# NAG Fortran Library Manual
# Mark 18

# Volume 8

# F08 – F11

F08 – Least-squares and Eigenvalue Problems (LAPACK)
F11 – Sparse Linear Algebra

NAG®

**NAG Fortran Library Manual, Mark 18**



Printed and produced by NAG

1st Edition – September 1997 ISBN 1-85206-147-2



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

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

NAG GmbH
Schleißheimerstraße 5
D-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.

[*NP3086/18*]

# 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 | $QR$ factorization of real general rectangular matrix (SGEQRF/DGEQRF) |
| F08AFF | 16 | Form all or part of orthogonal $Q$ from $QR$ factorization determined by F08AEF or F08BEF (SORGQR/DORGQR) |
| F08AGF | 16 | Apply orthogonal transformation determined by F08AEF or F08BEF (SORMQR/DORMQR) |
| F08AHF | 16 | $LQ$ factorization of real general rectangular matrix (SGELQF/DGELQF) |
| F08AJF | 16 | Form all or part of orthogonal $Q$ from $LQ$ factorization determined by F08AHF (SORGLQ/DORGLQ) |
| F08AKF | 16 | Apply orthogonal transformation determined by F08AHF (SORMLQ/DORMLQ) |
| F08ASF | 16 | $QR$ factorization of complex general rectangular matrix (CGEQRF/ZGEQRF) |
| F08ATF | 16 | Form all or part of unitary $Q$ from $QR$ factorization determined by F08ASF or F08BSF (CUNGQR/ZUNGQR) |
| F08AUF | 16 | Apply unitary transformation determined by F08ASF or F08BSF (CUNMQR/ZUNMQR) |
| F08AVF | 16 | $LQ$ factorization of complex general rectangular matrix (CGELQF/ZGELQF) |
| F08AWF | 16 | Form all or part of unitary $Q$ from $LQ$ factorization determined by F08AVF (CUNGLQ/ZUNGLQ) |
| F08AXF | 16 | Apply unitary transformation determined by F08AVF (CUNMLQ/ZUNMLQ) |
| F08BEF | 16 | Form all or part of orthogonal $Q$ from $QR$ factorization determined by F08AEF or F08BEF (SORGQR/DORGQR) |
| F08BSF | 16 | Form all or part of unitary $Q$ from $QR$ factorization determined by F08ASF or F08BSF (CUNGQR/ZUNGQR) |
| F08FEF | 16 | Orthogonal reduction of real symmetric matrix to symmetric tridiagonal form (SSYTRD/DSYTRD) |
| F08FFF | 16 | Generate orthogonal transformation matrix from reduction to tridiagonal form determined by F08FEF (SORGTR/DORGTR) |
| F08FGF | 16 | Apply orthogonal transformation determined by F08FEF (SORMTR/DORMTR) |
| F08FSF | 16 | Unitary reduction of complex Hermitian matrix to real symmetric tridiagonal form (CHETRD/ZHETRD) |
| F08FTF | 16 | Generate unitary transformation matrix from reduction to tridiagonal form determined by F08FSF (CUNGTR/ZUNGTR) |
| F08FUF | 16 | Apply unitary transformation matrix determined by F08FSF (CUNMTR/ZUNMTR) |
| F08GEF | 16 | Orthogonal reduction of real symmetric matrix to symmetric tridiagonal form, packed storage (SSPTRD/DSPTRD) |
| F08GFF | 16 | Generate orthogonal transformation matrix from reduction to tridiagonal form determined by F08GEF (SOPGTR/DOPGTR) |
| F08GGF | 16 | Apply orthogonal transformation determined by F08GEF (SOPMTR/DOPMTR) |
| F08GSF | 16 | Unitary reduction of complex Hermitian matrix to real symmetric tridiagonal form, packed storage (CHPTRD/ZHPTRD) |

| | | |
|---|---|---|
| F08GTF | 16 | Generate unitary transformation matrix from reduction to tridiagonal form determined by F08GSF (CUPGTR/ZUPGTR) |
| F08GUF | 16 | Apply unitary transformation matrix determined by F08GSF (CUPMTR/ZUPMTR) |
| F08HEF | 16 | Orthogonal reduction of real symmetric band matrix to symmetric tridiagonal form (SSBTRD/DBSTRD) |
| F08HSF | 16 | Unitary reduction of complex Hermitian band matrix to real symmetric tridiagonal form (CHBTRD/ZHBTRD) |
| F08JEF | 16 | All eigenvalues and eigenvectors of real symmetric tridiagonal matrix, reduced from real symmetric matrix using implicit $QL$ or $QR$ (SSTEQR/DSTEQR) |
| F08JFF | 16 | All eigenvalues of real symmetric tridiagonal matrix, root-free variant of $QL$ or $QR$ (SSTERF/DSTERF) |
| F08JGF | 16 | All eigenvalues and eigenvectors of real symmetric positive definite tridiagonal matrix, reduced from real symmetric positive definite matrix (SPTEQR/DPTEQR) |
| F08JJF | 16 | Selected eigenvalues of real symmetric tridiagonal matrix by bisection (SSTEBZ/DSTEBZ) |
| F08JKF | 16 | Selected eigenvectors of real symmetric tridiagonal matrix by inverse iteration, storing eigenvectors in real array (SSTEIN/DSTEIN) |
| F08JSF | 16 | All eigenvalues and eigenvectors of real symmetric tridiagonal matrix, reduced from complex Hermitian matrix, using implicit $QL$ or $QR$ (CSTEQR/ZSTEQR) |
| F08JUF | 16 | All eigenvalues and eigenvectors of real symmetric positive definite tridiagonal matrix, reduced from complex Hermitian positive definite matrix (CPTEQR/ZPTEQR) |
| F08JXF | 16 | Selected eigenvectors of real symmetric tridiagonal matrix by inverse iteration, storing eigenvectors in complex array (CSTEIN/ZSTEIN) |
| F08KEF | 16 | Orthogonal reduction of real general rectangular matrix to bidiagonal form (SGEBRD/DGEBRD) |
| F08KFF | 16 | Generate orthogonal transformation matrices from reduction to bidiagonal form determined by F08KEF (SORGBR/DORGBR) |
| F08KGF | 16 | Apply orthogonal transformations from reduction to bidiagonal form determined by F08KEF (SORMBR/DORMBR) |
| F08KSF | 16 | Unitary reduction of complex general rectangular matrix to bidiagonal form (CGEBRD/ZGEBRD) |
| F08KTF | 16 | Generate unitary transformation matrices from reduction to bidiagonal form determined by F08KSF (CUNGBR/ZUNGBR) |
| F08KUF | 16 | Apply unitary transformations from reduction to bidiagonal form determined by F08KSF (CUNMBR/ZUNMBR) |
| F08MEF | 16 | SVD of real bidiagonal matrix reduced from real general matrix (SBDSQR/DBDSQR) |
| F08MSF | 16 | SVD of real bidiagonal matrix reduced from complex general matrix (CBDSQR/ZBDSQR) |
| F08NEF | 16 | Orthogonal reduction of real general matrix to upper Hessenberg form (SGEHRD/DGEHRD) |
| F08NFF | 16 | Generate orthogonal transformation matrix from reduction to Hessenberg form determined by F08NEF (SORGHR/DORGHR) |
| F08NGF | 16 | Apply orthogonal transformation matrix from reduction to Hessenberg form determined by F08NEF (SORMHR/DORMHR) |
| F08NHF | 16 | Balance real general matrix (SGEBAL/DGEBAL) |
| F08NJF | 16 | Transform eigenvectors of real balanced matrix to those of original matrix supplied to F08NHF (SGEBAK/DGEBAK) |
| F08NSF | 16 | Unitary reduction of complex general matrix to upper Hessenberg form (CGEHRD/ZGEHRD) |

| | | |
|---|---|---|
| F08NTF | 16 | Generate unitary transformation matrix from reduction to Hessenberg form determined by F08NSF (CUNGHR/ZUNGHR) |
| F08NUF | 16 | Apply unitary transformation matrix from reduction to Hessenberg form determined by F08NSF (CUNMHR/ZUNMHR) |
| F08NVF | 16 | Balance complex general matrix (CGEBAL/ZGEBAL) |
| F08NWF | 16 | Transform eigenvectors of complex balanced matrix to those of original matrix supplied to F08NVF (CGEBAK/ZGEBAK) |
| F08PEF | 16 | Eigenvalues and Schur factorization of real upper Hessenberg matrix reduced from real general matrix (SHSEQR/DHSEQR) |
| F08PKF | 16 | Selected right and/or left eigenvectors of real upper Hessenberg matrix by inverse iteration (SHSEIN/DHSEIN) |
| F08PSF | 16 | Eigenvalues and Schur factorization of complex upper Hessenberg matrix reduced from complex general matrix (CHSEQR/ZHSEQR) |
| F08PXF | 16 | Selected right and/or left eigenvectors of complex upper Hessenberg matrix by inverse iteration (CHSEIN/ZHSEIN) |
| F08QFF | 16 | Reorder Schur factorization of real matrix using orthogonal similarity transformation (STREXC/DTREXC) |
| F08QGF | 16 | Reorder Schur factorization of real matrix, form orthonormal basis of right invariant subspace for selected eigenvalues, with estimates of sensitivities (STRSEN/DTRSEN) |
| F08QHF | 16 | Solve real Sylvester matrix equation $AX + XB = C$, $A$ and $B$ are upper quasi-triangular or transposes (STRSYL/DTRSYL) |
| F08QKF | 16 | Left and right eigenvectors of a real upper quasi-triangular matrix (STREVC/DTREVC) |
| F08QLF | 16 | Estimates of sensitivities of selected eigenvalues and eigenvectors of real upper quasi-triangular matrix (STRSNA/DTRSNA) |
| F08QTF | 16 | Reorder Schur factorization of complex matrix using unitary similarity transformation (CTREXC/ZTREXC) |
| F08QUF | 16 | Reorder Schur factorization of complex matrix, form orthonormal basis of right invariant subspace for selected eigenvalues, with estimates of sensitivities (CTRSEN/ZTRSEN) |
| F08QVF | 16 | Solve complex Sylvester matrix equation $AX + XB = C$, $A$ and $B$ are upper triangular or conjugate-transposes (CTRSYL/ZTRSYL) |
| F08QXF | 16 | Left and right eigenvectors of a complex upper triangular matrix (CTREVC/ZTREVC) |
| F08QYF | 16 | Estimates of sensitivities of selected eigenvalues and eigenvectors of complex upper triangular matrix (CTRSNA/ZTRSNA) |
| F08SEF | 16 | Reduction to standard form of real symmetric-definite generalized eigenproblem $Ax = \lambda Bx$, $ABx = \lambda x$ or $BAx = \lambda x$, $B$ factorized by F07FDF (SSYGST/DSYGST) |
| F08SSF | 16 | Reduction to standard form of complex Hermitian-definite generalized eigenproblem $Ax = \lambda Bx$, $ABx = \lambda x$ or $BAx = \lambda x$, $B$ factorized by F07FRF (CHEGST/ZHEGST) |
| F08TEF | 16 | 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 (SSPGST/DSPGST) |
| F08TSF | 16 | 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 (CHPGST/ZHPGST) |

# 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 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 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 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 F02 Chapter Introduction. The decision trees, at the end of 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. Where possible, routines in Chapter F02 call Chapter F08 routines to perform the necessary computational tasks.

The routines in this chapter (F08) handle only *dense*, *band* 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 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 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 $\Sigma$ is an $m$ by $n$ diagonal matrix with real diagonal elements, $\sigma_i$, such that

$$\sigma_1 \geq \sigma_2 \geq \cdots \sigma_{\min(m,n)} \geq 0.$$

The $\sigma_i$ 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 $i$th 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 $m < n$ 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 $\Sigma$ 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 $\Sigma$, 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*, $\lambda$, 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 $\lambda$ 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 $\Lambda$ 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 $\lambda$ 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 $\Lambda$ is diagonal. The diagonal entries of $\Lambda$ 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).

## 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. | $Az = \lambda Bz$ | $B = LL^T$<br>$B = U^TU$ | $C = L^{-1}AL^{-T}$<br>$C = U^{-T}AU^{-1}$ | $z = L^{-T}y$<br>$z = U^{-1}y$ |
| 2. | $ABz = \lambda z$ | $B = LL^T$<br>$B = U^TU$ | $C = L^TAL$<br>$C = UAU^T$ | $z = L^{-T}y$<br>$z = U^{-1}y$ |
| 3. | $BAz = \lambda z$ | $B = LL^T$<br>$B = U^TU$ | $C = L^TAL$<br>$C = UAU^T$ | $z = Ly$<br>$z = U^Ty$ |

**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 Symmetric 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 symmetric band matrices is described in Section 3.3.

## 2.9 Nonsymmetric Eigenvalue Problems

The *nonsymmetric eigenvalue problem* is to find the *eigenvalues*, $\lambda$, 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 $\lambda$ 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 $i$th computed eigenvalue $\hat{\lambda}_i$, where $\lambda_i$ is the true $i$th 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 $\alpha$ 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 $n$. 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 $n$. Each computed singular value $\hat{\sigma}_i$ differs from the true $\sigma_i$ by an amount satisfying the bound

$$|\hat{\sigma}_i - \sigma_i| \leq p(m,n)\epsilon\sigma_1.$$

Thus large singular values (those near $\sigma_1$) 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 $\sigma_i$ and the nearest other singular value. Thus, if $\sigma_i$ 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 $\sigma_i$ 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 $n$. Each computed eigenvalue $\hat{\lambda}_i$ differs from the true $\lambda_i$ 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 $\lambda_i$ and the nearest other eigenvalue. Thus, if $\lambda_i$ 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 $\lambda_i$ and the nearest other eigenvalue.

(1) $A - \lambda B$. The computed eigenvalues $\hat{\lambda}_i$ can differ from the true eigenvalues $\lambda_i$ 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 $\lambda_i$ 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 $i$th computed eigenvalue and $\lambda_i$ the $i$th 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 $\lambda_I$ 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 $\text{sep}_i$, 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 | $\|\hat{\lambda}_i - \lambda_i\| \lesssim \|E\|_2 / s_i$ |
| Eigenvalue cluster | $\|\hat{\lambda}_I - \lambda_I \lesssim \|E\|_2 / s_I$ |
| 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 $CQ^T - Q^HC$ or $CQ^H$ if complex).

| | **Factorize without pivoting** | **Factorize with pivoting** | **Generate Matrix $Q$** | **Apply matrix $Q$** |
|---|---|---|---|---|
| $QR$ factorization, real matrices | F08AEF<br>SGEQRF<br>DGEQRF | F08BEF<br>SGEQPF<br>DGEQPF | F08AFF<br>SORGQR<br>DORGQR | F08AGF<br>SORMQR<br>DORMQR |
| $LQ$ factorization, real matrices | F08AHF<br>SGELQF<br>DGELQF | | F08AJF<br>SORGLQ<br>DORGLQ | F08AKF<br>SORMLQ<br>DORMLQ |
| $QR$ factorization, complex matrices | F08ASF<br>CGEQRF<br>ZGEQRF | F08BSF<br>CGEQPF<br>ZGEQPF | F08ATF<br>CUNGQR<br>ZUNMQR | F08AUF<br>CUNMQR<br>ZUNGQR |
| $LQ$ factorization, complex matrices | F08AVF<br>CGELQF<br>ZGELQF | | F08AWF<br>CUNGLQ<br>ZUNGLQ | F08AXF<br>CUNMQL<br>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).

These routines 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 $A$.

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 $P$ | SVD of bidiagonal form ($QR$ algorithm) |
|---|---|---|---|---|
| real matrices | F08KEF<br>SGEBRD<br>DGEBRD | F08KFF<br>SORGBR<br>DORGBR | F08KGF<br>SORMBR<br>DORMBR | F08MEF<br>SBDSQR<br>DBDSQR |
| complex matrices | F08KSF<br>CGEBRD<br>ZGEBRD | F08KTF<br>CUNGBR<br>ZUNGBR | F08KUF<br>CUNMBR<br>ZUNMBR | F08MSF<br>CBDSQR<br>ZBDSQR |

To compute the singular values and vectors of a rectangular matrix, as described in Section 2.3, use the following sequence of calls:

real matrix, singular values and vectors F08KEF, F08KFF*, F08MEF
complex matrix, singular values and vectors F08KSF, F08KTF*, 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<br>SSYTRD<br>DSYTRD | F08FFF<br>SORGTR<br>DORGTR | F08FGF<br>SORMTR<br>DORMTR |
| real symmetric matrices (packed storage) | F08GEF<br>SSPTRD<br>DSPTRD | F08GFF<br>SOPGTR<br>DOPGTR | F08GGF<br>SOPMTR<br>DOPMTR |
| real symmetric band matrices | F08HEF<br>SSBTRD<br>DSBTRD | | |
| complex Hermitian matrices | F08FSF<br>CHETRD<br>ZHETRD | F08FTF<br>CUNGTR<br>ZUNGTR | F08FUF<br>CUNMTR<br>ZUNMTR |
| complex Hermitian matrices (packed storage) | F08GSF<br>CHPTRD<br>ZHPTRD | F08GTF<br>CUPGTR<br>ZUPGTR | F08GUF<br>CUPMTR<br>ZUPMTR |
| complex Hermitian band matrices | F08HSF<br>CHBTRD<br>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.

| | **all eigenvalues and eigenvectors ($QR$ algorithm)** | **all eigenvalues (root-free $QR$ algorithm)** | **all eigenvalues and eigenvectors (positive-definite)** | **selected eigenvalues (bisection)** | **selected eigenvectors (inverse iteration)** |
|---|---|---|---|---|---|
| real matrices | F08JEF<br>SSTEQR<br>DSTEQR | F08JFF<br>SSTERF<br>DSTERF | F08JGF<br>SPTEQR<br>DPTEQR | F08JJF<br>SSTEBZ<br>DSTEBZ | F08JKF<br>SSTEIN<br>DSTEIN |
| complex matrices | F08JSF<br>CSTEQR<br>ZSTEQR | | F08JUF<br>CPTEQR<br>ZPTEQR | | F08JXF<br>CSTEIN<br>ZSTEIN |

The following sequences of calls may be used to compute various combinations of eigenvalues and eigenvectors, as described in Section 2.5:

real symmetric matrix,
all eigenvalues and eigenvectors — `F08FEF, F08FFF*, F08JEF`

real symmetric matrix,
selected eigenvalues and eigenvectors — `F08FEF, F08JJF, F08JKF, F08FGF*`

real symmetric matrix,
all eigenvalues and eigenvectors — `F08HEF*, F08JEF`

complex Hermitian matrix,
all eigenvalues and eigenvectors — `F08FSF, F08FFF*, F08JSF`

complex Hermitian matrix,
selected eigenvalues and eigenvectors — `F08FSF, F08JJF, F08JXF, F08FUF*`

complex Hermitian matrix,
all eigenvalues and eigenvectors — `F08HSF*, F08JEF`

To use packed storage, simply replace the F08F- routines by the corresponding F08G- routines.

### 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.

| | **Reduce to standard problem** | **Reduce to standard problem (packed storage)** |
|---|---|---|
| real symmetric matrices | F08SEF<br>SSYGST<br>DSYGST | F08TEF<br>SSPGST<br>DSPGST |
| complex Hermitian matrices | F08SSF<br>CHEGST<br>ZHEGST | F08TSF<br>CHPGST<br>ZHPGST |

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 |
| complex Hermitian-definite problem | F07FRF, F08SSF*, F08FSF, F08JFF |
| complex Hermitian-definite problem, packed storage | F07GRF, F08TSF*, F08GSF, 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<br>SGEHRD<br>DGEHRD | F08NFF<br>SORGHR<br>DORGHR | F08NGF<br>SORMHR<br>DORMHR | F08NHF<br>SGEBAL<br>DGEBAL | F08NJF<br>SGEBAK<br>DGEBAK |
| complex matrices | F08NSF<br>CGEHRD<br>ZGEHRD | F08NTF<br>CUNGHR<br>ZUNGHR | F08NUF<br>CUNMHR<br>ZUNMHR | F08NVF<br>CGEBAL<br>ZGEBAL | F08NWF<br>CGEBAK<br>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<br>SHSEQR<br>DHSEQR | F08PKF<br>SHSEIN<br>DHSEIN | F08QKF<br>STREVC<br>DTREVC | F08QLF<br>STRSNA<br>DTRSNA |
| complex matrices | F08PSF<br>CHSEQR<br>ZHSEQR | F08PXF<br>CHSEIN<br>ZHSEIN | F08QXF<br>CTREVC<br>ZTREVC | F08QYF<br>CTRSNA<br>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<br>STREXC<br>DTREXC | F08QGF<br>STRSEN<br>DTRSEN |
| complex matrices | F08QTF<br>CTREXC<br>ZTREXC | F08QUF<br>CTRSEN<br>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<br>STRSYL<br>DTRSYL |
| complex matrices | F08QVF<br>CTRSYL<br>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

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 band symmetric or Hermitian 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 $\mathrm{A}(i,j)$.

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' | $\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ & a_{22} & a_{23} & a_{24} \\ & & a_{33} & a_{34} \\ & & & a_{44} \end{pmatrix}$ | $\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ * & a_{22} & a_{23} & a_{24} \\ * & * & a_{33} & a_{34} \\ * & * & * & a_{44} \end{pmatrix}$ |
| 'L' | $\begin{pmatrix} a_{11} & & & \\ a_{21} & a_{22} & & \\ a_{31} & a_{32} & a_{33} & \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$ | $\begin{pmatrix} a_{11} & * & * & * \\ a_{21} & a_{22} & * & * \\ a_{31} & a_{32} & a_{33} & * \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$ |

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' | $\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ \bar{a}_{12} & a_{22} & a_{23} & a_{24} \\ \bar{a}_{13} & \bar{a}_{23} & a_{33} & a_{34} \\ \bar{a}_{14} & \bar{a}_{24} & \bar{a}_{34} & a_{44} \end{pmatrix}$ | $\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ * & a_{22} & a_{23} & a_{24} \\ * & * & a_{33} & a_{34} \\ * & * & * & a_{44} \end{pmatrix}$ |
| 'L' | $\begin{pmatrix} a_{11} & \bar{a}_{21} & \bar{a}_{31} & \bar{a}_{41} \\ a_{21} & a_{22} & \bar{a}_{32} & \bar{a}_{42} \\ a_{31} & a_{32} & a_{33} & \bar{a}_{43} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$ | $\begin{pmatrix} a_{11} & * & * & * \\ a_{21} & a_{22} & * & * \\ a_{31} & a_{32} & a_{33} & * \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$ |

### 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 symmetric or Hermitian band matrix with $k$ subdiagonals and superdiagonals may be stored compactly in a two-dimensional array with $k+1$ 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. Only the upper or lower triangle (as specified by UPLO) need to be stored. This storage scheme should be used in practice only if $k \ll n$, although routines in Chapter F07 and Chapter F08 work correctly for all values of $k$. In Chapter F07 and Chapter F08, arrays that hold matrices in band storage have argument names ending in 'B'. So:

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

if UPLO = 'L', $a_{ij}$ is stored in 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 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 $\tau$ 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, $\tau$ 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 $\tau$ 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* $\beta$. 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

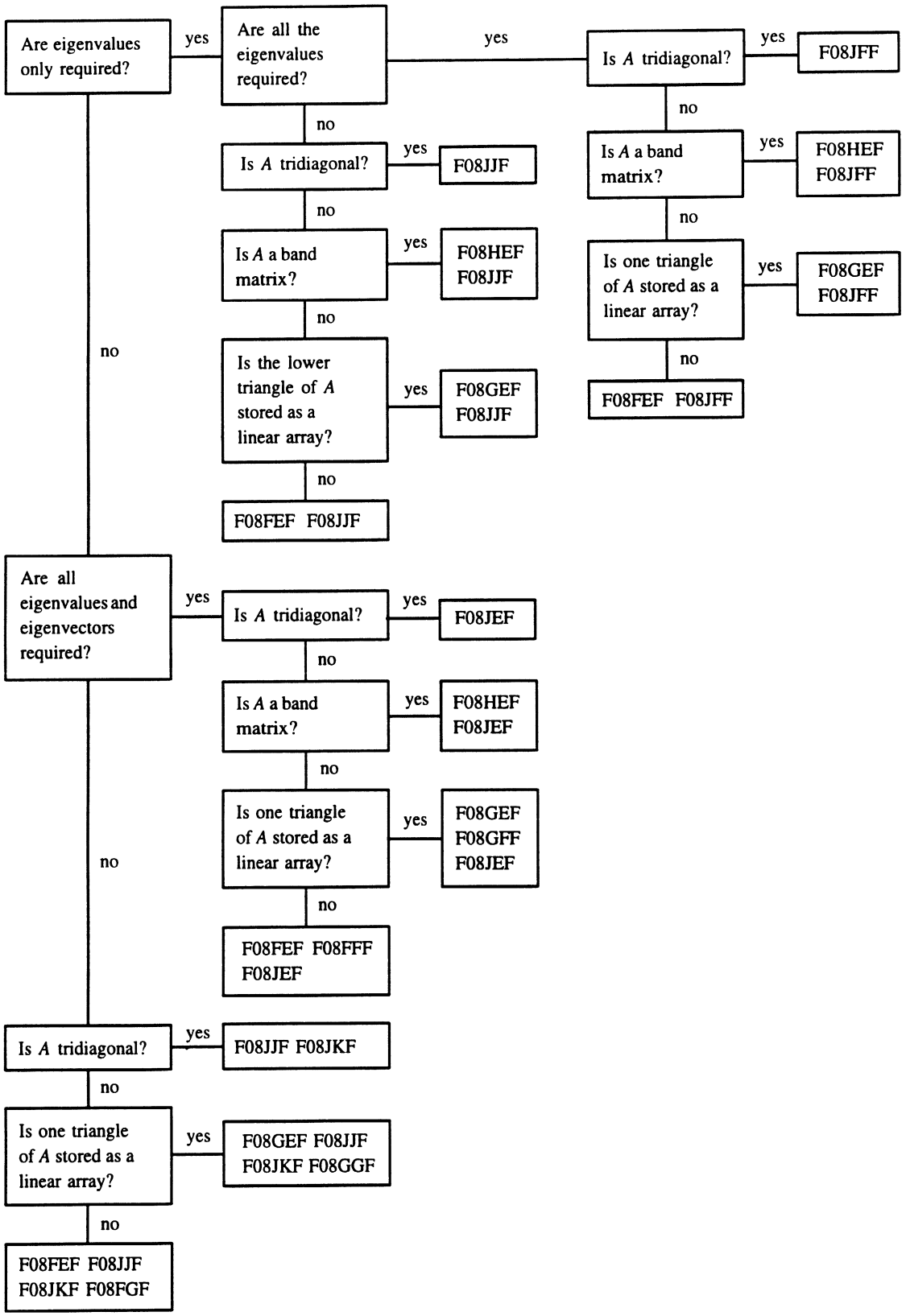
If the routine document specifies that the routine may terminate with INFO $< 0$, then it is **essential to test INFO on exit** from the routine. (This corresponds to a *soft failure* in terms of the usual NAG error-handling terminology.) No error message is output.

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 $i$th 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.)

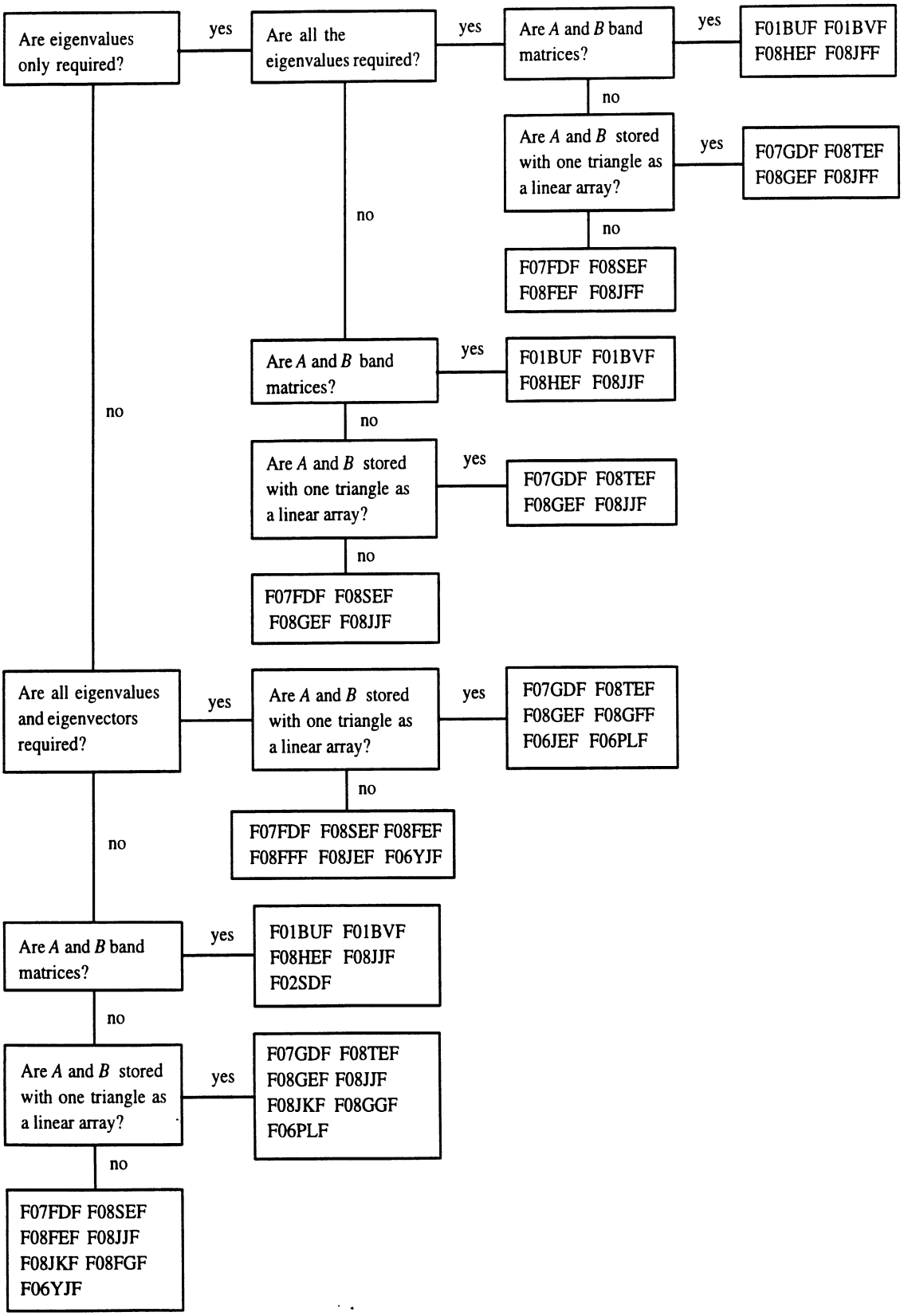
# 4 Decision Trees

## 4.1 General purpose routines (eigenvalues and eigenvectors)

**Tree 1: Real Symmetric Matrices**

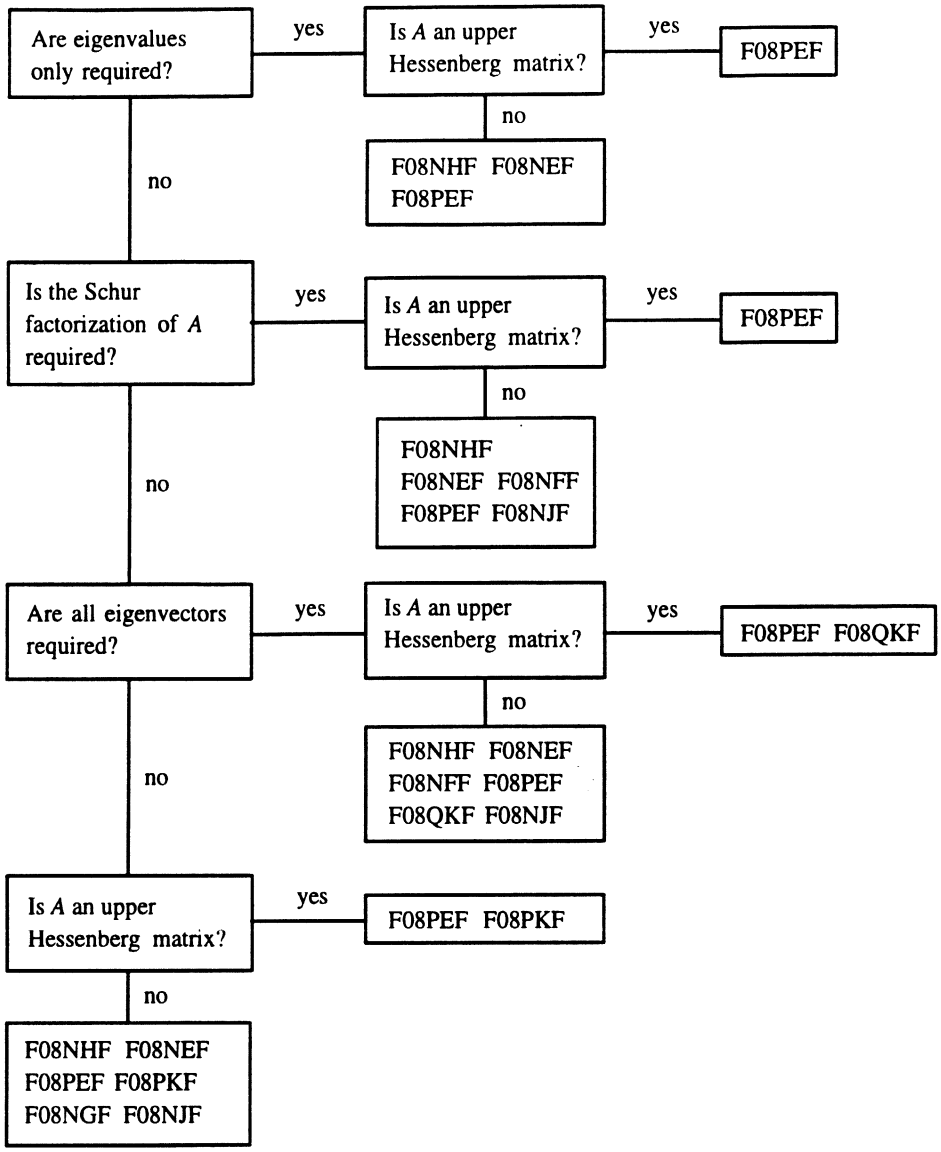


**Tree 2: Real Generalized Symmetric-definite Eigenvalue Problems**

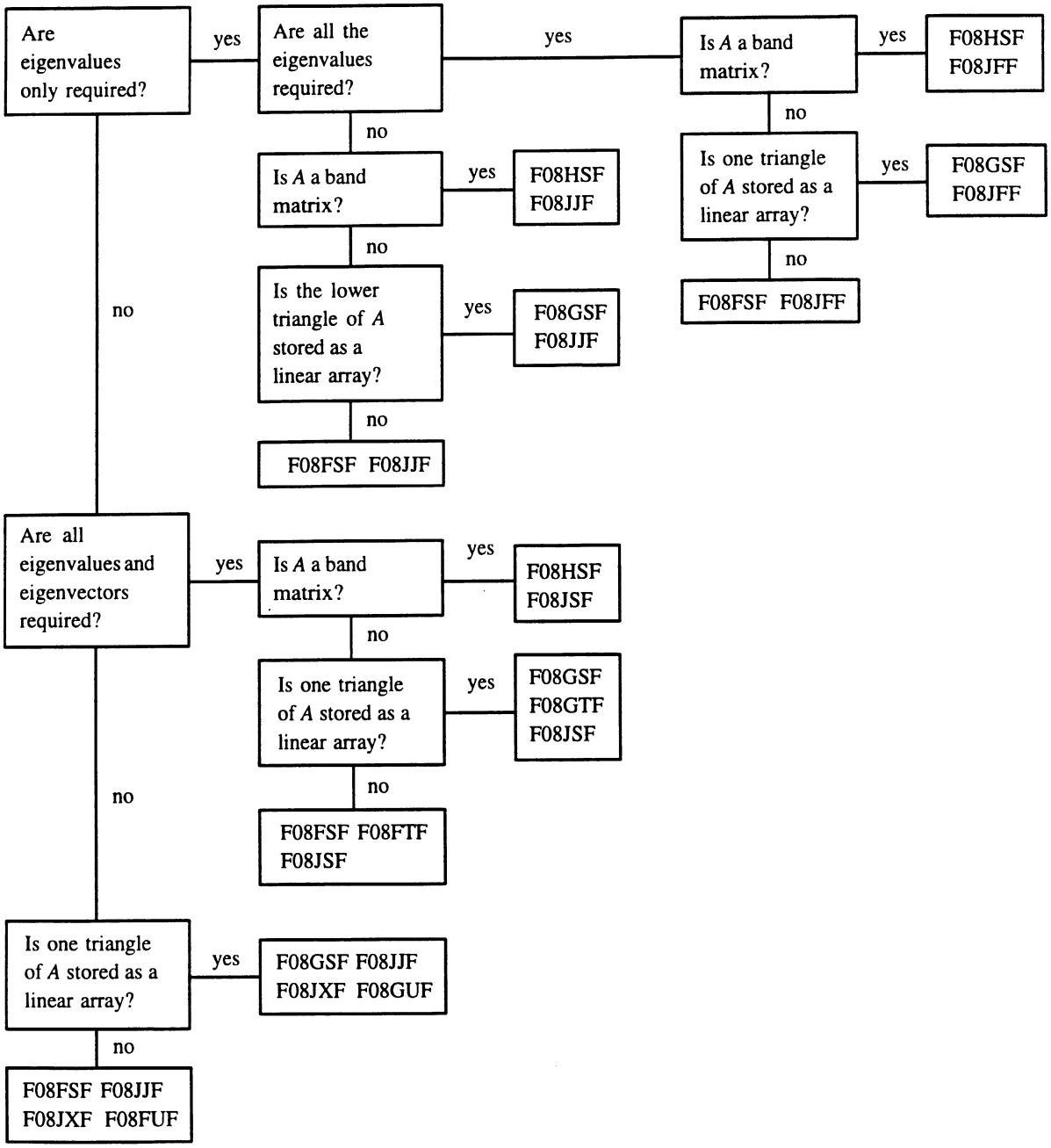



**Note:** the routines for band matrices only handle the problem $Ax = \lambda Bx$; the other routines handle all three types of problems ($Ax = \lambda Bx$, $ABx = \lambda x$ or $BAx = \lambda x$) except that, if the problem is $BAx = \lambda x$ and eigenvectors are required, F06PHF must be used instead of F06PLF, and F06YFF instead of F06YJF.

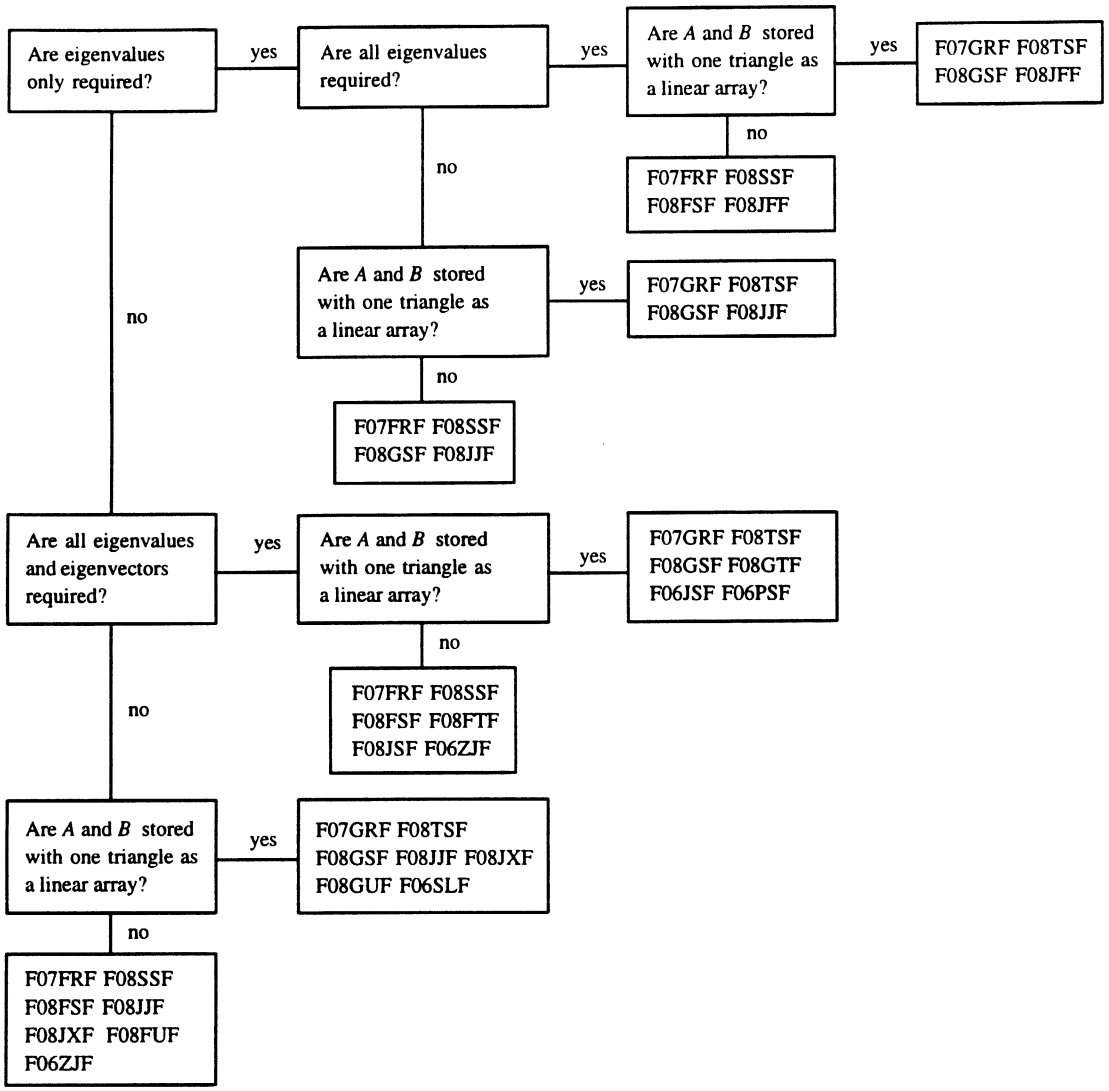
**Tree 3: Real Nonsymmetric Matrices**



**Tree 4: Complex Hermitian Matrices**

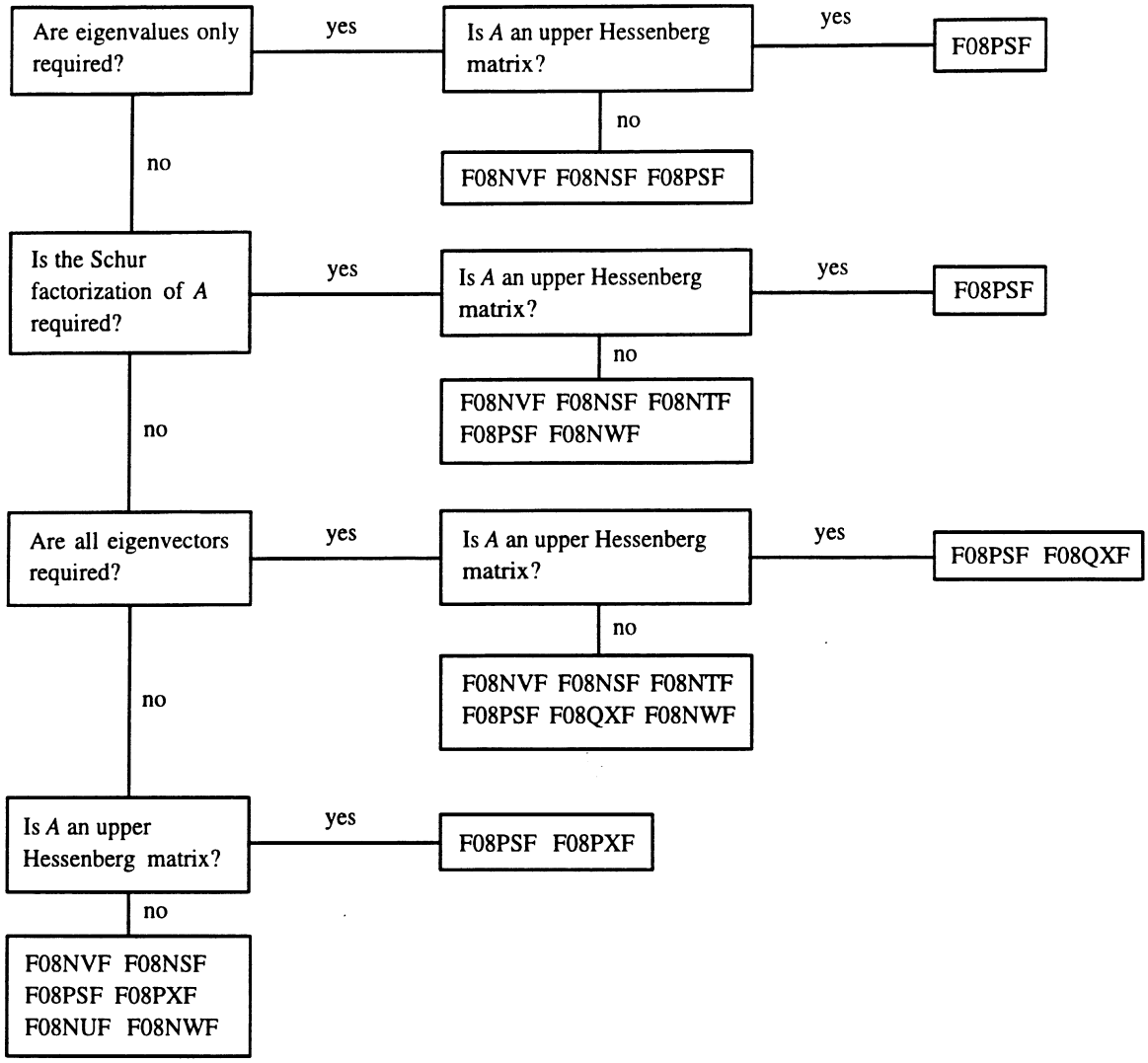


**Tree 5: Complex Generalized Hermitian-definite Eigenvalue Problems**

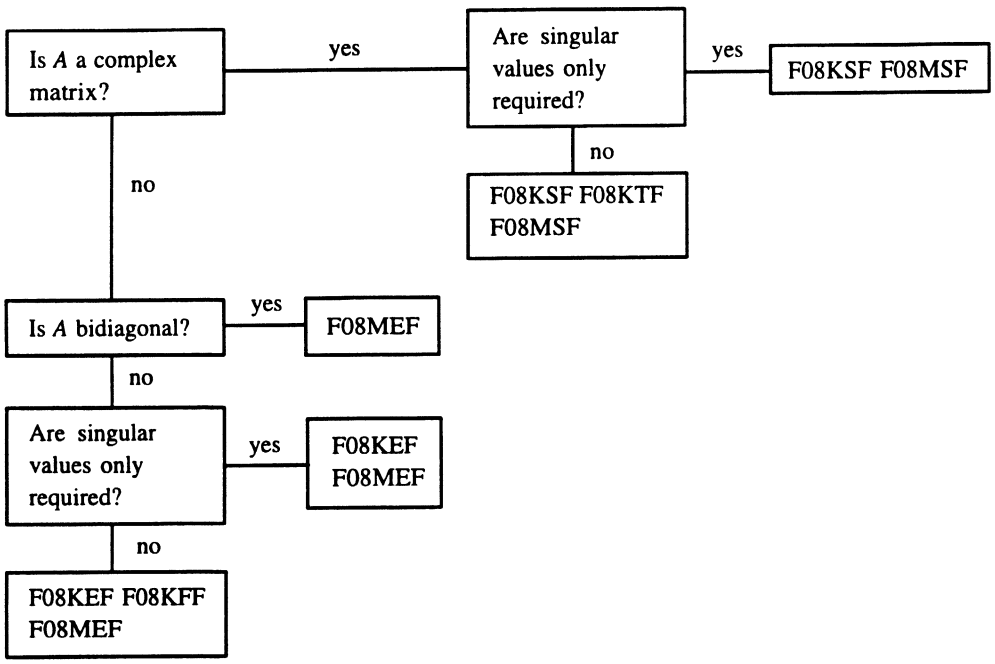



**Note:** the same routines are required for all three types of problem ($Ax = \lambda Bx$, $ABx = \lambda x$ or $BAx = \lambda x$) except that, if the problem is $BAx = \lambda x$ and eigenvectors are required, F06SHF must be used instead of F06SLF, and F06ZFF instead of F06ZJF.

**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 |
| 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 |
| SPTEQR | DPTEQR | F08JGF |
| SSBTRD | DSBTRD | F08HEF |
| SSPGST | DSPGST | F08TEF |
| SSPTRD | DSPTRD | F08GEF |
| SSTEBZ | DSTEBZ | F08JJF |
| SSTEIN | DSTEIN | F08JKF |
| SSTERF | DSTERF | F08JFF |
| SSTEQR | DSTEQR | F08JEF |
| 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 |
| CGEBAK | ZGEBAK | F08NWF |
| CGEBAL | ZGEBAL | F08NVF |
| CGEBRD | ZGEBRD | F08KSF |
| CGEHRD | ZGEHRD | F08NSF |
| CGELQF | ZGELQF | F08AVF |
| CGEQPF | ZGEQPF | F08BSF |
| CGEQRF | ZGEQRF | F08ASF |
| CHBTRD | ZHBTRD | F08HSF |
| CHEGST | ZHEGST | F08SSF |
| CHETRD | ZHETRD | F08FSF |
| CHPGST | ZHPGST | F08TSF |
| CHPTRD | ZHPTRD | F08GSF |
| CHSEIN | ZHSEIN | F08PXF |
| CHSEQR | ZHSEQR | F08PSF |
| 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 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 (1989) *Matrix Computations* Johns Hopkins University Press (2nd 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 $i$th 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 $\varepsilon$ is the ***machine precision***.

## 8. Further Comments

The total number of 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 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 $C = Q^TC$, 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 $i$th 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 $\varepsilon$ 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

$$QC, Q^TC, CQ \text{ or } CQ^T,$$

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 $i$th 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 $\varepsilon$ 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 ≥ 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 ≥ 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×*nb*, where *nb* is the ***blocksize***.

*Constraint*: LWORK ≥ 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 $i$th 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 $\varepsilon$ 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 $i$th 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 $\varepsilon$ 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 $i$th 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 $\varepsilon$ 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 $i$th 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 $\varepsilon$ 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 $i$th 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 $\varepsilon$ 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 $i$th 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 $\varepsilon$ 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 ≥ 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 ≥ 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×$nb$, where $nb$ is the ***blocksize***.

*Constraint*: LWORK ≥ 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 $i$th 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 $\varepsilon$ is the ***machine precision***.

## 8. Further Comments

The total number of real floating-point operations is approximately $\frac{1}{3}m^2(3n-m)$ if $m \leq n$ or $\frac{1}{3}n^2(3m-n)$ 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 ≥ 0.

2: N – INTEGER. *Input*

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

*Constraint*: N ≥ M.

3: K – INTEGER. *Input*

*On entry*: $k$, the number of elementary reflectors whose product defines the matrix $Q$.

*Constraint*: M ≥ 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 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 $i$th 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 $\varepsilon$ 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 $\frac{1}{3}m^2(3n-m)$.

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

$$QC,\ Q^HC,\ CQ \text{ or } CQ^H,$$

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 $i$th 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 $\varepsilon$ 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 $i$th 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 $i$th 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 $i$th 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 $k$th column of $A$ is moved to become the $i$th 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 $i$th 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 $\varepsilon$ 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 $i$th 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 $i$th 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 $i$th 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 $k$th column of $A$ is moved to become the $i$th 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 $i$th 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 $\varepsilon$ 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 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)
```

# F08FEF (SSYTRD/DSYTRD) – 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

F08FEF (SSYTRD/DSYTRD) reduces a real symmetric matrix to tridiagonal form.

## 2. Specification

```
SUBROUTINE F08FEF (UPLO, N, A, LDA, D, E, TAU, WORK, LWORK, INFO)
ENTRY      ssytrd  (UPLO, N, A, LDA, D, E, TAU, WORK, LWORK, INFO)
INTEGER          N, LDA, LWORK, INFO
real             A(LDA,*), D(*), E(*), TAU(*), WORK(LWORK)
CHARACTER*1      UPLO
```

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

## 3. Description

This routine reduces a real symmetric matrix $A$ to symmetric tridiagonal form $T$ by an orthogonal similarity transformation: $A = QTQ^T$.

The matrix $Q$ is not formed explicitly but is represented as a product of $n-1$ elementary reflectors (see the Chapter Introduction for details). Routines are provided to work with $Q$ in this representation (see Section 8).

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §8.2.
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 as follows:

if UPLO = 'U', then the upper triangular part of $A$ is stored;

if UPLO = 'L', then the lower triangular part of $A$ is stored.

*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*: 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*: $A$ is overwritten by the tridiagonal matrix $T$ and details of the orthogonal matrix $Q$ 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 F08FEF (SSYTRD/DSYTRD) is called.

*Constraint*: LDA ≥ max(1,N).

5: D(*) – ***real*** array. *Output*

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

*On exit*: the diagonal elements of the tridiagonal matrix $T$.

6: E(*) – ***real*** array. *Output*

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

*On exit*: the off-diagonal elements of the tridiagonal matrix $T$.

7: TAU(*) – ***real*** array. *Output*

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

*On exit*: further details of the orthogonal matrix $Q$.

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

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

9: LWORK – INTEGER. *Input*

*On entry*: the dimension of the array WORK as declared in the (sub)program from which F08FEF (SSYTRD/DSYTRD) is called.

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

*Constraint*: LWORK ≥ 1.

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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

## 7. Accuracy

The computed tridiagonal matrix $T$ is exactly similar to a nearby matrix $A + E$, where

$$\|E\|_2 \leq c(n)\varepsilon\|A\|_2,$$

$c(n)$ is a modestly increasing function of $n$, and $\varepsilon$ is the ***machine precision***.

The elements of $T$ themselves may be sensitive to small perturbations in $A$ or to rounding errors in the computation, but this does not affect the stability of the eigenvalues and eigenvectors.

## 8. Further Comments

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

To form the orthogonal matrix $Q$ this routine may be followed by a call to F08FFF (SORGTR/DORGTR):

```
CALL SORGTR (UPLO,N,A,LDA,TAU,WORK,LWORK,INFO)
```

To apply $Q$ to an $n$ by $p$ real matrix $C$ this routine may be followed by a call to F08FGF (SORMTR/DORMTR). For example,

```
      CALL SORMTR ('Left',UPLO,'No Transpose',N,P,A,LDA,TAU,C,LDC,
     +             WORK,LWORK,INFO)
```

forms the matrix product $QC$.

The complex analogue of this routine is F08FSF (CHETRD/ZHETRD).

## 9. Example

To reduce the matrix $A$ to tridiagonal form, 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.**

```
*     F08FEF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. 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, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             A(LDA,NMAX), D(NMAX), E(NMAX-1), TAU(NMAX-1),
     +                 WORK(LWORK)
*     .. External Subroutines ..
      EXTERNAL         ssytrd
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08FEF 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
*
*        Reduce A to tridiagonal form
*
         CALL ssytrd(UPLO,N,A,LDA,D,E,TAU,WORK,LWORK,INFO)
*
*        Print tridiagonal form
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Diagonal'
         WRITE (NOUT,99999) (D(I),I=1,N)
         WRITE (NOUT,*) 'Off-diagonal'
         WRITE (NOUT,99999) (E(I),I=1,N-1)
      END IF
      STOP
*
99999 FORMAT (1X,8F9.4)
      END
```

## 9.2. Program Data

```
F08FEF 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

```
 F08FEF Example Program Results

 Diagonal
    2.0700   1.4741  -0.6492  -1.6949
 Off-diagonal
   -5.8258   2.6240   0.9163
```

# F08FFF (SORGTR/DORGTR) – 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

F08FFF (SORGTR/DORGTR) generates the real orthogonal matrix $Q$, which was determined by F08FEF (SSYTRD/DSYTRD) when reducing a symmetric matrix to tridiagonal form.

## 2. Specification

```
SUBROUTINE F08FFF (UPLO, N, A, LDA, TAU, WORK, LWORK, INFO)
ENTRY      sorgtr  (UPLO, N, A, LDA, TAU, WORK, LWORK, INFO)
INTEGER          N, LDA, LWORK, INFO
real             A(LDA,*), TAU(*), WORK(LWORK)
CHARACTER*1      UPLO
```

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 F08FEF (SSYTRD/DSYTRD), which reduces a real symmetric matrix $A$ to symmetric tridiagonal form $T$ by an orthogonal similarity transformation: $A = QTQ^T$. F08FEF represents the orthogonal matrix $Q$ as a product of $n-1$ elementary reflectors.

This routine may be used to generate $Q$ explicitly as a square matrix.

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §8.2.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

## 5. Parameters

1: UPLO – CHARACTER*1. *Input*

*On entry*: this **must** be the same parameter UPLO as supplied to F08FEF (SSYTRD/DSYTRD).

*Constraint*: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

*On entry*: $n$, the order of the matrix $Q$.

*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 vectors which define the elementary reflectors, as returned by F08FEF (SSYTRD/DSYTRD).

*On exit*: the $n$ by $n$ orthogonal matrix $Q$.

4: LDA – INTEGER. *Input*

*On entry*: the first dimension of the array A as declared in the (sub)program from which F08FFF (SORGTR/DORGTR) is called.

*Constraint*: LDA ≥ max(1,N).

5: TAU(*) – ***real*** array. *Input*

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

*On entry*: further details of the elementary reflectors, as returned by F08FEF (SSYTRD/DSYTRD).

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 F08FFF (SORGTR/DORGTR) is called.

*Suggested value*: for optimum performance LWORK should be at least (N–1)×*nb*, where *nb* is the ***blocksize***.

*Constraint*: LWORK ≥ max(1,N–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 $i$th 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 $\varepsilon$ is the ***machine precision***.

## 8. Further Comments

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

The complex analogue of this routine is F08FTF (CUNGTR/ZUNGTR).

## 9. Example

To compute all the eigenvalues and eigenvectors 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 and must first be reduced to tridiagonal form by F08FEF (SSYTRD/DSYTRD). The program then calls F08FFF (SORGTR/DORGTR) to form $Q$, and passes this matrix to F08JEF (SSTEQR/DSTEQR) which computes the eigenvalues and eigenvectors 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.**

```
*     F08FFF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, LWORK, LDZ
      PARAMETER        (NMAX=8,LDA=NMAX,LWORK=64*NMAX,LDZ=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             A(LDA,NMAX), D(NMAX), E(NMAX), TAU(NMAX),
     +                 WORK(LWORK), Z(LDZ,NMAX)
*     .. External Subroutines ..
      EXTERNAL         sorgtr, ssteqr, ssytrd, F06QFF, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08FFF 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
*
*        Reduce A to tridiagonal form T = (Q**T)*A*Q
*
         CALL ssytrd(UPLO,N,A,LDA,D,E,TAU,WORK,LWORK,INFO)
*
*        Copy A into Z
*
         CALL F06QFF(UPLO,N,N,A,LDA,Z,LDZ)
*
*        Form Q explicitly, storing the result in Z
*
         CALL sorgtr(UPLO,N,Z,LDZ,TAU,WORK,LWORK,INFO)
*
*        Calculate all the eigenvalues and eigenvectors of A
*
         CALL ssteqr('V',N,D,E,Z,LDZ,WORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.GT.0) THEN
            WRITE (NOUT,*) 'Failure to converge.'
         ELSE
*
*           Print eigenvalues and eigenvectors
*
            WRITE (NOUT,*) 'Eigenvalues'
            WRITE (NOUT,99999) (D(I),I=1,N)
            WRITE (NOUT,*)
            IFAIL = 0
*
            CALL X04CAF('General',' ',N,N,Z,LDZ,'Eigenvectors',IFAIL)
*
```

```
         END IF
      END IF
      STOP
*
99999 FORMAT (3X,(8F8.4))
      END
```

## 9.2. Program Data

```
F08FFF 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

```
 F08FFF Example Program Results

 Eigenvalues
    -5.0034 -1.9987  0.2013  8.0008

 Eigenvectors
          1       2       3       4
 1   0.5658 -0.2328 -0.3965  0.6845
 2  -0.3478  0.7994 -0.1780  0.4564
 3  -0.4740 -0.4087  0.5381  0.5645
 4   0.5781  0.3737  0.7221  0.0676
```

# F08FGF (SORMTR/DORMTR) – 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

F08FGF (SORMTR/DORMTR) multiplies an arbitrary real matrix $C$ by the real orthogonal matrix $Q$ which was determined by F08FEF (SSYTRD/DSYTRD) when reducing a real symmetric matrix to tridiagonal form.

## 2. Specification

```
 SUBROUTINE F08FGF (SIDE, UPLO, TRANS, M, N, A, LDA, TAU, C, LDC, WORK,
1                   LWORK, INFO)
 ENTRY       sormtr  (SIDE, UPLO, TRANS, M, N, A, LDA, TAU, C, LDC, WORK,
1                   LWORK, INFO)
 INTEGER          M, N, LDA, LDC, LWORK, INFO
 real             A(LDA,*), TAU(*), C(LDC,*), WORK(LWORK)
 CHARACTER*1      SIDE, UPLO, 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 F08FEF (SSYTRD/DSYTRD), which reduces a real symmetric matrix $A$ to symmetric tridiagonal form $T$ by an orthogonal similarity transformation: $A = QTQ^T$. F08FEF 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).

A common application of this routine is to transform a matrix $Z$ of eigenvectors of $T$ to the matrix $QZ$ of eigenvectors of $A$.

## 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: UPLO – CHARACTER*1. *Input*

*On entry*: this **must** be the same parameter UPLO as supplied to F08FEF (SSYTRD/DSYTRD).

*Constraint*: UPLO = 'U' or 'L'.

3: 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'.

4: M – INTEGER. *Input*

*On entry*: $m$, the number of rows of the matrix $C$; $m$ is also the order of $Q$ if SIDE = 'L'.

*Constraint*: M ≥ 0.

5: N – INTEGER. *Input*

*On entry*: $n$, the number of columns of the matrix $C$; $n$ is also the order of $Q$ if SIDE = 'R'.

*Constraint*: N ≥ 0.

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 F08FEF (SSYTRD/DSYTRD).

7: LDA – INTEGER. *Input*

*On entry*: the first dimension of the array A as declared in the (sub)program from which F08FGF (SORMTR/DORMTR) 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,M–1) if SIDE = 'L' and at least max(1,N–1) if SIDE = 'R'.

*On entry*: further details of the elementary reflectors, as returned by F08FEF (SSYTRD/DSYTRD).

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 F08FGF (SORMTR/DORMTR) 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 F08FGF (SORMTR/DORMTR) 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 $i$th 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 $\varepsilon$ is the ***machine precision***.

## 8. Further Comments

The total number of floating-point operations is approximately $2m^2n$ if SIDE = 'L' and $2mn^2$ if SIDE = 'R'.

The complex analogue of this routine is F08FUF (CUNMTR/ZUNMTR).

## 9. Example

To compute the two smallest eigenvalues, and the associated eigenvectors, 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 and must first be reduced to tridiagonal form $T$ by F08FEF (SSYTRD/DSYTRD). The program then calls F08JJF (SSTEBZ/DSTEBZ) to compute the requested eigenvalues and F08JKF (SSTEIN/DSTEIN) to compute the associated eigenvectors of $T$. Finally F08FGF (SORMTR/DORMTR) is called to transform the eigenvectors to those 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.**

```
*     F08FGF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, LDZ, LWORK
      PARAMETER        (NMAX=8,LDA=NMAX,LDZ=NMAX,LWORK=64*NMAX)
      real             ZERO
      PARAMETER        (ZERO=0.0e0)
*     .. Local Scalars ..
      real             VL, VU
      INTEGER          I, IFAIL, INFO, J, M, N, NSPLIT
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             A(LDA,NMAX), D(NMAX), E(NMAX), TAU(NMAX),
     +                 W(NMAX), WORK(LWORK), Z(LDZ,NMAX)
      INTEGER          IBLOCK(NMAX), IFAILV(NMAX), ISPLIT(NMAX),
     +                 IWORK(NMAX)
*     .. External Subroutines ..
      EXTERNAL         sormtr, sstebz, sstein, ssytrd, X04CAF
```

```
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08FGF 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
*
*        Reduce A to tridiagonal form T = (Q**T)*A*Q
*
         CALL ssytrd(UPLO,N,A,LDA,D,E,TAU,WORK,LWORK,INFO)
*
*        Calculate the two smallest eigenvalues of T (same as A)
*
         CALL sstebz('I','B',N,VL,VU,1,2,ZERO,D,E,M,NSPLIT,W,IBLOCK,
     +               ISPLIT,WORK,IWORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.GT.0) THEN
            WRITE (NOUT,*) 'Failure to converge.'
         ELSE
            WRITE (NOUT,*) 'Eigenvalues'
            WRITE (NOUT,99999) (W(I),I=1,M)
*
*           Calculate the eigenvectors of T, storing the result in Z
*
            CALL sstein(N,D,E,M,W,IBLOCK,ISPLIT,Z,LDZ,WORK,IWORK,IFAILV,
     +                  INFO)
*
            IF (INFO.GT.0) THEN
               WRITE (NOUT,*) 'Failure to converge.'
            ELSE
*
*              Calculate the eigenvectors of A = Q * (eigenvectors of T)
*
               CALL sormtr('Left',UPLO,'No transpose',N,M,A,LDA,TAU,Z,
     +                     LDZ,WORK,LWORK,INFO)
*
*              Print eigenvectors
*
               WRITE (NOUT,*)
               IFAIL = 0
*
               CALL X04CAF('General',' ',N,M,Z,LDZ,'Eigenvectors',IFAIL)
*
            END IF
         END IF
      END IF
      STOP
*
99999 FORMAT (3X,(9F8.4))
      END
```

## 9.2. Program Data

```
F08FGF 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

```
F08FGF Example Program Results

Eigenvalues
   -5.0034 -1.9987

Eigenvectors
          1       2
1    0.5658 -0.2328
2   -0.3478  0.7994
3   -0.4740 -0.4087
4    0.5781  0.3737
```

# F08FSF (CHETRD/ZHETRD) – 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

F08FSF (CHETRD/ZHETRD) reduces a complex Hermitian matrix to tridiagonal form.

## 2. Specification

```
SUBROUTINE F08FSF (UPLO, N, A, LDA, D, E, TAU, WORK, LWORK, INFO)
ENTRY      chetrd  (UPLO, N, A, LDA, D, E, TAU, WORK, LWORK, INFO)
INTEGER          N, LDA, LWORK, INFO
real             D(*), E(*)
complex          A(LDA,*), TAU(*), WORK(LWORK)
CHARACTER*1      UPLO
```

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

## 3. Description

This routine reduces a complex Hermitian matrix $A$ to real symmetric tridiagonal form $T$ by a unitary similarity transformation: $A = QTQ^H$.

The matrix $Q$ is not formed explicitly but is represented as a product of $n-1$ elementary reflectors (see the Chapter Introduction for details). Routines are provided to work with $Q$ in this representation (see Section 8).

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §8.2.
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 as follows:

if UPLO = 'U', then the upper triangular part of $A$ is stored;

if UPLO = 'L', then the lower triangular part of $A$ is stored.

*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*: $A$ is overwritten by the tridiagonal matrix $T$ and details of the unitary matrix $Q$ 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 F08FSF (CHETRD/ZHETRD) is called.

*Constraint*: LDA ≥ max(1,N).

5: D(*) – ***real*** array. *Output*

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

*On exit*: the diagonal elements of the tridiagonal matrix $T$.

6: E(*) – ***real*** array. *Output*

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

*On exit*: the off-diagonal elements of the tridiagonal matrix $T$.

7: TAU(*) – ***complex*** array. *Output*

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

*On exit*: further details of the unitary matrix $Q$.

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

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

9: LWORK – INTEGER. *Input*

*On entry*: the dimension of the array WORK as declared in the (sub)program from which F08FSF (CHETRD/ZHETRD) is called.

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

*Constraint*: LWORK ≥ 1.

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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

## 7. Accuracy

The computed tridiagonal matrix $T$ is exactly similar to a nearby matrix $A + E$, where

$$\|E\|_2 \leq c(n)\varepsilon\|A\|_2,$$

$c(n)$ is a modestly increasing function of $n$, and $\varepsilon$ is the ***machine precision***.

The elements of $T$ themselves may be sensitive to small perturbations in $A$ or to rounding errors in the computation, but this does not affect the stability of the eigenvalues and eigenvectors.

## 8. Further Comments

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

To form the unitary matrix $Q$ this routine may be followed by a call to F08FTF (CUNGTR/ZUNGTR):

```
CALL CUNGTR (UPLO,N,A,LDA,TAU,WORK,LWORK,INFO)
```

To apply $Q$ to an $n$ by $p$ complex matrix $C$ this routine may be followed by a call to F08FUF (CUNMTR/ZUNMTR). For example,

```
CALL CUNMTR ('Left',UPLO,'No Transpose',N,P,A,LDA,TAU,C,LDC,
+            WORK,LWORK,INFO)
```

forms the matrix product $QC$.

The real analogue of this routine is F08FEF (SSYTRD/DSYTRD).

## 9. Example

To reduce the matrix $A$ to tridiagonal form, where

$$A = \begin{pmatrix} -2.28 + 0.00i & 1.78 - 2.03i & 2.26 + 0.10i & -0.12 + 2.53i \\ 1.78 + 2.03i & -1.12 + 0.00i & 0.01 + 0.43i & -1.07 + 0.86i \\ 2.26 - 0.10i & 0.01 - 0.43i & -0.37 + 0.00i & 2.31 - 0.92i \\ -0.12 - 2.53i & -1.07 - 0.86i & 2.31 + 0.92i & -0.73 + 0.00i \end{pmatrix}.$$

### 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.**

```
*     F08FSF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. 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, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          A(LDA,NMAX), TAU(NMAX-1), WORK(LWORK)
      real             D(NMAX), E(NMAX-1)
*     .. External Subroutines ..
      EXTERNAL         chetrd
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08FSF 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
*
*        Reduce A to tridiagonal form
*
         CALL chetrd(UPLO,N,A,LDA,D,E,TAU,WORK,LWORK,INFO)
*
*        Print tridiagonal form
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Diagonal'
         WRITE (NOUT,99999) (D(I),I=1,N)
         WRITE (NOUT,*) 'Off-diagonal'
         WRITE (NOUT,99999) (E(I),I=1,N-1)
      END IF
      STOP
*
```

```
99999 FORMAT (1X,8F9.4)
      END
```

## 9.2. Program Data

```
F08FSF Example Program Data
  4                                                                  :Value of N
  'L'                                                                :Value of UPLO
 (-2.28, 0.00)
 ( 1.78, 2.03) (-1.12, 0.00)
 ( 2.26,-0.10) ( 0.01,-0.43) (-0.37, 0.00)
 (-0.12,-2.53) (-1.07,-0.86) ( 2.31, 0.92) (-0.73, 0.00)    :End of matrix A
```

## 9.3. Program Results

```
 F08FSF Example Program Results

 Diagonal
   -2.2800  -0.1285  -0.1666  -1.9249
 Off-diagonal
   -4.3385  -2.0226  -1.8023
```

# F08FTF (CUNGTR/ZUNGTR) – 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

F08FTF (CUNGTR/ZUNGTR) generates the complex unitary matrix $Q$, which was determined by F08FSF (CHETRD/ZHETRD) when reducing a Hermitian matrix to tridiagonal form.

## 2. Specification

```
SUBROUTINE F08FTF (UPLO, N, A, LDA, TAU, WORK, LWORK, INFO)
ENTRY      cungtr (UPLO, N, A, LDA, TAU, WORK, LWORK, INFO)
INTEGER          N, LDA, LWORK, INFO
complex          A(LDA,*), TAU(*), WORK(LWORK)
CHARACTER*1      UPLO
```

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 F08FSF (CHETRD/ZHETRD), which reduces a complex Hermitian matrix $A$ to real symmetric tridiagonal form $T$ by a unitary similarity transformation: $A = QTQ^H$. F08FSF represents the unitary matrix $Q$ as a product of $n-1$ elementary reflectors.

This routine may be used to generate $Q$ explicitly as a square matrix.

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §8.2.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

## 5. Parameters

1: UPLO – CHARACTER*1. *Input*

*On entry*: this **must** be the same parameter UPLO as supplied to F08FSF (CHETRD/ZHETRD).

*Constraint*: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

*On entry*: $n$, the order of the matrix $Q$.

*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 vectors which define the elementary reflectors, as returned by F08FSF (CHETRD/ZHETRD).

*On exit*: the $n$ by $n$ unitary matrix $Q$.

4: LDA – INTEGER. *Input*

*On entry*: the first dimension of the array A as declared in the (sub)program from which F08FTF (CUNGTR/ZUNGTR) is called.

*Constraint*: LDA ≥ max(1,N).

5: TAU(*) – ***complex*** array. *Input*

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

*On entry*: further details of the elementary reflectors, as returned by F08FSF (CHETRD/ZHETRD).

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 F08FTF (CUNGTR/ZUNGTR) is called.

*Suggested value*: for optimum performance LWORK should be at least (N–1)×*nb*, where *nb* is the ***blocksize***.

*Constraint*: LWORK ≥ max(1,N–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 $i$th 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 $\varepsilon$ is the ***machine precision***.

## 8. Further Comments

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

The real analogue of this routine is F08FFF (SORGTR/DORGTR).

## 9. Example

To compute all the eigenvalues and eigenvectors of the matrix $A$, where

$$A = \begin{pmatrix} -2.28 + 0.00i & 1.78 - 2.03i & 2.26 + 0.10i & -0.12 + 2.53i \\ 1.78 + 2.03i & -1.12 + 0.00i & 0.01 + 0.43i & -1.07 + 0.86i \\ 2.26 - 0.10i & 0.01 - 0.43i & -0.37 + 0.00i & 2.31 - 0.92i \\ -0.12 - 2.53i & -1.07 - 0.86i & 2.31 + 0.92i & -0.73 + 0.00i \end{pmatrix}.$$

Here $A$ is Hermitian and must first be reduced to tridiagonal form by F08FSF (CHETRD/ZHETRD). The program then calls F08FTF (CUNGTR/ZUNGTR) to form $Q$, and passes this matrix to F08JSF (CSTEQR/ZSTEQR) which computes the eigenvalues and eigenvectors 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.**

```
*     F08FTF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, LWORK, LDZ
      PARAMETER        (NMAX=8,LDA=NMAX,LWORK=64*NMAX,LDZ=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          A(LDA,NMAX), TAU(NMAX), WORK(LWORK), Z(LDZ, NMAX)
      real             D(NMAX), E(NMAX), RWORK(2*NMAX-2)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         F06TFF, X04DBF, chetrd, csteqr, cungtr
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08FTF 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
*
*        Reduce A to tridiagonal form T = (Q**H)*A*Q
*
         CALL chetrd(UPLO,N,A,LDA,D,E,TAU,WORK,LWORK,INFO)
*
*        Copy A into Z
*
         CALL F06TFF(UPLO,N,N,A,LDA,Z,LDZ)
*
*        Form Q explicitly, storing the result in Z
*
         CALL cungtr(UPLO,N,Z,LDZ,TAU,WORK,LWORK,INFO)
*
*        Calculate all the eigenvalues and eigenvectors of A
*
         CALL csteqr('V',N,D,E,Z,LDZ,RWORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.GT.0) THEN
            WRITE (NOUT,*) 'Failure to converge.'
         ELSE
*
*           Print eigenvalues and eigenvectors
*
            WRITE (NOUT,*) 'Eigenvalues'
            WRITE (NOUT,99999) (D(I),I=1,N)
            WRITE (NOUT,*)
            IFAIL = 0
*
            CALL X04DBF('General',' ',N,N,Z,LDZ,'Bracketed','F7.4',
     +                  'Eigenvectors','Integer',RLABS,'Integer',CLABS,
     +                  80,0,IFAIL)
*
```

```
         END IF
      END IF
      STOP
*
99999 FORMAT (8X,4(F7.4,11X,:))
      END
```

## 9.2. Program Data

```
F08FTF Example Program Data
  4                                                              :Value of N
  'L'                                                            :Value of UPLO
 (-2.28, 0.00)
 ( 1.78, 2.03) (-1.12, 0.00)
 ( 2.26,-0.10) ( 0.01,-0.43) (-0.37, 0.00)
 (-0.12,-2.53) (-1.07,-0.86) ( 2.31, 0.92) (-0.73, 0.00)    :End of matrix A
```

## 9.3. Program Results

```
F08FTF Example Program Results

Eigenvalues
       -6.0002           -3.0030            0.5036            3.9996

Eigenvectors
                     1                   2                   3                   4
1   ( 0.7299, 0.0000) (-0.2120, 0.1497) ( 0.1000,-0.3570) ( 0.1991, 0.4720)
2   (-0.1663,-0.2061) ( 0.7307, 0.0000) ( 0.2863,-0.3353) (-0.2467, 0.3751)
3   (-0.4165,-0.1417) (-0.3291, 0.0479) ( 0.6890, 0.0000) ( 0.4468, 0.1466)
4   ( 0.1743, 0.4162) ( 0.5200, 0.1329) ( 0.0662, 0.4347) ( 0.5612, 0.0000)
```

# F08FUF (CUNMTR/ZUNMTR) – 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

F08FUF (CUNMTR/ZUNMTR) multiplies an arbitrary complex matrix $C$ by the complex unitary matrix $Q$ which was determined by F08FSF (CHETRD/ZHETRD) when reducing a complex Hermitian matrix to tridiagonal form.

## 2. Specification

```
 SUBROUTINE F08FUF (SIDE, UPLO, TRANS, M, N, A, LDA, TAU, C, LDC, WORK,
1                   LWORK, INFO)
 ENTRY      cunmtr (SIDE, UPLO, TRANS, M, N, A, LDA, TAU, C, LDC, WORK,
1                   LWORK, INFO)
 INTEGER          M, N, LDA, LDC, LWORK, INFO
 complex          A(LDA,*), TAU(*), C(LDC,*), WORK(LWORK)
 CHARACTER*1      SIDE, UPLO, 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 F08FSF (CHETRD/ZHETRD), which reduces a complex Hermitian matrix $A$ to real symmetric tridiagonal form $T$ by a unitary similarity transformation: $A = QTQ^H$. F08FSF 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).

A common application of this routine is to transform a matrix $Z$ of eigenvectors of $T$ to the matrix $QZ$ of eigenvectors of $A$.

## 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: UPLO – CHARACTER*1. *Input*

*On entry*: this **must** be the same parameter UPLO as supplied to F08FSF (CHETRD/ZHETRD).

*Constraint*: UPLO = 'U' or 'L'.

3: 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'.

4: M – INTEGER. *Input*

*On entry*: $m$, the number of rows of the matrix $C$; $m$ is also the order of $Q$ if SIDE = 'L'.

*Constraint*: M ≥ 0.

5: N – INTEGER. *Input*

*On entry*: $n$, the number of columns of the matrix $C$; $n$ is also the order of $Q$ if SIDE = 'R'.

*Constraint*: N ≥ 0.

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 F08FSF (CHETRD/ZHETRD).

7: LDA – INTEGER. *Input*

*On entry*: the first dimension of the array A as declared in the (sub)program from which F08FUF (CUNMTR/ZUNMTR) 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,M–1) if SIDE = 'L' and at least max(1,N–1) if SIDE = 'R'.

*On entry*: further details of the elementary reflectors, as returned by F08FSF (CHETRD/ZHETRD).

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 F08FUF (CUNMTR/ZUNMTR) 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 F08FUF (CUNMTR/ZUNMTR) 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 $i$th 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 $\varepsilon$ is the ***machine precision***.

## 8. Further Comments

The total number of real floating-point operations is approximately $8m^2n$ if SIDE = 'L' and $8mn^2$ if SIDE = 'R'.

The real analogue of this routine is F08FGF (SORMTR/DORMTR).

## 9. Example

To compute the two smallest eigenvalues, and the associated eigenvectors, of the matrix $A$, where

$$A = \begin{pmatrix} -2.28 + 0.00i & 1.78 - 2.03i & 2.26 + 0.10i & -0.12 + 2.53i \\ 1.78 + 2.03i & -1.12 + 0.00i & 0.01 + 0.43i & -1.07 + 0.86i \\ 2.26 - 0.10i & 0.01 - 0.43i & -0.37 + 0.00i & 2.31 - 0.92i \\ -0.12 - 2.53i & -1.07 - 0.86i & 2.31 + 0.92i & -0.73 + 0.00i \end{pmatrix}.$$

Here $A$ is Hermitian and must first be reduced to tridiagonal form $T$ by F08FSF (CHETRD/ZHETRD). The program then calls F08JJF (SSTEBZ/DSTEBZ) to compute the requested eigenvalues and F08JXF (CSTEIN/ZSTEIN) to compute the associated eigenvectors of $T$. Finally F08FUF (CUNMTR/ZUNMTR) is called to transform the eigenvectors to those 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.**

```
*     F08FUF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, LDZ, LWORK
      PARAMETER        (NMAX=8,LDA=NMAX,LDZ=NMAX,LWORK=64*NMAX)
      real             ZERO
      PARAMETER        (ZERO=0.0e0)
*     .. Local Scalars ..
      real             VL, VU
      INTEGER          I, IFAIL, INFO, J, M, N, NSPLIT
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          A(LDA,NMAX), TAU(NMAX), WORK(LWORK), Z(LDZ, NMAX)
      real             D(NMAX), E(NMAX), RWORK(5*NMAX), W(NMAX)
      INTEGER          IBLOCK(NMAX), IFAILV(NMAX), ISPLIT(NMAX),
     +                 IWORK(NMAX)
      CHARACTER        CLABS(1), RLABS(1)
```

```
*     .. External Subroutines ..
      EXTERNAL          sstebz, X04DBF, chetrd, cstein, cunmtr
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08FUF 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
*
*        Reduce A to tridiagonal form T = (Q**H)*A*Q
*
         CALL chetrd(UPLO,N,A,LDA,D,E,TAU,WORK,LWORK,INFO)
*
*        Calculate the two smallest eigenvalues of T (same as A)
*
         CALL sstebz('I','B',N,VL,VU,1,2,ZERO,D,E,M,NSPLIT,W,IBLOCK,
     +               ISPLIT,RWORK,IWORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.GT.0) THEN
            WRITE (NOUT,*) 'Failure to converge.'
         ELSE
            WRITE (NOUT,*) 'Eigenvalues'
            WRITE (NOUT,99999) (W(I),I=1,M)
*
*           Calculate the eigenvectors of T, storing the result in Z
*
            CALL cstein(N,D,E,M,W,IBLOCK,ISPLIT,Z,LDZ,RWORK,IWORK,
     +                  IFAILV,INFO)
*
            IF (INFO.GT.0) THEN
               WRITE (NOUT,*) 'Failure to converge.'
            ELSE
*
*              Calculate the eigenvectors of A = Q * (eigenvectors of T)
*
               CALL cunmtr('Left',UPLO,'No transpose',N,M,A,LDA,TAU,Z,
     +                     LDZ,WORK,LWORK,INFO)
*
*              Print eigenvectors
*
               WRITE (NOUT,*)
               IFAIL = 0
*
               CALL X04DBF('General',' ',N,M,Z,LDZ,'Bracketed','F7.4',
     +                     'Eigenvectors','Integer',RLABS,'Integer',
     +                     CLABS,80,0,IFAIL)
*
            END IF
         END IF
      END IF
      STOP
*
99999 FORMAT (8X,4(F7.4,11X,:))
      END
```

## 9.2. Program Data

```
F08FUF Example Program Data
  4                                                                  :Value of N
  'L'                                                                :Value of UPLO
 (-2.28, 0.00)
 ( 1.78, 2.03) (-1.12, 0.00)
 ( 2.26,-0.10) ( 0.01,-0.43) (-0.37, 0.00)
 (-0.12,-2.53) (-1.07,-0.86) ( 2.31, 0.92) (-0.73, 0.00)    :End of matrix A
```

## 9.3. Program Results

```
F08FUF Example Program Results

Eigenvalues
       -6.0002           -3.0030

Eigenvectors
                     1                    2
1   ( 0.7299, 0.0000) (-0.2595, 0.0000)
2   (-0.1663,-0.2061) ( 0.5969, 0.4214)
3   (-0.4165,-0.1417) (-0.2965,-0.1507)
4   ( 0.1743, 0.4162) ( 0.3482, 0.4085)
```

# F08GEF (SSPTRD/DSPTRD) – 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

F08GEF (SSPTRD/DSPTRD) reduces a real symmetric matrix to tridiagonal form, using packed storage.

