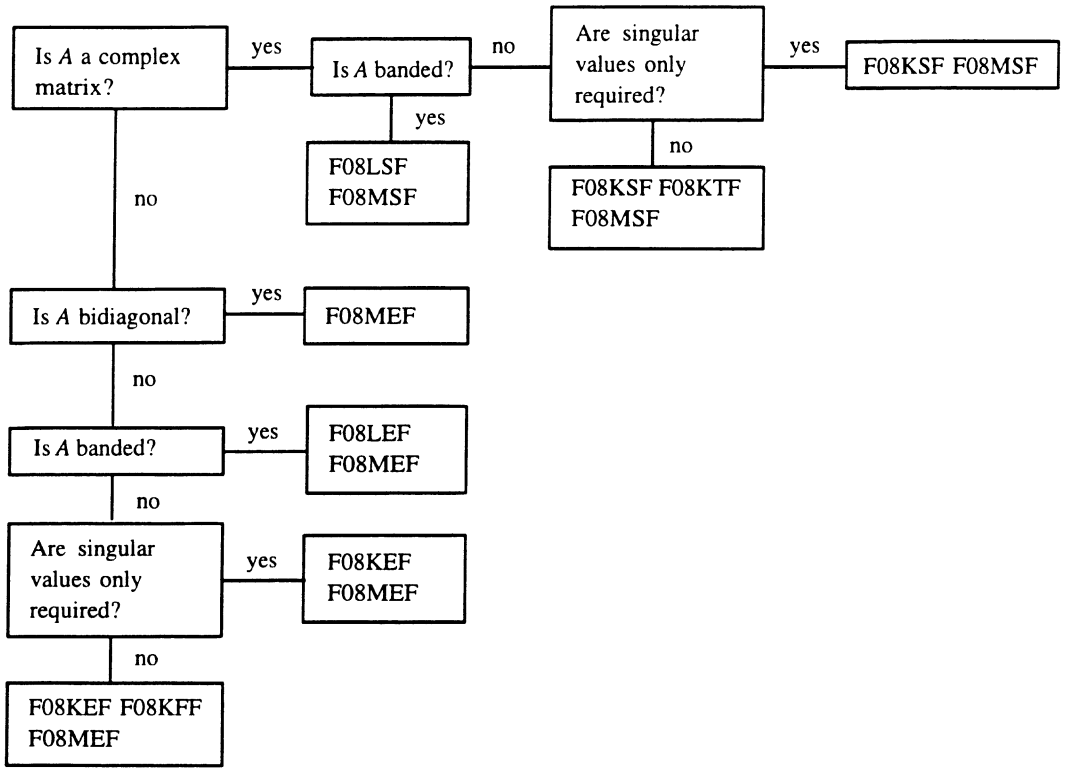


Note: the same routines are required for all three types of problem ($Ax = \lambda Bx$, $ABx = \lambda x$ or $BAx = \lambda x$) except that, if the problem is $BAx = \lambda x$ and eigenvectors are required, F06SHF must be used instead of F06SLF, and F06ZFF instead of F06ZJF.

Tree 6: Complex Nonhermitian Matrices



4.2 General purpose routines (singular value decomposition)



5 Indexes of LAPACK Routines

Real Matrices		
LAPACK single precision	LAPACK double precision	NAG
SBDSQR	DBDSQR	F08MEF
SGBBRD	DGBBRD	F08LEF
SGEBAK	DGEBAK	F08NJF
SGEBAL	DGEBAL	F08NHF
SGEBRD	DGEBRD	F08KEF
SGEHRD	DGEHRD	F08NEF
SGELQF	DGELQF	F08AHF
SGEQPF	DGEQPF	F08BEF
SGEQRF	DGEQRF	F08AEF
SHSEIN	DHSEIN	F08PKF
SHSEQR	DHSEQR	F08PEF
SOPGTR	DOPGTR	F08GFF
SOPMTR	DOPMTR	F08GGF
SORGBR	DORGBR	F08KFF
SORGHR	DORGHR	F08NFF
SORGLQ	DORGLQ	F08AJF
SORGQR	DORGQR	F08AFF
SORGTR	DORGTR	F08FFF
SORMBR	DORMBR	F08KGF
SORMHR	DORMHR	F08NGF
SORMLQ	DORMLQ	F08AKF
SORMQR	DORMQR	F08AGF
SORMTR	DORMTR	F08FGF
SPBSTF	DPBSTF	F08UFF
SPTEQR	DPTEQR	F08JGF
SSBEVD	DSBEVD	F08HCF
SSBGST	DSBGST	F08UEF
SSBTRD	DSBTRD	F08HEF
SSPEVD	DSPEVD	F08GCF
SSPGST	DSPGST	F08TEF
SSPTRD	DSPTRD	F08GEF
SSTEBZ	DSTEBZ	F08JJF
SSTEIN	DSTEIN	F08JKF
SSTEQR	DSTEQR	F08JEF
SSTERF	DSTERF	F08JFF
SSTEVD	DSTEVD	F08JCF
SSYEVD	DSYEVD	F08FCF
SSYGST	DSYGST	F08SEF
SSYTRD	DSYTRD	F08FEF
STREVC	DTREVC	F08QKF
STREXC	DTREXC	F08QFF
STRSEN	DTRSEN	F08QGF
STRSNA	DTRSNA	F08QLF
STRSYL	DTRSYL	F08QHF

Complex Matrices		
LAPACK single precision	LAPACK double precision	NAG
CBDSQR	ZBDSQR	F08MSF
CGBBRD	ZGBBRD	F08LSF
CGEBAK	ZGEBAK	F08NWF
CGEBAL	ZGEBAL	F08NVF
CGEBRD	ZGEBRD	F08KSF
CGEHRD	ZGEHRD	F08NSF
CGELQF	ZGELQF	F08AVF
CGEQPF	ZGEQPF	F08BSF
CGEQRF	ZGEQRF	F08ASF
CHBEVD	ZHBEVD	F08HQF
CHBGST	ZHBGST	F08USF
CHBTRD	ZHBTRD	F08HSF
CHEEVD	ZHEEVD	F08FQF
CHEGST	ZHEGST	F08SSF
CHETRD	ZHETRD	F08FSF
CHPEVD	ZHPEVD	F08GQF
CHPGST	ZHPGST	F08TSF
CHPTRD	ZHPTRD	F08GSF
CHSEIN	ZHSEIN	F08PXF
CHSEQR	ZHSEQR	F08PSF
CPBSTF	ZPBSTF	F08UTF
CPTEQR	ZPTEQR	F08JUF
CSTEIN	ZSTEIN	F08JXF
CSTEQR	ZSTEQR	F08JSF
CTREVC	ZTREVC	F08QXF
CTREXC	ZTREXC	F08QTF
CTRSEN	ZTRSEN	F08QUF
CTRSNA	ZTRSNA	F08QYF
CTRSYL	ZTRSYL	F08QVF
CUNGBR	ZUNGBR	F08KTF
CUNGHR	ZUNGHR	F08NTF
CUNGLQ	ZUNGLQ	F08AWF
CUNGQR	ZUNGQR	F08ATF
CUNGTR	ZUNGTR	F08FTF
CUNMBR	ZUNMBR	F08KUF
CUNMHR	ZUNMHR	F08NUF
CUNMLQ	ZUNMLQ	F08AXF
CUNMQR	ZUNMQR	F08AUF
CUNMTR	ZUNMTR	F08FUF
CUPGTR	ZUPGTR	F08GTF
CUPMTR	ZUPMTR	F08GUF

Table 4

6 Routines Withdrawn or Scheduled for Withdrawal

None since Mark 13.

7 References

[1] Anderson E, Bai Z, Bischof C, Demmel J, Dongarra J J, Du Croz J J, Greenbaum A, Hammarling S, McKenney A, Ostrouchov S and Sorensen D (1995) *LAPACK Users' Guide* (2nd Edition) SIAM, Philadelphia

[2] Arioli M, Duff I S and De Rijk P P M (1989) On the augmented system approach to sparse least-squares problems *Numer. Math.* **55** 667–684

[3] Demmel J W and Kahan W (1990) Accurate singular values of bidiagonal matrices *SIAM J. Sci. Statist. Comput.* **11** 873–912

[4] Golub G H and Van Loan C F (1996) *Matrix Computations* Johns Hopkins University Press (3rd Edition), Baltimore

[5] Parlett B N (1980) *The Symmetric Eigenvalue Problem* Prentice–Hall

[6] Wilkinson J H (1965) *The Algebraic Eigenvalue Problem* Oxford University Press, London

[7] Wilkinson J H and Reinsch C (1971) *Handbook for Automatic Computation II, Linear Algebra* Springer–Verlag

F08AEF (SGEQRF/DGEQRF) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F08AEF (SGEQRF/DGEQRF) computes the QR factorization of a real m by n matrix.

2. Specification

```
SUBROUTINE F08AEF (M, N, A, LDA, TAU, WORK, LWORK, INFO)
ENTRY      sgeqrf  (M, N, A, LDA, TAU, WORK, LWORK, INFO)
INTEGER        M, N, LDA, LWORK, INFO
real           A(LDA,*), TAU(*), WORK(LWORK)
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine forms the QR factorization of an arbitrary rectangular real m by n matrix. No pivoting is performed.

If $m \geq n$, the factorization is given by:

$$A = Q\begin{pmatrix} R \\ 0 \end{pmatrix}$$

where R is an n by n upper triangular matrix and Q is an m by m orthogonal matrix. It is sometimes more convenient to write the factorization as

$$A = (Q_1 \; Q_2)\begin{pmatrix} R \\ 0 \end{pmatrix}$$

which reduces to

$$A = Q_1 R,$$

where Q_1 consists of the first n columns of Q, and Q_2 the remaining $m-n$ columns.

If $m < n$, R is trapezoidal, and the factorization can be written

$$A = Q(R_1 \; R_2),$$

where R_1 is upper triangular and R_2 is rectangular.

The matrix Q is not formed explicitly but is represented as a product of $\min(m,n)$ elementary reflectors (see the Chapter Introduction for details). Routines are provided to work with Q in this representation (see Section 8).

Note also that for any $k < n$, the information returned in the first k columns of the array A represents a QR factorization of the first k columns of the original matrix A.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §5.2.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: M – INTEGER. *Input*

On entry: m, the number of rows of the matrix A.

Constraint: M $\geq$ 0.

2: N – INTEGER. *Input*

On entry: n, the number of columns of the matrix A.

Constraint: N $\geq$ 0.

3: A(LDA,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the m by n matrix A.

On exit: if $m \geq n$, the elements below the diagonal are overwritten by details of the orthogonal matrix Q and the upper triangle is overwritten by the corresponding elements of the n by n upper triangular matrix R.

If $m < n$, the strictly lower triangular part is overwritten by details of the orthogonal matrix Q and the remaining elements are overwritten by the corresponding elements of the m by n upper trapezoidal matrix R.

4: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F08AEF (SGEQRF/DGEQRF) is called.

Constraint: LDA $\geq$ max(1,M).

5: TAU(*) – ***real*** array. *Output*

Note: the dimension of the array TAU must be at least max(1,min(M,N)).

On exit: further details of the orthogonal matrix Q.

6: WORK(LWORK) – ***real*** array. *Workspace*

On exit: if INFO = 0, WORK(1) contains the minimum value of LWORK required for optimum performance.

7: LWORK – INTEGER. *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08AEF (SGEQRF/DGEQRF) is called.

Suggested value: for optimum performance LWORK should be at least N$\times nb$, where nb is the ***blocksize***.

Constraint: LWORK $\geq$ max(1,N).

8: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed factorization is the exact factorization of a nearby matrix $A + E$, where

$$\|E\|_2 = O(\varepsilon)\|A\|_2,$$

and ε is the ***machine precision***.

8. Further Comments

The total number of floating-point operations is approximately $\frac{2}{3}n^2(3m-n)$ if $m \geq n$ or $\frac{2}{3}m^2(3n-m)$ if $m < n$.

To form the orthogonal matrix Q this routine may be followed by a call to F08AFF (SORGQR/DORGQR):

```
CALL SORGQR (M,M,MIN(M,N),A,LDA,TAU,WORK,LWORK,INFO)
```

but note that the second dimension of the array A must be at least M, which may be larger than was required by F08AEF.

When $m \geq n$, it is often only the first n columns of Q that are required, and they may be formed by the call:

```
CALL SORGQR (M,N,N,A,LDA,TAU,WORK,LWORK,INFO)
```

To apply Q to an arbitrary real rectangular matrix C, this routine may be followed by a call to F08AGF (SORMQR/DORMQR). For example,

```
CALL SORMQR ('Left','Transpose',M,P,MIN(M,N),A,LDA,TAU,C,LDC,WORK,
+            LWORK,INFO)
```

forms $C = Q^T C$, where C is m by p.

To compute a QR factorization with column pivoting, use F08BEF (SGEQPF/DGEQPF).

The complex analogue of this routine is F08ASF (CGEQRF/ZGEQRF).

9. Example

To solve the linear least-squares problem

minimize $\|Ax_i - b_i\|_2$ for $i = 1,2$

where b_1 and b_2 are the columns of the matrix B,

$$A = \begin{pmatrix} -0.57 & -1.28 & -0.39 & 0.25 \\ -1.93 & 1.08 & -0.31 & -2.14 \\ 2.30 & 0.24 & 0.40 & -0.35 \\ -1.93 & 0.64 & -0.66 & 0.08 \\ 0.15 & 0.30 & 0.15 & -2.13 \\ -0.02 & 1.03 & -1.43 & 0.50 \end{pmatrix} \text{ and } B = \begin{pmatrix} -3.15 & 2.19 \\ -0.11 & -3.64 \\ 1.99 & 0.57 \\ -2.70 & 8.23 \\ 0.26 & -6.35 \\ 4.50 & -1.48 \end{pmatrix}.$$

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F08AEF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          MMAX, NMAX, LDA, LDB, NRHMAX, LWORK
      PARAMETER        (MMAX=8,NMAX=8,LDA=MMAX,LDB=MMAX,NRHMAX=NMAX,
     +                 LWORK=64*NMAX)
      real             ONE
      PARAMETER        (ONE=1.0e0)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, M, N, NRHS
*     .. Local Arrays ..
      real             A(LDA,NMAX), B(LDB,NRHMAX), TAU(NMAX),
     +                 WORK(LWORK)
*     .. External Subroutines ..
      EXTERNAL         sgeqrf, sormqr, strsm, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08AEF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) M, N, NRHS
```

```
      IF (M.LE.MMAX .AND. N.LE.NMAX .AND. M.GE.N .AND. NRHS.LE.NRHMAX)
     +    THEN
*
*        Read A and B from data file
*
         READ (NIN,*) ((A(I,J),J=1,N),I=1,M)
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,M)
*
*        Compute the QR factorization of A
*
         CALL sgeqrf(M,N,A,LDA,TAU,WORK,LWORK,INFO)
*
*        Compute C = (Q**T)*B, storing the result in B
*
         CALL sormqr('Left','Transpose',M,NRHS,N,A,LDA,TAU,B,LDB,WORK,
     +               LWORK,INFO)
*
*        Compute least-squares solution by backsubstitution in R*X = C
*
         CALL strsm('Left','Upper','No transpose','Non-Unit',N,NRHS,ONE,
     +              A,LDA,B,LDB)
*
*        Print least-squares solution(s)
*
         WRITE (NOUT,*)
         IFAIL = 0
*
         CALL X04CAF('General',' ',N,NRHS,B,LDB,
     +               'Least-squares solution(s)',IFAIL)
*
       END IF
       STOP
       END
```

9.2. Program Data

```
F08AEF Example Program Data
  6  4  2                          :Values of M, N and NRHS
 -0.57  -1.28  -0.39   0.25
 -1.93   1.08  -0.31  -2.14
  2.30   0.24   0.40  -0.35
 -1.93   0.64  -0.66   0.08
  0.15   0.30   0.15  -2.13
 -0.02   1.03  -1.43   0.50   :End of matrix A
 -3.15   2.19
 -0.11  -3.64
  1.99   0.57
 -2.70   8.23
  0.26  -6.35
  4.50  -1.48                  :End of matrix B
```

9.3. Program Results

```
F08AEF Example Program Results

Least-squares solution(s)
                1          2
1          1.5146    -1.5838
2          1.8621     0.5536
3         -1.4467     1.3491
4          0.0396     2.9600
```

F08AFF (SORGQR/DORGQR) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F08AFF (SORGQR/DORGQR) generates all or part of the real orthogonal matrix Q from a QR factorization computed by F08AEF (SGEQRF/DGEQRF) or F08BEF (SGEQPF/DGEQPF).

2. Specification

```
SUBROUTINE F08AFF (M, N, K, A, LDA, TAU, WORK, LWORK, INFO)
ENTRY      sorgqr  (M, N, K, A, LDA, TAU, WORK, LWORK, INFO)
INTEGER        M, N, K, LDA, LWORK, INFO
real           A(LDA,*), TAU(*), WORK(LWORK)
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine is intended to be used after a call to F08AEF (SGEQRF/DGEQRF) or F08BEF (SGEQPF/DGEQPF), which perform a QR factorization of a real matrix A. F08AEF and F08BEF represent the orthogonal matrix Q as a product of elementary reflectors.

This routine may be used to generate Q explicitly as a square matrix, or to form only its leading columns.

Usually Q is determined from the QR factorization of an m by p matrix A with $m \geq p$. The whole of Q may be computed by:

```
CALL SORGQR (M,M,P,A,LDA,TAU,WORK,LWORK,INFO)
```

(note that the array A must have at least m columns) or its leading p columns by:

```
CALL SORGQR (M,P,P,A,LDA,TAU,WORK,LWORK,INFO)
```

The columns of Q returned by the last call form an orthonormal basis for the space spanned by the columns of A; thus F08AEF followed by F08AFF can be used to orthogonalise the columns of A.

The information returned by the QR factorization routines also yields the QR factorization of the leading k columns of A, where $k < p$. The orthogonal matrix arising from this factorization can be computed by:

```
CALL SORGQR (M,M,K,A,LDA,TAU,WORK,LWORK,INFO)
```

or its leading k columns by:

```
CALL SORGQR (M,K,K,A,LDA,TAU,WORK,LWORK,INFO)
```

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §5.2.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: M – INTEGER. *Input*

On entry: m, the order of the orthogonal matrix Q.

Constraint: M $\geq$ 0.

2: N – INTEGER. *Input*

On entry: n, the number of columns of matrix Q that are required.

Constraint: M $\geq$ N $\geq$ 0.

3: K – INTEGER. *Input*

On entry: k, the number of elementary reflectors whose product defines the matrix Q.

Constraint: N ≥ K ≥ 0.

4: A(LDA,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array A must be at least max(1,N).

On entry: details of the vectors which define the elementary reflectors, as returned by F08AEF (SGEQRF/DGEQRF) or F08BEF (SGEQPF/DGEQPF).

On exit: the m by n matrix Q.

5: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F08AFF (SORGQR/DORGQR) is called.

Constraint: LDA ≥ max(1,M).

6: TAU(*) – ***real*** array. *Input*

Note: the dimension of the array TAU must be at least max(1,K).

On entry: further details of the elementary reflectors, as returned by F08AEF (SGEQRF/DGEQRF) or F08BEF (SGEQPF/DGEQPF).

7: WORK(LWORK) – ***real*** array. *Workspace*

On exit: if INFO = 0, WORK(1) contains the minimum value of LWORK required for optimum performance.

8: LWORK – INTEGER. *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08AFF (SORGQR/DORGQR) is called.

Suggested value: for optimum performance LWORK should be at least N×nb, where nb is the ***blocksize***.

Constraint: LWORK ≥ max(1,N).

9: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed matrix Q differs from an exactly orthogonal matrix by a matrix E such that

$$\|E\|_2 = O(\varepsilon),$$

where ε is the ***machine precision***.

8. Further Comments

The total number of floating-point operations is approximately $4mnk - 2(m+n)k^2 + \frac{4}{3}k^3$; when $n = k$, the number is approximately $\frac{2}{3}n^2(3m-n)$.

The complex analogue of this routine is F08ATF (CUNGQR/ZUNGQR).

9. Example

To form the leading 4 columns of the orthogonal matrix Q from the QR factorization of the matrix A, where

$$A = \begin{pmatrix} -0.57 & -1.28 & -0.39 & 0.25 \\ -1.93 & 1.08 & -0.31 & -2.14 \\ 2.30 & 0.24 & 0.40 & -0.35 \\ -1.93 & 0.64 & -0.66 & 0.08 \\ 0.15 & 0.30 & 0.15 & -2.13 \\ -0.02 & 1.03 & -1.43 & 0.50 \end{pmatrix}.$$

The columns of Q form an orthonormal basis for the space spanned by the columns of A.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F08AFF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          MMAX, NMAX, LDA, LWORK
      PARAMETER        (MMAX=8,NMAX=8,LDA=MMAX,LWORK=64*NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, M, N
      CHARACTER*30     TITLE
*     .. Local Arrays ..
      real             A(LDA,NMAX), TAU(NMAX), WORK(LWORK)
*     .. External Subroutines ..
      EXTERNAL         sgeqrf, sorgqr, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08AFF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) M, N
      IF (M.LE.MMAX .AND. N.LE.NMAX .AND. M.GE.N) THEN
*
*        Read A from data file
*
         READ (NIN,*) ((A(I,J),J=1,N),I=1,M)
*
*        Compute the QR factorization of A
*
         CALL sgeqrf(M,N,A,LDA,TAU,WORK,LWORK,INFO)
*
*        Form the leading N columns of Q explicitly
*
         CALL sorgqr(M,N,N,A,LDA,TAU,WORK,LWORK,INFO)
*
*        Print the leading N columns of Q only
*
         WRITE (NOUT,*)
         WRITE (TITLE,99999) N
         IFAIL = 0
*
         CALL X04CAF('General',' ',M,N,A,LDA,TITLE,IFAIL)
*
      END IF
      STOP
*
99999 FORMAT ('The leading ',I2,' columns of Q')
      END
```

9.2. Program Data

```
F08AFF Example Program Data
  6  4                                :Values of M and N
 -0.57  -1.28  -0.39   0.25
 -1.93   1.08  -0.31  -2.14
  2.30   0.24   0.40  -0.35
 -1.93   0.64  -0.66   0.08
  0.15   0.30   0.15  -2.13
 -0.02   1.03  -1.43   0.50   :End of matrix A
```

9.3. Program Results

```
F08AFF Example Program Results

The leading  4 columns of Q
         1       2       3       4
1  -0.1576  0.6744 -0.4571  0.4489
2  -0.5335 -0.3861  0.2583  0.3898
3   0.6358 -0.2928  0.0165  0.1930
4  -0.5335 -0.1692 -0.0834 -0.2350
5   0.0415 -0.1593  0.1475  0.7436
6  -0.0055 -0.5064 -0.8339  0.0335
```

F08AGF (SORMQR/DORMQR) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F08AGF (SORMQR/DORMQR) multiplies an arbitrary real matrix C by the real orthogonal matrix Q from a QR factorization computed by F08AEF (SGEQRF/DGEQRF) or F08BEF (SGEQPF/DGEQPF).

2. Specification

```
      SUBROUTINE F08AGF (SIDE, TRANS, M, N, K, A, LDA, TAU, C, LDC, WORK,
     1                   LWORK, INFO)
      ENTRY       sormqr (SIDE, TRANS, M, N, K, A, LDA, TAU, C, LDC, WORK,
     1                   LWORK, INFO)
      INTEGER          M, N, K, LDA, LDC, LWORK, INFO
      real             A(LDA,*), TAU(*), C(LDC,*), WORK(LWORK)
      CHARACTER*1      SIDE, TRANS
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine is intended to be used after a call to F08AEF (SGEQRF/DGEQRF) or F08BEF (SGEQPF/DGEQPF), which perform a QR factorization of a real matrix A. F08AEF and F08BEF represent the orthogonal matrix Q as a product of elementary reflectors.

This routine may be used to form one of the matrix products

$$QC, \quad Q^TC, \quad CQ \text{ or } CQ^T,$$

overwriting the result on C (which may be any real rectangular matrix).

A common application of this routine is in solving linear least-squares problems, as described in the Chapter Introduction, and illustrated in Section 9 of the document for F08AEF.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: SIDE – CHARACTER*1. *Input*

On entry: indicates how Q or Q^T is to be applied to C as follows:

if SIDE = 'L', then Q or Q^T is applied to C from the left;

if SIDE = 'R', then Q or Q^T is applied to C from the right.

Constraint: SIDE = 'L' or 'R'.

2: TRANS – CHARACTER*1. *Input*

On entry: indicates whether Q or Q^T is to be applied to C as follows:

if TRANS = 'N', then Q is applied to C;

if TRANS = 'T', then Q^T is applied to C.

Constraint: TRANS = 'N' or 'T'.

3: M – INTEGER. *Input*

On entry: m, the number of rows of the matrix C.

Constraint: M ≥ 0.

4: N – INTEGER. *Input*

On entry: n, the number of columns of the matrix C.

Constraint: N ≥ 0.

5: K – INTEGER. *Input*

On entry: k, the number of elementary reflectors whose product defines the matrix Q.

Constraints: M ≥ K ≥ 0 if SIDE = 'L',
N ≥ K ≥ 0 if SIDE = 'R'.

6: A(LDA,*) – ***real*** array. *Input*

Note: the second dimension of the array A must be at least max(1,K).

On entry: details of the vectors which define the elementary reflectors, as returned by F08AEF (SGEQRF/DGEQRF) or F08BEF (SGEQPF/DGEQPF).

7: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F08AGF (SORMQR/DORMQR) is called.

Constraints: LDA ≥ max(1,M) if SIDE = 'L',
LDA ≥ max(1,N) if SIDE = 'R'.

8: TAU(*) – ***real*** array. *Input*

Note: the dimension of the array TAU must be at least max(1,K).

On entry: further details of the elementary reflectors, as returned by F08AEF (SGEQRF/DGEQRF) or F08BEF (SGEQPF/DGEQPF).

9: C(LDC,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array C must be at least max(1,N).

On entry: the m by n matrix C.

On exit: C is overwritten by QC or Q^TC or CQ^T or CQ as specified by SIDE and TRANS.

10: LDC – INTEGER. *Input*

On entry: the first dimension of the array C as declared in the (sub)program from which F08AGF (SORMQR/DORMQR) is called.

Constraint: LDC ≥ max(1,M).

11: WORK(LWORK) – ***real*** array. *Workspace*

On exit: if INFO = 0, WORK(1) contains the minimum value of LWORK required for optimum performance.

12: LWORK – INTEGER. *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08AGF (SORMQR/DORMQR) is called.

Suggested value: for optimum performance LWORK should be at least N×nb if SIDE = 'L' and at least M×nb if SIDE = 'R', where nb is the ***blocksize***.

Constraints: LWORK ≥ max(1,N) if SIDE = 'L',
LWORK ≥ max(1,M) if SIDE = 'R'.

13: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed result differs from the exact result by a matrix E such that

$$\|E\|_2 = O(\varepsilon)\|C\|_2,$$

where ε is the ***machine precision***.

8. Further Comments

The total number of floating-point operations is approximately $2nk(2m-k)$ if SIDE = 'L' and $2mk(2n-k)$ if SIDE = 'R'.

The complex analogue of this routine is F08AUF (CUNMQR/ZUNMQR).

9. Example

See the example for F08AEF (SGEQRF/DGEQRF).

F08AHF (SGELQF/DGELQF) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F08AHF (SGELQF/DGELQF) computes the LQ factorization of a real m by n matrix.

2. Specification

```
SUBROUTINE F08AHF (M, N, A, LDA, TAU, WORK, LWORK, INFO)
ENTRY      sgelqf  (M, N, A, LDA, TAU, WORK, LWORK, INFO)
INTEGER        M, N, LDA, LWORK, INFO
real           A(LDA,*), TAU(*), WORK(LWORK)
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine forms the LQ factorization of an arbitrary rectangular real m by n matrix. No pivoting is performed.