## 2. Specification

```
SUBROUTINE F08GEF (UPLO, N, AP, D, E, TAU, INFO)
ENTRY      ssptrd  (UPLO, N, AP, D, E, TAU, INFO)
INTEGER            N, INFO
real               AP(*), D(*), E(*), TAU(*)
CHARACTER*1        UPLO
```

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

## 3. Description

This routine reduces a real symmetric matrix $A$, held in packed storage, to symmetric tridiagonal form $T$ by an orthogonal similarity transformation: $A = QTQ^T$.

The matrix $Q$ is not formed explicitly but is represented as a product of $n-1$ elementary reflectors (see the Chapter Introduction for details). Routines are provided to work with $Q$ in this representation (see Section 8).

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §8.2.
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 as follows:

if UPLO = 'U', then the upper triangular part of $A$ is stored;

if UPLO = 'L', then the lower triangular part of $A$ is stored.

*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*: 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($i$+(2$n$–$j$)($j$–1)/2) for $i \ge j$.

*On exit*: $A$ is overwritten by the tridiagonal matrix $T$ and details of the orthogonal matrix $Q$.

4: D(*) – ***real*** array. *Output*

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

*On exit*: the diagonal elements of the tridiagonal matrix $T$.

5: E(*) – ***real*** array. *Output*

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

*On exit*: the off-diagonal elements of the tridiagonal matrix $T$.

6: TAU(*) – ***real*** array. *Output*

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

*On exit*: further details of the orthogonal matrix $Q$.

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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

## 7. Accuracy

The computed tridiagonal matrix $T$ is exactly similar to a nearby matrix $A + E$, where

$$\|E\|_2 \leq c(n)\varepsilon\|A\|_2,$$

$c(n)$ is a modestly increasing function of $n$, and $\varepsilon$ is the ***machine precision***.

The elements of $T$ themselves may be sensitive to small perturbations in $A$ or to rounding errors in the computation, but this does not affect the stability of the eigenvalues and eigenvectors.

## 8. Further Comments

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

To form the orthogonal matrix $Q$ this routine may be followed by a call to F08GFF (SOPGTR/DOPGTR):

```
CALL SOPGTR (UPLO,N,AP,TAU,Q,LDQ,WORK,INFO)
```

To apply $Q$ to an $n$ by $p$ real matrix $C$ this routine may be followed by a call to F08GGF (SOPMTR/DOPMTR). For example,

```
CALL SOPMTR ('Left',UPLO,'No Transpose',N,P,AP,TAU,C,LDC,WORK,
+            INFO)
```

forms the matrix product $QC$.

The complex analogue of this routine is F08GSF (CHPTRD/ZHPTRD).

## 9. Example

To reduce the matrix $A$ to tridiagonal form, 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.**

```
*     F08GEF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=8)
*     .. Local Scalars ..
      INTEGER          I, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             AP(NMAX*(NMAX+1)/2), D(NMAX), E(NMAX-1),
     +                 TAU(NMAX-1)
*     .. External Subroutines ..
      EXTERNAL         ssptrd
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08GEF 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
*
*        Reduce A to tridiagonal form
*
         CALL ssptrd(UPLO,N,AP,D,E,TAU,INFO)
*
*        Print tridiagonal form
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Diagonal'
         WRITE (NOUT,99999) (D(I),I=1,N)
         WRITE (NOUT,*) 'Off-diagonal'
         WRITE (NOUT,99999) (E(I),I=1,N-1)
      END IF
      STOP
*
99999 FORMAT (1X,8F9.4)
      END
```

## 9.2. Program Data

```
F08GEF 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

```
F08GEF Example Program Results

Diagonal
   2.0700    1.4741   -0.6492   -1.6949
Off-diagonal
  -5.8258    2.6240    0.9163
```

# F08GFF (SOPGTR/DOPGTR) – 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

F08GFF (SOPGTR/DOPGTR) generates the real orthogonal matrix $Q$, which was determined by F08GEF (SSPTRD/DSPTRD) when reducing a symmetric matrix to tridiagonal form.

## 2. Specification

```
SUBROUTINE F08GFF (UPLO, N, AP, TAU, Q, LDQ, WORK, INFO)
ENTRY      sopgtr  (UPLO, N, AP, TAU, Q, LDQ, WORK, INFO)
INTEGER          N, LDQ, INFO
real             AP(*), TAU(*), Q(LDQ,*), WORK(*)
CHARACTER*1      UPLO
```

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 F08GEF (SSPTRD/DSPTRD), which reduces a real symmetric matrix $A$ to symmetric tridiagonal form $T$ by an orthogonal similarity transformation: $A = QTQ^T$. F08GEF represents the orthogonal matrix $Q$ as a product of $n-1$ elementary reflectors.

This routine may be used to generate $Q$ explicitly as a square matrix.

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §8.2.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

## 5. Parameters

1: UPLO – CHARACTER*1. *Input*

*On entry*: this **must** be the same parameter UPLO as supplied to F08GEF (SSPTRD/DSPTRD).

*Constraint*: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

*On entry*: $n$, the order of the matrix $Q$.

*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 vectors which define the elementary reflectors, as returned by F08GEF (SSPTRD/DSPTRD).

4: TAU(*) – ***real*** array. *Input*

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

*On entry*: further details of the elementary reflectors, as returned by F08GEF (SSPTRD/DSPTRD).

5: Q(LDQ,*) – ***real*** array. *Output*

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

*On exit*: the $n$ by $n$ orthogonal matrix $Q$.

6: LDQ – INTEGER. *Input*

*On entry*: the first dimension of the array Q as declared in the (sub)program from which F08GFF (SOPGTR/DOPGTR) is called.

*Constraint*: LDQ ≥ max(1,N).

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

**Note**: the dimension of the array WORK must be at least max(1,N–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 $i$th 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 $\varepsilon$ is the ***machine precision***.

## 8. Further Comments

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

The complex analogue of this routine is F08GTF (CUPGTR/ZUPGTR).

## 9. Example

To compute all the eigenvalues and eigenvectors 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. Here $A$ is symmetric and must first be reduced to tridiagonal form by F08GEF (SSPTRD/DSPTRD). The program then calls F08GFF (SOPGTR/DOPGTR) to form $Q$, and passes this matrix to F08JEF (SSTEQR/DSTEQR) which computes the eigenvalues and eigenvectors 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.

```
*     F08GFF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDZ
      PARAMETER        (NMAX=8,LDZ=NMAX)
```

```
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             AP(NMAX*(NMAX+1)/2), D(NMAX), E(NMAX), TAU(NMAX),
     +                 WORK(2*NMAX-2), Z(LDZ,NMAX)
*     .. External Subroutines ..
      EXTERNAL         sopgtr, ssptrd, ssteqr, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08GFF 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
*
*        Reduce A to tridiagonal form T = (Q**T)*A*Q
*
         CALL ssptrd(UPLO,N,AP,D,E,TAU,INFO)
*
*        Form Q explicitly, storing the result in Z
*
         CALL sopgtr(UPLO,N,AP,TAU,Z,LDZ,WORK,INFO)
*
*        Calculate all the eigenvalues and eigenvectors of A
*
         CALL ssteqr('V',N,D,E,Z,LDZ,WORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.GT.0) THEN
            WRITE (NOUT,*) 'Failure to converge.'
         ELSE
*
*           Print eigenvalues and eigenvectors
*
            WRITE (NOUT,*) 'Eigenvalues'
            WRITE (NOUT,99999) (D(I),I=1,N)
            WRITE (NOUT,*)
            IFAIL = 0
*
            CALL X04CAF('General',' ',N,N,Z,LDZ,'Eigenvectors',IFAIL)
*
         END IF
      END IF
      STOP
*
99999 FORMAT (3X,(8F8.4))
      END
```

## 9.2. Program Data

```
F08GFF 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

```
F08GFF Example Program Results

Eigenvalues
   -5.0034 -1.9987  0.2013  8.0008

Eigenvectors
          1       2       3       4
1   0.5658 -0.2328 -0.3965  0.6845
2  -0.3478  0.7994 -0.1780  0.4564
3  -0.4740 -0.4087  0.5381  0.5645
4   0.5781  0.3737  0.7221  0.0676
```

# F08GGF (SOPMTR/DOPMTR) – 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

F08GGF (SOPMTR/DOPMTR) multiplies an arbitrary real matrix $C$ by the real orthogonal matrix $Q$ which was determined by F08GEF (SSPTRD/DSPTRD) when reducing a real symmetric matrix to tridiagonal form.

## 2. Specification

```
SUBROUTINE F08GGF (SIDE, UPLO, TRANS, M, N, AP, TAU, C, LDC, WORK, INFO)
ENTRY      sopmtr (SIDE, UPLO, TRANS, M, N, AP, TAU, C, LDC, WORK, INFO)
INTEGER          M, N, LDC, INFO
real             AP(*), TAU(*), C(LDC,*), WORK(*)
CHARACTER*1      SIDE, UPLO, 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 F08GEF (SSPTRD/DSPTRD), which reduces a real symmetric matrix $A$ to symmetric tridiagonal form $T$ by an orthogonal similarity transformation: $A = QTQ^T$. F08GEF 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).

A common application of this routine is to transform a matrix $Z$ of eigenvectors of $T$ to the matrix $QZ$ of eigenvectors of $A$.

## 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: UPLO – CHARACTER*1. *Input*

*On entry*: this **must** be the same parameter UPLO as supplied to F08GEF (SSPTRD/DSPTRD).

*Constraint*: UPLO = 'U' or 'L'.

3: 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'.

4: M – INTEGER. *Input*

*On entry*: $m$, the number of rows of the matrix $C$; $m$ is also the order of $Q$ if SIDE = 'L'.

*Constraint*: M ≥ 0.

5: N – INTEGER. *Input*

*On entry*: $n$, the number of columns of the matrix $C$; $n$ is also the order of $Q$ if SIDE = 'R'.

*Constraint*: N ≥ 0.

6: AP(*) – ***real*** array. *Input*

**Note**: the dimension of the array AP must be at least max(1,M*(M+1)/2) if SIDE = 'L' and at least max(1,N*(N+1)/2) if SIDE = 'R'.

*On entry*: details of the vectors which define the elementary reflectors, as returned by F08GEF (SSPTRD/DSPTRD).

7: TAU(*) – ***real*** array. *Input*

**Note**: the dimension of the array TAU must be at least max(1,M–1) if SIDE = 'L' and at least max(1,N–1) if SIDE = 'R'.

*On entry*: further details of the elementary reflectors, as returned by F08GEF (SSPTRD/DSPTRD).

8: 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.

9: LDC – INTEGER. *Input*

*On entry*: the first dimension of the array C as declared in the (sub)program from which F08GGF (SOPMTR/DOPMTR) is called.

*Constraint*: LDC ≥ max(1,M).

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

**Note**: the dimension of the array WORK must be at least max(1,N) if SIDE = 'L' and at least max(1,M) if SIDE = 'R'.

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 $i$th 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 $\varepsilon$ is the ***machine precision***.

## 8. Further Comments

The total number of floating-point operations is approximately $2m^2n$ if SIDE = 'L' and $2mn^2$ if SIDE = 'R'.

The complex analogue of this routine is F08GUF (CUPMTR/ZUPMTR).

## 9. Example

To compute the two smallest eigenvalues, and the associated eigenvectors, 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. Here $A$ is symmetric and must first be reduced to tridiagonal form $T$ by F08GEF (SSPTRD/DSPTRD). The program then calls F08JJF (SSTEBZ/DSTEBZ) to compute the requested eigenvalues and F08JKF (SSTEIN/DSTEIN) to compute the associated eigenvectors of $T$. Finally F08GGF (SOPMTR/DOPMTR) is called to transform the eigenvectors to those 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.

```
*     F08GGF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDZ
      PARAMETER        (NMAX=8,LDZ=NMAX)
      real             ZERO
      PARAMETER        (ZERO=0.0e0)
*     .. Local Scalars ..
      real             VL, VU
      INTEGER          I, IFAIL, INFO, J, M, N, NSPLIT
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             AP(NMAX*(NMAX+1)/2), D(NMAX), E(NMAX), TAU(NMAX),
     +                 W(NMAX), WORK(5*NMAX), Z(LDZ,NMAX)
      INTEGER          IBLOCK(NMAX), IFAILV(NMAX), ISPLIT(NMAX),
     +                 IWORK(NMAX)
*     .. External Subroutines ..
      EXTERNAL         sopmtr, ssptrd, sstebz, sstein, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08GGF 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
*
```

```
*        Reduce A to tridiagonal form T = (Q**T)*A*Q
*
         CALL ssptrd(UPLO,N,AP,D,E,TAU,INFO)
*
*        Calculate the two smallest eigenvalues of T (same as A)
*
         CALL sstebz('I','B',N,VL,VU,1,2,ZERO,D,E,M,NSPLIT,W,IBLOCK,
     +               ISPLIT,WORK,IWORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.GT.0) THEN
            WRITE (NOUT,*) 'Failure to converge.'
         ELSE
            WRITE (NOUT,*) 'Eigenvalues'
            WRITE (NOUT,99999) (W(I),I=1,M)
*
*           Calculate the eigenvectors of T, storing the result in Z
*
            CALL sstein(N,D,E,M,W,IBLOCK,ISPLIT,Z,LDZ,WORK,IWORK,IFAILV,
     +                  INFO)
*
            IF (INFO.GT.0) THEN
               WRITE (NOUT,*) 'Failure to converge.'
            ELSE
*
*              Calculate the eigenvectors of A = Q * (eigenvectors of T)
*
               CALL sopmtr('Left',UPLO,'No transpose',N,M,AP,TAU,Z,LDZ,
     +                     WORK,INFO)
*
*              Print eigenvectors
*
               WRITE (NOUT,*)
               IFAIL = 0
*
               CALL X04CAF('General',' ',N,M,Z,LDZ,'Eigenvectors',IFAIL)
*
            END IF
         END IF
      END IF
      STOP
*
99999 FORMAT (3X,(9F8.4))
      END
```

## 9.2. Program Data

```
F08GGF 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

```
 F08GGF Example Program Results

 Eigenvalues
    -5.0034 -1.9987

 Eigenvectors
            1       2
 1    0.5658 -0.2328
 2   -0.3478  0.7994
 3   -0.4740 -0.4087
 4    0.5781  0.3737
```

# F08GSF (CHPTRD/ZHPTRD) – 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

F08GSF (CHPTRD/ZHPTRD) reduces a complex Hermitian matrix to tridiagonal form, using packed storage.

## 2. Specification

```
SUBROUTINE F08GSF (UPLO, N, AP, D, E, TAU, INFO)
ENTRY      chptrd  (UPLO, N, AP, D, E, TAU, INFO)

INTEGER          N, INFO
real             D(*), E(*)
complex          AP(*), TAU(*)
CHARACTER*1      UPLO
```

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

## 3. Description

This routine reduces a complex Hermitian matrix $A$, held in packed storage, to real symmetric tridiagonal form $T$ by a unitary similarity transformation: $A = QTQ^H$.

The matrix $Q$ is not formed explicitly but is represented as a product of $n-1$ elementary reflectors (see the Chapter Introduction for details). Routines are provided to work with $Q$ in this representation (see Section 8).

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §8.2.
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 as follows:

if UPLO = 'U', then the upper triangular part of $A$ is stored;

if UPLO = 'L', then the lower triangular part of $A$ is stored.

*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 $n$ 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 $i \le j$; if UPLO = 'L', the lower triangle of $A$ must be stored with element $a_{ij}$ in AP($i$+(2$n$–$j$)($j$–1)/2) for $i \ge j$.

*On exit*: $A$ is overwritten by the tridiagonal matrix $T$ and details of the unitary matrix $Q$.

4: D(*) – ***real*** array. *Output*

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

*On exit*: the diagonal elements of the tridiagonal matrix $T$.

5: E(*) – ***real*** array. *Output*

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

*On exit*: the off-diagonal elements of the tridiagonal matrix $T$.

6: TAU(*) – ***complex*** array. *Output*

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

*On exit*: further details of the unitary matrix $Q$.

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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

## 7. Accuracy

The computed tridiagonal matrix $T$ is exactly similar to a nearby matrix $A + E$, where

$\|E\|_2 \leq c(n)\varepsilon\|A\|_2$,

$c(n)$ is a modestly increasing function of $n$, and $\varepsilon$ is the ***machine precision***.

The elements of $T$ themselves may be sensitive to small perturbations in $A$ or to rounding errors in the computation, but this does not affect the stability of the eigenvalues and eigenvectors.

## 8. Further Comments

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

To form the unitary matrix $Q$ this routine may be followed by a call to F08GTF (CUPGTR/ZUPGTR):

```
CALL CUPGTR (UPLO,N,AP,TAU,Q,LDQ,WORK,INFO)
```

To apply $Q$ to an $n$ by $p$ complex matrix $C$ this routine may be followed by a call to F08GUF (CUPMTR/ZUPMTR). For example,

```
CALL CUPMTR ('Left',UPLO,'No Transpose',N,P,AP,TAU,C,LDC,WORK,
+            INFO)
```

forms the matrix product $QC$.

The real analogue of this routine is F08GEF (SSPTRD/DSPTRD).

## 9. Example

To reduce the matrix $A$ to tridiagonal form, where

$$A = \begin{pmatrix} -2.28 + 0.00i & 1.78 - 2.03i & 2.26 + 0.10i & -0.12 + 2.53i \\ 1.78 + 2.03i & -1.12 + 0.00i & 0.01 + 0.43i & -1.07 + 0.86i \\ 2.26 - 0.10i & 0.01 - 0.43i & -0.37 + 0.00i & 2.31 - 0.92i \\ -0.12 - 2.53i & -1.07 - 0.86i & 2.31 + 0.92i & -0.73 + 0.00i \end{pmatrix},$$

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.**

```
*     F08GSF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=8)
*     .. Local Scalars ..
      INTEGER          I, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          AP(NMAX*(NMAX+1)/2), TAU(NMAX-1)
      real             D(NMAX), E(NMAX-1)
*     .. External Subroutines ..
      EXTERNAL         chptrd
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08GSF 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
*
*        Reduce A to tridiagonal form
*
         CALL chptrd(UPLO,N,AP,D,E,TAU,INFO)
*
*        Print tridiagonal form
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Diagonal'
         WRITE (NOUT,99999) (D(I),I=1,N)
         WRITE (NOUT,*) 'Off-diagonal'
         WRITE (NOUT,99999) (E(I),I=1,N-1)
      END IF
      STOP
*
99999 FORMAT (1X,8F9.4)
      END
```

## 9.2. Program Data

```
F08GSF Example Program Data
  4                                                                 :Value of N
  'L'                                                               :Value of UPLO
 (-2.28, 0.00)
 ( 1.78, 2.03) (-1.12, 0.00)
 ( 2.26,-0.10) ( 0.01,-0.43) (-0.37, 0.00)
 (-0.12,-2.53) (-1.07,-0.86) ( 2.31, 0.92) (-0.73, 0.00)    :End of matrix A
```

## 9.3. Program Results

```
F08GSF Example Program Results

Diagonal
  -2.2800  -0.1285  -0.1666  -1.9249
Off-diagonal
  -4.3385  -2.0226  -1.8023
```

# F08GTF (CUPGTR/ZUPGTR) – 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

F08GTF (CUPGTR/ZUPGTR) generates the complex unitary matrix $Q$, which was determined by F08GSF (CHPTRD/ZHPTRD) when reducing a Hermitian matrix to tridiagonal form.

## 2. Specification

```
SUBROUTINE F08GTF (UPLO, N, AP, TAU, Q, LDQ, WORK, INFO)
ENTRY      cupgtr (UPLO, N, AP, TAU, Q, LDQ, WORK, INFO)
INTEGER          N, LDQ, INFO
complex          AP(*), TAU(*), Q(LDQ,*), WORK(*)
CHARACTER*1      UPLO
```

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 F08GSF (CHPTRD/ZHPTRD), which reduces a complex Hermitian matrix $A$ to real symmetric tridiagonal form $T$ by a unitary similarity transformation: $A = QTQ^H$. F08GSF represents the unitary matrix $Q$ as a product of $n-1$ elementary reflectors.

This routine may be used to generate $Q$ explicitly as a square matrix.

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §8.2.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

## 5. Parameters

1: UPLO – CHARACTER*1. *Input*

*On entry*: this **must** be the same parameter UPLO as supplied to F08GSF (CHPTRD/ZHPTRD).

*Constraint*: UPLO = 'U' or 'L'.

2: N – INTEGER. *Input*

*On entry*: $n$, the order of the matrix $Q$.

*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 vectors which define the elementary reflectors, as returned by F08GSF (CHPTRD/ZHPTRD).

4: TAU(*) – ***complex*** array. *Input*

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

*On entry*: further details of the elementary reflectors, as returned by F08GSF (CHPTRD/ZHPTRD).

5: Q(LDQ,*) – ***complex*** array. *Output*

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

*On exit*: the $n$ by $n$ unitary matrix $Q$.

6: LDQ – INTEGER. *Input*

*On entry*: the first dimension of the array Q as declared in the (sub)program from which F08GTF (CUPGTR/ZUPGTR) is called.

*Constraint*: LDQ ≥ max(1,N).

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

**Note**: the dimension of the array WORK must be at least max(1,N–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 $i$th 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 $\varepsilon$ is the ***machine precision***.

## 8. Further Comments

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

The real analogue of this routine is F08GFF (SOPGTR/DOPGTR).

## 9. Example

To compute all the eigenvalues and eigenvectors of the matrix $A$, where

$$A = \begin{pmatrix} -2.28 + 0.00i & 1.78 - 2.03i & 2.26 + 0.10i & -0.12 + 2.53i \\ 1.78 + 2.03i & -1.12 + 0.00i & 0.01 + 0.43i & -1.07 + 0.86i \\ 2.26 - 0.10i & 0.01 - 0.43i & -0.37 + 0.00i & 2.31 - 0.92i \\ -0.12 - 2.53i & -1.07 - 0.86i & 2.31 + 0.92i & -0.73 + 0.00i \end{pmatrix},$$

using packed storage. Here $A$ is Hermitian and must first be reduced to tridiagonal form by F08GSF (CHPTRD/ZHPTRD). The program then calls F08GTF (CUPGTR/ZUPGTR) to form $Q$, and passes this matrix to F08JSF (CSTEQR/ZSTEQR) which computes the eigenvalues and eigenvectors 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.

```
*     F08GTF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDZ
      PARAMETER        (NMAX=8,LDZ=NMAX)
```

```
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          AP(NMAX*(NMAX+1)/2), TAU(NMAX), WORK(NMAX-1),
     +                 Z(LDZ,NMAX)
      real             D(NMAX), E(NMAX), RWORK(2*NMAX-2)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         X04DBF, chptrd, csteqr, cupgtr
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08GTF 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
*
*        Reduce A to tridiagonal form T = (Q**H)*A*Q
*
         CALL chptrd(UPLO,N,AP,D,E,TAU,INFO)
*
*        Form Q explicitly, storing the result in Z
*
         CALL cupgtr(UPLO,N,AP,TAU,Z,LDZ,WORK,INFO)
*
*        Calculate all the eigenvalues and eigenvectors of A
*
         CALL csteqr('V',N,D,E,Z,LDZ,RWORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.GT.0) THEN
            WRITE (NOUT,*) 'Failure to converge.'
         ELSE
*
*           Print eigenvalues and eigenvectors
*
            WRITE (NOUT,*) 'Eigenvalues'
            WRITE (NOUT,99999) (D(I),I=1,N)
            WRITE (NOUT,*)
            IFAIL = 0
*
            CALL X04DBF('General',' ',N,N,Z,LDZ,'Bracketed','F7.4',
     +                  'Eigenvectors','Integer',RLABS,'Integer',CLABS,
     +                  80,0,IFAIL)
*
         END IF
      END IF
      STOP
*
99999 FORMAT (8X,4(F7.4,11X,:))
      END
```

## 9.2. Program Data

```
F08GTF Example Program Data
  4                                                              :Value of N
  'L'                                                            :Value of UPLO
 (-2.28, 0.00)
 ( 1.78, 2.03) (-1.12, 0.00)
 ( 2.26,-0.10) ( 0.01,-0.43) (-0.37, 0.00)
 (-0.12,-2.53) (-1.07,-0.86) ( 2.31, 0.92) (-0.73, 0.00)    :End of matrix A
```

## 9.3. Program Results

```
F08GTF Example Program Results

Eigenvalues
       -6.0002           -3.0030            0.5036            3.9996

Eigenvectors
                     1                   2                   3                   4
1  ( 0.7299, 0.0000) (-0.2120, 0.1497) ( 0.1000,-0.3570) ( 0.1991, 0.4720)
2  (-0.1663,-0.2061) ( 0.7307, 0.0000) ( 0.2863,-0.3353) (-0.2467, 0.3751)
3  (-0.4165,-0.1417) (-0.3291, 0.0479) ( 0.6890, 0.0000) ( 0.4468, 0.1466)
4  ( 0.1743, 0.4162) ( 0.5200, 0.1329) ( 0.0662, 0.4347) ( 0.5612, 0.0000)
```

# F08GUF (CUPMTR/ZUPMTR) – 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

F08GUF (CUPMTR/ZUPMTR) multiplies an arbitrary complex matrix $C$ by the complex unitary matrix $Q$ which was determined by F08GSF (CHPTRD/ZHPTRD) when reducing a complex Hermitian matrix to tridiagonal form.

## 2. Specification

```
SUBROUTINE F08GUF (SIDE, UPLO, TRANS, M, N, AP, TAU, C, LDC, WORK, INFO)
ENTRY      cupmtr (SIDE, UPLO, TRANS, M, N, AP, TAU, C, LDC, WORK, INFO)
INTEGER          M, N, LDC, INFO
complex          AP(*), TAU(*), C(LDC,*), WORK(*)
CHARACTER*1      SIDE, UPLO, 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 F08GSF (CHPTRD/ZHPTRD), which reduces a complex Hermitian matrix $A$ to real symmetric tridiagonal form $T$ by a unitary similarity transformation: $A = QTQ^H$. F08GSF represents the unitary matrix $Q$ as a product of elementary reflectors.

This routine may be used to form one of the matrix products

$$QC,\ Q^HC,\ CQ \text{ or } CQ^H,$$

overwriting the result on $C$ (which may be any complex rectangular matrix).

A common application of this routine is to transform a matrix $Z$ of eigenvectors of $T$ to the matrix $QZ$ of eigenvectors of $A$.

## 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: UPLO – CHARACTER*1. *Input*

*On entry*: this **must** be the same parameter UPLO as supplied to F08GSF (CHPTRD/ZHPTRD).

*Constraint*: UPLO = 'U' or 'L'.

3: 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'.

4: M – INTEGER. *Input*

*On entry*: $m$, the number of rows of the matrix $C$; $m$ is also the order of $Q$ if SIDE = 'L'.

*Constraint*: M ≥ 0.

5: N – INTEGER. *Input*

*On entry*: $n$, the number of columns of the matrix $C$; $n$ is also the order of $Q$ if SIDE = 'R'.

*Constraint*: N ≥ 0.

6: AP(*) – ***complex*** array. *Input*

**Note**: the dimension of the array AP must be at least max(1,M*(M+1)/2) if SIDE = 'L' and at least max(1,N*(N+1)/2) if SIDE = 'R'.

*On entry*: details of the vectors which define the elementary reflectors, as returned by F08GSF (CHPTRD/ZHPTRD).

7: TAU(*) – ***complex*** array. *Input*

**Note**: the dimension of the array TAU must be at least max(1,M–1) if SIDE = 'L' and at least max(1,N–1) if SIDE = 'R'.

*On entry*: further details of the elementary reflectors, as returned by F08GSF (CHPTRD/ZHPTRD).

8: 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.

9: LDC – INTEGER. *Input*

*On entry*: the first dimension of the array C as declared in the (sub)program from which F08GUF (CUPMTR/ZUPMTR) is called.

*Constraint*: LDC ≥ max(1,M).

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

**Note**: the dimension of the array WORK must be at least max(1,N) if SIDE = 'L' and at least max(1,M) if SIDE = 'R'.

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 $i$th 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 $\varepsilon$ is the ***machine precision***.

## 8. Further Comments

The total number of real floating-point operations is approximately $8m^2n$ if SIDE = 'L' and $8mn^2$ if SIDE = 'R'.

The real analogue of this routine is F08GGF (SOPMTR/DOPMTR).

## 9. Example

To compute the two smallest eigenvalues, and the associated eigenvectors, of the matrix $A$, where

$$A = \begin{pmatrix} -2.28 + 0.00i & 1.78 - 2.03i & 2.26 + 0.10i & -0.12 + 2.53i \\ 1.78 + 2.03i & -1.12 + 0.00i & 0.01 + 0.43i & -1.07 + 0.86i \\ 2.26 - 0.10i & 0.01 - 0.43i & -0.37 + 0.00i & 2.31 - 0.92i \\ -0.12 - 2.53i & -1.07 - 0.86i & 2.31 + 0.92i & -0.73 + 0.00i \end{pmatrix},$$

using packed storage. Here $A$ is Hermitian and must first be reduced to tridiagonal form $T$ by F08GSF (CHPTRD/ZHPTRD). The program then calls F08JJF (SSTEBZ/DSTEBZ) to compute the requested eigenvalues and F08JXF (CSTEIN/ZSTEIN) to compute the associated eigenvectors of $T$. Finally F08GUF (CUPMTR/ZUPMTR) is called to transform the eigenvectors to those 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.**

```
*     F08GUF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDZ
      PARAMETER        (NMAX=8,LDZ=NMAX)
      real             ZERO
      PARAMETER        (ZERO=0.0e0)
*     .. Local Scalars ..
      real             VL, VU
      INTEGER          I, IFAIL, INFO, J, M, N, NSPLIT
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          AP(NMAX*(NMAX+1)/2), TAU(NMAX), WORK(NMAX),
     +                 Z(LDZ,NMAX)
      real             D(NMAX), E(NMAX), RWORK(5*NMAX), W(NMAX)
      INTEGER          IBLOCK(NMAX), IFAILV(NMAX), ISPLIT(NMAX),
     +                 IWORK(NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         sstebz, X04DBF, chptrd, cstein, cupmtr
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08GUF 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
*
*           Reduce A to tridiagonal form T = (Q**H)*A*Q
*
            CALL chptrd(UPLO,N,AP,D,E,TAU,INFO)
*
*           Calculate the two smallest eigenvalues of T (same as A)
*
            CALL sstebz('I','B',N,VL,VU,1,2,ZERO,D,E,M,NSPLIT,W,IBLOCK,
     +                  ISPLIT,RWORK,IWORK,INFO)
*
            WRITE (NOUT,*)
            IF (INFO.GT.0) THEN
               WRITE (NOUT,*) 'Failure to converge.'
            ELSE
               WRITE (NOUT,*) 'Eigenvalues'
               WRITE (NOUT,99999) (W(I),I=1,M)
*
*              Calculate the eigenvectors of T, storing the result in Z
*
               CALL cstein(N,D,E,M,W,IBLOCK,ISPLIT,Z,LDZ,RWORK,IWORK,
     +                     IFAILV,INFO)
*
               IF (INFO.GT.0) THEN
                  WRITE (NOUT,*) 'Failure to converge.'
               ELSE
*
*                 Calculate the eigenvectors of A = Q * (eigenvectors of T)
*
                  CALL cupmtr('Left',UPLO,'No transpose',N,M,AP,TAU,Z,LDZ,
     +                        WORK,INFO)
*
*                 Print eigenvectors
*
                  WRITE (NOUT,*)
                  IFAIL = 0
*
                  CALL X04DBF('General',' ',N,M,Z,LDZ,'Bracketed','F7.4',
     +                        'Eigenvectors','Integer',RLABS,'Integer',
     +                        CLABS,80,0,IFAIL)
*
               END IF
            END IF
         END IF
         STOP
*
99999 FORMAT (8X,4(F7.4,11X,:))
      END
```

## 9.2. Program Data

```
F08GUF Example Program Data
  4                                                              :Value of N
  'L'                                                            :Value of UPLO
 (-2.28, 0.00)
 ( 1.78, 2.03) (-1.12, 0.00)
 ( 2.26,-0.10) ( 0.01,-0.43) (-0.37, 0.00)
 (-0.12,-2.53) (-1.07,-0.86) ( 2.31, 0.92) (-0.73, 0.00)    :End of matrix A
```

## 9.3. Program Results

```
F08GUF Example Program Results

Eigenvalues
       -6.0002           -3.0030

Eigenvectors
                        1                       2
1   ( 0.7299, 0.0000) (-0.2595, 0.0000)
2   (-0.1663,-0.2061) ( 0.5969, 0.4214)
3   (-0.4165,-0.1417) (-0.2965,-0.1507)
4   ( 0.1743, 0.4162) ( 0.3482, 0.4085)
```

# F08HEF (SSBTRD/DSBTRD) – 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

F08HEF (SSBTRD/DSBTRD) reduces a real symmetric band matrix to tridiagonal form.

## 2. Specification

```
SUBROUTINE F08HEF (VECT, UPLO, N, KD, AB, LDAB, D, E, Q, LDQ, WORK, INFO)
ENTRY      ssbtrd  (VECT, UPLO, N, KD, AB, LDAB, D, E, Q, LDQ, WORK, INFO)
INTEGER          N, KD, LDAB, LDQ, INFO
real             AB(LDAB,*), D(*), E(*), Q(LDQ,*), WORK(*)
CHARACTER*1      VECT, UPLO
```

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

## 3. Description

The symmetric band matrix $A$ is reduced to symmetric tridiagonal form $T$ by an orthogonal similarity transformation: $T = Q^T A Q$. The orthogonal matrix $Q$ is determined as a product of Givens rotation matrices, and may be formed explicitly by the routine if required.

The routine uses a vectorisable form of the reduction, due to Kaufman [1].

## 4. References

[1] KAUFMAN, L.
Banded Eigenvalue Solvers on Vector Machines.
ACM Trans. Math. Softw., 10, pp. 73-86, 1984.

[2] PARLETT, B.N.
The Symmetric Eigenvalue Problem, §7-5.
Prentice-Hall, Englewood Cliffs, New Jersey, 1980.

## 5. Parameters

1: VECT – CHARACTER*1. *Input*

*On entry*: indicates whether $Q$ is to be returned as follows:

if VECT = 'V', then $Q$ is returned;

if VECT = 'N', then $Q$ is not required.

*Constraint*: VECT = 'V' or 'N'.

2: UPLO – CHARACTER*1. *Input*

*On entry*: indicates whether the upper or lower triangular part of $A$ is stored as follows:

if UPLO = 'U', then the upper triangular part of $A$ is stored;

if UPLO = 'L', then the lower triangular part of $A$ is stored.

*Constraint*: UPLO = 'U' or 'L'.

3: N – INTEGER. *Input*

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

*Constraint*: N $\geq$ 0.

4: 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.

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 $n$ 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*: $A$ is overwritten.

6: LDAB – INTEGER. *Input*

*On entry*: the first dimension of the array AB as declared in the (sub)program from which F08HEF (SSBTRD/DSBTRD) is called.

*Constraint*: LDAB ≥ max(1,KD+1).

7: D(*) – ***real*** array. *Output*

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

*On exit*: the diagonal elements of the tridiagonal matrix $T$.

8: E(*) – ***real*** array. *Output*

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

*On exit*: the off-diagonal elements of the tridiagonal matrix $T$.

9: Q(LDQ,*) – ***real*** array. *Output*

**Note**: the second dimension of the array Q must be at least max(1,N) if VECT = 'V', and at least 1 if VECT = 'N'.

*On exit*: the $n$ by $n$ orthogonal matrix $Q$ if VECT = 'V'.

Q is not referenced if VECT = 'N'.

10: LDQ – INTEGER. *Input*

*On entry*: the first dimension of the array Q as declared in the (sub)program from which F08HEF (SSBTRD/DSBTRD) is called.

*Constraints*: LDQ ≥ max(1,N) if VECT = 'V',
LDQ ≥ 1 if VECT = 'N'.

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

**Note**: the dimension of the array WORK 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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

## 7. Accuracy

The computed tridiagonal matrix $T$ is exactly similar to a nearby matrix $A + E$, where

$$\|E\|_2 \leq c(n)\varepsilon\|A\|_2,$$

$c(n)$ is a modestly increasing function of $n$, and $\varepsilon$ is the ***machine precision***.

The elements of $T$ themselves may be sensitive to small perturbations in $A$ or to rounding errors in the computation, but this does not affect the stability of the eigenvalues and eigenvectors.

The computed matrix $Q$ differs from an exactly orthogonal matrix by a matrix $E$ such that

$$\|E\|_2 = O(\varepsilon),$$

where $\varepsilon$ is the ***machine precision***.

## 8. Further Comments

The total number of floating-point operations is approximately $6n^2k$ if VECT = 'N' with $3n^3(k-1)/k$ additional operations if VECT = 'V'.

The complex analogue of this routine is F08HSF (CHBTRD/ZHBTRD).

## 9. Example

To compute all the eigenvalues and eigenvectors of the matrix $A$, where

$$A = \begin{pmatrix} 4.99 & 0.04 & 0.22 & 0.00 \\ 0.04 & 1.05 & -0.79 & 1.04 \\ 0.22 & -0.79 & -2.31 & -1.30 \\ 0.00 & 1.04 & -1.30 & -0.43 \end{pmatrix}.$$

Here $A$ is symmetric and is treated as a band matrix. The program first calls F08HEF (SSBTRD/DSBTRD) to reduce $A$ to tridiagonal form $T$, and to form the orthogonal matrix $Q$; the results are then passed to F08JEF (SSTEQR/DSTEQR) which computes the eigenvalues and eigenvectors 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.**

```
*     F08HEF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, KMAX, LDAB, LDQ
      PARAMETER        (NMAX=8,KMAX=8,LDAB=KMAX+1,LDQ=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, KD, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             AB(LDAB,NMAX), D(NMAX), E(NMAX-1), Q(LDQ, NMAX),
     +                 WORK(2*NMAX-2)
*     .. External Subroutines ..
      EXTERNAL         ssbtrd, ssteqr, X04CAF
*     .. Intrinsic Functions ..
      INTRINSIC        MAX, MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08HEF 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
*
*           Reduce A to tridiagonal form T = (Q**T)*A*Q (and form Q)
*
            CALL ssbtrd('V',UPLO,N,KD,AB,LDAB,D,E,Q,LDQ,WORK,INFO)
*
*           Calculate all the eigenvalues and eigenvectors of A
*
            CALL ssteqr('V',N,D,E,Q,LDQ,WORK,INFO)
*
            WRITE (NOUT,*)
            IF (INFO.GT.0) THEN
               WRITE (NOUT,*) 'Failure to converge.'
            ELSE
*
*              Print eigenvalues and eigenvectors
*
               WRITE (NOUT,*) 'Eigenvalues'
               WRITE (NOUT,99999) (D(I),I=1,N)
               WRITE (NOUT,*)
               IFAIL = 0
*
               CALL X04CAF('General',' ',N,N,Q,LDQ,'Eigenvectors',IFAIL)
*
            END IF
         END IF
         STOP
*
99999 FORMAT (3X,(8F8.4))
         END
```

## 9.2. Program Data

```
F08HEF Example Program Data
  4  2                          :Values of N and KD
  'L'                           :Value of UPLO
  4.99
  0.04   1.05
  0.22  -0.79  -2.31
         1.04  -1.30  -0.43     :End of matrix A
```

## 9.3. Program Results

```
 F08HEF Example Program Results

 Eigenvalues
    -2.9943 -0.7000  1.9974  4.9969

 Eigenvectors
            1       2       3       4
 1  -0.0251  0.0162  0.0113  0.9995
 2   0.0656 -0.5859  0.8077  0.0020
 3   0.9002 -0.3135 -0.3006  0.0311
 4   0.4298  0.7471  0.5070 -0.0071
```

# F08HSF (CHBTRD/ZHBTRD) – 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

F08HSF (CHBTRD/ZHBTRD) reduces a complex Hermitian band matrix to tridiagonal form.

## 2. Specification

```
SUBROUTINE F08HSF (VECT, UPLO, N, KD, AB, LDAB, D, E, Q, LDQ, WORK, INFO)
ENTRY      chbtrd  (VECT, UPLO, N, KD, AB, LDAB, D, E, Q, LDQ, WORK, INFO)
INTEGER          N, KD, LDAB, LDQ, INFO
real             D(*), E(*)
complex          AB(LDAB,*), Q(LDQ,*), WORK(*)
CHARACTER*1      VECT, UPLO
```

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

## 3. Description

The Hermitian band matrix $A$ is reduced to real symmetric tridiagonal form $T$ by a unitary similarity transformation: $T = Q^H A Q$. The unitary matrix $Q$ is determined as a product of Givens rotation matrices, and may be formed explicitly by the routine if required.

The routine uses a vectorisable form of the reduction, due to Kaufman [1].

## 4. References

[1] KAUFMAN, L.
Banded Eigenvalue Solvers on Vector Machines.
ACM Trans. Math. Softw., 10, pp. 73-86, 1984.

[2] PARLETT, B.N.
The Symmetric Eigenvalue Problem, §7-5.
Prentice-Hall, Englewood Cliffs, New Jersey, 1980.

## 5. Parameters

1: VECT – CHARACTER*1. *Input*

*On entry*: indicates whether $Q$ is to be returned as follows:

if VECT = 'V', then $Q$ is returned;

if VECT = 'N', then $Q$ is not required.

*Constraint*: VECT = 'V' or 'N'.

2: UPLO – CHARACTER*1. *Input*

*On entry*: indicates whether the upper or lower triangular part of $A$ is stored as follows:

if UPLO = 'U', then the upper triangular part of $A$ is stored;

if UPLO = 'L', then the lower triangular part of $A$ is stored.

*Constraint*: UPLO = 'U' or 'L'.

3: N – INTEGER. *Input*

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

*Constraint*: N $\geq$ 0.

4: 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.

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 $n$ by $n$ Hermitian 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*: $A$ is overwritten.

6: LDAB – INTEGER. *Input*

*On entry*: the first dimension of the array AB as declared in the (sub)program from which F08HSF (CHBTRD/ZHBTRD) is called.

*Constraint*: LDAB ≥ max(1,KD+1).

7: D(*) – ***real*** array. *Output*

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

*On exit*: the diagonal elements of the tridiagonal matrix $T$.

8: E(*) – ***real*** array. *Output*

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

*On exit*: the off-diagonal elements of the tridiagonal matrix $T$.

9: Q(LDQ,*) – ***complex*** array. *Output*

**Note**: the second dimension of the array Q must be at least max(1,N) if VECT = 'V', and at least 1 if VECT = 'N'.

*On exit*: the $n$ by $n$ unitary matrix $Q$ if VECT = 'V'.

Q is not referenced if VECT = 'N'.

10: LDQ – INTEGER. *Input*

*On entry*: the first dimension of the array Q as declared in the (sub)program from which F08HSF (CHBTRD/ZHBTRD) is called.

*Constraints*: LDQ ≥ max(1,N) if VECT = 'V',
LDQ ≥ 1 if VECT = 'N'.

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

**Note**: the dimension of the array WORK 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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

## 7. Accuracy

The computed tridiagonal matrix $T$ is exactly similar to a nearby matrix $A + E$, where

$$\|E\|_2 \leq c(n)\varepsilon\|A\|_2,$$

$c(n)$ is a modestly increasing function of $n$, and $\varepsilon$ is the ***machine precision***.

The elements of $T$ themselves may be sensitive to small perturbations in $A$ or to rounding errors in the computation, but this does not affect the stability of the eigenvalues and eigenvectors.

The computed matrix $Q$ differs from an exactly unitary matrix by a matrix $E$ such that

$$\|E\|_2 = O(\varepsilon),$$

where $\varepsilon$ is the ***machine precision***.

## 8. Further Comments

The total number of real floating-point operations is approximately $20n^2k$ if VECT = 'N' with $10n^3(k-1)/k$ additional operations if VECT = 'V'.

The real analogue of this routine is F08HEF (SSBTRD/DSBTRD).

## 9. Example

To compute all the eigenvalues and eigenvectors of the matrix $A$, where

$$A = \begin{pmatrix} -3.13 + 0.00i & 1.94 - 2.10i & -3.40 + 0.25i & 0.00 + 0.00i \\ 1.94 + 2.10i & -1.91 + 0.00i & -0.82 - 0.89i & -0.67 + 0.34i \\ -3.40 - 0.25i & -0.82 + 0.89i & -2.87 + 0.00i & -2.10 - 0.16i \\ 0.00 + 0.00i & -0.67 - 0.34i & -2.10 + 0.16i & 0.50 + 0.00i \end{pmatrix}.$$

Here $A$ is Hermitian and is treated as a band matrix. The program first calls F08HSF (CHBTRD/ZHBTRD) to reduce $A$ to tridiagonal form $T$, and to form the unitary matrix $Q$; the results are then passed to F08JSF (CSTEQR/ZSTEQR) which computes the eigenvalues and eigenvectors 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.**

```
*     F08HSF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, KMAX, LDAB, LDQ
      PARAMETER        (NMAX=8,KMAX=8,LDAB=KMAX+1,LDQ=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, KD, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          AB(LDAB,NMAX), Q(LDQ,NMAX), WORK(NMAX)
      real             D(NMAX), E(NMAX-1), RWORK(2*NMAX-2)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         X04DBF, chbtrd, csteqr
*     .. Intrinsic Functions ..
      INTRINSIC        MAX, MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08HSF 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
*
*           Reduce A to tridiagonal form T = (Q**H)*A*Q (and form Q)
*
            CALL chbtrd('V',UPLO,N,KD,AB,LDAB,D,E,Q,LDQ,WORK,INFO)
*
*           Calculate all the eigenvalues and eigenvectors of A
*
            CALL csteqr('V',N,D,E,Q,LDQ,RWORK,INFO)
*
            WRITE (NOUT,*)
            IF (INFO.GT.0) THEN
               WRITE (NOUT,*) 'Failure to converge.'
            ELSE
*
*              Print eigenvalues and eigenvectors
*
               WRITE (NOUT,*) 'Eigenvalues'
               WRITE (NOUT,99999) (D(I),I=1,N)
               WRITE (NOUT,*)
               IFAIL = 0
*
               CALL X04DBF('General',' ',N,N,Q,LDQ,'Bracketed','F7.4',
     +                     'Eigenvectors','Integer',RLABS,'Integer',CLABS,
     +                     80,0,IFAIL)
*
            END IF
         END IF
         STOP
*
99999 FORMAT (8X,4(F7.4,11X,:))
      END
```

## 9.2. Program Data

```
F08HSF Example Program Data
  4  2                                                          :Values of N
 and KD
  'L'                                                          :Value of UPLO
 (-3.13, 0.00) ( 1.94,-2.10) (-3.40, 0.25)
 ( 1.94, 2.10) (-1.91, 0.00) (-0.82,-0.89) (-0.67, 0.34)
 (-3.40,-0.25) (-0.82, 0.89) (-2.87, 0.00) (-2.10,-0.16)
               (-0.67,-0.34) (-2.10, 0.16) ( 0.50, 0.00)       :End of matrix A
```

## 9.3. Program Results

```
 F08HSF Example Program Results

 Eigenvalues
        -7.0042           -4.0038            0.5968            3.0012

 Eigenvectors
                        1                   2                   3                   4
 1   ( 0.7293, 0.0000) (-0.2128, 0.1511) (-0.3354,-0.1604) (-0.5114,-0.0163)
 2   (-0.1654,-0.2046) ( 0.7316, 0.0000) (-0.2804,-0.3413) (-0.2374,-0.3796)
 3   ( 0.6081, 0.0301) ( 0.3910,-0.3843) (-0.0144, 0.1532) ( 0.5523, 0.0000)
 4   ( 0.1653,-0.0303) ( 0.2775,-0.1378) ( 0.8019, 0.0000) (-0.4517, 0.1693)
```

# F08JEF (SSTEQR/DSTEQR) – 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

F08JEF (SSTEQR/DSTEQR) computes all the eigenvalues, and optionally all the eigenvectors, of a real symmetric tridiagonal matrix, or of a real symmetric matrix which has been reduced to tridiagonal form.

## 2. Specification

```
SUBROUTINE F08JEF (COMPZ, N, D, E, Z, LDZ, WORK, INFO)
ENTRY      ssteqr  (COMPZ, N, D, E, Z, LDZ, WORK, INFO)
INTEGER            N, LDZ, INFO
real               D(*), E(*), Z(LDZ,*), WORK(*)
CHARACTER*1        COMPZ
```

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

## 3. Description

This routine computes all the eigenvalues, and optionally all the eigenvectors, of a real symmetric tridiagonal matrix $T$. In other words, it can compute the spectral factorization of $T$ as

$$T = Z\Lambda Z^T,$$

where $\Lambda$ is a diagonal matrix whose diagonal elements are the eigenvalues $\lambda_i$, and $Z$ is the orthogonal matrix whose columns are the eigenvectors $z_i$. Thus

$$Tz_i = \lambda_i z_i \text{ for } i = 1,2,...,n.$$

The routine may also be used to compute all the eigenvalues and eigenvectors of a real symmetric matrix $A$ which has been reduced to tridiagonal form $T$:

$$A = QTQ^T, \text{ where } Q \text{ is orthogonal,}$$
$$= (QZ)\Lambda(QZ)^T.$$

In this case, the matrix $Q$ must be formed explicitly and passed to F08JEF, which must be called with COMPZ = 'V'. The routines which must be called to perform the reduction to tridiagonal form and form $Q$ are:

| | |
|---|---|
| full matrix | F08FEF (SSYTRD/DSYTRD) + F08FFF (SORGTR/DORGTR) |
| full matrix, packed storage | F08GEF (SSPTRD/DSPTRD) + F08GFF (SOPGTR/DOPGTR) |
| band matrix | F08HEF (SSBTRD/DSBTRD) with VECT = 'V'. |

F08JEF uses the implicitly shifted $QR$ algorithm, switching between the $QR$ and $QL$ variants in order to handle graded matrices effectively (see Greenbaum and Dongarra [2]). The eigenvectors are normalized so that $\|z_i\|_2 = 1$, but are determined only to within a factor $\pm 1$.

If only the eigenvalues of $T$ are required, it is more efficient to call F08JFF (SSTERF/DSTERF) instead. If $T$ is positive-definite, small eigenvalues can be computed more accurately by F08JGF (SPTEQR/DPTEQR).

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §8.2.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

[2] GREENBAUM, A. and DONGARRA, J.J.
Experiments with QR/QL Methods for the Symmetric Tridiagonal Eigenproblem.
LAPACK Working Note No. 17 (Technical Report CS-89-92), University of Tennessee, Knoxville, 1989.

[3] PARLETT, B.N.
The Symmetric Eigenvalue Problem, §8.
Prentice-Hall, Englewood Cliffs, New Jersey, 1980.

## 5. Parameters

1: COMPZ – CHARACTER*1. *Input*

*On entry*: indicates whether the eigenvectors are to be computed as follows:

if COMPZ = 'N', then only the eigenvalues are computed (and the array Z is not referenced);

if COMPZ = 'I', then the eigenvalues and eigenvectors of $T$ are computed (and the array Z is initialized by the routine);

if COMPZ = 'V', then the eigenvalues and eigenvectors of $A$ are computed (and the array Z must contain the matrix $Q$ on entry).

*Constraint*: COMPZ = 'N', 'I' or 'V'.

2: N – INTEGER. *Input*

*On entry*: $n$, the order of the matrix $T$.

*Constraint*: N ≥ 0.

3: D(*) – ***real*** array. *Input/Output*

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

*On entry*: the diagonal elements of the tridiagonal matrix $T$.

*On exit*: the $n$ eigenvalues in ascending order, unless INFO > 0 (in which case see Section 6).

4: E(*) – ***real*** array. *Input/Output*

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

*On entry*: the off-diagonal elements of the tridiagonal matrix $T$.

*On exit*: the array is overwritten.

5: Z(LDZ,*) – ***real*** array. *Input/Output*

**Note**: the second dimension of the array Z must be at least max(1,N) if COMPZ = 'V' or 'I', and at least 1 if COMPZ = 'N'.

*On entry*: if COMPZ = 'V', Z must contain the orthogonal matrix $Q$ from the reduction to tridiagonal form. If COMPZ = 'I', Z need not be set.

*On exit*: if COMPZ = 'I' or 'V', the $n$ required orthonormal eigenvectors stored by columns; the $i$th column corresponds to the $i$th eigenvalue, where $i = 1,2,...,n$, unless INFO > 0.

Z is not referenced if COMPZ = 'N'.

6: LDZ – INTEGER. *Input*

*On entry*: the first dimension of the array Z as declared in the (sub)program from which F08JEF (SSTEQR/DSTEQR) is called.

*Constraints*: LDZ ≥ 1 if COMPZ = 'N',
LDZ ≥ max(1,N) if COMPZ = 'V' or 'I'.

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

**Note**: the dimension of the array WORK must be at least max(1,2*(N–1)) if COMPZ = 'V' or 'I', and at least 1 if COMPZ = 'N'.

WORK is not referenced if COMPZ = '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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

The algorithm has failed to find all the eigenvalues after a total of 30×N iterations. In this case, D and E contain on exit the diagonal and off-diagonal elements, respectively, of a tridiagonal matrix orthogonally similar to $T$. If INFO = $i$, then $i$ off-diagonal elements have not converged to zero.

## 7. Accuracy

The computed eigenvalues and eigenvectors are exact for a nearby matrix $T + E$, where

$$\|E\|_2 = O(\varepsilon)\|T\|_2,$$

and $\varepsilon$ is the ***machine precision***.

If $\lambda_i$ is an exact eigenvalue and $\tilde{\lambda}_i$ is the corresponding computed value, then

$$|\tilde{\lambda}_i - \lambda_i| \leq c(n)\varepsilon\|T\|_2,$$

where $c(n)$ is a modestly increasing function of $n$.

If $z_i$ is the corresponding exact eigenvector, and $\tilde{z}_i$ is the corresponding computed eigenvector, then the angle $\theta(\tilde{z}_i, z_i)$ between them is bounded as follows:

$$\theta(\tilde{z}_i, z_i) \leq \frac{c(n)\varepsilon\|T\|_2}{\min_{i \neq j}|\lambda_i - \lambda_j|}.$$

Thus the accuracy of a computed eigenvector depends on the gap between its eigenvalue and all the other eigenvalues.

## 8. Further Comments

The total number of floating-point operations is typically about $24n^2$ if COMPZ = 'N' and about $7n^3$ if COMPZ = 'V' or 'I', but depends on how rapidly the algorithm converges. When COMPZ = 'N', the operations are all performed in scalar mode; the additional operations to compute the eigenvectors when COMPZ = 'V' or 'I' can be vectorized and on some machines may be performed much faster.

The complex analogue of this routine is F08JSF (CSTEQR/ZSTEQR).

## 9. Example

To compute all the eigenvalues and eigenvectors of the symmetric tridiagonal matrix $T$, where

$$T = \begin{pmatrix} -6.99 & -0.44 & 0.00 & 0.00 \\ -0.44 & 7.92 & -2.63 & 0.00 \\ 0.00 & -2.63 & 2.34 & -1.18 \\ 0.00 & 0.00 & -1.18 & 0.32 \end{pmatrix}.$$

See also the examples for F08FFF, F08GFF or F08HEF, which illustrate the use of this routine to compute the eigenvalues and eigenvectors of a full or band symmetric matrix.

## 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.**

```
*     F08JEF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDZ
      PARAMETER        (NMAX=8,LDZ=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, N
*     .. Local Arrays ..
      real             D(NMAX), E(NMAX-1), WORK(2*NMAX-2), Z(LDZ, NMAX)
*     .. External Subroutines ..
      EXTERNAL         ssteqr, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08JEF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read T from data file
*
         READ (NIN,*) (D(I),I=1,N)
         READ (NIN,*) (E(I),I=1,N-1)
*
*        Calculate all the eigenvalues and eigenvectors of T
*
         CALL ssteqr('I',N,D,E,Z,LDZ,WORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.GT.0) THEN
            WRITE (NOUT,*) 'Failure to converge.'
         ELSE
*
*           Print eigenvalues and eigenvectors
*
            WRITE (NOUT,*) 'Eigenvalues'
            WRITE (NOUT,99999) (D(I),I=1,N)
            WRITE (NOUT,*)
            IFAIL = 0
*
            CALL X04CAF('General',' ',N,N,Z,LDZ,'Eigenvectors',IFAIL)
*
         END IF
      END IF
      STOP
*
99999 FORMAT (3X,(8F8.4))
      END
```

## 9.2. Program Data

```
F08JEF Example Program Data
  4                                  :Value of N
 -6.99   7.92   2.34   0.32
 -0.44  -2.63  -1.18                 :End of matrix T
```

## 9.3. Program Results

```
F08JEF Example Program Results

Eigenvalues
   -7.0037 -0.4059  2.0028  8.9968

Eigenvectors
          1       2       3       4
1    0.9995 -0.0109 -0.0167 -0.0255
2    0.0310  0.1627  0.3408  0.9254
3    0.0089  0.5170  0.7696 -0.3746
4    0.0014  0.8403 -0.5397  0.0509
```

# F08JFF (SSTERF/DSTERF) – 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

F08JFF (SSTERF/DSTERF) computes all the eigenvalues of a real symmetric tridiagonal matrix.

## 2. Specification

```
SUBROUTINE F08JFF (N, D, E, INFO)
ENTRY      ssterf  (N, D, E, INFO)
INTEGER            N, INFO
real               D(*), E(*)
```

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

## 3. Description

This routine computes all the eigenvalues of a real symmetric tridiagonal matrix, using a square-root-free variant of the *QR* algorithm.

The routine uses an explicit shift, and, like F08JEF (SSTEQR/DSTEQR), switches between the *QR* and *QL* variants in order to handle graded matrices effectively (see Greenbaum and Dongarra [1]).

## 4. References

[1] GREENBAUM, A. and DONGARRA, J.J.
Experiments with QR/QL Methods for the Symmetric Tridiagonal Eigenproblem.
LAPACK Working Note No. 17 (Technical Report CS-89-92), University of Tennessee, Knoxville, 1989.

[2] PARLETT, B.N.
The Symmetric Eigenvalue Problem, §8-15.
Prentice-Hall, Englewood Cliffs, New Jersey, 1980.

## 5. Parameters

1: N – INTEGER. *Input*

*On entry*: $n$, the order of the matrix $T$.

*Constraint*: N $\geq$ 0.

2: D(*) – ***real*** array. *Input/Output*

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

*On entry*: the diagonal elements of the tridiagonal matrix $T$.

*On exit*: the $n$ eigenvalues in ascending order, unless INFO > 0 (in which case see Section 6).

3: E(*) – ***real*** array. *Input/Output*

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

*On entry*: the off-diagonal elements of the tridiagonal matrix $T$.

*On exit*: the array is overwritten.

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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

The algorithm has failed to find all the eigenvalues after a total of 30×N iterations. If INFO = $i$, then on exit $i$ elements of E have not converged to zero.

## 7. Accuracy

The computed eigenvalues are exact for a nearby matrix $T + E$, where

$$\|E\|_2 = O(\varepsilon)\|T\|_2,$$

and $\varepsilon$ is the ***machine precision***.

If $\lambda_i$ is an exact eigenvalue and $\tilde{\lambda}_i$ is the corresponding computed value, then

$$|\tilde{\lambda}_i - \lambda_i| \leq c(n)\varepsilon\|T\|_2,$$

where $c(n)$ is a modestly increasing function of $n$.

## 8. Further Comments

The total number of floating-point operations is typically about $14n^2$, but depends on how rapidly the algorithm converges. The operations are all performed in scalar mode.

There is no complex analogue of this routine.

## 9. Example

To compute all the eigenvalues of the symmetric tridiagonal matrix $T$, where

$$T = \begin{pmatrix} -6.99 & -0.44 & 0.00 & 0.00 \\ -0.44 & 7.92 & -2.63 & 0.00 \\ 0.00 & -2.63 & 2.34 & -1.18 \\ 0.00 & 0.00 & -1.18 & 0.32 \end{pmatrix}.$$

### 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.**

```
*     F08JFF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=8)
*     .. Local Scalars ..
      INTEGER          I, INFO, N
*     .. Local Arrays ..
      real             D(NMAX), E(NMAX-1)
*     .. External Subroutines ..
      EXTERNAL         ssterf
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08JFF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read T from data file
*
```

```
         READ (NIN,*) (D(I),I=1,N)
         READ (NIN,*) (E(I),I=1,N-1)
*
*        Calculate the eigenvalues of T
*
         CALL ssterf(N,D,E,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.GT.0) THEN
            WRITE (NOUT,*) 'Failure to converge.'
         ELSE
            WRITE (NOUT,*) 'Eigenvalues'
            WRITE (NOUT,99999) (D(I),I=1,N)
         END IF
      END IF
      STOP
*
99999 FORMAT (3X,(9F8.4))
      END
```

## 9.2. Program Data

```
F08JFF Example Program Data
  4                                :Value of N
 -6.99    7.92    2.34    0.32
 -0.44   -2.63   -1.18             :End of matrix T
```

## 9.3. Program Results

```
 F08JFF Example Program Results

 Eigenvalues
    -7.0037 -0.4059  2.0028  8.9968
```

# F08JGF (SPTEQR/DPTEQR) – 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

F08JGF (SPTEQR/DPTEQR) computes all the eigenvalues, and optionally all the eigenvectors, of a real symmetric positive-definite tridiagonal matrix, or of a real symmetric positive-definite matrix which has been reduced to tridiagonal form.

## 2. Specification

```
SUBROUTINE F08JGF (COMPZ, N, D, E, Z, LDZ, WORK, INFO)
ENTRY      spteqr  (COMPZ, N, D, E, Z, LDZ, WORK, INFO)
INTEGER          N, LDZ, INFO
real             D(*), E(*), Z(LDZ,*), WORK(*)
CHARACTER*1      COMPZ
```

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

## 3. Description

This routine computes all the eigenvalues, and optionally all the eigenvectors, of a real symmetric positive-definite tridiagonal matrix $T$. In other words, it can compute the spectral factorization of $T$ as

$$T = Z\Lambda Z^T,$$

where $\Lambda$ is a diagonal matrix whose diagonal elements are the eigenvalues $\lambda_i$, and $Z$ is the orthogonal matrix whose columns are the eigenvectors $z_i$. Thus

$$Tz_i = \lambda_i z_i \text{ for } i = 1,2,...,n.$$

The routine may also be used to compute all the eigenvalues and eigenvectors of a real symmetric positive-definite matrix $A$ which has been reduced to tridiagonal form $T$:

$$A = QTQ^T, \text{ where } Q \text{ is orthogonal}$$
$$= (QZ)\Lambda(QZ)^T.$$

In this case, the matrix $Q$ must be formed explicitly and passed to F08JGF, which is called with COMPZ = 'V'. The routines which must be called to perform the reduction to tridiagonal form and form $Q$ are:

| | |
|---|---|
| full matrix | F08FEF (SSYTRD/DSYTRD) + F08FFF (SORGTR/DORGTR) |
| full matrix, packed storage | F08GEF (SSPTRD/DSPTRD) + F08GFF (SOPGTR/DOPGTR) |
| band matrix | F08HEF (SSBTRD/DSBTRD) with VECT = 'V'. |

The routine first factorizes $T$ as $LDL^T$ where $L$ is unit lower bidiagonal and $D$ is diagonal. It forms the bidiagonal matrix $B = LD^{\frac{1}{2}}$, and then calls F08MEF (SBDSQR/DBDSQR) to compute the singular values of $B$ which are the same as the eigenvalues of $T$. The method used by the routine allows high relative accuracy to be achieved in the small eigenvalues of $T$. The eigenvectors are normalized so that $\|z_i\|_2 = 1$, but are determined only to within a factor $\pm 1$.

## 4. References

[1] BARLOW, J. and DEMMEL, J.
Computing Accurate Eigensystems of Scaled Diagonally Dominant Matrices.
SIAM J. Num. Anal., 27, pp. 762-791, 1990.

## 5. Parameters

1: COMPZ – CHARACTER*1. *Input*

*On entry*: indicates whether the eigenvectors are to be computed as follows:

if COMPZ = 'N', then only the eigenvalues are computed (and the array Z is not referenced);

if COMPZ = 'I', then the eigenvalues and eigenvectors of $T$ are computed (and the array Z is initialized by the routine);

if COMPZ = 'V', then the eigenvalues and eigenvectors of $A$ are computed (and the array Z must contain the matrix $Q$ on entry).

*Constraint*: COMPZ = 'N', 'I' or 'V'.

2: N – INTEGER. *Input*

*On entry*: $n$, the order of the matrix $T$.

*Constraint*: N ≥ 0.

3: D(*) – ***real*** array. *Input/Output*

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

*On entry*: the diagonal elements of the tridiagonal matrix $T$.

*On exit*: the $n$ eigenvalues in descending order, unless INFO > 0, in which case the array is overwritten.

4: E(*) – ***real*** array. *Input/Output*

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

*On entry*: the off-diagonal elements of the tridiagonal matrix $T$.

*On exit*: the array is overwritten.

5: Z(LDZ,*) – ***real*** array. *Input/Output*

**Note**: the second dimension of the array Z must be at least max(1,N) if COMPZ = 'V' or 'I', and at least 1 if COMPZ = 'N'.

*On entry*: if COMPZ = 'V', Z must contain the orthogonal matrix $Q$ from the reduction to tridiagonal form. If COMPZ = 'I', Z need not be set.

*On exit*: if COMPZ = 'I' or 'V', the $n$ required orthonormal eigenvectors stored by columns; the $i$th column corresponds to the $i$th eigenvalue, where $i = 1,2,...,n$, unless INFO > 0.

Z is not referenced if COMPZ = 'N'.

6: LDZ – INTEGER. *Input*

*On entry*: the first dimension of the array Z as declared in the (sub)program from which F08JGF (SPTEQR/DPTEQR) is called.

*Constraints*: LDZ ≥ 1 if COMPZ = 'N',
LDZ ≥ max(1,N) if COMPZ = 'V' or 'I'.

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

**Note**: the dimension of the array WORK must be at least max(1,4*(N–1)) if COMPZ = 'V' or 'I', and at least 1 if COMPZ = 'N'.

WORK is not referenced if COMPZ = '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 $i$th 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 Cholesky factorization of $T$ could not be completed. Hence $T$ itself is not positive-definite.

If INFO = N + $i$, the algorithm to compute the singular values of the Cholesky factor $B$ failed to converge; $i$ off-diagonal elements did not converge to zero.

## 7. Accuracy

The eigenvalues and eigenvectors of $T$ are computed to high relative accuracy which means that if they vary widely in magnitude, then any small eigenvalues (and corresponding eigenvectors) will be computed more accurately than, for example, with the standard $QR$ method. However, the reduction to tridiagonal form (prior to calling the routine) may exclude the possibility of obtaining high relative accuracy in the small eigenvalues of the original matrix if its eigenvalues vary widely in magnitude.

To be more precise, let $H$ be the tridiagonal matrix defined by $H = DTD$, where $D$ is diagonal with $d_{ii} = t_{ii}^{-\frac{1}{2}}$, and $h_{ii} = 1$ for all $i$. If $\lambda_i$ is an exact eigenvalue of $T$ and $\tilde{\lambda}_i$ is the corresponding computed value, then

$$|\tilde{\lambda}_i - \lambda_i| \leq c(n)\varepsilon\kappa_2(H)\lambda_i$$

where $c(n)$ is a modestly increasing function of $n$, $\varepsilon$ is the ***machine precision***, and $\kappa_2(H)$ is the condition number of $H$ with respect to inversion defined by: $\kappa_2(H) = \|H\|.\|H^{-1}\|$.

If $z_i$ is the corresponding exact eigenvector of $T$, and $\tilde{z}_i$ is the corresponding computed eigenvector, then the angle $\theta(\tilde{z}_i, z_i)$ between them is bounded as follows:

$$\theta(\tilde{z}_i, z_i) \leq \frac{c(n)\varepsilon\kappa_2(H)}{relgap_i}$$

where $relgap_i$ is the relative gap between $\lambda_i$ and the other eigenvalues, defined by

$$relgap_i = \min_{i \neq j} \frac{|\lambda_i - \lambda_j|}{(\lambda_i + \lambda_j)}.$$

## 8. Further Comments

The total number of floating-point operations is typically about $30n^2$ if COMPZ = 'N' and about $6n^3$ if COMPZ = 'V' or 'I', but depends on how rapidly the algorithm converges. When COMPZ = 'N', the operations are all performed in scalar mode; the additional operations to compute the eigenvectors when COMPZ = 'V' or 'I' can be vectorized and on some machines may be performed much faster.

The complex analogue of this routine is F08JUF (CPTEQR/ZPTEQR).

## 9. Example

To compute all the eigenvalues and eigenvectors of the symmetric positive-definite tridiagonal matrix $T$, where

$$T = \begin{pmatrix} 4.16 & 3.17 & 0.00 & 0.00 \\ 3.17 & 5.25 & -0.97 & 0.00 \\ 0.00 & -0.97 & 1.09 & 0.55 \\ 0.00 & 0.00 & 0.55 & 0.62 \end{pmatrix}.$$

## 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.**

```
*     F08JGF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDZ
      PARAMETER        (NMAX=8,LDZ=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, N
*     .. Local Arrays ..
      real             D(NMAX), E(NMAX-1), WORK(4*NMAX-4), Z(LDZ, NMAX)
*     .. External Subroutines ..
      EXTERNAL         spteqr, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08JGF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read T from data file
*
         READ (NIN,*) (D(I),I=1,N)
         READ (NIN,*) (E(I),I=1,N-1)
*
*        Calculate all the eigenvalues and eigenvectors of T
*
         CALL spteqr('I',N,D,E,Z,LDZ,WORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.GT.0 .AND. INFO.LE.N) THEN
            WRITE (NOUT,*) 'T is not positive-definite.'
         ELSE IF (INFO.GT.N) THEN
            WRITE (NOUT,*) 'Failure to converge.'
         ELSE
*
*           Print eigenvalues and eigenvectors
*
            WRITE (NOUT,*) 'Eigenvalues'
            WRITE (NOUT,99999) (D(I),I=1,N)
            WRITE (NOUT,*)
            IFAIL = 0
*
            CALL X04CAF('General',' ',N,N,Z,LDZ,'Eigenvectors',IFAIL)
*
         END IF
      END IF
      STOP
*
99999 FORMAT (3X,(8F8.4))
      END
```

## 9.2. Program Data

```
F08JGF Example Program Data
  4                                :Value of N
  4.16   5.25   1.09   0.62
  3.17  -0.97   0.55             :End of matrix T
```

## 9.3. Program Results

```
F08JGF Example Program Results

Eigenvalues
    8.0023  1.9926  1.0014  0.1237

Eigenvectors
          1       2       3       4
1    0.6326  0.6245 -0.4191  0.1847
2    0.7668 -0.4270  0.4176 -0.2352
3   -0.1082  0.6071  0.4594 -0.6393
4   -0.0081  0.2432  0.6625  0.7084
```

# F08JJF (SSTEBZ/DSTEBZ) – 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

F08JJF (SSTEBZ/DSTEBZ) computes some (or all) of the eigenvalues of a real symmetric tridiagonal matrix, by bisection.

## 2. Specification

```
 SUBROUTINE F08JJF (RANGE, ORDER, N, VL, VU, IL, IU, ABSTOL, D, E, M,
1                   NSPLIT, W, IBLOCK, ISPLIT, WORK, IWORK, INFO)
 ENTRY       sstebz  (RANGE, ORDER, N, VL, VU, IL, IU, ABSTOL, D, E, M,
1                   NSPLIT, W, IBLOCK, ISPLIT, WORK, IWORK, INFO)
 INTEGER          N, IL, IU, M, NSPLIT, IBLOCK(*), ISPLIT(*), IWORK(*),
1                 INFO
 real             VL, VU, ABSTOL, D(*), E(*), W(*), WORK(*)
 CHARACTER*1      RANGE, ORDER
```

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

## 3. Description

This routine uses bisection to compute some or all of the eigenvalues of a real symmetric tridiagonal matrix $T$.

It searches for zero or negligible off-diagonal elements of $T$ to see if the matrix splits into block diagonal form:

$$T = \begin{pmatrix} T_1 & & & \\ & T_2 & & \\ & & \ddots & \\ & & & T_p \end{pmatrix}.$$

It performs bisection on each of the blocks $T_i$ and returns the block index of each computed eigenvalue, so that a subsequent call to F08JKF (SSTEIN/DSTEIN) to compute eigenvectors can also take advantage of the block structure.

## 4. References

[1] KAHAN, W.
Accurate Eigenvalues of a Symmetric Tridiagonal Matrix.
Report CS41, Computer Science Department, Stanford University, 1966.

## 5. Parameters

1: RANGE – CHARACTER*1. *Input*

*On entry*: indicates which eigenvalues are required as follows:

if RANGE = 'A', then all the eigenvalues are required;

if RANGE = 'V', then all the eigenvalues in the half-open interval (VL,VU] are required;

if RANGE = 'I', then eigenvalues with indices IL to IU are required.

*Constraint*: RANGE = 'A', 'V' or 'I'.

2: ORDER – CHARACTER*1. *Input*

*On entry*: indicates the order in which the eigenvalues and their block numbers are to be stored as follows:

if ORDER = 'B', then the eigenvalues are to be grouped by split-off block and ordered from smallest to largest within each block;

if ORDER = 'E', then the eigenvalues for the entire matrix are to be ordered from smallest to largest.

*Constraint*: ORDER = 'B' or 'E'.

3: N – INTEGER. *Input*

*On entry*: $n$, the order of the matrix $T$.

*Constraint*: N $\geq$ 0.

4: VL – ***real***. *Input*

5: VU – ***real***. *Input*

*On entry*: if RANGE = 'V', the lower and upper bounds, respectively, of the half-open interval (VL,VU] within which the required eigenvalues lie.

Not referenced if RANGE = 'A' or 'I'.

*Constraint*: VL < VU if RANGE = 'V'.

6: IL – INTEGER. *Input*

7: IU – INTEGER. *Input*

*On entry*: if RANGE = 'I', the indices of the first and last eigenvalues, respectively, to be computed (assuming that the eigenvalues are in ascending order).

Not referenced if RANGE = 'A' or 'V'.

*Constraint*: 1 $\leq$ IL $\leq$ IU $\leq$ N if RANGE = 'I'.

8: ABSTOL – ***real***. *Input*

*On entry*: the absolute tolerance to which each eigenvalue is required. An eigenvalue (or cluster) is considered to have converged if it lies in an interval of width $\leq$ ABSTOL. If ABSTOL $\leq$ 0.0, then the tolerance is taken as ***machine precision*** $\times \|T\|_1$.

9: D(*) – ***real*** array. *Input*

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

*On entry*: the diagonal elements of the tridiagonal matrix $T$.

10: E(*) – ***real*** array. *Input*

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

*On entry*: the off-diagonal elements of the tridiagonal matrix $T$.

11: M – INTEGER. *Output*

*On exit*: $m$, the actual number of eigenvalues found.

12: NSPLIT – INTEGER. *Output*

*On exit*: the number of diagonal blocks which constitute the tridiagonal matrix $T$.

13: W(*) – ***real*** array. *Output*

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

*On exit*: the required eigenvalues of the tridiagonal matrix $T$ stored in W(1) to W($m$).

14: IBLOCK(*) – INTEGER array. *Output*

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

*On exit*: at each row/column $j$ where E($j$) is zero or negligible, $T$ is considered to split into a block diagonal matrix and IBLOCK($i$) contains the block number of the eigenvalue stored in W($i$), for $i$ = 1,2,...,$m$. Note that IBLOCK($i$) < 0 for some $i$ whenever INFO = 1 or 3 (see Section 6) and RANGE = 'A' or 'V'.

15: ISPLIT(*) – INTEGER array. *Output*

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

*On exit*: the leading NSPLIT elements contain the points at which $T$ splits up into sub-matrices as follows. The first sub-matrix consists of rows/columns 1 to ISPLIT(1), the second sub-matrix consists of rows/columns ISPLIT(1) + 1 to ISPLIT(2), ..., and the NSPLIT(th) sub-matrix consists of rows/columns ISPLIT(NSPLIT–1) + 1 to ISPLIT(NSPLIT) (= $n$).

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

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

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

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

18: 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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO = 1

If RANGE = 'A' or 'V', the algorithm failed to compute some (or all) of the required eigenvalues to the desired accuracy. More precisely, IBLOCK($i$ ) < 0 indicates that the $i$th eigenvalue (stored in W($i$)) failed to converge.

INFO = 2

If RANGE = 'I', the algorithm failed to compute some (or all) of the required eigenvalues. Try calling the routine again with RANGE = 'A'.

INFO = 3

If RANGE = 'I', see the description above for INFO = 2.

If RANGE = 'A' or 'V', see the description above for INFO = 1.

INFO = 4

No eigenvalues have been computed. The floating-point arithmetic on the computer is not behaving as expected.

If failures with INFO ≥ 1 are causing persistent trouble and the user has checked that the routine is being called correctly, please contact NAG.

## 7. Accuracy

The eigenvalues of $T$ are computed to high relative accuracy which means that if they vary widely in magnitude, then any small eigenvalues will be computed more accurately than, for example, with the standard $QR$ method. However, the reduction to tridiagonal form (prior to calling the routine) may exclude the possibility of obtaining high relative accuracy in the small eigenvalues of the original matrix if its eigenvalues vary widely in magnitude.

## 8. Further Comments

There is no complex analogue of this routine.

## 9. Example

See the example for F08FGF (SORMTR/DORMTR).

# F08JKF (SSTEIN/DSTEIN) – 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

F08JKF (SSTEIN/DSTEIN) computes the eigenvectors of a real symmetric tridiagonal matrix corresponding to specified eigenvalues, by inverse iteration.

## 2. Specification