If $m \leq n$, the factorization is given by:

$$A = (L \ 0)Q$$

where L is an m by m lower triangular matrix and Q is an n by n orthogonal matrix. It is sometimes more convenient to write the factorization as

$$A = (L \ 0)\begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix}$$

which reduces to

$$A = LQ_1,$$

where Q_1 consists of the first m rows of Q, and Q_2 the remaining $n-m$ rows.

If $m > n$, L is trapezoidal, and the factorization can be written

$$A = \begin{pmatrix} L_1 \\ L_2 \end{pmatrix}Q$$

where L_1 is lower triangular and L_2 is rectangular.

The LQ factorization of A is essentially the same as the QR factorization of A^T, since

$$A = (L \ 0)Q \Leftrightarrow A^T = Q^T\begin{pmatrix} L^T \\ 0 \end{pmatrix}.$$

The matrix Q is not formed explicitly but is represented as a product of $\min(m,n)$ elementary reflectors (see the Chapter Introduction for details). Routines are provided to work with Q in this representation (see Section 8).

Note also that for any $k < m$, the information returned in the first k rows of the array A represents an LQ factorization of the first k rows of the original matrix A.

4. References

None.

5. Parameters

1: M – INTEGER. *Input*

On entry: m, the number of rows of the matrix A.

Constraint: M $\geq$ 0.

2: N – INTEGER. *Input*

On entry: n, the number of columns of the matrix A.

Constraint: N $\geq$ 0.

3: A(LDA,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the m by n matrix A.

On exit: if $m \leq n$, the elements above the diagonal are overwritten by details of the orthogonal matrix Q and the lower triangle is overwritten by the corresponding elements of the m by m lower triangular matrix L.

If $m > n$, the strictly upper triangular part is overwritten by details of the orthogonal matrix Q and the remaining elements are overwritten by the corresponding elements of the m by n lower trapezoidal matrix L.

4: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F08AHF (SGELQF/DGELQF) is called.

Constraint: LDA $\geq$ max(1,M).

5: TAU(*) – ***real*** array. *Output*

Note: the dimension of the array TAU must be at least max(1,min(M,N)).

On exit: further details of the orthogonal matrix Q.

6: WORK(LWORK) – ***real*** array. *Workspace*

On exit: if INFO = 0, WORK(1) contains the minimum value of LWORK required for optimum performance.

7: LWORK – INTEGER. *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08AHF (SGELQF/DGELQF) is called.

Suggested value: for optimum performance LWORK should be at least M$\times nb$, where nb is the ***blocksize***.

Constraint: LWORK $\geq$ max(1,M).

8: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed factorization is the exact factorization of a nearby matrix $A + E$, where

$$\|E\|_2 = O(\varepsilon)\|A\|_2,$$

and ε is the ***machine precision***.

8. Further Comments

The total number of floating-point operations is approximately $\frac{2}{3}m^2(3n-m)$ if $m \leq n$ or $\frac{2}{3}n^2(3m-n)$ if $m > n$.

To form the orthogonal matrix Q this routine may be followed by a call to F08AJF (SORGLQ/DORGLQ):

```
CALL SORGLQ (N,N,MIN(M,N),A,LDA,TAU,WORK,LWORK,INFO)
```

but note that the first dimension of the array A, specified by the parameter LDA, must be at least N, which may be larger than was required by F08AHF.

When $m \leq n$, it is often only the first m rows of Q that are required, and they may be formed by the call:

```
CALL SORGLQ (M,N,M,A,LDA,TAU,WORK,LWORK,INFO)
```

To apply Q to an arbitrary real rectangular matrix C, this routine may be followed by a call to F08AKF (SORMLQ/DORMLQ). For example,

```
CALL SORMLQ ('Left','Transpose',M,P,MIN(M,N),A,LDA,TAU,C,LDC,
+            WORK,LWORK,INFO)
```

forms the matrix product $C = Q^T C$, where C is m by p.

The complex analogue of this routine is F08AVF (CGELQF/ZGELQF).

9. Example

To find the minimum-norm solutions of the under-determined systems of linear equations

$$Ax_1 = b_1 \text{ and } Ax_2 = b_2$$

where b_1 and b_2 are the columns of the matrix B,

$$A = \begin{pmatrix} -5.42 & 3.28 & -3.68 & 0.27 & 2.06 & 0.46 \\ -1.65 & -3.40 & -3.20 & -1.03 & -4.06 & -0.01 \\ -0.37 & 2.35 & 1.90 & 4.31 & -1.76 & 1.13 \\ -3.15 & -0.11 & 1.99 & -2.70 & 0.26 & 4.50 \end{pmatrix} \text{ and } B = \begin{pmatrix} -2.87 & -5.23 \\ 1.63 & 0.29 \\ -3.52 & 4.76 \\ 0.45 & -8.41 \end{pmatrix}.$$

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F08AHF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          MMAX, NMAX, LDA, LDB, NRHMAX, LWORK
      PARAMETER        (MMAX=8,NMAX=8,LDA=MMAX,LDB=NMAX,NRHMAX=NMAX,
     +                 LWORK=64*NMAX)
      real             ZERO, ONE
      PARAMETER        (ZERO=0.0e0,ONE=1.0e0)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, M, N, NRHS
*     .. Local Arrays ..
      real             A(LDA,NMAX), B(LDB,NRHMAX), TAU(NMAX),
     +                 WORK(LWORK)
*     .. External Subroutines ..
      EXTERNAL         sgelqf, sormlq, strsm, F06QHF, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08AHF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) M, N, NRHS
      IF (M.LE.MMAX .AND. N.LE.NMAX .AND. M.LE.N .AND. NRHS.LE.NRHMAX)
     +    THEN
*
*        Read A and B from data file
```

```
*
      READ (NIN,*) ((A(I,J),J=1,N),I=1,M)
      READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,M)
*
*     Compute the LQ factorization of A
*
      CALL sgelqf(M,N,A,LDA,TAU,WORK,LWORK,INFO)
*
*     Solve L*Y = B, storing the result in B
*
      CALL strsm('Left','Lower','No transpose','Non-Unit',M,NRHS,ONE,
     +           A,LDA,B,LDB)
*
*     Set rows (M+1) to N of B to zero
*
      IF (M.LT.N) CALL F06QHF('General',N-M,NRHS,ZERO,ZERO,B(M+1,1),
     +                        LDB)
*
*     Compute minimum-norm solution X = (Q**T)*B in B
*
      CALL sormlq('Left','Transpose',N,NRHS,M,A,LDA,TAU,B,LDB,WORK,
     +            LWORK,INFO)
*
*     Print minimum-norm solution(s)
*
      WRITE (NOUT,*)
      IFAIL = 0
*
      CALL X04CAF('General',' ',N,NRHS,B,LDB,
     +            'Minimum-norm solution(s)',IFAIL)
*
    END IF
    STOP
    END
```

9.2. Program Data

```
F08AHF Example Program Data
  4  6  2                                          :Values of M, N and NRHS
 -5.42   3.28  -3.68   0.27   2.06   0.46
 -1.65  -3.40  -3.20  -1.03  -4.06  -0.01
 -0.37   2.35   1.90   4.31  -1.76   1.13
 -3.15  -0.11   1.99  -2.70   0.26   4.50   :End of matrix A
 -2.87  -5.23
  1.63   0.29
 -3.52   4.76
  0.45  -8.41                                     :End of matrix B
```

9.3. Program Results

```
F08AHF Example Program Results

Minimum-norm solution(s)
                1          2
1          0.2371     0.7383
2         -0.4575     0.0158
3         -0.0085    -0.0161
4         -0.5192     1.0768
5          0.0239    -0.6436
6         -0.0543    -0.6613
```

F08AJF (SORGLQ/DORGLQ) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F08AJF (SORGLQ/DORGLQ) generates all or part of the real orthogonal matrix Q from an LQ factorization computed by F08AHF (SGELQF/DGELQF).

2. Specification

```
SUBROUTINE F08AJF (M, N, K, A, LDA, TAU, WORK, LWORK, INFO)
ENTRY      sorglq  (M, N, K, A, LDA, TAU, WORK, LWORK, INFO)
INTEGER        M, N, K, LDA, LWORK, INFO
real           A(LDA,*), TAU(*), WORK(LWORK)
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine is intended to be used after a call to F08AHF (SGELQF/DGELQF), which performs an LQ factorization of a real matrix A. F08AHF represents the orthogonal matrix Q as a product of elementary reflectors.

This routine may be used to generate Q explicitly as a square matrix, or to form only its leading rows.

Usually Q is determined from the LQ factorization of a p by n matrix A with $p \leq n$. The whole of Q may be computed by:

```
CALL SORGLQ (N,N,P,A,LDA,TAU,WORK,LWORK,INFO)
```

(note that the array A must have at least n rows) or its leading p rows by:

```
CALL SORGLQ (P,N,P,A,LDA,TAU,WORK,LWORK,INFO)
```

The rows of Q returned by the last call form an orthonormal basis for the space spanned by the rows of A; thus F08AHF followed by F08AJF can be used to orthogonalise the rows of A.

The information returned by the LQ factorization routines also yields the LQ factorization of the leading k rows of A, where $k < p$. The orthogonal matrix arising from this factorization can be computed by:

```
CALL SORGLQ (N,N,K,A,LDA,TAU,WORK,LWORK,INFO)
```

or its leading k rows by:

```
CALL SORGLQ (K,N,K,A,LDA,TAU,WORK,LWORK,INFO)
```

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: M – INTEGER. *Input*

On entry: m, the number of rows of the matrix Q.

Constraint: M $\geq$ 0.

2: N – INTEGER. *Input*

On entry: n, the number of columns of the matrix Q.

Constraint: N $\geq$ M.

3: K – INTEGER. *Input*

On entry: k, the number of elementary reflectors whose product defines the matrix Q.

Constraint: $M \geq K \geq 0$.

4: A(LDA,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array A must be at least max(1,N).

On entry: details of the vectors which define the elementary reflectors, as returned by F08AHF (SGELQF/DGELQF).

On exit: the m by n matrix Q.

5: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F08AJF (SORGLQ/DORGLQ) is called.

Constraint: LDA ≥ max(1,M).

6: TAU(*) – ***real*** array. *Input*

Note: the dimension of the array TAU must be at least max(1,K).

On entry: further details of the elementary reflectors, as returned by F08AHF (SGELQF/DGELQF).

7: WORK(LWORK) – ***real*** array. *Workspace*

On exit: if INFO = 0, WORK(1) contains the minimum value of LWORK required for optimum performance.

8: LWORK – INTEGER. *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08AJF (SORGLQ/DORGLQ) is called.

Suggested value: for optimum performance LWORK should be at least M×nb, where nb is the ***blocksize***.

Constraint: LWORK ≥ max(1,M).

9: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed matrix Q differs from an exactly orthogonal matrix by a matrix E such that

$$\|E\|_2 = O(\varepsilon),$$

where ε is the ***machine precision***.

8. Further Comments

The total number of floating-point operations is approximately $4mnk - 2(m+n)k^2 + \frac{4}{3}k^3$; when $m = k$, the number is approximately $\frac{2}{3}m^2(3n-m)$.

The complex analogue of this routine is F08AWF (CUNGLQ/ZUNGLQ).

9. Example

To form the leading 4 rows of the orthogonal matrix Q from the LQ factorization of the matrix A, where

$$A = \begin{pmatrix} -5.42 & 3.28 & -3.68 & 0.27 & 2.06 & 0.46 \\ -1.65 & -3.40 & -3.20 & -1.03 & -4.06 & -0.01 \\ -0.37 & 2.35 & 1.90 & 4.31 & -1.76 & 1.13 \\ -3.15 & -0.11 & 1.99 & -2.70 & 0.26 & 4.50 \end{pmatrix}.$$

The rows of Q form an orthonormal basis for the space spanned by the rows of A.

9.1. Program Text

Note: the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F08AJF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          MMAX, NMAX, LDA, LWORK
      PARAMETER        (MMAX=8,NMAX=8,LDA=MMAX,LWORK=64*MMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, M, N
      CHARACTER*30     TITLE
*     .. Local Arrays ..
      real             A(LDA,NMAX), TAU(NMAX), WORK(LWORK)
*     .. External Subroutines ..
      EXTERNAL         sgelqf, sorglq, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08AJF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) M, N
      IF (M.LE.MMAX .AND. N.LE.NMAX .AND. M.LE.N) THEN
*
*        Read A from data file
*
         READ (NIN,*) ((A(I,J),J=1,N),I=1,M)
*
*        Compute the LQ factorization of A
*
         CALL sgelqf(M,N,A,LDA,TAU,WORK,LWORK,INFO)
*
*        Form the leading M rows of Q explicitly
*
         CALL sorglq(M,N,M,A,LDA,TAU,WORK,LWORK,INFO)
*
*        Print the leading M rows of Q only
*
         WRITE (NOUT,*)
         WRITE (TITLE,99999) M
         IFAIL = 0
*
         CALL X04CAF('General',' ',M,N,A,LDA,TITLE,IFAIL)
*
      END IF
      STOP
*
99999 FORMAT ('The leading ',I2,' rows of Q')
      END
```

9.2. Program Data

```
F08AJF Example Program Data
  4  6                                          :Values of M and N
 -5.42   3.28  -3.68   0.27   2.06   0.46
 -1.65  -3.40  -3.20  -1.03  -4.06  -0.01
 -0.37   2.35   1.90   4.31  -1.76   1.13
 -3.15  -0.11   1.99  -2.70   0.26   4.50   :End of matrix A
```

9.3. Program Results

```
F08AJF Example Program Results

The leading  4 rows of Q
         1       2       3       4       5       6
1  -0.7104  0.4299 -0.4824  0.0354  0.2700  0.0603
2  -0.2412 -0.5323 -0.4845 -0.1595 -0.6311 -0.0027
3   0.1287 -0.2619 -0.2108 -0.7447  0.5227 -0.2063
4  -0.3403 -0.0921  0.4546 -0.3869 -0.0465  0.7191
```

F08AKF (SORMLQ/DORMLQ) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F08AKF (SORMLQ/DORMLQ) multiplies an arbitrary real matrix C by the real orthogonal matrix Q from an LQ factorization computed by F08AHF (SGELQF/DGELQF).

2. Specification

```
      SUBROUTINE F08AKF (SIDE, TRANS, M, N, K, A, LDA, TAU, C, LDC, WORK,
     1                   LWORK, INFO)
      ENTRY       sormlq (SIDE, TRANS, M, N, K, A, LDA, TAU, C, LDC, WORK,
     1                   LWORK, INFO)
      INTEGER          M, N, K, LDA, LDC, LWORK, INFO
      real             A(LDA,*), TAU(*), C(LDC,*), WORK(LWORK)
      CHARACTER*1      SIDE, TRANS
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine is intended to be used after a call to F08AHF (SGELQF/DGELQF), which performs an LQ factorization of a real matrix A. F08AHF represents the orthogonal matrix Q as a product of elementary reflectors.

This routine may be used to form one of the matrix products

QC, Q^TC, CQ or CQ^T,

overwriting the result on C (which may be any real rectangular matrix).

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: SIDE – CHARACTER*1. *Input*

On entry: indicates how Q or Q^T is to be applied to C as follows:

if SIDE = 'L', then Q or Q^T is applied to C from the left;

if SIDE = 'R', then Q or Q^T is applied to C from the right.

Constraint: SIDE = 'L' or 'R'.

2: TRANS – CHARACTER*1. *Input*

On entry: indicates whether Q or Q^T is to be applied to C as follows:

if TRANS = 'N', then Q is applied to C;

if TRANS = 'T', then Q^T is applied to C.

Constraint: TRANS = 'N' or 'T'.

3: M – INTEGER. *Input*

On entry: m, the number of rows of the matrix C.

Constraint: M $\geq$ 0.

4: N – INTEGER. *Input*

On entry: n, the number of columns of the matrix C.

Constraint: N ≥ 0.

5: K – INTEGER. *Input*

On entry: k, the number of elementary reflectors whose product defines the matrix Q.

Constraints: M ≥ K ≥ 0 if SIDE = 'L',
N ≥ K ≥ 0 if SIDE = 'R'.

6: A(LDA,*) – ***real*** array. *Input*

Note: the second dimension of the array A must be at least max(1,M) if SIDE = 'L' and at least max(1,N) if SIDE = 'R'.

On entry: details of the vectors which define the elementary reflectors, as returned by F08AHF (SGELQF/DGELQF).

7: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F08AKF (SORMLQ/DORMLQ) is called.

Constraint: LDA ≥ max(1,K).

8: TAU(*) – ***real*** array. *Input*

Note: the dimension of the array TAU must be at least max(1,K).

On entry: further details of the elementary reflectors, as returned by F08AHF (SGELQF/DGELQF).

9: C(LDC,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array C must be at least max(1,N).

On entry: the m by n matrix C.

On exit: C is overwritten by QC or Q^TC or CQ^T or CQ as specified by SIDE and TRANS.

10: LDC – INTEGER. *Input*

On entry: the first dimension of the array C as declared in the (sub)program from which F08AKF (SORMLQ/DORMLQ) is called.

Constraint: LDC ≥ max(1,M).

11: WORK(LWORK) – ***real*** array. *Workspace*

On exit: if INFO = 0, WORK(1) contains the minimum value of LWORK required for optimum performance.

12: LWORK – INTEGER. *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08AKF (SORMLQ/DORMLQ) is called.

Suggested value: for optimum performance LWORK should be at least N×*nb* if SIDE = 'L' and at least M×*nb* if SIDE = 'R', where *nb* is the ***blocksize***.

Constraints: LWORK ≥ max(1,N) if SIDE = 'L',
LWORK ≥ max(1,M) if SIDE = 'R'.

13: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed result differs from the exact result by a matrix E such that

$$\|E\|_2 = O(\varepsilon)\|C\|_2,$$

where ε is the ***machine precision***.

8. Further Comments

The total number of floating-point operations is approximately $2nk(2m-k)$ if SIDE = 'L' and $2mk(2n-k)$ if SIDE = 'R'.

The complex analogue of this routine is F08AXF (CUNMLQ/ZUNMLQ).

9. Example

See the example for F08AHF (SGELQF/DGELQF).

F08ASF (CGEQRF/ZGEQRF) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F08ASF (CGEQRF/ZGEQRF) computes the QR factorization of a complex m by n matrix.

2. Specification

```
SUBROUTINE F08ASF (M, N, A, LDA, TAU, WORK, LWORK, INFO)
ENTRY      cgeqrf  (M, N, A, LDA, TAU, WORK, LWORK, INFO)
INTEGER        M, N, LDA, LWORK, INFO
complex        A(LDA,*), TAU(*), WORK(LWORK)
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine forms the QR factorization of an arbitrary rectangular complex m by n matrix. No pivoting is performed.

If $m \geq n$, the factorization is given by:

$$A = Q\begin{pmatrix} R \\ 0 \end{pmatrix}$$

where R is an n by n upper triangular matrix (with real diagonal elements) and Q is an m by m unitary matrix. It is sometimes more convenient to write the factorization as

$$A = (Q_1 \; Q_2)\begin{pmatrix} R \\ 0 \end{pmatrix}$$

which reduces to

$$A = Q_1 R,$$

where Q_1 consists of the first n columns of Q, and Q_2 the remaining $m-n$ columns.

If $m < n$, R is trapezoidal, and the factorization can be written

$$A = Q(R_1 \; R_2),$$

where R_1 is upper triangular and R_2 is rectangular.

The matrix Q is not formed explicitly but is represented as a product of $\min(m,n)$ elementary reflectors (see the Chapter Introduction for details). Routines are provided to work with Q in this representation (see Section 8).

Note also that for any $k < n$, the information returned in the first k columns of the array A represents a QR factorization of the first k columns of the original matrix A.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §5.2.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: M – INTEGER. *Input*

On entry: m, the number of rows of the matrix A.

Constraint: M $\geq$ 0.

2: N – INTEGER. *Input*

On entry: n, the number of columns of the matrix A.

Constraint: N $\geq$ 0.

3: A(LDA,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the m by n matrix A.

On exit: if $m \geq n$, the elements below the diagonal are overwritten by details of the unitary matrix Q and the upper triangle is overwritten by the corresponding elements of the n by n upper triangular matrix R.

If $m < n$, the strictly lower triangular part is overwritten by details of the unitary matrix Q and the remaining elements are overwritten by the corresponding elements of the m by n upper trapezoidal matrix R.

The diagonal elements of R are real.

4: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F08ASF (CGEQRF/ZGEQRF) is called.

Constraint: LDA $\geq$ max(1,M).

5: TAU(*) – ***complex*** array. *Output*

Note: the dimension of the array TAU must be at least max(1,min(M,N)).

On exit: further details of the unitary matrix Q.

6: WORK(LWORK) – ***complex*** array. *Workspace*

On exit: if INFO = 0, WORK(1) contains the minimum value of LWORK required for optimum performance.

7: LWORK – INTEGER. *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08ASF (CGEQRF/ZGEQRF) is called.

Suggested value: for optimum performance LWORK should be at least N$\times nb$, where nb is the ***blocksize***.

Constraint: LWORK $\geq$ max(1,N).

8: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed factorization is the exact factorization of a nearby matrix $A + E$, where

$$\|E\|_2 = O(\varepsilon)\|A\|_2,$$

and ε is the ***machine precision***.

8. Further Comments

The total number of real floating-point operations is approximately $\frac{1}{3}n^2(3m-n)$ if $m \geq n$ or $\frac{1}{3}m^2(3n-m)$ if $m < n$.

To form the unitary matrix Q this routine may be followed by a call to F08ATF (CUNGQR/ZUNGQR):

```
CALL CUNGQR (M,M,MIN(M,N),A,LDA,TAU,WORK,LWORK,INFO)
```

but note that the second dimension of the array A must be at least M, which may be larger than was required by F08ASF.

When $m \geq n$, it is often only the first n columns of Q that are required, and they may be formed by the call:

```
CALL CUNGQR (M,N,N,A,LDA,TAU,WORK,LWORK,INFO)
```

To apply Q to an arbitrary complex rectangular matrix C, this routine may be followed by a call to F08AUF (CUNMQR/ZUNMQR). For example,

```
CALL CUNMQR ('Left','Conjugate Transpose',M,P,MIN(M,N),A,LDA,TAU,
+            C,LDC,WORK,LWORK,INFO)
```

forms $C = Q^H C$, where C is m by p.

To compute a QR factorization with column pivoting, use F08BSF (CGEQPF/ZGEQPF).

The real analogue of this routine is F08AEF (SGEQRF/DGEQRF).

9. Example

To solve the linear least-squares problem

minimize $\|Ax_i - b_i\|_2$ for $i = 1,2$

where b_1 and b_2 are the columns of the matrix B,

$$A = \begin{pmatrix} 0.96 - 0.81i & -0.03 + 0.96i & -0.91 + 2.06i & -0.05 + 0.41i \\ -0.98 + 1.98i & -1.20 + 0.19i & -0.66 + 0.42i & -0.81 + 0.56i \\ 0.62 - 0.46i & 1.01 + 0.02i & 0.63 - 0.17i & -1.11 + 0.60i \\ -0.37 + 0.38i & 0.19 - 0.54i & -0.98 - 0.36i & 0.22 - 0.20i \\ 0.83 + 0.51i & 0.20 + 0.01i & -0.17 - 0.46i & 1.47 + 1.59i \\ 1.08 - 0.28i & 0.20 - 0.12i & -0.07 + 1.23i & 0.26 + 0.26i \end{pmatrix}$$

and

$$B = \begin{pmatrix} -1.54 + 0.76i & 3.17 - 2.09i \\ 0.12 - 1.92i & -6.53 + 4.18i \\ -9.08 - 4.31i & 7.28 + 0.73i \\ 7.49 + 3.65i & 0.91 - 3.97i \\ -5.63 - 2.12i & -5.46 - 1.64i \\ 2.37 + 8.03i & -2.84 - 5.86i \end{pmatrix}.$$

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F08ASF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          MMAX, NMAX, LDA, LDB, NRHMAX, LWORK
      PARAMETER        (MMAX=8,NMAX=8,LDA=MMAX,LDB=MMAX,NRHMAX=NMAX,
     +                 LWORK=64*NMAX)
      complex          ONE
      PARAMETER        (ONE=(1.0e0,0.0e0))
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, M, N, NRHS
```

```
*     .. Local Arrays ..
      complex           A(LDA,NMAX), B(LDB,NRHMAX), TAU(NMAX),
     +                  WORK(LWORK)
      CHARACTER         CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL          X04DBF, cgeqrf, ctrsm, cunmqr
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08ASF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) M, N, NRHS
      IF (M.LE.MMAX .AND. N.LE.NMAX .AND. M.GE.N .AND. NRHS.LE.NRHMAX)
     +    THEN
*
*        Read A and B from data file
*
         READ (NIN,*) ((A(I,J),J=1,N),I=1,M)
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,M)
*
*        Compute the QR factorization of A
*
         CALL cgeqrf(M,N,A,LDA,TAU,WORK,LWORK,INFO)
*
*        Compute C = (Q**H)*B, storing the result in B
*
         CALL cunmqr('Left','Conjugate transpose',M,NRHS,N,A,LDA,TAU,B,
     +               LDB,WORK,LWORK,INFO)
*
*        Compute least-squares solution by backsubstitution in R*X = C
*
         CALL ctrsm('Left','Upper','No transpose','Non-Unit',N,NRHS,ONE,
     +              A,LDA,B,LDB)
*
*        Print least-squares solution(s)
*
         WRITE (NOUT,*)
         IFAIL = 0
*
         CALL X04DBF('General',' ',N,NRHS,B,LDB,'Bracketed','F7.4',
     +               'Least-squares solution(s)','Integer',RLABS,
     +               'Integer',CLABS,80,0,IFAIL)
*
      END IF
      STOP
      END
```

9.2. Program Data

```
F08ASF Example Program Data
  6  4  2                                                  :Values of M, N and NRHS
 ( 0.96,-0.81) (-0.03, 0.96) (-0.91, 2.06) (-0.05, 0.41)
 (-0.98, 1.98) (-1.20, 0.19) (-0.66, 0.42) (-0.81, 0.56)
 ( 0.62,-0.46) ( 1.01, 0.02) ( 0.63,-0.17) (-1.11, 0.60)
 (-0.37, 0.38) ( 0.19,-0.54) (-0.98,-0.36) ( 0.22,-0.20)
 ( 0.83, 0.51) ( 0.20, 0.01) (-0.17,-0.46) ( 1.47, 1.59)
 ( 1.08,-0.28) ( 0.20,-0.12) (-0.07, 1.23) ( 0.26, 0.26)   :End of matrix A
 (-1.54, 0.76) ( 3.17,-2.09)
 ( 0.12,-1.92) (-6.53, 4.18)
 (-9.08,-4.31) ( 7.28, 0.73)
 ( 7.49, 3.65) ( 0.91,-3.97)
 (-5.63,-2.12) (-5.46,-1.64)
 ( 2.37, 8.03) (-2.84,-5.86)                               :End of matrix B
```

9.3. Program Results

```
F08ASF Example Program Results

Least-squares solution(s)
                     1                   2
1  (-0.4936,-1.1993) ( 0.7535, 1.4404)
2  (-2.4708, 2.8373) ( 5.1726,-3.6235)
3  ( 1.5060,-2.1830) (-2.6609, 2.1334)
4  ( 0.4459, 2.6848) (-2.6966, 0.2711)
```

F08ATF (CUNGQR/ZUNGQR) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F08ATF (CUNGQR/ZUNGQR) generates all or part of the complex unitary matrix Q from a QR factorization computed by F08ASF (CGEQRF/ZGEQRF) or F08BSF (CGEQPF/ZGEQPF).