```
 SUBROUTINE F08JKF (N, D, E, M, W, IBLOCK, ISPLIT, Z, LDZ, WORK, IWORK,
1                   IFAILV, INFO)
 ENTRY       sstein  (N, D, E, M, W, IBLOCK, ISPLIT, Z, LDZ, WORK, IWORK,
1                   IFAILV, INFO)
 INTEGER          N, M, IBLOCK(*), ISPLIT(*), LDZ, IWORK(*), IFAILV(*),
1                 INFO
 real             D(*), E(*), W(*), Z(LDZ,*), WORK(*)
```

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

## 3. Description

This routine computes the eigenvectors of a real symmetric tridiagonal matrix $T$ corresponding to specified eigenvalues, by inverse iteration (see Jessup and Ipsen [2]). It is designed to be used in particular after the specified eigenvalues have been computed by F08JJF (SSTEBZ/DSTEBZ) with ORDER = 'B', but may also be used when the eigenvalues have been computed by other F08 or F02 routines.

If $T$ has been formed by reduction of a full real symmetric matrix $A$ to tridiagonal form, then eigenvectors of $T$ may be transformed to eigenvectors of $A$ by a call to F08FGF (SORMTR/DORMTR) or F08GGF (SOPMTR/DOPMTR).

F08JJF determines whether the matrix $T$ splits into block diagonal form:

$$T = \begin{pmatrix} T_1 & & & \\ & T_2 & & \\ & & \ddots & \\ & & & T_p \end{pmatrix}$$

and passes details of the block structure to this routine in the arrays IBLOCK and ISPLIT. This routine can then take advantage of the block structure by performing inverse iteration on each block $T_i$ separately, which is more efficient than using the whole matrix.

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §7.6.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

[2] JESSUP, E.R. and IPSEN, I.C.F.
Improving the Accuracy of Inverse Iteration.
SIAM J. Sci. Stat. Comput., 13, pp. 550-572, 1992.

## 5. Parameters

1: N – INTEGER. *Input*

*On entry*: $n$, the order of the matrix $T$.

*Constraint*: N $\geq$ 0.

2: D(*) – ***real*** array. *Input*

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

*On entry*: the diagonal elements of the tridiagonal matrix $T$.

3: E(*) – ***real*** array. *Input*

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

*On entry*: the off-diagonal elements of the tridiagonal matrix $T$.

4: M – INTEGER. *Input*

*On entry*: $m$, the number of eigenvectors to be returned.

*Constraint*: $0 \leq M \leq N$.

5: W(*) – ***real*** array. *Input*

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

*On entry*: the eigenvalues of the tridiagonal matrix $T$ stored in W(1) to W($m$), as returned by F08JJF (SSTEBZ/DSTEBZ) with ORDER = 'B'. Eigenvalues associated with the first sub-matrix must be supplied first, in non-decreasing order; then those associated with the second sub-matrix, again in non-decreasing order; and so on.

*Constraint*: if IBLOCK($i$) = IBLOCK($i$+1), W($i$) ≤ W($i$+1) for $i = 1,2,...,m-1$.

6: IBLOCK(*) – INTEGER array. *Input*

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

*On entry*: the first $m$ elements must contain the sub-matrix indices associated with the specified eigenvalues, as returned by F08JJF (SSTEBZ/DSTEBZ) with ORDER = 'B'. If the eigenvalues were not computed by F08JJF with ORDER = 'B', set IBLOCK($i$) to 1 for $i = 1,2,...,m$.

*Constraint*: IBLOCK($i$) ≤ IBLOCK($i$+1) for $i = 1,2,...,m-1$.

7: ISPLIT(*) – INTEGER array. *Input*

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

*On entry*: the points at which $T$ breaks up into sub-matrices, as returned by F08JJF (SSTEBZ/DSTEBZ) with ORDER = 'B'. If the eigenvalues were not computed by F08JJF with ORDER = 'B', set ISPLIT(1) to N.

8: Z(LDZ,*) – ***real*** array. *Output*

**Note**: the second dimension of the array Z must be at least max(1,M).

*On exit*: the $m$ eigenvectors, stored by columns; the $i$th column corresponds to the $i$th specified eigenvalue, unless INFO > 0 (in which case see Section 6).

9: LDZ – INTEGER. *Input*

*On entry*: the first dimension of the array Z as declared in the (sub)program from which F08JKF (SSTEIN/DSTEIN) is called.

*Constraint*: LDZ ≥ max(1,N).

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

**Note**: the dimension of the array WORK must be at least max(1,5*N).

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

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

12: IFAILV(*) – INTEGER array. *Output*

**Note**: the dimension of the array IFAILV must be at least max(1,M).

*On exit*: if INFO = $i > 0$, the first $i$ elements of IFAILV contain the indices of any eigenvectors which have failed to converge. The rest of the first M elements of IFAILV are set to 0.

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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = $i$, then $i$ eigenvectors (as indicated by the parameter IFAILV above) each failed to converge in 5 iterations. The current iterate after 5 iterations is stored in the corresponding column of Z.

## 7. Accuracy

Each computed eigenvector $z_i$ is the exact eigenvector of a nearby matrix $A + E_i$, such that $\|E_i\| = O(\varepsilon)\|A\|$, where $\varepsilon$ is the ***machine precision***. Hence the residual is small:

$$\|Az_i - \lambda_i z_i\| = O(\varepsilon)\|A\|.$$

However a set of eigenvectors computed by this routine may not be orthogonal to so high a degree of accuracy as those computed by F08JEF (SSTEQR/DSTEQR).

## 8. Further Comments

The complex analogue of this routine is F08JXF (CSTEIN/ZSTEIN).

## 9. Example

See the example for F08FGF (SORMTR/DORMTR).

# F08JSF (CSTEQR/ZSTEQR) – 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

F08JSF (CSTEQR/ZSTEQR) computes all the eigenvalues, and optionally all the eigenvectors, of a complex Hermitian matrix which has been reduced to tridiagonal form.

## 2. Specification

```
SUBROUTINE F08JSF (COMPZ, N, D, E, Z, LDZ, WORK, INFO)
ENTRY      csteqr  (COMPZ, N, D, E, Z, LDZ, WORK, INFO)
INTEGER          N, LDZ, INFO
real             D(*), E(*), WORK(*)
complex          Z(LDZ,*)
CHARACTER*1      COMPZ
```

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

## 3. Description

This routine computes all the eigenvalues, and optionally all the eigenvectors, of a real symmetric tridiagonal matrix $T$. In other words, it can compute the spectral factorization of $T$ as

$$T = Z\Lambda Z^T,$$

where $\Lambda$ is a diagonal matrix whose diagonal elements are the eigenvalues $\lambda_i$, and $Z$ is the orthogonal matrix whose columns are the eigenvectors $z_i$. Thus

$$Tz_i = \lambda_i z_i \text{ for } i = 1,2,...,n.$$

The routine stores the real orthogonal matrix $Z$ in a ***complex*** array, so that it may also be used to compute all the eigenvalues and eigenvectors of a complex Hermitian matrix $A$ which has been reduced to tridiagonal form $T$:

$$\begin{aligned} A &= QTQ^H, \text{ where } Q \text{ is unitary,} \\ &= (QZ)\Lambda(QZ)^H. \end{aligned}$$

In this case, the matrix $Q$ must be formed explicitly and passed to F08JSF, which must be called with COMPZ = 'V'. The routines which must be called to perform the reduction to tridiagonal form and form $Q$ are:

| | |
|---|---|
| full matrix | F08FSF (CHETRD/ZHETRD) + F08FTF (CUNGTR/ZUNGTR) |
| full matrix, packed storage | F08GSF (CHPTRD/ZHPTRD) + F08GTF (CUPGTR/ZUPGTR) |
| band matrix | F08HSF (CHBTRD/ZHBTRD) with VECT = 'V'. |

F08JSF uses the implicitly shifted $QR$ algorithm, switching between the $QR$ and $QL$ variants in order to handle graded matrices effectively (see Greenbaum and Dongarra [2]). The eigenvectors are normalized so that $\|z_i\|_2 = 1$, but are determined only to within a complex factor of absolute value 1.

If only the eigenvalues of $T$ are required, it is more efficient to call F08JFF (SSTERF/DSTERF) instead. If $T$ is positive-definite, small eigenvalues can be computed more accurately by F08JUF (CPTEQR/ZPTEQR).

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §8.2.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

[2] GREENBAUM, A. and DONGARRA, J.J.
Experiments with QR/QL Methods for the Symmetric Tridiagonal Eigenproblem.
LAPACK Working Note No. 17 (Technical Report CS-89-92), University of Tennessee, Knoxville, 1989.

[3] PARLETT, B.N.
The Symmetric Eigenvalue Problem, §8.
Prentice-Hall, Englewood Cliffs, New Jersey, 1980.

## 5. Parameters

1: COMPZ – CHARACTER*1. *Input*

*On entry*: indicates whether the eigenvectors are to be computed as follows:

if COMPZ = 'N', then only the eigenvalues are computed (and the array Z is not referenced);

if COMPZ = 'I', then the eigenvalues and eigenvectors of $T$ are computed (and the array Z is initialized by the routine);

if COMPZ = 'V', then the eigenvalues and eigenvectors of $A$ are computed (and the array Z must contain the matrix $Q$ on entry).

*Constraint*: COMPZ = 'N', 'I' or 'V'.

2: N – INTEGER. *Input*

*On entry*: $n$, the order of the matrix $T$.

*Constraint*: N ≥ 0.

3: D(*) – ***real*** array. *Input/Output*

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

*On entry*: the diagonal elements of the tridiagonal matrix $T$.

*On exit*: the $n$ eigenvalues in ascending order, unless INFO > 0 (in which case see Section 6).

4: E(*) – ***real*** array. *Input/Output*

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

*On entry*: the off-diagonal elements of the tridiagonal matrix $T$.

*On exit*: the array is overwritten.

5: Z(LDZ,*) – ***complex*** array. *Input/Output*

**Note**: the second dimension of the array Z must be at least max(1,N) if COMPZ = 'V' or 'I', and at least 1 if COMPZ = 'N'.

*On entry*: if COMPZ = 'V', Z must contain the unitary matrix $Q$ from the reduction to tridiagonal form. If COMPZ = 'I', Z need not be set.

*On exit*: if COMPZ = 'I' or 'V', the $n$ required orthonormal eigenvectors stored by columns; the $i$th column corresponds to the $i$th eigenvalue, where $i = 1,2,...,n$, unless INFO > 0.

Z is not referenced if COMPZ = 'N'.

6: LDZ – INTEGER. *Input*

*On entry*: the first dimension of the array Z as declared in the (sub)program from which F08JSF (CSTEQR/ZSTEQR) is called.

*Constraints*: LDZ ≥ 1 if COMPZ = 'N',
LDZ ≥ max(1,N) if COMPZ = 'V' or 'I'.

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

**Note**: the dimension of the array WORK must be at least max(1,2*(N–1)) if COMPZ = 'V' or 'I', and at least 1 if COMPZ = 'N'.

WORK is not referenced if COMPZ = '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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

The algorithm has failed to find all the eigenvalues after a total of 30×N iterations. In this case, D and E contain on exit the diagonal and off-diagonal elements, respectively, of a tridiagonal matrix orthogonally similar to $T$. If INFO = $i$, then $i$ off-diagonal elements have not converged to zero.

## 7. Accuracy

The computed eigenvalues and eigenvectors are exact for a nearby matrix $T + E$, where

$$\|E\|_2 = O(\varepsilon)\|T\|_2,$$

and $\varepsilon$ is the ***machine precision***.

If $\lambda_i$ is an exact eigenvalue and $\tilde{\lambda}_i$ is the corresponding computed value, then

$$|\tilde{\lambda}_i - \lambda_i| \leq c(n)\varepsilon\|T\|_2,$$

where $c(n)$ is a modestly increasing function of $n$.

If $z_i$ is the corresponding exact eigenvector, and $\tilde{z}_i$ is the corresponding computed eigenvector, then the angle $\theta(\tilde{z}_i, z_i)$ between them is bounded as follows:

$$\theta(\tilde{z}_i, z_i) \leq \frac{c(n)\varepsilon\|T\|_2}{\min_{i \neq j}|\lambda_i - \lambda_j|}.$$

Thus the accuracy of a computed eigenvector depends on the gap between its eigenvalue and all the other eigenvalues.

## 8. Further Comments

The total number of real floating-point operations is typically about $24n^2$ if COMPZ = 'N' and about $14n^3$ if COMPZ = 'V' or 'I', but depends on how rapidly the algorithm converges. When COMPZ = 'N', the operations are all performed in scalar mode; the additional operations to compute the eigenvectors when COMPZ = 'V' or 'I' can be vectorized and on some machines may be performed much faster.

The real analogue of this routine is F08JEF (SSTEQR/DSTEQR).

## 9. Example

See the examples for F08FTF (CUNGTR/ZUNGTR), F08GTF (CUPGTR/ZUPGTR) or F08HSF (CHBTRD/ZHBTRD), which illustrate the use of this routine to compute the eigenvalues and eigenvectors of a full or band Hermitian matrix.

# F08JUF (CPTEQR/ZPTEQR) – 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

F08JUF (CPTEQR/ZPTEQR) computes all the eigenvalues, and optionally all the eigenvectors, of a complex Hermitian positive-definite matrix which has been reduced to tridiagonal form.

## 2. Specification

```
SUBROUTINE F08JUF (COMPZ, N, D, E, Z, LDZ, WORK, INFO)
ENTRY      cpteqr  (COMPZ, N, D, E, Z, LDZ, WORK, INFO)
INTEGER          N, LDZ, INFO
real             D(*), E(*), WORK(*)
complex          Z(LDZ,*)
CHARACTER*1      COMPZ
```

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

## 3. Description

This routine computes all the eigenvalues, and optionally all the eigenvectors, of a real symmetric positive-definite tridiagonal matrix $T$. In other words, it can compute the spectral factorization of $T$ as

$$T = Z\Lambda Z^T,$$

where $\Lambda$ is a diagonal matrix whose diagonal elements are the eigenvalues $\lambda_i$, and $Z$ is the orthogonal matrix whose columns are the eigenvectors $z_i$. Thus

$$Tz_i = \lambda_i z_i \text{ for } i = 1,2,...,n.$$

The routine stores the real orthogonal matrix $Z$ in a ***complex*** array, so that it may be used to compute all the eigenvalues and eigenvectors of a complex Hermitian positive-definite matrix $A$ which has been reduced to tridiagonal form $T$:

$$A = QTQ^H, \text{ where } Q \text{ is unitary}$$
$$= (QZ)\Lambda(QZ)^H.$$

In this case, the matrix $Q$ must be formed explicitly and passed to F08JUF, which is called with COMPZ = 'V'. The routines which must be called to perform the reduction to tridiagonal form and form $Q$ are:

| | |
|---|---|
| full matrix | F08FSF (CHETRD/ZHETRD) + F08FTF (CUNGTR/ZUNGTR) |
| full matrix, packed storage | F08GSF (CHPTRD/ZHPTRD) + F08GTF (CUPGTR/ZUPGTR) |
| band matrix | F08HSF (CHBTRD/ZHBTRD) with VECT = 'V'. |

The routine first factorizes $T$ as $LDL^H$ where $L$ is unit lower bidiagonal and $D$ is diagonal. It forms the bidiagonal matrix $B = LD^{\frac{1}{2}}$, and then calls F08MSF (CBDSQR/ZBDSQR) to compute the singular values of $B$ which are the same as the eigenvalues of $T$. The method used by the routine allows high relative accuracy to be achieved in the small eigenvalues of $T$. The eigenvectors are normalized so that $\|z_i\|_2 = 1$, but are determined only to within a complex factor of absolute value 1.

## 4. References

[1] BARLOW, J. and DEMMEL, J.
Computing Accurate Eigensystems of Scaled Diagonally Dominant Matrices.
SIAM J. Num. Anal., 27, pp. 762-791, 1990.

## 5. Parameters

1: COMPZ – CHARACTER*1. *Input*

*On entry*: indicates whether the eigenvectors are to be computed as follows:

if COMPZ = 'N', then only the eigenvalues are computed (and the array Z is not referenced);

if COMPZ = 'I', then the eigenvalues and eigenvectors of $T$ are computed (and the array Z is initialized by the routine);

if COMPZ = 'V', then the eigenvalues and eigenvectors of $A$ are computed (and the array Z must contain the matrix $Q$ on entry).

*Constraint*: COMPZ = 'N', 'I' or 'V'.

2: N – INTEGER. *Input*

*On entry*: $n$, the order of the matrix $T$.

*Constraint*: N ≥ 0.

3: D(*) – ***real*** array. *Input/Output*

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

*On entry*: the diagonal elements of the tridiagonal matrix $T$.

*On exit*: the $n$ eigenvalues in descending order, unless INFO > 0, in which case the array is overwritten.

4: E(*) – ***real*** array. *Input/Output*

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

*On entry*: the off-diagonal elements of the tridiagonal matrix $T$.

*On exit*: the array is overwritten.

5: Z(LDZ,*) – ***complex*** array. *Input/Output*

**Note**: the second dimension of the array Z must be at least max(1,N) if COMPZ = 'V' or 'I', and at least 1 if COMPZ = 'N'.

*On entry*: if COMPZ = 'V', Z must contain the unitary matrix $Q$ from the reduction to tridiagonal form. If COMPZ = 'I', Z need not be set.

*On exit*: if COMPZ = 'I' or 'V', the $n$ required orthonormal eigenvectors stored by columns; the $i$th column corresponds to the $i$th eigenvalue, where $i = 1,2,...,n$, unless INFO > 0.

Z is not referenced if COMPZ = 'N'.

6: LDZ – INTEGER. *Input*

*On entry*: the first dimension of the array Z as declared in the (sub)program from which F08JUF (CPTEQR/ZPTEQR) is called.

*Constraints*: LDZ ≥ 1 if COMPZ = 'N',
LDZ ≥ max(1,N) if COMPZ = 'V' or 'I'.

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

**Note**: the dimension of the array WORK must be at least max(1,4*(N–1)) if COMPZ = 'V' or 'I', and at least 1 if COMPZ = 'N'.

WORK is not referenced if COMPZ = '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 $i$th 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 Cholesky factorization of $T$ could not be completed. Hence $T$ itself is not positive-definite.

If INFO = N + $i$, the algorithm to compute the singular values of the Cholesky factor $B$ failed to converge; $i$ off-diagonal elements did not converge to zero.

## 7. Accuracy

The eigenvalues and eigenvectors of $T$ are computed to high relative accuracy which means that if they vary widely in magnitude, then any small eigenvalues (and corresponding eigenvectors) will be computed more accurately than, for example, with the standard $QR$ method. However, the reduction to tridiagonal form (prior to calling the routine) may exclude the possibility of obtaining high relative accuracy in the small eigenvalues of the original matrix if its eigenvalues vary widely in magnitude.

To be more precise, let $H$ be the tridiagonal matrix defined by $H = DTD$, where $D$ is diagonal with $d_{ii} = t_{ii}^{-\frac{1}{2}}$, and $h_{ii} = 1$ for all $i$. If $\lambda_i$ is an exact eigenvalue of $T$ and $\tilde{\lambda}_i$ is the corresponding computed value, then

$$|\tilde{\lambda}_i - \lambda_i| \leq c(n)\varepsilon\kappa_2(H)\lambda_i$$

where $c(n)$ is a modestly increasing function of $n$, $\varepsilon$ is the ***machine precision***, and $\kappa_2(H)$ is the condition number of $H$ with respect to inversion defined by: $\kappa_2(H) = \|H\|.\|H^{-1}\|$.

If $z_i$ is the corresponding exact eigenvector of $T$, and $\tilde{z}_i$ is the corresponding computed eigenvector, then the angle $\theta(\tilde{z}_i, z_i)$ between them is bounded as follows:

$$\theta(\tilde{z}_i, z_i) \leq \frac{c(n)\varepsilon\kappa_2(H)}{relgap_i}$$

where $relgap_i$ is the relative gap between $\lambda_i$ and the other eigenvalues, defined by

$$relgap_i = \min_{i \neq j} \frac{|\lambda_i - \lambda_j|}{(\lambda_i + \lambda_j)}.$$

## 8. Further Comments

The total number of real floating-point operations is typically about $30n^2$ if COMPZ = 'N' and about $12n^3$ if COMPZ = 'V' or 'I', but depends on how rapidly the algorithm converges. When COMPZ = 'N', the operations are all performed in scalar mode; the additional operations to compute the eigenvectors when COMPZ = 'V' or 'I' can be vectorized and on some machines may be performed much faster.

The real analogue of this routine is F08JGF (SPTEQR/DPTEQR).

## 9. Example

To compute all the eigenvalues and eigenvectors of the complex Hermitian positive-definite matrix $A$, where

$$A = \begin{pmatrix} 6.02 + 0.00i & -0.45 + 0.25i & -1.30 + 1.74i & 1.45 - 0.66i \\ -0.45 - 0.25i & 2.91 + 0.00i & 0.05 + 1.56i & -1.04 + 1.27i \\ -1.30 - 1.74i & 0.05 - 1.56i & 3.29 + 0.00i & 0.14 + 1.70i \\ 1.45 + 0.66i & -1.04 - 1.27i & 0.14 - 1.70i & 4.18 + 0.00i \end{pmatrix}.$$

## 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.**

```
*     F08JUF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, LWORK, LDZ
      PARAMETER        (NMAX=8,LDA=NMAX,LWORK=64*NMAX,LDZ=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          A(LDA,NMAX), TAU(NMAX), WORK(LWORK), Z(LDZ, NMAX)
      real             D(NMAX), E(NMAX), RWORK(4*NMAX-4)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         F06TFF, X04DBF, chetrd, cpteqr, cungtr
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08JUF 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
*
*        Reduce A to tridiagonal form T = (Q**H)*A*Q
*
         CALL chetrd(UPLO,N,A,LDA,D,E,TAU,WORK,LWORK,INFO)
*
*        Copy A into Z
*
         CALL F06TFF(UPLO,N,N,A,LDA,Z,LDZ)
*
*        Form Q explicitly, storing the result in Z
*
         CALL cungtr(UPLO,N,Z,LDZ,TAU,WORK,LWORK,INFO)
*
*        Calculate all the eigenvalues and eigenvectors of A
*
         CALL cpteqr('V',N,D,E,Z,LDZ,RWORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.GT.0) THEN
            WRITE (NOUT,*) 'Failure to converge.'
         ELSE
*
*           Print eigenvalues and eigenvectors
*
            WRITE (NOUT,*) 'Eigenvalues'
            WRITE (NOUT,99999) (D(I),I=1,N)
            WRITE (NOUT,*)
            IFAIL = 0
*
            CALL X04DBF('General',' ',N,N,Z,LDZ,'Bracketed','F7.4',
     +                  'Eigenvectors','Integer',RLABS,'Integer',CLABS,
     +                  80,0,IFAIL)
*
```

```
         END IF
      END IF
      STOP
*
99999 FORMAT (8X,4(F7.4,11X,:))
      END
```

## 9.2. Program Data

```
F08JUF Example Program Data
  4                                                                  :Value of N
  'L'                                                                :Value of UPLO
 ( 6.02, 0.00)
 (-0.45,-0.25) ( 2.91, 0.00)
 (-1.30,-1.74) ( 0.05,-1.56) ( 3.29, 0.00)
 ( 1.45, 0.66) (-1.04,-1.27) ( 0.14,-1.70) ( 4.18, 0.00)    :End of matrix A
```

## 9.3. Program Results

```
F08JUF Example Program Results

Eigenvalues
        7.9995            5.9976            2.0003            0.4026

Eigenvectors
                    1                 2                 3                 4
1  ( 0.7289, 0.0000) ( 0.2001, 0.4724) (-0.2133, 0.1498) ( 0.0995,-0.3573)
2  (-0.1651,-0.2067) (-0.2461, 0.3742) ( 0.7308, 0.0000) ( 0.2867,-0.3364)
3  (-0.4170,-0.1413) ( 0.4476, 0.1455) (-0.3282, 0.0471) ( 0.6890, 0.0000)
4  ( 0.1748, 0.4175) ( 0.5610, 0.0000) ( 0.5203, 0.1317) ( 0.0659, 0.4336)
```

# F08JXF (CSTEIN/ZSTEIN) – 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

F08JXF (CSTEIN/ZSTEIN) computes the eigenvectors of a real symmetric tridiagonal matrix corresponding to specified eigenvalues, by inverse iteration, storing the eigenvectors in a ***complex*** array.

## 2. Specification

```
 SUBROUTINE F08JXF (N, D, E, M, W, IBLOCK, ISPLIT, Z, LDZ, WORK, IWORK,
1                   IFAILV, INFO)
 ENTRY       cstein (N, D, E, M, W, IBLOCK, ISPLIT, Z, LDZ, WORK, IWORK,
1                   IFAILV, INFO)
 INTEGER          N, M, IBLOCK(*), ISPLIT(*), LDZ, IWORK(*), IFAILV(*),
1                 INFO
 real             D(*), E(*), W(*), WORK(*)
 complex          Z(LDZ,*)
```

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

## 3. Description

This routine computes the eigenvectors of a real symmetric tridiagonal matrix $T$ corresponding to specified eigenvalues, by inverse iteration (see Jessup and Ipsen [2]). It is designed to be used in particular after the specified eigenvalues have been computed by F08JJF (SSTEBZ/DSTEBZ) with ORDER = 'B', but may also be used when the eigenvalues have been computed by other F08 or F02 routines.

The eigenvectors of $T$ are real, but are stored by this routine in a ***complex*** array. If $T$ has been formed by reduction of a full complex Hermitian matrix $A$ to tridiagonal form, then eigenvectors of $T$ may be transformed to (complex) eigenvectors of $A$, by a call to F08FUF (CUNMTR/ZUNMTR) or F08GUF (CUPMTR/ZUPMTR).

F08JJF determines whether the matrix $T$ splits into block diagonal form:

$$T = \begin{pmatrix} T_1 & & & \\ & T_2 & & \\ & & \ddots & \\ & & & T_p \end{pmatrix}$$

and passes details of the block structure to this routine in the arrays IBLOCK and ISPLIT. This routine can then take advantage of the block structure by performing inverse iteration on each block $T_i$ separately, which is more efficient than using the whole matrix.

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §7.6.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

[2] JESSUP, E.R. and IPSEN, I.C.F.
Improving the Accuracy of Inverse Iteration.
SIAM J. Sci. Stat. Comput., 13, pp. 550-572, 1992.

## 5. Parameters

1: N – INTEGER. *Input*

*On entry*: $n$, the order of the matrix $T$.

*Constraint*: N ≥ 0.

2: D(*) – ***real*** array. *Input*

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

*On entry*: the diagonal elements of the tridiagonal matrix $T$.

3: E(*) – ***real*** array. *Input*

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

*On entry*: the off-diagonal elements of the tridiagonal matrix $T$.

4: M – INTEGER. *Input*

*On entry*: $m$, the number of eigenvectors to be returned.

*Constraint*: 0 ≤ M ≤ N.

5: W(*) – ***real*** array. *Input*

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

*On entry*: the eigenvalues of the tridiagonal matrix $T$ stored in W(1) to W($m$), as returned by F08JJF (SSTEBZ/DSTEBZ) with ORDER = 'B'. Eigenvalues associated with the first sub-matrix must be supplied first, in non-decreasing order; then those associated with the second sub-matrix, again in non-decreasing order; and so on.

*Constraint*: if IBLOCK($i$) = IBLOCK($i$+1), W($i$) ≤ W($i$+1) for $i$ = 1,2,...,$m$–1.

6: IBLOCK(*) – INTEGER array. *Input*

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

*On entry*: the first $m$ elements must contain the sub-matrix indices associated with the specified eigenvalues, as returned by F08JJF (SSTEBZ/DSTEBZ) with ORDER = 'B'. If the eigenvalues were not computed by F08JJF with ORDER = 'B', set IBLOCK($i$) to 1 for $i$ = 1,2,...,$m$.

*Constraint*: IBLOCK($i$) ≤ IBLOCK($i$+1) for $i$ = 1,2,...,$m$–1.

7: ISPLIT(*) – INTEGER array. *Input*

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

*On entry*: the points at which $T$ breaks up into sub-matrices, as returned by F08JJF (SSTEBZ/DSTEBZ) with ORDER = 'B'. If the eigenvalues were not computed by F08JJF with ORDER = 'B', set ISPLIT(1) to N.

8: Z(LDZ,*) – ***complex*** array. *Output*

**Note**: the second dimension of the array Z must be at least max(1,M).

*On exit*: the $m$ eigenvectors, stored by columns; the $i$th column corresponds to the $i$th specified eigenvalue, unless INFO > 0 (in which case see Section 6).

9: LDZ – INTEGER. *Input*

*On entry*: the first dimension of the array Z as declared in the (sub)program from which F08JXF (CSTEIN/ZSTEIN) is called.

*Constraint*: LDZ ≥ max(1,N).

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

**Note**: the dimension of the array WORK must be at least max(1,5*N).

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

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

12: IFAILV(*) – INTEGER array. *Output*

**Note**: the dimension of the array IFAILV must be at least max(1,M).

*On exit*: if INFO = $i > 0$, the first $i$ elements of IFAILV contain the indices of any eigenvectors which have failed to converge. The rest of the first M elements of IFAILV are set to 0.

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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = $i$, then $i$ eigenvectors (as indicated by the parameter IFAILV above) each failed to converge in 5 iterations. The current iterate after 5 iterations is stored in the corresponding column of Z.

## 7. Accuracy

Each computed eigenvector $z_i$ is the exact eigenvector of a nearby matrix $A + E_i$, such that $\|E_i\| = O(\varepsilon)\|A\|$, where $\varepsilon$ is the ***machine precision***. Hence the residual is small:

$$\|Az_i - \lambda_i z_i\| = O(\varepsilon)\|A\|.$$

However a set of eigenvectors computed by this routine may not be orthogonal to so high a degree of accuracy as those computed by F08JSF (CSTEQR/ZSTEQR).

## 8. Further Comments

The real analogue of this routine is F08JKF (SSTEIN/DSTEIN).

## 9. Example

See the example for F08FUF (CUNMTR/ZUNMTR).

# F08KEF (SGEBRD/DGEBRD) – 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

F08KEF (SGEBRD/DGEBRD) reduces a real $m$ by $n$ matrix to bidiagonal form.

## 2. Specification

```
SUBROUTINE F08KEF (M, N, A, LDA, D, E, TAUQ, TAUP, WORK, LWORK, INFO)
ENTRY      sgebrd (M, N, A, LDA, D, E, TAUQ, TAUP, WORK, LWORK, INFO)
INTEGER        M, N, LDA, LWORK, INFO
real           A(LDA,*), D(*), E(*), TAUQ(*), TAUP(*), WORK(LWORK)
```

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

## 3. Description

This routine reduces a real $m$ by $n$ matrix $A$ to bidiagonal form $B$ by an orthogonal transformation: $A = QBP^T$, where $Q$ and $P^T$ are orthogonal matrices of order $m$ and $n$ respectively.

If $m \geq n$, the reduction is given by:

$$A = Q\begin{pmatrix} B_1 \\ 0 \end{pmatrix}P^T = Q_1 B_1 P^T,$$

where $B_1$ is an $n$ by $n$ upper bidiagonal matrix and $Q_1$ consists of the first $n$ columns of $Q$.

If $m < n$, the reduction is given by

$$A = Q(B_1 \ 0)P^T = QB_1P_1^T,$$

where $B_1$ is an $m$ by $m$ lower bidiagonal matrix and $P_1^T$ consists of the first $m$ rows of $P^T$.

The orthogonal matrices $Q$ and $P$ are not formed explicitly but are represented as products of elementary reflectors (see the Chapter Introduction for details). Routines are provided to work with $Q$ and $P$ in this representation. (see Section 8).

## 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 diagonal and first super-diagonal are overwritten by the upper bidiagonal matrix $B$, elements below the diagonal are overwritten by details of the

orthogonal matrix $Q$ and elements above the first super-diagonal are overwritten by details of the orthogonal matrix $P$.

If $m < n$, the diagonal and first sub-diagonal are overwritten by the lower bidiagonal matrix $B$, elements below the first sub-diagonal are overwritten by details of the orthogonal matrix $Q$ and elements above the diagonal are overwritten by details of the orthogonal matrix $P$.

4: LDA – INTEGER. *Input*

*On entry*: the first dimension of the array A as declared in the (sub)program from which F08KEF (SGEBRD/DGEBRD) is called.

*Constraint*: LDA ≥ max(1,M).

5: D(*) – ***real*** array. *Output*

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

*On exit*: the diagonal elements of the bidiagonal matrix $B$.

6: E(*) – ***real*** array. *Output*

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

*On exit*: the off-diagonal elements of the bidiagonal matrix $B$.

7: TAUQ(*) – ***real*** array. *Output*

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

*On exit*: further details of the orthogonal matrix $Q$.

8: TAUP(*) – ***real*** array. *Output*

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

*On exit*: further details of the orthogonal matrix $P$.

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

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

10: LWORK – INTEGER. *Input*

*On entry*: the dimension of the array WORK as declared in the (sub)program from which F08KEF (SGEBRD/DGEBRD) is called.

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

*Constraint*: LWORK ≥ max(1,M,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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

## 7. Accuracy

The computed bidiagonal form $B$ satisfies $QBP^T = A + E$, where

$$\|E\|_2 \leq c(n)\varepsilon\|A\|_2,$$

$c(n)$ is a modestly increasing function of $n$, and $\varepsilon$ is the ***machine precision***.

The elements of $B$ themselves may be sensitive to small perturbations in $A$ or to rounding errors in the computation, but this does not affect the stability of the singular values and vectors.

## 8. Further Comments

The total number of 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 $m < n$.

If $m \gg n$, it can be more efficient to first call F08AEF (SGEQRF/DGEQRF) to perform a $QR$ factorization of $A$, and then to call this routine to reduce the factor $R$ to bidiagonal form. This requires approximately $2n^2(m+n)$ floating-point operations.

If $m \ll n$, it can be more efficient to first call F08AHF (SGELQF/DGELQF) to perform an $LQ$ factorization of $A$, and then to call this routine to reduce the factor $L$ to bidiagonal form. This requires approximately $2m^2(m+n)$ operations.

To form the orthogonal matrices $P^T$ and/or $Q$, this routine may be followed by calls to F08KFF (SORGBR/DORGBR):

to form the $m$ by $m$ orthogonal matrix $Q$

```
CALL SORGBR ('Q',M,M,N,A,LDA,TAUQ,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 F08KEF;

to form the $n$ by $n$ orthogonal matrix $P^T$

```
CALL SORGBR ('P',N,N,M,A,LDA,TAUP,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 F08KEF.

To apply $Q$ or $P$ to a real rectangular matrix $C$, this routine may be followed by a call to F08KGF (SORMBR/DORMBR).

The complex analogue of this routine is F08KSF (CGEBRD/ZGEBRD).

## 9. Example

To reduce the matrix $A$ to bidiagonal form, where

$$A = \begin{pmatrix} -0.57 & -1.28 & -0.39 & 0.25 \\ -1.93 & 1.08 & -0.31 & -2.14 \\ 2.30 & 0.24 & 0.40 & -0.35 \\ -1.93 & 0.64 & -0.66 & 0.08 \\ 0.15 & 0.30 & 0.15 & -2.13 \\ -0.02 & 1.03 & -1.43 & 0.50 \end{pmatrix}.$$

### 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.**

```
*     F08KEF 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+NMAX))
*     .. Local Scalars ..
      INTEGER          I, INFO, J, M, N
```

```
*     .. Local Arrays ..
      real               A(LDA,NMAX), D(NMAX), E(NMAX-1), TAUP(NMAX),
     +                   TAUQ(NMAX), WORK(LWORK)
*     .. External Subroutines ..
      EXTERNAL           sgebrd
*     .. Intrinsic Functions ..
      INTRINSIC          MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08KEF 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)
*
*        Reduce A to bidiagonal form
*
         CALL sgebrd(M,N,A,LDA,D,E,TAUQ,TAUP,WORK,LWORK,INFO)
*
*        Print bidiagonal form
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Diagonal'
         WRITE (NOUT,99999) (D(I),I=1,MIN(M,N))
         IF (M.GE.N) THEN
            WRITE (NOUT,*) 'Super-diagonal'
         ELSE
            WRITE (NOUT,*) 'Sub-diagonal'
         END IF
         WRITE (NOUT,99999) (E(I),I=1,MIN(M,N)-1)
      END IF
      STOP
*
99999 FORMAT (1X,8F9.4)
      END
```

## 9.2. Program Data

```
F08KEF 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

```
F08KEF Example Program Results

Diagonal
   3.6177   2.4161  -1.9213  -1.4265
Super-diagonal
   1.2587   1.5262  -1.1895
```

# F08KFF (SORGBR/DORGBR) – 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

F08KFF (SORGBR/DORGBR) generates one of the real orthogonal matrices $Q$ or $P^T$ which were determined by F08KEF (SGEBRD/DGEBRD) when reducing a real matrix to bidiagonal form.

## 2. Specification

```
SUBROUTINE F08KFF (VECT, M, N, K, A, LDA, TAU, WORK, LWORK, INFO)
ENTRY      sorgbr (VECT, M, N, K, A, LDA, TAU, WORK, LWORK, INFO)
INTEGER          M, N, K, LDA, LWORK, INFO
real             A(LDA,*), TAU(*), WORK(LWORK)
CHARACTER*1      VECT
```

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 F08KEF (SGEBRD/DGEBRD), which reduces a real rectangular matrix $A$ to bidiagonal form $B$ by an orthogonal transformation: $A = QBP^T$. F08KEF represents the matrices $Q$ and $P^T$ as products of elementary reflectors.

This routine may be used to generate $Q$ or $P^T$ explicitly as square matrices, or in some cases just the leading columns of $Q$ or the leading rows of $P^T$.

The various possibilities are specified by the parameters VECT, M, N and K. The appropriate values to cover the most likely cases are as follows (assuming that $A$ was an $m$ by $n$ matrix):

1. To form the full $m$ by $m$ matrix $Q$:

   ```
   CALL SORGBR ('Q',m,m,n,...)
   ```

   (note that the array A must have at least $m$ columns).

2. If $m > n$, to form the $n$ leading columns of $Q$:

   ```
   CALL SORGBR ('Q',m,n,n,...)
   ```

3. To form the full $n$ by $n$ matrix $P^T$:

   ```
   CALL SORGBR ('P',n,n,m,...)
   ```

   (note that the array A must have at least $n$ rows).

4. If $m < n$, to form the $m$ leading rows of $P^T$:

   ```
   CALL SORGBR ('P',m,n,m,...)
   ```

## 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: VECT – CHARACTER*1. *Input*

*On entry*: indicates whether the orthogonal matrix $Q$ or $P^T$ is generated as follows:

if VECT = 'Q', then $Q$ is generated;

if VECT = 'P', then $P^T$ is generated.

*Constraint*: VECT = 'Q' or 'P'.

2: M – INTEGER. *Input*

*On entry*: the number of rows of the orthogonal matrix $Q$ or $P^T$ to be returned.

*Constraint*: M ≥ 0.

3: N – INTEGER. *Input*

*On entry*: the number of columns of the orthogonal matrix $Q$ or $P^T$ to be returned.

*Constraints*: N ≥ 0.

If VECT = 'Q', M ≥ N ≥ K if M > K, or
M = N if M ≤ K;
if VECT = 'P', N ≥ M ≥ K if N > K, or
N = M if N ≤ K.

4: K – INTEGER. *Input*

*On entry*: if VECT = 'Q', the number of columns in the original matrix $A$; if VECT = 'P', the number of rows in the original matrix $A$.

*Constraint*: K ≥ 0.

5: 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 F08KEF (SGEBRD/DGEBRD).

*On exit*: the orthogonal matrix $Q$ or $P^T$, or the leading rows or columns thereof, as specified by VECT, M and N.

6: LDA – INTEGER. *Input*

*On entry*: the first dimension of the array A as declared in the (sub)program from which F08KFF (SORGBR/DORGBR) is called.

*Constraint*: LDA ≥ max(1,M).

7: TAU(*) – ***real*** array. *Input*

**Note**: the dimension of the array TAU must be at least max(1,min(M,K)) if VECT = 'Q', and at least max(1,min(N,K)) if VECT = 'P'.

*On entry*: further details of the elementary reflectors, as returned by F08KEF (SGEBRD/DGEBRD) in its parameter TAUQ if VECT = 'Q', or in its parameter TAUP if VECT = 'P'.

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

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

9: LWORK – INTEGER. *Input*

*On entry*: the dimension of the array WORK as declared in the (sub)program from which F08KFF (SORGBR/DORGBR) is called.

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

*Constraint*: LWORK ≥ max(1,min(M,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 $i$th 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 $\varepsilon$ is the ***machine precision***. A similar statement holds for the computed matrix $P^T$.

## 8. Further Comments

The total number of floating-point operations for the cases listed in Section 3 are approximately as follows:

1. To form the whole of $Q$: $\frac{4}{3}n(3m^2-3mn+n^2)$ if $m > n$, $\frac{4}{3}m^3$ if $m \leq n$;
2. To form the $n$ leading columns of $Q$ when $m > n$: $\frac{2}{3}n^2(3m-n)$;
3. To form the whole of $P^T$: $\frac{4}{3}n^3$ if $m \geq n$, $\frac{4}{3}m(3n^2-3mn+m^2)$ if $m < n$;
4. To form the $m$ leading rows of $P^T$ when $m < n$: $\frac{2}{3}m^2(3n-m)$.

The complex analogue of this routine is F08KTF (CUNGBR/ZUNGBR).

## 9. Example

For this routine two examples are presented, both of which involve computing the singular value decomposition of a matrix $A$, where

$$A = \begin{pmatrix} -0.57 & -1.28 & -0.39 & 0.25 \\ -1.93 & 1.08 & -0.31 & -2.14 \\ 2.30 & 0.24 & 0.40 & -0.35 \\ -1.93 & 0.64 & -0.66 & 0.08 \\ 0.15 & 0.30 & 0.15 & -2.13 \\ -0.02 & 1.03 & -1.43 & 0.50 \end{pmatrix}$$

in the first example and

$$A = \begin{pmatrix} -5.42 & 3.28 & -3.68 & 0.27 & 2.06 & 0.46 \\ -1.65 & -3.40 & -3.20 & -1.03 & -4.06 & -0.01 \\ -0.37 & 2.35 & 1.90 & 4.31 & -1.76 & 1.13 \\ -3.15 & -0.11 & 1.99 & -2.70 & 0.26 & 4.50 \end{pmatrix}$$

in the second. $A$ must first be reduced to tridiagonal form by F08KEF (SGEBRD/DGEBRD). The program then calls F08KFF (SORGBR/DORGBR) twice to form $Q$ and $P^T$, and passes these matrices to F08MEF (SBDSQR/DBDSQR), which computes the singular value decomposition 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.**

```
*     F08KFF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          MMAX, NMAX, LDA, LDVT, LDU, LDC, LWORK
      PARAMETER        (MMAX=8,NMAX=8,LDA=MMAX,LDVT=NMAX,LDU=MMAX,
     +                 LDC=NMAX,LWORK=64*(MMAX+NMAX))
```

```
*     .. Local Scalars ..
      INTEGER          I, IC, IFAIL, INFO, J, M, N
*     .. Local Arrays ..
      real             A(LDA,NMAX), C(LDC,NMAX), D(NMAX), E(NMAX-1),
     +                 TAUP(NMAX), TAUQ(NMAX), U(LDU,NMAX),
     +                 VT(LDVT,NMAX), WORK(LWORK)
*     .. External Subroutines ..
      EXTERNAL         sbdsqr, sgebrd, sorgbr, F06QFF, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08KFF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      DO 20 IC = 1, 2
         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)
*
*           Reduce A to bidiagonal form
*
            CALL sgebrd(M,N,A,LDA,D,E,TAUQ,TAUP,WORK,LWORK,INFO)
*
            IF (M.GE.N) THEN
*
*              Copy A to VT and U
*
               CALL F06QFF('Upper',N,N,A,LDA,VT,LDVT)
*
               CALL F06QFF('Lower',M,N,A,LDA,U,LDU)
*
*              Form P**T explicitly, storing the result in VT
*
               CALL sorgbr('P',N,N,M,VT,LDVT,TAUP,WORK,LWORK,INFO)
*
*              Form Q explicitly, storing the result in U
*
               CALL sorgbr('Q',M,N,N,U,LDU,TAUQ,WORK,LWORK,INFO)
*
*              Compute the SVD of A
*
               CALL sbdsqr('Upper',N,N,M,0,D,E,VT,LDVT,U,LDU,C,LDC,WORK,
     +                     INFO)
*
*              Print singular values, left & right singular vectors
*
               WRITE (NOUT,*)
               WRITE (NOUT,*) 'Example 1: singular values'
               WRITE (NOUT,99999) (D(I),I=1,N)
               WRITE (NOUT,*)
               IFAIL = 0
*
               CALL X04CAF('General',' ',N,N,VT,LDVT,
     +                     'Example 1: right singular vectors, by row',
     +                     IFAIL)
*
               WRITE (NOUT,*)
*
               CALL X04CAF('General',' ',M,N,U,LDU,
     +                     'Example 1: left singular vectors, by column'
     +                     ,IFAIL)
*
            ELSE
*
*              Copy A to VT and U
*
               CALL F06QFF('Upper',M,N,A,LDA,VT,LDVT)
*
               CALL F06QFF('Lower',M,M,A,LDA,U,LDU)
```

```
*
*              Form P**T explicitly, storing the result in VT
*
               CALL sorgbr('P',M,N,M,VT,LDVT,TAUP,WORK,LWORK,INFO)
*
*              Form Q explicitly, storing the result in U
*
               CALL sorgbr('Q',M,M,N,U,LDU,TAUQ,WORK,LWORK,INFO)
*
*              Compute the SVD of A
*
               CALL sbdsqr('Lower',M,N,M,0,D,E,VT,LDVT,U,LDU,C,LDC,WORK,
     +                     INFO)
*
*              Print singular values, left & right singular vectors
*
               WRITE (NOUT,*)
               WRITE (NOUT,*) 'Example 2: singular values'
               WRITE (NOUT,99999) (D(I),I=1,M)
               WRITE (NOUT,*)
               IFAIL = 0
*
               CALL X04CAF('General',' ',M,N,VT,LDVT,
     +                     'Example 2: right singular vectors, by row',
     +                     IFAIL)
*
               WRITE (NOUT,*)
*
               CALL X04CAF('General',' ',M,M,U,LDU,
     +                     'Example 2: left singular vectors, by column'
     +                     ,IFAIL)
*
            END IF
         END IF
   20 CONTINUE
      STOP
*
99999 FORMAT (3X,(8F8.4))
      END
```

## 9.2. Program Data

```
F08KFF Example Program Data
  6  4                                          :Values of M and N, Example 1
 -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
  4  6                                          :Values of M and N, Example 2
 -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

```
F08KFF Example Program Results

Example 1: singular values
    3.9987  3.0005  1.9967  0.9999

Example 1: right singular vectors, by row
          1       2       3       4
1    0.8251 -0.2794  0.2048  0.4463
2   -0.4530 -0.2121 -0.2622  0.8252
3   -0.2829 -0.7961  0.4952 -0.2026
4    0.1841 -0.4931 -0.8026 -0.2807
```

```
Example 1: left singular vectors, by column
          1       2       3       4
1  -0.0203  0.2794  0.4690  0.7692
2  -0.7284 -0.3464 -0.0169 -0.0383
3   0.4393 -0.4955 -0.2868  0.0822
4  -0.4678  0.3258 -0.1536 -0.1636
5  -0.2200 -0.6428  0.1125  0.3572
6  -0.0935  0.1927 -0.8132  0.4957

Example 2: singular values
    7.9987  7.0059  5.9952  4.9989

Example 2: right singular vectors, by row
          1       2       3       4       5       6
1  -0.7933  0.3163 -0.3342 -0.1514  0.2142  0.3001
2   0.1002  0.6442  0.4371  0.4890  0.3771  0.0501
3   0.0111  0.1724 -0.6367  0.4354 -0.0430 -0.6111
4   0.2361  0.0216 -0.1025 -0.5286  0.7460 -0.3120

Example 2: left singular vectors, by column
          1       2       3       4
1   0.8884  0.1275  0.4331  0.0838
2   0.0733 -0.8264  0.1943 -0.5234
3  -0.0361  0.5435  0.0756 -0.8352
4   0.4518 -0.0733 -0.8769 -0.1466
```

# F08KGF (SORMBR/DORMBR) – 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

F08KGF (SORMBR/DORMBR) multiplies an arbitrary real matrix $C$ by one of the real orthogonal matrices $Q$ or $P$ which were determined by F08KEF (SGEBRD/DGEBRD) when reducing a real matrix to bidiagonal form.

## 2. Specification

```
      SUBROUTINE F08KGF (VECT, SIDE, TRANS, M, N, K, A, LDA, TAU, C, LDC, WORK,
     1                   LWORK, INFO)
      ENTRY       sormbr (VECT, 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      VECT, 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 F08KEF (SGEBRD/DGEBRD), which reduces a real rectangular matrix $A$ to bidiagonal form $B$ by an orthogonal transformation: $A = QBP^T$. F08KEF represents the matrices $Q$ and $P^T$ as products of elementary reflectors.

This routine may be used to form one of the matrix products

$QC$, $Q^TC$, $CQ$, $CQ^T$, $PC$, $P^TC$, $CP$ or $CP^T$,

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

In the description below, $r$ denotes the order of $Q$ or $P^T$: $r$ = M if SIDE = 'L' and $r$ = N if SIDE = 'R'.

1: VECT – CHARACTER*1. *Input*

*On entry*: indicates whether $Q$ or $Q^T$ or $P$ or $P^T$ is to be applied to $C$ as follows:

if VECT = 'Q', then $Q$ or $Q^T$ is applied to $C$;

if VECT = 'P', then $P$ or $P^T$ is applied to $C$.

*Constraint*: VECT = 'Q' or 'P'.

2: SIDE – CHARACTER*1. *Input*

*On entry*: indicates how $Q$ or $Q^T$ or $P$ or $P^T$ is to be applied to $C$ as follows:

if SIDE = 'L', then $Q$ or $Q^T$ or $P$ or $P^T$ is applied to $C$ from the left;

if SIDE = 'R', then $Q$ or $Q^T$ or $P$ or $P^T$ is applied to $C$ from the right.

*Constraint*: SIDE = 'L' or 'R'.

3: TRANS – CHARACTER*1. *Input*

*On entry*: indicates whether $Q$ or $P$ or $Q^T$ or $P^T$ is to be applied to $C$ as follows:

if TRANS = 'N', then $Q$ or $P$ is applied to $C$;

if TRANS = 'T', then $Q^T$ or $P^T$ is applied to $C$.

*Constraint*: TRANS = 'N' or 'T'.

4: M – INTEGER. *Input*

*On entry*: $m_C$, the number of rows of the matrix $C$.

*Constraint*: M $\geq$ 0.

5: N – INTEGER. *Input*

*On entry*: $n_C$, the number of columns of the matrix $C$.

*Constraint*: N $\geq$ 0.

6: K – INTEGER. *Input*

*On entry*: if VECT = 'Q', the number of columns in the original matrix $A$; if VECT = 'P', the number of rows in the original matrix $A$.

*Constraint*: K $\geq$ 0.

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

**Note**: the second dimension of the array A must be at least max(1,min($r$,K)) if VECT = 'Q' and at least max(1,$r$) if VECT = 'P'.

*On entry*: details of the vectors which define the elementary reflectors, as returned by F08KEF (SGEBRD/DGEBRD).

8: LDA – INTEGER. *Input*

*On entry*: the first dimension of the array A as declared in the (sub)program from which F08KGF (SORMBR/DORMBR) is called.

*Constraints*: LDA $\geq$ max(1,$r$) if VECT = 'Q',
LDA $\geq$ max(1,min($r$,K)) if VECT = 'P'.

9: TAU(*) – ***real*** array. *Input*

**Note**: the dimension of the array TAU must be at least max(1,min($r$,K)).

*On entry*: further details of the elementary reflectors, as returned by F08KEF (SGEBRD/DGEBRD) in its parameter TAUQ if VECT = 'Q', or in its parameter TAUP if VECT = 'P'.

10: C(LDC,*) – ***real*** array. *Input/Output*

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

*On entry*: the matrix $C$.

*On exit*: $C$ is overwritten by $QC$ or $Q^TC$ or $CQ^T$ or $CQ$ or $PC$ or $P^TC$ or $CP^T$ or $CP$ as specified by VECT, SIDE and TRANS.

11: LDC – INTEGER. *Input*

*On entry*: the first dimension of the array C as declared in the (sub)program from which F08KGF (SORMBR/DORMBR) is called.

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

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

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

13: LWORK – INTEGER. *Input*

*On entry*: the dimension of the array WORK as declared in the (sub)program from which F08KGF (SORMBR/DORMBR) 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'.

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 $i$th 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 $\varepsilon$ is the ***machine precision***.

## 8. Further Comments

The total number of floating-point operations is approximately

$2n_C k(2m_C - k)$ if SIDE = 'L' and $m_C \geq k$;

$2m_C k(2n_C - k)$ if SIDE = 'R' and $n_C \geq k$;

$2m_C^2 n_C$ if SIDE = 'L' and $m_C < k$;

$2m_C n_C^2$ if SIDE = 'R' and $n_C < k$;

here $k$ is the value of the parameter K.

The complex analogue of this routine is F08KUF (CUNMBR/ZUNMBR).

## 9. Example

For this routine two examples are presented. Both illustrate how the reduction to bidiagonal form of a matrix $A$ may be preceded by a $QR$ or $LQ$ factorization of $A$.

In the first example, $m > n$, and

$$A = \begin{pmatrix} -0.57 & -1.28 & -0.39 & 0.25 \\ -1.93 & 1.08 & -0.31 & -2.14 \\ 2.30 & 0.24 & 0.40 & -0.35 \\ -1.93 & 0.64 & -0.66 & 0.08 \\ 0.15 & 0.30 & 0.15 & -2.13 \\ -0.02 & 1.03 & -1.43 & 0.50 \end{pmatrix}.$$

The routine first performs a $QR$ factorization of $A$ as $A = Q_a R$ and then reduces the factor $R$ to bidiagonal form $B$: $R = Q_b B P^T$. Finally it forms $Q_a$ and calls F08KGF (SORMBR/DORMBR) to form $Q = Q_a Q_b$.

In the second example, $m < n$, and

$$A = \begin{pmatrix} -5.42 & 3.28 & -3.68 & 0.27 & 2.06 & 0.46 \\ -1.65 & -3.40 & -3.20 & -1.03 & -4.06 & -0.01 \\ -0.37 & 2.35 & 1.90 & 4.31 & -1.76 & 1.13 \\ -3.15 & -0.11 & 1.99 & -2.70 & 0.26 & 4.50 \end{pmatrix}.$$

The routine first performs a $LQ$ factorization of $A$ as $A = LP_a^T$ and then reduces the factor $L$ to bidiagonal form $B$: $L = QBP_b^T$. Finally it forms $P_b^T$ and calls F08KGF (SORMBR/DORMBR) to form $P^T = P_b^T P_a^T$.

## 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.**

```
*     F08KGF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          MMAX, NMAX, LDA, LDPT, LDU, LWORK
      PARAMETER        (MMAX=8,NMAX=8,LDA=MMAX,LDPT=NMAX,LDU=MMAX,
     +                 LWORK=64*(MMAX+NMAX))
      real             ZERO
      PARAMETER        (ZERO=0.0e0)
*     .. Local Scalars ..
      INTEGER          I, IC, IFAIL, INFO, J, M, N
*     .. Local Arrays ..
      real             A(LDA,NMAX), D(NMAX), E(NMAX-1), PT(LDPT, NMAX),
     +                 TAU(NMAX), TAUP(NMAX), TAUQ(NMAX), U(LDU,NMAX),
     +                 WORK(LWORK)
*     .. External Subroutines ..
      EXTERNAL         sgebrd, sgelqf, sgeqrf, sorglq, sorgqr, sormbr,
     +                 F06QFF, F06QHF, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08KGF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      DO 20 IC = 1, 2
         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)
*
            IF (M.GE.N) THEN
*
*              Compute the QR factorization of A
*
               CALL sgeqrf(M,N,A,LDA,TAU,WORK,LWORK,INFO)
*
*              Copy A to U
*
               CALL F06QFF('Lower',M,N,A,LDA,U,LDU)
*
*              Form Q explicitly, storing the result in U
*
               CALL sorgqr(M,M,N,U,LDU,TAU,WORK,LWORK,INFO)
*
*              Copy R to PT (used as workspace)
*
               CALL F06QFF('Upper',N,N,A,LDA,PT,LDPT)
*
*              Set the strictly lower triangular part of R to zero
*
               CALL F06QHF('Lower',N-1,N-1,ZERO,ZERO,PT(2,1),LDPT)
*
*              Bidiagonalize R
*
               CALL sgebrd(N,N,PT,LDPT,D,E,TAUQ,TAUP,WORK,LWORK,INFO)
*
*              Update Q, storing the result in U
*
```

```
                  CALL sormbr('Q','Right','No transpose',M,N,N,PT,LDPT,
     +                        TAUQ,U,LDU,WORK,LWORK,INFO)
*
*                 Print bidiagonal form and matrix Q
*
                  WRITE (NOUT,*)
                  WRITE (NOUT,*) 'Example 1: bidiagonal matrix B'
                  WRITE (NOUT,*) 'Diagonal'
                  WRITE (NOUT,99999) (D(I),I=1,N)
                  WRITE (NOUT,*) 'Super-diagonal'
                  WRITE (NOUT,99999) (E(I),I=1,N-1)
                  WRITE (NOUT,*)
                  IFAIL = 0
*
                  CALL X04CAF('General',' ',M,N,U,LDU,
     +                        'Example 1: matrix Q',IFAIL)
*
               ELSE
*
*                 Compute the LQ factorization of A
*
                  CALL sgelqf(M,N,A,LDA,TAU,WORK,LWORK,INFO)
*
*                 Copy A to PT
*
                  CALL F06QFF('Upper',M,N,A,LDA,PT,LDPT)
*
*                 Form Q explicitly, storing the result in PT
*
                  CALL sorglq(N,N,M,PT,LDPT,TAU,WORK,LWORK,INFO)
*
*                 Copy L to U (used as workspace)
*
                  CALL F06QFF('Lower',M,M,A,LDA,U,LDU)
*
*                 Set the strictly upper triangular part of L to zero
*
                  CALL F06QHF('Upper',M-1,M-1,ZERO,ZERO,U(1,2),LDU)
*
*                 Bidiagonalize L
*
                  CALL sgebrd(M,M,U,LDU,D,E,TAUQ,TAUP,WORK,LWORK,INFO)
*
*                 Update P**T, storing the result in PT
*
                  CALL sormbr('P','Left','Transpose',M,N,M,U,LDU,TAUP,PT,
     +                        LDPT,WORK,LWORK,INFO)
*
*                 Print bidiagonal form and matrix P**T
*
                  WRITE (NOUT,*)
                  WRITE (NOUT,*) 'Example 2: bidiagonal matrix B'
                  WRITE (NOUT,*) 'Diagonal'
                  WRITE (NOUT,99999) (D(I),I=1,M)
                  WRITE (NOUT,*) 'Super-diagonal'
                  WRITE (NOUT,99999) (E(I),I=1,M-1)
                  WRITE (NOUT,*)
                  IFAIL = 0
*
                  CALL X04CAF('General',' ',M,N,PT,LDPT,
     +                        'Example 2: matrix P**T',IFAIL)
*
               END IF
            END IF
   20    CONTINUE
         STOP
*
99999    FORMAT (3X,(8F8.4))
         END
```

## 9.2. Program Data

```
F08KGF Example Program Data
  6  4                                          :Values of M and N, Example 1
 -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
  4  6                                          :Values of M and N, Example 2
 -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

```
F08KGF Example Program Results

Example 1: bidiagonal matrix B
Diagonal
    3.6177 -2.4161  1.9213 -1.4265
Super-diagonal
    1.2587 -1.5262  1.1895

Example 1: matrix Q
          1       2       3       4
1  -0.1576 -0.2690  0.2612  0.8513
2  -0.5335  0.5311 -0.2922  0.0184
3   0.6358  0.3495 -0.0250 -0.0210
4  -0.5335  0.0035  0.1537 -0.2592
5   0.0415  0.5572 -0.2917  0.4523
6  -0.0055  0.4614  0.8585 -0.0532

Example 2: bidiagonal matrix B
Diagonal
   -7.7724  6.1573 -6.0576  5.7933
Super-diagonal
    1.1926  0.5734 -1.9143

Example 2: matrix P**T
          1       2       3       4       5       6
1  -0.7104  0.4299 -0.4824  0.0354  0.2700  0.0603
2   0.3583  0.1382 -0.4110  0.4044  0.0951 -0.7148
3  -0.0507  0.4244  0.3795  0.7402 -0.2773  0.2203
4   0.2442  0.4016  0.4158 -0.1354  0.7666 -0.0137
```

# F08KSF (CGEBRD/ZGEBRD) – 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

F08KSF (CGEBRD/ZGEBRD) reduces a complex $m$ by $n$ matrix to bidiagonal form.

## 2. Specification

```
SUBROUTINE F08KSF (M, N, A, LDA, D, E, TAUQ, TAUP, WORK, LWORK, INFO)
ENTRY      cgebrd (M, N, A, LDA, D, E, TAUQ, TAUP, WORK, LWORK, INFO)
INTEGER        M, N, LDA, LWORK, INFO
real           D(*), E(*)
complex        A(LDA,*), TAUQ(*), TAUP(*), WORK(LWORK)
```

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

## 3. Description

This routine reduces a complex $m$ by $n$ matrix $A$ to real bidiagonal form $B$ by a unitary transformation: $A = QBP^H$, where $Q$ and $P^H$ are unitary matrices of order $m$ and $n$ respectively.

If $m \geq n$, the reduction is given by:

$$A = Q\begin{pmatrix} B_1 \\ 0 \end{pmatrix}P^H = Q_1 B_1 P^H,$$

where $B_1$ is a real $n$ by $n$ upper bidiagonal matrix and $Q_1$ consists of the first $n$ columns of $Q$.

If $m < n$, the reduction is given by

$$A = Q(B_1 \ 0)P^H = QB_1P_1^H,$$

where $B_1$ is a real $m$ by $m$ lower bidiagonal matrix and $P_1^H$ consists of the first $m$ rows of $P^H$.

The unitary matrices $Q$ and $P$ are not formed explicitly but are represented as products of elementary reflectors (see the Chapter Introduction for details). Routines are provided to work with $Q$ and $P$ in this representation. (see Section 8).

## 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 diagonal and first super-diagonal are overwritten by the upper bidiagonal matrix $B$, elements below the diagonal are overwritten by details of the unitary

matrix $Q$ and elements above the first super-diagonal are overwritten by details of the unitary matrix $P$.

If $m < n$, the diagonal and first sub-diagonal are overwritten by the lower bidiagonal matrix $B$, elements below the first sub-diagonal are overwritten by details of the unitary matrix $Q$ and elements above the diagonal are overwritten by details of the unitary matrix $P$.

4: LDA – INTEGER. *Input*

*On entry*: the first dimension of the array A as declared in the (sub)program from which F08KSF (CGEBRD/ZGEBRD) is called.

*Constraint*: LDA ≥ max(1,M).

5: D(*) – ***real*** array. *Output*

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

*On exit*: the diagonal elements of the bidiagonal matrix $B$.

6: E(*) – ***real*** array. *Output*

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

*On exit*: the off-diagonal elements of the bidiagonal matrix $B$.

7: TAUQ(*) – ***complex*** array. *Output*

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

*On exit*: further details of the unitary matrix $Q$.

8: TAUP(*) – ***complex*** array. *Output*

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

*On exit*: further details of the unitary matrix $P$.

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

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

10: LWORK – INTEGER. *Input*

*On entry*: the dimension of the array WORK as declared in the (sub)program from which F08KSF (CGEBRD/ZGEBRD) is called.

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

*Constraint*: LWORK ≥ max(1,M,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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

## 7. Accuracy

The computed bidiagonal form $B$ satisfies $QBP^H = A + E$, where

$$\|E\|_2 \leq c(n)\varepsilon\|A\|_2,$$

$c(n)$ is a modestly increasing function of $n$, and $\varepsilon$ is the ***machine precision***.

The elements of $B$ themselves may be sensitive to small perturbations in $A$ or to rounding errors in the computation, but this does not affect the stability of the singular values and vectors.

## 8. Further Comments

The total number of real floating-point operations is approximately $16n^2(3m-n)/3$ if $m \geq n$ or $16m^2(3n-m)/3$ if $m < n$.

If $m \gg n$, it can be more efficient to first call F08ASF (CGEQRF/ZGEQRF) to perform a $QR$ factorization of $A$, and then to call this routine to reduce the factor $R$ to bidiagonal form. This requires approximately $8n^2(m+n)$ floating-point operations.

If $m \ll n$, it can be more efficient to first call F08AVF (CGELQF/ZGELQF) to perform an $LQ$ factorization of $A$, and then to call this routine to reduce the factor $L$ to bidiagonal form. This requires approximately $8m^2(m+n)$ operations.

To form the unitary matrices $P^H$ and/or $Q$, this routine may be followed by calls to F08KTF (CUNGBR/ZUNGBR):

to form the $m$ by $m$ unitary matrix $Q$

```
CALL CUNGBR ('Q',M,M,N,A,LDA,TAUQ,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 F08KSF;

to form the $n$ by $n$ unitary matrix $P^H$

```
CALL CUNGBR ('P',N,N,M,A,LDA,TAUP,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 F08KSF.

To apply $Q$ or $P$ to a complex rectangular matrix $C$, this routine may be followed by a call to F08KUF (CUNMBR/ZUNMBR).

The real analogue of this routine is F08KEF (SGEBRD/DGEBRD).

## 9. Example

To reduce the matrix $A$ to bidiagonal form, where

$$A = \begin{pmatrix} 0.96 - 0.81i & -0.03 + 0.96i & -0.91 + 2.06i & -0.05 + 0.41i \\ -0.98 + 1.98i & -1.20 + 0.19i & -0.66 + 0.42i & -0.81 + 0.56i \\ 0.62 - 0.46i & 1.01 + 0.02i & 0.63 - 0.17i & -1.11 + 0.60i \\ -0.37 + 0.38i & 0.19 - 0.54i & -0.98 - 0.36i & 0.22 - 0.20i \\ 0.83 + 0.51i & 0.20 + 0.01i & -0.17 - 0.46i & 1.47 + 1.59i \\ 1.08 - 0.28i & 0.20 - 0.12i & -0.07 + 1.23i & 0.26 + 0.26i \end{pmatrix}.$$

### 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.**

```
*     F08KSF 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+NMAX))
*     .. Local Scalars ..
      INTEGER          I, INFO, J, M, N
```

```
*     .. Local Arrays ..
      complex          A(LDA,NMAX), TAUP(NMAX), TAUQ(NMAX), WORK(LWORK)
      real             D(NMAX), E(NMAX-1)
*     .. External Subroutines ..
      EXTERNAL         cgebrd
*     .. Intrinsic Functions ..
      INTRINSIC        MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08KSF 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)
*
*        Reduce A to bidiagonal form
*
         CALL cgebrd(M,N,A,LDA,D,E,TAUQ,TAUP,WORK,LWORK,INFO)
*
*        Print bidiagonal form
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Diagonal'
         WRITE (NOUT,99999) (D(I),I=1,MIN(M,N))
         IF (M.GE.N) THEN
            WRITE (NOUT,*) 'Super-diagonal'
         ELSE
            WRITE (NOUT,*) 'Sub-diagonal'
         END IF
         WRITE (NOUT,99999) (E(I),I=1,MIN(M,N)-1)
      END IF
      STOP
*
99999 FORMAT (1X,8F9.4)
      END
```

## 9.2. Program Data

```
F08KSF 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

```
F08KSF Example Program Results

Diagonal
  -3.0870    2.0660    1.8731    2.0022
Super-diagonal
   2.1126    1.2628   -1.6126
```

# F08KTF (CUNGBR/ZUNGBR) – 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

F08KTF (CUNGBR/ZUNGBR) generates one of the complex unitary matrices $Q$ or $P^H$ which were determined by F08KSF (CGEBRD/ZGEBRD) when reducing a complex matrix to bidiagonal form.

## 2. Specification

```
SUBROUTINE F08KTF (VECT, M, N, K, A, LDA, TAU, WORK, LWORK, INFO)
ENTRY      cungbr (VECT, M, N, K, A, LDA, TAU, WORK, LWORK, INFO)
INTEGER          M, N, K, LDA, LWORK, INFO
complex          A(LDA,*), TAU(*), WORK(LWORK)
CHARACTER*1      VECT
```

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 F08KSF (CGEBRD/ZGEBRD), which reduces a complex rectangular matrix $A$ to real bidiagonal form $B$ by a unitary transformation: $A = QBP^H$. F08KSF represents the matrices $Q$ and $P^H$ as products of elementary reflectors.

This routine may be used to generate $Q$ or $P^H$ explicitly as square matrices, or in some cases just the leading columns of $Q$ or the leading rows of $P^H$.

The various possibilities are specified by the parameters VECT, M, N and K. The appropriate values to cover the most likely cases are as follows (assuming that $A$ was an $m$ by $n$ matrix):

1. To form the full $m$ by $m$ matrix $Q$:

   ```
   CALL CUNGBR ('Q',m,m,n,...)
   ```

   (note that the array A must have at least $m$ columns).

2. If $m > n$, to form the $n$ leading columns of $Q$:

   ```
   CALL CUNGBR ('Q',m,n,n,...)
   ```

3. To form the full $n$ by $n$ matrix $P^H$:

   ```
   CALL CUNGBR ('P',n,n,m,...)
   ```

   (note that the array A must have at least $n$ rows).

4. If $m < n$, to form the $m$ leading rows of $P^H$:

   ```
   CALL CUNGBR ('P',m,n,m,...)
   ```

## 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: VECT – CHARACTER*1. *Input*

*On entry*: indicates whether the unitary matrix $Q$ or $P^H$ is generated as follows:

if VECT = 'Q', then $Q$ is generated;

if VECT = 'P', then $P^H$ is generated.

*Constraint*: VECT = 'Q' or 'P'.

2: M – INTEGER. *Input*

*On entry*: the number of rows of the unitary matrix $Q$ or $P^H$ to be returned.

*Constraint*: M ≥ 0.

3: N – INTEGER. *Input*

*On entry*: the number of columns of the unitary matrix $Q$ or $P^H$ to be returned.

*Constraints*: N ≥ 0.

If VECT = 'Q', M ≥ N ≥ K if M > K, or
M = N if M ≤ K;
if VECT = 'P', N ≥ M ≥ K if N > K, or
N = M if N ≤ K.

4: K – INTEGER. *Input*

*On entry*: if VECT = 'Q', the number of columns in the original matrix $A$; if VECT = 'P', the number of rows in the original matrix $A$.

*Constraint*: K ≥ 0.

5: 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 F08KSF (CGEBRD/ZGEBRD).

*On exit*: the unitary matrix $Q$ or $P^H$, or the leading rows or columns thereof, as specified by VECT, M and N.

6: LDA – INTEGER. *Input*

*On entry*: the first dimension of the array A as declared in the (sub)program from which F08KTF (CUNGBR/ZUNGBR) is called.

*Constraint*: LDA ≥ max(1,M).

7: TAU(*) – ***complex*** array. *Input*

**Note**: the dimension of the array TAU must be at least max(1,min(M,K)) if VECT = 'Q', and at least max(1,min(N,K)) if VECT = 'P'.

*On entry*: further details of the elementary reflectors, as returned by F08KSF (CGEBRD/ZGEBRD) in its parameter TAUQ if VECT = 'Q', or in its parameter TAUP if VECT = 'P'.

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

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

9: LWORK – INTEGER. *Input*

*On entry*: the dimension of the array WORK as declared in the (sub)program from which F08KTF (CUNGBR/ZUNGBR) is called.

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

*Constraint*: LWORK ≥ max(1,min(M,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 $i$th 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 $\varepsilon$ is the ***machine precision***. A similar statement holds for the computed matrix $P^H$.

## 8. Further Comments

The total number of real floating-point operations for the cases listed in Section 3 are approximately as follows:

1. To form the whole of $Q$: $16n(3m^2-3mn+n^2)/3$ if $m > n$, $\frac{16m^3}{3}$ if $m \leq n$;
2. To form the $n$ leading columns of $Q$ when $m > n$: $\frac{4}{3}n^2(3m-n)$;
3. To form the whole of $P^H$: $\frac{16n^3}{3}$ if $m \geq n$, $16m(3n^2-3mn+m^2)/3$ if $m < n$;
4. To form the $m$ leading rows of $P^H$ when $m < n$: $\frac{4}{3}m^2(3n-m)$.

The real analogue of this routine is F08KFF (SORGBR/DORGBR).

## 9. Example

For this routine two examples are presented, both of which involve computing the singular value decomposition of a matrix $A$, where

$$A = \begin{pmatrix} 0.96 - 0.81i & -0.03 + 0.96i & -0.91 + 2.06i & -0.05 + 0.41i \\ -0.98 + 1.98i & -1.20 + 0.19i & -0.66 + 0.42i & -0.81 + 0.56i \\ 0.62 - 0.46i & 1.01 + 0.02i & 0.63 - 0.17i & -1.11 + 0.60i \\ -0.37 + 0.38i & 0.19 - 0.54i & -0.98 - 0.36i & 0.22 - 0.20i \\ 0.83 + 0.51i & 0.20 + 0.01i & -0.17 - 0.46i & 1.47 + 1.59i \\ 1.08 - 0.28i & 0.20 - 0.12i & -0.07 + 1.23i & 0.26 + 0.26i \end{pmatrix}$$

in the first example and

$$A = \begin{pmatrix} 0.28 - 0.36i & 0.50 - 0.86i & -0.77 - 0.48i & 1.58 + 0.66i \\ -0.50 - 1.10i & -1.21 + 0.76i & -0.32 - 0.24i & -0.27 - 1.15i \\ 0.36 - 0.51i & -0.07 + 1.33i & -0.75 + 0.47i & -0.08 + 1.01i \end{pmatrix}$$

in the second. $A$ must first be reduced to tridiagonal form by F08KSF (CGEBRD/ZGEBRD). The program then calls F08KTF (CUNGBR/ZUNGBR) twice to form $Q$ and $P^H$, and passes these matrices to F08MSF (CBDSQR/ZBDSQR), which computes the singular value decomposition 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.

```
*     F08KTF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          MMAX, NMAX, LDA, LDVT, LDU, LDC, LWORK
      PARAMETER        (MMAX=8,NMAX=8,LDA=MMAX,LDVT=NMAX,LDU=MMAX,
     +                 LDC=NMAX,LWORK=64*(MMAX+NMAX))
*     .. Local Scalars ..
      INTEGER          I, IC, IFAIL, INFO, J, M, N
*     .. Local Arrays ..
      complex          A(LDA,NMAX), C(LDC,NMAX), TAUP(NMAX), TAUQ(NMAX),
     +                 U(LDU,NMAX), VT(LDVT,NMAX), WORK(LWORK)
      real             D(NMAX), E(NMAX-1), RWORK(4*NMAX-4)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         F06TFF, X04DBF, cbdsqr, cgebrd, cungbr
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08KTF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      DO 20 IC = 1, 2
         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)
*
*           Reduce A to bidiagonal form
*
            CALL cgebrd(M,N,A,LDA,D,E,TAUQ,TAUP,WORK,LWORK,INFO)
*
            IF (M.GE.N) THEN
*
*              Copy A to VT and U
*
               CALL F06TFF('Upper',N,N,A,LDA,VT,LDVT)
*
               CALL F06TFF('Lower',M,N,A,LDA,U,LDU)
*
*              Form P**H explicitly, storing the result in VT
*
               CALL cungbr('P',N,N,M,VT,LDVT,TAUP,WORK,LWORK,INFO)
*
*              Form Q explicitly, storing the result in U
*
               CALL cungbr('Q',M,N,N,U,LDU,TAUQ,WORK,LWORK,INFO)
*
*              Compute the SVD of A
*
               CALL cbdsqr('Upper',N,N,M,0,D,E,VT,LDVT,U,LDU,C,LDC,
     +                     RWORK,INFO)
*
*              Print singular values, left & right singular vectors
*
               WRITE (NOUT,*)
               WRITE (NOUT,*) 'Example 1: singular values'
               WRITE (NOUT,99999) (D(I),I=1,N)
               WRITE (NOUT,*)
               IFAIL = 0
*
```

```
                  CALL X04DBF('General',' ',N,N,VT,LDVT,'Bracketed','F7.4',
     +                        'Example 1: right singular vectors, by row',
     +                        'Integer',RLABS,'Integer',CLABS,80,0,IFAIL)
*
                  WRITE (NOUT,*)
*
                  CALL X04DBF('General',' ',M,N,U,LDU,'Bracketed','F7.4',
     +                        'Example 1: left singular vectors, by column'
     +                        ,'Integer',RLABS,'Integer',CLABS,80,0,IFAIL)
*
               ELSE
*
*                 Copy A to VT and U
*
                  CALL F06TFF('Upper',M,N,A,LDA,VT,LDVT)
*
                  CALL F06TFF('Lower',M,M,A,LDA,U,LDU)
*
*                 Form P**H explicitly, storing the result in VT
*
                  CALL cungbr('P',M,N,M,VT,LDVT,TAUP,WORK,LWORK,INFO)
*
*                 Form Q explicitly, storing the result in U
*
                  CALL cungbr('Q',M,M,N,U,LDU,TAUQ,WORK,LWORK,INFO)
*
*                 Compute the SVD of A
*
                  CALL cbdsqr('Lower',M,N,M,0,D,E,VT,LDVT,U,LDU,C,LDC,
     +                        RWORK,INFO)
*
*                 Print singular values, left & right singular vectors
*
                  WRITE (NOUT,*)
                  WRITE (NOUT,*) 'Example 2: singular values'
                  WRITE (NOUT,99999) (D(I),I=1,M)
                  WRITE (NOUT,*)
                  IFAIL = 0
*
                  CALL X04DBF('General',' ',M,N,VT,LDVT,'Bracketed','F7.4',
     +                        'Example 2: right singular vectors, by row',
     +                        'Integer',RLABS,'Integer',CLABS,80,0,IFAIL)
*
                  WRITE (NOUT,*)
*
                  CALL X04DBF('General',' ',M,M,U,LDU,'Bracketed','F7.4',
     +                        'Example 2: left singular vectors, by column'
     +                        ,'Integer',RLABS,'Integer',CLABS,80,0,IFAIL)
*
               END IF
            END IF
   20    CONTINUE
         STOP
*
99999 FORMAT (8X,4(F7.4,11X,:))
      END
```

## 9.2. Program Data

```
F08KTF Example Program Data
  6  4                                                :Values of M and N, Example 1
 ( 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
  3  4                                                :Values of M and N, Example 2
 ( 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

```
F08KTF Example Program Results

Example 1: singular values
        3.9994            3.0003            1.9944            0.9995

Example 1: right singular vectors, by row
                    1                 2                 3                 4
1  (-0.6971, 0.0000) (-0.0867,-0.3548) ( 0.0560,-0.5400) (-0.1878,-0.2253)
2  ( 0.2403, 0.0000) ( 0.0725,-0.2336) (-0.2477,-0.5291) ( 0.7026, 0.2177)
3  ( 0.5123, 0.0000) ( 0.3030, 0.1735) (-0.0678,-0.5162) (-0.4418,-0.3864)
4  ( 0.4403, 0.0000) (-0.5294,-0.6361) ( 0.3027, 0.0346) (-0.1667,-0.0258)

Example 1: left singular vectors, by column
                    1                 2                 3                 4
1  (-0.5634, 0.0016) (-0.2687,-0.2749) (-0.2451,-0.4657) (-0.3787,-0.2987)
2  ( 0.1205,-0.6108) (-0.2909, 0.1085) (-0.4329, 0.1758) ( 0.0182, 0.0437)
3  (-0.0816, 0.1613) (-0.1660, 0.3885) ( 0.4667,-0.3821) ( 0.0800, 0.2276)
4  ( 0.1441,-0.1532) ( 0.1984,-0.1737) ( 0.0034,-0.1555) (-0.2608, 0.5382)
5  (-0.2487,-0.0926) ( 0.6253, 0.3304) (-0.2643, 0.0194) (-0.1002,-0.0140)
6  (-0.3758, 0.0793) (-0.0307,-0.0816) (-0.1266,-0.1747) ( 0.4175, 0.4058)

Example 2: singular values
        3.0004            1.9967            0.9973

Example 2: right singular vectors, by row
                    1                 2                 3                 4
1  ( 0.2454,-0.0001) ( 0.2942,-0.5843) ( 0.0162,-0.0810) ( 0.6794, 0.2083)
2  (-0.1692, 0.5194) ( 0.1915,-0.4374) ( 0.5205,-0.0244) (-0.3149,-0.3208)
3  (-0.5553, 0.1403) ( 0.1438,-0.1507) (-0.5684,-0.5505) (-0.0318,-0.0378)

Example 2: left singular vectors, by column
                    1                 2                 3
1  ( 0.6518, 0.0000) (-0.4312, 0.0000) ( 0.6239, 0.0000)
2  (-0.4437,-0.5027) (-0.3794, 0.1026) ( 0.2014, 0.5961)
3  (-0.2012, 0.2916) (-0.8122, 0.0030) (-0.3511,-0.3026)
```

# F08KUF (CUNMBR/ZUNMBR) – 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

F08KUF (CUNMBR/ZUNMBR) multiplies an arbitrary complex matrix $C$ by one of the complex unitary matrices $Q$ or $P$ which were determined by F08KSF (CGEBRD/ZGEBRD) when reducing a complex matrix to bidiagonal form.

## 2. Specification

```
      SUBROUTINE F08KUF (VECT, SIDE, TRANS, M, N, K, A, LDA, TAU, C, LDC, WORK,
     1                   LWORK, INFO)
      ENTRY       cunmbr (VECT, 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      VECT, 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 F08KSF (CGEBRD/ZGEBRD), which reduces a complex rectangular matrix $A$ to real bidiagonal form $B$ by a unitary transformation: $A = QBP^H$. F08KSF represents the matrices $Q$ and $P^H$ as products of elementary reflectors.

This routine may be used to form one of the matrix products

$QC$, $Q^HC$, $CQ$, $CQ^H$, $PC$, $P^HC$, $CP$ or $CP^H$,

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

In the description below, $r$ denotes the order of $Q$ or $P^H$: $r$ = M if SIDE = 'L' and $r$ = N if SIDE = 'R'.

1: VECT – CHARACTER*1. *Input*

*On entry*: indicates whether $Q$ or $Q^H$ or $P$ or $P^H$ is to be applied to $C$ as follows:

if VECT = 'Q', then $Q$ or $Q^H$ is applied to $C$;

if VECT = 'P', then $P$ or $P^H$ is applied to $C$.

*Constraint*: VECT = 'Q' or 'P'.

2: SIDE – CHARACTER*1. *Input*

*On entry*: indicates how $Q$ or $Q^H$ or $P$ or $P^H$ is to be applied to $C$ as follows:

if SIDE = 'L', then $Q$ or $Q^H$ or $P$ or $P^H$ is applied to $C$ from the left;

if SIDE = 'R', then $Q$ or $Q^H$ or $P$ or $P^H$ is applied to $C$ from the right.

*Constraint*: SIDE = 'L' or 'R'.

3: TRANS – CHARACTER*1. *Input*

*On entry*: indicates whether $Q$ or $P$ or $Q^H$ or $P^H$ is to be applied to $C$ as follows:

if TRANS = 'N', then $Q$ or $P$ is applied to $C$;

if TRANS = 'C', then $Q^H$ or $P^H$ is applied to $C$.

*Constraint*: TRANS = 'N' or 'C'.

4: M – INTEGER. *Input*

*On entry*: $m_C$, the number of rows of the matrix $C$.

*Constraint*: M $\geq$ 0.

5: N – INTEGER. *Input*

*On entry*: $n_C$, the number of columns of the matrix $C$.

*Constraint*: N $\geq$ 0.

6: K – INTEGER. *Input*

*On entry*: if VECT = 'Q', the number of columns in the original matrix $A$; if VECT = 'P', the number of rows in the original matrix $A$.

*Constraint*: K $\geq$ 0.

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

**Note**: the second dimension of the array A must be at least max(1,min($r$,K)) if VECT = 'Q' and at least max(1,$r$) if VECT = 'P'.

*On entry*: details of the vectors which define the elementary reflectors, as returned by F08KSF (CGEBRD/ZGEBRD).

8: LDA – INTEGER. *Input*

*On entry*: the first dimension of the array A as declared in the (sub)program from which F08KUF (CUNMBR/ZUNMBR) is called.

*Constraints*: LDA $\geq$ max(1,$r$) if VECT = 'Q',
LDA $\geq$ max(1,min($r$,K)) if VECT = 'P'.

9: TAU(*) – ***complex*** array. *Input*

**Note**: the dimension of the array TAU must be at least max(1,min($r$,K)).

*On entry*: further details of the elementary reflectors, as returned by F08KSF (CGEBRD/ZGEBRD) in its parameter TAUQ if VECT = 'Q', or in its parameter TAUP if VECT = 'P'.

10: C(LDC,*) – ***complex*** array. *Input/Output*

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

*On entry*: the matrix $C$.

*On exit*: $C$ is overwritten by $QC$ or $Q^HC$ or $CQ^H$ or $CQ$ or $PC$ or $P^HC$ or $CP^H$ or $CP$ as specified by VECT, SIDE and TRANS.

11: LDC – INTEGER. *Input*

*On entry*: the first dimension of the array C as declared in the (sub)program from which F08KUF (CUNMBR/ZUNMBR) is called.

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

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

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

13: LWORK – INTEGER. *Input*

*On entry*: the dimension of the array WORK as declared in the (sub)program from which F08KUF (CUNMBR/ZUNMBR) 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'.

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 $i$th 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 $\varepsilon$ is the ***machine precision***.

## 8. Further Comments

The total number of real floating-point operations is approximately

$8n_C k(2m_C - k)$ if SIDE = 'L' and $m_C \geq k$;

$8m_C k(2n_C - k)$ if SIDE = 'R' and $n_C \geq k$;

$8m_C^2 n_C$ if SIDE = 'L' and $m_C < k$;

$8m_C n_C^2$ if SIDE = 'R' and $n_C < k$;

here $k$ is the value of the parameter K.

The real analogue of this routine is F08KGF (SORMBR/DORMBR).

## 9. Example

For this routine two examples are presented. Both illustrate how the reduction to bidiagonal form of a matrix $A$ may be preceded by a $QR$ or $LQ$ factorization of $A$.

In the first example, $m > n$, and

$$A = \begin{pmatrix} 0.96 - 0.81i & -0.03 + 0.96i & -0.91 + 2.06i & -0.05 + 0.41i \\ -0.98 + 1.98i & -1.20 + 0.19i & -0.66 + 0.42i & -0.81 + 0.56i \\ 0.62 - 0.46i & 1.01 + 0.02i & 0.63 - 0.17i & -1.11 + 0.60i \\ -0.37 + 0.38i & 0.19 - 0.54i & -0.98 - 0.36i & 0.22 - 0.20i \\ 0.83 + 0.51i & 0.20 + 0.01i & -0.17 - 0.46i & 1.47 + 1.59i \\ 1.08 - 0.28i & 0.20 - 0.12i & -0.07 + 1.23i & 0.26 + 0.26i \end{pmatrix}.$$

The routine first performs a $QR$ factorization of $A$ as $A = Q_a R$ and then reduces the factor $R$ to bidiagonal form $B$: $R = Q_b B P^H$. Finally it forms $Q_a$ and calls F08KUF (CUNMBR/ZUNMBR) to form $Q = Q_a Q_b$.

In the second example, $m < n$, and

$$A = \begin{pmatrix} 0.28 - 0.36i & 0.50 - 0.86i & -0.77 - 0.48i & 1.58 + 0.66i \\ -0.50 - 1.10i & -1.21 + 0.76i & -0.32 - 0.24i & -0.27 - 1.15i \\ 0.36 - 0.51i & -0.07 + 1.33i & -0.75 + 0.47i & -0.08 + 1.01i \end{pmatrix}.$$

The routine first performs a $LQ$ factorization of $A$ as $A = LP_a^H$ and then reduces the factor $L$ to

bidiagonal form $B$: $L = QBP_b^H$. Finally it forms $P_b^H$ and calls F08KUF (CUNMBR/ZUNMBR) to form $P^H = P_b^H P_a^H$.

## 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.

```
*     F08KUF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          MMAX, NMAX, LDA, LDPH, LDU, LWORK
      PARAMETER        (MMAX=8,NMAX=8,LDA=MMAX,LDPH=NMAX,LDU=MMAX,
     +                 LWORK=64*(MMAX+NMAX))
      complex          ZERO
      PARAMETER        (ZERO=(0.0e0,0.0e0))
*     .. Local Scalars ..
      INTEGER          I, IC, IFAIL, INFO, J, M, N
*     .. Local Arrays ..
      complex          A(LDA,NMAX), PH(LDPH,NMAX), TAU(NMAX),
     +                 TAUP(NMAX), TAUQ(NMAX), U(LDU,NMAX), WORK(LWORK)
      real             D(NMAX), E(NMAX-1)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         F06TFF, F06THF, X04DBF, cgebrd, cgelqf, cgeqrf,
     +                 cunglq, cungqr, cunmbr
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08KUF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      DO 20 IC = 1, 2
         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)
*
            IF (M.GE.N) THEN
*
*              Compute the QR factorization of A
*
               CALL cgeqrf(M,N,A,LDA,TAU,WORK,LWORK,INFO)
*
*              Copy A to U
*
               CALL F06TFF('Lower',M,N,A,LDA,U,LDU)
*
*              Form Q explicitly, storing the result in U
*
               CALL cungqr(M,M,N,U,LDU,TAU,WORK,LWORK,INFO)
*
*              Copy R to PH (used as workspace)
*
               CALL F06TFF('Upper',N,N,A,LDA,PH,LDPH)
*
*              Set the strictly lower triangular part of R to zero
*
               CALL F06THF('Lower',N-1,N-1,ZERO,ZERO,PH(2,1),LDPH)
*
*              Bidiagonalize R
*
               CALL cgebrd(N,N,PH,LDPH,D,E,TAUQ,TAUP,WORK,LWORK,INFO)
*
*              Update Q, storing the result in U
*
               CALL cunmbr('Q','Right','No transpose',M,N,N,PH,LDPH,
     +                     TAUQ,U,LDU,WORK,LWORK,INFO)
```

```
*
*              Print bidiagonal form and matrix Q
*
               WRITE (NOUT,*)
               WRITE (NOUT,*) 'Example 1: bidiagonal matrix B'
               WRITE (NOUT,*) 'Diagonal'
               WRITE (NOUT,99999) (D(I),I=1,N)
               WRITE (NOUT,*) 'Super-diagonal'
               WRITE (NOUT,99999) (E(I),I=1,N-1)
               WRITE (NOUT,*)
               IFAIL = 0
*
               CALL X04DBF('General',' ',M,N,U,LDU,'Bracketed','F7.4',
     +                     'Example 1: matrix Q','Integer',RLABS,
     +                     'Integer',CLABS,80,0,IFAIL)
*
            ELSE
*
*              Compute the LQ factorization of A
*
               CALL cgelqf(M,N,A,LDA,TAU,WORK,LWORK,INFO)
*
*              Copy A to PH
*
               CALL F06TFF('Upper',M,N,A,LDA,PH,LDPH)
*
*              Form Q explicitly, storing the result in PH
*
               CALL cunglq(N,N,M,PH,LDPH,TAU,WORK,LWORK,INFO)
*
*              Copy L to U (used as workspace)
*
               CALL F06TFF('Lower',M,M,A,LDA,U,LDU)
*
*              Set the strictly upper triangular part of L to zero
*
               CALL F06THF('Upper',M-1,M-1,ZERO,ZERO,U(1,2),LDU)
*
*              Bidiagonalize L
*
               CALL cgebrd(M,M,U,LDU,D,E,TAUQ,TAUP,WORK,LWORK,INFO)
*
*              Update P**H, storing the result in PH
*
               CALL cunmbr('P','Left','Conjugate transpose',M,N,M,U,LDU,
     +                     TAUP,PH,LDPH,WORK,LWORK,INFO)
*
*              Print bidiagonal form and matrix P**H
*
               WRITE (NOUT,*)
               WRITE (NOUT,*) 'Example 2: bidiagonal matrix B'
               WRITE (NOUT,*) 'Diagonal'
               WRITE (NOUT,99999) (D(I),I=1,M)
               WRITE (NOUT,*) 'Super-diagonal'
               WRITE (NOUT,99999) (E(I),I=1,M-1)
               WRITE (NOUT,*)
               IFAIL = 0
*
               CALL X04DBF('General',' ',M,N,PH,LDPH,'Bracketed','F7.4',
     +                     'Example 2: matrix P**H','Integer',RLABS,
     +                     'Integer',CLABS,80,0,IFAIL)
*
            END IF
         END IF
   20 CONTINUE
      STOP
*
99999 FORMAT (3X,(8F8.4))
      END
```

## 9.2. Program Data

```
F08KUF Example Program Data
  6  4                                                :Values of M and N, Example 1
 ( 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
  3  4                                                :Values of M and N, Example 2
 ( 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

```
F08KUF Example Program Results

Example 1: bidiagonal matrix B
Diagonal
   -3.0870 -2.0660 -1.8731 -2.0022
Super-diagonal
    2.1126 -1.2628  1.6126

Example 1: matrix Q
                      1                      2                      3                      4
1  (-0.3110, 0.2624) ( 0.6521, 0.5532) ( 0.0427, 0.0361) (-0.2634,-0.0741)
2  ( 0.3175,-0.6414) ( 0.3488, 0.0721) ( 0.2287, 0.0069) ( 0.1101,-0.0326)
3  (-0.2008, 0.1490) (-0.3103, 0.0230) ( 0.1855,-0.1817) (-0.2956, 0.5648)
4  ( 0.1199,-0.1231) (-0.0046,-0.0005) (-0.3305, 0.4821) (-0.0675, 0.3464)
5  (-0.2689,-0.1652) ( 0.1794,-0.0586) (-0.5235,-0.2580) ( 0.3927, 0.1450)
6  (-0.3499, 0.0907) ( 0.0829,-0.0506) ( 0.3202, 0.3038) ( 0.3174, 0.3241)

Example 2: bidiagonal matrix B
Diagonal
    2.7615  1.6298 -1.3275
Super-diagonal
   -0.9500 -1.0183

Example 2: matrix P**H
                      1                      2                      3                      4
1  (-0.1258, 0.1618) (-0.2247, 0.3864) ( 0.3460, 0.2157) (-0.7099,-0.2966)
2  ( 0.4148, 0.1795) ( 0.1368,-0.3976) ( 0.6885, 0.3386) ( 0.1667,-0.0494)
3  ( 0.4575,-0.4807) (-0.2733, 0.4981) (-0.0230, 0.3861) ( 0.1730, 0.2395)
```

# F08MEF (SBDSQR/DBDSQR) – 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.

**Warning:** the specification for the parameter WORK changed at Mark 17: the length of WORK may need to be increased if NCVT = NRU = NCC = 0.

## 1 Purpose

F08MEF (SBDSQR/DBDSQR) computes the singular value decomposition of a real upper or lower bidiagonal matrix, or of a real general matrix which has been reduced to bidiagonal form.

## 2 Specification

```
      SUBROUTINE F08MEF(UPLO, N, NCVT, NRU, NCC, D, E, VT, LDVT, U, LDU,
     1                  C, LDC, WORK, INFO)
      ENTRY       sbdsqr(UPLO, N, NCVT, NRU, NCC, D, E, VT, LDVT, U, LDU,
     1                  C, LDC, WORK, INFO)
      INTEGER           N, NCVT, NRU, NCC, LDVT, LDU, LDC, INFO
      real              D(*), E(*), VT(LDVT,*), U(LDU,*), C(LDC,*),
     1                  WORK(*)
      CHARACTER*1       UPLO
```

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

## 3 Description

This routine computes the singular values, and optionally, the left or right singular vectors of a real upper or lower bidiagonal matrix $B$. In other words, it can compute the singular value decomposition (SVD) of $B$ as

$$B = U\Sigma V^T.$$

Here $\Sigma$ is a diagonal matrix with real diagonal elements $\sigma_i$ (the singular values of $B$), such that

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0;$$

$U$ is an orthogonal matrix whose columns are the left singular vectors $u_i$; $V$ is an orthogonal matrix whose rows are the right singular vectors $v_i$. Thus

$$Bu_i = \sigma_i v_i \text{ and } B^T v_i = \sigma_i u_i \text{ for } i = 1, 2, \ldots, n.$$

To compute $U$ and/or $V^T$, the arrays U and/or VT must be initialized to the unit matrix before F08MEF is called.

The routine may also be used to compute the SVD of a real general matrix $A$ which has been reduced to bidiagonal form by an orthogonal transformation: $A = QBP^T$. If $A$ is $m$ by $n$ with $m \geq n$, then $Q$ is $m$ by $n$ and $P^T$ is $n$ by $n$; if $A$ is $n$ by $p$ with $n < p$, then $Q$ is $n$ by $n$ and $P^T$ is $n$ by $p$. In this case, the matrices $Q$ and/or $P^T$ must be formed explicitly by F08KFF (SORGBR/DORGBR) and passed to F08MEF in the arrays U and/or VT respectively.

F08MEF also has the capability of forming $U^T C$, where $C$ is an arbitrary real matrix; this is needed when using the SVD to solve linear least-squares problems.

F08MEF uses two different algorithms. If any singular vectors are required (that is, if NCVT > 0 or NRU > 0 or NCC > 0), the bidiagonal $QR$ algorithm is used, switching between zero-shift and implicitly shifted forms to preserve the accuracy of small singular values, and switching between $QR$ and $QL$ variants in order to handle graded matrices effectively (see Demmel and Kahan [1]). If only singular values are required (that is, if NCVT = NRU = NCC = 0), they are computed by the differential qd algorithm (see Fernando and Parlett [2]), which is faster and can achieve even greater accuracy.

The singular vectors are normalized so that $\| u_i \| = \| v_i \| = 1$, but are determined only to within a factor $\pm 1$.

# 4 References

[1] Demmel J W and Kahan W (1990) Accurate singular values of bidiagonal matrices *SIAM J. Sci. Statist. Comput.* **11** 873–912

[2] Fernando K V and Parlett B N (1994) Accurate singular values and differential qd algorithms *Numer. Math.* **67** 191–229

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

# 5 Parameters

**1:** UPLO — CHARACTER*1 *Input*

*On entry:* indicates whether $B$ is an upper or lower bidiagonal matrix as follows:

if UPLO = 'U', then $B$ is an upper bidiagonal matrix;

if UPLO = 'L', then $B$ is a lower bidiagonal matrix.

*Constraint:* UPLO = 'U' or 'L'.

**2:** N — INTEGER *Input*

*On entry:* $n$, the order of the matrix $B$.

*Constraint:* N $\geq 0$.

**3:** NCVT — INTEGER *Input*

*On entry:* *ncvt*, the number of columns of the matrix $V^T$ of right singular vectors. Set NCVT = 0 if no right singular vectors are required.

*Constraint:* NCVT $\geq 0$.

**4:** NRU — INTEGER *Input*

*On entry:* *nru*, the number of rows of the matrix $U$ of left singular vectors. Set NRU = 0 if no left singular vectors are required.

*Constraint:* NRU $\geq 0$.

**5:** NCC — INTEGER *Input*

*On entry:* *ncc*, the number of columns of the matrix $C$. Set NCC = 0 if no matrix $C$ is supplied.

*Constraint:* NCC $\geq 0$.

**6:** D(*) — ***real*** array *Input/Output*

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

*On entry:* the diagonal elements of the bidiagonal matrix $B$.

*On exit:* the singular values in decreasing order of magnitude, unless INFO > 0 (in which case see Section 6).

**7:** E(*) — ***real*** array *Input/Output*

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

*On entry:* the off-diagonal elements of the bidiagonal matrix $B$.

*On exit:* the array is overwritten, but if INFO > 0 see Section 6.

**8:** VT(LDVT,∗) — ***real*** array *Input/Output*

**Note:** the second dimension of the array VT must be at least max(1, NCVT).

*On entry:* if NCVT > 0, VT must contain an $n$ by $ncvt$ matrix. If the right singular vectors of $B$ are required, $ncvt = n$ and VT must contain the unit matrix; if the right singular vectors of $A$ are required, VT must contain the orthogonal matrix $P^T$ returned by F08KFF (SORGBR/DORGBR) with VECT = 'P'.

*On exit:* the $n$ by $ncvt$ matrix $V^T$ or $V^T P^T$ of right singular vectors, stored by rows.

VT is not referenced if NCVT = 0.

**9:** LDVT — INTEGER *Input*

*On entry:* the first dimension of the array VT as declared in the (sub)program from which F08MEF (SBDSQR/DBDSQR) is called.

*Constraints:*

LDVT ≥ max(1, N) if NCVT > 0,
LDVT ≥ 1 otherwise.

**10:** U(LDU,∗) — ***real*** array *Input/Output*

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

*On entry:* if NRU > 0, U must contain an $nru$ by $n$ matrix. If the left singular vectors of $B$ are required, $nru = n$ and U must contain the unit matrix; if the left singular vectors of $A$ are required, U must contain the orthogonal matrix $Q$ returned by F08KFF (SORGBR/DORGBR) with VECT = 'Q'.

*On exit:* the $nru$ by $n$ matrix $U$ or $QU$ of left singular vectors, stored by columns.

U is not referenced if NRU = 0.

**11:** LDU — INTEGER *Input*

*On entry:* the first dimension of the array U as declared in the (sub)program from which F08MEF (SBDSQR/DBDSQR) is called.

*Constraint:* LDU ≥ max(1, NRU).

**12:** C(LDC,∗) — ***real*** array *Input/Output*

**Note:** the second dimension of the array C must be at least max(1, NCC).

*On entry:* the $n$ by $ncc$ matrix $C$ if NCC > 0.

*On exit:* $C$ is overwritten by the matrix $U^T C$.

C is not referenced if NCC = 0.

**13:** LDC — INTEGER *Input*

*On entry:* the first dimension of the array C as declared in the (sub)program from which F08MEF (SBDSQR/DBDSQR) is called.

*Constraints:*

LDC ≥ max(1, N) if NCC > 0,
LDC ≥ 1 otherwise.

**14:** WORK(∗) — ***real*** array *Workspace*

**Note:** the dimension of the array WORK must be at least max(1, 2∗N) if NCVT = NRU = NCC = 0, and at least max(1, 4 ∗ (N − 1)) otherwise.

**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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

The algorithm failed to converge and INFO specifies how many off-diagonals did not converge. In this case, D and E contain on exit the diagonal and off-diagonal elements, respectively, of a bidiagonal matrix orthogonally equivalent to $B$.

# 7 Accuracy

Each singular value and singular vector is computed to high relative accuracy. However, the reduction to bidiagonal form (prior to calling the routine) may exclude the possibility of obtaining high relative accuracy in the small singular values of the original matrix if its singular values vary widely in magnitude.

If $\sigma_i$ is an exact singular value of $B$ and $\tilde{\sigma}_i$ is the corresponding computed value, then

$$|\tilde{\sigma}_i - \sigma_i| \leq p(m,n)\epsilon\sigma_i$$

where $p(m,n)$ is a modestly increasing function of $m$ and $n$, and $\epsilon$ is the ***machine precision***. If only singular values are computed, they are computed more accurately (that is, the function $p(m,n)$ is smaller), than when some singular vectors are also computed.

If $u_i$ is the corresponding exact left singular vector of $B$, and $\tilde{u}_i$ is the corresponding computed left singular vector, then the angle $\theta(\tilde{u}_i, u_i)$ between them is bounded as follows:

$$\theta(\tilde{u}_i, u_i) \leq \frac{p(m,n)\epsilon}{relgap_i}$$

where $relgap_i$ is the relative gap between $\sigma_i$ and the other singular values, defined by

$$relgap_i = \min_{i \neq j} \frac{|\sigma_i - \sigma_j|}{(\sigma_i + \sigma_j)}.$$

A similar error bound holds for the right singular vectors.

# 8 Further Comments

The total number of floating-point operations is roughly proportional to $n^2$ if only the singular values are computed. About $6n^2 \times nru$ additional operations are required to compute the left singular vectors and about $6n^2 \times ncvt$ to compute the right singular vectors. The operations to compute the singular values must all be performed in scalar mode; the additional operations to compute the singular vectors can be vectorized and on some machines may be performed much faster.

The complex analogue of this routine is F08MSF (CBDSQR/ZBDSQR).

# 9 Example

To compute the singular value decomposition of the upper bidiagonal matrix $B$, where

$$B = \begin{pmatrix} 3.62 & 1.26 & 0.00 & 0.00 \\ 0.00 & -2.41 & -1.53 & 0.00 \\ 0.00 & 0.00 & 1.92 & 1.19 \\ 0.00 & 0.00 & 0.00 & -1.43 \end{pmatrix}.$$

See also the example for F08KFF, which illustrates the use of the routine to compute the singular value decomposition of a general matrix.

## 9.1 Example 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.

```
*     F08MEF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDVT, LDU, LDC
      PARAMETER        (NMAX=8,LDVT=NMAX,LDU=NMAX,LDC=1)
      real             ZERO, ONE
      PARAMETER        (ZERO=0.0e0,ONE=1.0e0)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             C(LDC,1), D(NMAX), E(NMAX-1), U(LDU,NMAX),
     +                 VT(LDVT,NMAX), WORK(4*NMAX-4)
*     .. External Subroutines ..
      EXTERNAL         sbdsqr, F06QHF, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08MEF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read B from data file
*
         READ (NIN,*) (D(I),I=1,N)
         READ (NIN,*) (E(I),I=1,N-1)
*
         READ (NIN,*) UPLO
*
*        Initialise U and VT to be the unit matrix
*
         CALL F06QHF('General',N,N,ZERO,ONE,U,LDU)
*
         CALL F06QHF('General',N,N,ZERO,ONE,VT,LDVT)
*
*        Calculate the SVD of B
*
         CALL sbdsqr(UPLO,N,N,N,0,D,E,VT,LDVT,U,LDU,C,LDC,WORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.GT.0) THEN
            WRITE (NOUT,*) 'Failure to converge.'
         ELSE
*
*           Print singular values, left & right singular vectors
*
            WRITE (NOUT,*) 'Singular values'
            WRITE (NOUT,99999) (D(I),I=1,N)
            WRITE (NOUT,*)
            IFAIL = 0
*
            CALL X04CAF('General',' ',N,N,VT,LDVT,
```

```
     +                 'Right singular vectors, by row',IFAIL)
*
            WRITE (NOUT,*)
            IFAIL = 0
*
            CALL X04CAF('General',' ',N,N,U,LDU,
     +                 'Left singular vectors, by column',IFAIL)
*
         END IF
       END IF
       STOP
*
99999 FORMAT (3X,(8F8.4))
      END
```

## 9.2 Example Data

```
F08MEF Example Program Data
  4                              :Value of N
  3.62  -2.41   1.92  -1.43
  1.26  -1.53   1.19             :End of matrix B
  'U'                            :Value of UPLO
```

## 9.3 Example Results

```
 F08MEF Example Program Results

 Singular values
     4.0001  3.0006  1.9960  0.9998

 Right singular vectors, by row
          1       2       3       4
 1   0.8261  0.5246  0.2024  0.0369
 2   0.4512 -0.4056 -0.7350 -0.3030
 3   0.2823 -0.5644  0.1731  0.7561
 4   0.1852 -0.4916  0.6236 -0.5789

 Left singular vectors, by column
          1       2       3       4
 1   0.9129  0.3740  0.1556  0.0512
 2  -0.3935  0.7005  0.5489  0.2307
 3   0.1081 -0.5904  0.6173  0.5086
 4  -0.0132  0.1444 -0.5417  0.8280
```

# F08MSF (CBDSQR/ZBDSQR) – 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.

**Warning:** the specification for the parameter WORK changed at Mark 17: the length of WORK may need to be increased if NCVT = NRU = NCC = 0.

## 1 Purpose

F08MSF (CBDSQR/ZBDSQR) computes the singular value decomposition of a complex general matrix which has been reduced to bidiagonal form.

## 2 Specification

```
      SUBROUTINE F08MSF(UPLO, N, NCVT, NRU, NCC, D, E, VT, LDVT, U, LDU,
     1                  C, LDC, WORK, INFO)
      ENTRY       cbdsqr(UPLO, N, NCVT, NRU, NCC, D, E, VT, LDVT, U, LDU,
     1                  C, LDC, WORK, INFO)
      INTEGER           N, NCVT, NRU, NCC, LDVT, LDU, LDC, INFO
      real              D(*), E(*), WORK(*)
      complex           VT(LDVT,*), U(LDU,*), C(LDC,*)
      CHARACTER*1       UPLO
```

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

## 3 Description

This routine computes the singular values, and optionally, the left or right singular vectors of a real upper or lower bidiagonal matrix $B$. In other words, it can compute the singular value decomposition (SVD) of $B$ as

$$B = U\Sigma V^T.$$

Here $\Sigma$ is a diagonal matrix with real diagonal elements $\sigma_i$ (the singular values of $B$), such that

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0;$$

$U$ is an orthogonal matrix whose columns are the left singular vectors $u_i$; $V$ is an orthogonal matrix whose rows are the right singular vectors $v_i$. Thus

$$Bu_i = \sigma_i v_i \text{ and } B^T v_i = \sigma_i u_i \text{ for } i = 1, 2, \ldots, n.$$

To compute $U$ and/or $V^T$, the arrays U and/or VT must be initialized to the unit matrix before F08MSF is called.

The routine stores the real orthogonal matrices $U$ and $V^T$ in ***complex*** arrays U and VT, so that it may also be used to compute the SVD of a complex general matrix $A$ which has been reduced to bidiagonal form by a unitary transformation: $A = QBP^H$. If $A$ is $m$ by $n$ with $m \geq n$, then $Q$ is $m$ by $n$ and $P^H$ is $n$ by $n$; if $A$ is $n$ by $p$ with $n < p$, then $Q$ is $n$ by $n$ and $P^H$ is $n$ by $p$. In this case, the matrices $Q$ and/or $P^H$ must be formed explicitly by F08KTF (CUNGBR/ZUNGBR) and passed to F08MSF in the arrays U and/or VT respectively.

F08MSF also has the capability of forming $U^H C$, where $C$ is an arbitrary complex matrix; this is needed when using the SVD to solve linear least-squares problems.

F08MSF uses two different algorithms. If any singular vectors are required (that is, if NCVT > 0 or NRU > 0 or NCC > 0), the bidiagonal $QR$ algorithm is used, switching between zero-shift and implicitly shifted forms to preserve the accuracy of small singular values, and switching between $QR$ and $QL$ variants in order to handle graded matrices effectively (see Demmel and Kahan [1]). If only singular values are required (that is, if NCVT = NRU = NCC = 0), they are computed by the differential qd algorithm (see Fernando and Parlett [2]), which is faster and can achieve even greater accuracy.

The singular vectors are normalized so that $\| u_i \| = \| v_i \| = 1$, but are determined only to within a complex factor of absolute value 1.

# 4 References

[1] Demmel J W and Kahan W (1990) Accurate singular values of bidiagonal matrices *SIAM J. Sci. Statist. Comput.* **11** 873–912

[2] Fernando K V and Parlett B N (1994) Accurate singular values and differential qd algorithms *Numer. Math.* **67** 191–229

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

# 5 Parameters

**1:** UPLO — CHARACTER*1 *Input*

*On entry:* indicates whether $B$ is an upper or lower bidiagonal matrix as follows:

if UPLO = 'U', then $B$ is an upper bidiagonal matrix;

if UPLO = 'L', then $B$ is a lower bidiagonal matrix.

*Constraint:* UPLO = 'U' or 'L'.

**2:** N — INTEGER *Input*

*On entry:* $n$, the order of the matrix $B$.

*Constraint:* N $\geq 0$.

**3:** NCVT — INTEGER *Input*

*On entry:* *ncvt*, the number of columns of the matrix $V^H$ of right singular vectors. Set NCVT = 0 if no right singular vectors are required.

*Constraint:* NCVT $\geq 0$.

**4:** NRU — INTEGER *Input*

*On entry:* *nru*, the number of rows of the matrix $U$ of left singular vectors. Set NRU = 0 if no left singular vectors are required.

*Constraint:* NRU $\geq 0$.

**5:** NCC — INTEGER *Input*

*On entry:* *ncc*, the number of columns of the matrix $C$. Set NCC = 0 if no matrix $C$ is supplied.

*Constraint:* NCC $\geq 0$.

**6:** D(*) — ***real*** array *Input/Output*

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

*On entry:* the diagonal elements of the bidiagonal matrix $B$.

*On exit:* the singular values in decreasing order of magnitude, unless INFO > 0 (in which case see Section 6).

**7:** E(*) — ***real*** array *Input/Output*

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

*On entry:* the off-diagonal elements of the bidiagonal matrix $B$.

*On exit:* the array is overwritten, but if INFO > 0 see Section 6.

**8:** VT(LDVT,∗) — ***complex*** array *Input/Output*

**Note:** the second dimension of the array VT must be at least max(1, NCVT).

*On entry:* if NCVT > 0, VT must contain an $n$ by $ncvt$ matrix. If the right singular vectors of $B$ are required, $ncvt = n$ and VT must contain the unit matrix; if the right singular vectors of $A$ are required, VT must contain the unitary matrix $P^H$ returned by F08KTF (CUNGBR/ZUNGBR) with VECT = 'P' .

*On exit:* the $n$ by $ncvt$ matrix $V^H$ or $V^H P^H$ of right singular vectors, stored by rows.

VT is not referenced if NCVT = 0.

**9:** LDVT — INTEGER *Input*

*On entry:* the first dimension of the array VT as declared in the (sub)program from which F08MSF (CBDSQR/ZBDSQR) is called.

*Constraints:*

LDVT ≥ max(1, N) if NCVT > 0,
LDVT ≥ 1 otherwise.

**10:** U(LDU,∗) — ***complex*** array *Input/Output*

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

*On entry:* if NRU > 0, U must contain an $nru$ by $n$ matrix. If the left singular vectors of $B$ are required, $nru = n$ and U must contain the unit matrix; if the left singular vectors of $A$ are required, U must contain the unitary matrix $Q$ returned by F08KTF (CUNGBR/ZUNGBR) with VECT = 'Q'.

*On exit:* the $nru$ by $n$ matrix $U$ or $QU$ of left singular vectors, stored by columns.

U is not referenced if NRU = 0.

**11:** LDU — INTEGER *Input*

*On entry:* the first dimension of the array U as declared in the (sub)program from which F08MSF (CBDSQR/ZBDSQR) is called.

*Constraint:* LDU ≥ max(1, NRU).

**12:** C(LDC,∗) — ***complex*** array *Input/Output*

**Note:** the second dimension of the array C must be at least max(1, NCC).

*On entry:* the $n$ by $ncc$ matrix $C$ if NCC > 0.

*On exit:* $C$ is overwritten by the matrix $U^H C$.

C is not referenced if NCC = 0.

**13:** LDC — INTEGER *Input*

*On entry:* the first dimension of the array C as declared in the (sub)program from which F08MSF (CBDSQR/ZBDSQR) is called.

*Constraints:*

LDC ≥ max(1, N) if NCC > 0,
LDC ≥ 1 otherwise.

**14:** WORK(∗) — ***real*** array *Workspace*

**Note:** the dimension of the array WORK must be at least max(1, 2∗N) if NCVT = NRU = NCC = 0, and at least max(1, 4 ∗ (N − 1)) otherwise.

**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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

The algorithm failed to converge and INFO specifies how many off-diagonals did not converge. In this case, D and E contain on exit the diagonal and off-diagonal elements, respectively, of a bidiagonal matrix orthogonally equivalent to $B$.

# 7 Accuracy

Each singular value and singular vector is computed to high relative accuracy. However, the reduction to bidiagonal form (prior to calling the routine) may exclude the possibility of obtaining high relative accuracy in the small singular values of the original matrix if its singular values vary widely in magnitude.

If $\sigma_i$ is an exact singular value of $B$ and $\tilde{\sigma}_i$ is the corresponding computed value, then

$$|\tilde{\sigma}_i - \sigma_i| \leq p(m,n)\epsilon\sigma_i$$

where $p(m,n)$ is a modestly increasing function of $m$ and $n$, and $\epsilon$ is the ***machine precision***. If only singular values are computed, they are computed more accurately (that is, the function $p(m,n)$ is smaller), than when some singular vectors are also computed.

If $u_i$ is the corresponding exact left singular vector of $B$, and $\tilde{u}_i$ is the corresponding computed left singular vector, then the angle $\theta(\tilde{u}_i, u_i)$ between them is bounded as follows:

$$\theta(\tilde{u}_i, u_i) \leq \frac{p(m,n)\epsilon}{relgap_i}$$

where $relgap_i$ is the relative gap between $\sigma_i$ and the other singular values, defined by

$$relgap_i = \min_{i \neq j} \frac{|\sigma_i - \sigma_j|}{(\sigma_i + \sigma_j)}.$$

A similar error bound holds for the right singular vectors.

# 8 Further Comments

The total number of real floating-point operations is roughly proportional to $n^2$ if only the singular values are computed. About $12n^2 \times nru$ additional operations are required to compute the left singular vectors and about $12n^2 \times ncvt$ to compute the right singular vectors. The operations to compute the singular values must all be performed in scalar mode; the additional operations to compute the singular vectors can be vectorized and on some machines may be performed much faster.

The real analogue of this routine is F08MEF (SBDSQR/DBDSQR).

# 9 Example

See the example for F08KTF (CUNGBR/ZUNGBR), which illustrates the use of the routine to compute the singular value decomposition of a general matrix.

# F08NEF (SGEHRD/DGEHRD) – 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

F08NEF (SGEHRD/DGEHRD) reduces a real general matrix to Hessenberg form.

## 2. Specification

```
SUBROUTINE F08NEF (N, ILO, IHI, A, LDA, TAU, WORK, LWORK, INFO)
ENTRY      sgehrd (N, ILO, IHI, A, LDA, TAU, WORK, LWORK, INFO)
INTEGER          N, ILO, IHI, 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 reduces a real general matrix $A$ to upper Hessenberg form $H$ by an orthogonal similarity transformation: $A = QHQ^T$.

The matrix $Q$ is not formed explicitly, but is represented as a product of elementary reflectors (see the Chapter Introduction for details). Routines are provided to work with $Q$ in this representation (see Section 8).

The routine can take advantage of a previous call to F08NHF (SGEBAL/DGEBAL), which may produce a matrix with the structure:

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} \\ & A_{22} & A_{23} \\ & & A_{33} \end{pmatrix}$$

where $A_{11}$ and $A_{33}$ are upper triangular. If so, only the central diagonal block $A_{22}$, in rows and columns $i_{lo}$ to $i_{hi}$, needs to be reduced to Hessenberg form (the blocks $A_{12}$ and $A_{23}$ will also be affected by the reduction). Therefore the values of $i_{lo}$ and $i_{hi}$ determined by F08NHF can be supplied to the routine directly. If F08NHF has not previously been called however, then $i_{lo}$ must be set to 1 and $i_{hi}$ to $n$.

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §7.4.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

## 5. Parameters

1: N – INTEGER. *Input*

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

*Constraint*: N ≥ 0.

2: ILO – INTEGER. *Input*

3: IHI – INTEGER. *Input*

*On entry*: if $A$ has been output by F08NHF (SGEBAL/DGEBAL), then ILO and IHI **must** contain the values returned by that routine. Otherwise, ILO must be set to 1 and IHI to N.

*Constraints*: 1 ≤ ILO ≤ IHI ≤ N if N > 0,
ILO = 1 and IHI = 0 if N = 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 $n$ general matrix $A$.

*On exit*: $A$ is overwritten by the upper Hessenberg matrix $H$ and details of the orthogonal matrix $Q$.

5: LDA – INTEGER. *Input*

*On entry*: the first dimension of the array A as declared in the (sub)program from which F08NEF (SGEHRD/DGEHRD) is called.

*Constraint*: LDA ≥ max(1,N).

6: TAU(*) – ***real*** array. *Output*

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

*On exit*: further details of the orthogonal matrix $Q$.

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 F08NEF (SGEHRD/DGEHRD) 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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

## 7. Accuracy

The computed Hessenberg matrix $H$ is exactly similar to a nearby matrix $A + E$, where

$$\|E\|_2 \leq c(n)\varepsilon\|A\|_2,$$

$c(n)$ is a modestly increasing function of $n$, and $\varepsilon$ is the ***machine precision***.

The elements of $H$ themselves may be sensitive to small perturbations in $A$ or to rounding errors in the computation, but this does not affect the stability of the eigenvalues, eigenvectors or Schur factorization.

## 8. Further Comments

The total number of floating-point operations is approximately $\frac{2}{3}q^2(2q+3n)$, where $q = i_{hi}-i_{lo}$; if $i_{lo} = 1$ and $i_{hi} = n$, the number is approximately $10n^3/3$.

To form the orthogonal matrix $Q$ this routine may be followed by a call to F08NFF (SORGHR/DORGHR):

```
CALL SORGHR (N, ILO, IHI, A, LDA, TAU, WORK, LWORK, INFO)
```

To apply $Q$ to an $m$ by $n$ real matrix $C$ this routine may be followed by a call to F08NGF (SORMHR/DORMHR). For example,

```
      CALL SORMHR ('Left','No Transpose',M,N,ILO,IHI,A,LDA,TAU,C,LDC,
     +             WORK,LWORK,INFO)
```

forms the matrix product $QC$.

The complex analogue of this routine is F08NSF (CGEHRD/ZGEHRD).

## 9. Example

To compute the upper Hessenberg form of the matrix $A$, where

$$A = \begin{pmatrix} 0.35 & 0.45 & -0.14 & -0.17 \\ 0.09 & 0.07 & -0.54 & 0.35 \\ -0.44 & -0.33 & -0.03 & 0.17 \\ 0.25 & -0.32 & -0.13 & 0.11 \end{pmatrix}.$$

### 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.**

```
*     F08NEF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, LWORK
      PARAMETER        (NMAX=8,LDA=NMAX,LWORK=64*NMAX)
      real             ZERO
      PARAMETER        (ZERO=0.0e0)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
*     .. Local Arrays ..
      real             A(LDA,NMAX), TAU(NMAX-1), WORK(LWORK)
*     .. External Subroutines ..
      EXTERNAL         sgehrd, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08NEF 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)
*
*        Reduce A to upper Hessenberg form
*
         CALL sgehrd(N,1,N,A,LDA,TAU,WORK,LWORK,INFO)
*
*        Set the elements below the first sub-diagonal to zero
*
         DO 40 I = 1, N - 2
            DO 20 J = I + 2, N
               A(J,I) = ZERO
   20       CONTINUE
   40    CONTINUE
*
*        Print upper Hessenberg form
*
         WRITE (NOUT,*)
         IFAIL = 0
*
```

```
         CALL X04CAF('General',' ',N,N,A,LDA,'Upper Hessenberg form',
     +               IFAIL)
*
      END IF
      STOP
      END
```

## 9.2. Program Data

```
F08NEF Example Program Data
  4                                   :Value of N
  0.35   0.45  -0.14  -0.17
  0.09   0.07  -0.54   0.35
 -0.44  -0.33  -0.03   0.17
  0.25  -0.32  -0.13   0.11           :End of matrix A
```

## 9.3. Program Results

```
F08NEF Example Program Results

Upper Hessenberg form
         1       2       3       4
1   0.3500 -0.1160 -0.3886 -0.2942
2  -0.5140  0.1225  0.1004  0.1126
3   0.0000  0.6443 -0.1357 -0.0977
4   0.0000  0.0000  0.4262  0.1632
```

# F08NFF (SORGHR/DORGHR) – 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

F08NFF (SORGHR/DORGHR) generates the real orthogonal matrix $Q$ which was determined by F08NEF (SGEHRD/DGEHRD), when reducing a real general matrix $A$ to Hessenberg form.

## 2. Specification

```
SUBROUTINE F08NFF (N, ILO, IHI, A, LDA, TAU, WORK, LWORK, INFO)
ENTRY      sorghr (N, ILO, IHI, A, LDA, TAU, WORK, LWORK, INFO)
INTEGER        N, ILO, IHI, 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 following a call to F08NEF (SGEHRD/DGEHRD), which reduces a real general matrix $A$ to upper Hessenberg form $H$ by an orthogonal similarity transformation: $A = QHQ^T$. F08NEF represents the matrix $Q$ as a product of $i_{hi}-i_{lo}$ elementary reflectors. Here $i_{lo}$ and $i_{hi}$ are values determined by F08NHF (SGEBAL/DGEBAL) when balancing the matrix; if the matrix has not been balanced, $i_{lo} = 1$ and $i_{hi} = n$.

This routine may be used to generate $Q$ explicitly as a square matrix. $Q$ has the structure:

$$Q = \begin{pmatrix} I & 0 & 0 \\ 0 & Q_{22} & 0 \\ 0 & 0 & I \end{pmatrix}$$

where $Q_{22}$ occupies rows and columns $i_{lo}$ to $i_{hi}$.

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §7.4.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

## 5. Parameters

1: N – INTEGER. *Input*

*On entry*: $n$, the order of the matrix $Q$.

*Constraint*: N ≥ 0.

2: ILO – INTEGER. *Input*

3: IHI – INTEGER. *Input*

*On entry*: these **must** be the same parameters ILO and IHI, respectively, as supplied to F08NEF (SGEHRD/DGEHRD).

*Constraints*: 1 ≤ ILO ≤ IHI ≤ N if N > 0,
ILO = 1 and IHI = 0 if N = 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 F08NEF (SGEHRD/DGEHRD).

*On exit*: the $n$ by $n$ orthogonal matrix $Q$.

5: LDA – INTEGER. *Input*

*On entry*: the first dimension of the array A as declared in the (sub)program from which F08NFF (SORGHR/DORGHR) is called.

*Constraint*: LDA ≥ max(1,N).

6: TAU(*) – ***real*** array. *Input*

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

*On entry*: further details of the elementary reflectors, as returned by F08NEF (SGEHRD/DGEHRD).

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 F08NFF (SORGHR/DORGHR) is called.

*Suggested value*: for optimum performance LWORK should be at least (IHI–ILO)×*nb*, where *nb* is the ***blocksize***.

*Constraint*: LWORK ≥ max(1,IHI–ILO).

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 $i$th 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 $\varepsilon$ is the ***machine precision***.

## 8. Further Comments

The total number of floating-point operations is approximately $\frac{4}{3}q^3$, where $q = i_{hi} - i_{lo}$.

The complex analogue of this routine is F08NTF (CUNGHR/ZUNGHR).

## 9. Example

To compute the Schur factorization of the matrix $A$, where

$$A = \begin{pmatrix} 0.35 & 0.45 & -0.14 & -0.17 \\ 0.09 & 0.07 & -0.54 & 0.35 \\ -0.44 & -0.33 & -0.03 & 0.17 \\ 0.25 & -0.32 & -0.13 & 0.11 \end{pmatrix}.$$

Here $A$ is general and must first be reduced to Hessenberg form by F08NEF (SGEHRD/DGEHRD). The program then calls F08NFF (SORGHR/DORGHR) to form $Q$, and passes this matrix to F08PEF (SHSEQR/DHSEQR) which computes the Schur factorization 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.**

```
*     F08NFF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, LDZ, LWORK
      PARAMETER        (NMAX=8,LDA=NMAX,LDZ=NMAX,LWORK=64*(NMAX-1))
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
*     .. Local Arrays ..
      real             A(LDA,NMAX), TAU(NMAX), WI(NMAX), WORK(LWORK),
     +                 WR(NMAX), Z(LDZ,NMAX)
*     .. External Subroutines ..
      EXTERNAL         sgehrd, shseqr, sorghr, F06QFF, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08NFF 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)
*
*        Reduce A to upper Hessenberg form H = (Q**T)*A*Q
*
         CALL sgehrd(N,1,N,A,LDA,TAU,WORK,LWORK,INFO)
*
*        Copy A into Z
*
         CALL F06QFF('General',N,N,A,LDA,Z,LDZ)
*
*        Form Q explicitly, storing the result in Z
*
         CALL sorghr(N,1,N,Z,LDZ,TAU,WORK,LWORK,INFO)
*
*        Calculate the Schur factorization of H = Y*T*(Y**T) and form
*        Q*Y explicitly, storing the result in Z
*
*        Note that A = Z*T*(Z**T), where Z = Q*Y
*
         CALL shseqr('Schur form','Vectors',N,1,N,A,LDA,WR,WI,Z,LDZ,
     +               WORK,LWORK,INFO)
*
*        Print Schur form
*
         WRITE (NOUT,*)
         IFAIL = 0
*
         CALL X04CAF('General',' ',N,N,A,LDA,'Schur form',IFAIL)
*
*        Print Schur vectors
*
         WRITE (NOUT,*)
         IFAIL = 0
*
         CALL X04CAF('General',' ',N,N,Z,LDZ,'Schur vectors of A',IFAIL)
*
      END IF
      STOP
*
      END
```

## 9.2. Program Data

```
F08NFF Example Program Data
  4                                   :Value of N
  0.35   0.45  -0.14  -0.17
  0.09   0.07  -0.54   0.35
 -0.44  -0.33  -0.03   0.17
  0.25  -0.32  -0.13   0.11           :End of matrix A
```

## 9.3. Program Results

```
F08NFF Example Program Results

Schur form
          1       2       3       4
1    0.7995 -0.1144 -0.0060  0.0336
2    0.0000 -0.0994 -0.2478  0.3474
3    0.0000  0.6483 -0.0994 -0.2026
4    0.0000  0.0000  0.0000 -0.1007

Schur vectors of A
          1       2       3       4
1    0.6551  0.1037 -0.3450  0.6641
2    0.5236 -0.5807  0.6141 -0.1068
3   -0.5362 -0.3073  0.2935  0.7293
4    0.0956  0.7467  0.6463  0.1249
```

# F08NGF (SORMHR/DORMHR) – 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

F08NGF (SORMHR/DORMHR) multiplies an arbitrary real matrix $C$ by the real orthogonal matrix $Q$ which was determined by F08NEF (SGEHRD/DGEHRD) when reducing a real general matrix to Hessenberg form.

## 2. Specification

```
      SUBROUTINE F08NGF (SIDE, TRANS, M, N, ILO, IHI, A, LDA, TAU, C, LDC,
     1                   WORK, LWORK, INFO)
      ENTRY       sormhr (SIDE, TRANS, M, N, ILO, IHI, A, LDA, TAU, C, LDC,
     1                   WORK, LWORK, INFO)
      INTEGER          M, N, ILO, IHI, 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 following a call to F08NEF (SGEHRD/DGEHRD), which reduces a real general matrix $A$ to upper Hessenberg form $H$ by an orthogonal similarity transformation: $A = QHQ^T$. F08NEF represents the matrix $Q$ as a product of $i_{hi}-i_{lo}$ elementary reflectors. Here $i_{lo}$ and $i_{hi}$ are values determined by F08NHF (SGEBAL/DGEBAL) when balancing the matrix; if the matrix has not been balanced, $i_{lo} = 1$ and $i_{hi} = n$.

This routine may be used to form one of the matrix products

$$QC, Q^TC, CQ \text{ or } CQ^T,$$

overwriting the result on $C$ (which may be any real rectangular matrix).

A common application of this routine is to transform a matrix $V$ of eigenvectors of $H$ to the matrix $QV$ of eigenvectors of $A$.

## 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 $C$; $m$ is also the order of $Q$ if SIDE = 'L'.

*Constraint*: M ≥ 0.

4: N – INTEGER. *Input*

*On entry*: $n$, the number of columns of the matrix $C$; $n$ is also the order of $Q$ if SIDE = 'R'.

*Constraint*: N ≥ 0.

5: ILO – INTEGER. *Input*

6: IHI – INTEGER. *Input*

*On entry*: these **must** be the same parameters ILO and IHI, respectively, as supplied to F08NEF (SGEHRD/DGEHRD).

*Constraints*: 1 ≤ ILO ≤ IHI ≤ M if SIDE = 'L' and M > 0;
ILO = 1 and IHI = 0 if SIDE = 'L' and M = 0;
1 ≤ ILO ≤ IHI ≤ N if SIDE = 'R' and N > 0;
ILO = 1 and IHI = 0 if SIDE = 'R' and N = 0.

7: 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 F08NEF (SGEHRD/DGEHRD).

8: LDA – INTEGER. *Input*

*On entry*: the first dimension of the array A as declared in the (sub)program from which F08NGF (SORMHR/DORMHR) is called.

*Constraints*: LDA ≥ max(1,M) if SIDE = 'L',
LDA ≥ max(1,N) if SIDE = 'R'.

9: TAU(*) – ***real*** array. *Input*

**Note**: the dimension of the array TAU must be at least max(1,M–1) if SIDE = 'L' and at least max(1,N–1) if SIDE = 'R'.

*On entry*: further details of the elementary reflectors, as returned by F08NEF (SGEHRD/DGEHRD).

10: 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.

11: LDC – INTEGER. *Input*

*On entry*: the first dimension of the array C as declared in the (sub)program from which F08NGF (SORMHR/DORMHR) is called.

*Constraint*: LDC ≥ max(1,M).

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

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

13: LWORK – INTEGER. *Input*

*On entry*: the dimension of the array WORK as declared in the (sub)program from which F08NGF (SORMHR/DORMHR) 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'.

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 $i$th 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 $\varepsilon$ is the ***machine precision***.

## 8. Further Comments

The total number of floating-point operations is approximately $2nq^2$ if SIDE = 'L' and $2mq^2$ if SIDE = 'R', where $q = i_{hi} - i_{lo}$.

The complex analogue of this routine is F08NUF (CUNMHR/ZUNMHR).

## 9. Example

To compute all the eigenvalues of the matrix $A$, where

$$A = \begin{pmatrix} 0.35 & 0.45 & -0.14 & -0.17 \\ 0.09 & 0.07 & -0.54 & 0.35 \\ -0.44 & -0.33 & -0.03 & 0.17 \\ 0.25 & -0.32 & -0.13 & 0.11 \end{pmatrix},$$

and those eigenvectors which correspond to eigenvalues $\lambda$ such that $Re(\lambda) < 0$. Here $A$ is general and must first be reduced to upper Hessenberg form $H$ by F08NEF (SGEHRD/DGEHRD). The program then calls F08PEF (SHSEQR/DHSEQR) to compute the eigenvalues, and F08PKF (SHSEIN/DHSEIN) to compute the required eigenvectors of $H$ by inverse iteration. Finally F08NGF (SORMHR/DORMHR) is called to transform the eigenvectors of $H$ back to eigenvectors of the original matrix $A$.

### 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.**

```
*     F08NGF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, LDH, LDZ, LWORK, LDVL, LDVR
      PARAMETER        (NMAX=8,LDA=NMAX,LDH=NMAX,LDZ=1,LWORK=64*NMAX,
     +                 LDVL=NMAX,LDVR=NMAX)
*     .. Local Scalars ..
      real             THRESH
      INTEGER          I, IFAIL, INFO, J, M, N
```

```
*     .. Local Arrays ..
      real               A(LDA,NMAX), H(LDH,NMAX), TAU(NMAX),
     +                   VL(LDVL,NMAX), VR(LDVR,NMAX), WI(NMAX),
     +                   WORK(LWORK), WR(NMAX), Z(LDZ,1)
      INTEGER            IFAILL(NMAX), IFAILR(NMAX)
      LOGICAL            SELECT(NMAX)
*     .. External Subroutines ..
      EXTERNAL           sgehrd, shsein, shseqr, sormhr, F06QFF, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08NGF 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)
*
         READ (NIN,*) THRESH
*
*        Reduce A to upper Hessenberg form H = (Q**T)*A*Q
*
         CALL sgehrd(N,1,N,A,LDA,TAU,WORK,LWORK,INFO)
*
*        Copy A to H
*
         CALL F06QFF('General',N,N,A,LDA,H,LDH)
*
*        Calculate the eigenvalues of H (same as A)
*
         CALL shseqr('Eigenvalues','No vectors',N,1,N,H,LDH,WR,WI,Z,LDZ,
     +               WORK,LWORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.GT.0) THEN
            WRITE (NOUT,*) 'Failure to converge.'
         ELSE
            WRITE (NOUT,*) 'Eigenvalues'
            WRITE (NOUT,99999) (' (',WR(I),',',WI(I),')',I=1,N)
*
            DO 20 I = 1, N
               SELECT(I) = WR(I) .LT. THRESH
   20       CONTINUE
*
*           Calculate the eigenvectors of H (as specified by SELECT),
*           storing the result in VR
*
            CALL shsein('Right','QR','No initial vectors',SELECT,N,A,
     +                  LDA,WR,WI,VL,LDVL,VR,LDVR,N,M,WORK,IFAILL,
     +                  IFAILR,INFO)
*
*           Calculate the eigenvectors of A = Q * (eigenvectors of H)
*
            CALL sormhr('Left','No transpose',N,M,1,N,A,LDA,TAU,VR,LDVR,
     +                  WORK,LWORK,INFO)
*
*           Print eigenvectors
*
            WRITE (NOUT,*)
            IFAIL = 0
*
            CALL X04CAF('General',' ',N,M,VR,LDVR,
     +                  'Contents of array VR',IFAIL)
*
         END IF
      END IF
      STOP
*
99999 FORMAT (1X,A,F8.4,A,F8.4,A)
      END
```

## 9.2. Program Data

```
F08NGF Example Program Data
  4                                   :Value of N
  0.35   0.45  -0.14  -0.17
  0.09   0.07  -0.54   0.35
 -0.44  -0.33  -0.03   0.17
  0.25  -0.32  -0.13   0.11           :End of matrix A
  0.0                                 :Value of THRESH
```

## 9.3. Program Results

```
F08NGF Example Program Results

Eigenvalues
 (  0.7995,  0.0000)
 ( -0.0994,  0.4008)
 ( -0.0994, -0.4008)
 ( -0.1007,  0.0000)

Contents of array VR
          1       2       3
1   0.3881  0.0574  0.1493
2  -0.7107  0.0380  0.3956
3  -0.3891  0.0778  0.7075
4  -0.3996 -0.7270  0.8603
```

# F08NHF (SGEBAL/DGEBAL) – 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

F08NHF (SGEBAL/DGEBAL) balances a real general matrix in order to improve the accuracy of computed eigenvalues and/or eigenvectors.

## 2. Specification

```
SUBROUTINE F08NHF (JOB, N, A, LDA, ILO, IHI, SCALE, INFO)
ENTRY      sgebal  (JOB, N, A, LDA, ILO, IHI, SCALE, INFO)
INTEGER        N, LDA, ILO, IHI, INFO
real           A(LDA,*), SCALE(*)
CHARACTER*1    JOB
```

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

## 3. Description

This routine balances a real general matrix $A$. The term 'balancing' covers two steps, each of which involves a similarity transformation of $A$. The routine can perform either or both of these steps.

1. The routine first attempts to permute $A$ to block upper triangular form by a similarity transformation:

$$PAP^T = A' = \begin{pmatrix} A'_{11} & A'_{12} & A'_{13} \\ 0 & A'_{22} & A'_{23} \\ 0 & 0 & A'_{33} \end{pmatrix}$$

where $P$ is a permutation matrix, and $A'_{11}$ and $A'_{33}$ are upper triangular. Then the diagonal elements of $A'_{11}$ and $A'_{33}$ are eigenvalues of $A$. The rest of the eigenvalues of $A$ are the eigenvalues of the central diagonal block $A'_{22}$, in rows and columns $i_{lo}$ to $i_{hi}$. Subsequent operations to compute the eigenvalues of $A$ (or its Schur factorization) need only be applied to these rows and columns; this can save a significant amount of work if $i_{lo} > 1$ and $i_{hi} < n$. If no suitable permutation exists (as is often the case), the routine sets $i_{lo} = 1$ and $i_{hi} = n$, and $A'_{22}$ is the whole of $A$.

2. The routine applies a diagonal similarity transformation to $A'$, to make the rows and columns of $A'_{22}$ as close in norm as possible:

$$A'' = DA'D^{-1} = \begin{pmatrix} I & 0 & 0 \\ 0 & D_{22} & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} A'_{11} & A'_{12} & A'_{13} \\ 0 & A'_{22} & A'_{23} \\ 0 & 0 & A'_{33} \end{pmatrix} \begin{pmatrix} I & 0 & 0 \\ 0 & D_{22}^{-1} & 0 \\ 0 & 0 & I \end{pmatrix}.$$

This scaling can reduce the norm of the matrix (that is, $\|A''_{22}\| < \|A'_{22}\|$), and hence reduce the effect of rounding errors on the accuracy of computed eigenvalues and eigenvectors.

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §7.5.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

## 5. Parameters

1: JOB – CHARACTER*1. *Input*

*On entry*: indicates whether $A$ is to be permuted to and/or scaled (or neither), as follows:

if JOB = 'N', then $A$ is neither permuted nor scaled (but values are assigned to ILO, IHI and SCALE);

if JOB = 'P', then $A$ is permuted but not scaled;

if JOB = 'S', then $A$ is scaled but not permuted;

if JOB = 'B', then $A$ is both permuted and scaled.

*Constraint*: JOB = 'N', 'P', 'S' or 'B'.

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*: the $n$ by $n$ general matrix $A$.

*On exit*: $A$ is overwritten by the balanced matrix.

A is not referenced if JOB = 'N'.

4: LDA – INTEGER. *Input*

*On entry*: the first dimension of the array A as declared in the (sub)program from which F08NHF (SGEBAL/DGEBAL) is called.

*Constraint*: LDA ≥ max(1,N).

5: ILO – INTEGER. *Output*

6: IHI – INTEGER. *Output*

*On exit*: the values $i_{lo}$ and $i_{hi}$ such that on exit A($i$,$j$) is zero if $i > j$ and $1 \le j < i_{lo}$ or $i_{hi} < i \le n$.

If JOB = 'N' or 'S', then $i_{lo} = 1$ and $i_{hi} = n$.

7: SCALE(*) – ***real*** array. *Output*

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

*On exit*: details of the permutations and scaling factors applied to $A$. More precisely, if $p_j$ is the index of the row and column interchanged with row and column $j$ and $d_j$ is the scaling factor used to balance row and column $j$, then

$$\text{SCALE}(j) = \begin{cases} p_j, & \text{for } j = 1,2,\ldots,i_{lo}-1 \\ d_j, & \text{for } j = i_{lo},i_{lo}+1,\ldots,i_{hi} \text{ and} \\ p_j, & \text{for } j = i_{hi}+1,i_{hi}+2,\ldots,n. \end{cases}$$

The order in which the interchanges are made is $n$ to $i_{hi}+1$, then 1 to $i_{lo}-1$.

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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

## 7. Accuracy

The errors are negligible, compared with those in subsequent computations.

## 8. Further Comments

If the matrix $A$ is balanced by this routine, then any eigenvectors computed subsequently are eigenvectors of the matrix $A''$ (see Section 3) and hence F08NJF (SGEBAK/DGEBAK) **must** then be called to transform them back to eigenvectors of $A$.

If the Schur vectors of $A$ are required, then this routine must **not** be called with JOB = 'S' or 'B', because then the balancing transformation is not orthogonal. If this routine is called with JOB = 'P', then any Schur vectors computed subsequently are Schur vectors of the matrix $A''$, and F08NJF **must** be called (with SIDE = 'R') to transform them back to Schur vectors of $A$.

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

The complex analogue of this routine is F08NVF (CGEBAL/ZGEBAL).

## 9. Example

To compute all the eigenvalues and right eigenvectors of the matrix $A$, where

$$A = \begin{pmatrix} 5.14 & 0.91 & 0.00 & -32.80 \\ 0.91 & 0.20 & 0.00 & 34.50 \\ 1.90 & 0.80 & -0.40 & -3.00 \\ -0.33 & 0.35 & 0.00 & 0.66 \end{pmatrix}.$$

The program first calls F08NHF (SGEBAL/DGEBAL) to balance the matrix; it then computes the Schur factorization of the balanced matrix, by reduction to Hessenberg form and the $QR$ algorithm. Then it calls F08QKF (STREVC/DTREVC) to compute the right eigenvectors of the balanced matrix, and finally calls F08NJF (SGEBAK/DGEBAK) to transform the eigenvectors back to eigenvectors of the original matrix $A$.

### 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.**

```
*     F08NHF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, LDH, LWORK, LDVL, LDVR
      PARAMETER        (NMAX=8,LDA=NMAX,LDH=NMAX,LWORK=64*NMAX,LDVL=1,
     +                 LDVR=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, IHI, ILO, INFO, J, M, N
*     .. Local Arrays ..
      real             A(LDA,NMAX), H(LDH,NMAX), SCALE(NMAX), TAU(NMAX),
     +                 VL(LDVL,1), VR(LDVR,NMAX), WI(NMAX), WORK(LWORK),
     +                 WR(NMAX)
      LOGICAL          SELECT(1)
*     .. External Subroutines ..
      EXTERNAL         sgebak, sgebal, sgehrd, shseqr, sorghr, strevc,
     +                 F06QFF, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08NHF 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)
*
```

```
*        Balance A
*
         CALL sgebal('Both',N,A,LDA,ILO,IHI,SCALE,INFO)
*
*        Reduce A to upper Hessenberg form H = (Q**T)*A*Q
*
         CALL sgehrd(N,ILO,IHI,A,LDA,TAU,WORK,LWORK,INFO)
*
*        Copy A to H
*
         CALL F06QFF('General',N,N,A,LDA,H,LDH)
*
*        Copy A into VR
*
         CALL F06QFF('General',N,N,A,LDA,VR,LDVR)
*
*        Form Q explicitly, storing the result in VR
*
         CALL sorghr(N,1,N,VR,LDVR,TAU,WORK,LWORK,INFO)
*
*        Calculate the eigenvalues and Schur factorization of A
*
         CALL shseqr('Schur form','Vectors',N,ILO,IHI,H,LDH,WR,WI,VR,
     +               LDVR,WORK,LWORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.GT.0) THEN
            WRITE (NOUT,*) 'Failure to converge.'
         ELSE
            WRITE (NOUT,*) 'Eigenvalues'
            WRITE (NOUT,99999) (' (',WR(I),',',WI(I),')',I=1,N)
*
*           Calculate the eigenvectors of A, storing the result in VR
*
            CALL strevc('Right','Overwrite',SELECT,N,H,LDH,VL,LDVL,VR,
     +                  LDVR,N,M,WORK,INFO)
*
*           Backtransform eigenvectors
*
            CALL sgebak('Both','Right',N,ILO,IHI,SCALE,M,VR,LDVR, INFO)
*
*           Print eigenvectors
*
            WRITE (NOUT,*)
            IFAIL = 0
*
            CALL X04CAF('General',' ',N,M,VR,LDVR,
     +                  'Contents of array VR',IFAIL)
*
         END IF
       END IF
       STOP
*
99999  FORMAT (1X,A,F8.4,A,F8.4,A)
       END
```

## 9.2. Program Data

```
F08NHF Example Program Data
  4                              :Value of N
 5.14   0.91    0.00 -32.80
 0.91   0.20    0.00  34.50
 1.90   0.80   -0.40  -3.00
-0.33   0.35    0.00   0.66    :End of matrix A
```

## 9.3. Program Results

```
F08NHF Example Program Results

Eigenvalues
 ( -0.4000,  0.0000)
 (  3.0136,  0.0000)
 ( -4.0208,  0.0000)
 (  7.0072,  0.0000)

Contents of array VR
                1          2          3          4
1          0.0000     1.1688     4.4886     3.8149
2          0.0000     1.9812    -9.1416    -0.6873
3          1.0000     1.0000     0.4930     1.0000
4          0.0000     0.1307     1.0000    -0.2362
```

# F08NJF (SGEBAK/DGEBAK) – 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

F08NJF (SGEBAK/DGEBAK) transforms eigenvectors of a balanced matrix to those of the original real nonsymmetric matrix.

## 2. Specification

```
SUBROUTINE F08NJF (JOB, SIDE, N, ILO, IHI, SCALE, M, V, LDV, INFO)
ENTRY      sgebak (JOB, SIDE, N, ILO, IHI, SCALE, M, V, LDV, INFO)
INTEGER          N, ILO, IHI, M, LDV, INFO
real             SCALE(*), V(LDV,*)
CHARACTER*1      JOB, SIDE
```

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

## 3. Description

This routine is intended to be used after a real nonsymmetric matrix $A$ has been balanced by F08NHF (SGEBAL/DGEBAL), and eigenvectors of the balanced matrix $A''_{22}$ have subsequently been computed.

For a description of balancing, see the document for F08NHF. The balanced matrix $A''$ is obtained as $A'' = DPAP^TD^{-1}$, where $P$ is a permutation matrix and $D$ is a diagonal scaling matrix. This routine transforms left or right eigenvectors as follows:

– if $x$ is a right eigenvector of $A''$,

$P^TD^{-1}x$ is a right eigenvector of $A$;

– if $y$ is a left eigenvector of $A''$,

$P^TDy$ is a left eigenvector of $A$.

## 4. References

None.

## 5. Parameters

1: JOB – CHARACTER*1. *Input*

*On entry*: this **must** be the same parameter JOB as supplied to F08NHF (SGEBAL/DGEBAL).

*Constraint*: JOB = 'N', 'P', 'S' or 'B'.

2: SIDE – CHARACTER*1. *Input*

*On entry*: indicates whether left or right eigenvectors are to be transformed, as follows:

if SIDE = 'L', then left eigenvectors are transformed;

if SIDE = 'R', then right eigenvectors are transformed.

*Constraint*: SIDE = 'L' or 'R'.

3: N – INTEGER. *Input*

*On entry*: $n$, the number of rows of the matrix of eigenvectors.

*Constraint*: $N \geq 0$.

4: ILO – INTEGER. *Input*
5: IHI – INTEGER. *Input*

*On entry*: the values $i_{lo}$ and $i_{hi}$, as returned by F08NHF (SGEBAL/DGEBAL).

*Constraints*: $1 \leq \text{ILO} \leq \text{IHI} \leq \text{N}$ if $\text{N} > 0$,
ILO = 1 and IHI = 0 if N = 0.

6: SCALE(*) – ***real*** array. *Input*

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

*On entry*: details of the permutations and/or the scaling factors used to balance the original real nonsymmetric matrix, as returned by F08NHF (SGEBAL/DGEBAL).

7: M – INTEGER. *Input*

*On entry*: $m$, the number of columns of the matrix of eigenvectors.

*Constraint*: $\text{M} \geq 0$.

8: V(LDV,*) – ***real*** array. *Input/Output*

**Note**: the second dimension of the array V must be at least max(1,M).

*On entry*: the matrix of left or right eigenvectors to be transformed.

*On exit*: the transformed eigenvectors.

9: LDV – INTEGER. *Input*

*On entry*: the first dimension of the array V as declared in the (sub)program from which F08NJF (SGEBAK/DGEBAK) is called.

*Constraint*: $\text{LDV} \geq \max(1,\text{N})$.

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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

## 7. Accuracy

The errors are negligible.

## 8. Further Comments

The total number of floating-point operations is approximately proportional to $nm$.

The complex analogue of this routine is F08NWF (CGEBAK/ZGEBAK).

## 9. Example

See the example for F08NHF (SGEBAL/DGEBAL).

# F08NSF (CGEHRD/ZGEHRD) – 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

F08NSF (CGEHRD/ZGEHRD) reduces a complex general matrix to Hessenberg form.

## 2. Specification

```
SUBROUTINE F08NSF (N, ILO, IHI, A, LDA, TAU, WORK, LWORK, INFO)
ENTRY      cgehrd (N, ILO, IHI, A, LDA, TAU, WORK, LWORK, INFO)
INTEGER        N, ILO, IHI, 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 reduces a complex general matrix $A$ to upper Hessenberg form $H$ by a unitary similarity transformation: $A = QHQ^H$. $H$ has real subdiagonal elements.

The matrix $Q$ is not formed explicitly, but is represented as a product of elementary reflectors (see the Chapter Introduction for details). Routines are provided to work with $Q$ in this representation (see Section 8).

The routine can take advantage of a previous call to F08NVF (CGEBAL/ZGEBAL), which may produce a matrix with the structure:

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} \\ & A_{22} & A_{23} \\ & & A_{33} \end{pmatrix}$$

where $A_{11}$ and $A_{33}$ are upper triangular. If so, only the central diagonal block $A_{22}$, in rows and columns $i_{lo}$ to $i_{hi}$, needs to be reduced to Hessenberg form (the blocks $A_{12}$ and $A_{23}$ will also be affected by the reduction). Therefore the values of $i_{lo}$ and $i_{hi}$ determined by F08NVF can be supplied to the routine directly. If F08NVF has not previously been called however, then $i_{lo}$ must be set to 1 and $i_{hi}$ to $n$.

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §7.4.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

## 5. Parameters

1: N – INTEGER. *Input*

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

*Constraint*: N ≥ 0.

2: ILO – INTEGER. *Input*

3: IHI – INTEGER. *Input*

*On entry*: if $A$ has been output by F08NVF (CGEBAL/ZGEBAL), then ILO and IHI **must** contain the values returned by that routine. Otherwise, ILO must be set to 1 and IHI to N.

*Constraints*: 1 ≤ ILO ≤ IHI ≤ N if N > 0,
ILO = 1 and IHI = 0 if N = 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 $n$ general matrix $A$.

*On exit*: $A$ is overwritten by the upper Hessenberg matrix $H$ and details of the unitary matrix $Q$. The subdiagonal elements of $H$ are real.

5: LDA – INTEGER. *Input*

*On entry*: the first dimension of the array A as declared in the (sub)program from which F08NSF (CGEHRD/ZGEHRD) is called.

*Constraint*: LDA ≥ max(1,N).

6: TAU(*) – ***complex*** array. *Output*

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

*On exit*: further details of the unitary matrix $Q$.

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 F08NSF (CGEHRD/ZGEHRD) 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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

## 7. Accuracy

The computed Hessenberg matrix $H$ is exactly similar to a nearby matrix $A + E$, where

$$\|E\|_2 \leq c(n)\varepsilon\|A\|_2,$$

$c(n)$ is a modestly increasing function of $n$, and $\varepsilon$ is the ***machine precision***.

The elements of $H$ themselves may be sensitive to small perturbations in $A$ or to rounding errors in the computation, but this does not affect the stability of the eigenvalues, eigenvectors or Schur factorization.

## 8. Further Comments

The total number of real floating-point operations is approximately $\frac{1}{3}q^2(2q+3n)$, where $q = i_{hi}-i_{lo}$; if $i_{lo} = 1$ and $i_{hi} = n$, the number is approximately $\frac{40n^3}{3}$.

To form the unitary matrix $Q$ this routine may be followed by a call to F08NTF (CUNGHR/ZUNGHR):

```
CALL CUNGHR (N,ILO,IHI,A,LDA,TAU,WORK,LWORK,INFO)
```

To apply $Q$ to an $m$ by $n$ complex matrix $C$ this routine may be followed by a call to F08NUF (CUNMHR/ZUNMHR). For example,

```
      CALL CUNMHR ('Left','No Transpose',M,N,ILO,IHI,A,LDA,TAU,C,LDC,
     +             WORK,LWORK,INFO)
```

forms the matrix product $QC$.

The real analogue of this routine is F08NEF (SGEHRD/DGEHRD).

## 9. Example

To compute the upper Hessenberg form of the matrix $A$, where

$$A = \begin{pmatrix} -3.97 - 5.04i & -4.11 + 3.70i & -0.34 + 1.01i & 1.29 - 0.86i \\ 0.34 - 1.50i & 1.52 - 0.43i & 1.88 - 5.38i & 3.36 + 0.65i \\ 3.31 - 3.85i & 2.50 + 3.45i & 0.88 - 1.08i & 0.64 - 1.48i \\ -1.10 + 0.82i & 1.81 - 1.59i & 3.25 + 1.33i & 1.57 - 3.44i \end{pmatrix}.$$

### 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.**

```
*     F08NSF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, LWORK
      PARAMETER        (NMAX=8,LDA=NMAX,LWORK=64*NMAX)
      complex          ZERO
      PARAMETER        (ZERO=(0.0e0,0.0e0))
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
*     .. Local Arrays ..
      complex          A(LDA,NMAX), TAU(NMAX-1), WORK(LWORK)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         X04DBF, cgehrd
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08NSF 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)
*
*        Reduce A to upper Hessenberg form
*
         CALL cgehrd(N,1,N,A,LDA,TAU,WORK,LWORK,INFO)
*
*        Set the elements below the first sub-diagonal to zero
*
         DO 40 I = 1, N - 2
            DO 20 J = I + 2, N
               A(J,I) = ZERO
   20       CONTINUE
   40    CONTINUE
*
*        Print upper Hessenberg form
*
         WRITE (NOUT,*)
         IFAIL = 0
*
```

```
         CALL X04DBF('General',' ',N,N,A,LDA,'Bracketed','F7.4',
     +               'Upper Hessenberg form','Integer',RLABS,'Integer',
     +               CLABS,80,0,IFAIL)
*
      END IF
      STOP
      END
```

## 9.2. Program Data

```
F08NSF Example Program Data
  4                                                                 :Value of N
 (-3.97,-5.04) (-4.11, 3.70) (-0.34, 1.01) ( 1.29,-0.86)
 ( 0.34,-1.50) ( 1.52,-0.43) ( 1.88,-5.38) ( 3.36, 0.65)
 ( 3.31,-3.85) ( 2.50, 3.45) ( 0.88,-1.08) ( 0.64,-1.48)
 (-1.10, 0.82) ( 1.81,-1.59) ( 3.25, 1.33) ( 1.57,-3.44)    :End of matrix A
```

## 9.3. Program Results

```
F08NSF Example Program Results

Upper Hessenberg form
                     1                   2                   3                   4
1  (-3.9700,-5.0400) (-1.1318,-2.5693) (-4.6027,-0.1426) (-1.4249, 1.7330)
2  (-5.4797, 0.0000) ( 1.8585,-1.5502) ( 4.4145,-0.7638) (-0.4805,-1.1976)
3  ( 0.0000, 0.0000) ( 6.2673, 0.0000) (-0.4504,-0.0290) (-1.3467, 1.6579)
4  ( 0.0000, 0.0000) ( 0.0000, 0.0000) (-3.5000, 0.0000) ( 2.5619,-3.3708)
```

# F08NTF (CUNGHR/ZUNGHR) – 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

F08NTF (CUNGHR/ZUNGHR) generates the complex unitary matrix $Q$ which was determined by F08NSF (CGEHRD/ZGEHRD), when reducing a complex general matrix $A$ to Hessenberg form.

## 2. Specification

```
SUBROUTINE F08NTF (N, ILO, IHI, A, LDA, TAU, WORK, LWORK, INFO)
ENTRY      cunghr (N, ILO, IHI, A, LDA, TAU, WORK, LWORK, INFO)
INTEGER        N, ILO, IHI, 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 following a call to F08NSF (CGEHRD/ZGEHRD), which reduces a complex general matrix $A$ to upper Hessenberg form $H$ by a unitary similarity transformation: $A = QHQ^H$. F08NSF represents the matrix $Q$ as a product of $i_{hi}-i_{lo}$ elementary reflectors. Here $i_{lo}$ and $i_{hi}$ are values determined by F08NVF (CGEBAL/ZGEBAL) when balancing the matrix; if the matrix has not been balanced, $i_{lo} = 1$ and $i_{hi} = n$.

This routine may be used to generate $Q$ explicitly as a square matrix. $Q$ has the structure:

$$Q = \begin{pmatrix} I & 0 & 0 \\ 0 & Q_{22} & 0 \\ 0 & 0 & I \end{pmatrix}$$

where $Q_{22}$ occupies rows and columns $i_{lo}$ to $i_{hi}$.

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §7.4.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

## 5. Parameters

1: N – INTEGER. *Input*

*On entry*: $n$, the order of the matrix $Q$.

*Constraint*: N ≥ 0.

2: ILO – INTEGER. *Input*

3: IHI – INTEGER. *Input*

*On entry*: these **must** be the same parameters ILO and IHI, respectively, as supplied to F08NSF (CGEHRD/ZGEHRD).

*Constraints*: 1 ≤ ILO ≤ IHI ≤ N if N > 0,
ILO = 1 and IHI = 0 if N = 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 F08NSF (CGEHRD/ZGEHRD).

*On exit*: the $n$ by $n$ unitary matrix $Q$.

5: LDA – INTEGER. *Input*

*On entry*: the first dimension of the array A as declared in the (sub)program from which F08NTF (CUNGHR/ZUNGHR) is called.

*Constraint*: LDA ≥ max(1,N).

6: TAU(*) – ***complex*** array. *Input*

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

*On entry*: further details of the elementary reflectors, as returned by F08NSF (CGEHRD/ZGEHRD).

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 F08NTF (CUNGHR/ZUNGHR) is called.

*Suggested value*: for optimum performance LWORK should be at least (IHI–ILO)×*nb*, where *nb* is the ***blocksize***.

*Constraint*: LWORK ≥ max(1,IHI–ILO).

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 $i$th 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 $\varepsilon$ is the ***machine precision***.

## 8. Further Comments

The total number of real floating-point operations is approximately $\frac{16q^3}{3}$, where $q = i_{hi} - i_{lo}$.

The real analogue of this routine is F08NFF (SORGHR/DORGHR).

## 9. Example

To compute the Schur factorization of the matrix $A$, where

$$A = \begin{pmatrix} -3.97 - 5.04i & -4.11 + 3.70i & -0.34 + 1.01i & 1.29 - 0.86i \\ 0.34 - 1.50i & 1.52 - 0.43i & 1.88 - 5.38i & 3.36 + 0.65i \\ 3.31 - 3.85i & 2.50 + 3.45i & 0.88 - 1.08i & 0.64 - 1.48i \\ -1.10 + 0.82i & 1.81 - 1.59i & 3.25 + 1.33i & 1.57 - 3.44i \end{pmatrix}.$$

Here $A$ is general and must first be reduced to Hessenberg form by F08NSF (CGEHRD/ZGEHRD). The program then calls F08NTF (CUNGHR/ZUNGHR) to form $Q$, and passes this matrix to F08PSF (CHSEQR/ZHSEQR) which computes the Schur factorization 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.**

```
*     F08NTF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, LDZ, LWORK
      PARAMETER        (NMAX=8,LDA=NMAX,LDZ=NMAX,LWORK=64*(NMAX-1))
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
*     .. Local Arrays ..
      complex          A(LDA,NMAX), TAU(NMAX), W(NMAX), WORK(LWORK),
     +                 Z(LDZ,NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         F06TFF, X04DBF, cgehrd, chseqr, cunghr
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08NTF 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)
*
*        Reduce A to upper Hessenberg form H = (Q**H)*A*Q
*
         CALL cgehrd(N,1,N,A,LDA,TAU,WORK,LWORK,INFO)
*
*        Copy A into Z
*
         CALL F06TFF('General',N,N,A,LDA,Z,LDZ)
*
*        Form Q explicitly, storing the result in Z
*
         CALL cunghr(N,1,N,Z,LDZ,TAU,WORK,LWORK,INFO)
*
*        Calculate the Schur factorization of H = Y*T*(Y**H) and form
*        Q*Y explicitly, storing the result in Z
*
*        Note that A = Z*T*(Z**H), where Z = Q*Y
*
         CALL chseqr('Schur form','Vectors',N,1,N,A,LDA,W,Z,LDZ,WORK,
     +               LWORK,INFO)
*
*        Print Schur form
*
         WRITE (NOUT,*)
         IFAIL = 0
*
         CALL X04DBF('General',' ',N,N,A,LDA,'Bracketed','F7.4',
     +               'Schur form','Integer',RLABS,'Integer',CLABS,80,0,
     +               IFAIL)
*
*        Print Schur vectors
*
         WRITE (NOUT,*)
         IFAIL = 0
*
         CALL X04DBF('General',' ',N,N,Z,LDZ,'Bracketed','F7.4',
     +               'Schur vectors of A','Integer',RLABS,'Integer',
     +               CLABS,80,0,IFAIL)
*
```

```
      END IF
      STOP
*
      END
```

## 9.2. Program Data

```
F08NTF Example Program Data
  4                                                                  :Value of N
 (-3.97,-5.04) (-4.11, 3.70) (-0.34, 1.01) ( 1.29,-0.86)
 ( 0.34,-1.50) ( 1.52,-0.43) ( 1.88,-5.38) ( 3.36, 0.65)
 ( 3.31,-3.85) ( 2.50, 3.45) ( 0.88,-1.08) ( 0.64,-1.48)
 (-1.10, 0.82) ( 1.81,-1.59) ( 3.25, 1.33) ( 1.57,-3.44)   :End of matrix A
```

## 9.3. Program Results

```
F08NTF Example Program Results

Schur form
                     1                     2                     3                     4
1  (-6.0004,-6.9998) (-0.4701,-0.2119) ( 0.0438, 0.5124) (-0.9097,-0.0925)
2  ( 0.0000, 0.0000) (-5.0000, 2.0060) ( 0.7150,-0.1028) (-0.0580, 0.2575)
3  ( 0.0000, 0.0000) ( 0.0000, 0.0000) ( 7.9982,-0.9964) (-0.2232,-1.0549)
4  ( 0.0000, 0.0000) ( 0.0000, 0.0000) ( 0.0000, 0.0000) ( 3.0023,-3.9998)

Schur vectors of A
                     1                     2                     3                     4
1  ( 0.8457, 0.0000) (-0.3613, 0.1351) (-0.1755, 0.2297) ( 0.1099,-0.2007)
2  (-0.0177, 0.3036) (-0.3366, 0.4660) ( 0.7228, 0.0000) ( 0.0336, 0.2312)
3  ( 0.0875, 0.3115) ( 0.6311, 0.0000) ( 0.2871, 0.4999) ( 0.0944,-0.3947)
4  (-0.0561,-0.2906) (-0.1045,-0.3339) ( 0.2476, 0.0195) ( 0.8534, 0.0000)
```

# F08NUF (CUNMHR/ZUNMHR) – 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

F08NUF (CUNMHR/ZUNMHR) multiplies an arbitrary complex matrix $C$ by the complex unitary matrix $Q$ which was determined by F08NSF (CGEHRD/ZGEHRD) when reducing a complex general matrix to Hessenberg form.