2. Specification

```
SUBROUTINE F08ATF (M, N, K, A, LDA, TAU, WORK, LWORK, INFO)
ENTRY      cungqr (M, N, K, A, LDA, TAU, WORK, LWORK, INFO)
INTEGER        M, N, K, LDA, LWORK, INFO
complex        A(LDA,*), TAU(*), WORK(LWORK)
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine is intended to be used after a call to F08ASF (CGEQRF/ZGEQRF) or F08BSF (CGEQPF/ZGEQPF), which perform a QR factorization of a complex matrix A. F08ASF and F08BSF represent the unitary matrix Q as a product of elementary reflectors.

This routine may be used to generate Q explicitly as a square matrix, or to form only its leading columns.

Usually Q is determined from the QR factorization of an m by p matrix A with $m \geq p$. The whole of Q may be computed by:

```
CALL CUNGQR (M,M,P,A,LDA,TAU,WORK,LWORK,INFO)
```

(note that the array A must have at least m columns) or its leading p columns by:

```
CALL CUNGQR (M,P,P,A,LDA,TAU,WORK,LWORK,INFO)
```

The columns of Q returned by the last call form an orthonormal basis for the space spanned by the columns of A; thus F08ASF followed by F08ATF can be used to orthogonalise the columns of A.

The information returned by the QR factorization routines also yields the QR factorization of the leading k columns of A, where $k < p$. The unitary matrix arising from this factorization can be computed by:

```
CALL CUNGQR (M,M,K,A,LDA,TAU,WORK,LWORK,INFO)
```

or its leading k columns by:

```
CALL CUNGQR (M,K,K,A,LDA,TAU,WORK,LWORK,INFO)
```

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §5.2.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: M – INTEGER. *Input*

On entry: m, the order of the unitary matrix Q.

Constraint: M $\geq$ 0.

2: N – INTEGER. *Input*

On entry: n, the number of columns of matrix Q that are required.

Constraint: M $\geq$ N $\geq$ 0.

3: K – INTEGER. *Input*

On entry: k, the number of elementary reflectors whose product defines the matrix Q.

Constraint: N ≥ K ≥ 0.

4: A(LDA,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array A must be at least max(1,N).

On entry: details of the vectors which define the elementary reflectors, as returned by F08ASF (CGEQRF/ZGEQRF) or F08BSF (CGEQPF/ZGEQPF).

On exit: the m by n matrix Q.

5: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F08ATF (CUNGQR/ZUNGQR) is called.

Constraint: LDA ≥ max(1,M).

6: TAU(*) – ***complex*** array. *Input*

Note: the dimension of the array TAU must be at least max(1,K).

On entry: further details of the elementary reflectors, as returned by F08ASF (CGEQRF/ZGEQRF) or F08BSF (CGEQPF/ZGEQPF).

7: WORK(LWORK) – ***complex*** array. *Workspace*

On exit: if INFO = 0, WORK(1) contains the minimum value of LWORK required for optimum performance.

8: LWORK – INTEGER. *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08ATF (CUNGQR/ZUNGQR) is called.

Suggested value: for optimum performance LWORK should be at least N×nb, where nb is the ***blocksize***.

Constraint: LWORK ≥ max(1,N).

9: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed matrix Q differs from an exactly unitary matrix by a matrix E such that

$$\|E\|_2 = O(\varepsilon),$$

where ε is the ***machine precision***.

8. Further Comments

The total number of real floating-point operations is approximately $16mnk - 8(m+n)k^2 + \frac{16k^3}{3}$; when $n = k$, the number is approximately $\frac{8}{3}n^2(3m-n)$.

The real analogue of this routine is F08AFF (SORGQR/DORGQR).

9. Example

To form the leading 4 columns of the unitary matrix Q from the QR factorization of the matrix A, where

$$A = \begin{pmatrix} 0.96 - 0.81i & -0.03 + 0.96i & -0.91 + 2.06i & -0.05 + 0.41i \\ -0.98 + 1.98i & -1.20 + 0.19i & -0.66 + 0.42i & -0.81 + 0.56i \\ 0.62 - 0.46i & 1.01 + 0.02i & 0.63 - 0.17i & -1.11 + 0.60i \\ -0.37 + 0.38i & 0.19 - 0.54i & -0.98 - 0.36i & 0.22 - 0.20i \\ 0.83 + 0.51i & 0.20 + 0.01i & -0.17 - 0.46i & 1.47 + 1.59i \\ 1.08 - 0.28i & 0.20 - 0.12i & -0.07 + 1.23i & 0.26 + 0.26i \end{pmatrix}$$

The columns of Q form an orthonormal basis for the space spanned by the columns of A.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F08ATF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          MMAX, NMAX, LDA, LWORK
      PARAMETER        (MMAX=8,NMAX=8,LDA=MMAX,LWORK=64*NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, M, N
      CHARACTER*30     TITLE
*     .. Local Arrays ..
      complex          A(LDA,NMAX), TAU(NMAX), WORK(LWORK)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         X04DBF, cgeqrf, cungqr
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08ATF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) M, N
      IF (M.LE.MMAX .AND. N.LE.NMAX .AND. M.GE.N) THEN
*
*        Read A from data file
*
         READ (NIN,*) ((A(I,J),J=1,N),I=1,M)
*
*        Compute the QR factorization of A
*
         CALL cgeqrf(M,N,A,LDA,TAU,WORK,LWORK,INFO)
*
*        Form the leading N columns of Q explicitly
*
         CALL cungqr(M,N,N,A,LDA,TAU,WORK,LWORK,INFO)
*
*        Print the leading N columns of Q only
*
         WRITE (NOUT,*)
         WRITE (TITLE,99999) N
         IFAIL = 0
*
         CALL X04DBF('General',' ',M,N,A,LDA,'Bracketed','F7.4',TITLE,
     +               'Integer',RLABS,'Integer',CLABS,80,0,IFAIL)
*
      END IF
      STOP
*
99999 FORMAT ('The leading ',I2,' columns of Q')
      END
```

9.2. Program Data

```
F08ATF Example Program Data
  6  4                                                        :Values of M and N
 ( 0.96,-0.81) (-0.03, 0.96) (-0.91, 2.06) (-0.05, 0.41)
 (-0.98, 1.98) (-1.20, 0.19) (-0.66, 0.42) (-0.81, 0.56)
 ( 0.62,-0.46) ( 1.01, 0.02) ( 0.63,-0.17) (-1.11, 0.60)
 (-0.37, 0.38) ( 0.19,-0.54) (-0.98,-0.36) ( 0.22,-0.20)
 ( 0.83, 0.51) ( 0.20, 0.01) (-0.17,-0.46) ( 1.47, 1.59)
 ( 1.08,-0.28) ( 0.20,-0.12) (-0.07, 1.23) ( 0.26, 0.26)   :End of matrix A
```

9.3. Program Results

```
F08ATF Example Program Results

The leading  4 columns of Q
                      1                     2                     3                     4
1  (-0.3110, 0.2624) (-0.3175, 0.4835) ( 0.4966,-0.2997) (-0.0072,-0.3718)
2  ( 0.3175,-0.6414) (-0.2062, 0.1577) (-0.0793,-0.3094) (-0.0282,-0.1491)
3  (-0.2008, 0.1490) ( 0.4892,-0.0900) ( 0.0357,-0.0219) ( 0.5625,-0.0710)
4  ( 0.1199,-0.1231) ( 0.2566,-0.3055) ( 0.4489,-0.2141) (-0.1651, 0.1800)
5  (-0.2689,-0.1652) ( 0.1697,-0.2491) (-0.0496, 0.1158) (-0.4885,-0.4540)
6  (-0.3499, 0.0907) (-0.0491,-0.3133) (-0.1256,-0.5300) ( 0.1039, 0.0450)
```

F08AUF (CUNMQR/ZUNMQR) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F08AUF (CUNMQR/ZUNMQR) multiplies an arbitrary complex matrix C by the complex unitary matrix Q from a QR factorization computed by F08ASF (CGEQRF/ZGEQRF) or F08BSF (CGEQPF/ZGEQPF).

2. Specification

```
      SUBROUTINE F08AUF (SIDE, TRANS, M, N, K, A, LDA, TAU, C, LDC, WORK,
     1                   LWORK, INFO)
      ENTRY       cunmqr (SIDE, TRANS, M, N, K, A, LDA, TAU, C, LDC, WORK,
     1                   LWORK, INFO)
      INTEGER          M, N, K, LDA, LDC, LWORK, INFO
      complex          A(LDA,*), TAU(*), C(LDC,*), WORK(LWORK)
      CHARACTER*1      SIDE, TRANS
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine is intended to be used after a call to F08ASF (CGEQRF/ZGEQRF) or F08BSF (CGEQPF/ZGEQPF), which perform a QR factorization of a complex matrix A. F08ASF and F08BSF represent the unitary matrix Q as a product of elementary reflectors.

This routine may be used to form one of the matrix products

$$QC,\ Q^HC,\ CQ \text{ or } CQ^H,$$

overwriting the result on C (which may be any complex rectangular matrix).

A common application of this routine is in solving linear least-squares problems, as described in the Chapter Introduction, and illustrated in Section 9 of the document for F08ASF.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: SIDE – CHARACTER*1. *Input*

On entry: indicates how Q or Q^H is to be applied to C as follows:

if SIDE = 'L', then Q or Q^H is applied to C from the left;

if SIDE = 'R', then Q or Q^H is applied to C from the right.

Constraint: SIDE = 'L' or 'R'.

2: TRANS – CHARACTER*1. *Input*

On entry: indicates whether Q or Q^H is to be applied to C as follows:

if TRANS = 'N', then Q is applied to C;

if TRANS = 'C', then Q^H is applied to C.

Constraint: TRANS = 'N' or 'C'.

3: M – INTEGER. *Input*

On entry: m, the number of rows of the matrix C.

Constraint: M ≥ 0.

4: N – INTEGER. *Input*

On entry: n, the number of columns of the matrix C.

Constraint: N ≥ 0.

5: K – INTEGER. *Input*

On entry: k, the number of elementary reflectors whose product defines the matrix Q.

Constraints: M ≥ K ≥ 0 if SIDE = 'L',
N ≥ K ≥ 0 if SIDE = 'R'.

6: A(LDA,*) – ***complex*** array. *Input*

Note: the second dimension of the array A must be at least max(1,K).

On entry: details of the vectors which define the elementary reflectors, as returned by F08ASF (CGEQRF/ZGEQRF) or F08BSF (CGEQPF/ZGEQPF).

7: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F08AUF (CUNMQR/ZUNMQR) is called.

Constraints: LDA ≥ max(1,M) if SIDE = 'L',
LDA ≥ max(1,N) if SIDE = 'R'.

8: TAU(*) – ***complex*** array. *Input*

Note: the dimension of the array TAU must be at least max(1,K).

On entry: further details of the elementary reflectors, as returned by F08ASF (CGEQRF/ZGEQRF) or F08BSF (CGEQPF/ZGEQPF).

9: C(LDC,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array C must be at least max(1,N).

On entry: the m by n matrix C.

On exit: C is overwritten by QC or Q^HC or CQ^H or CQ as specified by SIDE and TRANS.

10: LDC – INTEGER. *Input*

On entry: the first dimension of the array C as declared in the (sub)program from which F08AUF (CUNMQR/ZUNMQR) is called.

Constraint: LDC ≥ max(1,M).

11: WORK(LWORK) – ***complex*** array. *Workspace*

On exit: if INFO = 0, WORK(1) contains the minimum value of LWORK required for optimum performance.

12: LWORK – INTEGER. *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08AUF (CUNMQR/ZUNMQR) is called.

Suggested value: for optimum performance LWORK should be at least N×nb if SIDE = 'L' and at least M×nb if SIDE = 'R', where nb is the ***blocksize***.

Constraints: LWORK ≥ max(1,N) if SIDE = 'L',
LWORK ≥ max(1,M) if SIDE = 'R'.

13: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed result differs from the exact result by a matrix E such that

$$\|E\|_2 = O(\varepsilon)\|C\|_2,$$

where ε is the ***machine precision***.

8. Further Comments

The total number of real floating-point operations is approximately $8nk(2m-k)$ if SIDE = 'L' and $8mk(2n-k)$ if SIDE = 'R'.

The real analogue of this routine is F08AGF (SORMQR/DORMQR).

9. Example

See the example for F08ASF (CGEQRF/ZGEQRF).

F08AVF (CGELQF/ZGELQF) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F08AVF (CGELQF/ZGELQF) computes the LQ factorization of a complex m by n matrix.

2. Specification

```
SUBROUTINE F08AVF (M, N, A, LDA, TAU, WORK, LWORK, INFO)
ENTRY      cgelqf  (M, N, A, LDA, TAU, WORK, LWORK, INFO)
INTEGER        M, N, LDA, LWORK, INFO
complex        A(LDA,*), TAU(*), WORK(LWORK)
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine forms the LQ factorization of an arbitrary rectangular complex m by n matrix. No pivoting is performed.