## 2. Specification

```
 SUBROUTINE F08NUF (SIDE, TRANS, M, N, ILO, IHI, A, LDA, TAU, C, LDC,
1                   WORK, LWORK, INFO)
 ENTRY       cunmhr (SIDE, TRANS, M, N, ILO, IHI, A, LDA, TAU, C, LDC,
1                   WORK, LWORK, INFO)
 INTEGER          M, N, ILO, IHI, 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 following a call to F08NSF (CGEHRD/ZGEHRD), which reduces a complex general matrix $A$ to upper Hessenberg form $H$ by a unitary similarity transformation: $A = QHQ^H$. F08NSF represents the matrix $Q$ as a product of $i_{hi}-i_{lo}$ elementary reflectors. Here $i_{lo}$ and $i_{hi}$ are values determined by F08NVF (CGEBAL/ZGEBAL) when balancing the matrix; if the matrix has not been balanced, $i_{lo} = 1$ and $i_{hi} = n$.

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 to transform a matrix $V$ of eigenvectors of $H$ to the matrix $QV$ of eigenvectors of $A$.

## 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 $C$; $m$ is also the order of $Q$ if SIDE = 'L'.

*Constraint*: M ≥ 0.

4: N – INTEGER. *Input*

*On entry*: $n$, the number of columns of the matrix $C$; $n$ is also the order of $Q$ if SIDE = 'R'.

*Constraint*: N ≥ 0.

5: ILO – INTEGER. *Input*

6: IHI – INTEGER. *Input*

*On entry*: these **must** be the same parameters ILO and IHI, respectively, as supplied to F08NSF (CGEHRD/ZGEHRD).

*Constraints*: 1 ≤ ILO ≤ IHI ≤ M if SIDE = 'L' and M > 0;
ILO = 1 and IHI = 0 if SIDE = 'L' and M = 0;
1 ≤ ILO ≤ IHI ≤ N if SIDE = 'R' and N > 0;
ILO = 1 and IHI = 0 if SIDE = 'R' and N = 0.

7: 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 F08NSF (CGEHRD/ZGEHRD).

8: LDA – INTEGER. *Input*

*On entry*: the first dimension of the array A as declared in the (sub)program from which F08NUF (CUNMHR/ZUNMHR) is called.

*Constraints*: LDA ≥ max(1,M) if SIDE = 'L',
LDA ≥ max(1,N) if SIDE = 'R'.

9: TAU(*) – ***complex*** array. *Input*

**Note**: the dimension of the array TAU must be at least max(1,M–1) if SIDE = 'L' and at least max(1,N–1) if SIDE = 'R'.

*On entry*: further details of the elementary reflectors, as returned by F08NSF (CGEHRD/ZGEHRD).

10: 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.

11: LDC – INTEGER. *Input*

*On entry*: the first dimension of the array C as declared in the (sub)program from which F08NUF (CUNMHR/ZUNMHR) is called.

*Constraint*: LDC ≥ max(1,M).

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

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

13: LWORK – INTEGER. *Input*

*On entry*: the dimension of the array WORK as declared in the (sub)program from which F08NUF (CUNMHR/ZUNMHR) 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'.

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 $i$th 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 $\varepsilon$ is the ***machine precision***.

## 8. Further Comments

The total number of real floating-point operations is approximately $8nq^2$ if SIDE = 'L' and $8mq^2$ if SIDE = 'R', where $q = i_{hi} - i_{lo}$.

The real analogue of this routine is F08NGF (SORMHR/DORMHR).

## 9. Example

To compute all the eigenvalues of the matrix $A$, where

$$A = \begin{pmatrix} -3.97 - 5.04i & -4.11 + 3.70i & -0.34 + 1.01i & 1.29 - 0.86i \\ 0.34 - 1.50i & 1.52 - 0.43i & 1.88 - 5.38i & 3.36 + 0.65i \\ 3.31 - 3.85i & 2.50 + 3.45i & 0.88 - 1.08i & 0.64 - 1.48i \\ -1.10 + 0.82i & 1.81 - 1.59i & 3.25 + 1.33i & 1.57 - 3.44i \end{pmatrix},$$

and those eigenvectors which correspond to eigenvalues $\lambda$ such that $Re(\lambda) < 0$. Here $A$ is general and must first be reduced to upper Hessenberg form $H$ by F08NSF (CGEHRD/ZGEHRD). The program then calls F08PSF (CHSEQR/ZHSEQR) to compute the eigenvalues, and F08PXF (CHSEIN/ZHSEIN) to compute the required eigenvectors of $H$ by inverse iteration. Finally F08NUF (CUNMHR/ZUNMHR) is called to transform the eigenvectors of $H$ back to eigenvectors of the original matrix $A$.

### 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.

```
*     F08NUF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, LDH, LDZ, LWORK, LDVL, LDVR
      PARAMETER        (NMAX=8,LDA=NMAX,LDH=NMAX,LDZ=1,LWORK=64*NMAX,
     +                 LDVL=NMAX,LDVR=NMAX)
*     .. Local Scalars ..
      real             THRESH
      INTEGER          I, IFAIL, INFO, J, M, N
```

```
*     .. Local Arrays ..
      complex           A(LDA,NMAX), H(LDH,NMAX), TAU(NMAX),
     +                  VL(LDVL,NMAX), VR(LDVR,NMAX), W(NMAX),
     +                  WORK(LWORK), Z(LDZ,1)
      real              RWORK(NMAX)
      INTEGER           IFAILL(NMAX), IFAILR(NMAX)
      LOGICAL           SELECT(NMAX)
      CHARACTER         CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL          F06TFF, X04DBF, cgehrd, chsein, chseqr, cunmhr
*     .. Intrinsic Functions ..
      INTRINSIC         real, imag
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08NUF 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)
*
         READ (NIN,*) THRESH
*
*        Reduce A to upper Hessenberg form H = (Q**H)*A*Q
*
         CALL cgehrd(N,1,N,A,LDA,TAU,WORK,LWORK,INFO)
*
*        Copy A to H
*
         CALL F06TFF('General',N,N,A,LDA,H,LDH)
*
*        Calculate the eigenvalues of H (same as A)
*
         CALL chseqr('Eigenvalues','No vectors',N,1,N,H,LDH,W,Z,LDZ,
     +               WORK,LWORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.GT.0) THEN
            WRITE (NOUT,*) 'Failure to converge.'
         ELSE
            WRITE (NOUT,*) 'Eigenvalues'
            WRITE (NOUT,99999) (' (',real(W(I)),',',imag(W(I)),')',I=1,
     +        N)
*
            DO 20 I = 1, N
               SELECT(I) = real(W(I)) .LT. THRESH
   20       CONTINUE
*
*           Calculate the eigenvectors of H (as specified by SELECT),
*           storing the result in VR
*
            CALL chsein('Right','QR','No initial vectors',SELECT,N,A,
     +                  LDA,W,VL,LDVL,VR,LDVR,N,M,WORK,RWORK,IFAILL,
     +                  IFAILR,INFO)
*
*           Calculate the eigenvectors of A = Q * (eigenvectors of H)
*
            CALL cunmhr('Left','No transpose',N,M,1,N,A,LDA,TAU,VR,LDVR,
     +                  WORK,LWORK,INFO)
*
*           Print eigenvectors
*
            WRITE (NOUT,*)
            IFAIL = 0
*
            CALL X04DBF('General',' ',N,M,VR,LDVR,'Bracketed','F7.4',
     +                  'Contents of array VR','Integer',RLABS,
     +                  'Integer',CLABS,80,0,IFAIL)
```

```
*
         END IF
      END IF
      STOP
*
99999 FORMAT ((3X,4(A,F7.4,A,F7.4,A,:)))
      END
```

## 9.2. Program Data

```
F08NUF Example Program Data
  4                                                                  :Value of N
 (-3.97,-5.04) (-4.11, 3.70) (-0.34, 1.01) ( 1.29,-0.86)
 ( 0.34,-1.50) ( 1.52,-0.43) ( 1.88,-5.38) ( 3.36, 0.65)
 ( 3.31,-3.85) ( 2.50, 3.45) ( 0.88,-1.08) ( 0.64,-1.48)
 (-1.10, 0.82) ( 1.81,-1.59) ( 3.25, 1.33) ( 1.57,-3.44)           :End of matrix A
  0.0                                                               :Value of THRESH
```

## 9.3. Program Results

```
 F08NUF Example Program Results

 Eigenvalues
    (-6.0004,-6.9998) (-5.0000, 2.0060) ( 7.9982,-0.9964) ( 3.0023,-3.9998)

 Contents of array VR
                      1                   2
 1   ( 1.0000, 0.0000) ( 0.2613, 0.5284)
 2   (-0.0210, 0.3590) ( 0.6485, 0.4683)
 3   ( 0.1035, 0.3683) (-0.0323,-0.8516)
 4   (-0.0664,-0.3436) (-0.4521, 0.1368)
```

# F08NVF (CGEBAL/ZGEBAL) – 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

F08NVF (CGEBAL/ZGEBAL) balances a complex general matrix in order to improve the accuracy of computed eigenvalues and/or eigenvectors.

## 2. Specification

```
SUBROUTINE F08NVF (JOB, N, A, LDA, ILO, IHI, SCALE, INFO)
ENTRY      cgebal  (JOB, N, A, LDA, ILO, IHI, SCALE, INFO)
INTEGER          N, LDA, ILO, IHI, INFO
real             SCALE(*)
complex          A(LDA,*)
CHARACTER*1      JOB
```

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

## 3. Description

This routine balances a complex general matrix $A$. The term 'balancing' covers two steps, each of which involves a similarity transformation of $A$. The routine can perform either or both of these steps.

1. The routine first attempts to permute $A$ to block upper triangular form by a similarity transformation:

$$PAP^T = A' = \begin{pmatrix} A'_{11} & A'_{12} & A'_{13} \\ 0 & A'_{22} & A'_{23} \\ 0 & 0 & A'_{33} \end{pmatrix}$$

where $P$ is a permutation matrix, and $A'_{11}$ and $A'_{33}$ are upper triangular. Then the diagonal elements of $A'_{11}$ and $A'_{33}$ are eigenvalues of $A$. The rest of the eigenvalues of $A$ are the eigenvalues of the central diagonal block $A'_{22}$, in rows and columns $i_{lo}$ to $i_{hi}$. Subsequent operations to compute the eigenvalues of $A$ (or its Schur factorization) need only be applied to these rows and columns; this can save a significant amount of work if $i_{lo} > 1$ and $i_{hi} < n$. If no suitable permutation exists (as is often the case), the routine sets $i_{lo} = 1$ and $i_{hi} = n$, and $A'_{22}$ is the whole of $A$.

2. The routine applies a diagonal similarity transformation to $A'$, to make the rows and columns of $A'_{22}$ as close in norm as possible:

$$A'' = DA'D^{-1} = \begin{pmatrix} I & 0 & 0 \\ 0 & D_{22} & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} A'_{11} & A'_{12} & A'_{13} \\ 0 & A'_{22} & A'_{23} \\ 0 & 0 & A'_{33} \end{pmatrix} \begin{pmatrix} I & 0 & 0 \\ 0 & D_{22}^{-1} & 0 \\ 0 & 0 & I \end{pmatrix}.$$

This scaling can reduce the norm of the matrix (that is, $\|A''_{22}\| < \|A'_{22}\|$), and hence reduce the effect of rounding errors on the accuracy of computed eigenvalues and eigenvectors.

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §7.5.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

## 5. Parameters

1: JOB – CHARACTER*1. *Input*

*On entry*: indicates whether $A$ is to be permuted to and/or scaled (or neither), as follows:

if JOB = 'N', then $A$ is neither permuted nor scaled (but values are assigned to ILO, IHI and SCALE);

if JOB = 'P', then $A$ is permuted but not scaled;

if JOB = 'S', then $A$ is scaled but not permuted;

if JOB = 'B', then $A$ is both permuted and scaled.

*Constraint*: JOB = 'N', 'P', 'S' or 'B'.

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 $n$ general matrix $A$.

*On exit*: $A$ is overwritten by the balanced matrix.

A is not referenced if JOB = 'N'.

4: LDA – INTEGER. *Input*

*On entry*: the first dimension of the array A as declared in the (sub)program from which F08NVF (CGEBAL/ZGEBAL) is called.

*Constraint*: LDA ≥ max(1,N).

5: ILO – INTEGER. *Output*

6: IHI – INTEGER. *Output*

*On exit*: the values $i_{lo}$ and $i_{hi}$ such that on exit A($i$,$j$) is zero if $i > j$ and $1 \le j < i_{lo}$ or $i_{hi} < i \le n$.

If JOB = 'N' or 'S', then $i_{lo} = 1$ and $i_{hi} = n$.

7: SCALE(*) – ***real*** array. *Output*

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

*On exit*: details of the permutations and scaling factors applied to $A$. More precisely, if $p_j$ is the index of the row and column interchanged with row and column $j$ and $d_j$ is the scaling factor used to balance row and column $j$, then

$$\text{SCALE}(j) = \begin{cases} p_j, & \text{for } j = 1,2,\ldots,i_{lo}-1 \\ d_j, & \text{for } j = i_{lo},i_{lo}+1,\ldots,i_{hi} \text{ and} \\ p_j, & \text{for } j = i_{hi}+1,i_{hi}+2,\ldots,n. \end{cases}$$

The order in which the interchanges are made is $n$ to $i_{hi}+1$, then 1 to $i_{lo}-1$.

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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

## 7. Accuracy

The errors are negligible, compared with those in subsequent computations.

## 8. Further Comments

If the matrix $A$ is balanced by this routine, then any eigenvectors computed subsequently are eigenvectors of the matrix $A''$ (see Section 3) and hence F08NWF (CGEBAK/ZGEBAK) **must** then be called to transform them back to eigenvectors of $A$.

If the Schur vectors of $A$ are required, then this routine must **not** be called with JOB = 'S' or 'B', because then the balancing transformation is not unitary. If this routine is called with JOB = 'P', then any Schur vectors computed subsequently are Schur vectors of the matrix $A''$, and F08NWF **must** be called (with SIDE = 'R') to transform them back to Schur vectors of $A$.

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

The real analogue of this routine is F08NHF (SGEBAL/DGEBAL).

## 9. Example

To compute all the eigenvalues and right eigenvectors of the matrix $A$, where

$$A = \begin{pmatrix} 1.50 - 2.75i & 0.00 + 0.00i & 0.00 + 0.00i & 0.00 + 0.00i \\ -8.06 - 1.24i & -2.50 - 0.50i & 0.00 + 0.00i & -0.75 + 0.50i \\ -2.09 + 7.56i & 1.39 + 3.97i & -1.25 + 0.75i & -4.82 - 5.67i \\ 6.18 + 9.79i & -0.92 - 0.62i & 0.00 + 0.00i & -2.50 - 0.50i \end{pmatrix}.$$

The program first calls F08NVF (CGEBAL/ZGEBAL) to balance the matrix; it then computes the Schur factorization of the balanced matrix, by reduction to Hessenberg form and the $QR$ algorithm. Then it calls F08QXF (CTREVC/ZTREVC) to compute the right eigenvectors of the balanced matrix, and finally calls F08NWF (CGEBAK/ZGEBAK) to transform the eigenvectors back to eigenvectors of the original matrix $A$.

### 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.

```
*     F08NVF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, LDH, LWORK, LDVL, LDVR
      PARAMETER        (NMAX=8,LDA=NMAX,LDH=NMAX,LWORK=64*NMAX,LDVL=1,
     +                 LDVR=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, IHI, ILO, INFO, J, M, N
*     .. Local Arrays ..
      complex          A(LDA,NMAX), H(LDH,NMAX), TAU(NMAX), VL(LDVL, 1),
     +                 VR(LDVR,NMAX), W(NMAX), WORK(LWORK)
      real             RWORK(NMAX), SCALE(NMAX)
      LOGICAL          SELECT(1)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         F06TFF, X04DBF, cgebak, cgebal, cgehrd, chseqr,
     +                 ctrevc, cunghr
*     .. Intrinsic Functions ..
      INTRINSIC        real, imag
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08NVF 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)
*
*     Balance A
*
      CALL cgebal('Both',N,A,LDA,ILO,IHI,SCALE,INFO)
*
*     Reduce A to upper Hessenberg form H = (Q**H)*A*Q
*
      CALL cgehrd(N,ILO,IHI,A,LDA,TAU,WORK,LWORK,INFO)
*
*     Copy A to H
*
      CALL F06TFF('General',N,N,A,LDA,H,LDH)
*
*     Copy A into VR
*
      CALL F06TFF('General',N,N,A,LDA,VR,LDVR)
*
*     Form Q explicitly, storing the result in VR
*
      CALL cunghr(N,1,N,VR,LDVR,TAU,WORK,LWORK,INFO)
*
*     Calculate the eigenvalues and Schur factorization of A
*
      CALL chseqr('Schur form','Vectors',N,ILO,IHI,H,LDH,W,VR,LDVR,
     +            WORK,LWORK,INFO)
*
      WRITE (NOUT,*)
      IF (INFO.GT.0) THEN
         WRITE (NOUT,*) 'Failure to converge.'
      ELSE
         WRITE (NOUT,*) 'Eigenvalues'
         WRITE (NOUT,99999) (' (',real(W(I)),',',imag(W(I)),')',I=1,
     +     N)
*
*        Calculate the eigenvectors of A, storing the result in VR
*
         CALL ctrevc('Right','Overwrite',SELECT,N,H,LDH,VL,LDVL,VR,
     +               LDVR,N,M,WORK,RWORK,INFO)
*
*        Backtransform eigenvectors
*
         CALL cgebak('Both','Right',N,ILO,IHI,SCALE,M,VR,LDVR, INFO)
*
*        Print eigenvectors
*
         WRITE (NOUT,*)
         IFAIL = 0
*
         CALL X04DBF('General',' ',N,M,VR,LDVR,'Bracketed','F7.4',
     +               'Contents of array VR','Integer',RLABS,
     +               'Integer',CLABS,80,0,IFAIL)
*
      END IF
    END IF
    STOP
*
99999 FORMAT ((3X,4(A,F7.4,A,F7.4,A,:)))
      END
```

## 9.2. Program Data

```
F08NVF Example Program Data
  4                                                              :Value of N
 ( 1.50,-2.75) ( 0.00, 0.00) ( 0.00, 0.00) ( 0.00, 0.00)
 (-8.06,-1.24) (-2.50,-0.50) ( 0.00, 0.00) (-0.75, 0.50)
 (-2.09, 7.56) ( 1.39, 3.97) (-1.25, 0.75) (-4.82,-5.67)
 ( 6.18, 9.79) (-0.92,-0.62) ( 0.00, 0.00) (-2.50,-0.50)    :End of matrix A
```

## 9.3. Program Results

```
F08NVF Example Program Results

Eigenvalues
   (-1.2500, 0.7500) (-1.5000,-0.4975) (-3.5000,-0.5025) ( 1.5000,-2.7500)

Contents of array VR
                     1                   2                   3                   4
1  ( 0.0000, 0.0000) ( 0.0000, 0.0000) ( 0.0000, 0.0000) ( 0.1452, 0.0000)
2  ( 0.0000, 0.0000) (-0.0616, 0.0413) ( 0.4613, 0.0000) (-0.2072,-0.2450)
3  ( 1.0000, 0.0000) ( 0.6032,-0.3968) ( 0.2983, 0.7017) ( 0.7768, 0.2232)
4  ( 0.0000, 0.0000) ( 0.0822, 0.0000) ( 0.4251, 0.2850) (-0.0119, 0.4372)
```

# F08NWF (CGEBAK/ZGEBAK) – 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

F08NWF (CGEBAK/ZGEBAK) transforms eigenvectors of a balanced matrix to those of the original complex general matrix.

## 2. Specification

```
SUBROUTINE F08NWF (JOB, SIDE, N, ILO, IHI, SCALE, M, V, LDV, INFO)
ENTRY      cgebak (JOB, SIDE, N, ILO, IHI, SCALE, M, V, LDV, INFO)
INTEGER          N, ILO, IHI, M, LDV, INFO
real             SCALE(*)
complex          V(LDV,*)
CHARACTER*1      JOB, SIDE
```

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

## 3. Description

This routine is intended to be used after a complex general matrix $A$ has been balanced by F08NVF (CGEBAL/ZGEBAL), and eigenvectors of the balanced matrix $A''_{22}$ have subsequently been computed.

For a description of balancing, see the document for F08NVF. The balanced matrix $A''$ is obtained as $A'' = DPAP^TD^{-1}$, where $P$ is a permutation matrix and $D$ is a diagonal scaling matrix. This routine transforms left or right eigenvectors as follows:

– if $x$ is a right eigenvector of $A''$,

$P^TD^{-1}x$ is a right eigenvector of $A$;

– if $y$ is a left eigenvector of $A''$,

$P^TDy$ is a left eigenvector of $A$.

## 4. References

None.

## 5. Parameters

1: JOB – CHARACTER*1. *Input*

*On entry*: this **must** be the same parameter JOB as supplied to F08NVF (CGEBAL/ZGEBAL).

*Constraint*: JOB = 'N', 'P', 'S' or 'B'.

2: SIDE – CHARACTER*1. *Input*

*On entry*: indicates whether left or right eigenvectors are to be transformed, as follows:

if SIDE = 'L', then left eigenvectors are transformed;

if SIDE = 'R', then right eigenvectors are transformed.

*Constraint*: SIDE = 'L' or 'R'.

3: N – INTEGER. *Input*

*On entry*: $n$, the number of rows of the matrix of eigenvectors.

*Constraint*: N $\geq$ 0.

4: ILO – INTEGER. *Input*

5: IHI – INTEGER. *Input*

*On entry*: the values $i_{lo}$ and $i_{hi}$, as returned by F08NVF (CGEBAL/ZGEBAL).

*Constraints*: $1 \leq \text{ILO} \leq \text{IHI} \leq \text{N}$ if $\text{N} > 0$,
ILO = 1 and IHI = 0 if N = 0.

6: SCALE(*) – ***real*** array. *Input*

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

*On entry*: details of the permutations and/or the scaling factors used to balance the original complex general matrix, as returned by F08NVF (CGEBAL/ZGEBAL).

7: M – INTEGER. *Input*

*On entry*: $m$, the number of columns of the matrix of eigenvectors.

*Constraint*: $\text{M} \geq 0$.

8: V(LDV,*) – ***complex*** array. *Input/Output*

**Note**: the second dimension of the array V must be at least max(1,M).

*On entry*: the matrix of left or right eigenvectors to be transformed.

*On exit*: the transformed eigenvectors.

9: LDV – INTEGER. *Input*

*On entry*: the first dimension of the array V as declared in the (sub)program from which F08NWF (CGEBAK/ZGEBAK) is called.

*Constraint*: $\text{LDV} \geq \max(1,\text{N})$.

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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

## 7. Accuracy

The errors are negligible.

## 8. Further Comments

The total number of real floating-point operations is approximately proportional to $nm$.

The real analogue of this routine is F08NJF (SGEBAK/DGEBAK).

## 9. Example

See the example for F08NVF (CGEBAL/ZGEBAL).

# F08PEF (SHSEQR/DHSEQR) – 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

F08PEF (SHSEQR/DHSEQR) computes all the eigenvalues, and optionally the Schur factorization, of a real Hessenberg matrix or a real general matrix which has been reduced to Hessenberg form.

## 2. Specification

```
 SUBROUTINE F08PEF (JOB, COMPZ, N, ILO, IHI, H, LDH, WR, WI, Z, LDZ, WORK,
1                   LWORK, INFO)
 ENTRY       shseqr (JOB, COMPZ, N, ILO, IHI, H, LDH, WR, WI, Z, LDZ, WORK,
1                   LWORK, INFO)
 INTEGER          N, ILO, IHI, LDH, LDZ, LWORK, INFO
 real             H(LDH,*), WR(*), WI(*), Z(LDZ,*), WORK(*)
 CHARACTER*1      JOB, COMPZ
```

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

## 3. Description

This routine computes all the eigenvalues, and optionally the Schur factorization, of a real upper Hessenberg matrix $H$:

$$H = ZTZ^T,$$

where $T$ is an upper quasi-triangular matrix (the Schur form of $H$), and $Z$ is the orthogonal matrix whose columns are the Schur vectors $z_i$. See Section 8 for details of the structure of $T$.

The routine may also be used to compute the Schur factorization of a real general matrix $A$ which has been reduced to upper Hessenberg form $H$:

$$\begin{aligned} A &= QHQ^T, \text{ where } Q \text{ is orthogonal,} \\ &= (QZ)T(QZ)^T. \end{aligned}$$

In this case, after F08NEF (SGEHRD/DGEHRD) has been called to reduce $A$ to Hessenberg form, F08NFF (SORGHR/DORGHR) must be called to form $Q$ explicitly; $Q$ is then passed to F08PEF, which must be called with COMPZ = 'V'.

The routine can also take advantage of a previous call to F08NHF (SGEBAL/DGEBAL) which may have balanced the original matrix before reducing it to Hessenberg form, so that the Hessenberg matrix $H$ has the structure:

$$\begin{pmatrix} H_{11} & H_{12} & H_{13} \\ & H_{22} & H_{23} \\ & & H_{33} \end{pmatrix}$$

where $H_{11}$ and $H_{33}$ are upper triangular. If so, only the central diagonal block $H_{22}$ (in rows and columns $i_{lo}$ to $i_{hi}$) needs to be further reduced to Schur form (the blocks $H_{12}$ and $H_{23}$ are also affected). Therefore the values of $i_{lo}$ and $i_{hi}$ can be supplied to F08PEF directly. Also, F08NJF (SGEBAK/DGEBAK) must be called after this routine to permute the Schur vectors of the balanced matrix to those of the original matrix. If F08NHF has not been called however, then $i_{lo}$ must be set to 1 and $i_{hi}$ to $n$. Note that if the Schur factorization of $A$ is required, F08NHF must not be called with JOB = 'S' or 'B', because the balancing transformation is not orthogonal.

F08PEF uses a multishift form of the upper Hessenberg $QR$ algorithm, due to Bai and Demmel [1]. The Schur vectors are normalized so that $\|z_i\|_2 = 1$, but are determined only to within a factor $\pm 1$.

## 4. References

[1] BAI, Z. and DEMMEL, J.W.
On a Block Implementation of Hessenberg Multishift *QR* Iteration.
Int. J. High Speed Comp., 1, pp. 97-112, 1989.

[2] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §7.5.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

## 5. Parameters

1: JOB – CHARACTER*1. *Input*

*On entry*: indicates whether eigenvalues only or the Schur form $T$ is required, as follows:

if JOB = 'E', then eigenvalues only are required;

if JOB = 'S', then the Schur form $T$ is required.

*Constraint*: JOB = 'E' or 'S'.

2: COMPZ – CHARACTER*1. *Input*

*On entry*: indicates whether the Schur vectors are to be computed as follows:

if COMPZ = 'N', then no Schur vectors are computed (and the array Z is not referenced);

if COMPZ = 'I', then the Schur vectors of $H$ are computed (and the array Z is initialized by the routine);

if COMPZ = 'V', then the Schur vectors of $A$ are computed (and the array Z must contain the matrix $Q$ on entry).

*Constraint*: COMPZ = 'N', 'I' or 'V'.

3: N – INTEGER. *Input*

*On entry*: $n$, the order of the matrix $H$.

*Constraint*: N ≥ 0.

4: ILO – INTEGER. *Input*

5: IHI – INTEGER. *Input*

*On entry*: if the matrix $A$ has been balanced by F08NHF (SGEBAL/DGEBAL), then ILO and IHI must contain the values returned by that routine. Otherwise, ILO must be set to 1 and IHI to N.

*Constraints*: ILO ≥ 1 and min(ILO,N) ≤ IHI ≤ N.

6: H(LDH,*) – ***real*** array. *Input/Output*

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

*On entry*: the $n$ by $n$ upper Hessenberg matrix $H$, as returned by F08NEF (SGEHRD/DGEHRD).

*On exit*: if JOB = 'E', then the array contains no useful information. If JOB = 'S', then $H$ is overwritten by the upper quasi-triangular matrix $T$ from the Schur decomposition (the Schur form) unless INFO > 0.

7: LDH – INTEGER. *Input*

*On entry*: the first dimension of the array H as declared in the (sub)program from which F08PEF (SHSEQR/DHSEQR) is called.

*Constraint*: LDH ≥ max(1,N).

8: WR(*) – ***real*** array. *Output*

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

9: WI(*) – ***real*** array. *Output*

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

*On exit*: the real and imaginary parts, respectively, of the computed eigenvalues, unless INFO > 0 (in which case see Section 6). Complex conjugate pairs of eigenvalues appear consecutively with the eigenvalue having positive imaginary part first. The eigenvalues are stored in the same order as on the diagonal of the Schur form $T$ (if computed); see Section 8 for details.

10: Z(LDZ,*) – ***real*** array. *Input/Output*

**Note**: the second dimension of the array Z must be at least max(1,N) if COMPZ = 'V' or 'I' and at least 1 if COMPZ = 'N'.

*On entry*: if COMPZ = 'V', Z must contain the orthogonal matrix $Q$ from the reduction to Hessenberg form; if COMPZ = 'I', Z need not be set.

*On exit*: if COMPZ = 'V' or 'I', Z contains the orthogonal matrix of the required Schur vectors, unless INFO > 0.

Z is not referenced if COMPZ = 'N'.

11: LDZ – INTEGER. *Input*

*On entry*: the first dimension of the array Z as declared in the (sub)program from which F08PEF (SHSEQR/DHSEQR) is called.

*Constraints*: LDZ ≥ 1 if COMPZ = 'N',
LDZ ≥ max(1,N) if COMPZ = 'V' or 'I'.

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

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

13: LWORK – INTEGER. *Dummy*

This parameter is currently redundant.

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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

The algorithm has failed to find all the eigenvalues after a total of 30×(IHI–ILO+1) iterations. If INFO = $i$, elements 1,2,...,ILO–1 and $i$+1,$i$+2,...,$n$ of WR and WI contain the real and imaginary parts of the eigenvalues which have been found.

## 7. Accuracy

The computed Schur factorization is the exact factorization of a nearby matrix $H + E$, where

$$\|E\|_2 = O(\varepsilon)\|H\|_2,$$

and $\varepsilon$ is the ***machine precision***.

If $\lambda_i$ is an exact eigenvalue, and $\tilde{\lambda}_i$ is the corresponding computed value, then

$$|\tilde{\lambda}_i - \lambda_i| \leq \frac{c(n)\varepsilon\|H\|_2}{s_i},$$

where $c(n)$ is a modestly increasing function of $n$, and $s_i$ is the reciprocal condition number of $\lambda_i$. The condition numbers $s_i$ may be computed by calling F08QLF (STRSNA/DTRSNA).

## 8. Further Comments

The total number of floating-point operations depends on how rapidly the algorithm converges, but is typically about:

$7n^3$ if only eigenvalues are computed;
$10n^3$ if the Schur form is computed;
$20n^3$ if the full Schur factorization is computed.

The Schur form $T$ has the following structure (referred to as **canonical** Schur form).

If all the computed eigenvalues are real, $T$ is upper triangular, and the diagonal elements of $T$ are the eigenvalues; WR($i$) = $t_{ii}$ for $i$ = 1,2,...,$n$ and WI($i$) = 0.0.

If some of the computed eigenvalues form complex conjugate pairs, then $T$ has 2 by 2 diagonal blocks. Each diagonal block has the form

$$\begin{pmatrix} t_{ii} & t_{i,i+1} \\ t_{i+1,i} & t_{i+1,i+1} \end{pmatrix} = \begin{pmatrix} \alpha & \beta \\ \gamma & \alpha \end{pmatrix}$$

where $\beta\gamma < 0$. The corresponding eigenvalues are $\alpha \pm \sqrt{\beta\gamma}$; WR($i$) = WR($i$+1) = $\alpha$; WI($i$) = $+\sqrt{|\beta\gamma|}$; WI($i$+1) = $-$WI($i$).

The complex analogue of this routine is F08PSF (CHSEQR/ZHSEQR).

## 9. Example

To compute all the eigenvalues and the Schur factorization of the upper Hessenberg matrix $H$, where

$$H = \begin{pmatrix} 0.3500 & -0.1160 & -0.3886 & -0.2942 \\ -0.5140 & 0.1225 & 0.1004 & 0.1126 \\ 0.0000 & 0.6443 & -0.1357 & -0.0977 \\ 0.0000 & 0.0000 & 0.4262 & 0.1632 \end{pmatrix}.$$

See also the example for F08NFF (SORGHR/DORGHR), which illustrates the use of this routine to compute the Schur factorization of a general matrix.

### 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.**

```
*     F08PEF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDH, LWORK, LDZ
      PARAMETER        (NMAX=8,LDH=NMAX,LWORK=NMAX,LDZ=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
*     .. Local Arrays ..
      real             H(LDH,NMAX), WI(NMAX), WORK(LWORK), WR(NMAX),
     +                 Z(LDZ,NMAX)
*     .. External Subroutines ..
      EXTERNAL         shseqr, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08PEF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
```

```
      IF (N.LE.NMAX) THEN
*
*        Read H from data file
*
         READ (NIN,*) ((H(I,J),J=1,N),I=1,N)
*
*        Calculate the eigenvalues and Schur factorization of H
*
         CALL shseqr('Schur form','Initialize Z',N,1,N,H,LDH,WR,WI,Z,
     +               LDZ,WORK,LWORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.GT.0) THEN
            WRITE (NOUT,*) 'Failure to converge.'
         ELSE
            WRITE (NOUT,*) 'Eigenvalues'
            WRITE (NOUT,99999) (' (',WR(I),',',WI(I),')',I=1,N)
*
*           Print Schur form
*
            WRITE (NOUT,*)
            IFAIL = 0
*
            CALL X04CAF('General',' ',N,N,H,LDH,'Schur form',IFAIL)
*
*           Print Schur vectors
*
            WRITE (NOUT,*)
            IFAIL = 0
*
            CALL X04CAF('General',' ',N,N,Z,LDZ,'Schur vectors of H',
     +                  IFAIL)
*
         END IF
      END IF
      STOP
*
99999 FORMAT (1X,A,F8.4,A,F8.4,A)
      END
```

## 9.2. Program Data

```
F08PEF Example Program Data
  4                                          :Value of N
  0.3500  -0.1160  -0.3886  -0.2942
 -0.5140   0.1225   0.1004   0.1126
  0.0000   0.6443  -0.1357  -0.0977
  0.0000   0.0000   0.4262   0.1632          :End of matrix H
```

## 9.3. Program Results

```
 F08PEF Example Program Results

 Eigenvalues
  (  0.7995,  0.0000)
  ( -0.0994,  0.4008)
  ( -0.0994, -0.4008)
  ( -0.1007,  0.0000)

 Schur form
          1       2       3       4
1   0.7995 -0.1144  0.0061  0.0335
2   0.0000 -0.0994  0.2477  0.3474
3   0.0000 -0.6483 -0.0994  0.2026
4   0.0000  0.0000  0.0000 -0.1007
```

```
Schur vectors of H
          1        2        3        4
1    0.6551   0.1036   0.3450   0.6641
2   -0.5972  -0.5246   0.1706   0.5823
3   -0.3845   0.5789   0.7143  -0.0821
4   -0.2576   0.6156  -0.5845   0.4616
```

# F08PKF (SHSEIN/DHSEIN) – 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

F08PKF (SHSEIN/DHSEIN) computes selected left and/or right eigenvectors of a real upper Hessenberg matrix corresponding to specified eigenvalues, by inverse iteration.

## 2. Specification

```
      SUBROUTINE F08PKF (JOB, EIGSRC, INITV, SELECT, N, H, LDH, WR, WI, VL,
     1                   LDVL, VR, LDVR, MM, M, WORK, IFAILL, IFAILR, INFO)
      ENTRY       shsein (JOB, EIGSRC, INITV, SELECT, N, H, LDH, WR, WI, VL,
     1                   LDVL, VR, LDVR, MM, M, WORK, IFAILL, IFAILR, INFO)
      INTEGER           N, LDH, LDVL, LDVR, MM, M, IFAILL(*), IFAILR(*), INFO
      real              H(LDH,*), WR(*), WI(*), VL(LDVL,*), VR(LDVR,*), WORK(*)
      LOGICAL           SELECT(*)
      CHARACTER*1       JOB, EIGSRC, INITV
```

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

## 3. Description

This routine computes left and/or right eigenvectors of a real upper Hessenberg matrix $H$, corresponding to selected eigenvalues.

The right eigenvector $x$, and the left eigenvector $y$, corresponding to an eigenvalue $\lambda$, are defined by:

$$Hx = \lambda x \text{ and } y^H H = \lambda y^H \text{ (or } H^T y = \bar{\lambda} y).$$

Note that even though $H$ is real, $\lambda$, $x$ and $y$ may be complex. If $x$ is an eigenvector corresponding to a complex eigenvalue $\lambda$, then the complex conjugate vector $\bar{x}$ is the eigenvector corresponding to the complex conjugate eigenvalue $\bar{\lambda}$.

The eigenvectors are computed by inverse iteration. They are scaled so that, for a real eigenvector $x$, $\max(|x_i|) = 1$, and for a complex eigenvector, $\max(|Re(x_i)|+|Im(x_i)|) = 1$.

If $H$ has been formed by reduction of a real general matrix $A$ to upper Hessenberg form, then eigenvectors of $H$ may be transformed to eigenvectors of $A$ by a call to F08NGF (SORMHR/DORMHR).

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §7.6.1.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

## 5. Parameters

1: JOB – CHARACTER*1. *Input*

*On entry*: indicates whether left and/or right eigenvectors are to be computed as follows:

if JOB = 'R', then only right eigenvectors are computed;

if JOB = 'L', then only left eigenvectors are computed;

if JOB = 'B', then both left and right eigenvectors are computed.

*Constraint*: JOB = 'R', 'L' or 'B'.

2: EIGSRC – CHARACTER*1. *Input*

*On entry*: indicates whether the eigenvalues of $H$ (stored in WR and WI) were found using F08PEF (SHSEQR/DHSEQR) as follows:

if EIGSRC = 'Q', then the eigenvalues of $H$ were found using F08PEF (SHSEQR/DHSEQR); thus if $H$ has any zero sub-diagonal elements (and so is block triangular), then the $j$th eigenvalue can be assumed to be an eigenvalue of the block containing the $j$th row/column. This property allows the routine to perform inverse iteration on just one diagonal block;

if EIGSRC = 'N', then no such assumption is made and the routine performs inverse iteration using the whole matrix.

*Constraint*: EIGSRC = 'Q' or 'N'.

3: INITV – CHARACTER*1. *Input*

*On entry*: indicates whether the user is supplying initial estimates for the selected eigenvectors as follows:

if INITV = 'N', then no initial estimates are supplied;

if INITV = 'U', then initial estimates are supplied in VL and/or VR.

*Constraint*: INITV = 'N' or 'U'.

4: SELECT(*) – LOGICAL array. *Input/Output*

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

*On entry*: SELECT specifies which eigenvectors are to be computed. To obtain the real eigenvector corresponding to the real eigenvalue WR($j$), SELECT($j$) must be set .TRUE.. To select the complex eigenvector corresponding to the complex eigenvalue (WR($j$),WI($j$)) with complex conjugate (WR($j$+1),WI($j$+1)), SELECT($j$) and/or SELECT($j$+1) must be set .TRUE.; the eigenvector corresponding to the **first** eigenvalue in the pair is computed.

*On exit*: if a complex eigenvector was selected as specified above, then SELECT($j$) is set to .TRUE. and SELECT($j$+1) to .FALSE..

5: N – INTEGER. *Input*

*On entry*: $n$, the order of the matrix $H$.

*Constraint*: N $\geq$ 0.

6: H(LDH,*) – ***real*** array. *Input*

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

*On entry*: the $n$ by $n$ upper Hessenberg matrix $H$.

7: LDH – INTEGER. *Input*

*On entry*: the first dimension of the array H as declared in the (sub)program from which F08PKF (SHSEIN/DHSEIN) is called.

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

8: WR(*) – ***real*** array. *Input/Output*

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

9: WI(*) – ***real*** array. *Input*

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

*On entry*: the real and imaginary parts, respectively, of the eigenvalues of the matrix $H$. Complex conjugate pairs of values must be stored in consecutive elements of the arrays. If EIGSRC = 'Q', the arrays **must** be exactly as returned by F08PEF (SHSEQR/DHSEQR).

*On exit*: some elements of WR may be modified, as close eigenvalues are perturbed slightly in searching for independent eigenvectors.

10: VL(LDVL,*) – ***real*** array. *Input/Output*

**Note**: the second dimension of the array VL must be at least max(1,MM) if JOB = 'L' or 'B' and at least 1 if JOB = 'R'.

*On entry*: if INITV = 'U' and JOB = 'L' or 'B', VL must contain starting vectors for inverse iteration for the left eigenvectors. Each starting vector must be stored in the same column or columns as will be used to store the corresponding eigenvector (see below). If INITV = 'N', VL need not be set.

*On exit*: if JOB = 'L' or 'B', VL contains the computed left eigenvectors (as specified by SELECT). The eigenvectors are stored consecutively in the columns of the array, in the same order as their eigenvalues. Corresponding to each selected real eigenvalue is a real eigenvector, occupying one column. Corresponding to each selected complex eigenvalue is a complex eigenvector, occupying two columns: the first column holds the real part and the second column holds the imaginary part.

VL is not referenced if JOB = 'R'.

11: LDVL – INTEGER. *Input*

*On entry*: the first dimension of the array VL as declared in the (sub)program from which F08PKF (SHSEIN/DHSEIN) is called.

*Constraints*: LDVL ≥ max(1,N) if JOB = 'L' or 'B',
LDVL ≥ 1 if JOB = 'R'.

12: VR(LDVR,*) – ***real*** array. *Input/Output*

**Note**: the second dimension of the array VR must be at least max(1,MM) if JOB = 'R' or 'B' and at least 1 if JOB = 'L'.

*On entry*: if INITV = 'U' and JOB = 'R' or 'B', VR must contain starting vectors for inverse iteration for the right eigenvectors. Each starting vector must be stored in the same column or columns as will be used to store the corresponding eigenvector (see below). If INITV = 'N', VR need not be set.

*On exit*: if JOB = 'R' or 'B', VR contains the computed right eigenvectors (as specified by SELECT). The eigenvectors are stored consecutively in the columns of the array, in the same order as their eigenvalues. Corresponding to each selected real eigenvalue is a real eigenvector, occupying one column. Corresponding to each selected complex eigenvalue is a complex eigenvector, occupying two columns: the first column holds the real part and the second column holds the imaginary part.

VR is not referenced if JOB = 'L'.

13: LDVR – INTEGER. *Input*

*On entry*: the first dimension of the array VR as declared in the (sub)program from which F08PKF (SHSEIN/DHSEIN) is called.

*Constraints*: LDVR ≥ max(1,N) if JOB = 'R' or 'B',
LDVR ≥ 1 if JOB = 'L'.

14: MM – INTEGER. *Input*

*On entry*: the number of columns in the arrays VL and/or VR. The actual number of columns required, $m$, is obtained by counting 1 for each selected real eigenvector and 2 for each selected complex eigenvector (see SELECT); $0 \leq m \leq n$.

*Constraint*: MM ≥ $m$.

15: M – INTEGER. *Output*

*On exit*: $m$, the number of columns of VL and/or VR required to store the selected eigenvectors.

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

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

17: IFAILL(*) – INTEGER array. *Output*

**Note**: the dimension of the array IFAILL must be at least max(1,MM) if JOB = 'L' or 'B' and at least 1 if JOB = 'R'.

*On exit*: if JOB = 'L' or 'B', then IFAILL($i$) = 0 if the selected left eigenvector converged and IFAILL($i$) = $j$ > 0 if the eigenvector stored in the $i$th column of VL (corresponding to the $j$th eigenvalue) failed to converge. If the $i$th and ($i$+1)th columns of VL contain a selected complex eigenvector, then IFAILL($i$) and IFAILL($i$+1) are set to the same value.

IFAILL is not referenced if JOB = 'R'.

18: IFAILR(*) – INTEGER array. *Output*

**Note**: the dimension of the array IFAILR must be at least max(1,MM) if JOB = 'R' or 'B' and at least 1 if JOB = 'L'.

*On exit*: if JOB = 'R' or 'B', then IFAILR($i$) = 0 if the selected right eigenvector converged and IFAILR($i$) = $j$ > 0 if the eigenvector stored in the $i$th column of VR (corresponding to the $j$th eigenvalue) failed to converge. If the $i$th and ($i$+1)th columns of VR contain a selected complex eigenvector, then IFAILR($i$) and IFAILR($i$+1) are set to the same value.

IFAILR is not referenced if JOB = 'L'.

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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = $i$, then $i$ eigenvectors (as indicated by the parameters IFAILL and/or IFAILR above) failed to converge. The corresponding columns of VL and/or VR contain no useful information.

## 7. Accuracy

Each computed right eigenvector $x_i$ is the exact eigenvector of a nearby matrix $A + E_i$, such that $\|E_i\| = O(\varepsilon)\|A\|$. Hence the residual is small:

$$\|Ax_i - \lambda_i x_i\| = O(\varepsilon)\|A\|.$$

However eigenvectors corresponding to close or coincident eigenvalues may not accurately span the relevant subspaces.

Similar remarks apply to computed left eigenvectors.

## 8. Further Comments

The complex analogue of this routine is F08PXF (CHSEIN/ZHSEIN).

## 9. Example

See the example for F08NGF (SORMHR/DORMHR).

# F08PSF (CHSEQR/ZHSEQR) – 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

F08PSF (CHSEQR/ZHSEQR) computes all the eigenvalues, and optionally the Schur factorization, of a complex Hessenberg matrix or a complex general matrix which has been reduced to Hessenberg form.

## 2. Specification

```
 SUBROUTINE F08PSF (JOB, COMPZ, N, ILO, IHI, H, LDH, W, Z, LDZ, WORK,
1                   LWORK, INFO)
 ENTRY       chseqr (JOB, COMPZ, N, ILO, IHI, H, LDH, W, Z, LDZ, WORK,
1                   LWORK, INFO)
 INTEGER          N, ILO, IHI, LDH, LDZ, LWORK, INFO
 complex          H(LDH,*), W(*), Z(LDZ,*), WORK(*)
 CHARACTER*1      JOB, COMPZ
```

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

## 3. Description

This routine computes all the eigenvalues, and optionally the Schur factorization, of a complex upper Hessenberg matrix $H$:

$$H = ZTZ^H,$$

where $T$ is an upper triangular matrix (the Schur form of $H$), and $Z$ is the unitary matrix whose columns are the Schur vectors $z_i$. The diagonal elements of $T$ are the eigenvalues of $H$.

The routine may also be used to compute the Schur factorization of a complex general matrix $A$ which has been reduced to upper Hessenberg form $H$:

$$\begin{aligned} A &= QHQ^H, \text{ where } Q \text{ is unitary,} \\ &= (QZ)T(QZ)^H. \end{aligned}$$

In this case, after F08NSF (CGEHRD/ZGEHRD) has been called to reduce $A$ to Hessenberg form, F08NTF (CUNGHR/ZUNGHR) must be called to form $Q$ explicitly; $Q$ is then passed to F08PSF, which must be called with COMPZ = 'V'.

The routine can also take advantage of a previous call to F08NVF (CGEBAL/ZGEBAL) which may have balanced the original matrix before reducing it to Hessenberg form, so that the Hessenberg matrix $H$ has the structure:

$$\begin{pmatrix} H_{11} & H_{12} & H_{13} \\ & H_{22} & H_{23} \\ & & H_{33} \end{pmatrix}$$

where $H_{11}$ and $H_{33}$ are upper triangular. If so, only the central diagonal block $H_{22}$ (in rows and columns $i_{lo}$ to $i_{hi}$) needs to be further reduced to Schur form (the blocks $H_{12}$ and $H_{23}$ are also affected). Therefore the values of $i_{lo}$ and $i_{hi}$ can be supplied to F08PSF directly. Also, F08NWF (CGEBAK/ZGEBAK) must be called after this routine to permute the Schur vectors of the balanced matrix to those of the original matrix. If F08NVF has not been called however, then $i_{lo}$ must be set to 1 and $i_{hi}$ to $n$. Note that if the Schur factorization of $A$ is required, F08NVF must **not** be called with JOB = 'S' or 'B', because the balancing transformation is not unitary.

F08PSF uses a multishift form of the upper Hessenberg $QR$ algorithm, due to Bai and Demmel [1]. The Schur vectors are normalized so that $\|z_i\|_2 = 1$, but are determined only to within a complex factor of absolute value 1.

## 4. References

[1] BAI, Z. and DEMMEL, J.W.
On a Block Implementation of Hessenberg Multishift *QR* Iteration.
Int. J. High Speed Comp., 1, pp. 97-112, 1989.

[2] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §7.5.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

## 5. Parameters

1: JOB – CHARACTER*1. *Input*

*On entry*: indicates whether eigenvalues only or the Schur form $T$ is required, as follows:

if JOB = 'E', then eigenvalues only are required;

if JOB = 'S', then the Schur form $T$ is required.

*Constraint*: JOB = 'E' or 'S'.

2: COMPZ – CHARACTER*1. *Input*

*On entry*: indicates whether the Schur vectors are to be computed as follows:

if COMPZ = 'N', then no Schur vectors are computed (and the array Z is not referenced);

if COMPZ = 'I', then the Schur vectors of $H$ are computed (and the array Z is initialized by the routine);

if COMPZ = 'V', then the Schur vectors of $A$ are computed (and the array Z must contain the matrix $Q$ on entry).

*Constraint*: COMPZ = 'N', 'I' or 'V'.

3: N – INTEGER. *Input*

*On entry*: $n$, the order of the matrix $H$.

*Constraint*: N ≥ 0.

4: ILO – INTEGER. *Input*

5: IHI – INTEGER. *Input*

*On entry*: if the matrix $A$ has been balanced by F08NVF (CGEBAL/ZGEBAL), then ILO and IHI must contain the values returned by that routine. Otherwise, ILO must be set to 1 and IHI to N.

*Constraints*: ILO ≥ 1 and min(ILO,N) ≤ IHI ≤ N.

6: H(LDH,*) – ***complex*** array. *Input/Output*

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

*On entry*: the $n$ by $n$ upper Hessenberg matrix $H$, as returned by F08NSF (CGEHRD/ZGEHRD).

*On exit*: if JOB = 'E', then the array contains no useful information. If JOB = 'S', then $H$ is overwritten by the upper triangular matrix $T$ from the Schur decomposition (the Schur form) unless INFO > 0.

7: LDH – INTEGER. *Input*

*On entry*: the first dimension of the array H as declared in the (sub)program from which F08PSF (CHSEQR/ZHSEQR) is called.

*Constraint*: LDH ≥ max(1,N).

8: W(*) – ***complex*** array. *Output*

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

*On exit*: the computed eigenvalues, unless INFO > 0 (in which case see Section 6). The eigenvalues are stored in the same order as on the diagonal of the Schur form $T$ (if computed).

9: Z(LDZ,*) – ***complex*** array. *Input/Output*

**Note**: the second dimension of the array Z must be at least max(1,N) if COMPZ = 'V' or 'I' and at least 1 if COMPZ = 'N'.

*On entry*: if COMPZ = 'V', Z must contain the unitary matrix $Q$ from the reduction to Hessenberg form; if COMPZ = 'I', Z need not be set.

*On exit*: if COMPZ = 'V' or 'I', Z contains the unitary matrix of the required Schur vectors, unless INFO > 0.

Z is not referenced if COMPZ = 'N'.

10: LDZ – INTEGER. *Input*

*On entry*: the first dimension of the array Z as declared in the (sub)program from which F08PSF (CHSEQR/ZHSEQR) is called.

*Constraints*: LDZ ≥ 1 if COMPZ = 'N',
LDZ ≥ max(1,N) if COMPZ = 'V' or 'I'.

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

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

12: LWORK – INTEGER. *Dummy*

This parameter is currently redundant.

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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

The algorithm has failed to find all the eigenvalues after a total of 30×(IHI–ILO+1) iterations. If INFO = $i$, elements 1,2,...,ILO–1 and $i$+1,$i$+2,...,$n$ of W contain the eigenvalues which have been found.

## 7. Accuracy

The computed Schur factorization is the exact factorization of a nearby matrix $H + E$, where

$$\|E\|_2 = O(\varepsilon)\|H\|_2,$$

and $\varepsilon$ is the ***machine precision***.

If $\lambda_i$ is an exact eigenvalue, and $\tilde{\lambda}_i$ is the corresponding computed value, then

$$|\tilde{\lambda}_i - \lambda_i| \leq \frac{c(n)\varepsilon\|H\|_2}{s_i},$$

where $c(n)$ is a modestly increasing function of $n$, and $s_i$ is the reciprocal condition number of $\lambda_i$. The condition numbers $s_i$ may be computed by calling F08QYF (CTRSNA/ZTRSNA).

## 8. Further Comments

The total number of real floating-point operations depends on how rapidly the algorithm converges, but is typically about:

$25n^3$ if only eigenvalues are computed;
$35n^3$ if the Schur form is computed;
$70n^3$ if the full Schur factorization is computed.

The real analogue of this routine is F08PEF (SHSEQR/DHSEQR).

## 9. Example

To compute all the eigenvalues and the Schur factorization of the upper Hessenberg matrix $H$, where

$$H = \begin{pmatrix} -3.9700 - 5.0400i & -1.1318 - 2.5693i & -4.6027 - 0.1426i & -1.4249 + 1.7330i \\ -5.4797 + 0.0000i & 1.8585 - 1.5502i & 4.4145 - 0.7638i & -0.4805 - 1.1976i \\ 0.0000 + 0.0000i & 6.2673 + 0.0000i & -0.4504 - 0.0290i & -1.3467 + 1.6579i \\ 0.0000 + 0.0000i & 0.0000 + 0.0000i & -3.5000 + 0.0000i & 2.5619 - 3.3708i \end{pmatrix}.$$

See also the example for F08NTF (CUNGHR/ZUNGHR), which illustrates the use of this routine to compute the Schur factorization of a general matrix.

### 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.

```
*     F08PSF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDH, LWORK, LDZ
      PARAMETER        (NMAX=8,LDH=NMAX,LWORK=NMAX,LDZ=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
*     .. Local Arrays ..
      complex          H(LDH,NMAX), W(NMAX), WORK(LWORK), Z(LDZ, NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         X04DBF, chseqr
*     .. Intrinsic Functions ..
      INTRINSIC        real, imag
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08PSF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read H from data file
*
         READ (NIN,*) ((H(I,J),J=1,N),I=1,N)
*
*        Calculate the eigenvalues and Schur factorization of H
*
         CALL chseqr('Schur form','Initialize Z',N,1,N,H,LDH,W,Z,LDZ,
     +               WORK,LWORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.GT.0) THEN
            WRITE (NOUT,*) 'Failure to converge.'
         ELSE
            WRITE (NOUT,*) 'Eigenvalues'
            WRITE (NOUT,99999) (' (',real(W(I)),',',imag(W(I)),')',I=1,
     +        N)
*
```

```
*           Print Schur form
*
            WRITE (NOUT,*)
            IFAIL = 0
*
            CALL X04DBF('General',' ',N,N,H,LDH,'Bracketed','F7.4',
     +                  'Schur form','Integer',RLABS,'Integer',CLABS,80,
     +                  0,IFAIL)
*
*           Print Schur vectors
*
            WRITE (NOUT,*)
            IFAIL = 0
*
            CALL X04DBF('General',' ',N,N,Z,LDZ,'Bracketed','F7.4',
     +                  'Schur vectors of H','Integer',RLABS,'Integer',
     +                  CLABS,80,0,IFAIL)
*
         END IF
       END IF
       STOP
*
99999 FORMAT ((3X,4(A,F7.4,A,F7.4,A,:)))
      END
```

## 9.2. Program Data

```
F08PSF Example Program Data
  4                                                                 :Value of N
 (-3.9700,-5.0400) (-1.1318,-2.5693) (-4.6027,-0.1426) (-1.4249, 1.7330)
 (-5.4797, 0.0000) ( 1.8585,-1.5502) ( 4.4145,-0.7638) (-0.4805,-1.1976)
 ( 0.0000, 0.0000) ( 6.2673, 0.0000) (-0.4504,-0.0290) (-1.3467, 1.6579)
 ( 0.0000, 0.0000) ( 0.0000, 0.0000) (-3.5000, 0.0000) ( 2.5619,-3.3708)
                                                                   :End of matrix H
```

## 9.3. Program Results

```
F08PSF Example Program Results

Eigenvalues
   (-6.0004,-6.9998) (-5.0000, 2.0060) ( 7.9982,-0.9964) ( 3.0023,-3.9998)

Schur form
                     1                   2                   3                   4
1  (-6.0004,-6.9998) (-0.2080, 0.4719) (-0.4829, 0.1768) ( 0.1301, 0.9052)
2  ( 0.0000, 0.0000) (-5.0000, 2.0060) (-0.6653, 0.2814) ( 0.0038, 0.2639)
3  ( 0.0000, 0.0000) ( 0.0000, 0.0000) ( 7.9982,-0.9964) ( 0.2004, 1.0595)
4  ( 0.0000, 0.0000) ( 0.0000, 0.0000) ( 0.0000, 0.0000) ( 3.0023,-3.9998)

Schur vectors of H
                     1                   2                   3                   4
1  ( 0.8457, 0.0000) ( 0.1380, 0.3602) (-0.2677,-0.1091) (-0.2213,-0.0582)
2  ( 0.2824,-0.3304) (-0.4612, 0.2075) ( 0.6846, 0.0000) ( 0.2927, 0.0320)
3  ( 0.0748, 0.2800) ( 0.7239, 0.0000) ( 0.5924,-0.0189) (-0.0229, 0.2005)
4  ( 0.0670, 0.0860) ( 0.2169, 0.1560) (-0.2745, 0.1454) ( 0.9057, 0.0000)
```

# F08PXF (CHSEIN/ZHSEIN) – 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

F08PXF (CHSEIN/ZHSEIN) computes selected left and/or right eigenvectors of a complex upper Hessenberg matrix corresponding to specified eigenvalues, by inverse iteration.

## 2. Specification

```
 SUBROUTINE F08PXF (JOB, EIGSRC, INITV, SELECT, N, H, LDH, W, VL,
1                   LDVL, VR, LDVR, MM, M, WORK, RWORK, IFAILL, IFAILR,
2                   INFO)
 ENTRY       chsein (JOB, EIGSRC, INITV, SELECT, N, H, LDH, W, VL,
1                   LDVL, VR, LDVR, MM, M, WORK, RWORK, IFAILL, IFAILR,
2                   INFO)
 INTEGER         N, LDH, LDVL, LDVR, MM, M, IFAILL(*), IFAILR(*), INFO
 real            RWORK(*)
 complex         H(LDH,*), W(*), VL(LDVL,*), VR(LDVR,*), WORK(*)
 LOGICAL         SELECT(*)
 CHARACTER*1     JOB, EIGSRC, INITV
```

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

## 3. Description

This routine computes left and/or right eigenvectors of a complex upper Hessenberg matrix $H$, corresponding to selected eigenvalues.

The right eigenvector $x$, and the left eigenvector $y$, corresponding to an eigenvalue $\lambda$, are defined by:

$$Hx = \lambda x \text{ and } y^H H = \lambda y^H \text{ (or } H^H y = \bar{\lambda} y).$$

The eigenvectors are computed by inverse iteration. They are scaled so that $\max(|Re(x_i)|+|Im(x_i)|) = 1$.

If $H$ has been formed by reduction of a complex general matrix $A$ to upper Hessenberg form, then the eigenvectors of $H$ may be transformed to eigenvectors of $A$ by a call to F08NUF (CUNMHR/ZUNMHR).

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §7.6.1.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

## 5. Parameters

1: JOB – CHARACTER*1. *Input*

*On entry*: indicates whether left and/or right eigenvectors are to be computed as follows:

if JOB = 'R', then only right eigenvectors are computed;

if JOB = 'L', then only left eigenvectors are computed;

if JOB = 'B', then both left and right eigenvectors are computed.

*Constraint*: JOB = 'R', 'L' or 'B'.

2: EIGSRC – CHARACTER*1. *Input*

*On entry*: indicates whether the eigenvalues of $H$ (stored in W) were found using F08PSF (CHSEQR/ZHSEQR) as follows:

if EIGSRC = 'Q', then the eigenvalues of $H$ were found using F08PSF (CHSEQR/ZHSEQR); thus if $H$ has any zero sub-diagonal elements (and so is block triangular), then the $j$th eigenvalue can be assumed to be an eigenvalue of the block containing the $j$th row/column. This property allows the routine to perform inverse iteration on just one diagonal block;

if EIGSRC = 'N', then no such assumption is made and the routine performs inverse iteration using the whole matrix.

*Constraint*: EIGSRC = 'Q' or 'N'.

3: INITV – CHARACTER*1. *Input*

*On entry*: indicates whether the user is supplying initial estimates for the selected eigenvectors as follows:

if INITV = 'N', then no initial estimates are supplied;

if INITV = 'U', then initial estimates are supplied in VL and/or VR.

*Constraint*: INITV = 'N' or 'U'.

4: SELECT(*) – LOGICAL array. *Input*

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

*On entry*: SELECT specifies which eigenvectors are to be computed. To select the eigenvector corresponding to the eigenvalue W($j$), SELECT($j$) must be set .TRUE..

5: N – INTEGER. *Input*

*On entry*: $n$, the order of the matrix $H$.

*Constraint*: N $\geq$ 0.

6: H(LDH,*) – ***complex*** array. *Input*

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

*On entry*: the $n$ by $n$ upper Hessenberg matrix $H$.

7: LDH – INTEGER. *Input*

*On entry*: the first dimension of the array H as declared in the (sub)program from which F08PXF (CHSEIN/ZHSEIN) is called.

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

8: W(*) – ***complex*** array. *Input/Output*

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

*On entry*: the eigenvalues of the matrix $H$. If EIGSRC = 'Q', the array **must** be exactly as returned by F08PSF (CHSEQR/ZHSEQR).

*On exit*: the real parts of some elements of W may be modified, as close eigenvalues are perturbed slightly in searching for independent eigenvectors.

9: VL(LDVL,*) – ***complex*** array. *Input/Output*

**Note**: the second dimension of the array VL must be at least max(1,MM) if JOB = 'L' or 'B' and at least 1 if JOB = 'R'.

*On entry*: if INITV = 'U' and JOB = 'L' or 'B', VL must contain starting vectors for inverse iteration for the left eigenvectors. Each starting vector must be stored in the same column as will be used to store the corresponding eigenvector (see below). If INITV = 'N', VL need not be set.

*On exit*: if JOB = 'L' or 'B', VL contains the computed left eigenvectors (as specified by SELECT). The eigenvectors are stored consecutively in the columns of the array, in the same order as their eigenvalues.

VL is not referenced if JOB = 'R'.

10: LDVL – INTEGER. *Input*

*On entry*: the first dimension of the array VL as declared in the (sub)program from which F08PXF (CHSEIN/ZHSEIN) is called.

*Constraints*: LDVL ≥ max(1,N) if JOB = 'L' or 'B',
LDVL ≥ 1 if JOB = 'R'.

11: VR(LDVR,*) – ***complex*** array. *Input/Output*

**Note**: the second dimension of the array VR must be at least max(1,MM) if JOB = 'R' or 'B' and at least 1 if JOB = 'L'.

*On entry*: if INITV = 'U' and JOB = 'R' or 'B', VR must contain starting vectors for inverse iteration for the right eigenvectors. Each starting vector must be stored in the same column as will be used to store the corresponding eigenvector (see below). If INITV = 'N', VR need not be set.

*On exit*: if JOB = 'R' or 'B', VR contains the computed right eigenvectors (as specified by SELECT). The eigenvectors are stored consecutively in the columns of the array, in the same order as their eigenvalues.

VR is not referenced if JOB = 'L'.

12: LDVR – INTEGER. *Input*

*On entry*: the first dimension of the array VR as declared in the (sub)program from which F08PXF (CHSEIN/ZHSEIN) is called.

*Constraints*: LDVR ≥ max(1,N) if JOB = 'R' or 'B',
LDVR ≥ 1 if JOB = 'L'.

13: MM – INTEGER. *Input*

*On entry*: the number of columns in the arrays VL and/or VR. The actual number of columns required, $m$, is equal to the number of selected eigenvectors (see SELECT); $0 \leq m \leq n$.

*Constraint*: MM ≥ $m$.

14: M – INTEGER. *Output*

*On exit*: $m$, the number of selected eigenvectors.

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

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

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

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

17: IFAILL(*) – INTEGER array. *Output*

**Note**: the dimension of the array IFAILL must be at least max(1,MM) if JOB = 'L' or 'B' and at least 1 if JOB = 'R'.

*On exit*: if JOB = 'L' or 'B', then IFAILL($i$) = 0 if the selected left eigenvector converged and IFAILL($i$) = $j$ > 0 if the eigenvector stored in the $i$th column of VL (corresponding to the $j$th eigenvalue) failed to converge.

IFAILL is not referenced if JOB = 'R'.

18: IFAILR(*) – INTEGER array. *Output*

**Note**: the dimension of the array IFAILR must be at least max(1,MM) if JOB = 'R' or 'B' and at least 1 if JOB = 'L'.

*On exit*: if JOB = 'R' or 'B', then IFAILR($i$) = 0 if the selected right eigenvector converged and IFAILR($i$) = $j$ > 0 if the eigenvector stored in the $i$th column of VR (corresponding to the $j$th eigenvalue) failed to converge.

IFAILR is not referenced if JOB = 'L'.

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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = $i$, then $i$ eigenvectors (as indicated by the parameters IFAILL and/or IFAILR above) failed to converge. The corresponding columns of VL and/or VR contain no useful information.

## 7. Accuracy

Each computed right eigenvector $x_i$ is the exact eigenvector of a nearby matrix $A + E_i$, such that $\|E_i\| = O(\varepsilon)\|A\|$. Hence the residual is small:

$$\|Ax_i - \lambda_i x_i\| = O(\varepsilon)\|A\|.$$

However eigenvectors corresponding to close or coincident eigenvalues may not accurately span the relevant subspaces.

Similar remarks apply to computed left eigenvectors.

## 8. Further Comments

The real analogue of this routine is F08PKF (SHSEIN/DHSEIN).

## 9. Example

See the example for F08NUF (CUNMHR/ZUNMHR).

# F08QFF (STREXC/DTREXC) – 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

F08QFF (STREXC/DTREXC) reorders the Schur factorization of a real general matrix.

## 2. Specification

```
SUBROUTINE F08QFF (COMPQ, N, T, LDT, Q, LDQ, IFST, ILST, WORK, INFO)
ENTRY      strexc  (COMPQ, N, T, LDT, Q, LDQ, IFST, ILST, WORK, INFO)
INTEGER          N, LDT, LDQ, IFST, ILST, INFO
real             T(LDT,*), Q(LDQ,*), WORK(*)
CHARACTER*1      COMPQ
```

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

## 3. Description

This routine reorders the Schur factorization of a real general matrix $A = QTQ^T$, so that the diagonal element or block of $T$ with row index IFST is moved to row ILST.

The reordered Schur form $\tilde{T}$ is computed by an orthogonal similarity transformation: $\tilde{T} = Z^T T Z$. Optionally the updated matrix $\tilde{Q}$ of Schur vectors is computed as $\tilde{Q} = QZ$, giving $A = \tilde{Q}\tilde{T}\tilde{Q}^T$.

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §7.6.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

## 5. Parameters

1: COMPQ – CHARACTER*1. *Input*

*On entry*: indicates whether the matrix $Q$ of Schur vectors is to be updated, as follows:

if COMPQ = 'V', then the matrix $Q$ of Schur vectors is updated;

if COMPQ = 'N', then no Schur vectors are updated.

*Constraint*: COMPQ = 'V' or 'N'.

2: N – INTEGER. *Input*

*On entry*: $n$, the order of the matrix $T$.

*Constraint*: N ≥ 0.

3: T(LDT,*) – ***real*** array. *Input/Output*

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

*On entry*: the $n$ by $n$ upper quasi-triangular matrix $T$ in canonical Schur form, as returned by F08PEF (SHSEQR/DHSEQR).

*On exit*: $T$ is overwritten by the updated matrix $\tilde{T}$. See also Section 8.

4: LDT – INTEGER. *Input*

*On entry*: the first dimension of the array T as declared in the (sub)program from which F08QFF (STREXC/DTREXC) is called.

*Constraint*: LDT ≥ max(1,N).

5: Q(LDQ,*) – ***real*** array. *Input/Output*

**Note**: the second dimension of the array Q must be at least max(1,N) if COMPQ = 'V' and at least 1 if COMPQ = 'N'.

*On entry*: if COMPQ = 'V', Q must contain the $n$ by $n$ orthogonal matrix $Q$ of Schur vectors.

*On exit*: if COMPQ = 'V', Q contains the updated matrix of Schur vectors.

Q is not referenced if COMPQ = 'N'.

6: LDQ – INTEGER. *Input*

*On entry*: the first dimension of the array Q as declared in the (sub)program from which F08QFF (STREXC/DTREXC) is called.

*Constraints*: LDQ ≥ max(1,N) if COMPQ = 'V',
LDQ ≥ 1 if COMPQ = 'N'.

7: IFST – INTEGER. *Input/Output*

8: ILST – INTEGER. *Input/Output*

*On entry*: IFST and ILST must specify the reordering of the diagonal elements or blocks of $T$. The element or block with row index IFST is moved to row ILST by a sequence of exchanges between adjacent elements or blocks.

*On exit*: if IFST pointed to the second row of a 2 by 2 block on entry, it is changed to point to the first row. ILST always points to the first row of the block in its final position (which may differ from its input value by ± 1).

*Constraints*: 1 ≤ IFST ≤ N,
1 ≤ ILST ≤ N.

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

**Note**: the dimension of the array WORK 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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO = 1

Two adjacent diagonal elements or blocks could not be successfully exchanged. This error can only occur if the exchange involves at least one 2 by 2 block; it implies that the problem is very ill-conditioned, and that the eigenvalues of the two blocks are very close. On exit, $T$ may have been partially reordered, and ILST points to the first row of the current position of the block being moved; $Q$ (if requested) is updated consistently with $T$.

## 7. Accuracy

The computed matrix $\tilde{T}$ is exactly similar to a matrix $T + E$, where

$$\|E\|_2 = O(\varepsilon)\|T\|_2,$$

and $\varepsilon$ is the ***machine precision***.

Note that if a 2 by 2 diagonal block is involved in the re-ordering, its off-diagonal elements are in general changed; the diagonal elements and the eigenvalues of the block are unchanged unless the block is sufficiently ill-conditioned, in which case they may be noticeably altered. It is possible for a 2 by 2 block to break into two 1 by 1 blocks, that is, for a pair of complex

eigenvalues to become purely real. The values of real eigenvalues however are never changed by the re-ordering.

## 8. Further Comments

The total number of floating-point operations is approximately $6nr$ if COMPQ = 'N', and $12nr$ if COMPQ = 'V', where $r$ = |IFST–ILST|.

The input matrix $T$ must be in canonical Schur form, as is the output matrix $\tilde{T}$. This has the following structure.

If all the computed eigenvalues are real, $T$ is upper triangular and its diagonal elements are the eigenvalues.

If some of the computed eigenvalues form complex conjugate pairs, then $T$ has 2 by 2 diagonal blocks. Each diagonal block has the form

$$\begin{pmatrix} t_{ii} & t_{i,i+1} \\ t_{i+1,i} & t_{i+1,i+1} \end{pmatrix} = \begin{pmatrix} \alpha & \beta \\ \gamma & \alpha \end{pmatrix}$$

where $\beta\gamma < 0$. The corresponding eigenvalues are $\alpha \pm \sqrt{\beta\gamma}$.

The complex analogue of this routine is F08QTF (CTREXC/ZTREXC).

## 9. Example

To reorder the Schur factorization of the matrix $T$ so that the 2 by 2 block with row index 2 is moved to row 1, where

$$T = \begin{pmatrix} 0.80 & -0.11 & 0.01 & 0.03 \\ 0.00 & -0.10 & 0.25 & 0.35 \\ 0.00 & -0.65 & -0.10 & 0.20 \\ 0.00 & 0.00 & 0.00 & -0.10 \end{pmatrix}.$$

### 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.**

```
*     F08QFF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDT, LDQ
      PARAMETER        (NMAX=8,LDT=NMAX,LDQ=1)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, IFST, ILST, INFO, J, N
*     .. Local Arrays ..
      real             Q(LDQ,1), T(LDT,NMAX), WORK(NMAX)
*     .. External Subroutines ..
      EXTERNAL         strexc, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08QFF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read T from data file
*
         READ (NIN,*) ((T(I,J),J=1,N),I=1,N)
*
         READ (NIN,*) IFST, ILST
*
*        Reorder the Schur factorization T
*
         CALL strexc('No update',N,T,LDT,Q,LDQ,IFST,ILST,WORK,INFO)
*
```

```
*        Print reordered Schur form
*
         WRITE (NOUT,*)
         IFAIL = 0
*
         CALL X04CAF('General',' ',N,N,T,LDT,'Reordered Schur form',
     +               IFAIL)
*
       END IF
       STOP
       END
```

## 9.2. Program Data

```
F08QFF Example Program Data
  4                                :Value of N
  0.80  -0.11   0.01   0.03
  0.00  -0.10   0.25   0.35
  0.00  -0.65  -0.10   0.20
  0.00   0.00   0.00  -0.10        :End of matrix T
  2  1                             :Values of IFST and ILST
```

## 9.3. Program Results

```
 F08QFF Example Program Results

 Reordered Schur form
          1       2       3       4
1  -0.1000 -0.6463  0.0874  0.2010
2   0.2514 -0.1000  0.0927  0.3505
3   0.0000  0.0000  0.8000 -0.0117
4   0.0000  0.0000  0.0000 -0.1000
```

# F08QGF (STRSEN/DTRSEN) – 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

F08QGF (STRSEN/DTRSEN) reorders the Schur factorization of a real general matrix so that a selected cluster of eigenvalues appears in the leading elements or blocks on the diagonal of the Schur form. The routine also optionally computes the reciprocal condition numbers of the cluster of eigenvalues and/or the invariant subspace.

## 2. Specification

```
 SUBROUTINE F08QGF (JOB, COMPQ, SELECT, N, T, LDT, Q, LDQ, WR, WI, M, S,
1                   SEP, WORK, LWORK, IWORK, LIWORK, INFO)
 ENTRY       strsen  (JOB, COMPQ, SELECT, N, T, LDT, Q, LDQ, WR, WI, M, S,
1                   SEP, WORK, LWORK, IWORK, LIWORK, INFO)
 INTEGER          N, LDT, LDQ, M, LWORK, IWORK(LIWORK), LIWORK, INFO
 real             T(LDT,*), Q(LDQ,*), WR(*), WI(*), S, SEP, WORK(LWORK)
 LOGICAL          SELECT(*)
 CHARACTER*1      JOB, COMPQ
```

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

## 3. Description

This routine reorders the Schur factorization of a real general matrix $A = QTQ^T$, so that a selected cluster of eigenvalues appears in the leading diagonal elements or blocks of the Schur form.

The reordered Schur form $\tilde{T}$ is computed by an orthogonal similarity transformation: $\tilde{T} = Z^T T Z$. Optionally the updated matrix $\tilde{Q}$ of Schur vectors is computed as $\tilde{Q} = QZ$, giving $A = \tilde{Q}\tilde{T}\tilde{Q}^T$.

Let $\tilde{T} = \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix}$, where the selected eigenvalues are precisely the eigenvalues of the leading $m$ by $m$ submatrix $T_{11}$. Let $\tilde{Q}$ be correspondingly partitioned as $(Q_1 \ Q_2)$ where $Q_1$ consists of the first $m$ columns of $Q$. Then $AQ_1 = Q_1 T_{11}$, and so the $m$ columns of $Q_1$ form an orthonormal basis for the invariant subspace corresponding to the selected cluster of eigenvalues.

Optionally the routine also computes estimates of the reciprocal condition numbers of the average of the cluster of eigenvalues and of the invariant subspace.

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §7.2 and §7.6.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

## 5. Parameters

1: JOB – CHARACTER*1. *Input*

*On entry*: indicates whether condition numbers are required for the cluster of eigenvalues and/or the invariant subspace, as follows:

if JOB = 'N', then no condition numbers are required;

if JOB = 'E', then only the condition number for the cluster of eigenvalues is computed;

if JOB = 'V', then only the condition number for the invariant subspace is computed;

if JOB = 'B', then condition numbers for both the cluster of eigenvalues and the invariant subspace are computed.

*Constraint*: JOB = 'N', 'E', 'V' or 'B'.

2: COMPQ – CHARACTER*1. *Input*

*On entry*: indicates whether the matrix $Q$ of Schur vectors is to be updated, as follows:

if COMPQ = 'V', then the matrix $Q$ of Schur vectors is updated;

if COMPQ = 'N', then no Schur vectors are updated.

*Constraint*: COMPQ = 'V' or 'N'.

3: SELECT(*) – LOGICAL array. *Input*

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

*On entry*: SELECT specifies the eigenvalues in the selected cluster. To select a real eigenvalue $\lambda_j$, SELECT($j$) must be set .TRUE.. To select a complex conjugate pair of eigenvalues $\lambda_j$ and $\lambda_{j+1}$ (corresponding to a 2 by 2 diagonal block), SELECT($j$) and/or SELECT($j$+1) must be set to .TRUE.. A complex conjugate pair of eigenvalues **must** be either both included in the cluster or both excluded. See also Section 8.

4: N – INTEGER. *Input*

*On entry*: $n$, the order of the matrix $T$.

*Constraint*: N ≥ 0.

5: T(LDT,*) – ***real*** array. *Input/Output*

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

*On entry*: the $n$ by $n$ upper quasi-triangular matrix $T$ in canonical Schur form, as returned by F08PEF (SHSEQR/DHSEQR). See also Section 8.

*On exit*: $T$ is overwritten by the updated matrix $\tilde{T}$.

6: LDT – INTEGER. *Input*

*On entry*: the first dimension of the array T as declared in the (sub)program from which F08QGF (STRSEN/DTRSEN) is called.

*Constraint*: LDT ≥ max(1,N).

7: Q(LDQ,*) – ***real*** array. *Input/Output*

**Note**: the second dimension of the array Q must be at least max(1,N) if COMPQ = 'V' and at least 1 if COMPQ = 'N'.

*On entry*: if COMPQ = 'V', Q must contain the $n$ by $n$ orthogonal matrix $Q$ of Schur vectors, as returned by F08PEF (SHSEQR/DHSEQR).

*On exit*: if COMPQ = 'V', Q contains the updated matrix of Schur vectors; the first $m$ columns of Q form an orthonormal basis for the specified invariant subspace.

Q is not referenced if COMPQ = 'N'.

8: LDQ – INTEGER. *Input*

*On entry*: the first dimension of the array Q as declared in the (sub)program from which F08QGF (STRSEN/DTRSEN) is called.

*Constraints*: LDQ ≥ max(1,N) if COMPQ = 'V',
LDQ ≥ 1 if COMPQ = 'N'.

9: WR(*) – ***real*** array. *Output*

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

10: WI(*) – ***real*** array. *Output*

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

*On exit*: the real and imaginary parts, respectively, of the reordered eigenvalues of $\tilde{T}$. The eigenvalues are stored in the same order as on the diagonal of $\tilde{T}$; see Section 8 for details. Note that if a complex eigenvalue is sufficiently ill-conditioned, then its value may differ significantly from its value before reordering.

11: M – INTEGER. *Output*

*On exit*: $m$, the dimension of the specified invariant subspace. The value of $m$ is obtained by counting 1 for each selected real eigenvalue and 2 for each selected complex conjugate pair of eigenvalues (see SELECT); $0 \leq m \leq n$.

12: S – ***real***. *Output*

*On exit*: if JOB = 'E' or 'B', S is a lower bound on the reciprocal condition number of the average of the selected cluster of eigenvalues. If M = 0 or N, then S = 1; if INFO = 1 (see Section 6), then S is set to zero.

S is not referenced if JOB = 'N' or 'V'.

13: SEP – ***real***. *Output*

*On exit*: if JOB = 'V' or 'B', SEP is the estimated reciprocal condition number of the specified invariant subspace. If M = 0 or N, SEP = $\|T\|$; if INFO = 1 (see Section 6), then SEP is set to zero.

SEP is not referenced if JOB = 'N' or 'E'.

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

15: LWORK – INTEGER. *Input*

*On entry*: the dimension of the array WORK as declared in the (sub)program from which F08QGF (STRSEN/DTRSEN) is called.

*Constraints*: if JOB = 'N', then LWORK $\geq$ max(1,N),
if JOB = 'E', then LWORK $\geq$ max(1,$m$×(N–$m$)),
if JOB = 'V' or 'B', then LWORK $\geq$ max(1,2×$m$×(N–$m$)).

The actual amount of workspace required cannot exceed $N^2/4$ if JOB = 'E' or $N^2/2$ if JOB = 'V' or 'B'.

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

17: LIWORK – INTEGER. *Input*

*On entry*: the dimension of the array IWORK as declared in the (sub)program from which F08QGF (STRSEN/DTRSEN) is called.

*Constraints*: if JOB = 'N' or 'E', then LIWORK $\geq$ 1,
if JOB = 'V' or 'B', then LIWORK $\geq$ max(1,$m$×(N–$m$)).

The actual amount of workspace required cannot exceed $N^2/2$ if JOB = 'V' or 'B'.

IWORK is not referenced if JOB = 'N' or 'E'.

18: 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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO = 1

The reordering of $T$ failed because a selected eigenvalue was too close to an eigenvalue which was not selected; this error exit can only occur if at least one of the eigenvalues involved was complex. The problem is too ill-conditioned: consider modifying the selection of eigenvalues so that eigenvalues which are very close together are either all included in the cluster or all excluded. On exit, $T$ may have been partially reordered, but WR, WI and $Q$ (if requested) are updated consistently with $T$; S and SEP (if requested) are both set to zero.

## 7. Accuracy

The computed matrix $\tilde{T}$ is exactly similar to a matrix $T + E$, where

$$\|E\|_2 = O(\varepsilon)\|T\|_2,$$

and $\varepsilon$ is the ***machine precision***.

S cannot underestimate the true reciprocal condition number by more than a factor of $\sqrt{\min(m,n-m)}$. SEP may differ from the true value by $\sqrt{m(n-m)}$. The angle between the computed invariant subspace and the true subspace is $\dfrac{O(\varepsilon)\|A\|_2}{sep}$.

Note that if a 2 by 2 diagonal block is involved in the re-ordering, its off-diagonal elements are in general changed; the diagonal elements and the eigenvalues of the block are unchanged unless the block is sufficiently ill-conditioned, in which case they may be noticeably altered. It is possible for a 2 by 2 block to break into two 1 by 1 blocks, that is, for a pair of complex eigenvalues to become purely real. The values of real eigenvalues however are never changed by the re-ordering.

## 8. Further Comments

The input matrix $T$ must be in canonical Schur form, as is the output matrix $\tilde{T}$. This has the following structure.

If all the computed eigenvalues are real, $\tilde{T}$ is upper triangular, and the diagonal elements of $\tilde{T}$ are the eigenvalues; WR($i$) = $\tilde{t}_{ii}$ for $i = 1,2,...,n$ and WI($i$) = 0.0.

If some of the computed eigenvalues form complex conjugate pairs, then $\tilde{T}$ has 2 by 2 diagonal blocks. Each diagonal block has the form

$$\begin{pmatrix} \tilde{t}_{ii} & \tilde{t}_{i,i+1} \\ \tilde{t}_{i+1,i} & \tilde{t}_{i+1,i+1} \end{pmatrix} = \begin{pmatrix} \alpha & \beta \\ \gamma & \alpha \end{pmatrix}$$

where $\beta\gamma < 0$. The corresponding eigenvalues are $\alpha \pm \sqrt{\beta\gamma}$; WR($i$) = WR($i$+1) = $\alpha$; WI($i$) = $+\sqrt{|\beta\gamma|}$; WI($i$+1) = $-$WI($i$).

The complex analogue of this routine is F08QUF (CTRSEN/ZTRSEN).

## 9. Example

To reorder the Schur factorization of the matrix $A = QTQ^T$ such that the two real eigenvalues appear as the leading elements on the diagonal of the reordered matrix $\tilde{T}$, where

$$T = \begin{pmatrix} 0.7995 & -0.1144 & 0.0060 & 0.0336 \\ 0.0000 & -0.0994 & 0.2478 & 0.3474 \\ 0.0000 & -0.6483 & -0.0994 & 0.2026 \\ 0.0000 & 0.0000 & 0.0000 & -0.1007 \end{pmatrix} \text{ and}$$

$$Q = \begin{pmatrix} 0.6551 & 0.1037 & 0.3450 & 0.6641 \\ 0.5236 & -0.5807 & -0.6141 & -0.1068 \\ -0.5362 & -0.3073 & -0.2935 & 0.7293 \\ 0.0956 & 0.7467 & -0.6463 & 0.1249 \end{pmatrix}.$$

The original matrix $A$ is given in Section 9 of the document for F08NFF (SORGHR/DORGHR).

## 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.**