If $m \leq n$, the factorization is given by:

$$A = (L \ 0)Q$$

where L is an m by m lower triangular matrix (with real diagonal elements) and Q is an n by n unitary matrix. It is sometimes more convenient to write the factorization as

$$A = (L \ 0)\begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix}$$

which reduces to

$$A = LQ_1,$$

where Q_1 consists of the first m rows of Q, and Q_2 the remaining $n-m$ rows.

If $m > n$, L is trapezoidal, and the factorization can be written

$$A = \begin{pmatrix} L_1 \\ L_2 \end{pmatrix}Q$$

where L_1 is lower triangular and L_2 is rectangular.

The LQ factorization of A is essentially the same as the QR factorization of A^H, since

$$A = (L \ 0)Q \Leftrightarrow A^H = Q^H\begin{pmatrix} L^H \\ 0 \end{pmatrix}.$$

The matrix Q is not formed explicitly but is represented as a product of $\min(m,n)$ elementary reflectors (see the Chapter Introduction for details). Routines are provided to work with Q in this representation (see Section 8).

Note also that for any $k < m$, the information returned in the first k rows of the array A represents an LQ factorization of the first k rows of the original matrix A.

4. References

None.

5. Parameters

1: M – INTEGER. *Input*

On entry: m, the number of rows of the matrix A.

Constraint: M $\geq$ 0.

2: N – INTEGER. *Input*

On entry: n, the number of columns of the matrix A.

Constraint: N $\geq$ 0.

3: A(LDA,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the m by n matrix A.

On exit: if $m \leq n$, the elements above the diagonal are overwritten by details of the unitary matrix Q and the lower triangle is overwritten by the corresponding elements of the m by m lower triangular matrix L.

If $m > n$, the strictly upper triangular part is overwritten by details of the unitary matrix Q and the remaining elements are overwritten by the corresponding elements of the m by n lower trapezoidal matrix L.

The diagonal elements of L are real.

4: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F08AVF (CGELQF/ZGELQF) is called.

Constraint: LDA $\geq$ max(1,M).

5: TAU(*) – ***complex*** array. *Output*

Note: the dimension of the array TAU must be at least max(1,min(M,N)).

On exit: further details of the unitary matrix Q.

6: WORK(LWORK) – ***complex*** array. *Workspace*

On exit: if INFO = 0, WORK(1) contains the minimum value of LWORK required for optimum performance.

7: LWORK – INTEGER. *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08AVF (CGELQF/ZGELQF) is called.

Suggested value: for optimum performance LWORK should be at least M$\times nb$, where nb is the ***blocksize***.

Constraint: LWORK $\geq$ max(1,M).

8: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed factorization is the exact factorization of a nearby matrix $A + E$, where

$$\|E\|_2 = O(\varepsilon)\|A\|_2,$$

and ε is the ***machine precision***.

8. Further Comments

The total number of real floating-point operations is approximately $\frac{8}{3}m^2(3n-m)$ if $m \leq n$ or $\frac{8}{3}n^2(3m-n)$ if $m > n$.

To form the unitary matrix Q this routine may be followed by a call to F08AWF (CUNGLQ/ZUNGLQ):

```
CALL CUNGLQ (N,N,MIN(M,N),A,LDA,TAU,WORK,LWORK,INFO)
```

but note that the first dimension of the array A, specified by the parameter LDA, must be at least N, which may be larger than was required by F08AVF.

When $m \leq n$, it is often only the first m rows of Q that are required, and they may be formed by the call:

```
CALL CUNGLQ (M,N,M,A,LDA,TAU,WORK,LWORK,INFO)
```

To apply Q to an arbitrary complex rectangular matrix C, this routine may be followed by a call to F08AXF (CUNMLQ/ZUNMLQ). For example,

```
CALL CUNMLQ ('Left','Conjugate Transpose',M,P,MIN(M,N),A,LDA,TAU,
+            C,LDC,WORK,LWORK,INFO)
```

forms the matrix product $C = Q^H C$, where C is m by p.

The real analogue of this routine is F08AHF (SGELQF/DGELQF).

9. Example

To find the minimum-norm solutions of the under-determined systems of linear equations

$$Ax_1 = b_1 \text{ and } Ax_2 = b_2$$

where b_1 and b_2 are the columns of the matrix B,

$$A = \begin{pmatrix} 0.28 - 0.36i & 0.50 - 0.86i & -0.77 - 0.48i & 1.58 + 0.66i \\ -0.50 - 1.10i & -1.21 + 0.76i & -0.32 - 0.24i & -0.27 - 1.15i \\ 0.36 - 0.51i & -0.07 + 1.33i & -0.75 + 0.47i & -0.08 + 1.01i \end{pmatrix}$$

and

$$B = \begin{pmatrix} -1.35 + 0.19i & 4.83 - 2.67i \\ 9.41 - 3.56i & -7.28 + 3.34i \\ -7.57 + 6.93i & 0.62 + 4.53i \end{pmatrix}.$$

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F08AVF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          MMAX, NMAX, LDA, LDB, NRHMAX, LWORK
      PARAMETER        (MMAX=8,NMAX=8,LDA=MMAX,LDB=NMAX,NRHMAX=NMAX,
     +                 LWORK=64*NMAX)
      complex          ZERO, ONE
      PARAMETER        (ZERO=(0.0e0,0.0e0),ONE=(1.0e0, 0.0e0))
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, M, N, NRHS
*     .. Local Arrays ..
      complex          A(LDA,NMAX), B(LDB,NRHMAX), TAU(NMAX),
     +                 WORK(LWORK)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         F06THF, X04DBF, cgelqf, ctrsm, cunmlq
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08AVF Example Program Results'
```

```
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) M, N, NRHS
      IF (M.LE.MMAX .AND. N.LE.NMAX .AND. M.LE.N .AND. NRHS.LE.NRHMAX)
     +    THEN
*
*        Read A and B from data file
*
         READ (NIN,*) ((A(I,J),J=1,N),I=1,M)
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,M)
*
*        Compute the LQ factorization of A
*
         CALL cgelqf(M,N,A,LDA,TAU,WORK,LWORK,INFO)
*
*        Solve L*Y = B, storing the result in B
*
         CALL ctrsm('Left','Lower','No transpose','Non-Unit',M,NRHS,ONE,
     +              A,LDA,B,LDB)
*
*        Set rows (M+1) to N of B to zero
*
         IF (M.LT.N) CALL F06THF('General',N-M,NRHS,ZERO,ZERO,B(M+1,1),
     +                           LDB)
*
*        Compute minimum-norm solution X = (Q**H)*B in B
*
         CALL cunmlq('Left','Conjugate transpose',N,NRHS,M,A,LDA,TAU,B,
     +               LDB,WORK,LWORK,INFO)
*
*        Print minimum-norm solution(s)
*
         WRITE (NOUT,*)
         IFAIL = 0
*
         CALL X04DBF('General',' ',N,NRHS,B,LDB,'Bracketed','F7.4',
     +               'Minimum-norm solution(s)','Integer',RLABS,
     +               'Integer',CLABS,80,0,IFAIL)
*
      END IF
      STOP
      END
```

9.2. Program Data

```
F08AVF Example Program Data
  3  4  2                                                  :Values of M, N and NRHS
 ( 0.28,-0.36) ( 0.50,-0.86) (-0.77,-0.48) ( 1.58, 0.66)
 (-0.50,-1.10) (-1.21, 0.76) (-0.32,-0.24) (-0.27,-1.15)
 ( 0.36,-0.51) (-0.07, 1.33) (-0.75, 0.47) (-0.08, 1.01)    :End of matrix A
 (-1.35, 0.19) ( 4.83,-2.67)
 ( 9.41,-3.56) (-7.28, 3.34)
 (-7.57, 6.93) ( 0.62, 4.53)                                :End of matrix B
```

9.3. Program Results

```
 F08AVF Example Program Results

 Minimum-norm solution(s)
                      1                   2
 1  (-2.8501, 6.4683) (-1.1682,-1.8886)
 2  ( 1.6264,-0.7799) ( 2.8377, 0.7654)
 3  ( 6.9290, 4.6481) (-1.7610,-0.7041)
 4  ( 1.4048, 3.2400) ( 1.0518,-1.6365)
```

F08AWF (CUNGLQ/ZUNGLQ) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F08AWF (CUNGLQ/ZUNGLQ) generates all or part of the complex unitary matrix Q from an LQ factorization computed by F08AVF (CGELQF/ZGELQF).

2. Specification

```
SUBROUTINE F08AWF (M, N, K, A, LDA, TAU, WORK, LWORK, INFO)
ENTRY      cunglq  (M, N, K, A, LDA, TAU, WORK, LWORK, INFO)
INTEGER        M, N, K, LDA, LWORK, INFO
complex        A(LDA,*), TAU(*), WORK(LWORK)
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine is intended to be used after a call to F08AVF (CGELQF/ZGELQF), which performs an LQ factorization of a complex matrix A. F08AVF represents the unitary matrix Q as a product of elementary reflectors.

This routine may be used to generate Q explicitly as a square matrix, or to form only its leading rows.

Usually Q is determined from the LQ factorization of a p by n matrix A with $p \leq n$. The whole of Q may be computed by:

```
CALL CUNGLQ (N,N,P,A,LDA,TAU,WORK,LWORK,INFO)
```

(note that the array A must have at least n rows) or its leading p rows by:

```
CALL CUNGLQ (P,N,P,A,LDA,TAU,WORK,LWORK,INFO)
```

The rows of Q returned by the last call form an orthonormal basis for the space spanned by the rows of A; thus F08AVF followed by F08AWF can be used to orthogonalise the rows of A.

The information returned by the LQ factorization routines also yields the LQ factorization of the leading k rows of A, where $k < p$. The unitary matrix arising from this factorization can be computed by:

```
CALL CUNGLQ (N,N,K,A,LDA,TAU,WORK,LWORK,INFO)
```

or its leading k rows by:

```
CALL CUNGLQ (K,N,K,A,LDA,TAU,WORK,LWORK,INFO)
```

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: M – INTEGER. *Input*

On entry: m, the number of rows of the matrix Q.

Constraint: M $\geq$ 0.

2: N – INTEGER. *Input*

On entry: n, the number of columns of the matrix Q.

Constraint: N $\geq$ M.

3: K – INTEGER. *Input*

On entry: k, the number of elementary reflectors whose product defines the matrix Q.

Constraint: $M \geq K \geq 0$.

4: A(LDA,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array A must be at least max(1,N).

On entry: details of the vectors which define the elementary reflectors, as returned by F08AVF (CGELQF/ZGELQF).

On exit: the m by n matrix Q.

5: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F08AWF (CUNGLQ/ZUNGLQ) is called.

Constraint: LDA ≥ max(1,M).

6: TAU(*) – ***complex*** array. *Input*

Note: the dimension of the array TAU must be at least max(1,K).

On entry: further details of the elementary reflectors, as returned by F08AVF (CGELQF/ZGELQF).

7: WORK(LWORK) – ***complex*** array. *Workspace*

On exit: if INFO = 0, WORK(1) contains the minimum value of LWORK required for optimum performance.

8: LWORK – INTEGER. *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08AWF (CUNGLQ/ZUNGLQ) is called.

Suggested value: for optimum performance LWORK should be at least M×*nb*, where *nb* is the ***blocksize***.

Constraint: LWORK ≥ max(1,M).

9: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed matrix Q differs from an exactly unitary matrix by a matrix E such that

$$\|E\|_2 = O(\varepsilon),$$

where ε is the ***machine precision***.

8. Further Comments

The total number of real floating-point operations is approximately $16mnk - 8(m+n)k^2 + \frac{16k^3}{3}$; when $m = k$, the number is approximately $\frac{8}{3}m^2(3n-m)$.

The real analogue of this routine is F08AJF (SORGLQ/DORGLQ).

9. Example

To form the leading 4 rows of the unitary matrix Q from the LQ factorization of the matrix A, where

$$A = \begin{pmatrix} 0.28 - 0.36i & 0.50 - 0.86i & -0.77 - 0.48i & 1.58 + 0.66i \\ -0.50 - 1.10i & -1.21 + 0.76i & -0.32 - 0.24i & -0.27 - 1.15i \\ 0.36 - 0.51i & -0.07 + 1.33i & -0.75 + 0.47i & -0.08 + 1.01i \end{pmatrix}.$$

The rows of Q form an orthonormal basis for the space spanned by the rows of A.