```
*     F08QGF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDT, LDQ, LWORK, LIWORK
      PARAMETER        (NMAX=8,LDT=NMAX,LDQ=NMAX,LWORK=NMAX*NMAX/2,
     +                 LIWORK=NMAX*NMAX/2)
*     .. Local Scalars ..
      real             S, SEP
      INTEGER          I, IFAIL, INFO, J, M, N
*     .. Local Arrays ..
      real             Q(LDQ,NMAX), T(LDT,NMAX), WI(NMAX), WORK(LWORK),
     +                 WR(NMAX)
      INTEGER          IWORK(LIWORK)
      LOGICAL          SELECT(NMAX)
*     .. External Subroutines ..
      EXTERNAL         strsen, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08QGF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read T and Q from data file
*
         READ (NIN,*) ((T(I,J),J=1,N),I=1,N)
         READ (NIN,*) ((Q(I,J),J=1,N),I=1,N)
*
         READ (NIN,*) (SELECT(I),I=1,N)
*
*        Reorder the Schur factorization
*
         CALL strsen('Both','Vectors',SELECT,N,T,LDT,Q,LDQ,WR,WI,M,S,
     +               SEP,WORK,LWORK,IWORK,LIWORK,INFO)
*
         WRITE (NOUT,*)
         IFAIL = 0
*
         CALL X04CAF('General',' ',N,N,T,LDT,'Reordered Schur form',
     +               IFAIL)
*
         WRITE (NOUT,*)
         IFAIL = 0
*
         CALL X04CAF('General',' ',N,M,Q,LDQ,
     +               'Basis of invariant subspace',IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,99999) 'Condition number estimate',
     +     ' of the selected cluster of eigenvalues = ', 1.0e0/S
         WRITE (NOUT,*)
         WRITE (NOUT,99999) 'Condition number estimate of the spec',
     +     'ified invariant subspace    = ', 1.0e0/SEP
      END IF
      STOP
*
99999 FORMAT (1X,A,A,e10.2)
      END
```

## 9.2. Program Data

```
F08QGF Example Program Data
  4                                          :Value of N
  0.7995  -0.1144   0.0060   0.0336
  0.0000  -0.0994   0.2478   0.3474
  0.0000  -0.6483  -0.0994   0.2026
  0.0000   0.0000   0.0000  -0.1007   :End of matrix T
  0.6551   0.1037   0.3450   0.6641
  0.5236  -0.5807  -0.6141  -0.1068
 -0.5362  -0.3073  -0.2935   0.7293
  0.0956   0.7467  -0.6463   0.1249   :End of matrix Q
  T   F   F   T                         :End of SELECT
```

## 9.3. Program Results

```
F08QGF Example Program Results

Reordered Schur form
        1       2       3       4
1   0.7995 -0.0059  0.0751 -0.0927
2   0.0000 -0.1007 -0.3936  0.3569
3   0.0000  0.0000 -0.0994  0.5128
4   0.0000  0.0000 -0.3133 -0.0994

Basis of invariant subspace
        1       2
1   0.6551  0.1211
2   0.5236  0.3286
3  -0.5362  0.5974
4   0.0956  0.7215

Condition number estimate of the selected cluster of eigenvalues =   0.18E+01

Condition number estimate of the specified invariant subspace    =   0.32E+01
```

# F08QHF (STRSYL/DTRSYL) – 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

F08QHF (STRSYL/DTRSYL) solves the real quasi-triangular Sylvester matrix equation.

## 2. Specification

```
      SUBROUTINE F08QHF (TRANA, TRANB, ISGN, M, N, A, LDA, B, LDB, C, LDC,
     1                   SCALE, INFO)
      ENTRY       strsyl (TRANA, TRANB, ISGN, M, N, A, LDA, B, LDB, C, LDC,
     1                   SCALE, INFO)
      INTEGER           ISGN, M, N, LDA, LDB, LDC, INFO
      real              A(LDA,*), B(LDB,*), C(LDC,*), SCALE
      CHARACTER*1       TRANA, TRANB
```

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

## 3. Description

This routine solves the real Sylvester matrix equation

$$\mathrm{op}(A)X \pm X\mathrm{op}(B) = \alpha C,$$

where $\mathrm{op}(A) = A$ or $A^T$, and the matrices $A$ and $B$ are upper quasi-triangular matrices in canonical Schur form (as returned by F08PEF (SHSEQR/DHSEQR)); $\alpha$ is a scale factor ($\leq 1$) determined by the routine to avoid overflow in $X$; $A$ is $m$ by $m$ and $B$ is $n$ by $n$ while the right-hand side matrix $C$ and the solution matrix $X$ are both $m$ by $n$. The matrix $X$ is obtained by a straightforward process of back substitution (see [1]).

Note that the equation has a unique solution if and only if $\alpha_i \pm \beta_j \neq 0$, where $\{\alpha_i\}$ and $\{\beta_j\}$ are the eigenvalues of $A$ and $B$ respectively and the sign (+ or −) is the same as that used in the equation to be solved.

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §7.2.5 and §7.6.3.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

[2] HIGHAM, N.J.
Perturbation Theory and Backward Error for $AX - XB = C$.
Numerical Analysis Report No. 211, Dept. of Mathematics, University of Manchester, 1992.

## 5. Parameters

1: TRANA – CHARACTER*1. *Input*

*On entry*: specifies the option $\mathrm{op}(A)$ as follows:

if TRANA = 'N', then $\mathrm{op}(A) = A$;

if TRANA = 'T' or 'C', then $\mathrm{op}(A) = A^T$.

*Constraint*: TRANA = 'N', 'T' or 'C'.

2: TRANB – CHARACTER*1. *Input*

*On entry*: specifies the option $\mathrm{op}(B)$ as follows:

if TRANB = 'N', then $\mathrm{op}(B) = B$;

if TRANB = 'T' or 'C', then $\mathrm{op}(B) = B^T$.

*Constraint*: TRANB = 'N', 'T' or 'C'.

3: ISGN – INTEGER. *Input*

*On entry*: indicates the form of the Sylvester equation as follows:

if ISGN = +1, then the equation is of the form $\mathrm{op}(A)X + X\mathrm{op}(B) = \alpha C$;

if ISGN = −1, then the equation is of the form $\mathrm{op}(A)X - X\mathrm{op}(B) = \alpha C$.

*Constraint*: ISGN = ±1.

4: M – INTEGER. *Input*

*On entry*: $m$, the order of the matrix $A$, and the number of rows in the matrices $X$ and $C$.

*Constraint*: M ≥ 0.

5: N – INTEGER. *Input*

*On entry*: $n$, the order of the matrix $B$, and the number of columns in the matrices $X$ and $C$.

*Constraint*: N ≥ 0.

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

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

*On entry*: the $m$ by $m$ upper quasi-triangular matrix $A$ in canonical Schur form, as returned by F08PEF (SHSEQR/DHSEQR).

7: LDA – INTEGER. *Input*

*On entry*: the first dimension of the array A as declared in the (sub)program from which F08QHF (STRSYL/DTRSYL) is called.

*Constraint*: LDA ≥ max(1,M).

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

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

*On entry*: the $n$ by $n$ upper quasi-triangular matrix $B$ in canonical Schur form, as returned by F08PEF (SHSEQR/DHSEQR).

9: LDB – INTEGER. *Input*

*On entry*: the first dimension of the array B as declared in the (sub)program from which F08QHF (STRSYL/DTRSYL) is called.

*Constraint*: LDB ≥ max(1,N).

10: 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 $n$ right-hand side matrix $C$.

*On exit*: $C$ is overwritten by the solution matrix $X$.

11: LDC – INTEGER. *Input*

*On entry*: the first dimension of the array C as declared in the (sub)program from which F08QHF (STRSYL/DTRSYL) is called.

*Constraint*: LDC ≥ max(1,M).

12: SCALE – ***real***. *Output*

*On exit*: the value of the scale factor $\alpha$.

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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO = 1

$A$ and $B$ have common or close eigenvalues, perturbed values of which were used to solve the equation.

## 7. Accuracy

Consider the equation $AX - XB = C$. (To apply the remarks to the equation $AX + XB = C$, simply replace $B$ by $-B$).

Let $\tilde{X}$ be the computed solution and $R$ the residual matrix:

$$R = C - (A\tilde{X} - \tilde{X}B).$$

Then the residual is always small:

$$\|R\|_F = O(\varepsilon)(\|A\|_F + \|B\|_F)\|\tilde{X}\|_F.$$

However, $\tilde{X}$ is **not** necessarily the exact solution of a slightly perturbed equation; in other words, the solution is not backwards stable.

For the forward error, the following bound holds:

$$\|\tilde{X} - X\|_F \leq \frac{\|R\|_F}{sep(A,B)}$$

but this may be a considerable overestimate. See Golub and Van Loan [1] for a definition of $sep(A,B)$, and Higham [2] for further details.

These remarks also apply to the solution of a general Sylvester equation, as described in Section 8.

## 8. Further Comments

The total number of floating-point operations is approximately $mn(m+n)$.

To solve the **general** real Sylvester equation

$$AX \pm XB = C$$

where $A$ and $B$ are general nonsymmetric matrices, $A$ and $B$ must first be reduced to Schur form (by calling F02EAF, for example):

$$A = Q_1 \tilde{A} Q_1^T \text{ and } B = Q_2 \tilde{B} Q_2^T$$

where $\tilde{A}$ and $\tilde{B}$ are upper quasi-triangular and $Q_1$ and $Q_2$ are orthogonal. The original equation may then be transformed to:

$$\tilde{A}\tilde{X} \pm \tilde{X}\tilde{B} = \tilde{C}$$

where $\tilde{X} = Q_1^T X Q_2$ and $\tilde{C} = Q_1^T C Q_2$. $\tilde{C}$ may be computed by matrix multiplication; F08QHF (STRSYL/DTRSYL) may be used to solve the transformed equation; and the solution to the original equation can be obtained as $X = Q_1 \tilde{X} Q_2^T$.

The complex analogue of this routine is F08QVF (CTRSYL/ZTRSYL).

## 9. Example

To solve the Sylvester equation

$$AX + XB = C,$$

where

$$A = \begin{pmatrix} 0.10 & 0.50 & 0.68 & -0.21 \\ -0.50 & 0.10 & -0.24 & 0.67 \\ 0.00 & 0.00 & 0.19 & -0.35 \\ 0.00 & 0.00 & 0.00 & -0.72 \end{pmatrix}, \quad B = \begin{pmatrix} -0.99 & -0.17 & 0.39 & 0.58 \\ 0.00 & 0.48 & -0.84 & -0.15 \\ 0.00 & 0.00 & 0.75 & 0.25 \\ 0.00 & 0.00 & -0.25 & 0.75 \end{pmatrix} \text{ and}$$

$$C = \begin{pmatrix} 0.63 & -0.56 & 0.08 & -0.23 \\ -0.45 & -0.31 & 0.27 & 1.21 \\ 0.20 & -0.35 & 0.41 & 0.84 \\ 0.49 & -0.05 & -0.52 & -0.08 \end{pmatrix}.$$

### 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.**

```
*     F08QHF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          MMAX, NMAX, LDA, LDB, LDC
      PARAMETER        (MMAX=8,NMAX=8,LDA=MMAX,LDB=NMAX,LDC=MMAX)
*     .. Local Scalars ..
      real             SCALE
      INTEGER          I, IFAIL, INFO, J, M, N
*     .. Local Arrays ..
      real             A(LDA,MMAX), B(LDB,NMAX), C(LDC,NMAX)
*     .. External Subroutines ..
      EXTERNAL         strsyl, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08QHF 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, B and C from data file
*
         READ (NIN,*) ((A(I,J),J=1,M),I=1,M)
         READ (NIN,*) ((B(I,J),J=1,N),I=1,N)
         READ (NIN,*) ((C(I,J),J=1,N),I=1,M)
*
*        Solve the Sylvester equation A*X + X*B = C for X
*
         CALL strsyl('No transpose','No transpose',1,M,N,A,LDA,B,LDB,C,
     +               LDC,SCALE,INFO)
*
*        Print the solution matrix X
*
         WRITE (NOUT,*)
         IFAIL = 0
*
         CALL X04CAF('General',' ',M,N,C,LDC,'Solution matrix X',IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,99999) 'SCALE = ', SCALE
      END IF
      STOP
*
99999 FORMAT (1X,A,1P,e10.2)
      END
```

## 9.2. Program Data

```
F08QHF Example Program Data
  4  4                                  :Values of M and N
  0.10   0.50   0.68  -0.21
 -0.50   0.10  -0.24   0.67
  0.00   0.00   0.19  -0.35
  0.00   0.00   0.00  -0.72   :End of matrix A
 -0.99  -0.17   0.39   0.58
  0.00   0.48  -0.84  -0.15
  0.00   0.00   0.75   0.25
  0.00   0.00  -0.25   0.75   :End of matrix B
  0.63  -0.56   0.08  -0.23
 -0.45  -0.31   0.27   1.21
  0.20  -0.35   0.41   0.84
  0.49  -0.05  -0.52  -0.08   :End of matrix C
```

## 9.3. Program Results

```
F08QHF Example Program Results

Solution matrix X
         1       2       3       4
1  -0.4209  0.1764  0.2438 -0.9577
2   0.5600 -0.8337 -0.7221  0.5386
3  -0.1246 -0.3392  0.6221  0.8691
4  -0.2865  0.4113  0.5535  0.3174

SCALE =   1.00E+00
```

# F08QKF (STREVC/DTREVC) – 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

F08QKF (STREVC/DTREVC) computes selected left and/or right eigenvectors of a real upper quasi-triangular matrix.

## 2. Specification

```
 SUBROUTINE F08QKF (JOB, HOWMNY, SELECT, N, T, LDT, VL, LDVL, VR, LDVR,
1                   MM, M, WORK, INFO)
 ENTRY       strevc  (JOB, HOWMNY, SELECT, N, T, LDT, VL, LDVL, VR, LDVR,
1                   MM, M, WORK, INFO)
 INTEGER          N, LDT, LDVL, LDVR, MM, M, INFO
 real             T(LDT,*), VL(LDVL,*), VR(LDVR,*), WORK(*)
 LOGICAL          SELECT(*)
 CHARACTER*1      JOB, HOWMNY
```

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

## 3. Description

This routine computes left and/or right eigenvectors of a real upper quasi-triangular matrix $T$ in canonical Schur form. Such a matrix arises from the Schur factorization of a real general matrix, as computed by F08PEF (SHSEQR/DHSEQR), for example.

The right eigenvector $x$, and the left eigenvector $y$, corresponding to an eigenvalue $\lambda$, are defined by:

$$Tx = \lambda x \text{ and } y^H T = \lambda y^H \text{ (or } T^T y = \bar{\lambda} y).$$

Note that even though $T$ is real, $\lambda$, $x$ and $y$ may be complex. If $x$ is an eigenvector corresponding to a complex eigenvalue $\lambda$, then the complex conjugate vector $\bar{x}$ is the eigenvector corresponding to the complex conjugate eigenvalue $\bar{\lambda}$.

The routine can compute the eigenvectors corresponding to selected eigenvalues, or it can compute all the eigenvectors. In the latter case the eigenvectors may optionally be pre-multiplied by an input matrix $Q$. Normally $Q$ is an orthogonal matrix from the Schur factorization of a matrix $A$ as $A = QTQ^T$; if $x$ is a (left or right) eigenvector of $T$, then $Qx$ is an eigenvector of $A$.

The eigenvectors are computed by forward or backward substitution. They are scaled so that, for a real eigenvector $x$, $\max(|x_i|) = 1$, and for a complex eigenvector, $\max(|Re(x_i)|+|Im(x_i)|) = 1$.

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §7.6.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

## 5. Parameters

1: JOB – CHARACTER*1. *Input*

*On entry*: indicates whether left and/or right eigenvectors are to be computed as follows:

if JOB = 'R', then only right eigenvectors are computed;

if JOB = 'L', then only left eigenvectors are computed;

if JOB = 'B', then both left and right eigenvectors are computed.

*Constraint*: JOB = 'R', 'L' or 'B'.

2: HOWMNY – CHARACTER*1. *Input*

*On entry*: indicates how many eigenvectors are to be computed as follows:

if HOWMNY = 'A', then all eigenvectors (as specified by JOB) are computed;

if HOWMNY = 'O', then all eigenvectors (as specified by JOB) are computed and then pre-multiplied by the matrix $Q$ (which is overwritten);

if HOWMNY = 'S', then selected eigenvectors (as specified by JOB and SELECT) are computed.

*Constraint*: HOWMNY = 'A', 'O' or 'S'.

3: SELECT(*) – LOGICAL array. *Input/Output*

**Note**: the dimension of the array SELECT must be at least max(1,N) if HOWMNY = 'S' and at least 1 otherwise.

*On entry*: SELECT specifies which eigenvectors are to be computed if HOWMNY = 'S'. To obtain the real eigenvector corresponding to the real eigenvalue $\lambda_j$, SELECT($j$) must be set .TRUE.. To select the complex eigenvector corresponding to a complex conjugate pair of eigenvalues $\lambda_j$ and $\lambda_{j+1}$, SELECT($j$) and/or SELECT($j$+1) must be set .TRUE.; the eigenvector corresponding to the **first** eigenvalue in the pair is computed.

*On exit*: if a complex eigenvector was selected as specified above, then SELECT($j$) is set to .TRUE. and SELECT($j$+1) to .FALSE..

SELECT is not referenced if HOWMNY = 'A' or 'O'.

4: N – INTEGER. *Input*

*On entry*: $n$, the order of the matrix $T$.

*Constraint*: N ≥ 0.

5: T(LDT,*) – ***real*** array. *Input*

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

*On entry*: the $n$ by $n$ upper quasi-triangular matrix $T$ in canonical Schur form, as returned by F08PEF (SHSEQR/DHSEQR).

6: LDT – INTEGER. *Input*

*On entry*: the first dimension of the array T as declared in the (sub)program from which F08QKF (STREVC/DTREVC) is called.

*Constraint*: LDT ≥ max(1,N).

7: VL(LDVL,*) – ***real*** array. *Input/Output*

**Note**: the second dimension of the array VL must be at least max(1,MM) if JOB = 'L' or 'B' and at least 1 if JOB = 'R'.

*On entry*: if HOWMNY = 'O' and JOB = 'L' or 'B', VL must contain an $n$ by $n$ matrix $Q$ (usually the matrix of Schur vectors returned by F08PEF (SHSEQR/DHSEQR)). If HOWMNY = 'A' or 'S', VL need not be set.

*On exit*: if JOB = 'L' or 'B', VL contains the computed left eigenvectors (as specified by HOWMNY and SELECT). The eigenvectors are stored consecutively in the columns of the array, in the same order as their eigenvalues. Corresponding to each real eigenvalue is a real eigenvector, occupying one column. Corresponding to each complex conjugate pair of eigenvalues, is a complex eigenvector occupying two columns; the first column holds the real part and the second column holds the imaginary part.

VL is not referenced if JOB = 'R'.

8: LDVL – INTEGER. *Input*

*On entry*: the first dimension of the array VL as declared in the (sub)program from which F08QKF (STREVC/DTREVC) is called.

*Constraints*: LDVL ≥ max(1,N) if JOB = 'L' or 'B',
LDVL ≥ 1 if JOB = 'R'.

9: VR(LDVR,*) – ***real*** array. *Input/Output*

**Note**: the second dimension of the array VR must be at least max(1,MM) if JOB = 'R' or 'B' and at least 1 if JOB = 'L'.

*On entry*: if HOWMNY = 'O' and JOB = 'R' or 'B', VR must contain an $n$ by $n$ matrix $Q$ (usually the matrix of Schur vectors returned by F08PEF (SHSEQR/DHSEQR)). If HOWMNY = 'A' or 'S', VR need not be set.

*On exit*: if JOB = 'R' or 'B', VR contains the computed right eigenvectors (as specified by HOWMNY and SELECT). The eigenvectors are stored consecutively in the columns of the array, in the same order as their eigenvalues. Corresponding to each real eigenvalue is a real eigenvector, occupying one column. Corresponding to each complex conjugate pair of eigenvalues, is a complex eigenvector occupying two columns; the first column holds the real part and the second column holds the imaginary part.

VR is not referenced if JOB = 'L'.

10: LDVR – INTEGER. *Input*

*On entry*: the first dimension of the array VR as declared in the (sub)program from which F08QKF (STREVC/DTREVC) is called.

*Constraints*: LDVR ≥ max(1,N) if JOB = 'R' or 'B',
LDVR ≥ 1 if JOB = 'L'.

11: MM – INTEGER. *Input*

*On entry*: the number of columns in the arrays VL and/or VR. The precise number of columns required, $m$, is $n$ if HOWMNY = 'A' or 'O'; if HOWMNY = 'S', $m$ is obtained by counting 1 for each selected real eigenvector and 2 for each selected complex eigenvector (see SELECT), in which case $0 \leq m \leq n$.

*Constraint*: MM ≥ $m$.

12: M – INTEGER. *Output*

*On exit*: $m$, the number of columns of VL and/or VR actually used to store the computed eigenvectors. If HOWMNY = 'A' or 'O', M is set to $n$.

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

**Note**: the dimension of the array WORK must be at least max(1,3*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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

## 7. Accuracy

If $x_i$ is an exact right eigenvector, and $\tilde{x}_i$ is the corresponding computed eigenvector, then the angle $\theta(\tilde{x}_i, x_i)$ between them is bounded as follows:

$$\theta(\tilde{x}_i, x_i) \leq \frac{c(n)\varepsilon\|T\|_2}{sep_i}$$

where $sep_i$ is the reciprocal condition number of $x_i$.

The condition number $sep_i$ may be computed by calling F08QLF (STRSNA/DTRSNA).

## 8. Further Comments

For a description of canonical Schur form, see the document for F08PEF (SHSEQR/DHSEQR).

The complex analogue of this routine is F08QXF (CTREVC/ZTREVC).

## 9. Example

See the example for F08NHF (SGEBAL/DGEBAL).

# F08QLF (STRSNA/DTRSNA) – 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

F08QLF (STRSNA/DTRSNA) estimates condition numbers for specified eigenvalues and/or right eigenvectors of a real upper quasi-triangular matrix.

## 2. Specification

```
 SUBROUTINE F08QLF (JOB, HOWMNY, SELECT, N, T, LDT, VL, LDVL, VR, LDVR,
1                   S, SEP, MM, M, WORK, LDWORK, IWORK, INFO)
 ENTRY       strsna  (JOB, HOWMNY, SELECT, N, T, LDT, VL, LDVL, VR, LDVR,
1                   S, SEP, MM, M, WORK, LDWORK, IWORK, INFO)
 INTEGER          N, LDT, LDVL, LDVR, MM, M, LDWORK, IWORK(*), INFO
 real             T(LDT,*), VL(LDVL,*), VR(LDVR,*), S(*), SEP(*),
1                 WORK(LDWORK,*)
 LOGICAL          SELECT(*)
 CHARACTER*1      JOB, HOWMNY
```

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

## 3. Description

This routine estimates condition numbers for specified eigenvalues and/or right eigenvectors of a real upper quasi-triangular matrix $T$ in canonical Schur form. These are the same as the condition numbers of the eigenvalues and right eigenvectors of an original matrix $A = ZTZ^T$ (with orthogonal $Z$), from which $T$ may have been derived.

F08QLF computes the reciprocal of the condition number of an eigenvalue $\lambda_i$ as

$$s_i = \frac{|v^H u|}{\|u\|_E \|v\|_E},$$

where $u$ and $v$ are the right and left eigenvectors of $T$, respectively, corresponding to $\lambda_i$. This reciprocal condition number always lies between zero (i.e. ill-conditioned) and one (i.e. well-conditioned).

An approximate error estimate for a computed eigenvalue $\lambda_i$ is then given by

$$\frac{\varepsilon \|T\|}{s_i},$$

where $\varepsilon$ is the ***machine precision***.

To estimate the reciprocal of the condition number of the right eigenvector corresponding to $\lambda_i$, the routine first calls F08QFF (STREXC/DTREXC) to reorder the eigenvalues so that $\lambda_i$ is in the leading position:

$$T = Q \begin{pmatrix} \lambda_i & c^T \\ 0 & T_{22} \end{pmatrix} Q^T.$$

The reciprocal condition number of the eigenvector is then estimated as $sep_i$, the smallest singular value of the matrix $(T_{22} - \lambda_i I)$. This number ranges from zero (i.e. ill-conditioned) to very large (i.e. well-conditioned).

An approximate error estimate for a computed right eigenvector $u$ corresponding to $\lambda_i$ is then given by

$$\frac{\varepsilon \|T\|}{sep_i}.$$

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §7.2.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

## 5. Parameters

1: JOB – CHARACTER*1. *Input*

*On entry*: indicates whether condition numbers are required for eigenvalues and/or eigenvectors, as follows:

if JOB = 'E', then condition numbers for eigenvalues only are computed;

if JOB = 'V', then condition numbers for eigenvectors only are computed;

if JOB = 'B', then condition numbers for both eigenvalues and eigenvectors are computed.

*Constraint*: JOB = 'E', 'V' or 'B'.

2: HOWMNY – CHARACTER*1. *Input*

*On entry*: indicates how many condition numbers are to be computed, as follows:

if HOWMNY = 'A', then condition numbers for all eigenpairs are computed;

if HOWMNY = 'S', then condition numbers for selected eigenpairs (as specified by SELECT) are computed.

*Constraint*: HOWMNY = 'A' or 'S'.

3: SELECT(*) – LOGICAL array. *Input*

**Note**: the dimension of the array SELECT must be at least max(1,N) if HOWMNY = 'S' and at least 1 otherwise.

*On entry*: SELECT specifies the eigenpairs for which condition numbers are to be computed if HOWMNY = 'S'. To select condition numbers for the eigenpair corresponding to the real eigenvalue $\lambda_j$, SELECT($j$) must be set .TRUE.. To select condition numbers corresponding to a complex conjugate pair of eigenvalues $\lambda_j$ and $\lambda_{j+1}$, SELECT($j$) and/or SELECT($j$+1) must be set .TRUE..

SELECT is not referenced if HOWMNY = 'A'.

4: N – INTEGER. *Input*

*On entry*: $n$, the order of the matrix $T$.

*Constraint*: N ≥ 0.

5: T(LDT,*) – ***real*** array. *Input*

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

*On entry*: the $n$ by $n$ upper quasi-triangular matrix $T$ in canonical Schur form, as returned by F08PEF (SHSEQR/DHSEQR).

6: LDT – INTEGER. *Input*

*On entry*: the first dimension of the array T as declared in the (sub)program from which F08QLF (STRSNA/DTRSNA) is called.

*Constraint*: LDT ≥ max(1,N).

7: VL(LDVL,*) – ***real*** array. *Input*

**Note**: the second dimension of the array VL must be at least max(1,MM) if JOB = 'E' or 'B' and at least 1 if JOB = 'V'.

*On entry*: if JOB = 'E' or 'B', VL must contain the left eigenvectors of $T$ (or of any matrix $QTQ^T$ with $Q$ orthogonal) corresponding to the eigenpairs specified by HOWMNY and SELECT. The eigenvectors **must** be stored in consecutive columns of VL, as returned by F08QKF (STREVC/DTREVC) or F08PKF (SHSEIN/DHSEIN).

VL is not referenced if JOB = 'V'.

8: LDVL – INTEGER. *Input*

*On entry*: the first dimension of the array VL as declared in the (sub)program from which F08QLF (STRSNA/DTRSNA) is called.

*Constraints*: LDVL ≥ max(1,N) if JOB = 'E' or 'B',
LDVL ≥ 1 if JOB = 'V'.

9: VR(LDVR,*) – ***real*** array. *Input*

**Note**: the second dimension of the array VR must be at least max(1,MM) if JOB = 'E' or 'B' and at least 1 if JOB = 'V'.

*On entry*: if JOB = 'E' or 'B', VR must contain the right eigenvectors of $T$ (or of any matrix $QTQ^T$ with $Q$ orthogonal) corresponding to the eigenpairs specified by HOWMNY and SELECT. The eigenvectors **must** be stored in consecutive columns of VR, as returned by F08QKF (STREVC/DTREVC) or F08PKF (SHSEIN/DHSEIN).

VR is not referenced if JOB = 'V'.

10: LDVR – INTEGER. *Input*

*On entry*: the first dimension of the array VR as declared in the (sub)program from which F08QLF (STRSNA/DTRSNA) is called.

*Constraints*: LDVR ≥ max(1,N) if JOB = 'E' or 'B',
LDVR ≥ 1 if JOB = 'V'.

11: S(*) – ***real*** array. *Output*

**Note**: the dimension of the array S must be at least max(1,MM) if JOB = 'E' or 'B' and at least 1 if JOB = 'V'.

*On exit*: the reciprocal condition numbers of the selected eigenvalues if JOB = 'E' or 'B', stored in consecutive elements of the array. Thus S($j$), SEP($j$) and the $j$th columns of VL and VR all correspond to the same eigenpair (but not in general the $j$th eigenpair unless all eigenpairs have been selected). For a complex conjugate pair of eigenvalues, two consecutive elements of S are set to the same value.

S is not referenced if JOB = 'V'.

12: SEP(*) – ***real*** array. *Output*

**Note**: the dimension of the array SEP must be at least max(1,MM) if JOB = 'V' or 'B' and at least 1 if JOB = 'E'.

*On exit*: the estimated reciprocal condition numbers of the selected right eigenvectors if JOB = 'V' or 'B', stored in consecutive elements of the array. For a complex eigenvector, two consecutive elements of SEP are set to the same value. If the eigenvalues cannot be reordered to compute SEP($j$), then SEP($j$) is set to zero; this can only occur when the true value would be very small anyway.

SEP is not referenced if JOB = 'E'.

13: MM – INTEGER. *Input*

*On entry*: the number of elements in the arrays S and SEP, and the number of columns in the arrays VL and VR (if used). The precise number required, $m$, is $n$ if HOWMNY = 'A'; if HOWMNY = 'S', $m$ is obtained by counting 1 for each selected real eigenvalue, and 2 for each selected complex conjugate pair of eigenvalues (see SELECT), in which case $0 \leq m \leq n$.

*Constraint*: MM ≥ M.

14: M – INTEGER. *Output*

*On exit*: $m$, the number of elements of S and/or SEP actually used to store the estimated condition numbers. If HOWMNY = 'A', M is set to $n$.

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

**Note**: the second dimension of the array WORK must be at least max(1,N+6) if JOB = 'V' or 'B' and at least 1 if JOB = 'E'.

WORK is not referenced if JOB = 'E'.

16: LDWORK – INTEGER. *Input*

*On entry*: the first dimension of the array WORK as declared in the (sub)program from which F08QLF (STRSNA/DTRSNA) is called.

*Constraints*: LDWORK ≥ max(1,N) if JOB = 'V' or 'B',
LDWORK ≥ 1 if JOB = 'E'.

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

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

18: 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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

## 7. Accuracy

The computed values $sep_i$ may overestimate the true value, but seldom by a factor of more than 3.

## 8. Further Comments

For a description of canonical Schur form, see the document for F08PEF (SHSEQR/DHSEQR).

The complex analogue of this routine is F08QYF (CTRSNA/ZTRSNA).

## 9. Example

To compute approximate error estimates for all the eigenvalues and right eigenvectors of the matrix $T$, where

$$T = \begin{pmatrix} 0.7995 & -0.1144 & 0.0060 & 0.0336 \\ 0.0000 & -0.0994 & 0.2478 & 0.3474 \\ 0.0000 & -0.6483 & -0.0994 & 0.2026 \\ 0.0000 & 0.0000 & 0.0000 & -0.1007 \end{pmatrix}.$$

## 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.**

```
*     F08QLF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDT, LDWORK, LDVL, LDVR
      PARAMETER        (NMAX=8,LDT=NMAX,LDWORK=NMAX,LDVL=NMAX,LDVR=NMAX)
*     .. Local Scalars ..
      real             EPS, TNORM
      INTEGER          I, INFO, J, M, N
*     .. Local Arrays ..
      real             S(NMAX), SEP(NMAX), T(LDT,NMAX), VL(LDVL, NMAX),
     +                 VR(LDVR,NMAX), WORK(LDWORK,NMAX+6)
      INTEGER          IWORK(2*NMAX-2)
      LOGICAL          SELECT(1)
*     .. External Functions ..
      real             F06RAF, X02AJF
      EXTERNAL         F06RAF, X02AJF
*     .. External Subroutines ..
      EXTERNAL         strevc, strsna
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08QLF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read T from data file
*
         READ (NIN,*) ((T(I,J),J=1,N),I=1,N)
*
*        Calculate the left and right eigenvectors of T
*
         CALL strevc('Both','All',SELECT,N,T,LDT,VL,LDVL,VR,LDVR,N,M,
     +               WORK,INFO)
*
*        Estimate condition numbers for all the eigenvalues and right
*        eigenvectors of T
*
         CALL strsna('Both','All',SELECT,N,T,LDT,VL,LDVL,VR,LDVR,S,SEP,
     +               N,M,WORK,LDWORK,IWORK,INFO)
*
*        Print condition numbers of eigenvalues and right eigenvectors
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'S'
         WRITE (NOUT,99999) (S(I),I=1,M)
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'SEP'
         WRITE (NOUT,99999) (SEP(I),I=1,M)
*
*        Calculate approximate error estimates (using the 1-norm)
*
         EPS = X02AJF()
         TNORM = F06RAF('1-norm',N,N,T,LDT,WORK)
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Approximate error estimates for eigenvalues ',
     +     'of T (machine-dependent)'
         WRITE (NOUT,99999) (EPS*TNORM/S(I),I=1,M)
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Approximate error estimates for right ',
     +     'eigenvectors of T (machine-dependent)'
         WRITE (NOUT,99999) (EPS*TNORM/SEP(I),I=1,M)
      END IF
```

```
      STOP
*
99999 FORMAT ((3X,1P,7e11.1))
      END
```

## 9.2. Program Data

```
F08QLF Example Program Data
  4                                        :Value of N
  0.7995  -0.1144   0.0060   0.0336
  0.0000  -0.0994   0.2478   0.3474
  0.0000  -0.6483  -0.0994   0.2026
  0.0000   0.0000   0.0000  -0.1007   :End of matrix T
```

## 9.3. Program Results

```
 F08QLF Example Program Results

 S
       9.9E-01    7.0E-01    7.0E-01    5.7E-01

 SEP
       6.3E-01    3.7E-01    3.7E-01    3.1E-01

 Approximate error estimates for eigenvalues of T (machine-dependent)
       9.6E-17    1.4E-16    1.4E-16    1.7E-16

 Approximate error estimates for right eigenvectors of T (machine-dependent)
       1.5E-16    2.6E-16    2.6E-16    3.1E-16
```

# F08QTF (CTREXC/ZTREXC) – 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

F08QTF (CTREXC/ZTREXC) reorders the Schur factorization of a complex general matrix.

## 2. Specification

```
SUBROUTINE F08QTF (COMPQ, N, T, LDT, Q, LDQ, IFST, ILST, INFO)
ENTRY      ctrexc  (COMPQ, N, T, LDT, Q, LDQ, IFST, ILST, INFO)
INTEGER          N, LDT, LDQ, IFST, ILST, INFO
complex          T(LDT,*), Q(LDQ,*)
CHARACTER*1      COMPQ
```

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

## 3. Description

This routine reorders the Schur factorization of a complex general matrix $A = QTQ^H$, so that the diagonal element of $T$ with row index IFST is moved to row ILST.

The reordered Schur form $\tilde{T}$ is computed by a unitary similarity transformation: $\tilde{T} = Z^H T Z$. Optionally the updated matrix $\tilde{Q}$ of Schur vectors is computed as $\tilde{Q} = QZ$, giving $A = \tilde{Q}\tilde{T}\tilde{Q}^H$.

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §7.6.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

## 5. Parameters

1: COMPQ – CHARACTER*1. *Input*

*On entry*: indicates whether the matrix $Q$ of Schur vectors is to be updated, as follows:

if COMPQ = 'V', then the matrix $Q$ of Schur vectors is updated;

if COMPQ = 'N', then no Schur vectors are updated.

*Constraint*: COMPQ = 'V' or 'N'.

2: N – INTEGER. *Input*

*On entry*: $n$, the order of the matrix $T$.

*Constraint*: N ≥ 0.

3: T(LDT,*) – ***complex*** array. *Input/Output*

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

*On entry*: the $n$ by $n$ upper triangular matrix $T$, as returned by F08PSF (CHSEQR/ZHSEQR).

*On exit*: $T$ is overwritten by the updated matrix $\tilde{T}$.

4: LDT – INTEGER. *Input*

*On entry*: the first dimension of the array T as declared in the (sub)program from which F08QTF (CTREXC/ZTREXC) is called.

*Constraint*: LDT ≥ max(1,N).

5: Q(LDQ,*) – ***complex*** array. *Input/Output*

**Note**: the second dimension of the array Q must be at least max(1,N) if COMPQ = 'V' and at least 1 if COMPQ = 'N'.

*On entry*: if COMPQ = 'V', Q must contain the $n$ by $n$ unitary matrix $Q$ of Schur vectors.

*On exit*: if COMPQ = 'V', Q contains the updated matrix of Schur vectors.

Q is not referenced if COMPQ = 'N'.

6: LDQ – INTEGER. *Input*

*On entry*: the first dimension of the array Q as declared in the (sub)program from which F08QTF (CTREXC/ZTREXC) is called.

*Constraints*: LDQ ≥ max(1,N) if COMPQ = 'V',
LDQ ≥ 1 if COMPQ = 'N'.

7: IFST – INTEGER. *Input*

8: ILST – INTEGER. *Input*

*On entry*: IFST and ILST must specify the reordering of the diagonal elements of $T$. The element with row index IFST is moved to row ILST by a sequence of exchanges between adjacent elements.

*Constraints*: 1 ≤ IFST ≤ N,
1 ≤ ILST ≤ 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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

## 7. Accuracy

The computed matrix $\tilde{T}$ is exactly similar to a matrix $T + E$, where

$$\|E\|_2 = O(\varepsilon)\|T\|_2,$$

and $\varepsilon$ is the ***machine precision***.

The values of the eigenvalues are never changed by the re-ordering.

## 8. Further Comments

The total number of real floating-point operations is approximately $20nr$ if COMPQ = 'N', and $40nr$ if COMPQ = 'V', where $r$ = |IFST–ILST|.

The real analogue of this routine is F08QFF (STREXC/DTREXC).

## 9. Example

To reorder the Schur factorization of the matrix $T$ so that element $t_{11}$ is moved to $t_{44}$, where

$$T = \begin{pmatrix} -6.00 - 7.00i & 0.36 - 0.36i & -0.19 + 0.48i & 0.88 - 0.25i \\ 0.00 + 0.00i & -5.00 + 2.00i & -0.03 - 0.72i & -0.23 + 0.13i \\ 0.00 + 0.00i & 0.00 + 0.00i & 8.00 - 1.00i & 0.94 + 0.53i \\ 0.00 + 0.00i & 0.00 + 0.00i & 0.00 + 0.00i & 3.00 - 4.00i \end{pmatrix}.$$

## 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.**

```
*     F08QTF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDT, LDQ
      PARAMETER        (NMAX=8,LDT=NMAX,LDQ=1)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, IFST, ILST, INFO, J, N
*     .. Local Arrays ..
      complex          Q(LDQ,1), T(LDT,NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         X04DBF, ctrexc
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08QTF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read T from data file
*
         READ (NIN,*) ((T(I,J),J=1,N),I=1,N)
*
         READ (NIN,*) IFST, ILST
*
*        Reorder the Schur factorization T
*
         CALL ctrexc('No update',N,T,LDT,Q,LDQ,IFST,ILST,INFO)
*
*        Print reordered Schur form
*
         WRITE (NOUT,*)
         IFAIL = 0
*
         CALL X04DBF('General',' ',N,N,T,LDT,'Bracketed','F7.4',
     +               'Reordered Schur form','Integer',RLABS,'Integer',
     +               CLABS,80,0,IFAIL)
*
      END IF
      STOP
      END
```

## 9.2. Program Data

```
F08QTF Example Program Data
  4                                                          :Value of N
 (-6.00,-7.00) ( 0.36,-0.36) (-0.19, 0.48) ( 0.88,-0.25)
 ( 0.00, 0.00) (-5.00, 2.00) (-0.03,-0.72) (-0.23, 0.13)
 ( 0.00, 0.00) ( 0.00, 0.00) ( 8.00,-1.00) ( 0.94, 0.53)
 ( 0.00, 0.00) ( 0.00, 0.00) ( 0.00, 0.00) ( 3.00,-4.00)   :End of matrix T
  1  4                                                    :Values of IFST and ILST
```

## 9.3. Program Results

```
F08QTF Example Program Results

Reordered Schur form
                     1                     2                     3                     4
1  (-5.0000, 2.0000) (-0.1574, 0.7143) ( 0.1781,-0.1913) ( 0.3950, 0.3861)
2  ( 0.0000, 0.0000) ( 8.0000,-1.0000) ( 1.0742, 0.1447) ( 0.2515,-0.3397)
3  ( 0.0000, 0.0000) ( 0.0000, 0.0000) ( 3.0000,-4.0000) ( 0.2264, 0.8962)
4  ( 0.0000, 0.0000) ( 0.0000, 0.0000) ( 0.0000, 0.0000) (-6.0000,-7.0000)
```

# F08QUF (CTRSEN/ZTRSEN) – 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

F08QUF (CTRSEN/ZTRSEN) reorders the Schur factorization of a complex general matrix so that a selected cluster of eigenvalues appears in the leading elements on the diagonal of the Schur form. The routine also optionally computes the reciprocal condition numbers of the cluster of eigenvalues and/or the invariant subspace.

## 2. Specification

```
 SUBROUTINE F08QUF (JOB, COMPQ, SELECT, N, T, LDT, Q, LDQ, W, M, S, SEP,
1                   WORK, LWORK, INFO)
 ENTRY       ctrsen  (JOB, COMPQ, SELECT, N, T, LDT, Q, LDQ, W, M, S, SEP,
1                   WORK, LWORK, INFO)
 INTEGER          N, LDT, LDQ, M, LWORK, INFO
 real             S, SEP
 complex          T(LDT,*), Q(LDQ,*), W(*), WORK(LWORK)
 LOGICAL          SELECT(*)
 CHARACTER*1      JOB, COMPQ
```

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

## 3. Description

This routine reorders the Schur factorization of a complex general matrix $A = QTQ^H$, so that a selected cluster of eigenvalues appears in the leading diagonal elements of the Schur form.

The reordered Schur form $\tilde{T}$ is computed by a unitary similarity transformation: $\tilde{T} = Z^H T Z$. Optionally the updated matrix $\tilde{Q}$ of Schur vectors is computed as $\tilde{Q} = QZ$, giving $A = \tilde{Q}\tilde{T}\tilde{Q}^H$.

Let $\tilde{T} = \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix}$, where the selected eigenvalues are precisely the eigenvalues of the leading $m$ by $m$ submatrix $T_{11}$. Let $\tilde{Q}$ be correspondingly partitioned as $(Q_1 \ Q_2)$ where $Q_1$ consists of the first $m$ columns of $Q$. Then $AQ_1 = Q_1 T_{11}$, and so the $m$ columns of $Q_1$ form an orthonormal basis for the invariant subspace corresponding to the selected cluster of eigenvalues.

Optionally the routine also computes estimates of the reciprocal condition numbers of the average of the cluster of eigenvalues and of the invariant subspace.

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §7.2 and §7.6.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

## 5. Parameters

1: JOB – CHARACTER*1. *Input*

*On entry*: indicates whether condition numbers are required for the cluster of eigenvalues and/or the invariant subspace, as follows:

if JOB = 'N', then no condition numbers are required;

if JOB = 'E', then only the condition number for the cluster of eigenvalues is computed;

if JOB = 'V', then only the condition number for the invariant subspace is computed;

if JOB = 'B', then condition numbers for both the cluster of eigenvalues and the invariant subspace are computed.

*Constraint*: JOB = 'N', 'E', 'V' or 'B'.

2: COMPQ – CHARACTER*1. *Input*

*On entry*: indicates whether the matrix $Q$ of Schur vectors is to be updated, as follows:

if COMPQ = 'V', then the matrix $Q$ of Schur vectors is updated;

if COMPQ = 'N', then no Schur vectors are updated.

*Constraint*: COMPQ = 'V' or 'N'.

3: SELECT(*) – LOGICAL array. *Input*

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

*On entry*: SELECT specifies the eigenvalues in the selected cluster. To select a complex eigenvalue $\lambda_j$, SELECT($j$) must be set .TRUE..

4: N – INTEGER. *Input*

*On entry*: $n$, the order of the matrix $T$.

*Constraint*: N ≥ 0.

5: T(LDT,*) – ***complex*** array. *Input/Output*

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

*On entry*: the $n$ by $n$ upper triangular matrix $T$, as returned by F08PSF (CHSEQR/ZHSEQR).

*On exit*: $T$ is overwritten by the updated matrix $\tilde{T}$.

6: LDT – INTEGER. *Input*

*On entry*: the first dimension of the array T as declared in the (sub)program from which F08QUF (CTRSEN/ZTRSEN) is called.

*Constraint*: LDT ≥ max(1,N).

7: Q(LDQ,*) – ***complex*** array. *Input/Output*

**Note**: the second dimension of the array Q must be at least max(1,N) if COMPQ = 'V' and at least 1 if COMPQ = 'N'.

*On entry*: if COMPQ = 'V', Q must contain the $n$ by $n$ unitary matrix $Q$ of Schur vectors, as returned by F08PSF (CHSEQR/ZHSEQR).

*On exit*: if COMPQ = 'V', Q contains the updated matrix of Schur vectors; the first $m$ columns of Q form an orthonormal basis for the specified invariant subspace.

Q is not referenced if COMPQ = 'N'.

8: LDQ – INTEGER. *Input*

*On entry*: the first dimension of the array Q as declared in the (sub)program from which F08QUF (CTRSEN/ZTRSEN) is called.

*Constraints*: LDQ ≥ max(1,N) if COMPQ = 'V',
LDQ ≥ 1 if COMPQ = 'N'.

9: W(*) – ***complex*** array. *Output*

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

*On exit*: the reordered eigenvalues of $\tilde{T}$. The eigenvalues are stored in the same order as on the diagonal of $\tilde{T}$.

10: M – INTEGER. *Output*

*On exit*: $m$, the dimension of the specified invariant subspace, which is the same as the number of selected eigenvalues (see SELECT); $0 \leq m \leq n$.

11: S – ***real.*** *Output*

*On exit*: if JOB = 'E' or 'B', S is a lower bound on the reciprocal condition number of the average of the selected cluster of eigenvalues. If M = 0 or N, then S = 1.

S is not referenced if JOB = 'N' or 'V'.

12: SEP – ***real.*** *Output*

*On exit*: if JOB = 'V' or 'B', SEP is the estimated reciprocal condition number of the specified invariant subspace. If M = 0 or N, SEP = $\|T\|$.

SEP is not referenced if JOB = 'N' or 'E'.

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

WORK is not referenced if JOB = 'N'.

14: LWORK – INTEGER. *Input*

*On entry*: the dimension of the array WORK as declared in the (sub)program from which F08QUF (CTRSEN/ZTRSEN) is called.

*Constraints*: if JOB = 'N', then LWORK ≥ 1,
if JOB = 'E', then LWORK ≥ max(1,$m$×(N–$m$)),
if JOB = 'V' or 'B', then LWORK ≥ max(1,2×$m$×(N–$m$)).

The actual amount of workspace required cannot exceed $N^2/4$ if JOB = 'E' or $N^2/2$ if JOB = 'V' or 'B'.

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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

## 7. Accuracy

The computed matrix $\tilde{T}$ is exactly similar to a matrix $T + E$, where

$$\|E\|_2 = O(\varepsilon)\|T\|_2,$$

and $\varepsilon$ is the ***machine precision.***

S cannot underestimate the true reciprocal condition number by more than a factor of $\sqrt{\min(m,n-m)}$. SEP may differ from the true value by $\sqrt{m(n-m)}$. The angle between the computed invariant subspace and the true subspace is $\dfrac{O(\varepsilon)\|A\|_2}{sep}$.

The values of the eigenvalues are never changed by the re-ordering.

## 8. Further Comments

The real analogue of this routine is F08QGF (STRSEN/DTRSEN).

## 9. Example

To reorder the Schur factorization of the matrix $A = QTQ^H$ such that the eigenvalues stored in elements $t_{11}$ and $t_{44}$ appear as the leading elements on the diagonal of the reordered matrix $\tilde{T}$, where

$$T = \begin{pmatrix} -6.0004 - 6.9999i & 0.3637 - 0.3656i & -0.1880 + 0.4787i & 0.8785 - 0.2539i \\ 0.0000 + 0.0000i & -5.0000 + 2.0060i & -0.0307 - 0.7217i & -0.2290 + 0.1313i \\ 0.0000 + 0.0000i & 0.0000 + 0.0000i & 7.9982 - 0.9964i & 0.9357 + 0.5359i \\ 0.0000 + 0.0000i & 0.0000 + 0.0000i & 0.0000 + 0.0000i & 3.0023 - 3.9998i \end{pmatrix}$$

and

$$Q = \begin{pmatrix} -0.8347 - 0.1364i & -0.0628 + 0.3806i & 0.2765 - 0.0846i & 0.0633 - 0.2199i \\ 0.0664 - 0.2968i & 0.2365 + 0.5240i & -0.5877 - 0.4208i & 0.0835 + 0.2183i \\ -0.0362 - 0.3215i & 0.3143 - 0.5473i & 0.0576 - 0.5736i & 0.0057 - 0.4058i \\ 0.0086 + 0.2958i & -0.3416 - 0.0757i & -0.1900 - 0.1600i & 0.8327 - 0.1868i \end{pmatrix}.$$

The original matrix $A$ is given in Section 9 of the document for F08NTF (CUNGHR/ZUNGHR).

### 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.**

```
*     F08QUF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDT, LDQ, LWORK
      PARAMETER        (NMAX=8,LDT=NMAX,LDQ=NMAX,LWORK=NMAX*NMAX/2)
*     .. Local Scalars ..
      real             S, SEP
      INTEGER          I, IFAIL, INFO, J, M, N
*     .. Local Arrays ..
      complex          Q(LDQ,NMAX), T(LDT,NMAX), W(NMAX), WORK(LWORK)
      LOGICAL          SELECT(NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         X04DBF, ctrsen
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08QUF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read T and Q from data file
*
         READ (NIN,*) ((T(I,J),J=1,N),I=1,N)
         READ (NIN,*)
         READ (NIN,*) ((Q(I,J),J=1,N),I=1,N)
         READ (NIN,*)
*
         READ (NIN,*) (SELECT(I),I=1,N)
*
*        Reorder the Schur factorization
*
         CALL ctrsen('Both','Vectors',SELECT,N,T,LDT,Q,LDQ,W,M,S,SEP,
     +               WORK,LWORK,INFO)
*
         WRITE (NOUT,*)
         IFAIL = 0
*
```

```
            CALL X04DBF('General',' ',N,N,T,LDT,'Bracketed','F7.4',
     +                  'Reordered Schur form','Integer',RLABS,'Integer',
     +                  CLABS,80,0,IFAIL)
*
            WRITE (NOUT,*)
            IFAIL = 0
*
            CALL X04DBF('General',' ',N,M,Q,LDQ,'Bracketed','F7.4',
     +                  'Basis of invariant subspace','Integer',RLABS,
     +                  'Integer',CLABS,80,0,IFAIL)
*
            WRITE (NOUT,*)
            WRITE (NOUT,99999) 'Condition number estimate',
     +        ' of the selected cluster of eigenvalues = ', 1.0e0/S
            WRITE (NOUT,*)
            WRITE (NOUT,99999) 'Condition number estimate of the spec',
     +        'ified invariant subspace    = ', 1.0e0/SEP
          END IF
          STOP
*
99999 FORMAT (1X,A,A,e10.2)
          END
```

## 9.2. Program Data

```
F08QUF Example Program Data
  4                                                                 :Value of N
 (-6.0004,-6.9999) ( 0.3637,-0.3656) (-0.1880, 0.4787) ( 0.8785,-0.2539)
 ( 0.0000, 0.0000) (-5.0000, 2.0060) (-0.0307,-0.7217) (-0.2290, 0.1313)
 ( 0.0000, 0.0000) ( 0.0000, 0.0000) ( 7.9982,-0.9964) ( 0.9357, 0.5359)
 ( 0.0000, 0.0000) ( 0.0000, 0.0000) ( 0.0000, 0.0000) ( 3.0023,-3.9998)
                                                          :End of matrix T
 (-0.8347,-0.1364) (-0.0628, 0.3806) ( 0.2765,-0.0846) ( 0.0633,-0.2199)
 ( 0.0664,-0.2968) ( 0.2365, 0.5240) (-0.5877,-0.4208) ( 0.0835, 0.2183)
 (-0.0362,-0.3215) ( 0.3143,-0.5473) ( 0.0576,-0.5736) ( 0.0057,-0.4058)
 ( 0.0086, 0.2958) (-0.3416,-0.0757) (-0.1900,-0.1600) ( 0.8327,-0.1868)
                                                          :End of matrix Q
  T   F   F   T                                             :End of SELECT
```

## 9.3. Program Results

```
F08QUF Example Program Results

Reordered Schur form
                    1                   2                   3                   4
1  (-6.0004,-6.9999) (-0.9433, 0.0086) (-0.4839,-0.2426) ( 0.1539, 0.4000)
2  ( 0.0000, 0.0000) ( 3.0023,-3.9998) ( 0.3028, 0.1519) ( 1.0421,-0.2338)
3  ( 0.0000, 0.0000) ( 0.0000, 0.0000) (-5.0000, 2.0060) ( 0.6891,-0.1546)
4  ( 0.0000, 0.0000) ( 0.0000, 0.0000) ( 0.0000, 0.0000) ( 7.9982,-0.9964)

Basis of invariant subspace
                    1                   2
1  ( 0.8458, 0.0000) ( 0.0488,-0.2073)
2  (-0.0177, 0.3036) ( 0.1275, 0.3006)
3  ( 0.0876, 0.3115) ( 0.0398,-0.2711)
4  (-0.0562,-0.2905) ( 0.8792, 0.0000)

Condition number estimate of the selected cluster of eigenvalues =   0.10E+01

Condition number estimate of the specified invariant subspace    =   0.18E+00
```

# F08QVF (CTRSYL/ZTRSYL) – 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

F08QVF (CTRSYL/ZTRSYL) solves the complex triangular Sylvester matrix equation.

## 2. Specification

```
      SUBROUTINE F08QVF (TRANA, TRANB, ISGN, M, N, A, LDA, B, LDB, C, LDC,
     1                   SCALE, INFO)
      ENTRY       ctrsyl (TRANA, TRANB, ISGN, M, N, A, LDA, B, LDB, C, LDC,
     1                   SCALE, INFO)
      INTEGER           ISGN, M, N, LDA, LDB, LDC, INFO
      real              SCALE
      complex           A(LDA,*), B(LDB,*), C(LDC,*)
      CHARACTER*1       TRANA, TRANB
```

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

## 3. Description

This routine solves the complex Sylvester matrix equation

$$\mathrm{op}(A)X \pm X\mathrm{op}(B) = \alpha C,$$

where $\mathrm{op}(A) = A$ or $A^H$, and the matrices $A$ and $B$ are upper triangular; $\alpha$ is a scale factor ($\leq 1$) determined by the routine to avoid overflow in $X$; $A$ is $m$ by $m$ and $B$ is $n$ by $n$ while the right-hand side matrix $C$ and the solution matrix $X$ are both $m$ by $n$. The matrix $X$ is obtained by a straightforward process of back substitution (see [1]).

Note that the equation has a unique solution if and only if $\alpha_i \pm \beta_i \neq 0$, where $\{\alpha_i\}$ and $\{\beta_i\}$ are the eigenvalues of $A$ and $B$ respectively and the sign (+ or −) is the same as that used in the equation to be solved.

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §7.2.5 and §7.6.3.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

[2] HIGHAM, N.J.
Perturbation Theory and Backward Error for $AX - XB = C$.
Numerical Analysis Report No. 211, Dept. of Mathematics, University of Manchester, 1992.

## 5. Parameters

1: TRANA – CHARACTER*1. *Input*

*On entry*: specifies the option $\mathrm{op}(A)$ as follows:

if TRANA = 'N', then $\mathrm{op}(A) = A$;

if TRANA = 'C', then $\mathrm{op}(A) = A^H$.

*Constraint*: TRANA = 'N' or 'C'.

2: TRANB – CHARACTER*1. *Input*

*On entry*: specifies the option $\mathrm{op}(B)$ as follows:

if TRANB = 'N', then $\mathrm{op}(B) = B$;

if TRANB = 'C', then $\mathrm{op}(B) = B^H$.

*Constraint*: TRANB = 'N' or 'C'.

3: ISGN – INTEGER. *Input*

*On entry*: indicates the form of the Sylvester equation as follows:

if ISGN = +1, then the equation is of the form $\mathrm{op}(A)X + X\mathrm{op}(B) = \alpha C$;

if ISGN = −1, then the equation is of the form $\mathrm{op}(A)X - X\mathrm{op}(B) = \alpha C$.

*Constraint*: ISGN = ±1.

4: M – INTEGER. *Input*

*On entry*: $m$, the order of the matrix $A$, and the number of rows in the matrices $X$ and $C$.

*Constraint*: M ≥ 0.

5: N – INTEGER. *Input*

*On entry*: $n$, the order of the matrix $B$, and the number of columns in the matrices $X$ and $C$.

*Constraint*: N ≥ 0.

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

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

*On entry*: the $m$ by $m$ upper triangular matrix $A$.

7: LDA – INTEGER. *Input*

*On entry*: the first dimension of the array A as declared in the (sub)program from which F08QVF (CTRSYL/ZTRSYL) is called.

*Constraint*: LDA ≥ max(1,M).

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

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

*On entry*: the $n$ by $n$ upper triangular matrix $B$.

9: LDB – INTEGER. *Input*

*On entry*: the first dimension of the array B as declared in the (sub)program from which F08QVF (CTRSYL/ZTRSYL) is called.

*Constraint*: LDB ≥ max(1,N).

10: 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 $n$ right-hand side matrix $C$.

*On exit*: $C$ is overwritten by the solution matrix $X$.

11: LDC – INTEGER. *Input*

*On entry*: the first dimension of the array C as declared in the (sub)program from which F08QVF (CTRSYL/ZTRSYL) is called.

*Constraint*: LDC ≥ max(1,M).

12: SCALE – ***real***. *Output*

*On exit*: the value of the scale factor $\alpha$.

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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO = 1

$A$ and $B$ have common or close eigenvalues, perturbed values of which were used to solve the equation.

## 7. Accuracy

Consider the equation $AX - XB = C$. (To apply the remarks to the equation $AX + XB = C$, simply replace $B$ by $-B$).

Let $\tilde{X}$ be the computed solution and $R$ the residual matrix:

$$R = C - (A\tilde{X} - \tilde{X}B).$$

Then the residual is always small:

$$\|R\|_F = O(\epsilon)(\|A\|_F + \|B\|_F)\|\tilde{X}\|_F.$$

However, $\tilde{X}$ is **not** necessarily the exact solution of a slightly perturbed equation; in other words, the solution is not backwards stable.

For the forward error, the following bound holds:

$$\|\tilde{X} - X\|_F \leq \frac{\|R\|_F}{sep(A,B)}$$

but this may be a considerable overestimate. See Golub and Van Loan [1] for a definition of $sep(A,B)$, and Higham [2] for further details.

These remarks also apply to the solution of a general Sylvester equation, as described in Section 8.

## 8. Further Comments

The total number of real floating-point operations is approximately $4mn(m+n)$.

To solve the **general** complex Sylvester equation

$$AX \pm XB = C$$

where $A$ and $B$ are general general matrices, $A$ and $B$ must first be reduced to Schur form (by calling F02GAF, for example):

$$A = Q_1 \tilde{A} Q_1^H \text{ and } B = Q_2 \tilde{B} Q_2^H$$

where $\tilde{A}$ and $\tilde{B}$ are upper triangular and $Q_1$ and $Q_2$ are unitary. The original equation may then be transformed to:

$$\tilde{A}\tilde{X} \pm \tilde{X}\tilde{B} = \tilde{C}$$

where $\tilde{X} = Q_1^H X Q_2$ and $\tilde{C} = Q_1^H C Q_2$. $\tilde{C}$ may be computed by matrix multiplication; F08QVF (CTRSYL/ZTRSYL) may be used to solve the transformed equation; and the solution to the original equation can be obtained as $X = Q_1 \tilde{X} Q_2^H$.

The real analogue of this routine is F08QHF (STRSYL/DTRSYL).

## 9. Example

To solve the Sylvester equation

$$AX + XB = C,$$

where

$$A = \begin{pmatrix} -6.00 - 7.00i & 0.36 - 0.36i & -0.19 + 0.48i & 0.88 - 0.25i \\ 0.00 + 0.00i & -5.00 + 2.00i & -0.03 - 0.72i & -0.23 + 0.13i \\ 0.00 + 0.00i & 0.00 + 0.00i & 8.00 - 1.00i & 0.94 + 0.53i \\ 0.00 + 0.00i & 0.00 + 0.00i & 0.00 + 0.00i & 3.00 - 4.00i \end{pmatrix},$$

$$B = \begin{pmatrix} 0.50 - 0.20i & -0.29 - 0.16i & -0.37 + 0.84i & -0.55 + 0.73i \\ 0.00 + 0.00i & -0.40 + 0.90i & 0.06 + 0.22i & -0.43 + 0.17i \\ 0.00 + 0.00i & 0.00 + 0.00i & -0.90 - 0.10i & -0.89 - 0.42i \\ 0.00 + 0.00i & 0.00 + 0.00i & 0.00 + 0.00i & 0.30 - 0.70i \end{pmatrix}$$

and

$$C = \begin{pmatrix} 0.63 + 0.35i & 0.45 - 0.56i & 0.08 - 0.14i & -0.17 - 0.23i \\ -0.17 + 0.09i & -0.07 - 0.31i & 0.27 - 0.54i & 0.35 + 1.21i \\ -0.93 - 0.44i & -0.33 - 0.35i & 0.41 - 0.03i & 0.57 + 0.84i \\ 0.54 + 0.25i & -0.62 - 0.05i & -0.52 - 0.13i & 0.11 - 0.08i \end{pmatrix}.$$

### 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.**

```
*     F08QVF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          MMAX, NMAX, LDA, LDB, LDC
      PARAMETER        (MMAX=8,NMAX=8,LDA=MMAX,LDB=NMAX,LDC=MMAX)
*     .. Local Scalars ..
      real             SCALE
      INTEGER          I, IFAIL, INFO, J, M, N
*     .. Local Arrays ..
      complex          A(LDA,MMAX), B(LDB,NMAX), C(LDC,NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         X04DBF, ctrsyl
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08QVF 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, B and C from data file
*
         READ (NIN,*) ((A(I,J),J=1,M),I=1,M)
         READ (NIN,*) ((B(I,J),J=1,N),I=1,N)
         READ (NIN,*) ((C(I,J),J=1,N),I=1,M)
*
*        Solve the Sylvester equation A*X + X*B = C for X
*
         CALL ctrsyl('No transpose','No transpose',1,M,N,A,LDA,B,LDB,C,
     +               LDC,SCALE,INFO)
*
*        Print the solution matrix X
*
         WRITE (NOUT,*)
         IFAIL = 0
```

```
*
         CALL X04DBF('General',' ',M,N,C,LDC,'Bracketed','F7.4',
     +               'Solution matrix X','Integer',RLABS,'Integer',
     +               CLABS,80,0,IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,99999) 'SCALE = ', SCALE
       END IF
       STOP
*
99999 FORMAT (1X,A,1P,e10.2)
      END
```

## 9.2. Program Data

```
F08QVF Example Program Data
  4  4                                                          :Values of M and N
 (-6.00,-7.00) ( 0.36,-0.36) (-0.19, 0.48) ( 0.88,-0.25)
 ( 0.00, 0.00) (-5.00, 2.00) (-0.03,-0.72) (-0.23, 0.13)
 ( 0.00, 0.00) ( 0.00, 0.00) ( 8.00,-1.00) ( 0.94, 0.53)
 ( 0.00, 0.00) ( 0.00, 0.00) ( 0.00, 0.00) ( 3.00,-4.00)    :End of matrix A
 ( 0.50,-0.20) (-0.29,-0.16) (-0.37, 0.84) (-0.55, 0.73)
 ( 0.00, 0.00) (-0.40, 0.90) ( 0.06, 0.22) (-0.43, 0.17)
 ( 0.00, 0.00) ( 0.00, 0.00) (-0.90,-0.10) (-0.89,-0.42)
 ( 0.00, 0.00) ( 0.00, 0.00) ( 0.00, 0.00) ( 0.30,-0.70)    :End of matrix B
 ( 0.63, 0.35) ( 0.45,-0.56) ( 0.08,-0.14) (-0.17,-0.23)
 (-0.17, 0.09) (-0.07,-0.31) ( 0.27,-0.54) ( 0.35, 1.21)
 (-0.93,-0.44) (-0.33,-0.35) ( 0.41,-0.03) ( 0.57, 0.84)
 ( 0.54, 0.25) (-0.62,-0.05) (-0.52,-0.13) ( 0.11,-0.08)    :End of matrix C
```

## 9.3. Program Results

```
F08QVF Example Program Results

Solution matrix X
                      1                     2                     3                     4
1  (-0.0611, 0.0249) (-0.0031, 0.0798) (-0.0062, 0.0165) ( 0.0054,-0.0063)
2  ( 0.0215,-0.0003) (-0.0155, 0.0570) (-0.0665, 0.0718) ( 0.0290,-0.2636)
3  (-0.0949,-0.0785) (-0.0415,-0.0298) ( 0.0357, 0.0244) ( 0.0284, 0.1108)
4  ( 0.0281, 0.1052) (-0.0970,-0.1214) (-0.0271,-0.0940) ( 0.0402, 0.0048)

SCALE =   1.00E+00
```

# F08QXF (CTREVC/ZTREVC) – 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

F08QXF (CTREVC/ZTREVC) computes selected left and/or right eigenvectors of a complex upper triangular matrix.

## 2. Specification

```
 SUBROUTINE F08QXF (JOB, HOWMNY, SELECT, N, T, LDT, VL, LDVL, VR, LDVR,
1                   MM, M, WORK, RWORK, INFO)
 ENTRY       ctrevc  (JOB, HOWMNY, SELECT, N, T, LDT, VL, LDVL, VR, LDVR,
1                   MM, M, WORK, RWORK, INFO)
 INTEGER          N, LDT, LDVL, LDVR, MM, M, INFO
 real             RWORK(*)
 complex          T(LDT,*), VL(LDVL,*), VR(LDVR,*), WORK(*)
 LOGICAL          SELECT(*)
 CHARACTER*1      JOB, HOWMNY
```

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

## 3. Description

This routine computes left and/or right eigenvectors of a complex upper triangular matrix $T$. Such a matrix arises from the Schur factorization of a complex general matrix, as computed by F08PSF (CHSEQR/ZHSEQR), for example.

The right eigenvector $x$, and the left eigenvector $y$, corresponding to an eigenvalue $\lambda$, are defined by:

$$Tx = \lambda x \text{ and } y^H T = \lambda y^H \text{ (or } T^H y = \bar{\lambda} y).$$

The routine can compute the eigenvectors corresponding to selected eigenvalues, or it can compute all the eigenvectors. In the latter case the eigenvectors may optionally be pre-multiplied by an input matrix $Q$. Normally $Q$ is a unitary matrix from the Schur factorization of a matrix $A$ as $A = QTQ^H$; if $x$ is a (left or right) eigenvector of $T$, then $Qx$ is an eigenvector of $A$.

The eigenvectors are computed by forward or backward substitution. They are scaled so that $\max(|Re(x_i)|+|Im(x_i)|) = 1$.

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §7.6.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

## 5. Parameters

1: JOB – CHARACTER*1. *Input*

*On entry*: indicates whether left and/or right eigenvectors are to be computed as follows:

if JOB = 'R', then only right eigenvectors are computed;

if JOB = 'L', then only left eigenvectors are computed;

if JOB = 'B', then both left and right eigenvectors are computed.

*Constraint*: JOB = 'R', 'L' or 'B'.

2: HOWMNY – CHARACTER*1. *Input*

*On entry*: indicates how many eigenvectors are to be computed as follows:

if HOWMNY = 'A', then all eigenvectors (as specified by JOB) are computed;

if HOWMNY = 'O', then all eigenvectors (as specified by JOB) are computed and then pre-multiplied by the matrix $Q$ (which is overwritten);

if HOWMNY = 'S', then selected eigenvectors (as specified by JOB and SELECT) are computed.

*Constraint*: HOWMNY = 'A', 'O' or 'S'.

3: SELECT(*) – LOGICAL array. *Input*

**Note**: the dimension of the array SELECT must be at least max(1,N) if HOWMNY = 'S' and at least 1 otherwise.

*On entry*: SELECT specifies which eigenvectors are to be computed if HOWMNY = 'S'. To obtain the eigenvector corresponding to the eigenvalue $\lambda_j$, SELECT($j$) must be set .TRUE..

SELECT is not referenced if HOWMNY = 'A' or 'O'.

4: N – INTEGER. *Input*

*On entry*: $n$, the order of the matrix $T$.

*Constraint*: N ≥ 0.

5: T(LDT,*) – ***complex*** array. *Input*

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

*On entry*: the $n$ by $n$ upper triangular matrix $T$, as returned by F08PSF (CHSEQR/ZHSEQR).

6: LDT – INTEGER. *Input*

*On entry*: the first dimension of the array T as declared in the (sub)program from which F08QXF (CTREVC/ZTREVC) is called.

*Constraint*: LDT ≥ max(1,N).

7: VL(LDVL,*) – ***complex*** array. *Input/Output*

**Note**: the second dimension of the array VL must be at least max(1,MM) if JOB = 'L' or 'B' and at least 1 if JOB = 'R'.

*On entry*: if HOWMNY = 'O' and JOB = 'L' or 'B', VL must contain an $n$ by $n$ matrix $Q$ (usually the matrix of Schur vectors returned by F08PSF (CHSEQR/ZHSEQR)). If HOWMNY = 'A' or 'S', VL need not be set.

*On exit*: if JOB = 'L' or 'B', VL contains the computed left eigenvectors (as specified by HOWMNY and SELECT). The eigenvectors are stored consecutively in the columns of the array, in the same order as their eigenvalues.

VL is not referenced if JOB = 'R'.

8: LDVL – INTEGER. *Input*

*On entry*: the first dimension of the array VL as declared in the (sub)program from which F08QXF (CTREVC/ZTREVC) is called.

*Constraints*: LDVL ≥ max(1,N) if JOB = 'L' or 'B',
LDVL ≥ 1 if JOB = 'R'.

9: VR(LDVR,*) – ***complex*** array. *Input/Output*

**Note**: the second dimension of the array VR must be at least max(1,MM) if JOB = 'R' or 'B' and at least 1 if JOB = 'L'.

*On entry*: if HOWMNY = 'O' and JOB = 'R' or 'B', VR must contain an $n$ by $n$ matrix $Q$ (usually the matrix of Schur vectors returned by F08PSF (CHSEQR/ZHSEQR)). If HOWMNY = 'A' or 'S', VR need not be set.

*On exit*: if JOB = 'R' or 'B', VR contains the computed right eigenvectors (as specified by HOWMNY and SELECT). The eigenvectors are stored consecutively in the columns of the array, in the same order as their eigenvalues.

VR is not referenced if JOB = 'L'.

10: LDVR – INTEGER. *Input*

*On entry*: the first dimension of the array VR as declared in the (sub)program from which F08QXF (CTREVC/ZTREVC) is called.

*Constraints*: LDVR ≥ max(1,N) if JOB = 'R' or 'B',
LDVR ≥ 1 if JOB = 'L'.

11: MM – INTEGER. *Input*

*On entry*: the number of columns in the arrays VL and/or VR. The precise number of columns required, $m$, is $n$ if HOWMNY = 'A' or 'O'; if HOWMNY = 'S', $m$ is the number of selected eigenvectors (see SELECT), in which case $0 \leq m \leq n$.

*Constraint*: MM ≥ $m$.

12: M – INTEGER. *Output*

*On exit*: $m$, the number of selected eigenvectors. If HOWMNY = 'A' or 'O', M is set to $n$.

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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

## 7. Accuracy

If $x_i$ is an exact right eigenvector, and $\tilde{x}_i$ is the corresponding computed eigenvector, then the angle $\theta(\tilde{x}_i,x_i)$ between them is bounded as follows:

$$\theta(\tilde{x}_i,x_i) \leq \frac{c(n)\varepsilon\|T\|_2}{sep_i}$$

where $sep_i$ is the reciprocal condition number of $x_i$.

The condition number $sep_i$ may be computed by calling F08QYF (CTRSNA/ZTRSNA).

## 8. Further Comments

The real analogue of this routine is F08QKF (STREVC/DTREVC).

## 9. Example

See the example for F08NVF (CGEBAL/ZGEBAL).

# F08QYF (CTRSNA/ZTRSNA) – 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

F08QYF (CTRSNA/ZTRSNA) estimates condition numbers for specified eigenvalues and/or right eigenvectors of a complex upper triangular matrix.

## 2. Specification

```
 SUBROUTINE F08QYF (JOB, HOWMNY, SELECT, N, T, LDT, VL, LDVL, VR, LDVR,
1                   S, SEP, MM, M, WORK, LDWORK, RWORK, INFO)
 ENTRY       ctrsna  (JOB, HOWMNY, SELECT, N, T, LDT, VL, LDVL, VR, LDVR,
1                   S, SEP, MM, M, WORK, LDWORK, RWORK, INFO)
 INTEGER          N, LDT, LDVL, LDVR, MM, M, LDWORK, INFO
 real             S(*), SEP(*), RWORK(*)
 complex          T(LDT,*), VL(LDVL,*), VR(LDVR,*), WORK(LDWORK,*)
 LOGICAL          SELECT(*)
 CHARACTER*1      JOB, HOWMNY
```

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

## 3. Description

This routine estimates condition numbers for specified eigenvalues and/or right eigenvectors of a complex upper triangular matrix $T$. These are the same as the condition numbers of the eigenvalues and right eigenvectors of an original matrix $A = ZTZ^H$ (with unitary $Z$), from which $T$ may have been derived.

F08QYF computes the reciprocal of the condition number of an eigenvalue $\lambda_i$ as

$$s_i = \frac{|v^H u|}{\|u\|_E \|v\|_E},$$

where $u$ and $v$ are the right and left eigenvectors of $T$, respectively, corresponding to $\lambda_i$. This reciprocal condition number always lies between zero (i.e. ill-conditioned) and one (i.e. well-conditioned).

An approximate error estimate for a computed eigenvalue $\lambda_i$ is then given by

$$\frac{\varepsilon \|T\|}{s_i},$$

where $\varepsilon$ is the ***machine precision***.

To estimate the reciprocal of the condition number of the right eigenvector corresponding to $\lambda_i$, the routine first calls F08QTF (CTREXC/ZTREXC) to reorder the eigenvalues so that $\lambda_i$ is in the leading position:

$$T = Q \begin{pmatrix} \lambda_i & c^H \\ 0 & T_{22} \end{pmatrix} Q^H.$$

The reciprocal condition number of the eigenvector is then estimated as $sep_i$, the smallest singular value of the matrix $(T_{22} - \lambda_i I)$. This number ranges from zero (i.e. ill-conditioned) to very large (i.e. well-conditioned).

An approximate error estimate for a computed right eigenvector $u$ corresponding to $\lambda_i$ is then given by

$$\frac{\varepsilon \|T\|}{sep_i}.$$

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §7.2.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

## 5. Parameters

1: JOB – CHARACTER*1. *Input*

*On entry*: indicates whether condition numbers are required for eigenvalues and/or eigenvectors, as follows:

if JOB = 'E', then condition numbers for eigenvalues only are computed;

if JOB = 'V', then condition numbers for eigenvectors only are computed;

if JOB = 'B', then condition numbers for both eigenvalues and eigenvectors are computed.

*Constraint*: JOB = 'E', 'V' or 'B'.

2: HOWMNY – CHARACTER*1. *Input*

*On entry*: indicates how many condition numbers are to be computed, as follows:

if HOWMNY = 'A', then condition numbers for all eigenpairs are computed;

if HOWMNY = 'S', then condition numbers for selected eigenpairs (as specified by SELECT) are computed.

*Constraint*: HOWMNY = 'A' or 'S'.

3: SELECT(*) – LOGICAL array. *Input*

**Note**: the dimension of the array SELECT must be at least max(1,N) if HOWMNY = 'S' and at least 1 otherwise.

*On entry*: SELECT specifies the eigenpairs for which condition numbers are to be computed if HOWMNY = 'S'. To select condition numbers for the eigenpair corresponding to the eigenvalue $\lambda_j$, SELECT($j$) must be set .TRUE..

SELECT is not referenced if HOWMNY = 'A'.

4: N – INTEGER. *Input*

*On entry*: $n$, the order of the matrix $T$.

*Constraint*: N $\geq$ 0.

5: T(LDT,*) – ***complex*** array. *Input*

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

*On entry*: the $n$ by $n$ upper triangular matrix $T$, as returned by F08PSF (CHSEQR/ZHSEQR).

6: LDT – INTEGER. *Input*

*On entry*: the first dimension of the array T as declared in the (sub)program from which F08QYF (CTRSNA/ZTRSNA) is called.

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

7: VL(LDVL,*) – ***complex*** array. *Input*

**Note**: the second dimension of the array VL must be at least max(1,MM) if JOB = 'E' or 'B' and at least 1 if JOB = 'V'.

*On entry*: if JOB = 'E' or 'B', VL must contain the left eigenvectors of $T$ (or of any matrix $QTQ^H$ with $Q$ unitary) corresponding to the eigenpairs specified by HOWMNY and SELECT. The eigenvectors **must** be stored in consecutive columns of VL, as returned by F08QXF (CTREVC/ZTREVC) or F08PXF (CHSEIN/ZHSEIN).

VL is not referenced if JOB = 'V'.

8: LDVL – INTEGER. *Input*

*On entry*: the first dimension of the array VL as declared in the (sub)program from which F08QYF (CTRSNA/ZTRSNA) is called.

*Constraints*: LDVL ≥ max(1,N) if JOB = 'E' or 'B',
LDVL ≥ 1 if JOB = 'V'.

9: VR(LDVR,*) – ***complex*** array. *Input*

**Note**: the second dimension of the array VR must be at least max(1,MM) if JOB = 'E' or 'B' and at least 1 if JOB = 'V'.

*On entry*: if JOB = 'E' or 'B', VR must contain the right eigenvectors of $T$ (or of any matrix $QTQ^H$ with $Q$ unitary) corresponding to the eigenpairs specified by HOWMNY and SELECT. The eigenvectors **must** be stored in consecutive columns of VR, as returned by F08QXF (CTREVC/ZTREVC) or F08PXF (CHSEIN/ZHSEIN).

VR is not referenced if JOB = 'V'.

10: LDVR – INTEGER. *Input*

*On entry*: the first dimension of the array VR as declared in the (sub)program from which F08QYF (CTRSNA/ZTRSNA) is called.

*Constraints*: LDVR ≥ max(1,N) if JOB = 'E' or 'B',
LDVR ≥ 1 if JOB = 'V'.

11: S(*) – ***real*** array. *Output*

**Note**: the dimension of the array S must be at least max(1,MM) if JOB = 'E' or 'B' and at least 1 if JOB = 'V'.

*On exit*: the reciprocal condition numbers of the selected eigenvalues if JOB = 'E' or 'B', stored in consecutive elements of the array. Thus S($j$), SEP($j$) and the $j$th columns of VL and VR all correspond to the same eigenpair (but not in general the $j$th eigenpair unless all eigenpairs have been selected).

S is not referenced if JOB = 'V'.

12: SEP(*) – ***real*** array. *Output*

**Note**: the dimension of the array SEP must be at least max(1,MM) if JOB = 'V' or 'B' and at least 1 if JOB = 'E'.

*On exit*: the estimated reciprocal condition numbers of the selected right eigenvectors if JOB = 'V' or 'B', stored in consecutive elements of the array.

SEP is not referenced if JOB = 'E'.

13: MM – INTEGER. *Input*

*On entry*: the number of elements in the arrays S and SEP, and the number of columns in the arrays VL and VR (if used). The precise number required, $m$, is $n$ if HOWMNY = 'A'; if HOWMNY = 'S', $m$ is the number of selected eigenpairs (see SELECT), in which case $0 \leq m \leq n$.

*Constraint*: MM ≥ $m$.

14: M – INTEGER. *Output*

*On exit*: $m$, the number of selected eigenpairs. If HOWMNY = 'A', M is set to $n$.

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

**Note**: the second dimension of the array WORK must be at least max(1,N+1) if JOB = 'V' or 'B' and at least 1 if JOB = 'E'.

WORK is not referenced if JOB = 'E'.

16: LDWORK – INTEGER. *Input*

*On entry*: the first dimension of the array WORK as declared in the (sub)program from which F08QYF (CTRSNA/ZTRSNA) is called.

*Constraints*: LDWORK ≥ max(1,N) if JOB = 'V' or 'B',
LDWORK ≥ 1 if JOB = 'E'.

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

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

18: 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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

## 7. Accuracy

The computed values $sep_i$ may overestimate the true value, but seldom by a factor of more than 3.

## 8. Further Comments

The real analogue of this routine is F08QLF (STRSNA/DTRSNA).

## 9. Example

To compute approximate error estimates for all the eigenvalues and right eigenvectors of the matrix $T$, where

$$T = \begin{pmatrix} -6.0004 - 6.9999i & 0.3637 - 0.3656i & -0.1880 + 0.4787i & 0.8785 - 0.2539i \\ 0.0000 + 0.0000i & -5.0000 + 2.0060i & -0.0307 - 0.7217i & -0.2290 + 0.1313i \\ 0.0000 + 0.0000i & 0.0000 + 0.0000i & 7.9982 - 0.9964i & 0.9357 + 0.5359i \\ 0.0000 + 0.0000i & 0.0000 + 0.0000i & 0.0000 + 0.0000i & 3.0023 - 3.9998i \end{pmatrix}.$$

### 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.