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F08AWF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          MMAX, NMAX, LDA, LWORK
      PARAMETER        (MMAX=8,NMAX=8,LDA=MMAX,LWORK=64*MMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, M, N
      CHARACTER*30     TITLE
*     .. Local Arrays ..
      complex          A(LDA,NMAX), TAU(NMAX), WORK(LWORK)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         X04DBF, cgelqf, cunglq
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08AWF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) M, N
      IF (M.LE.MMAX .AND. N.LE.NMAX .AND. M.LE.N) THEN
*
*        Read A from data file
*
         READ (NIN,*) ((A(I,J),J=1,N),I=1,M)
*
*        Compute the LQ factorization of A
*
         CALL cgelqf(M,N,A,LDA,TAU,WORK,LWORK,INFO)
*
*        Form the leading M rows of Q explicitly
*
         CALL cunglq(M,N,M,A,LDA,TAU,WORK,LWORK,INFO)
*
*        Print the leading M rows of Q only
*
         WRITE (NOUT,*)
         WRITE (TITLE,99999) M
         IFAIL = 0
*
         CALL X04DBF('General',' ',M,N,A,LDA,'Bracketed','F7.4',TITLE,
     +               'Integer',RLABS,'Integer',CLABS,80,0,IFAIL)
*
      END IF
      STOP
*
99999 FORMAT ('The leading ',I2,' rows of Q')
      END
```

9.2. Program Data

```
F08AWF Example Program Data
  3  4                                                        :Values of M and N
 ( 0.28,-0.36) ( 0.50,-0.86) (-0.77,-0.48) ( 1.58, 0.66)
 (-0.50,-1.10) (-1.21, 0.76) (-0.32,-0.24) (-0.27,-1.15)
 ( 0.36,-0.51) (-0.07, 1.33) (-0.75, 0.47) (-0.08, 1.01)   :End of matrix A
```

9.3. Program Results

```
F08AWF Example Program Results

The leading  3 rows of Q
                      1                   2                   3                   4
1  (-0.1258, 0.1618) (-0.2247, 0.3864) ( 0.3460, 0.2157) (-0.7099,-0.2966)
2  (-0.1163,-0.6380) (-0.3240, 0.4272) (-0.1995,-0.5009) (-0.0323,-0.0162)
3  (-0.4607, 0.1090) ( 0.2171,-0.4062) ( 0.2733,-0.6106) (-0.0994,-0.3261)
```

F08AXF (CUNMLQ/ZUNMLQ) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F08AXF (CUNMLQ/ZUNMLQ) multiplies an arbitrary complex matrix C by the complex unitary matrix Q from an LQ factorization computed by F08AVF (CGELQF/ZGELQF).

2. Specification

```
 SUBROUTINE F08AXF (SIDE, TRANS, M, N, K, A, LDA, TAU, C, LDC, WORK,
1                   LWORK, INFO)
 ENTRY       cunmlq (SIDE, TRANS, M, N, K, A, LDA, TAU, C, LDC, WORK,
1                   LWORK, INFO)
 INTEGER          M, N, K, LDA, LDC, LWORK, INFO
 complex          A(LDA,*), TAU(*), C(LDC,*), WORK(LWORK)
 CHARACTER*1      SIDE, TRANS
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine is intended to be used after a call to F08AVF (CGELQF/ZGELQF), which performs an LQ factorization of a complex matrix A. F08AVF represents the unitary matrix Q as a product of elementary reflectors.

This routine may be used to form one of the matrix products

QC, Q^HC, CQ or CQ^H,

overwriting the result on C (which may be any complex rectangular matrix).

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: SIDE – CHARACTER*1. *Input*

On entry: indicates how Q or Q^H is to be applied to C as follows:

if SIDE = 'L', then Q or Q^H is applied to C from the left;

if SIDE = 'R', then Q or Q^H is applied to C from the right.

Constraint: SIDE = 'L' or 'R'.

2: TRANS – CHARACTER*1. *Input*

On entry: indicates whether Q or Q^H is to be applied to C as follows:

if TRANS = 'N', then Q is applied to C;

if TRANS = 'C', then Q^H is applied to C.

Constraint: TRANS = 'N' or 'C'.

3: M – INTEGER. *Input*

On entry: m, the number of rows of the matrix C.

Constraint: M $\geq$ 0.

4: N – INTEGER. *Input*

On entry: n, the number of columns of the matrix C.

Constraint: N ≥ 0.

5: K – INTEGER. *Input*

On entry: k, the number of elementary reflectors whose product defines the matrix Q.

Constraints: M ≥ K ≥ 0 if SIDE = 'L',
N ≥ K ≥ 0 if SIDE = 'R'.

6: A(LDA,*) – ***complex*** array. *Input*

Note: the second dimension of the array A must be at least max(1,M) if SIDE = 'L' and at least max(1,N) if SIDE = 'R'.

On entry: details of the vectors which define the elementary reflectors, as returned by F08AVF (CGELQF/ZGELQF).

7: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F08AXF (CUNMLQ/ZUNMLQ) is called.

Constraint: LDA ≥ max(1,K).

8: TAU(*) – ***complex*** array. *Input*

Note: the dimension of the array TAU must be at least max(1,K).

On entry: further details of the elementary reflectors, as returned by F08AVF (CGELQF/ZGELQF).

9: C(LDC,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array C must be at least max(1,N).

On entry: the m by n matrix C.

On exit: C is overwritten by QC or Q^HC or CQ^H or CQ as specified by SIDE and TRANS.

10: LDC – INTEGER. *Input*

On entry: the first dimension of the array C as declared in the (sub)program from which F08AXF (CUNMLQ/ZUNMLQ) is called.

Constraint: LDC ≥ max(1,M).

11: WORK(LWORK) – ***complex*** array. *Workspace*

On exit: if INFO = 0, WORK(1) contains the minimum value of LWORK required for optimum performance.

12: LWORK – INTEGER. *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08AXF (CUNMLQ/ZUNMLQ) is called.

Suggested value: for optimum performance LWORK should be at least N×nb if SIDE = 'L' and at least M×nb if SIDE = 'R', where nb is the ***blocksize***.

Constraints: LWORK ≥ max(1,N) if SIDE = 'L',
LWORK ≥ max(1,M) if SIDE = 'R'.

13: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed result differs from the exact result by a matrix E such that

$$\|E\|_2 = O(\varepsilon)\|C\|_2,$$

where ε is the ***machine precision***.

8. Further Comments

The total number of real floating-point operations is approximately $8nk(2m-k)$ if SIDE = 'L' and $8mk(2n-k)$ if SIDE = 'R'.

The real analogue of this routine is F08AKF (SORMLQ/DORMLQ).

9. Example

See the example for F08AVF (CGELQF/ZGELQF).

F08BEF (SGEQPF/DGEQPF) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F08BEF (SGEQPF/DGEQPF) computes the QR factorization, with column pivoting, of a real m by n matrix.

2. Specification

```
SUBROUTINE F08BEF (M, N, A, LDA, JPVT, TAU, WORK, INFO)
ENTRY      sgeqpf  (M, N, A, LDA, JPVT, TAU, WORK, INFO)
INTEGER        M, N, LDA, JPVT(*), INFO
real           A(LDA,*), TAU(*), WORK(*)
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine forms the QR factorization with column pivoting of an arbitrary rectangular real m by n matrix.

If $m \geq n$, the factorization is given by:

$$AP = Q\begin{pmatrix} R \\ 0 \end{pmatrix}$$

where R is an n by n upper triangular matrix, Q is an m by m orthogonal matrix and P is an n by n permutation matrix. It is sometimes more convenient to write the factorization as

$$AP = (Q_1 \ Q_2)\begin{pmatrix} R \\ 0 \end{pmatrix}$$

which reduces to

$$AP = Q_1 R,$$

where Q_1 consists of the first n columns of Q, and Q_2 the remaining $m-n$ columns.

If $m < n$, R is trapezoidal, and the factorization can be written

$$AP = Q(R_1 \ R_2),$$

where R_1 is upper triangular and R_2 is rectangular.

The matrix Q is not formed explicitly but is represented as a product of $\min(m,n)$ elementary reflectors (see the Chapter Introduction for details). Routines are provided to work with Q in this representation (see Section 8).

Note also that for any $k < n$, the information returned in the first k columns of the array A represents a QR factorization of the first k columns of the permuted matrix AP.

The routine allows specified columns of A to be moved to the leading columns of AP at the start of the factorization and fixed there. The remaining columns are free to be interchanged so that at the ith stage the pivot column is chosen to be the column which maximizes the 2-norm of elements i to m over columns i to n.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §5.4.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: M – INTEGER. *Input*

On entry: m, the number of rows of the matrix A.

Constraint: M $\geq$ 0.

2: N – INTEGER. *Input*

On entry: n, the number of columns of the matrix A.

Constraint: N $\geq$ 0.

3: A(LDA,*) – ***real*** array. *Input/Output*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the m by n matrix A.

On exit: if $m \geq n$, the elements below the diagonal are overwritten by details of the orthogonal matrix Q and the upper triangle is overwritten by the corresponding elements of the n by n upper triangular matrix R.

If $m < n$, the strictly lower triangular part is overwritten by details of the orthogonal matrix Q and the remaining elements are overwritten by the corresponding elements of the m by n upper trapezoidal matrix R.

4: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F08BEF (SGEQPF/DGEQPF) is called.

Constraint: LDA $\geq$ max(1,M).

5: JPVT(*) – INTEGER array. *Input/Output*

Note: the dimension of the array JPVT must be at least max(1,N).

On entry: if JPVT(i) $\neq$ 0, then the ith column of A is moved to the beginning of AP before the decomposition is computed and is fixed in place during the computation. Otherwise, the ith column of A is a free column (i.e. one which may be interchanged during the computation with any other free column).

On exit: details of the permutation matrix P. More precisely, if JPVT(i) = k, then the kth column of A is moved to become the ith column of AP; in other words, the columns of AP are the columns of A in the order JPVT(1),JPVT(2),...,JPVT(n).

6: TAU(*) – ***real*** array. *Output*

Note: the dimension of the array TAU must be at least max(1,min(M,N)).

On exit: further details of the orthogonal matrix Q.

7: WORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,3*N).

8: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed factorization is the exact factorization of a nearby matrix $A + E$, where

$$\|E\|_2 = O(\varepsilon)\|A\|_2,$$

and ε is the ***machine precision***.

8. Further Comments

The total number of floating-point operations is approximately $\frac{2}{3}n^2(3m-n)$ if $m \geq n$ or $\frac{2}{3}m^2(3n-m)$ if $m < n$.

To form the orthogonal matrix Q this routine may be followed by a call to F08AFF (SORGQR/DORGQR):

```
CALL SORGQR (M,M,MIN(M,N),A,LDA,TAU,WORK,LWORK,INFO)
```

but note that the second dimension of the array A must be at least M, which may be larger than was required by F08BEF.

When $m \geq n$, it is often only the first n columns of Q that are required, and they may be formed by the call:

```
CALL SORGQR (M,N,N,A,LDA,TAU,WORK,LWORK,INFO)
```

To apply Q to an arbitrary real rectangular matrix C, this routine may be followed by a call to F08AGF (SORMQR/DORMQR). For example,

```
CALL SORMQR ('Left','Transpose',M,P,MIN(M,N),A,LDA,TAU,C,LDC,WORK,
+            LWORK,INFO)
```

forms $C = Q^T C$, where C is m by p.

To compute a QR factorization without column pivoting, use F08AEF (SGEQRF/DGEQRF).

The complex analogue of this routine is F08BSF (CGEQPF/ZGEQPF).

9. Example

To solve the linear least-squares problem

$$\text{minimize } \|Ax_i - b_i\|_2 \text{ for } i = 1,2$$

where b_1 and b_2 are the columns of the matrix B,

$$\text{where } A = \begin{pmatrix} -0.09 & 0.14 & -0.46 & 0.68 & 1.29 \\ -1.56 & 0.20 & 0.29 & 1.09 & 0.51 \\ -1.48 & -0.43 & 0.89 & -0.71 & -0.96 \\ -1.09 & 0.84 & 0.77 & 2.11 & -1.27 \\ 0.08 & 0.55 & -1.13 & 0.14 & 1.74 \\ -1.59 & -0.72 & 1.06 & 1.24 & 0.34 \end{pmatrix} \text{ and } B = \begin{pmatrix} -0.01 & -0.04 \\ 0.04 & -0.03 \\ 0.05 & 0.01 \\ -0.03 & -0.02 \\ 0.02 & 0.05 \\ -0.06 & 0.07 \end{pmatrix}.$$

Here A is approximately rank-deficient, and hence it is preferable to use F08BEF (SGEQPF/DGEQPF) rather than F08AEF (SGEQRF/DGEQRF).

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F08BEF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          MMAX, NMAX, LDA, LDB, LDX, NRHMAX, LWORK
      PARAMETER        (MMAX=8,NMAX=8,LDA=MMAX,LDB=MMAX,LDX=MMAX,
     +                 NRHMAX=NMAX,LWORK=64*NMAX)
      real             ZERO
      PARAMETER        (ZERO=0.0e0)
```

```
*     .. Local Scalars ..
      real               TOL
      INTEGER            I, IFAIL, INFO, J, K, M, N, NRHS
*     .. Local Arrays ..
      real               A(LDA,NMAX), B(LDB,NRHMAX), TAU(NMAX),
     +                   WORK(LWORK), X(LDX,NRHMAX)
      INTEGER            JPVT(NMAX)
*     .. External Subroutines ..
      EXTERNAL           sgeqpf, sormqr, strsv, F06DBF, F06FBF, X04CAF
*     .. Intrinsic Functions ..
      INTRINSIC          ABS
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08BEF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) M, N, NRHS
      IF (M.LE.MMAX .AND. N.LE.NMAX .AND. M.GE.N .AND. NRHS.LE.NRHMAX)
     +    THEN
*
*        Read A and B from data file
*
         READ (NIN,*) ((A(I,J),J=1,N),I=1,M)
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,M)
*
*        Initialize JPVT to be zero so that all columns are free
*
         CALL F06DBF(N,0,JPVT,1)
*
*        Compute the QR factorization of A
*
         CALL sgeqpf(M,N,A,LDA,JPVT,TAU,WORK,INFO)
*
*        Choose TOL to reflect the relative accuracy of the input data
*
         TOL = 0.01e0
*
*        Determine which columns of R to use
*
         DO 20 K = 1, N
            IF (ABS(A(K,K)).LE.TOL*ABS(A(1,1))) GO TO 40
   20    CONTINUE
*
*        Compute C = (Q**T)*B, storing the result in B
*
   40    K = K - 1
*
         CALL sormqr('Left','Transpose',M,NRHS,N,A,LDA,TAU,B,LDB,WORK,
     +               LWORK,INFO)
*
*        Compute least-squares solution by backsubstitution in R*B = C
*
         DO 60 I = 1, NRHS
*
            CALL strsv('Upper','No transpose','Non-Unit',K,A,LDA,B(1,I),
     +                 1)
*
*           Set the unused elements of the I-th solution vector to zero
*
            CALL F06FBF(N-K,ZERO,B(K+1,I),1)
*
   60    CONTINUE
*
*        Unscramble the least-squares solution stored in B
*
         DO 100 I = 1, N
            DO 80 J = 1, NRHS
               X(JPVT(I),J) = B(I,J)
   80       CONTINUE
  100    CONTINUE
*
```

```
*        Print least-squares solution
*
         WRITE (NOUT,*)
         IFAIL = 0
*
         CALL X04CAF('General',' ',N,NRHS,X,LDX,
     +               'Least-squares solution',IFAIL)
*
       END IF
       STOP
       END
```