```
*     F08QYF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDT, LDWORK, LDVL, LDVR
      PARAMETER        (NMAX=8,LDT=NMAX,LDWORK=NMAX,LDVL=NMAX,LDVR=NMAX)
*     .. Local Scalars ..
      real             EPS, TNORM
      INTEGER          I, INFO, J, M, N
```

```
*     .. Local Arrays ..
      complex           T(LDT,NMAX), VL(LDVL,NMAX), VR(LDVR,NMAX),
     +                  WORK(LDWORK,NMAX+6)
      real              RWORK(NMAX), S(NMAX), SEP(NMAX)
      LOGICAL           SELECT(1)
*     .. External Functions ..
      real              F06UAF, X02AJF
      EXTERNAL          F06UAF, X02AJF
*     .. External Subroutines ..
      EXTERNAL          ctrevc, ctrsna
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08QYF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read T from data file
*
         READ (NIN,*) ((T(I,J),J=1,N),I=1,N)
*
*        Calculate the left and right eigenvectors of T
*
         CALL ctrevc('Both','All',SELECT,N,T,LDT,VL,LDVL,VR,LDVR,N,M,
     +               WORK,RWORK,INFO)
*
*        Estimate condition numbers for all the eigenvalues and right
*        eigenvectors of T
*
         CALL ctrsna('Both','All',SELECT,N,T,LDT,VL,LDVL,VR,LDVR,S,SEP,
     +               N,M,WORK,LDWORK,RWORK,INFO)
*
*        Print condition numbers of eigenvalues and right eigenvectors
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'S'
         WRITE (NOUT,99999) (S(I),I=1,M)
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'SEP'
         WRITE (NOUT,99999) (SEP(I),I=1,M)
*
*        Calculate approximate error estimates (using the 1-norm)
*
         EPS = X02AJF()
         TNORM = F06UAF('1-norm',N,N,T,LDT,RWORK)
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Approximate error estimates for eigenvalues ',
     +     'of T (machine-dependent)'
         WRITE (NOUT,99999) (EPS*TNORM/S(I),I=1,M)
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Approximate error estimates for right ',
     +     'eigenvectors of T (machine-dependent)'
         WRITE (NOUT,99999) (EPS*TNORM/SEP(I),I=1,M)
      END IF
      STOP
*
99999 FORMAT ((3X,1P,7e11.1))
      END
```

## 9.2. Program Data

```
F08QYF Example Program Data
  4                                                              :Value of N
 (-6.0004,-6.9999) ( 0.3637,-0.3656) (-0.1880, 0.4787) ( 0.8785,-0.2539)
 ( 0.0000, 0.0000) (-5.0000, 2.0060) (-0.0307,-0.7217) (-0.2290, 0.1313)
 ( 0.0000, 0.0000) ( 0.0000, 0.0000) ( 7.9982,-0.9964) ( 0.9357, 0.5359)
 ( 0.0000, 0.0000) ( 0.0000, 0.0000) ( 0.0000, 0.0000) ( 3.0023,-3.9998)
                                                        :End of matrix T
```

## 9.3. Program Results

```
F08QYF Example Program Results

S
      9.9E-01    1.0E+00    9.8E-01    9.8E-01

SEP
      8.4E+00    8.0E+00    5.8E+00    5.8E+00

Approximate error estimates for eigenvalues of T (machine-dependent)
      1.0E-15    1.0E-15    1.1E-15    1.1E-15

Approximate error estimates for right eigenvectors of T (machine-dependent)
      1.2E-16    1.3E-16    1.8E-16    1.8E-16
```

# F08SEF (SSYGST/DSYGST) – 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

F08SEF (SSYGST/DSYGST) reduces a real symmetric-definite generalized eigenproblem $Az = \lambda Bz$, $ABz = \lambda z$ or $BAz = \lambda z$ to the standard form $Cy = \lambda y$, where $A$ is a real symmetric matrix and $B$ has been factorized by F07FDF (SPOTRF/DPOTRF).

## 2. Specification

```
SUBROUTINE F08SEF (ITYPE, UPLO, N, A, LDA, B, LDB, INFO)
ENTRY      ssygst  (ITYPE, UPLO, N, A, LDA, B, LDB, INFO)
INTEGER          ITYPE, N, 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 reduce the real symmetric-definite generalized eigenproblem $Az = \lambda Bz$, $ABz = \lambda z$ or $BAz = \lambda z$ to the standard form $Cy = \lambda y$, this routine must be preceded by a call to F07FDF (SPOTRF/DPOTRF) which computes the Cholesky factorization of $B$; $B$ must be positive-definite.

The different problem types are specified by the parameter ITYPE, as indicated in the table below. The table shows how $C$ is computed by the routine, and also how the eigenvectors $z$ of the original problem can be recovered from the eigenvectors of the standard form.

| ITYPE | Problem | UPLO | $B$ | $C$ | $z$ |
|---|---|---|---|---|---|
| 1 | $Az = \lambda Bz$ | 'U' | $U^TU$ | $U^{-T}AU^{-1}$ | $U^{-1}y$ |
| | | 'L' | $LL^T$ | $L^{-1}AL^{-T}$ | $L^{-T}y$ |
| 2 | $ABz = \lambda z$ | 'U' | $U^TU$ | $UAU^T$ | $U^{-1}y$ |
| | | 'L' | $LL^T$ | $L^TAL$ | $L^{-T}y$ |
| 3 | $BAz = \lambda z$ | 'U' | $U^TU$ | $UAU^T$ | $U^Ty$ |
| | | 'L' | $LL^T$ | $L^TAL$ | $Ly$ |

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §8.7.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

## 5. Parameters

1: ITYPE – INTEGER. *Input*

*On entry*: indicates how the standard form is computed as follows:

if ITYPE = 1, then $C = U^{-T}AU^{-1}$ if UPLO = 'U',
$C = L^{-1}AL^{-T}$ if UPLO = 'L';

if ITYPE = 2 or 3, then $C = UAU^{T}$ if UPLO = 'U',
$C = L^{T}AL$ if UPLO = 'L'.

*Constraint*: $1 \leq$ ITYPE $\leq 3$.

2: UPLO – CHARACTER*1. *Input*

*On entry*: indicates whether the upper or lower triangular part of $A$ is stored and how $B$ has been factorized, as follows:

if UPLO = 'U', then the upper triangular part of $A$ is stored and $B = U^{T}U$;

if UPLO = 'L', then the lower triangular part of $A$ is stored and $B = LL^{T}$.

*Constraint*: UPLO = 'U' or 'L'.

3: N – INTEGER. *Input*

*On entry*: $n$, the order of the matrices $A$ and $B$.

*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 $n$ 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 the corresponding upper or lower triangle of $C$ as specified by ITYPE and UPLO.

5: LDA – INTEGER. *Input*

*On entry*: the first dimension of the array A as declared in the (sub)program from which F08SEF (SSYGST/DSYGST) is called.

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

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

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

*On entry*: the Cholesky factor of $B$ as specified by UPLO and returned by F07FDF (SPOTRF/DPOTRF).

7: LDB – INTEGER. *Input*

*On entry*: the first dimension of the array B as declared in the (sub)program from which F08SEF (SSYGST/DSYGST) 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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

## 7. Accuracy

Forming the reduced matrix $C$ is a stable procedure. However it involves implicit multiplication by $B^{-1}$ if (ITYPE = 1) or $B$ (if ITYPE = 2 or 3). When the routine is used as a step in the computation of eigenvalues and eigenvectors of the original problem, there may be a significant loss of accuracy if $B$ is ill-conditioned with respect to inversion. See the document for F02FDF for further details.

## 8. Further Comments

The total number of floating-point operations is approximately $n^3$.

The complex analogue of this routine is F08SSF (CHEGST/ZHEGST).

## 9. Example

To compute all the eigenvalues of $Az = \lambda Bz$, where

$$A = \begin{pmatrix} 0.24 & 0.39 & 0.42 & -0.16 \\ 0.39 & -0.11 & 0.79 & 0.63 \\ 0.42 & 0.79 & -0.25 & 0.48 \\ -0.16 & 0.63 & 0.48 & -0.03 \end{pmatrix} \text{ and } B = \begin{pmatrix} 4.16 & -3.12 & 0.56 & -0.10 \\ -3.12 & 5.03 & -0.83 & 1.18 \\ 0.56 & -0.83 & 0.76 & 0.34 \\ -0.10 & 1.18 & 0.34 & 1.18 \end{pmatrix}.$$

Here $B$ is symmetric positive-definite and must first be factorized by F07FDF (SPOTRF/DPOTRF). The program calls F08SEF (SSYGST/DSYGST) to reduce the problem to the standard form $Cy = \lambda y$; then F08FEF (SSYTRD/DSYTRD) to reduce $C$ to tridiagonal form, and F08JFF (SSTERF/DSTERF) to compute the eigenvalues.

### 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.**

```
*     F08SEF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, LDB, LWORK
      PARAMETER        (NMAX=8,LDA=NMAX,LDB=NMAX,LWORK=64*NMAX)
*     .. Local Scalars ..
      INTEGER          I, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             A(LDA,NMAX), B(LDB,NMAX), D(NMAX), E(NMAX-1),
     +                 TAU(NMAX), WORK(LWORK)
*     .. External Subroutines ..
      EXTERNAL         spotrf, ssterf, ssygst, ssytrd
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08SEF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) 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)
            READ (NIN,*) ((B(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)
            READ (NIN,*) ((B(I,J),J=1,I),I=1,N)
         END IF
*
*        Compute the Cholesky factorization of B
*
```

```
         CALL spotrf(UPLO,N,B,LDB,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.GT.0) THEN
            WRITE (NOUT,*) 'B is not positive-definite.'
         ELSE
*
*           Reduce the problem to standard form C*y = lambda*y, storing
*           the result in A
*
            CALL ssygst(1,UPLO,N,A,LDA,B,LDB,INFO)
*
*           Reduce C to tridiagonal form T = (Q**T)*C*Q
*
            CALL ssytrd(UPLO,N,A,LDA,D,E,TAU,WORK,LWORK,INFO)
*
*           Calculate the eigenvalues of T (same as C)
*
            CALL ssterf(N,D,E,INFO)
*
            IF (INFO.GT.0) THEN
               WRITE (NOUT,*) 'Failure to converge.'
            ELSE
*
*              Print eigenvalues
*
               WRITE (NOUT,*) 'Eigenvalues'
               WRITE (NOUT,99999) (D(I),I=1,N)
            END IF
         END IF
      END IF
      STOP
*
99999 FORMAT (3X,(9F8.4))
      END
```

## 9.2. Program Data

```
F08SEF Example Program Data
  4                              :Value of N
  'L'                            :Value of UPLO
  0.24
  0.39  -0.11
  0.42   0.79  -0.25
 -0.16   0.63   0.48  -0.03      :End of matrix A
  4.16
 -3.12   5.03
  0.56  -0.83   0.76
 -0.10   1.09   0.34   1.18      :End of matrix B
```

## 9.3. Program Results

```
 F08SEF Example Program Results

 Eigenvalues
    -2.2254 -0.4548  0.1001  1.1270
```

# F08SSF (CHEGST/ZHEGST) – 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

F08SSF (CHEGST/ZHEGST) reduces a complex Hermitian-definite generalized eigenproblem $Az = \lambda Bz$, $ABz = \lambda z$ or $BAz = \lambda z$ to the standard form $Cy = \lambda y$, where $A$ is a complex Hermitian matrix and $B$ has been factorized by F07FRF (CPOTRF/ZPOTRF).

## 2. Specification

```
SUBROUTINE F08SSF (ITYPE, UPLO, N, A, LDA, B, LDB, INFO)
ENTRY      chegst  (ITYPE, UPLO, N, A, LDA, B, LDB, INFO)
INTEGER          ITYPE, N, 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 reduce the complex Hermitian-definite generalized eigenproblem $Az = \lambda Bz$, $ABz = \lambda z$ or $BAz = \lambda z$ to the standard form $Cy = \lambda y$, this routine must be preceded by a call to F07FRF (CPOTRF/ZPOTRF) which computes the Cholesky factorization of $B$; $B$ must be positive-definite.

The different problem types are specified by the parameter ITYPE, as indicated in the table below. The table shows how $C$ is computed by the routine, and also how the eigenvectors $z$ of the original problem can be recovered from the eigenvectors of the standard form.

| ITYPE | Problem | UPLO | $B$ | $C$ | $z$ |
|---|---|---|---|---|---|
| 1 | $Az = \lambda Bz$ | 'U' | $U^H U$ | $U^{-H} A U^{-1}$ | $U^{-1} y$ |
| | | 'L' | $LL^H$ | $L^{-1} A L^{-H}$ | $L^{-H} y$ |
| 2 | $ABz = \lambda z$ | 'U' | $U^H U$ | $UAU^H$ | $U^{-1} y$ |
| | | 'L' | $LL^H$ | $L^H AL$ | $L^{-H} y$ |
| 3 | $BAz = \lambda z$ | 'U' | $U^H U$ | $UAU^H$ | $U^H y$ |
| | | 'L' | $LL^H$ | $L^H AL$ | $Ly$ |

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §8.7.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

## 5. Parameters

1: ITYPE – INTEGER. *Input*

*On entry*: indicates how the standard form is computed as follows:

if ITYPE = 1, then $C = U^{-H}AU^{-1}$ if UPLO = 'U',
$C = L^{-1}AL^{-H}$ if UPLO = 'L';

if ITYPE = 2 or 3, then $C = UAU^{H}$ if UPLO = 'U',
$C = L^{H}AL$ if UPLO = 'L'.

*Constraint*: 1 ≤ ITYPE ≤ 3.

2: UPLO – CHARACTER*1. *Input*

*On entry*: indicates whether the upper or lower triangular part of $A$ is stored and how $B$ has been factorized, as follows:

if UPLO = 'U', then the upper triangular part of $A$ is stored and $B = U^{H}U$;

if UPLO = 'L', then the lower triangular part of $A$ is stored and $B = LL^{H}$.

*Constraint*: UPLO = 'U' or 'L'.

3: N – INTEGER. *Input*

*On entry*: $n$, the order of the matrices $A$ and $B$.

*Constraint*: N ≥ 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 $n$ 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 the corresponding upper or lower triangle of $C$ as specified by ITYPE and UPLO.

5: LDA – INTEGER. *Input*

*On entry*: the first dimension of the array A as declared in the (sub)program from which F08SSF (CHEGST/ZHEGST) is called.

*Constraint*: LDA ≥ max(1,N).

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

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

*On entry*: the Cholesky factor of $B$ as specified by UPLO and returned by F07FRF (CPOTRF/ZPOTRF).

7: LDB – INTEGER. *Input*

*On entry*: the first dimension of the array B as declared in the (sub)program from which F08SSF (CHEGST/ZHEGST) 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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

## 7. Accuracy

Forming the reduced matrix $C$ is a stable procedure. However it involves implicit multiplication by $B^{-1}$ if (ITYPE = 1) or $B$ (if ITYPE = 2 or 3). When the routine is used as a step in the computation of eigenvalues and eigenvectors of the original problem, there may be a significant loss of accuracy if $B$ is ill-conditioned with respect to inversion. See the document for F02HDF for further details.

## 8. Further Comments

The total number of real floating-point operations is approximately $4n^3$.

The real analogue of this routine is F08SEF (SSYGST/DSYGST).

## 9. Example

To compute all the eigenvalues of $Az = \lambda Bz$, where

$$A = \begin{pmatrix} -7.36 + 0.00i & 0.77 - 0.43i & -0.64 - 0.92i & 3.01 - 6.97i \\ 0.77 + 0.43i & 3.49 + 0.00i & 2.19 + 4.45i & 1.90 + 3.73i \\ -0.64 + 0.92i & 2.19 - 4.45i & 0.12 + 0.00i & 2.88 - 3.17i \\ 3.01 + 6.97i & 1.90 - 3.73i & 2.88 + 3.17i & -2.54 + 0.00i \end{pmatrix}$$

and

$$B = \begin{pmatrix} 3.23 + 0.00i & 1.51 - 1.92i & 1.90 + 0.84i & 0.42 + 2.50i \\ 1.51 + 1.92i & 3.58 + 0.00i & -0.23 + 1.11i & -1.18 + 1.37i \\ 1.90 - 0.84i & -0.23 - 1.11i & 4.09 + 0.00i & 2.33 - 0.14i \\ 0.42 - 2.50i & -1.18 - 1.37i & 2.33 + 0.14i & 4.29 + 0.00i \end{pmatrix}.$$

Here $B$ is Hermitian positive-definite and must first be factorized by F07FRF (DPOTRF/ZPOTRF). The program calls F08SSF (CHEGST/ZHEGST) to reduce the problem to the standard form $Cy = \lambda y$; then F08FSF (CHETRD/ZHETRD) to reduce $C$ to tridiagonal form, and F08JFF (SSTERF/DSTERF) to compute the eigenvalues.

### 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.

```
*     F08SSF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA, LDB, LWORK
      PARAMETER        (NMAX=8,LDA=NMAX,LDB=NMAX,LWORK=64*NMAX)
*     .. Local Scalars ..
      INTEGER          I, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          A(LDA,NMAX), B(LDB,NMAX), TAU(NMAX), WORK(LWORK)
      real             D(NMAX), E(NMAX-1)
*     .. External Subroutines ..
      EXTERNAL         ssterf, chegst, chetrd, cpotrf
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08SSF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) 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)
            READ (NIN,*) ((B(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)
            READ (NIN,*) ((B(I,J),J=1,I),I=1,N)
         END IF
*
*        Compute the Cholesky factorization of B
*
         CALL cpotrf(UPLO,N,B,LDB,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.GT.0) THEN
            WRITE (NOUT,*) 'B is not positive-definite.'
         ELSE
*
*           Reduce the problem to standard form C*y = lambda*y, storing
*           the result in A
*
            CALL chegst(1,UPLO,N,A,LDA,B,LDB,INFO)
*
*           Reduce C to tridiagonal form T = (Q**H)*C*Q
*
            CALL chetrd(UPLO,N,A,LDA,D,E,TAU,WORK,LWORK,INFO)
*
*           Calculate the eigenvalues of T (same as C)
*
            CALL ssterf(N,D,E,INFO)
*
            IF (INFO.GT.0) THEN
               WRITE (NOUT,*) 'Failure to converge.'
            ELSE
*
*              Print eigenvalues
*
               WRITE (NOUT,*) 'Eigenvalues'
               WRITE (NOUT,99999) (D(I),I=1,N)
            END IF
         END IF
      END IF
      STOP
*
99999 FORMAT (3X,(9F8.4))
      END
```

## 9.2. Program Data

```
F08SSF Example Program Data
  4                                                                  :Value of N
  'L'                                                                :Value of UPLO
 (-7.36, 0.00)
 ( 0.77, 0.43) ( 3.49, 0.00)
 (-0.64, 0.92) ( 2.19,-4.45) ( 0.12, 0.00)
 ( 3.01, 6.97) ( 1.90,-3.73) ( 2.88, 3.17) (-2.54, 0.00)           :End of matrix A
 ( 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 B
```

## 9.3. Program Results

```
 F08SSF Example Program Results

 Eigenvalues
    -5.9990 -2.9936  0.5047  3.9990
```

# F08TEF (SSPGST/DSPGST) – 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

F08TEF (SSPGST/DSPGST) reduces a real symmetric-definite generalized eigenproblem $Az = \lambda Bz$, $ABz = \lambda z$ or $BAz = \lambda z$ to the standard form $Cy = \lambda y$, where $A$ is a real symmetric matrix and $B$ has been factorized by F07GDF (SPPTRF/DPPTRF), using packed storage.

## 2. Specification

```
SUBROUTINE F08TEF (ITYPE, UPLO, N, AP, BP, INFO)
ENTRY      sspgst  (ITYPE, UPLO, N, AP, BP, INFO)
INTEGER          ITYPE, N, INFO
real             AP(*), BP(*)
CHARACTER*1      UPLO
```

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

## 3. Description

To reduce the real symmetric-definite generalized eigenproblem $Az = \lambda Bz$, $ABz = \lambda z$ or $BAz = \lambda z$ to the standard form $Cy = \lambda y$ using packed storage, this routine must be preceded by a call to F07GDF (SPPTRF/DPPTRF) which computes the Cholesky factorization of $B$; $B$ must be positive-definite.

The different problem types are specified by the parameter ITYPE, as indicated in the table below. The table shows how $C$ is computed by the routine, and also how the eigenvectors $z$ of the original problem can be recovered from the eigenvectors of the standard form.

| ITYPE | Problem | UPLO | $B$ | $C$ | $z$ |
|---|---|---|---|---|---|
| 1 | $Az = \lambda Bz$ | 'U' | $U^TU$ | $U^{-T}AU^{-1}$ | $U^{-1}y$ |
| | | 'L' | $LL^T$ | $L^{-1}AL^{-T}$ | $L^{-T}y$ |
| 2 | $ABz = \lambda z$ | 'U' | $U^TU$ | $UAU^T$ | $U^{-1}y$ |
| | | 'L' | $LL^T$ | $L^TAL$ | $L^{-T}y$ |
| 3 | $BAz = \lambda z$ | 'U' | $U^TU$ | $UAU^T$ | $U^Ty$ |
| | | 'L' | $LL^T$ | $L^TAL$ | $Ly$ |

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §8.7.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

## 5. Parameters

1: ITYPE – INTEGER. *Input*

*On entry*: indicates how the standard form is computed as follows:

if ITYPE = 1, then $C = U^{-T}AU^{-1}$ if UPLO = 'U',
$C = L^{-1}AL^{-T}$ if UPLO = 'L';

if ITYPE = 2 or 3, then $C = UAU^{T}$ if UPLO = 'U',
$C = L^{T}AL$ if UPLO = 'L'.

*Constraint*: $1 \leq$ ITYPE $\leq 3$.

2: UPLO – CHARACTER*1. *Input*

*On entry*: indicates whether the upper or lower triangular part of $A$ is stored and how $B$ has been factorized, as follows:

if UPLO = 'U', then the upper triangular part of $A$ is stored and $B = U^{T}U$;

if UPLO = 'L', then the lower triangular part of $A$ is stored and $B = LL^{T}$.

*Constraint*: UPLO = 'U' or 'L'.

3: N – INTEGER. *Input*

*On entry*: $n$, the order of the matrices $A$ and $B$.

*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 $n$ 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($i$+(2$n$–$j$)($j$–1)/2) for $i \geq j$.

*On exit*: the upper or lower triangle of $A$ is overwritten by the corresponding upper or lower triangle of $C$ as specified by ITYPE and UPLO, using the same packed storage format as described above.

5: BP(*) – ***real*** array. *Input*

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

*On entry*: the Cholesky factor of $B$ as specified by UPLO and returned by F07GDF (SPPTRF/DPPTRF).

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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

## 7. Accuracy

Forming the reduced matrix $C$ is a stable procedure. However it involves implicit multiplication by $B^{-1}$ if (ITYPE = 1) or $B$ (if ITYPE = 2 or 3). When the routine is used as a step in the computation of eigenvalues and eigenvectors of the original problem, there may be a significant loss of accuracy if $B$ is ill-conditioned with respect to inversion. See the document for F02FDF for further details.

## 8. Further Comments

The total number of floating-point operations is approximately $n^3$.

The complex analogue of this routine is F08TSF (CHPGST/ZHPGST).

## 9. Example

To compute all the eigenvalues of $Az = \lambda Bz$, where

$$A = \begin{pmatrix} 0.24 & 0.39 & 0.42 & -0.16 \\ 0.39 & -0.11 & 0.79 & 0.63 \\ 0.42 & 0.79 & -0.25 & 0.48 \\ -0.16 & 0.63 & 0.48 & -0.03 \end{pmatrix} \text{ and } B = \begin{pmatrix} 4.16 & -3.12 & 0.56 & -0.10 \\ -3.12 & 5.03 & -0.83 & 1.18 \\ 0.56 & -0.83 & 0.76 & 0.34 \\ -0.10 & 1.18 & 0.34 & 1.18 \end{pmatrix},$$

using packed storage. Here $B$ is symmetric positive-definite and must first be factorized by F07GDF (SPPTRF/DPPTRF). The program calls F08TEF (SSPGST/DSPGST) to reduce the problem to the standard form $Cy = \lambda y$; then F08GEF (SSPTRD/DSPTRD) to reduce $C$ to tridiagonal form, and F08JFF (SSTERF/DSTERF) to compute the eigenvalues.

### 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.**

```
*     F08TEF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=8)
*     .. Local Scalars ..
      INTEGER          I, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             AP(NMAX*(NMAX+1)/2), BP(NMAX*(NMAX+1)/2),
     +                 D(NMAX), E(NMAX-1), TAU(NMAX)
*     .. External Subroutines ..
      EXTERNAL         spptrf, sspgst, ssptrd, ssterf
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08TEF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) 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)
            READ (NIN,*) ((BP(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)
            READ (NIN,*) ((BP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
*
*        Compute the Cholesky factorization of B
*
         CALL spptrf(UPLO,N,BP,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.GT.0) THEN
            WRITE (NOUT,*) 'B is not positive-definite.'
         ELSE
*
*           Reduce the problem to standard form C*y = lambda*y, storing
*           the result in A
```

```
*
            CALL sspgst(1,UPLO,N,AP,BP,INFO)
*
*           Reduce C to tridiagonal form T = (Q**T)*C*Q
*
            CALL ssptrd(UPLO,N,AP,D,E,TAU,INFO)
*
*           Calculate the eigenvalues of T (same as C)
*
            CALL ssterf(N,D,E,INFO)
*
            IF (INFO.GT.0) THEN
               WRITE (NOUT,*) 'Failure to converge.'
            ELSE
*
*              Print eigenvalues
*
               WRITE (NOUT,*) 'Eigenvalues'
               WRITE (NOUT,99999) (D(I),I=1,N)
            END IF
         END IF
      END IF
      STOP
*
99999 FORMAT (3X,(9F8.4))
      END
```

## 9.2. Program Data

```
F08TEF Example Program Data
  4                              :Value of N
  'L'                            :Value of UPLO
  0.24
  0.39  -0.11
  0.42   0.79  -0.25
 -0.16   0.63   0.48  -0.03     :End of matrix A
  4.16
 -3.12   5.03
  0.56  -0.83   0.76
 -0.10   1.09   0.34   1.18     :End of matrix B
```

## 9.3. Program Results

```
 F08TEF Example Program Results

 Eigenvalues
    -2.2254 -0.4548  0.1001  1.1270
```

# F08TSF (CHPGST/ZHPGST) – 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

F08TSF (CHPGST/ZHPGST) reduces a complex Hermitian-definite generalized eigenproblem $Az = \lambda Bz$, $ABz = \lambda z$ or $BAz = \lambda z$ to the standard form $Cy = \lambda y$, where $A$ is a complex Hermitian matrix and $B$ has been factorized by F07GRF (CPPTRF/ZPPTRF), using packed storage.

## 2. Specification

```
SUBROUTINE F08TSF (ITYPE, UPLO, N, AP, BP, INFO)
ENTRY      chpgst  (ITYPE, UPLO, N, AP, BP, INFO)
INTEGER          ITYPE, N, INFO
complex          AP(*), BP(*)
CHARACTER*1      UPLO
```

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

## 3. Description

To reduce the complex Hermitian-definite generalized eigenproblem $Az = \lambda Bz$, $ABz = \lambda z$ or $BAz = \lambda z$ to the standard form $Cy = \lambda y$ using packed storage, this routine must be preceded by a call to F07GRF (CPPTRF/ZPPTRF) which computes the Cholesky factorization of $B$; $B$ must be positive-definite.

The different problem types are specified by the parameter ITYPE, as indicated in the table below. The table shows how $C$ is computed by the routine, and also how the eigenvectors $z$ of the original problem can be recovered from the eigenvectors of the standard form.

| ITYPE | Problem | UPLO | $B$ | $C$ | $z$ |
|---|---|---|---|---|---|
| 1 | $Az = \lambda Bz$ | 'U' | $U^HU$ | $U^{-H}AU^{-1}$ | $U^{-1}y$ |
| | | 'L' | $LL^H$ | $L^{-1}AL^{-H}$ | $L^{-H}y$ |
| 2 | $ABz = \lambda z$ | 'U' | $U^HU$ | $UAU^H$ | $U^{-1}y$ |
| | | 'L' | $LL^H$ | $L^HAL$ | $L^{-H}y$ |
| 3 | $BAz = \lambda z$ | 'U' | $U^HU$ | $UAU^H$ | $U^Hy$ |
| | | 'L' | $LL^H$ | $L^HAL$ | $Ly$ |

## 4. References

[1] GOLUB, G.H. and VAN LOAN, C.F.
Matrix Computations, §8.7.
Johns Hopkins University Press, Baltimore, Maryland, (2nd Edition) 1989.

## 5. Parameters

1: ITYPE – INTEGER. *Input*

*On entry*: indicates how the standard form is computed as follows:

if ITYPE = 1, then $C = U^{-H}AU^{-1}$ if UPLO = 'U',
$C = L^{-1}AL^{-H}$ if UPLO = 'L';

if ITYPE = 2 or 3, then $C = UAU^{H}$ if UPLO = 'U',
$C = L^{H}AL$ if UPLO = 'L'.

*Constraint*: $1 \leq$ ITYPE $\leq 3$.

2: UPLO – CHARACTER*1. *Input*

*On entry*: indicates whether the upper or lower triangular part of $A$ is stored and how $B$ has been factorized, as follows:

if UPLO = 'U', then the upper triangular part of $A$ is stored and $B = U^{H}U$;

if UPLO = 'L', then the lower triangular part of $A$ is stored and $B = LL^{H}$.

*Constraint*: UPLO = 'U' or 'L'.

3: N – INTEGER. *Input*

*On entry*: $n$, the order of the matrices $A$ and $B$.

*Constraint*: N $\geq$ 0.

4: 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($i$+(2$n$–$j$)($j$–1)/2) for $i \geq j$.

*On exit*: the upper or lower triangle of $A$ is overwritten by the corresponding upper or lower triangle of $C$ as specified by ITYPE and UPLO, using the same packed storage format as described above.

5: BP(*) – ***complex*** array. *Input*

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

*On entry*: the Cholesky factor of $B$ as specified by UPLO and returned by F07GRF (CPPTRF/ZPPTRF).

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 $i$th parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

## 7. Accuracy

Forming the reduced matrix $C$ is a stable procedure. However it involves implicit multiplication by $B^{-1}$ if (ITYPE = 1) or $B$ (if ITYPE = 2 or 3). When the routine is used as a step in the computation of eigenvalues and eigenvectors of the original problem, there may be a significant loss of accuracy if $B$ is ill-conditioned with respect to inversion. See the document for F02HDF for further details.

## 8. Further Comments

The total number of real floating-point operations is approximately $4n^3$.

The real analogue of this routine is F08TEF (SSPGST/DSPGST).

## 9. Example

To compute all the eigenvalues of $Az = \lambda Bz$, where

$$A = \begin{pmatrix} -7.36 + 0.00i & 0.77 - 0.43i & -0.64 - 0.92i & 3.01 - 6.97i \\ 0.77 + 0.43i & 3.49 + 0.00i & 2.19 + 4.45i & 1.90 + 3.73i \\ -0.64 + 0.92i & 2.19 - 4.45i & 0.12 + 0.00i & 2.88 - 3.17i \\ 3.01 + 6.97i & 1.90 - 3.73i & 2.88 + 3.17i & -2.54 + 0.00i \end{pmatrix}$$

and

$$B = \begin{pmatrix} 3.23 + 0.00i & 1.51 - 1.92i & 1.90 + 0.84i & 0.42 + 2.50i \\ 1.51 + 1.92i & 3.58 + 0.00i & -0.23 + 1.11i & -1.18 + 1.37i \\ 1.90 - 0.84i & -0.23 - 1.11i & 4.09 + 0.00i & 2.33 - 0.14i \\ 0.42 - 2.50i & -1.18 - 1.37i & 2.33 + 0.14i & 4.29 + 0.00i \end{pmatrix},$$

using packed storage. Here $B$ is Hermitian positive-definite and must first be factorized by F07GRF (CPPTRF/ZPPTRF). The program calls F08TSF (CHPGST/ZHPGST) to reduce the problem to the standard form $Cy = \lambda y$; then F08GSF (CHPTRD/ZHPTRD) to reduce $C$ to tridiagonal form, and F08JFF (SSTERF/DSTERF) to compute the eigenvalues.

### 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.**

```
*     F08TSF Example Program Text
*     Mark 16 Release. NAG Copyright 1992.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX
      PARAMETER        (NMAX=8)
*     .. Local Scalars ..
      INTEGER          I, INFO, J, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          AP(NMAX*(NMAX+1)/2), BP(NMAX*(NMAX+1)/2),
     +                 TAU(NMAX)
      real             D(NMAX), E(NMAX-1)
*     .. External Subroutines ..
      EXTERNAL         ssterf, chpgst, chptrd, cpptrf
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08TSF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) 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)
            READ (NIN,*) ((BP(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)
            READ (NIN,*) ((BP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
*
*        Compute the Cholesky factorization of B
*
```

```
         CALL cpptrf(UPLO,N,BP,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.GT.0) THEN
            WRITE (NOUT,*) 'B is not positive-definite.'
         ELSE
*
*           Reduce the problem to standard form C*y = lambda*y, storing
*           the result in A
*
            CALL chpgst(1,UPLO,N,AP,BP,INFO)
*
*           Reduce C to tridiagonal form T = (Q**H)*C*Q
*
            CALL chptrd(UPLO,N,AP,D,E,TAU,INFO)
*
*           Calculate the eigenvalues of T (same as C)
*
            CALL ssterf(N,D,E,INFO)
*
            IF (INFO.GT.0) THEN
               WRITE (NOUT,*) 'Failure to converge.'
            ELSE
*
*              Print eigenvalues
*
               WRITE (NOUT,*) 'Eigenvalues'
               WRITE (NOUT,99999) (D(I),I=1,N)
            END IF
         END IF
      END IF
      STOP
*
99999 FORMAT (3X,(9F8.4))
      END
```

## 9.2. Program Data

```
F08TSF Example Program Data
  4                                                              :Value of N
  'L'                                                            :Value of UPLO
 (-7.36, 0.00)
 ( 0.77, 0.43) ( 3.49, 0.00)
 (-0.64, 0.92) ( 2.19,-4.45) ( 0.12, 0.00)
 ( 3.01, 6.97) ( 1.90,-3.73) ( 2.88, 3.17) (-2.54, 0.00)  :End of matrix A
 ( 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 B
```

## 9.3. Program Results

```
 F08TSF Example Program Results

 Eigenvalues
    -5.9990 -2.9936  0.5047  3.9990
```

# Chapter F11 – Sparse Linear Algebra

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

| Routine Name | Mark of Introduction | Purpose |
|---|---|---|
| F11BAF | 18 | Real sparse nonsymmetric linear systems, set-up for F11BBF |
| F11BBF | 18 | Real sparse nonsymmetric linear systems, preconditioned RGMRES, CGS or Bi-CGSTAB |
| F11BCF | 18 | Real sparse nonsymmetric linear systems, diagnostic for F11BBF |
| F11DAF | 18 | Real sparse nonsymmetric linear systems, incomplete $LU$ factorization |
| F11DBF | 18 | Solution of linear system involving incomplete $LU$ preconditioning matrix generated by F11DAF |
| F11DCF | 18 | Solution of real sparse nonsymmetric linear system, RGMRES, CGS or Bi-CGSTAB method, preconditioner computed by F11DAF (Black Box) |
| F11DDF | 18 | Solution of linear system involving pre-conditioning matrix gnerated by applying SSOR to real sparse nonsymmetric matrix |
| F11DEF | 18 | Solution of real sparse nonsymmetric linear system, RGMRES, CGS, or Bi-CGSTAB method, Jacobi or SSOR preconditioner (Black Box) |
| F11GAF | 17 | Real sparse symmetric linear systems, set-up for F11GBF |
| F11GBF | 17 | Real sparse symmetric linear systems, pre-conditioned conjugate gradient or Lanczos |
| F11GCF | 17 | Real sparse symmetric linear systems, diagnostic for F11GBF |
| F11JAF | 17 | Real sparse symmetric matrix, incomplete Cholesky factorization |
| F11JBF | 17 | Solution of linear system involving incomplete Cholseky preconditioning matrix generated by F11JAF |
| F11JCF | 17 | Solution of real sparse symmetric linear system, conjugate gradient/Lanczos method, preconditioner computed by F11JAF (Black Box) |
| F11JDF | 17 | Solution of linear system involving preconditioning matrix generated by applying SSOR to real sparse symmetric matrix |
| F11JEF | 17 | Solution of real sparse symmetric linear system, conjugate gradient/Lanczos method, Jacobi or SSOR preconditioner (Black Box) |
| F11XAF | 18 | Real sparse nonsymmetric matrix vector multiply |
| F11XEF | 17 | Real sparse symmetric matrix vector multiply |
| F11ZAF | 18 | Real sparse nonsymmetric matrix reorder routine |
| F11ZBF | 17 | Real sparse symmetric matrix reorder routine |

# Chapter F11

# Sparse Linear Algebra

## Contents

# 1 Scope of the Chapter

This chapter provides routines for the solution of large sparse systems of simultaneous linear equations. These include **iterative** methods for real nonsymmetric and symmetric linear systems. Some further direct methods are currently available in Chapter F01 and Chapter F04, and will be added to this chapter at future marks.

For a wider selection of routines for sparse linear algebra, users are referred to the Harwell Sparse Matrix Library (available from NAG), especially for direct methods for solving linear systems (see Section 2.2).

# 2 Background to the Problems

This section is only a brief introduction to the solution of sparse linear systems. For a more detailed discussion see for example Duff *et al.* [2] for direct methods, or Barrett *et al.* [1] for iterative methods.

## 2.1 Sparse Matrices and Their Storage

A matrix $A$ may be described as **sparse** if the number of zero elements is sufficiently large that it is worthwhile using algorithms which avoid computations involving zero elements.

If $A$ is sparse, and the chosen algorithm requires the matrix coefficients to be stored, a significant saving in storage can often be made by storing only non-zero elements. A number of different formats may be used to represent sparse matrices economically. These differ according to the amount of storage required, the amount of indirect addressing required for fundamental operations such as matrix–vector products, and their suitability for vector and/or parallel architectures. For a survey of some of these storage formats see Barrett *et al.* [1].

Some of the routines in this chapter have been designed to be independent of the matrix storage format. This allows the user to choose his or her own preferred format, or to avoid storing the matrix altogether. Other routines are black boxes, which are easier to use, but are based on fixed storage formats. Two such fixed formats are currently catered for. These are known as coordinate storage (CS) format, and symmetric coordinate storage (SCS) format.

### 2.1.1 Coordinate storage (CS) format

This storage format represents a sparse nonsymmetric matrix $A$, with NNZ non-zero elements, in terms of a real array A and two integer arrays IROW and ICOL. These arrays are all of rank 1 and of dimension at least NNZ. A contains the non-zero elements themselves, while IROW and ICOL store the corresponding row and column indices respectively.

For example, the matrix

$$A = \begin{pmatrix} 1 & 2 & -1 & -1 & -3 \\ 0 & -1 & 0 & 0 & -4 \\ 3 & 0 & 0 & 0 & 2 \\ 2 & 0 & 4 & 1 & 1 \\ -2 & 0 & 0 & 0 & 1 \end{pmatrix}$$

might be represented in the arrays A, IROW and ICOL as

A = (1,2,−1,−1,−3,−1,−4,3,2,2,4,1,1,−2,1)

IROW = (1,1,1,1,1,2,2,3,3,4,4,4,4,5,5)

ICOL = (1,2,3,4,5,2,5,1,5,1,3,4,5,1,5)

Notes

(i) The general format specifies no ordering of the array elements, but some routines may impose a specific ordering. For example, the non-zero elements may be required to be ordered by increasing row index and by increasing column index within each row, as in the example above. A utility routine is provided to order the elements appropriately.

(ii) With this storage format it is possible to enter duplicate elements. These may be interpreted in various ways (raising an error, ignoring all but the first entry, all but the last, or summing, for example).

### 2.1.2 Symmetric coordinate storage (SCS) format

This storage format is suitable for symmetric matrices, and is identical to the CS format described in Section 2.1.1, except that only the lower triangular non-zero elements are stored. Thus, for example, the matrix

$$A = \begin{pmatrix} 4 & 1 & 0 & 0 & -1 & 2 \\ 1 & 5 & 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 1 & 0 & -1 \\ 0 & 2 & 1 & 3 & 1 & 0 \\ -1 & 0 & 0 & 1 & 4 & 0 \\ 2 & 0 & -1 & 0 & 0 & 3 \end{pmatrix}$$

might be represented in the arrays A, IROW and ICOL as

$A$ = (4,1,5,2,2,1,3,−1,1,4,2,−1,3)

IROW = (1,2,2,3,4,4,4,5,5,5,6,6,6)

ICOL = (1,1,2,3,2,3,4,1,4,5,1,3,6)

## 2.2 Direct Methods

**Direct** methods for the solution of the linear algebraic system

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

aim to determine the solution vector $x$ in a fixed number of arithmetic operations, which is determined a priori by the number of unknowns. For example, an $LU$ factorization of $A$ followed by forward and backward substitution is a direct method for (1).

If the matrix $A$ is sparse it is possible to design direct methods which exploit the sparsity pattern and are therefore much more computationally efficient than the algorithms in Chapter F07, which in general take no account of sparsity. However, if the matrix is very large and sparse, then **iterative** methods (see Section 2.3) are generally more efficient still.

This chapter currently provides direct methods for sparse real nonsymmetric, and symmetric positive-definite systems. Further direct methods may be found in Chapter F01, Chapter F04 and Chapter F07, and will be introduced into Chapter F11 at a future mark. For more details see Section 3.4.

## 2.3 Iterative Methods

In contrast to the direct methods discussed in Section 2.2 **iterative** methods for (1) approach the solution through a sequence of approximations, until some user-specified termination criterion is met or until some predefined maximum number of iterations has been carried out. The number of iterations required for convergence is not generally known in advance, as it depends on the accuracy required, and on the matrix $A$ — its sparsity pattern, conditioning and eigenvalue spectrum.

Faster convergence can often be achieved using a **preconditioner** (Golub and Van Loan [3], Barrett *et al.* [1]). A preconditioner maps the original system of equations onto a different system

$$\bar{A}\bar{x} = \bar{b}, \tag{2}$$

which hopefully exhibits better convergence characteristics: for example, the condition number of the matrix $\bar{A}$ may be better than that of $A$, or it may have eigenvalues of greater multiplicity.

An unsuitable preconditioner or no preconditioning at all may result in a very slow rate or lack of convergence. However, preconditioning involves a trade-off between the reduction in the number of iterations required for convergence and the additional computational costs per iteration. Also, setting up a preconditioner may involve non-negligible overheads. The application of preconditioners to nonsymmetric and symmetric systems of equations is further considered in Section 2.4 and Section 2.5.

## 2.4 Iterative Methods for Nonsymmetric Linear Systems

Many of the most effective iterative methods for the solution of (1) lie in the class of non-stationary **Krylov subspace methods** [1]. For **nonsymmetric** matrices this class includes the restarted generalized minimum residual (RGMRES) method [8], conjugate gradient squared (CGS) method [10], and stabilized bi-conjugate gradient (Bi-CGSTAB) method [11], [9]. Here we just give a brief overview of these algorithms as implemented in Chapter F11. For full details see Section 3 of the document for F11BAF.

RGMRES is based on the Arnoldi method, which explicitly generates an orthogonal basis for the Krylov subspace span$\{A^k r_0\}$, $k = 0, 1, 2, \ldots$, where $r_0$ is the initial residual. The solution is then expanded onto the orthogonal basis so as to minimize the residual norm. For nonsymmetric matrices the generation of the basis requires a 'long' recurrence relation, resulting in prohibitive computational and storage costs. RGMRES limits these costs by restarting the Arnoldi process from the latest available residual every $m$ iterations. The value of $m$ is chosen in advance and is fixed throughout the computation. Unfortunately, an optimum value of $m$ cannot easily be predicted.

CGS is a development of the bi-conjugate gradient method where the nonsymmetric Lanczos method is applied to reduce the coefficient matrix to real tridiagonal form: two bi-orthogonal sequences of vectors are generated starting from the initial residual $r_0$ and from the *shadow residual* $\hat{r}_0$ corresponding to the arbitrary problem $A^T\hat{x} = \hat{b}$, where $\hat{b}$ is chosen so that $r_0 = \hat{r}_0$. In the course of the iteration, the residual and shadow residual $r_i = P_i(A)r_0$ and $\hat{r}_i = P_i(A^T)\hat{r}_0$ are generated, where $P_i$ is a polynomial of order $i$, and bi-orthogonality is exploited by computing the vector product $\rho_i = (\hat{r}_i, r_i) = (P_i(A^T)\hat{r}_0, P_i(A)r_0) = (\hat{r}_0, P_i^2(A)r_0)$. Applying the 'contraction' operator $P_i(A)$ twice, the iteration coefficients can still be recovered without advancing the solution of the shadow problem, which is of no interest. The CGS method often provides fast convergence; however, there is no reason why the contraction operator should also reduce the once reduced vector $P_i(A)r_0$: this can lead to a highly irregular convergence.

Bi-CGSTAB ($\ell$)) is similar to the CGS method. However, instead of generating the sequence $\{P_i^2(A)r_0\}$, it generates the sequence $\{Q_i(A)P_i(A)r_0\}$ where the $Q_i(A)$ are polynomials chosen to minimize the residual *after* the application of the contraction operator $P_i(A)$. Two main steps can be identified for each iteration: an OR (Orthogonal Residuals) step where a basis of order $\ell$ is generated by a Bi-CG iteration and an MR (Minimum Residuals) step where the residual is minimized over the basis generated, by a method akin to GMRES. For $\ell = 1$, the method corresponds to the Bi-CGSTAB method of van der Vorst [11]. For $\ell > 1$, more information about complex eigenvalues of the iteration matrix can be taken into account, and this may lead to improved convergence and robustness. However, as $\ell$ increases, numerical instabilities may arise.

Faster convergence can usually be achieved by using a **preconditioner**. A *left* preconditioner $M^{-1}$ can be used by the RGMRES and CGS methods, such that $\bar{A} = M^{-1}A \sim I_n$ in (2), where $I_n$ is the identity matrix of order $n$; a *right* preconditioner $M^{-1}$ can be used by the Bi-CGSTAB ($\ell$) method, such that $\bar{A} = AM^{-1} \sim I_n$. These are formal definitions, used only in the design of the algorithms; in practice, only the means to compute the matrix–vector products $v = Au$ and $v = A^T u$ (the latter only being required when an estimate of $\|A\|_1$ or $\|A\|_\infty$ is computed internally), and to solve the preconditioning equations $Mv = u$ are required, that is, explicit information about $M$, or its inverse is not required at any stage.

Preconditioning matrices $M$ are typically based on incomplete factorizations [6], or on the approximate inverses occurring in stationary iterative methods (see Young [12]). A common example is the **incomplete** $LU$ **factorization**

$$M = PLDUQ = A - R$$

where $L$ is lower triangular with unit diagonal elements, $D$ is diagonal, $U$ is upper triangular with unit diagonals, $P$ and $Q$ are permutation matrices, and $R$ is a remainder matrix. A **zero-fill** incomplete $LU$ factorization is one for which the matrix

$$S = P(L + D + U)Q$$

has the same pattern of non-zero entries as $A$. This is obtained by discarding any **fill** elements (non-zero elements of $S$ arising during the factorization in locations where $A$ has zero elements). Allowing some of these fill elements to be kept rather than discarded generally increases the accuracy of the factorization at the expense of some loss of sparsity. For further details see [1].

## 2.5 Iterative Methods for Symmetric Linear Systems

Two of the best known iterative methods applicable to symmetric linear systems are the conjugate gradient (CG) method [4], [3] and a Lanczos type method based on SYMMLQ [7].

For the CG method the matrix $A$ should ideally be positive-definite. The application of CG to indefinite matrices may lead to failure, or to lack of convergence. The SYMMLQ method is suitable for both positive-definite and indefinite symmetric matrices. It is more robust than CG, but less efficient when $A$ is positive-definite.

Both methods start from the residual $r_0 = b - Ax_0$, where $x_0$ is an initial estimate for the solution (often $x_0 = 0$), and generate an orthogonal basis for the Krylov subspace span$\{A^k r_0\}$, for $k = 0, 1, \ldots$, by means of three-term recurrence relations (Golub and Van Loan [3]). A sequence of symmetric tridiagonal matrices $\{T_k\}$ is also generated. Here and in the following, the index $k$ denotes the iteration count. The resulting symmetric tridiagonal systems of equations are usually more easily solved than the original problem. A sequence of solution iterates $\{x_k\}$ is thus generated such that the sequence of the norms of the residuals $\{\|r_k\|\}$ converges to a required tolerance. Note that, in general, the convergence is not monotonic.

In exact arithmetic, after $n$ iterations, this process is equivalent to an orthogonal reduction of $A$ to symmetric tridiagonal form, $T_n = Q^T AQ$; the solution $x_n$ would thus achieve exact convergence. In finite-precision arithmetic, cancellation and round-off errors accumulate causing loss of orthogonality. These methods must therefore be viewed as genuinely iterative methods, able to converge to a solution **within a prescribed tolerance**.

The orthogonal basis is not formed explicitly in either method. The basic difference between the two methods lies in the method of solution of the resulting symmetric tridiagonal systems of equations: the CG method is equivalent to carrying out an $LDL^T$ (Cholesky) factorization whereas the Lanczos method (SYMMLQ) uses an $LQ$ factorization.

A preconditioner for these methods must be **symmetric and positive-definite**, i.e., representable by $M = EE^T$, where $M$ is non-singular, and such that $\bar{A} = E^{-1}AE^{-T} \sim I_n$ in (2), where $I_n$ is the identity matrix of order $n$. These are formal definitions, used only in the design of the algorithms; in practice, only the means to compute the matrix-vector products $v = Au$ and to solve the preconditioning equations $Mv = u$ are required.

Preconditioning matrices $M$ are typically based on incomplete factorizations [5], or on the approximate inverses occurring in stationary iterative methods (see Young [12]). A common example is the **incomplete Cholesky factorization**

$$M = PLDL^T P^T = A - R$$

where $P$ is a permutation matrix, $L$ is lower triangular with unit diagonal elements, $D$ is diagonal and $R$ is a remainder matrix. A **zero-fill** incomplete Cholesky factorization is one for which the matrix

$$S = P(L + D + L^T)P^T$$

has the same pattern of non-zero entries as $A$. This is obtained by discarding any **fill** elements (non-zero elements of $S$ arising during the factorization in locations where $A$ has zero elements). Allowing some of these fill elements to be kept rather than discarded generally increases the accuracy of the factorization at the expense of some loss of sparsity. For further details see [1].

# 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 Types of Routine Available

The routines available in this chapter divide essentially into three types: basic routines, utility routines and black boxes.

**Basic routines** are grouped in suites of three, and implement the underlying iterative method. Each suite comprises a set-up routine, a solver, and a routine to return additional information. The solver routine is independent of the matrix storage format (indeed the matrix need not be stored at all) and the

type of preconditioner. It uses **reverse communication**, i.e., it returns repeatedly to the calling program with the parameter IREVCM set to specified values which require the calling program to carry out a specific task (either to compute a matrix-vector product or to solve the preconditioning equation), to signal the completion of the computation or to allow the calling program to monitor the solution. Reverse communication has the following advantages.

(i) Maximum flexibility in the representation and storage of sparse matrices. All matrix operations are carried out outside the solver routine, thereby avoiding the need for a complicated interface with enough flexibility to cope with all types of storage schemes and sparsity patterns. This applies also to preconditioners.
(ii) Enhanced user interaction: the progress of the solution can be closely monitored by the user and tidy or immediate termination can be requested. This is useful, for example, when alternative termination criteria are to be employed or in case of failure of the external routines used to perform matrix operations.

At present there are two suites of basic routines, for real symmetric, and for real nonsymmetric systems, respectively.

**Utility routines** perform such tasks as initializing the preconditioning matrix $M$, solving linear systems involving $M$, or computing matrix-vector products, for particular preconditioners and matrix storage formats. Used in combination basic routines and utility routines therefore provide iterative methods with a considerable degree of flexibility; allowing the user to select from different termination criteria, monitor the approximate solution, and compute various diagnostic parameters. The tasks of computing the matrix-vector products and dealing with the preconditioner are removed from the user, but at the expense of sacrificing some flexibility in the choice of preconditioner and matrix storage format.

**Black box** routines call basic and utility routines in order to provide easy-to-use routines for particular preconditioners and sparse matrix storage formats. They are much less flexible than the basic routines, but do not use reverse communication, and may be suitable in many simple cases.

The structure of this chapter has been designed to cater for as many types of application as possible. If a black box exists which is suitable for a given application you are recommended to use it. If you then decide you need some additional flexibility it is easy to achieve this by using basic and utility routines which reproduce the algorithm used in the black box, but allow more access to algorithmic control parameters and monitoring. If you wish to use a preconditioner or storage format for which no utility routines are provided, you must call basic routines, and provide your own utility routines.

## 3.2 Iterative Methods for Nonsymmetric Linear Systems

The suite of basic routines F11BAF, F11BBF and F11BCF implements either RGMRES, CGS, or Bi-CGSTAB($\ell$), for the iterative solution of the sparse nonsymmetric linear system $Ax = b$. These routines allow a choice of termination criteria and the norms used in them, allow monitoring of the approximate solution, and can return estimates of the norm of $A$ and the largest singular value of the preconditioned matrix $\bar{A}$.

In general, it is not possible to recommend one of these methods in preference to another. RGMRES is popular, but requires the most storage, and can easily stagnate when the size $m$ of the orthogonal basis is too small, or the preconditioner is not good enough. CGS can be the fastest method, but the computed residuals can exhibit instability which may greatly affect the convergence and quality of the solution. Bi-CGSTAB($\ell$) seems robust and reliable, but it can be slower than the other methods. Some further discussion of the relative merits of these methods can be found in [1].

The utility routines provided for nonsymmetric matrices use the co-ordinate storage (CS) format described in Section 2.1.1. F11DAF computes a preconditioning matrix based on incomplete $LU$ factorization, and F11DBF solves linear systems involving the preconditioner generated by F11DAF. The amount of fill-in occurring in the incomplete factorization can be controlled by specifying either the level of fill, or the drop tolerance. Partial or complete pivoting may optionally be employed, and the factorization can be modified to preserve row-sums.

F11DDF is similar to F11DBF, but solves linear systems involving the preconditioner corresponding to symmetric successive-over-relaxation (SSOR). The value of the relaxation parameter $\omega$ must currently be supplied by the user. Automatic procedures for choosing $\omega$ will be included in the chapter at a future mark.

F11XAF computes matrix-vector products for nonsymmetric matrices stored in ordered CS format. An additional utility routine F11ZAF orders the non-zero elements of a sparse nonsymmetric matrix stored in general CS format.

The black box routine F11DCF makes calls to F11BAF, F11BBF, F11BCF, F11DBF and F11XAF, to solve a sparse nonsymmetric linear system, represented in CS format, using RGMRES, CGS, or Bi-CGSTAB($\ell$), with incomplete $LU$ preconditioning. F11DEF is similar, but has options for no preconditioning, Jacobi preconditioning or SSOR preconditioning.

## 3.3 Iterative Methods for Symmetric Linear Systems

The suite of basic routines F11GAF, F11GBF and F11GCF implement either the conjugate gradient (CG) method, or a Lanczos method based on SYMMLQ, for the iterative solution of the sparse symmetric linear system $Ax = b$. If $A$ is known to be positive-definite the CG method should be chosen; the Lanczos method is more robust but less efficient for positive-definite matrices. These routines allow a choice of termination criteria and the norms used in them, allow monitoring of the approximate solution, and can return estimates of the norm of $A$ and the largest singular value of the preconditioned matrix $\bar{A}$.

The utility routines provided for symmetric matrices use the symmetric co-ordinate storage (SCS) format described in Section 2.1.2. F11JAF computes a preconditioning matrix based on incomplete Cholesky factorization, and F11JBF solves linear systems involving the preconditioner generated by F11JAF. The amount of fill-in occurring in the incomplete factorization can be controlled by specifying either the level of fill, or the drop tolerance. Diagonal Markowitz pivoting may optionally be employed, and the factorization can be modified to preserve row-sums.

F11JDF is similar to F11JBF, but solves linear systems involving the preconditioner corresponding to symmetric successive-over-relaxation (SSOR). The value of the relaxation parameter $\omega$ must currently be supplied by the user. Automatic procedures for choosing $\omega$ will be included in the chapter at a future mark.

F11XEF computes matrix-vector products for symmetric matrices stored in ordered SCS format. An additional utility routine F11ZBF orders the non-zero elements of a sparse symmetric matrix stored in general SCS format.

The black box routine F11JCF makes calls to F11GAF, F11GBF, F11GCF, F11JBF and F11XEF, to solve a sparse symmetric linear system, represented in SCS format, using a conjugate gradient or Lanczos method, with incomplete Cholesky preconditioning. F11JEF is similar, but has options for no preconditioning, Jacobi preconditioning or SSOR preconditioning.

## 3.4 Direct Methods

Chapter F11 does not currently provide any routines **specifically** designed for direct solution of sparse linear systems. However, the arguments of F11DAF and F11DBF may be chosen in such a way as to produce a direct solution.

The routine F11DBF solves a linear system involving the incomplete $LU$ preconditioning matrix

$$M = PLDUQ = A - R$$

generated by F11DAF, where $P$ and $Q$ are permutation matrices, $L$ is lower triangular with unit diagonal elements, $U$ is upper triangular with unit diagonal elements, $D$ is diagonal and $R$ is a remainder matrix.

If $A$ is non-singular, a call to F11DAF with LFILL $< 0$ and DTOL $= 0.0$ results in a zero remainder matrix $R$ and a **complete** factorization. A subsequent call to F11DBF will therefore result in a direct method for real sparse nonsymmetric systems.

If $A$ is known to be symmetric positive-definite, F11JAF and F11JBF may similarly be used to give a direct solution. For further details see Section 8.4 of the document for F11JAF.

Routines specifically designed for direct solution of sparse linear systems can currently be found in Chapter F01, Chapter F04 and Chapter F07. In particular, the following routines allow the direct solution of nonsymmetric systems:

| | |
|---|---|
| Band | F07BDF and F07BEF |
| Almost block-diagonal | F01LHF and F04LHF |
| Tridiagonal | F01LEF and F04LEF or F04EAF |
| Sparse | F01BRF (or F01BSF) and F04AXF |

and the following routines allow the direct solution of symmetric positive-definite systems:

| | |
|---|---|
| Band | F07HDF and F07HEF |
| Variable band (skyline) | F01MCF and F04MCF |
| Tridiagonal | F04FAF |

# 4 Index



# 5 References


**[1]** Barrett R, Berry M, Chan T F, Demmel J, Donato J, Dongarra J, Eijkhout V, Pozo R, Romine C and van der Vorst H (1994) *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods* SIAM, Philadelphia

**[2]** Duff I S, Erisman A M, Reid J K (1986) *Direct Methods for Sparse Matrices* Oxford University Press, London

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

**[4]** Hestenes M and Stiefel E (1952) Methods of conjugate gradients for solving linear systems *J. Res. Nat. Bur. Stand.* **49** 409–436

**[5]** Meijerink J and van der Vorst H (1977) An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix *Math. Comput.* **31** 148–162

[6] Meijerink J and van der Vorst H (1981) Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems *J. Comput. Phys.* **44** 134–155

[7] Paige C C and Saunders M A (1975) Solution of sparse indefinite systems of linear equations *SIAM J. Numer. Anal.* **12** 617–629

[8] Saad Y and Schultz M (1986) GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems *SIAM J. Sci. Statist. Comput.* **7** 856–869

[9] Sleijpen G L G and Fokkema D R (1993) BiCGSTAB($\ell$) for linear equations involving matrices with complex spectrum *ETNA* **1** 11–32

[10] Sonneveld P (1989) CGS, a fast Lanczos-type solver for nonsymmetric linear systems *SIAM J. Sci. Statist. Comput.* **10** 36–52

[11] van der Vorst H (1989) Bi-CGSTAB, A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems *SIAM J. Sci. Statist. Comput.* **13** 631–644

[12] Young D (1971) *Iterative Solution of Large Linear Systems* Academic Press, New York

# F11BAF – 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.

## 1 Purpose

F11BAF is a set-up routine, the first in a suite of three routines for the iterative solution of a general (nonsymmetric) system of simultaneous linear equations. F11BAF must be called before F11BBF, the iterative solver. The third routine in the suite, F11BCF, can be used to return additional information about the computation.

These three routines are suitable for the solution of large sparse general (nonsymmetric) systems of equations.

## 2 Specification

```
SUBROUTINE F11BAF(METHOD, PRECON, NORM, WEIGHT, ITERM, N, M, TOL,
1                 MAXITN, ANORM, SIGMAX, MONIT, LWREQ, IFAIL)
INTEGER           ITERM, N, M, MAXITN, MONIT, LWREQ, IFAIL
real              TOL, ANORM, SIGMAX
CHARACTER*1       PRECON, NORM, WEIGHT
CHARACTER*(*)     METHOD
```

## 3 Description

The suite consisting of the routines F11BAF, F11BBF, F11BCF is designed to solve the general (nonsymmetric) system of simultaneous linear equations $Ax = b$ of order $n$, where $n$ is large and the coefficients matrix $A$ is sparse.

F11BAF is a set-up routine which must be called before F11BBF, the iterative solver. The third routine in the suite, F11BCF, can be used to return additional information about the computation. A choice of methods is available:

restarted generalized minimum residual method (RGMRES);

conjugate gradient squared method (CGS);

bi-conjugate gradient stabilized ($\ell$) method (Bi-CGSTAB ($\ell$)).

### 3.1 Restarted Generalized Minimum Residual Method (RGMRES)

The restarted generalized minimum residual method (RGMRES) (Saad and Schultz [4], Barrett *et al.* [2], Dias da Cunha and Hopkins [3]) starts from the residual $r_0 = b - Ax_0$, where $x_0$ is an initial estimate for the solution (often $x_0 = 0$). An orthogonal basis for the Krylov subspace span$\{A^k r_0\}$, for $k = 0, 1, 2, \ldots$, is generated explicitly: this is referred to as Arnoldi's method (Arnoldi [8]). The solution is then expanded onto the orthogonal basis so as to minimize the residual norm $\|b - Ax\|_2$. The lack of symmetry of $A$ implies that the orthogonal basis is generated by applying a 'long' recurrence relation, whose length increases linearly with the iteration count. For all but the most trivial problems, computational and storage costs can quickly become prohibitive as the iteration count increases. RGMRES limits these costs by employing a restart strategy: every $m$ iterations at most, the Arnoldi process is restarted from $r_l = b - Ax_l$, where the subscript $l$ denotes the last available iterate. Each group of $m$ iterations is referred to as a 'super-iteration'. The value of $m$ is chosen in advance and is fixed throughout the computation. Unfortunately, an optimum value of $m$ cannot easily be predicted.

### 3.2 Conjugate Gradient Squared Method (CGS)

The conjugate gradient squared method (CGS) (Sonneveld [5], Barrett *et al.* [2], Dias da Cunha and Hopkins [3]) is a development of the bi-conjugate gradient method where the nonsymmetric Lanczos method is applied to reduce the coefficients matrix to real tridiagonal form: two bi-orthogonal sequences

of vectors are generated starting from the residual $r_0 = b - Ax_0$, where $x_0$ is an initial estimate for the solution (often $x_0 = 0$) and from the *shadow residual* $\hat{r}_0$ corresponding to the arbitrary problem $A^T\hat{x} = \hat{b}$, where $\hat{b}$ can be any vector, but in practice is chosen so that $r_0 = \hat{r}_0$. In the course of the iteration, the residual and shadow residual $r_i = P_i(A)r_0$ and $\hat{r}_i = P_i(A^T)\hat{r}_0$ are generated, where $P_i$ is a polynomial of order $i$, and bi-orthogonality is exploited by computing the vector product $\rho_i = (\hat{r}_i, r_i) = (P_i(A^T)\hat{r}_0, P_i(A)r_0) = (\hat{r}_0, P_i^2(A)r_0)$. Applying the 'contraction' operator $P_i(A)$ twice, the iteration coefficients can still be recovered without advancing the solution of the shadow problem, which is of no interest. The CGS method often provides fast convergence; however, there is no reason why the contraction operator should also reduce the once reduced vector $P_i(A)r_0$: this may well lead to a highly irregular convergence which may result in large cancellation errors.

## 3.3 Bi-Conjugate Gradient Stabilized ($\ell$) Method (Bi-CGSTAB ($\ell$))

The bi-conjugate gradient stabilized ($\ell$) method (Bi-CGSTAB ($\ell$)) (van der Vorst [6], Sleijpen and Fokkema [7], Dias da Cunha and Hopkins [3]) is similar to the CGS method above. However, instead of generating the sequence $\{P_i^2(A)r_0\}$, it generates the sequence $\{Q_i(A)P_i(A)r_0\}$ where the $Q_i(A)$ are polynomials chosen to minimize the residual *after* the application of the contraction operator $P_i(A)$. Two main steps can be identified for each iteration: an OR (Orthogonal Residuals) step where a basis of order $\ell$ is generated by a Bi-CG iteration and an MR (Minimum Residuals) step where the residual is minimized over the basis generated, by a method akin to GMRES. For $\ell = 1$, the method corresponds to the Bi-CGSTAB method of van der Vorst [6]. For $\ell > 1$, more information about complex eigenvalues of the iteration matrix can be taken into account, and this may lead to improved convergence and robustness. However, as $\ell$ increases, numerical instabilities may arise. For this reason, a maximum value of $\ell = 10$ is imposed, but probably $\ell = 4$ is sufficient in many cases.

## 3.4 General Considerations

For each method, a sequence of solution iterates $\{x_i\}$ is generated such that, hopefully, the sequence of the residual norms $\{\|r_i\|\}$ converges to a required tolerance. Note that, in general, convergence, when it occurs, is not monotonic.

In the RGMRES and Bi-CGSTAB ($\ell$) methods above, the user's program must provide the **maximum** number of basis vectors used, $m$ or $\ell$, respectively; however, a **smaller** number of basis vectors may be generated and used when the stability of the solution process requires this (see Section 8).

Faster convergence can be achieved using a **preconditioner** (Golub and Van Loan [1], Barrett *et al.* [2]). A preconditioner maps the original system of equations onto a different system, say

$$\bar{A}\bar{x} = \bar{b}, \tag{1}$$

with, hopefully, better characteristics with respect to its speed of convergence: for example, the condition number of coefficients matrix can be improved or eigenvalues in its spectrum can be made to coalesce. An orthogonal basis for the Krylov subspace span$\{\bar{A}^k\bar{r}_0\}$, for $k = 0, 1, \ldots$, is generated and the solution proceeds as outlined above. The algorithms used are such that the solution and residual iterates of the original system are produced, not their preconditioned counterparts. Note that an unsuitable preconditioner or no preconditioning at all may result in a very slow rate, or lack, of convergence. However, preconditioning involves a trade-off between the reduction in the number of iterations required for convergence and the additional computational costs per iteration. Also, setting up a preconditioner may involve non-negligible overheads.

A *left* preconditioner $M^{-1}$ can be used by the RGMRES and CGS methods, such that $\bar{A} = M^{-1}A \sim I_n$ in (1), where $I_n$ is the identity matrix of order $n$; a *right* preconditioner $M^{-1}$ can be used by the Bi-CGSTAB ($\ell$) method, such that $\bar{A} = AM^{-1} \sim I_n$. These are formal definitions, used only in the design of the algorithms; in practice, only the means to compute the matrix–vector products $v = Au$ and $v = A^Tu$ (the latter only being required when an estimate of $\|A\|_1$ or $\|A\|_\infty$ is computed internally), and to solve the preconditioning equations $Mv = u$ are required, that is, explicit information about $M$, or its inverse is not required at any stage.

The first termination criterion

$$\|r_k\|_p \leq \tau(\|b\|_p + \|A\|_p \|x_k\|_p) \tag{2}$$

is also available for all three methods. In (2), $p = 1,\infty$ or 2 and $\tau$ denotes a user-specified tolerance subject to $\max(10,\sqrt{n})\epsilon \leq \tau < 1$, where $\epsilon$ is the ***machine precision***. Facilities are provided for the estimation of the norm of the coefficients matrix $\|A\|_1$ or $\|A\|_\infty$, when this is not known in advance, by applying Higham's method (Higham [9]). Note that $\|A\|_2$ cannot be estimated internally. This criterion uses an error bound derived from **backward** error analysis to ensure that the computed solution is the exact solution of a problem as close to the original as the termination tolerance requires. Termination criteria employing bounds derived from **forward** error analysis are not used because any such criteria would require information about the condition number $\kappa(A)$ which is not easily obtainable.

The second termination criterion

$$\|\bar{r}_k\|_2 \leq \tau\,(\|\bar{r}_0\|_2 + \sigma_1(\bar{A})\,\|\Delta\bar{x}_k\|_2) \tag{3}$$

is also available for all three methods. In (3), $\sigma_1(\bar{A}) = \|\bar{A}\|_2$ is the largest singular value of the (preconditioned) iteration matrix $\bar{A}$. This termination criterion monitors the progress of the solution of the preconditioned system of equations and is less expensive to apply than criterion (2) for the Bi-CGSTAB ($\ell$) method with $\ell > 1$. Only the RGMRES method provides facilities to estimate $\sigma_1(\bar{A})$ internally, when this is not supplied (see Section 8).

Termination criterion (2) is the recommended choice, despite its additional costs per iteration when using the Bi-CGSTAB ($\ell$) method with $\ell > 1$. Also, if the norm of the initial estimate is much larger than the norm of the solution, that is, if $\|x_0\| \gg \|x\|$, a dramatic loss of significant digits could result in complete lack of convergence. The use of criterion (2) will enable the detection of such a situation, and the iteration will be restarted at a suitable point. No such restart facilities are provided for criterion (3).

Optionally, a vector $w$ of user-specified weights can be used in the computation of the vector norms in termination criterion (2), i.e., $\|v\|_p^{(w)} = \|v^{(w)}\|_p$, where $(v^{(w)})_i = w_i\,v_i$, for $i = 1,2,\ldots,n$. Note that the use of weights increases the computational costs.

The sequence of calls to the routines comprising the suite is enforced: first, the set-up routine F11BAF must be called, followed by the solver F11BBF. F11BCF can be called either when F11BBF is carrying out a monitoring step or after F11BBF has completed its tasks. Incorrect sequencing will raise an error condition.

In general, it is not possible to recommend one method in preference to another. RGMRES is often used in the solution of systems arising from PDEs. On the other hand, it can easily stagnate when the size $m$ of the orthogonal basis is too small, or the preconditioner is not good enough. CGS can be the fastest method, but the computed residuals can exhibit instability which may greatly affect the convergence and quality of the solution. Bi-CGSTAB ($\ell$) seems robust and reliable, but it can be slower than the other methods: if a preconditioner is used and $\ell > 1$, Bi-CGSTAB ($\ell$) computes the solution of the preconditioned system $\bar{x}_k = Mx_k$: the preconditioning equations must be solved to obtain the required solution. The algorithm employed limits to 10% or less, when no intermediate monitoring is requested, the number of times the preconditioner has to be thus applied compared with the total number of applications of the preconditioner. Also, when the termination criterion (2) is used, the CGS and Bi-CGSTAB ($\ell$) methods will restart the iteration automatically when necessary in order to solve the given problem.

# 4 References

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

[2] Barrett R, Berry M, Chan T F, Demmel J, Donato J, Dongarra J, Eijkhout V, Pozo R, Romine C and van der Vorst H (1994) *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods* SIAM, Philadelphia

[3] Dias da Cunha R and Hopkins T (1994) PIM 1.1 — the parallel iterative method package for systems of linear equations user's guide — Fortran 77 version *Technical Report* Computing Laboratory, University of Kent at Canterbury, Kent CT2 7NZ, UK

[4] Saad Y and Schultz M (1986) GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems *SIAM J. Sci. Statist. Comput.* **7** 856–869

[5] Sonneveld P (1989) CGS, a fast Lanczos-type solver for nonsymmetric linear systems *SIAM J. Sci. Statist. Comput.* **10** 36–52

[6] van der Vorst H (1989) Bi-CGSTAB, A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems *SIAM J. Sci. Statist. Comput.* **13** 631–644

[7] Sleijpen G L G and Fokkema D R (1993) BiCGSTAB($\ell$) for linear equations involving matrices with complex spectrum *ETNA* **1** 11–32

[8] Arnoldi W (1951) The principle of minimized iterations in the solution of the matrix eigenvalue problem *Quart. Appl. Math.* **9** 17–29

[9] Higham N J (1988) 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

# 5 Parameters

**1:** METHOD — CHARACTER*(*) *Input*

*On entry:* the iterative method to be used. The possible choices are:

'RGMRES' restarted generalized minimum residual method;
'CGS' conjugate gradient squared method;
'BICGSTAB' bi-conjugate gradient stabilized ($\ell$) method;

*Constraint:* METHOD = 'RGMRES', 'CGS' or 'BICGSTAB'.

**2:** PRECON — CHARACTER*1 *Input*

*On entry:* determines whether preconditioning is used. The possible choices are:

'N' no preconditioning;
'P' preconditioning.

*Constraint:* PRECON = 'N' or 'P'.

**3:** NORM — CHARACTER*1 *Input*

*On entry:* defines the matrix and vector norm to be used in the termination criteria. The possible choices are:

'1' use the $l_1$ norm;
'I' use the $l_\infty$ norm;
'2' use the $l_2$ norm.

*Suggested value:* NORM = 'I', if ITERM = 1; NORM = '2', if ITERM = 2.

*Constraints:* if ITERM = 1, then NORM = '1', 'I' or '2'; if ITERM = 2, then NORM = '2'.

**4:** WEIGHT — CHARACTER*1 *Input*

*On entry:* specifies whether a vector $w$ of user-supplied weights is to be used in the computation of the vector norms required in termination criterion (2) (ITERM = 1): $\|v\|_p^{(w)} = \|v^{(w)}\|_p$, where $v_i^{(w)} = w_i\, v_i$, for $i = 1, 2, \ldots, n$. The suffix $p = 1, 2, \infty$ denotes the vector norm used, as specified by the parameter NORM. Note that weights cannot be used when ITERM = 2, i.e., when criterion (3) is used. The possible choices are:

'W' user-supplied weights are to be used and must be supplied on initial entry to F11BBF.
'N' all weights are implicitly set equal to one. Weights do not need to be supplied on initial entry to F11BBF.

*Suggested value:* WEIGHT = 'N'.

*Constraints:* if ITERM = 1, then WEIGHT = 'W' or 'N'; if ITERM = 2, then WEIGHT = 'N'.

**5:** ITERM — INTEGER *Input*

*On entry:* defines the termination criterion to be used. The possible choices are:

1 use the termination criterion defined in (2);
2 use the termination criterion defined in (3).

*Suggested value:* ITERM = 1.

*Constraints:* ITERM = 1 or 2.

**6:** N — INTEGER *Input*

*On entry:* the order $n$ of the matrix $A$.

*Constraint:* N > 0.

**7:** M — INTEGER *Input*

*On entry:* if METHOD = 'RGMRES', M is the dimension $m$ of the restart subspace.

If METHOD = 'BICGSTAB', M is the order $\ell$ of the polynomial Bi-CGSTAB method; otherwise, M is not referenced.

*Constraints:*

if METHOD = 'RGMRES', 0 < M ≤ min(N,50);
if METHOD = 'BICGSTAB', 0 < M ≤ min(N,10).

**8:** TOL — ***real*** *Input*

*On entry:* the tolerance $\tau$ for the termination criterion. If TOL ≤ 0.0, $\tau = \max(\sqrt{\epsilon},\sqrt{n}\,\epsilon)$ is used, where $\epsilon$ is the ***machine precision***. Otherwise $\tau = \max(\text{TOL},10\epsilon,\sqrt{n}\,\epsilon)$ is used.

*Constraint:* TOL < 1.0.

**9:** MAXITN — INTEGER *Input*

*On entry:* the maximum number of iterations.

*Constraint:* MAXITN > 0.

**10:** ANORM — ***real*** *Input*

*On entry:* if ANORM > 0.0, the value of $\|A\|_p$ to be used in the termination criterion (2) (ITERM = 1).

If ANORM ≤ 0.0, ITERM = 1 and NORM = '1' or 'I', then $\|A\|_1 = \|A\|_\infty$ is estimated internally by F11BBF.

If ITERM = 2, then ANORM is not referenced.

*Constraint:* if ITERM = 1 and NORM = '2', then ANORM > 0.0.

**11:** SIGMAX — ***real*** *Input*

*On entry:* if ITERM = 2, the largest singular value $\sigma_1$ of the preconditioned iteration matrix; otherwise, SIGMAX is not referenced.

If SIGMAX ≤ 0.0, ITERM = 2 and METHOD = 'RGMRES', then the value of $\sigma_1$ will be estimated internally.

*Constraint:* if METHOD = 'CGS' or 'BICGSTAB' and ITERM = 2, then SIGMAX > 0.0.

**12:** MONIT — INTEGER *Input*

*On entry:* if MONIT > 0, the frequency at which a monitoring step is executed by F11BBF: if METHOD = 'CGS' a monitoring step is executed every MONIT iterations; otherwise, a monitoring step is executed every MONIT super-iterations (groups of up to $m$ or $\ell$ iterations for RGMRES or Bi-CGSTAB ($\ell$), respectively).

There are some additional computational costs involved in monitoring the solution and residual vectors when the Bi-CGSTAB ($\ell$) method is used with $\ell > 1$.

*Constraint:* MONIT $\leq$ MAXITN.

**13:** LWREQ — INTEGER *Output*

*On exit:* the minimum amount of workspace required by F11BBF. (See also Section 5 of the document for F11BBF.)

**14:** IFAIL — INTEGER *Input/Output*

*On entry:* IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

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

# 6 Errors and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL = $-i$

On entry, the $i$th argument had an illegal value.

IFAIL = 1

F11BAF has been called out of sequence.

# 7 Accuracy

Not applicable.

# 8 Further Comments

RGMRES can estimate internally the maximum singular value $\sigma_1$ of the iteration matrix, using $\sigma_1 \sim ||T||_1$ where $T$ is the upper triangular matrix obtained by $QR$ factorization of the upper Hessenberg matrix generated by the Arnoldi process. The computational costs of this computation are negligible when compared to the overall costs.

Loss of orthogonality in the RGMRES method, or of bi-orthogonality in the Bi-CGSTAB ($\ell$) method may degrade the solution and speed of convergence. For both methods, the algorithms employed include checks on the basis vectors so that the number of basis vectors used for a given super-iteration may be less than the value specified in the input parameter M. Also, the Bi-CGSTAB ($\ell$) method will automatically restart the computation from the last available iterates, when the stability of the solution process requires it.

Termination criterion (3) involves only the residual (or norm of the residual) produced directly by the iteration process: this may differ from the norm of the true residual $\tilde{r}_k = b - Ax_k$, particularly when the norm of the residual is very small. Also, if the norm of the initial estimate of the solution is much larger than the norm of the exact solution, convergence can be achieved despite very large errors in the solution. On the other hand, termination criterion (3) is cheaper to use and inspects the progress of the actual iteration. Termination criterion (2) should be preferred in most cases, despite its slightly larger costs.

# 9 Example

The example solves an $8 \times 8$ nonsymmetric system of simultaneous linear equations using the bi-conjugate gradient stabilized method of order $\ell = 2$, where the coefficients matrix $A$ has random sparsity pattern. An incomplete $LU$ preconditioner is used (routines F11DAF and F11DBF).

## 9.1 Example 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.