9.2. Program Data

```
F08BEF Example Program Data
  6  5  2                                   :Values of M, N and NRHS
 -0.09   0.14  -0.46   0.68   1.29
 -1.56   0.20   0.29   1.09   0.51
 -1.48  -0.43   0.89  -0.71  -0.96
 -1.09   0.84   0.77   2.11  -1.27
  0.08   0.55  -1.13   0.14   1.74
 -1.59  -0.72   1.06   1.24   0.34   :End of matrix A
 -0.01  -0.04
  0.04  -0.03
  0.05   0.01
 -0.03  -0.02
  0.02   0.05
 -0.06   0.07                           :End of matrix B
```

9.3. Program Results

```
F08BEF Example Program Results

Least-squares solution
          1       2
1  -0.0370 -0.0044
2   0.0647 -0.0335
3   0.0000  0.0000
4  -0.0515  0.0018
5   0.0066  0.0102
```

F08BSF (CGEQPF/ZGEQPF) – NAG Fortran Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details. The routine name may be precision-dependent.

1. Purpose

F08BSF (CGEQPF/ZGEQPF) computes the *QR* factorization, with column pivoting, of a complex m by n matrix.

2. Specification

```
SUBROUTINE F08BSF (M, N, A, LDA, JPVT, TAU, WORK, RWORK, INFO)
ENTRY      cgeqpf (M, N, A, LDA, JPVT, TAU, WORK, RWORK, INFO)
INTEGER          M, N, LDA, JPVT(*), INFO
real             RWORK (*)
complex          A(LDA,*), TAU(*), WORK(*)
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3. Description

This routine forms the *QR* factorization with column pivoting of an arbitrary rectangular complex m by n matrix.

If $m \geq n$, the factorization is given by:

$$AP = Q\begin{pmatrix} R \\ 0 \end{pmatrix}$$

where R is an n by n upper triangular matrix (with real diagonal elements), Q is an m by m unitary matrix and P is an n by n permutation matrix. It is sometimes more convenient to write the factorization as

$$AP = (Q_1 \ Q_2)\begin{pmatrix} R \\ 0 \end{pmatrix}$$

which reduces to

$$AP = Q_1 R,$$

where Q_1 consists of the first n columns of Q, and Q_2 the remaining $m-n$ columns.

If $m < n$, R is trapezoidal, and the factorization can be written

$$AP = Q(R_1 \ R_2),$$

where R_1 is upper triangular and R_2 is rectangular.

The matrix Q is not formed explicitly but is represented as a product of $\min(m,n)$ elementary reflectors (see the Chapter Introduction for details). Routines are provided to work with Q in this representation (see Section 8).

Note also that for any $k < n$, the information returned in the first k columns of the array A represents a *QR* factorization of the first k columns of the permuted matrix AP.

The routine allows specified columns of A to be moved to the leading columns of AP at the start of the factorization and fixed there. The remaining columns are free to be interchanged so that at the ith stage the pivot column is chosen to be the column which maximizes the 2-norm of elements i to m over columns i to n.

4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §5.4.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

5. Parameters

1: M – INTEGER. *Input*

On entry: m, the number of rows of the matrix A.

Constraint: M $\geq$ 0.

2: N – INTEGER. *Input*

On entry: n, the number of columns of the matrix A.

Constraint: N $\geq$ 0.

3: A(LDA,*) – ***complex*** array. *Input/Output*

Note: the second dimension of the array A must be at least max(1,N).

On entry: the m by n matrix A.

On exit: if $m \geq n$, the elements below the diagonal are overwritten by details of the unitary matrix Q and the upper triangle is overwritten by the corresponding elements of the n by n upper triangular matrix R.

If $m < n$, the strictly lower triangular part is overwritten by details of the unitary matrix Q and the remaining elements are overwritten by the corresponding elements of the m by n upper trapezoidal matrix R.

The diagonal elements of R are real.

4: LDA – INTEGER. *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F08BSF (CGEQPF/ZGEQPF) is called.

Constraint: LDA $\geq$ max(1,M).

5: JPVT(*) – INTEGER array. *Input/Output*

Note: the dimension of the array JPVT must be at least max(1,N).

On entry: if JPVT(i) $\neq$ 0, then the ith column of A is moved to the beginning of AP before the decomposition is computed and is fixed in place during the computation. Otherwise, the ith column of A is a free column (i.e. one which may be interchanged during the computation with any other free column).

On exit: details of the permutation matrix P. More precisely, if JPVT(i) = k, then the kth column of A is moved to become the ith column of AP; in other words, the columns of AP are the columns of A in the order JPVT(1),JPVT(2),...,JPVT(n).

6: TAU(*) – ***complex*** array. *Output*

Note: the dimension of the array TAU must be at least max(1,min(M,N)).

On exit: further details of the unitary matrix Q.

7: WORK(*) – ***complex*** array. *Workspace*

Note: the dimension of the array WORK must be at least max(1,3*N).

8: RWORK(*) – ***real*** array. *Workspace*

Note: the dimension of the array RWORK must be at least max(1,2*N).

9: INFO – INTEGER. *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6. Error Indicators and Warnings

INFO < 0

If INFO = $-i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed factorization is the exact factorization of a nearby matrix $A + E$, where

$$\|E\|_2 = O(\varepsilon)\|A\|_2,$$

and ε is the ***machine precision.***

8. Further Comments

The total number of real floating-point operations is approximately $\frac{8}{3}n^2(3m-n)$ if $m \geq n$ or $\frac{8}{3}m^2(3n-m)$ if $m < n$.

To form the unitary matrix Q this routine may be followed by a call to F08ATF (CUNGQR/ZUNGQR):

```
CALL CUNGQR (M,M,MIN(M,N),A,LDA,TAU,WORK,LWORK,INFO)
```

but note that the second dimension of the array A must be at least M, which may be larger than was required by F08BSF.

When $m \geq n$, it is often only the first n columns of Q that are required, and they may be formed by the call:

```
CALL CUNGQR (M,N,N,A,LDA,TAU,WORK,LWORK,INFO)
```

To apply Q to an arbitrary complex rectangular matrix C, this routine may be followed by a call to F08AUF (CUNMQR/ZUNMQR). For example,

```
      CALL CUNMQR ('Left','Conjugate Transpose',M,P,MIN(M,N),A,LDA,TAU,
     +             C,LDC,WORK,LWORK,INFO)
```

forms $C = Q^H C$, where C is m by p.

To compute a QR factorization without column pivoting, use F08ASF (CGEQRF/ZGEQRF).

The real analogue of this routine is F08BEF (SGEQPF/DGEQPF).

9. Example

To solve the linear least-squares problem

minimize $\|Ax_i - b_i\|_2$ for $i = 1,2$

where b_1 and b_2 are the columns of the matrix B,

$$A = \begin{pmatrix} 0.47 - 0.34i & -0.40 + 0.54i & 0.60 + 0.01i & 0.80 - 1.02i \\ -0.32 - 0.23i & -0.05 + 0.20i & -0.26 - 0.44i & -0.43 + 0.17i \\ 0.35 - 0.60i & -0.52 - 0.34i & 0.87 - 0.11i & -0.34 - 0.09i \\ 0.89 + 0.71i & -0.45 - 0.45i & -0.02 - 0.57i & 1.14 - 0.78i \\ -0.19 + 0.06i & 0.11 - 0.85i & 1.44 + 0.80i & 0.07 + 1.14i \end{pmatrix}$$

and

$$B = \begin{pmatrix} -0.85 - 1.63i & 2.49 + 4.01i \\ -2.16 + 3.52i & -0.14 + 7.98i \\ 4.57 - 5.71i & 8.36 - 0.28i \\ 6.38 - 7.40i & -3.55 + 1.29i \\ 8.41 + 9.39i & -6.72 + 5.03i \end{pmatrix}.$$

Here A is approximately rank-deficient, and hence it is preferable to use F08BSF (CGEQPF/ZGEQPF) rather than F08ASF (CGEQRF/ZGEQRF).

9.1. Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F08BSF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          MMAX, NMAX, LDA, LDB, LDX, NRHMAX, LWORK
      PARAMETER        (MMAX=8,NMAX=8,LDA=MMAX,LDB=MMAX,LDX=MMAX,
     +                 NRHMAX=NMAX,LWORK=64*NMAX)
      complex          ZERO
      PARAMETER        (ZERO=(0.0e0,0.0e0))
*     .. Local Scalars ..
      real             TOL
      INTEGER          I, IFAIL, INFO, J, K, M, N, NRHS
*     .. Local Arrays ..
      complex          A(LDA,NMAX), B(LDB,NRHMAX), TAU(NMAX),
     +                 WORK(LWORK), X(LDX,NRHMAX)
      real             RWORK(2*NMAX)
      INTEGER          JPVT(NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         F06DBF, F06HBF, X04DBF, cgeqpf, ctrsv, cunmqr
*     .. Intrinsic Functions ..
      INTRINSIC        ABS
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08BSF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) M, N, NRHS
      IF (M.LE.MMAX .AND. N.LE.NMAX .AND. M.GE.N .AND. NRHS.LE.NRHMAX)
     +    THEN
*
*        Read A and B from data file
*
         READ (NIN,*) ((A(I,J),J=1,N),I=1,M)
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,M)
*
*        Initialize JPVT to be zero so that all columns are free
*
         CALL F06DBF(N,0,JPVT,1)
*
*        Compute the QR factorization of A
*
         CALL cgeqpf(M,N,A,LDA,JPVT,TAU,WORK,RWORK,INFO)
*
*        Choose TOL to reflect the relative accuracy of the input data
*
         TOL = 0.01e0
*
*        Determine which columns of R to use
*
         DO 20 K = 1, N
            IF (ABS(A(K,K)).LE.TOL*ABS(A(1,1))) GO TO 40
   20    CONTINUE
*
*        Compute C = (Q**H)*B, storing the result in B
*
   40    K = K - 1
*
         CALL cunmqr('Left','Conjugate Transpose',M,NRHS,K,A,LDA,TAU,B,
     +               LDB,WORK,LWORK,INFO)
*
*        Compute least-squares solution by backsubstitution in R*B = C
*
         DO 60 I = 1, NRHS
```

```
*
            CALL ctrsv('Upper','No transpose','Non-Unit',K,A,LDA,B(1,I),
     +                 1)
*
*           Set the unused elements of the I-th solution vector to zero
*
            CALL F06HBF(N-K,ZERO,B(K+1,I),1)
*
   60     CONTINUE
*
*         Unscramble the least-squares solution stored in B
*
          DO 100 I = 1, N
             DO 80 J = 1, NRHS
                X(JPVT(I),J) = B(I,J)
   80        CONTINUE
  100     CONTINUE
*
*         Print least-squares solution
*
          WRITE (NOUT,*)
          IFAIL = 0
*
          CALL X04DBF('General',' ',N,NRHS,X,LDX,'Bracketed','F7.4',
     +               'Least-squares solution','Integer',RLABS,'Integer',
     +               CLABS,80,0,IFAIL)
*
       END IF
       STOP
       END
```

9.2. Program Data

```
F08BSF Example Program Data
  5  4  2                                                  :Values of M, N and NRHS
 ( 0.47,-0.34) (-0.40, 0.54) ( 0.60, 0.01) ( 0.80,-1.02)
 (-0.32,-0.23) (-0.05, 0.20) (-0.26,-0.44) (-0.43, 0.17)
 ( 0.35,-0.60) (-0.52,-0.34) ( 0.87,-0.11) (-0.34,-0.09)
 ( 0.89, 0.71) (-0.45,-0.45) (-0.02,-0.57) ( 1.14,-0.78)
 (-0.19, 0.06) ( 0.11,-0.85) ( 1.44, 0.80) ( 0.07, 1.14)    :End of matrix A
 (-0.85,-1.63) ( 2.49, 4.01)
 (-2.16, 3.52) (-0.14, 7.98)
 ( 4.57,-5.71) ( 8.36,-0.28)
 ( 6.38,-7.40) (-3.55, 1.29)
 ( 8.41, 9.39) (-6.72, 5.03)                                :End of matrix B
```

9.3. Program Results

```
 F08BSF Example Program Results

 Least-squares solution
                        1                       2
 1   ( 0.0000, 0.0000) ( 0.0000, 0.0000)
 2   ( 2.6925, 8.0446) (-2.0563,-2.9759)
 3   ( 2.7602, 2.5455) ( 1.0588, 1.4635)
 4   ( 2.7383, 0.5123) (-1.4150, 0.2982)
```