```
*     F11BAF Example Program Text
*     Mark 18 Release. NAG Copyright 1997.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LA, LIWORK, LWORK
      PARAMETER        (NMAX=1000,LA=10000,LIWORK=7*NMAX+2,LWORK=6*NMAX)
*     .. Local Scalars ..
      real             ANORM, DTOL, SIGMAX, STPLHS, STPRHS, TOL
      INTEGER          I, IFAIL, IFAIL1, IFAILX, IREVCM, ITERM, ITN,
     +                 LFILL, LWREQ, M, MAXITN, MONIT, N, NNZ, NNZC,
     +                 NPIVM
      LOGICAL          LOOP
      CHARACTER        MILU, NORM, PRECON, PSTRAT, WEIGHT
      CHARACTER*8      METHOD
*     .. Local Arrays ..
      real             A(LA), B(NMAX), WORK(LWORK), X(NMAX)
      INTEGER          ICOL(LA), IDIAG(NMAX), IPIVP(NMAX), IPIVQ(NMAX),
     +                 IROW(LA), ISTR(NMAX+1), IWORK(LIWORK)
*     .. External Subroutines ..
      EXTERNAL         F11BAF, F11BBF, F11BCF, F11DAF, F11DBF, F11XAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11BAF Example Program Results'
*
*     Skip heading in data file
*
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read or initialize the parameters for the iterative solver
*
         READ (NIN,*) METHOD
         READ (NIN,*) PRECON, NORM, WEIGHT, ITERM
         READ (NIN,*) M, TOL, MAXITN
         READ (NIN,*) MONIT
         ANORM = 0.0e0
         SIGMAX = 0.0e0
*
*        Read the parameters for the preconditioner
*
         READ (NIN,*) LFILL, DTOL
         READ (NIN,*) MILU, PSTRAT
*
```

```
*        Read the number of non-zero elements of the matrix A, then read
*        the non-zero elements
*
         READ (NIN,*) NNZ
         DO 20 I = 1, NNZ
            READ (NIN,*) A(I), IROW(I), ICOL(I)
   20    CONTINUE
*
*        Read right-hand side vector b and initial approximate solution x
*
         READ (NIN,*) (B(I),I=1,N)
         READ (NIN,*) (X(I),I=1,N)
*
*        Calculate incomplete LU factorization
*
         IFAIL = 0
         MILU = 'N'
         CALL F11DAF(N,NNZ,A,LA,IROW,ICOL,LFILL,DTOL,PSTRAT,MILU,IPIVP,
     +               IPIVQ,ISTR,IDIAG,NNZC,NPIVM,IWORK,LIWORK,IFAIL)
*
*        Call F11BAF to initialize the solver
*
         IFAIL = 0
         CALL F11BAF(METHOD,PRECON,NORM,WEIGHT,ITERM,N,M,TOL,MAXITN,
     +               ANORM,SIGMAX,MONIT,LWREQ,IFAIL)
*
*        Call repeatedly F11BBF to solve the equations
*        Note that the arrays B and X are overwritten
*
*        On final exit, X will contain the solution and B the residual
*        vector
*
         IFAIL = 0
         IREVCM = 0
         LOOP = .TRUE.
*
         LWREQ = LWORK
   40    CONTINUE
         CALL F11BBF(IREVCM,X,B,WORK,LWREQ,IFAIL)
         IF (IREVCM.EQ.-1) THEN
            CALL F11XAF('Transpose',N,NNZ,A,IROW,ICOL,'No checking',X,B,
     +                  IFAILX)
         ELSE IF (IREVCM.EQ.1) THEN
            CALL F11XAF('No transpose',N,NNZ,A,IROW,ICOL,'No checking',
     +                  X,B,IFAILX)
         ELSE IF (IREVCM.EQ.2) THEN
            IFAIL1 = -1
            CALL F11DBF('No transpose',N,A,LA,IROW,ICOL,IPIVP,IPIVQ,
     +                  ISTR,IDIAG,'No checking',X,B,IFAIL1)
            IF (IFAIL1.NE.0) IREVCM = 6
         ELSE IF (IREVCM.EQ.3) THEN
            IFAIL1 = 0
            CALL F11BCF(ITN,STPLHS,STPRHS,ANORM,SIGMAX,IFAIL1)
            WRITE (NOUT,99999) ITN, STPLHS
            WRITE (NOUT,99998)
            WRITE (NOUT,99997) (X(I),B(I),I=1,N)
         ELSE IF (IREVCM.EQ.4) THEN
```

```
            LOOP = .FALSE.
          END IF
          IF (LOOP) GO TO 40
*
*         Obtain information about the computation
*
          IFAIL1 = 0
          CALL F11BCF(ITN,STPLHS,STPRHS,ANORM,SIGMAX,IFAIL1)
*
*         Print the output data
*
          WRITE (NOUT,99996)
          WRITE (NOUT,99995) 'Number of iterations for convergence:    ',
     +      ITN
          WRITE (NOUT,99994) 'Residual norm:                           ',
     +      STPLHS
          WRITE (NOUT,99994) 'Right-hand side of termination criterion:',
     +      STPRHS
          WRITE (NOUT,99994) '1-norm of matrix A:                      ',
     +      ANORM
*
*         Output x
*
          WRITE (NOUT,99998)
          WRITE (NOUT,99997) (X(I),B(I),I=1,N)
        END IF
        STOP
*
99999 FORMAT (/1X,'Monitoring at iteration no.',I4,/1X,1P,'residual no',
     +       'rm: ',e14.4)
99998 FORMAT (/2X,'  Solution vector',2X,' Residual vector')
99997 FORMAT (1X,1P,e16.4,1X,e16.4)
99996 FORMAT (/1X,'Final Results')
99995 FORMAT (2X,A,I4)
99994 FORMAT (1X,A,1P,e14.4)
      END
```

## 9.2 Example Data

```
F11BAF Example Program Data
  8                       N
  'BICGSTAB'              METHOD
  'P'  '1'  'N'  1        PRECON, SIGCMP, NORM, WEIGHT, ITERM
  1  1.0e-8   20          M, TOL, MAXITN
  1                       MONIT
  0  0.0                  LFILL, DTOL
  'N'  'C'                MILU, PSTRAT
 24                       NNZ
  2.   1    1
 -1.   1    4
  1.   1    8
  4.   2    1
 -3.   2    2
  2.   2    5
 -7.   3    3
  2.   3    6
  3.   4    1
 -4.   4    3
```

```
 5.   4    4
 5.   4    7
-1.   5    2
 8.   5    5
-3.   5    7
-6.   6    1
 5.   6    3
 2.   6    6
-5.   7    3
-1.   7    5
 6.   7    7
-1.   8    2
 2.   8    6
 3.   8    8          A(I), IROW(I), ICOL(I), I=1,...,NNZ
 6.   8.  -9.  46.
17.  21.  22.  34.    B(I), I=1,...,N
 0.   0.   0.   0.
 0.   0.   0.   0.    X(I), I=1,...,N
```

## 9.3 Example Results

```
F11BAF Example Program Results

Monitoring at iteration no.   1
residual norm:     1.4059E+02

Solution vector   Residual vector
     -4.5858E+00       1.5256E+01
      1.0154E+00       2.6624E+01
     -2.2234E+00      -8.7498E+00
      6.0097E+00       1.8602E+01
      1.3827E+00       8.2821E+00
     -7.9070E+00       2.0416E+01
      4.4270E-01       9.6094E+00
      5.9248E+00       3.3055E+01

Monitoring at iteration no.   2
residual norm:     3.2742E+01

   Solution vector   Residual vector
      4.1642E+00      -2.9585E+00
      4.9370E+00      -5.5523E+00
      4.8101E+00       8.2070E-01
      5.4324E+00      -1.6828E+01
      5.8531E+00       5.5975E-01
      1.1925E+01      -1.9150E+00
      8.4826E+00       1.0081E+00
      6.0625E+00      -3.1004E+00

Final Results
 Number of iterations for convergence:       3
Residual norm:                               1.0373E-08
Right-hand side of termination criterion:    5.5900E-06
1-norm of matrix A:                          1.1000E+01
```

```
Solution vector   Residual vector
   1.0000E+00      -1.3554E-09
   2.0000E+00      -2.6109E-09
   3.0000E+00       2.2471E-10
   4.0000E+00      -3.2203E-09
   5.0000E+00       6.3045E-10
   6.0000E+00      -5.2431E-10
   7.0000E+00       9.5771E-10
   8.0000E+00      -8.4890E-10
```

# F11BBF – 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.

## 1 Purpose

F11BBF is an iterative solver for a general (nonsymmetric) system of simultaneous linear equations; F11BBF is the second in a suite of three routines, where the first routine, F11BAF, must be called prior to F11BBF to set-up the suite, and the third routine in the suite, F11BCF, can be used to return additional information about the computation.

These three routines are suitable for the solution of large sparse general (nonsymmetric) systems of equations.

## 2 Specification

```
SUBROUTINE F11BBF(IREVCM, U, V, WORK, LWORK, IFAIL)
INTEGER           IREVCM, LWORK, IFAIL
real              U(*), V(*), WORK(*)
```

## 3 Description

F11BBF solves the general (nonsymmetric) system of linear simultaneous equations $Ax = b$ using one of three available methods: RGMRES, the preconditioned restarted generalized minimum residual method (Saad and Schultz [1]); CGS, the preconditioned conjugate gradient squared method (Sonneveld [2]); or Bi-CGSTAB ($\ell$), the bi-conjugate gradient stabilized method of order $\ell$ (Van der Vorst [3], Sleijpen and Fokkema [4]).

For a general description of the methods employed you are referred to Section 3 of the document for F11BAF.

F11BBF can solve the system after the first routine in the suite, F11BAF, has been called to initialize the computation and specify the method of solution. The third routine in the suite, F11BCF, can be used to return additional information generated by the computation during monitoring steps and after F11BBF has completed its tasks.

F11BBF uses **reverse communication**, i.e., it returns repeatedly to the calling program with the parameter IREVCM (see Section 5) set to specified values which require the calling program to carry out one of the following tasks:

compute the matrix-vector product $v = Au$ or $v = A^T u$ (the three methods require the matrix transpose-vector product only if $\|A\|_1$ or $\|A\|_\infty$ is estimated internally by Higham's method (Higham [5]));

solve the preconditioning equation $Mv = u$;

notify the completion of the computation;

allow the calling program to monitor the solution.

Through the parameter IREVCM the calling program can cause immediate or tidy termination of the execution. On final exit, the last iterates of the solution and of the residual vectors of the original system of equations are returned.

Reverse communication has the following advantages.

(1) Maximum flexibility in the representation and storage of sparse matrices: all matrix operations are performed outside the solver routine, thereby avoiding the need for a complicated interface with enough flexibility to cope with all types of storage schemes and sparsity patterns. This applies also to preconditioners.

(2) Enhanced user interaction: the progress of the solution can be closely monitored by the user and tidy or immediate termination can be requested. This is useful, for example, when alternative termination criteria are to be employed or in case of failure of the external routines used to perform matrix operations.

# 4 References

[1] Saad Y and Schultz M (1986) GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems *SIAM J. Sci. Statist. Comput.* **7** 856–869

[2] Sonneveld P (1989) CGS, a fast Lanczos-type solver for nonsymmetric linear systems *SIAM J. Sci. Statist. Comput.* **10** 36–52

[3] van der Vorst H (1989) Bi-CGSTAB, A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems *SIAM J. Sci. Statist. Comput.* **13** 631–644

[4] Sleijpen G L G and Fokkema D R (1993) BiCGSTAB($\ell$) for linear equations involving matrices with complex spectrum *ETNA* **1** 11–32

[5] Higham N J (1988) 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

# 5 Parameters

**Note.** This routine uses **reverse communication**. Its use involves an initial entry, intermediate exits and re-entries, and a final exit, as indicated by the **parameter IREVCM**. Between intermediate exits and re-entries **all parameters other than IREVCM and V must remain unchanged.**

1: IREVCM — INTEGER *Input/Output*

*On initial entry:* IREVCM = 0, otherwise an error condition will be raised.

*On intermediate re-entry:* IREVCM must either be unchanged from its previous exit value, or can have one of the following values.

5 Tidy termination: the computation will terminate at the end of the current iteration. Further reverse communication exits may occur depending on when the termination request is issued. F11BBF will then return with the termination code IREVCM = 4. Note that before calling F11BBF with IREVCM = 5 the calling program must have performed the tasks required by the value of IREVCM returned by the previous call to F11BBF, otherwise subsequently returned values may be invalid.

6 Immediate termination: F11BBF will return immediately with termination code IREVCM = 4 and with any useful information available. This includes the last iterate of the solution. The residual vector is generally not available. F11BBF will then return with the termination code IREVCM = 4.
Immediate termination may be useful, for example, when errors are detected during matrix-vector multiplication or during the solution of the preconditioning equation.

Changing IREVCM to any other value between calls will result in an error.

*On intermediate exit:* IREVCM has the following meanings.

$-1$ The calling program must compute the matrix-vector product $v = A^T u$, where $u$ and $v$ are stored in U and V, respectively; RGMRES, CGS and Bi-CGSTAB ($\ell$) methods return IREVCM = $-1$ only if the matrix norm $\|A\|_1$ or $\|A\|_\infty$ is estimated internally using Higham's method. This can only happen if ITERM = 1 in F11BAF.

1 The calling program must compute the matrix-vector product $v = Au$, where $u$ and $v$ are stored in U and V, respectively.

2 The calling program must solve the preconditioning equation $Mv = u$, where $u$ and $v$ are stored in U and V, respectively.

3 Monitoring step: the solution and residual at the current iteration are returned in the arrays U and V, respectively. No action by the calling program is required. F11BCF can be called at this step to return additional information.

*On final exit:* IREVCM = 4: F11BBF has completed its tasks. The value of IFAIL determines whether the iteration has been successfully completed, errors have been detected or the calling program has requested termination.

*Constraints:* on initial entry, IREVCM = 0; on re-entry, either IREVCM must remain unchanged or be reset to 5 or 6.

**2:** U(*) — ***real*** array *Input/Output*

**Note:** the dimension of the array U must be at least $n$.

*On initial entry:* an initial estimate, $x_0$, of the solution of the system of equations $Ax = b$.

*On intermediate re-entry:* U must remain unchanged.

*On intermediate exit:* the returned value of IREVCM determines the contents of U in the following way:

| | |
|---|---|
| IREVCM = −1, 1 or 2 | U holds the vector $u$ on which the operation specified by IREVCM is to be carried out; |
| IREVCM = 3 | U holds the current iterate of the solution vector. |

*On final exit:* if IFAIL = 3 or < 0, the array U is unchanged from the initial entry to F11BBF. If IFAIL = 1, the array U is unchanged from the last entry to F11BBF. Otherwise, U holds the last available iterate of the solution of the system of equations, for all returned values of IFAIL.

**3:** V(*) — ***real*** array *Input/Output*

**Note:** the dimension of the array V must be at least $n$.

*On initial entry:* the right-hand side $b$ of the system of equations $Ax = b$.

*On intermediate re-entry:* the returned value of IREVCM determines the contents of V in the following way:

| | |
|---|---|
| IREVCM = −1, 1 or 2 | V must store the vector $v$, the result of the operation specified by the value of IREVCM returned by the previous call to F11BBF; |
| IREVCM = 3 | V must remain unchanged. |

*On intermediate exit:* if IREVCM = 3, V holds the current iterate of the residual vector. Note that this is an approximation to the true residual vector. Otherwise, it does not contain any useful information.

*On final exit:* if IFAIL = 3 or < 0, the array V is unchanged from the initial entry to F11BBF. If IFAIL = 1, the array V is unchanged from the last entry to F11BBF. If IFAIL= 0 or 2, the array V contains the true residual vector of the system of equations (see also Section 6); otherwise, V stores the last available iterate of the residual vector unless IFAIL = 8 is returned on last entry, in which case V is set to 0.0.

**4:** WORK (*) — ***real*** array *Input/Output*

*On initial entry:* if user-supplied weights are used in the computation of the vector norms in the termination criterion (see Sections 3 and 5 of the document for F11BAF), these must be stored in WORK(1:$n$); otherwise, WORK need not be initialized.

*On intermediate re-entry:* WORK must remain unchanged.

*On final exit:* if weights are used, WORK(1:$n$) remains unchanged from the values supplied on initial entry.

**5:** LWORK — INTEGER *Input*

*On initial entry:* the dimension of the array WORK as declared in the (sub)program from which F11BBF was called. The required amount of workspace is as follows:

| **Method** | **Requirements** |
|---|---|
| RGMRES | LWORK $= 3n + m(m + n + 4) + 1 + p$, where $m$ is the dimension of the basis |
| CGS | LWORK $= 7n + p$ |
| Bi-CGSTAB ($\ell$) | LWORK $= 2n(\ell + 2) + \ell(\ell + 2) + p + q$, where $\ell$ is the order of the method; |

where

| | |
|---|---|
| $p = n$ | if user-defined weights are used in the computation of vector norms for the termination criterion (see Sections 3 and 5 of the document for F11BAF); |
| 0 | otherwise. |
| $k = 2n$ | if $\ell > 1$ and ITERM = 2 was supplied to F11BAF (see Sections 3 and 5 of the document for F11BAF); |
| $n$ | if $\ell > 1$ and a preconditioner is used or ITERM = 2 was supplied to F11BAF (see Sections 3 and 5 of the document for F11BAF). |
| 0 | otherwise. |

*Constraint:* LWORK $\geq$ LWREQ, where LWREQ is returned by F11BAF.

**6:** IFAIL — INTEGER *Input*

*On entry:* IFAIL must be set to 0, $-1$ or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

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

# 6 Errors and Warnings

If on entry IFAIL = 0 or $-1$, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL $= -i$

On entry, the $i$th argument had an illegal value.

IFAIL = 1

F11BBF has been called again after returning the termination code IREVCM = 4. No further computation has been carried out and all input data and data stored for access by F11BCF have remained unchanged.

IFAIL = 2

The required accuracy could not be obtained. However, F11BBF has terminated with reasonable accuracy: the last iterate of the residual satisfied the termination criterion but the exact residual $r = b - Ax$, did not. After the first occurrence of this situation, the iteration was restarted once, but F11BBF could not improve on the accuracy. This error code usually implies that your problem has been fully and satisfactorily solved to within or close to the accuracy available on your system. Further iterations are unlikely to improve on this situation. You should call F11BCF to check the values of the left-and right-hand side of the termination condition.

IFAIL = 3

F11BAF was either not called before calling F11BBF or it returned an error. The arguments U and V remain unchanged.

IFAIL = 4

The calling program requested a tidy termination before the solution had converged. The arrays U and V return the last iterates available of the solution and of the residual vector, respectively.

IFAIL = 5

The solution did not converge within the maximum number of iterations allowed. The arrays U and V return the last iterates available of the solution and of the residual vector, respectively.

IFAIL = 6

Algorithmic breakdown. The last available iterates of the solution and residuals are returned, although it is possible that they are completely inaccurate.

IFAIL = 8

The calling program requested an immediate termination. However, the array U returns the last iterate of the solution, the array V returns the last iterate of the residual vector, for the CGS method only.

# 7 Accuracy

On completion, i.e., IREVCM = 4 on exit, the arrays U and V will return the solution and residual vectors, $_k$ and $r_k = b - Ax_k$, respectively, at the $k$th iteration, the last iteration performed, unless an immediate termination was requested.

On successful completion, the termination criterion is satisfied to within the user-specified tolerance, as described in Section 3 of the document for F11BAF. The computed values of the left- and right-hand sides of the termination criterion selected can be obtained by a call to F11BCF.

# 8 Further Comments

The number of operations carried out by F11BBF for each iteration is likely to be principally determined by the computation of the matrix-vector products $v = Au$ and by the solution of the preconditioning equation $Mv = u$ in the calling program. Each of these operations is carried out once every iteration.

The number of the remaining operations in F11BBF for each iteration is approximately proportional to $n$.

The number of iterations required to achieve a prescribed accuracy cannot be easily determined at the onset, as it can depend dramatically on the conditioning and spectrum of the preconditioned matrix of the coefficients $\bar{A} = M^{-1}A$ (RGMRES and CGS methods) or $\bar{A} = AM^{-1}$ (Bi-CGSTAB ($\ell$) method).

Additional matrix-vector products are required for the computation of $||A||_1$ or $||A||_\infty$, when this has not been supplied to F11BAF and is required by the termination criterion employed.

If the termination criterion $||r_k||_p \leq \tau(||b||_p + ||A||_p ||x_k||_p)$ is used (see Section 3 of the document for F11BAF) and $||x_0|| \gg ||x_k||$, then the required accuracy cannot be obtained due to loss of significant digits. The iteration is restarted automatically at some suitable point: F11BBF sets $x_0 = x_k$ and the computation begins again. For particularly badly scaled problems, more than one restart may be necessary. This does not apply to the RGMRES method which, by its own nature, self-restarts every super-iteration. Naturally, restarting adds to computational costs: it is recommended that the iteration should start from a value $x_0$ which is as close to the true solution $\tilde{x}$ as can be estimated. Otherwise, the iteration should start from $x_0 = 0$.

# 9 Example

See the example for F11BAF.

# F11BCF – 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.

## 1 Purpose

F11BCF is the third in a suite of three routines for the iterative solution of a general (nonsymmetric) system of simultaneous linear equations (Golub and Van Loan [1]). F11BCF returns information about the computations during an iteration and/or after this has been completed. The first routine of the suite, F11BAF, is a set-up routine; the second routine, F11BBF is the iterative solver itself.

These three routines are suitable for the solution of large sparse general (nonsymmetric) systems of equations.

## 2 Specification

```
SUBROUTINE F11BCF(ITN, STPLHS, STPRHS, ANORM, SIGMAX, IFAIL)
INTEGER           ITN, IFAIL
real              STPLHS, STPRHS, ANORM, SIGMAX
```

## 3 Description

F11BCF returns information about the solution process. It can be called either during a monitoring step of F11BBF or after F11BBF has completed its tasks. Calling F11BCF at any other time will result in an error condition being raised.

For further information you should read the documentation for F11BAF and F11BBF.

## 4 References

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

## 5 Parameters

**1:** ITN — INTEGER *Output*

*On exit:* the number of iterations carried out by F11BBF.

**2:** STPLHS — ***real*** *Output*

*On exit:* the current value of the left-hand side of the termination criterion used by F11BBF.

**3:** STPRHS — ***real*** *Output*

*On exit:* the current value of the right-hand side of the termination criterion used by F11BBF.

**4:** ANORM — ***real*** *Output*

*On exit:* the norm $\|A\|_1$ or $\|A\|_\infty$ when it is used by the termination criterion in F11BBF, either when it has been supplied to F11BAF or it has been estimated by F11BBF (see also Section 3 of the document for F11BAF and Section 5 of the document for F11BAF); otherwise, ANORM = 0.0 is returned.

**5:** SIGMAX — ***real*** *Output*

*On exit:* the current estimate of the largest singular value $\sigma_1(\bar{A})$ of the preconditioned iteration matrix when it is used by the termination criterion in F11BBF, either when it has been supplied to F11BAF or it has been estimated by F11BBF (see also Sections 3 and 5 of the document for F11BAF); otherwise, SIGMAX = 0.0 is returned.

**6:** IFAIL — INTEGER *Input/Output*

*On entry:* IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

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

# 6 Errors and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL = 1

F11BCF has been called out of sequence. For example, the last call to F11BBF did not return IREVCM = 3 or 4.

# 7 Accuracy

Not applicable.

# 8 Further Comments

None.

# 9 Example

See the example for F11BAF.

# F11DAF – 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.

## 1 Purpose

F11DAF computes an incomplete $LU$ factorization of a real sparse nonsymmetric matrix, represented in coordinate storage format. This factorization may be used as a preconditioner in combination with F11DCF or F11BBF.

## 2 Specification

```
 SUBROUTINE F11DAF(N, NNZ, A, LA, IROW, ICOL, LFILL, DTOL, PSTRAT,
1                  MILU, IPIVP, IPIVQ, ISTR, IDIAG, NNZC, NPIVM,
2                  IWORK, LIWORK, IFAIL)
 INTEGER           N, NNZ, LA, IROW(LA), ICOL(LA), LFILL, IPIVP(N),
1                  IPIVQ(N), ISTR(N+1), IDIAG(N), NNZC, NPIVM,
2                  IWORK(LIWORK), LIWORK, IFAIL
 real              A(LA), DTOL
 CHARACTER*1       PSTRAT, MILU
```

## 3 Description

This routine computes an incomplete $LU$ factorization [1], [2] of a real sparse nonsymmetric $n$ by $n$ matrix $A$. The factorization is intended primarily for use as a preconditioner with one of the iterative solvers F11DCF or F11BBF.

The decomposition is written in the form

$$A = M + R$$

where

$$M = PLDUQ$$

and $L$ is lower triangular with unit diagonal elements, $D$ is diagonal, $U$ is upper triangular with unit diagonals, $P$ and $Q$ are permutation matrices, and $R$ is a remainder matrix.

The amount of fill-in occurring in the factorization can vary from zero to complete fill, and can be controlled by specifying either the maximum level of fill LFILL, or the drop tolerance DTOL.

The argument PSTRAT defines the pivoting strategy to be used. The options currently available are no pivoting, user-defined pivoting, partial pivoting by columns for stability, and complete pivoting by rows for sparsity and by columns for stability. The factorization may optionally be modified to preserve the row-sums of the original matrix.

The sparse matrix $A$ is represented in coordinate storage (CS) format (see Section 2.1.1 of the Chapter Introduction). The array A stores all the non-zero elements of the matrix $A$, while arrays IROW and ICOL store the corresponding row and column indices respectively. Multiple non-zero elements may not be specified for the same row and column index.

The preconditioning matrix $M$ is returned in terms of the CS representation of the matrix

$$C = L + D^{-1} + U - 2I.$$

Further algorithmic details are given in Section 8.3

## 4 References

[1] Meijerink J and van der Vorst H (1977) An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix *Math. Comput.* **31** 148–162

[2] Meijerink J and van der Vorst H (1981) Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems *J. Comput. Phys.* **44** 134–155

[3] Salvini S A and Shaw G J (1996) An evaluation of new NAG Library solvers for large sparse unsymmetric linear systems *NAG Technical Report TR2/96*

# 5 Parameters

**1:** N — INTEGER *Input*

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

*Constraint:* N $\geq 1$.

**2:** NNZ — INTEGER *Input*

*On entry:* the number of non-zero elements in the matrix $A$.

*Constraint:* $1 \leq$ NNZ $\leq$ N$^2$.

**3:** A(LA) — ***real*** array *Input/Output*

*On entry:* the non-zero elements in the matrix $A$, ordered by increasing row index, and by increasing column index within each row. Multiple entries for the same row and column indices are not permitted. The routine F11ZAF may be used to order the elements in this way.

*On exit:* the first NNZ entries of A contain the non-zero elements of $A$ and the next NNZC entries contain the elements of the matrix $C$. Matrix elements are ordered by increasing row index, and by increasing column index within each row.

**4:** LA — INTEGER *Input*

*On entry:* the dimension of the arrays A, IROW and ICOL as declared in the (sub)program from which F11DAF is called. These arrays must be of sufficient size to store both $A$ (NNZ elements) and $C$ (NNZC elements).

*Constraint:* LA $\geq 2 \times$ NNZ.

**5:** IROW(LA) — INTEGER array *Input/Output*

**6:** ICOL(LA) — INTEGER array *Input/Output*

*On entry:* the row and column indices of the non-zero elements supplied in A.

*Constraint:* IROW and ICOL must satisfy the following constraints (which may be imposed by a call to F11ZAF):

$1 \leq$ IROW($i$) $\leq$ N and $1 \leq$ ICOL($i$) $\leq$ N, for $i = 1, 2, \ldots,$NNZ.
IROW($i-1$) $<$ IROW($i$), or
IROW($i-1$) = IROW($i$) and ICOL($i-1$) $<$ ICOL($i$), for $i = 2, 3, \ldots,$NNZ.

*On exit:* the row and column indices of the non-zero elements returned in A.

**7:** LFILL — INTEGER *Input*

*On entry:* If LFILL $\geq 0$ its value is the maximum level of fill allowed in the decomposition (see Section 8.2). A negative value of LFILL indicates that DTOL will be used to control the fill instead.

**8:** DTOL — ***real*** *Input*

*On entry:* if LFILL $< 0$ then DTOL is used as a drop tolerance to control the fill-in (see Section 8.2); otherwise DTOL is not referenced.

*Constraint:* DTOL $\geq 0.0$ if LFILL $< 0$.

**9:** PSTRAT — CHARACTER*1 *Input*

*On entry:* specifies the pivoting strategy to be adopted as follows:

if PSTRAT = 'N', then no pivoting is carried out;
if PSTRAT = 'U', then pivoting is carried out according to the user-defined input values of IPIVP and IPIVQ;
if PSTRAT = 'P', then partial pivoting by columns for stability is carried out.
if PSTRAT = 'C', then complete pivoting by rows for sparsity, and by columns for stability, is carried out.

*Suggested value:* PSTRAT = 'C'.

*Constraint:* PSTRAT = 'N', 'U', 'P', or 'C'.

**10:** MILU — CHARACTER*1 *Input*

*On entry:* indicates whether or not the factorization should be modified to preserve row-sums (see Section 8.4):

if MILU = 'M', the factorization is modified (MILU);
if MILU = 'N', the factorization is not modified.

*Constraint:* MILU = 'M' or 'N'.

**11:** IPIVP(N) — INTEGER array *Input/Output*

**12:** IPIVQ(N) — INTEGER array *Input/Output*

*On entry:* if PSTRAT = 'U', then IPIVP($k$) and IPIVQ($k$) must specify the row and column indices of the element used as a pivot at elimination stage $k$. Otherwise IPIVP and IPIVQ need not be initialized.

*Constraint:* if PSTRAT = 'U', then IPIVP and IPIVQ must both hold valid permutations of the integers on [1,N].

*On exit:* the pivot indices. If IPIVP($k$) = $i$ and IPIVQ($k$) = $j$ then the element in row $i$ and column $j$ was used as the pivot at elimination stage $k$.

**13:** ISTR(N+1) — INTEGER array *Output*

*On exit:* ISTR($i$), for $i = 1, 2, \ldots$,N holds the index of arrays A, IROW and ICOL where row $i$ of the matrix $C$ starts. ISTR(N+1) holds the address of the last non-zero element in $C$ plus one.

**14:** IDIAG(N) — INTEGER array *Output*

*On exit:* IDIAG($i$), for $i = 1, 2, \ldots$,N holds the index of arrays A, IROW and ICOL which holds the diagonal element in row $i$ of the matrix $C$.

**15:** NNZC — INTEGER *Output*

*On exit:* the number of non-zero elements in the matrix $C$.

**16:** NPIVM — INTEGER *Output*

*On exit:* if NPIVM > 0 it gives the number of pivots which were modified during the factorization to ensure that $M$ exists. If NPIVM = $-1$ no pivot modifications were required, but a local restart occurred (see Section 8.3) .The quality of the preconditioner will generally depend on the returned value of NPIVM. If NPIVM is large the preconditioner may not be satisfactory. In this case it may be advantageous to call F11DAF again with an increased value of LFILL, a reduced value of DTOL, or PSTRAT set to 'C'. See also Section 8.5.

**17:** IWORK(LIWORK) — INTEGER array *Workspace*

**18:** LIWORK — INTEGER *Input*

*On entry:* the dimension of the array IWORK as declared in the (sub)program from which F11DAF is called.

*Constraint:* LIWORK $\geq 7 \times$ N + 2.

**19:** IFAIL — INTEGER *Input/Output*

*On entry:* IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

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

# 6 Errors and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL= 1

On entry, N < 1,
or NNZ < 1,
or NNZ > $N^2$,
or LA < 2 × NNZ,
or LFILL < 0 and DTOL < 0.0,
or PSTRAT ≠ 'N', 'U', 'P', or 'C',
or MILU ≠ 'M' or 'N',
or LIWORK < 7 × N + 2.

IFAIL= 2

On entry, the arrays IROW and ICOL fail to satisfy the following constraints:

$1 \leq \text{IROW}(i) \leq \text{N}$ and $1 \leq \text{ICOL}(i) \leq \text{N}$, for $i = 1, 2, \ldots,$NNZ.
$\text{IROW}(i-1) < \text{IROW}(i)$, or
$\text{IROW}(i-1) = \text{IROW}(i)$ and $\text{ICOL}(i-1) < \text{ICOL}(i)$, for $i = 2, 3, \ldots,$NNZ.

Therefore a non-zero element has been supplied which does not lie within the matrix $A$, is out of order, or has duplicate row and column indices. Call F11ZAF to reorder and sum or remove duplicates.

IFAIL= 3

On entry, PSTRAT = 'U', but one or both of IPIVP and IPIVQ does not represent a valid permutation of the integers in [1,N]. An input value of IPIVP or IPIVQ is either out of range or repeated.

IFAIL = 4

LA is too small, resulting in insufficient storage space for fill-in elements. The decomposition has been terminated before completion. Either increase LA or reduce the amount of fill by reducing LFILL, or increasing DTOL.

IFAIL= 5

A serious error has occurred in an internal call to F11ZAF. Check all subroutine calls and array sizes. Seek expert help.

# 7 Accuracy

The accuracy of the factorization will be determined by the size of the elements that are dropped and the size of any modifications made to the pivot elements. If these sizes are small then the computed factors will correspond to a matrix close to $A$. The factorization can generally be made more accurate by increasing LFILL, or by reducing DTOL with LFILL < 0.

If F11DAF is used in combination with F11BBF, or F11DCF, the more accurate the factorization the fewer iterations will be required. However, the cost of the decomposition will also generally increase.

# 8 Further Comments

## 8.1 Timing

The time taken for a call to F11DAF is roughly proportional to $\mathrm{NNZC}^2/\mathrm{N}$.

## 8.2 Control of Fill-in

If LFILL $\geq 0$ the amount of fill-in occurring in the incomplete factorization is controlled by limiting the maximum **level** of fill-in to LFILL. The original non-zero elements of $A$ are defined to be of level 0. The fill level of a new non-zero location occurring during the factorization is defined as:

$$k = \max(k_e, k_c) + 1,$$

where $k_e$ is the level of fill of the element being eliminated, and $k_c$ is the level of fill of the element causing the fill-in.

If LFILL $< 0$ the fill-in is controlled by means of the **drop tolerance** DTOL. A potential fill-in element $a_{ij}$ occurring in row $i$ and column $j$ will not be included if:

$$|a_{ij}| < \mathrm{DTOL} \times \alpha,$$

where $\alpha$ is the maximum absolute value element in the matrix $A$.

For either method of control, any elements which are not included are discarded unless MILU = 'M', in which case their contributions are subtracted from the pivot element in the relevant elimination row, to preserve the row-sums of the original matrix.

Should the factorization process break down a local restart process is implemented as described in Section 8.3. This will affect the amount of fill present in the final factorization.

## 8.3 Algorithmic Details

The factorization is constructed row by row. At each elimination stage a row index is chosen. In the case of complete pivoting this index is chosen in order to reduce fill-in. Otherwise the rows are treated in the order given, or some user-defined order.

The chosen row is copied from the original matrix $A$ and modified according to those previous elimination stages which affect it. During this process any fill-in elements are either dropped or kept according to the values of LFILL or DTOL. In the case of a modified factorization (MILU = 'M') the sum of the dropped terms for the given row is stored.

Finally the pivot element for the row is chosen and the multipliers are computed for this elimination stage. For partial or complete pivoting the pivot element is chosen in the interests of stability as the element of largest absolute value in the row. Otherwise the pivot element is chosen in the order given, or some user-defined order.

If the factorization breaks down because the chosen pivot element is zero, or there is no non-zero pivot available, a local restart recovery process is implemented. The modification of the given pivot row according to previous elimination stages is repeated, but this time keeping all fill. Note that in this case the final factorization will include more fill than originally specified by the user-supplied value of LFILL or DTOL. The local restart usually results in a suitable non-zero pivot arising. The original criteria for dropping fill-in elements is then resumed for the next elimination stage (hence the **local** nature of the restart process). Should this restart process also fail to produce a non-zero pivot element an arbitrary unit pivot is introduced in an arbitrarily chosen column. F11DAF returns an integer parameter NPIVM which gives the number of these arbitrary unit pivots introduced. If no pivots were modified but local restarts occurred NPIVM $= -1$ is returned.

## 8.4 Choice of Parameters

There is unfortunately no choice of the various algorithmic parameters which is optimal for all types of matrix, and some experimentation will generally be required for each new type of matrix encountered.

If the matrix $A$ is not known to have any particular special properties the following strategy is recommended. Start with LFILL = 0 and PSTRAT = 'C'. If the value returned for NPIVM is significantly

larger than zero, i.e., a large number of pivot modifications were required to ensure that $M$ existed, the preconditioner is not likely to be satisfactory. In this case increase LFILL until NPIVM falls to a value close to zero.

If $A$ has non-positive off-diagonal elements, is non-singular, and has only non-negative elements in its inverse, it is called an 'M-matrix'. It can be shown that no pivot modifications are required in the incomplete $LU$ factorization of an M-matrix [1]. In this case a good preconditioner can generally be expected by setting LFILL = 0, PSTRAT = 'N' and MILU = 'M'.

Some illustrations of the application of F11DAF to linear systems arising from the discretization of two-dimensional elliptic partial differential equations, and to random-valued randomly structured linear systems, can be found in [3].

### 8.5 Direct Solution of Sparse Linear Systems

Although it is not their primary purpose F11DAF and F11DBF may be used together to obtain a **direct** solution to a non-singular sparse linear system. To achieve this the call to F11DBF should be preceeded by a **complete** $LU$ factorization

$$A = PLDUQ = M.$$

A complete factorization is obtained from a call to F11DAF with LFILL $< 0$ and DTOL = 0.0, provided NPIVM $\leq 0$ on exit. A positive value of NPIVM indicates that $A$ is singular, or ill-conditioned. A factorization with positive NPIVM may serve as a preconditioner, but will not result in a direct solution. It is therefore **essential** to check the output value of NPIVM if a direct solution is required.

The use of F11DAF and F11DBF as a direct method is illustrated in Section 9 of the document for F11DBF.

## 9 Example

This example program reads in a sparse matrix $A$ and calls F11DAF to compute an incomplete $LU$ factorization. It then outputs the non-zero elements of both $A$ and $C = L + D^{-1} + U - 2I$.

The call to F11DAF has LFILL = 0, and PSTRAT = 'C', giving an unmodified zero-fill $LU$ factorization, with row pivoting for sparsity and column pivoting for stability.

### 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.

```
*     F11DAF Example Program Text
*     Mark 18 Release. NAG Copyright 1997.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LA, LIWORK
      PARAMETER        (NMAX=1000,LA=10000,LIWORK=7*NMAX+2)
*     .. Local Scalars ..
      real             DTOL
      INTEGER          I, IFAIL, LFILL, N, NNZ, NNZC, NPIVM
      CHARACTER        MILU, PSTRAT
*     .. Local Arrays ..
      real             A(LA)
      INTEGER          ICOL(LA), IDIAG(NMAX), IPIVP(NMAX), IPIVQ(NMAX),
     +                 IROW(LA), ISTR(NMAX+1), IWORK(LIWORK)
*     .. External Subroutines ..
      EXTERNAL         F11DAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11DAF Example Program Results'
```

```
      WRITE (NOUT,*)
*
*     Skip heading in data file
*
      READ (NIN,*)
*
*     Read algorithmic parameters
*
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
         READ (NIN,*) NNZ
         READ (NIN,*) LFILL, DTOL
         READ (NIN,*) PSTRAT
         READ (NIN,*) MILU
*
*        Read the matrix A
*
         DO 20 I = 1, NNZ
            READ (NIN,*) A(I), IROW(I), ICOL(I)
   20    CONTINUE
*
*        Calculate incomplete LU factorization
*
         IFAIL = 0
         CALL F11DAF(N,NNZ,A,LA,IROW,ICOL,LFILL,DTOL,PSTRAT,MILU,IPIVP,
     +               IPIVQ,ISTR,IDIAG,NNZC,NPIVM,IWORK,LIWORK,IFAIL)
*
*        Output original matrix
*
         WRITE (NOUT,*) ' Original Matrix'
         WRITE (NOUT,'(A,I4)') ' N     =', N
         WRITE (NOUT,'(A,I4)') ' NNZ   =', NNZ
         DO 40 I = 1, NNZ
            WRITE (NOUT,'(I8,e16.4,2I8)') I, A(I), IROW(I), ICOL(I)
   40    CONTINUE
         WRITE (NOUT,*)
*
*        Output details of the factorization
*
         WRITE (NOUT,*) ' Factorization'
         WRITE (NOUT,'(A,I4)') ' N     =', N
         WRITE (NOUT,'(A,I4)') ' NNZ   =', NNZC
         WRITE (NOUT,'(A,I4)') ' NPIVM =', NPIVM
         DO 60 I = NNZ + 1, NNZ + NNZC
            WRITE (NOUT,'(I8,e16.4,2I8)') I, A(I), IROW(I), ICOL(I)
   60    CONTINUE
         WRITE (NOUT,*)
*
         WRITE (NOUT,*) '        I       IPIVP(I)  IPIVQ(I)'
         DO 80 I = 1, N
            WRITE (NOUT,'(3I10)') I, IPIVP(I), IPIVQ(I)
   80    CONTINUE
*
      END IF
      STOP
      END
```

## 9.2 Program Data

```
F11DAF Example Program Data
  4                    N
  11                   NNZ
  0 0.0                LFILL, DTOL
  'C'                  PSTRAT
  'N'                  MILU
  1.   1    2
  1.   1    3
 -1.   2    1
  2.   2    3
  2.   2    4
  3.   3    1
 -2.   3    4
  1.   4    1
 -2.   4    2
  1.   4    3
  1.   4    4         A(I), IROW(I), ICOL(I), I=1,...,NNZ
```

## 9.3 Program Results

```
F11DAF Example Program Results

 Original Matrix
N     =   4
NNZ   =  11
      1      0.1000E+01       1       2
      2      0.1000E+01       1       3
      3     -0.1000E+01       2       1
      4      0.2000E+01       2       3
      5      0.2000E+01       2       4
      6      0.3000E+01       3       1
      7     -0.2000E+01       3       4
      8      0.1000E+01       4       1
      9     -0.2000E+01       4       2
     10      0.1000E+01       4       3
     11      0.1000E+01       4       4

 Factorization
N     =   4
NNZ   =  11
NPIVM =   0
     12      0.1000E+01       1       1
     13      0.1000E+01       1       3
     14      0.3333E+00       2       2
     15     -0.6667E+00       2       4
     16     -0.3333E+00       3       2
     17      0.5000E+00       3       3
     18      0.6667E+00       3       4
     19     -0.2000E+01       4       1
     20      0.3333E+00       4       2
     21      0.1500E+01       4       3
     22     -0.3000E+01       4       4
```

| I | IPIVP(I) | IPIVQ(I) |
|---|---|---|
| 1 | 1 | 2 |
| 2 | 3 | 1 |
| 3 | 2 | 3 |
| 4 | 4 | 4 |

# F11DBF – 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.

## 1 Purpose

F11DBF solves a system of linear equations involving the incomplete $LU$ preconditioning matrix generated by F11DAF.

## 2 Specification

```
SUBROUTINE F11DBF(TRANS, N, A, LA, IROW, ICOL, IPIVP, IPIVQ, ISTR,
1                 IDIAG, CHECK, Y, X, IFAIL)
INTEGER           N, LA, IROW(LA), ICOL(LA), IPIVP(N), IPIVQ(N),
1                 ISTR(N+1), IDIAG(N), IFAIL
real              A(LA), Y(N), X(N)
CHARACTER*1       TRANS, CHECK
```

## 3 Description

This routine solves a system of linear equations

$$Mx = y, \quad \text{or} \quad M^T x = y,$$

according to the value of the argument TRANS, where the matrix $M = PLDUQ$, corresponds to an incomplete $LU$ decomposition of a sparse matrix stored in coordinate storage (CS) format (see Section 2.1.1 of the Chapter Introduction), as generated by F11DAF.

In the above decomposition $L$ is a lower triangular sparse matrix with unit diagonal elements, $D$ is a diagonal matrix, $U$ is an upper triangular sparse matrix with unit diagonal elements and $P$ and $Q$ are permutation matrices. $L$, $D$ and $U$ are supplied to F11DBF through the matrix

$$C = L + D^{-1} + U - 2I$$

which is an N by N sparse matrix, stored in CS format, as returned by F11DAF. The permutation matrices $P$ and $Q$ are returned from F11DAF via the arrays IPIVP and IPIVQ.

It is envisaged that a common use of F11DBF will be to carry out the preconditioning step required in the application of F11BBF to sparse linear systems. F11DBF is used for this purpose by the black-box routine F11DCF.

F11DBF may also be used in combination with F11DAF to solve a sparse system of linear equations directly (see Section 8.5 of the document for F11DAF). This use of F11DBF is demonstrated in Section 9.

## 4 References

None.

## 5 Parameters

1: TRANS — CHARACTER*1 *Input*

*On entry:* specifies whether or not the matrix $M$ is transposed:

if TRANS = 'N', then $Mx = y$ is solved;
if TRANS = 'T', then $M^T x = y$ is solved.

*Constraint:* TRANS = 'N' or 'T'.

**2:** N — INTEGER *Input*

*On entry:* $n$, the order of the matrix $M$. This **must** be the same value as was supplied in the preceeding call to F11DAF.

*Constraint:* N $\geq$ 1.

**3:** A(LA) — ***real*** array *Input*

*On entry:* the values returned in the array A by a previous call to F11DAF.

**4:** LA — INTEGER *Input*

*On entry:* the dimension of the arrays A, IROW and ICOL as declared in the (sub)program from which F11DBF is called. This **must** be the same value as was supplied in the preceeding call to F11DAF.

**5:** IROW(LA) — INTEGER array *Input*

**6:** ICOL(LA) — INTEGER array *Input*

**7:** IPIVP(N) — INTEGER array *Input*

**8:** IPIVQ(N) — INTEGER array *Input*

**9:** ISTR(N+1) — INTEGER array *Input*

**10:** IDIAG(N) — INTEGER array *Input*

*On entry:* the values returned in arrays IROW, ICOL, IPIVP, IPIVQ, ISTR and IDIAG by a previous call to F11DAF.

**11:** CHECK — CHARACTER*1 *Input*

*On entry:* specifies whether or not the CS representation of the matrix $M$ should be checked:

if CHECK = 'C', checks are carried on the values of N, IROW, ICOL, IPIVP, IPIVQ, ISTR and IDIAG;
if CHECK = 'N', none of these checks are carried out.

See also Section 8.2.

*Constraint:* CHECK = 'C' or 'N'.

**12:** Y(N) — ***real*** array *Input*

*On entry:* the right-hand side vector $y$.

**13:** X(N) — ***real*** array *Output*

*On exit:* the solution vector $x$.

**14:** IFAIL — INTEGER *Input/Output*

*On entry:* IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

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

# 6 Errors and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL = 1

On entry, TRANS ≠ 'N' or 'T',

or CHECK ≠ 'C' or 'N'.

IFAIL = 2

On entry, N < 1.

IFAIL = 3

On entry, the CS representation of the preconditioning matrix $M$ is invalid. Further details are given in the error message. Check that the call to F11DBF has been preceeded by a valid call to F11DAF and that the arrays A, IROW, ICOL, IPIVP, IPIVQ, ISTR and IDIAG have not been corrupted between the two calls.

# 7 Accuracy

If TRANS = 'N' the computed solution $x$ is the exact solution of a perturbed system of equations $(M + \delta M)x = y$, where

$$|\delta M| \leq c(n)\epsilon P|L||D||U|Q,$$

$c(n)$ is a modest linear function of $n$, and $\epsilon$ is the ***machine precision***. An equivalent result holds when TRANS = 'T'.

# 8 Further Comments

## 8.1 Timing

The time taken for a call to F11DBF is proportional to the value of NNZC returned from F11DAF.

## 8.2 Use of CHECK

It is expected that a common use of F11DBF will be to carry out the preconditioning step required in the application of F11BBF to sparse linear systems. In this situation F11DBF is likely to be called many times with the same matrix $M$. In the interests of both reliability and efficiency, you are recommended to set CHECK to 'C' for the first of such calls, and to 'N' for all subsequent calls.

# 9 Example

This example program reads in a sparse nonsymmetric matrix $A$ and a vector $y$. It then calls F11DAF, with LFILL = −1 and DTOL = 0.0, to compute the **complete** $LU$ decomposition

$$A = PLDUQ.$$

Finally it calls F11DBF to solve the system

$$PLDUQx = y.$$

## 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.

```
*     F11DBF Example Program Text
*     Mark 18 Release. NAG Copyright 1997.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LA, LIWORK
      PARAMETER        (NMAX=1000,LA=10000,LIWORK=7*NMAX+2)
*     .. Local Scalars ..
      real             DTOL
      INTEGER          I, IFAIL, LFILL, N, NNZ, NNZC, NPIVM
      CHARACTER        CHECK, MILU, PSTRAT, TRANS
*     .. Local Arrays ..
      real             A(LA), X(NMAX), Y(NMAX)
      INTEGER          ICOL(LA), IDIAG(NMAX), IPIVP(NMAX), IPIVQ(NMAX),
     +                 IROW(LA), ISTR(NMAX+1), IWORK(LIWORK)
*     .. External Subroutines ..
      EXTERNAL         F11DAF, F11DBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11DBF Example Program Results'
      WRITE (NOUT,*)
*
*     Skip heading in data file
*
      READ (NIN,*)
*
*     Read order of matrix and number of non-zero entries
*
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
         READ (NIN,*) NNZ
*
*        Read the matrix A
*
         DO 20 I = 1, NNZ
            READ (NIN,*) A(I), IROW(I), ICOL(I)
   20    CONTINUE
*
*        Read the vector y
*
         READ (NIN,*) (Y(I),I=1,N)
*
*        Calculate LU factorization
*
         LFILL = -1
         DTOL = 0.e0
         PSTRAT = 'C'
         MILU = 'N'
         IFAIL = 0
*
         CALL F11DAF(N,NNZ,A,LA,IROW,ICOL,LFILL,DTOL,PSTRAT,MILU,IPIVP,
     +               IPIVQ,ISTR,IDIAG,NNZC,NPIVM,IWORK,LIWORK,IFAIL)
*
*        Check value of NPIVM
*
```

```
         IF (NPIVM.GT.0) THEN
*
            WRITE (NOUT,*) 'Factorization is not complete'
*
         ELSE
*
*           Solve P L D U x = y
*
            TRANS = 'N'
            CHECK = 'C'
*
            CALL F11DBF(TRANS,N,A,LA,IROW,ICOL,IPIVP,IPIVQ,ISTR,IDIAG,
     +                  CHECK,Y,X,IFAIL)
*
*           Output results
*
            WRITE (NOUT,*) ' Solution of linear system'
            DO 40 I = 1, N
               WRITE (NOUT,'(e16.4)') X(I)
   40       CONTINUE
*
         END IF
*
      END IF
      STOP
      END
```

## 9.2 Program Data

```
F11DBF Example Program Data
  4                          N
 11                          NNZ
  1.   1    2
  1.   1    3
 -1.   2    1
  2.   2    3
  2.   2    4
  3.   3    1
 -2.   3    4
  1.   4    1
 -2.   4    2
  1.   4    3
  1.   4    4               A(I), IROW(I), ICOL(I), I=1,...,NNZ
  5.0  13.0 -5.0  4.0       Y(I), I=1,...,N
```

## 9.3 Program Results

```
 F11DBF Example Program Results

  Solution of linear system
      0.1000E+01
      0.2000E+01
      0.3000E+01
      0.4000E+01
```

# F11DCF – 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.

## 1 Purpose

F11DCF solves a real sparse nonsymmetric system of linear equations, represented in coordinate storage format, using a restarted generalized minimal residual (RGMRES), conjugate gradient squared (CGS), or stabilized bi-conjugate gradient (Bi-CGSTAB) method, with incomplete $LU$ preconditioning.

## 2 Specification

```
 SUBROUTINE F11DCF(METHOD, N, NNZ, A, LA, IROW, ICOL, IPIVP, IPIVQ,
1                  ISTR, IDIAG, B, M, TOL, MAXITN, X, RNORM, ITN,
2                  WORK, LWORK, IFAIL)
 INTEGER           N, NNZ, LA, IROW(LA), ICOL(LA), IPIVP(N),
1                  IPIVQ(N), ISTR(N+1), IDIAG(N), M, MAXITN, ITN,
2                  LWORK, IFAIL
 real              A(LA), B(N), TOL, X(N), RNORM, WORK(LWORK)
 CHARACTER*(*)     METHOD
```

## 3 Description

This routine solves a real sparse nonsymmetric linear system of equations:

$$Ax = b,$$

using a preconditioned RGMRES [1], CGS [4], or Bi-CGSTAB($\ell$) [5], [3] method.

F11DCF uses the incomplete $LU$ factorization determined by F11DAF as the preconditioning matrix. A call to F11DCF must always be preceeded by a call to F11DAF. Alternative preconditioners for the same storage scheme are available by calling F11DEF.

The matrix $A$, and the preconditioning matrix $M$, are represented in coordinate storage (CS) format (see Section 2.1.1 of the Chapter Introduction) in the arrays A, IROW and ICOL, as returned from F11DAF. The array A holds the non-zero entries in these matrices, while IROW and ICOL hold the corresponding row and column indices.

F11DCF is a black-box routine which calls F11BAF, F11BBF and F11BCF. If you wish to use an alternative storage scheme, preconditioner, or termination criterion, or require additional diagnostic information, you should call these underlying routines directly.

## 4 References

[1] Saad Y and Schultz M (1986) GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems *SIAM J. Sci. Statist. Comput.* **7** 856–869

[2] Salvini S A and Shaw G J (1996) An evaluation of new NAG Library solvers for large sparse unsymmetric linear systems *NAG Technical Report TR2/96*

[3] Sleijpen G L G and Fokkema D R (1993) BiCGSTAB($\ell$) for linear equations involving matrices with complex spectrum *ETNA* **1** 11–32

[4] Sonneveld P (1989) CGS, a fast Lanczos-type solver for nonsymmetric linear systems *SIAM J. Sci. Statist. Comput.* **10** 36–52

[5] van der Vorst H (1989) Bi-CGSTAB, A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems *SIAM J. Sci. Statist. Comput.* **13** 631–644

# 5 Parameters

**1:** METHOD — CHARACTER*(*) *Input*

*On entry:* specifies the iterative method to be used. The possible choices are:

| | |
|---|---|
| 'RGMRES' | restarted generalized minimum residual method; |
| 'CGS' | conjugate gradient squared method; |
| 'BICGSTAB' | bi-conjugate gradient stabilized ($\ell$) method; |

*Constraint:* METHOD = 'RGMRES', 'CGS' or 'BICGSTAB'.

**2:** N — INTEGER *Input*

*On entry:* $n$, the order of the matrix $A$. This **must** be the same value as was supplied in the preceeding call to F11DAF.

*Constraint:* $\mathrm{N} \geq 1$.

**3:** NNZ — INTEGER *Input*

*On entry:* the number of non-zero elements in the matrix $A$. This **must** be the same value as was supplied in the preceeding call to F11DAF.

*Constraint:* $1 \leq \mathrm{NNZ} \leq \mathrm{N}^2$.

**4:** A(LA) — ***real*** array *Input*

*On entry:* the values returned in array A by a previous call to F11DAF.

**5:** LA — INTEGER *Input*

*On entry:* the dimension of the arrays A, IROW and ICOL as declared in the (sub)program from which F11DCF is called. This **must** be the same value as was supplied in the preceeding call to F11DAF.

*Constraint:* $\mathrm{LA} \geq 2 \times \mathrm{NNZ}$.

**6:** IROW(LA) — INTEGER array *Input*

**7:** ICOL(LA) — INTEGER array *Input*

**8:** IPIVP(N) — INTEGER array *Input*

**9:** IPIVQ(N) — INTEGER array *Input*

**10:** ISTR(N+1) — INTEGER array *Input*

**11:** IDIAG(N) — INTEGER array *Input*

*On entry:* the values returned in arrays IROW, ICOL, IPIVP, IPIVQ, ISTR and IDIAG by a previous call to F11DAF.

**12:** B(N) — ***real*** array *Input*

*On entry:* the right-hand side vector $b$.

**13:** M — INTEGER *Input*

*On entry:* if METHOD = 'RGMRES', M is the dimension of the restart subspace.

If METHOD = 'BICGSTAB', M is the order $\ell$ of the polynomial Bi-CGSTAB method; otherwise, M is not referenced.

*Constraints:*

if METHOD = 'RGMRES', $0 < \mathrm{M} \leq \min(\mathrm{N},50)$;
if METHOD = 'BICGSTAB', $0 < \mathrm{M} \leq \min(\mathrm{N},10)$.

**14:** TOL — ***real*** *Input*

*On entry:* the required tolerance. Let $x_k$ denote the approximate solution at iteration $k$, and $r_k$ the corresponding residual. The algorithm is considered to have converged at iteration $k$ if:

$$\|r_k\|_\infty \leq \tau \times (\|b\|_\infty + \|A\|_\infty \|x_k\|_\infty).$$

If TOL $\leq 0.0$, $\tau = \max(\sqrt{\epsilon}, \sqrt{n}\,\epsilon)$ is used, where $\epsilon$ is the ***machine precision***. Otherwise $\tau = \max(\text{TOL}, 10\epsilon, \sqrt{n}\,\epsilon)$ is used.

*Constraint:* TOL $< 1.0$.

**15:** MAXITN — INTEGER *Input*

*On entry:* the maximum number of iterations allowed.

*Constraint:* MAXITN $\geq 1$.

**16:** X(N) — ***real*** array *Input/Output*

*On entry:* an initial approximation to the solution vector $x$.

*On exit:* an improved approximation to the solution vector $x$.

**17:** RNORM — ***real*** *Output*

*On exit:* the final value of the residual norm $\|r_k\|_\infty$,where $k$ is the output value of ITN.

**18:** ITN — INTEGER *Output*

*On exit:* the number of iterations carried out.

**19:** WORK(LWORK) — ***real*** array *Workspace*

**20:** LWORK — INTEGER *Input*

*On entry:* the dimension of the array WORK as declared in the (sub)program from which F11DCF is called.

*Constraints:*

if METHOD = 'RGMRES' then LWORK $\geq 4 \times$ N $+$M $\times$ (M+N+4) + 1;
if METHOD = 'CGS' then LWORK $\geq 8 \times$ N;
if METHOD = 'BICGSTAB' then LWORK $\geq 2 \times$N $\times$ (M+3) + M $\times$ (M+2).

**21:** IFAIL — INTEGER *Input/Output*

*On entry:* IFAIL must be set to 0, $-1$ or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

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

# 6 Errors and Warnings

If on entry IFAIL = 0 or $-1$, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL = 1

On entry, METHOD $\neq$ 'RGMRES', 'CGS', or 'BICGSTAB',
or N $< 1$,
or NNZ $< 1$,
or NNZ $>$ N$^2$,
or LA $< 2 \times$ NNZ,

or M < 1 and METHOD = 'RGMRES' or METHOD = 'BICGSTAB',

or M > min(N,50), with METHOD = 'RGMRES',

or M > min(N,10), with METHOD = 'BICGSTAB',

or TOL $\geq$ 1.0,

or MAXITN < 1,

or LWORK too small.

IFAIL = 2

On entry, the CS representation of $A$ is invalid. Further details are given in the error message. Check that the call to F11DCF has been preceeded by a valid call to F11DAF, and that the arrays A, IROW, and ICOL have not been corrupted between the two calls.

IFAIL = 3

On entry, the CS representation of the preconditioning matrix $M$ is invalid. Further details are given in the error message. Check that the call to F11DCF has been preceeded by a valid call to F11DAF, and that the arrays A, IROW, ICOL, IPIVP, IPIVQ, ISTR and IDIAG have not been corrupted between the two calls.

IFAIL = 4

The required accuracy could not be obtained. However, a reasonable accuracy may have been obtained, and further iterations could not improve the result. You should check the output value of RNORM for acceptability. This error code usually implies that your problem has been fully and satisfactorily solved to within or close to the accuracy available on your system. Further iterations are unlikely to improve on this situation.

IFAIL = 5

Required accuracy not obtained in MAXITN iterations.

IFAIL = 6

A serious error has occurred in an internal call to F11BAF, F11BBF or F11BCF. Check all subroutine calls and array sizes. Seek expert help.

# 7 Accuracy

On successful termination, the final residual $r_k = b - Ax_k$, where $k$ = ITN, satisfies the termination criterion

$$\|r_k\|_\infty \leq \tau \times (\|b\|_\infty + \|A\|_\infty \|x_k\|_\infty).$$

The value of the final residual norm is returned in RNORM.

# 8 Further Comments

The time taken by F11DCF for each iteration is roughly proportional to the value of NNZC returned from the preceeding call to F11DAF.

The number of iterations required to achieve a prescribed accuracy cannot be easily determined a priori, as it can depend dramatically on the conditioning and spectrum of the preconditioned coefficient matrix $\bar{A} = M^{-1}A$.

Some illustrations of the application of F11DCF to linear systems arising from the discretization of two-dimensional elliptic partial differential equations, and to random-valued randomly structured linear systems, can be found in [2].

# 9 Example

This example program solves a sparse linear system of equations using the CGS method, with incomplete $LU$ preconditioning.

## 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.

```
*     F11DCF Example Program Text
*     Mark 18 Release. NAG Copyright 1997.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LA, LIWORK, LWORK
      PARAMETER        (NMAX=1000,LA=10000,LIWORK=7*NMAX+2,LWORK=10000)
*     .. Local Scalars ..
      real             DTOL, RNORM, TOL
      INTEGER          I, IFAIL, ITN, LFILL, LWREQ, M, MAXITN, N, NNZ,
     +                 NNZC, NPIVM
      CHARACTER        MILU, PSTRAT
      CHARACTER*8      METHOD
*     .. Local Arrays ..
      real             A(LA), B(NMAX), WORK(LWORK), X(NMAX)
      INTEGER          ICOL(LA), IDIAG(NMAX), IPIVP(NMAX), IPIVQ(NMAX),
     +                 IROW(LA), ISTR(NMAX+1), IWORK(LIWORK)
*     .. External Subroutines ..
      EXTERNAL         F11DAF, F11DCF
*     .. Intrinsic Functions ..
      INTRINSIC        MAX
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11DCF Example Program Results'
      WRITE (NOUT,*)
*     Skip heading in data file
      READ (NIN,*)
*
*     Read algorithmic parameters
*
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
         READ (NIN,*) NNZ
         READ (NIN,*) METHOD
         READ (NIN,*) LFILL, DTOL
         READ (NIN,*) PSTRAT
         READ (NIN,*) MILU
         READ (NIN,*) M, TOL, MAXITN
*
*        Check size of workspace
*
         LWREQ = MAX(4*N+M*(M+N+4)+1,8*N,2*N*(M+3)+M*(M+2))
         IF (LWORK.LT.LWREQ) THEN
            WRITE (NOUT,'(A,I4)') 'LWORK must be at least', LWREQ
            STOP
         END IF
*
*        Read the matrix A
*
         DO 20 I = 1, NNZ
            READ (NIN,*) A(I), IROW(I), ICOL(I)
   20    CONTINUE
*
```

```
*        Read right-hand side vector b and initial approximate solution x
*
         READ (NIN,*) (B(I),I=1,N)
         READ (NIN,*) (X(I),I=1,N)
*
*        Calculate incomplete LU factorization
*
         IFAIL = 0
         CALL F11DAF(N,NNZ,A,LA,IROW,ICOL,LFILL,DTOL,PSTRAT,MILU,IPIVP,
     +               IPIVQ,ISTR,IDIAG,NNZC,NPIVM,IWORK,LIWORK,IFAIL)
*
*        Solve Ax = b using F11DCF
*
         CALL F11DCF(METHOD,N,NNZ,A,LA,IROW,ICOL,IPIVP,IPIVQ,ISTR,IDIAG,
     +               B,M,TOL,MAXITN,X,RNORM,ITN,WORK,LWORK,IFAIL)
*
         WRITE (NOUT,'(A,I10,A)') 'Converged in', ITN, ' iterations'
         WRITE (NOUT,'(A,1P,e16.3)') 'Final residual norm =', RNORM
         WRITE (NOUT,*)
*
*        Output x
*
         WRITE (NOUT,*) '           X'
         DO 40 I = 1, N
            WRITE (NOUT,'(1X,1P,e16.4)') X(I)
   40    CONTINUE
      END IF
*
      STOP
      END
```

## 9.2 Program Data

```
F11DCF Example Program Data
  8                        N
  24                       NNZ
  'CGS'                    METHOD
  0  0.0                   LFILL, DTOL
  'C'                      PSTRAT
  'N'                      MILU
  4  1.0e-10   100         M, TOL, MAXITN
  2.   1    1
 -1.   1    4
  1.   1    8
  4.   2    1
 -3.   2    2
  2.   2    5
 -7.   3    3
  2.   3    6
  3.   4    1
 -4.   4    3
  5.   4    4
  5.   4    7
 -1.   5    2
  8.   5    5
 -3.   5    7
 -6.   6    1
```

```
 5.   6    3
 2.   6    6
-5.   7    3
-1.   7    5
 6.   7    7
-1.   8    2
 2.   8    6
 3.   8    8          A(I), IROW(I), ICOL(I), I=1,...,NNZ
 6.   8.  -9.  46.
17.  21.  22.  34.    B(I), I=1,...,N
 0.   0.   0.   0.
 0.   0.   0.   0.    X(I), I=1,...,N
```

## 9.3 Program Results

```
 F11DCF Example Program Results

Converged in         4 iterations
Final residual norm =       8.527E-14

            X
       1.0000E+00
       2.0000E+00
       3.0000E+00
       4.0000E+00
       5.0000E+00
       6.0000E+00
       7.0000E+00
       8.0000E+00
```

# F11DDF – 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.

## 1 Purpose

F11DDF solves a system of linear equations involving the preconditioning matrix corresponding to SSOR applied to a real sparse nonsymmetric matrix, represented in coordinate storage format.

## 2 Specification

```
SUBROUTINE F11DDF(TRANS, N, NNZ, A, IROW, ICOL, RDIAG, OMEGA,
1                 CHECK, Y, X, IWORK, IFAIL)
INTEGER           N, NNZ, IROW(NNZ), ICOL(NNZ), IWORK(2*N+1), IFAIL
real              A(NNZ), RDIAG(N), OMEGA, Y(N), X(N)
CHARACTER*1       TRANS, CHECK
```

## 3 Description

This routine solves a system of linear equations

$$Mx = y, \quad \text{or} \quad M^T x = y,$$

according to the value of the argument TRANS, where the matrix

$$M = \frac{1}{\omega(2-\omega)}(D+\omega L)D^{-1}(D+\omega U)$$

corresponds to symmetric successive-over-relaxation (SSOR) [1] applied to a linear system $Ax = b$, where $A$ is a real sparse nonsymmetric matrix stored in coordinate storage (CS) format (see Section 2.1.1 of the Chapter Introduction).

In the definition of $M$ given above $D$ is the diagonal part of $A$, $L$ is the strictly lower triangular part of $A$, $U$ is the strictly upper triangular part of $A$, and $\omega$ is a user-defined relaxation parameter.

It is envisaged that a common use of F11DDF will be to carry out the preconditioning step required in the application of F11BBF to sparse linear systems. For an illustration of this use of F11DDF see the example program given in Section 9. F11DDF is also used for this purpose by the black-box routine F11DEF.

## 4 References

[1] Young D (1971) *Iterative Solution of Large Linear Systems* Academic Press, New York

## 5 Parameters

**1:** TRANS — CHARACTER*1 *Input*

*On entry:* specifies whether or not the matrix $M$ is transposed:

if TRANS = 'N', then $Mx = y$ is solved;
if TRANS = 'T', then $M^T x = y$ is solved.

*Constraint:* TRANS = 'N' or 'T'.

**2:** N — INTEGER *Input*

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

*Constraint:* N $\geq$ 1.

**3:** NNZ — INTEGER *Input*

*On entry:* the number of non-zero elements in the matrix $A$.

*Constraint:* $1 \leq \text{NNZ} \leq \text{N}^2$.

**4:** A(NNZ) — ***real*** array *Input*

*On entry:* the non-zero elements in the matrix $A$, ordered by increasing row index, and by increasing column index within each row. Multiple entries for the same row and column indices are not permitted. The routine F11ZAF may be used to order the elements in this way.

**5:** IROW(NNZ) — INTEGER array *Input*

**6:** ICOL(NNZ) — INTEGER array *Input*

*On entry:* the row and column indices of the non-zero elements supplied in A.

*Constraints:* IROW and ICOL must satisfy the following constraints (which may be imposed by a call to F11ZAF):

$1 \leq \text{IROW}(i) \leq \text{N}$, $1 \leq \text{ICOL}(i) \leq \text{N}$, for $i = 1, 2, \ldots, \text{NNZ}$.
$\text{IROW}(i-1) < \text{IROW}(i)$, or
$\text{IROW}(i-1) = \text{IROW}(i)$ and $\text{ICOL}(i-1) < \text{ICOL}(i)$, for $i = 2, 3, \ldots, \text{NNZ}$.

**7:** RDIAG(N) — ***real*** array *Input*

*On entry:* the elements of the diagonal matrix $D^{-1}$, where $D$ is the diagonal part of $A$.

**8:** OMEGA — ***real*** *Input*

*On entry:* the relaxation parameter $\omega$.

*Constraint:* $0.0 < \text{OMEGA} < 2.0$.

**9:** CHECK — CHARACTER*1 *Input*

*On entry:* specifies whether or not the CS representation of the matrix $M$ should be checked:

if CHECK = 'C', checks are carried on the values of N, NNZ, IROW, ICOL and OMEGA;
if CHECK = 'N', none of these checks are carried out.

See also Section 8.2.

*Constraint:* CHECK = 'C' or 'N'.

**10:** Y(N) — ***real*** array *Input*

*On entry:* the right-hand side vector $y$.

**11:** X(N) — ***real*** array *Output*

*On exit:* the solution vector $x$.

**12:** IWORK(2*N+1) — INTEGER array *Workspace*

**13:** IFAIL — INTEGER *Input/Output*

*On entry:* IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

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

# 6 Errors and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL = 1

On entry, TRANS ≠ 'N' or 'T',
or CHECK ≠ 'C' or 'N'.

IFAIL = 2

On entry, N < 1,
or NNZ < 1,
or NNZ > $N^2$,
or OMEGA lies outside the interval (0.0,2.0),

IFAIL = 3

On entry, the arrays IROW and ICOL fail to satisfy the following constraints:

$1 \leq \text{IROW}(i) \leq \text{N}$ and $1 \leq \text{ICOL}(i) \leq \text{N}$, for $i = 1, 2, \ldots,$NNZ.
$\text{IROW}(i-1) < \text{IROW}(i)$, or
$\text{IROW}(i-1) = \text{IROW}(i)$ and $\text{ICOL}(i-1) < \text{ICOL}\}(i)$, for $i = 2, 3, \ldots,$NNZ.

Therefore a non-zero element has been supplied which does not lie in the matrix $A$, is out of order, or has duplicate row and column indices. Call F11ZAF to reorder and sum or remove duplicates.

IFAIL = 4

On entry, the matrix $A$ has a zero diagonal element. The SSOR preconditioner is not appropriate for this problem.

# 7 Accuracy

If TRANS = 'N' the computed solution $x$ is the exact solution of a perturbed system of equations $(M + \delta M)x = y$, where

$$|\delta M| \leq c(n)\epsilon |D + \omega L||D^{-1}||D + \omega U|,$$

$c(n)$ is a modest linear function of $n$, and $\epsilon$ is the ***machine precision***. An equivalent result holds when TRANS = 'T'.

# 8 Further Comments

## 8.1 Timing

The time taken for a call to F11DDF is proportional to NNZ.

## 8.2 Use of CHECK

It is expected that a common use of F11DDF will be to carry out the preconditioning step required in the application of F11BBF to sparse linear systems. In this situation F11DDF is likely to be called many times with the same matrix $M$. In the interests of both reliability and efficiency, you are recommended to set CHECK to 'C' for the first of such calls, and to 'N' for all subsequent calls.

# 9 Example

This example program solves a sparse linear system of equations:

$$Ax = b,$$

using RGMRES with SSOR preconditioning.

The RGMRES algorithm itself is implemented by the reverse communication routine F11BBF, which returns repeatedly to the calling program with various values of the parameter IREVCM. This parameter indicates the action to be taken by the calling program.

If IREVCM = 1 a matrix-vector product $v = Au$ is required. This is implemented by a call to F11XAF.

If IREVCM = $-1$ a transposed matrix-vector product $v = A^T u$ is required in the estimation of the norm of $A$. This is implemented by a call to F11XAF.

If IREVCM = 2 a solution of the preconditioning equation $Mv = u$ is required. This is achieved by a call to F11DDF.

If IREVCM = 4 F11BBF has completed its tasks. Either the iteration has terminated, or an error condition has arisen.

For further details see the routine document for F11BBF.

## 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.

```
*     F11DDF Example Program Text
*     Mark 18 Release. NAG Copyright 1997.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LA, LIWORK, LWORK
      PARAMETER        (NMAX=1000,LA=10000,LIWORK=2*NMAX+1,LWORK=10000)
*     .. Local Scalars ..
      real             ANORM, OMEGA, SIGMAX, STPLHS, STPRHS, TOL
      INTEGER          I, IFAIL, IREVCM, ITERM, ITN, LWNEED, LWREQ, M,
     +                 MAXITN, MONIT, N, NNZ
      CHARACTER        CKDDF, CKXAF, NORM, PRECON, TRANS, WEIGHT
      CHARACTER*8      METHOD
*     .. Local Arrays ..
      real             A(LA), B(NMAX), RDIAG(NMAX), WORK(LWORK), X(NMAX)
      INTEGER          ICOL(LA), IROW(LA), IWORK(LIWORK)
*     .. External Subroutines ..
      EXTERNAL         F11BAF, F11BBF, F11BCF, F11DDF, F11XAF
*     .. Intrinsic Functions ..
      INTRINSIC        MAX
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11DDF Example Program Results'
      WRITE (NOUT,*)
*     Skip heading in data file
      READ (NIN,*)
*
```

```
*     Read algorithmic parameters
*
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
         READ (NIN,*) NNZ
         READ (NIN,*) METHOD
         READ (NIN,*) PRECON, NORM, ITERM
         READ (NIN,*) M, TOL, MAXITN
         READ (NIN,*) ANORM, SIGMAX
         READ (NIN,*) OMEGA
*
*        Check size of workspace
*
         LWREQ = MAX(3*N+M*(M+N+4)+1,7*N,2*N*(M+2)+M*(M+2)+2*N)
         IF (LWORK.LT.LWREQ) THEN
            WRITE (NOUT,*) 'LWORK must be at least', LWREQ
            STOP
         END IF
*
*        Read the matrix A
*
         DO 20 I = 1, NNZ
            READ (NIN,*) A(I), IROW(I), ICOL(I)
   20    CONTINUE
*
*        Read right-hand side vector b and initial approximate solution x
*
         READ (NIN,*) (B(I),I=1,N)
         READ (NIN,*) (X(I),I=1,N)
*
*        Call F11BAF to initialize solver
*
         WEIGHT = 'N'
         MONIT = 0
         IFAIL = 0
         CALL F11BAF(METHOD,PRECON,NORM,WEIGHT,ITERM,N,M,TOL,MAXITN,
     +               ANORM,SIGMAX,MONIT,LWNEED,IFAIL)
*
*        Calculate reciprocal diagonal matrix elements if necessary
*
         IF (PRECON.EQ.'P' .OR. PRECON.EQ.'p') THEN
*
            DO 40 I = 1, N
               IWORK(I) = 0
   40       CONTINUE
*
            DO 60 I = 1, NNZ
               IF (IROW(I).EQ.ICOL(I)) THEN
                  IWORK(IROW(I)) = IWORK(IROW(I)) + 1
                  IF (A(I).NE.0.e0) THEN
                     RDIAG(IROW(I)) = 1.e0/A(I)
                  ELSE
                     WRITE (NOUT,*) 'Matrix has a zero diagonal element'
                     GO TO 140
                  END IF
               END IF
   60       CONTINUE
```

```
*
            DO 80 I = 1, N
               IF (IWORK(I).EQ.0) THEN
                  WRITE (NOUT,*) 'Matrix has a missing diagonal element'
                  GO TO 140
               END IF
               IF (IWORK(I).GE.2) THEN
                  WRITE (NOUT,*)
     +              'Matrix has a multiple diagonal element'
                  GO TO 140
               END IF
   80       CONTINUE
*
         END IF
*
*        Call F11BBF to solve the linear system
*
         IREVCM = 0
         CKXAF = 'C'
         CKDDF = 'C'
*
  100    CONTINUE
*
         CALL F11BBF(IREVCM,X,B,WORK,LWORK,IFAIL)
*
         IF (IREVCM.EQ.1) THEN
*
*           Compute matrix-vector product
*
            TRANS = 'Not transposed'
            CALL F11XAF(TRANS,N,NNZ,A,IROW,ICOL,CKXAF,X,B,IFAIL)
            CKXAF = 'N'
            GO TO 100
*
         ELSE IF (IREVCM.EQ.-1) THEN
*
*           Compute transposed matrix-vector product
*
            TRANS = 'Transposed'
            CALL F11XAF(TRANS,N,NNZ,A,IROW,ICOL,CKXAF,X,B,IFAIL)
            CKXAF = 'N'
            GO TO 100
*
         ELSE IF (IREVCM.EQ.2) THEN
*
*           SSOR preconditioning
*
            TRANS = 'Not transposed'
            CALL F11DDF(TRANS,N,NNZ,A,IROW,ICOL,RDIAG,OMEGA,CKDDF,X,B,
     +                  IWORK,IFAIL)
            CKDDF = 'N'
            GO TO 100
*
         ELSE IF (IREVCM.EQ.4) THEN
*
```

```
*           Termination
*
            CALL F11BCF(ITN,STPLHS,STPRHS,ANORM,SIGMAX,IFAIL)
*
            WRITE (NOUT,'(A,I10,A)') 'Converged in', ITN, ' iterations'
            WRITE (NOUT,'(A,1P,e16.3)') 'Matrix norm =', ANORM
            WRITE (NOUT,'(A,1P,e16.3)') 'Final residual norm =', STPLHS
            WRITE (NOUT,*)
*
*           Output x
*
            WRITE (NOUT,*) '           X'
            DO 120 I = 1, N
               WRITE (NOUT,'(1X,1P,e16.4)') X(I)
  120       CONTINUE
*
         END IF
*
  140    CONTINUE
*
      END IF
      STOP
      END
```

## 9.2 Program Data

```
F11DDF Example Program Data
  5                    N
 16                    NNZ
 'RGMRES'              METHOD
 'P' 'I' 1             PRECON, NORM, ITERM
 2 1.e-10 1000         M, TOL, MAXITN
 0.e0 0.e0             ANORM, SIGMAX
 1.1e0                 OMEGA
  2.   1    1
  1.   1    2
 -1.   1    4
 -3.   2    2
 -2.   2    3
  1.   2    5
  1.   3    1
  5.   3    3
  3.   3    4
  1.   3    5
 -2.   4    1
 -3.   4    4
 -1.   4    5
  4.   5    2
 -2.   5    3
 -6.   5    5          A(I), IROW(I), ICOL(I), I=1,...,NNZ
  0.  -7.  33.
-19. -28.               B(I), I=1,...,N
  0.   0.   0.
  0.   0.               X(I), I=1,...,N
```

## 9.3 Program Results

```
 F11DDF Example Program Results

Converged in         12 iterations
Matrix norm =        1.200E+01
Final residual norm =        3.841E-09

           X
      1.0000E+00
      2.0000E+00
      3.0000E+00
      4.0000E+00
      5.0000E+00
```

# F11DEF – 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.

## 1 Purpose

F11DEF solves a real sparse nonsymmetric system of linear equations, represented in coordinate storage format, using a restarted generalized minimal residual (RGMRES), conjugate gradient squared (CGS), or stabilized bi-conjugate gradient (Bi-CGSTAB) method, without preconditioning, with Jacobi, or with SSOR preconditioning.

## 2 Specification

```
      SUBROUTINE F11DEF(METHOD, PRECON, N, NNZ, A, IROW, ICOL, OMEGA, B,
     1                  M, TOL, MAXITN, X, RNORM, ITN, WORK, LWORK,
     2                  IWORK, IFAIL)
      INTEGER           N, NNZ, IROW(NNZ), ICOL(NNZ), M, MAXITN, ITN,
     1                  IWORK(2*N+1), LWORK, IFAIL
      real              A(NNZ), OMEGA, B(N), TOL, X(N), RNORM,
     1                  WORK(LWORK)
      CHARACTER*(*)     METHOD
      CHARACTER*1       PRECON
```

## 3 Description

This routine solves a real sparse nonsymmetric system of linear equations:

$$Ax = b,$$

using an RGMRES [1], CGS [3], or Bi-CGSTAB($\ell$) [4], [2] method.

The routine allows the following choices for the preconditioner:

no preconditioning;

Jacobi preconditioning [5];

symmetric successive-over-relaxation (SSOR) preconditioning [5].

For incomplete $LU$ (ILU) preconditioning see F11DCF.

The matrix $A$ is represented in coordinate storage (CS) format (see Section 2.1.1 of the Chapter Introduction) in the arrays A, IROW and ICOL. The array A holds the non-zero entries in the matrix, while IROW and ICOL hold the corresponding row and column indices.

F11DEF is a black-box routine which calls F11BAF, F11BBF and F11BCF. If you wish to use an alternative storage scheme, preconditioner, or termination criterion, or require additional diagnostic information, you should call these underlying routines directly.

## 4 References

[1] Saad Y and Schultz M (1986) GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems *SIAM J. Sci. Statist. Comput.* **7** 856–869

[2] Sleijpen G L G and Fokkema D R (1993) BiCGSTAB($\ell$) for linear equations involving matrices with complex spectrum *ETNA* **1** 11–32

[3] Sonneveld P (1989) CGS, a fast Lanczos-type solver for nonsymmetric linear systems *SIAM J. Sci. Statist. Comput.* **10** 36–52

[4] van der Vorst H (1989) Bi-CGSTAB, A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems *SIAM J. Sci. Statist. Comput.* **13** 631–644

[5] Young D (1971) *Iterative Solution of Large Linear Systems* Academic Press, New York

# 5 Parameters

**1:** METHOD — CHARACTER*(*) *Input*

*On entry:* the iterative method to be used. The possible choices are:

| | |
|---|---|
| 'RGMRES' | restarted generalized minimum residual method; |
| 'CGS' | conjugate gradient squared method; |
| 'BICGSTAB' | bi-conjugate gradient stabilized ($\ell$) method; |

*Constraint:* METHOD = 'RGMRES', 'CGS' or 'BICGSTAB'.

**2:** PRECON — CHARACTER*1 *Input*

*On entry:* specifies the type of preconditioning to be used. The possible choices are:

'N' no preconditioning;
'J' Jacobi;
'S' symmetric successive-over-relaxation.

*Constraint:* PRECON = 'N', 'J', or 'S'.

**3:** N — INTEGER *Input*

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

*Constraint:* N $\geq 1$.

**4:** NNZ — INTEGER *Input*

*On entry:* the number of non-zero elements in the matrix $A$.

*Constraint:* $1 \leq \text{NNZ} \leq \text{N}^2$.

**5:** A(NNZ) — ***real*** array *Input*

*On entry:* the non-zero elements of the matrix $A$, ordered by increasing row index, and by increasing column index within each row. Multiple entries for the same row and column indices are not permitted. The routine F11ZAF may be used to order the elements in this way.

**6:** IROW(NNZ) — INTEGER array *Input*

**7:** ICOL(NNZ) — INTEGER array *Input*

*On entry:* the row and column indices of the non-zero elements supplied in A.

*Constraints:* IROW and ICOL must satisfy the following constraints (which may be imposed by a call to F11ZAF):

$1 \leq \text{IROW}(i) \leq \text{N}$ and $1 \leq \text{ICOL}(i) \leq \text{N}$, for $i = 1, 2, \ldots, \text{NNZ}$.
$\text{IROW}(i-1) < \text{IROW}(i)$, or
$\text{IROW}(i-1) = \text{IROW}(i)$ and $\text{ICOL}(i-1) < \text{ICOL}(i)$, for $i = 2, 3, \ldots, \text{NNZ}$.

**8:** OMEGA — ***real*** *Input*

*On entry:* if PRECON = 'S', OMEGA is the relaxation parameter $\omega$ to be used in the SSOR method. Otherwise OMEGA need not be initialized and is not referenced.

*Constraint:* 0.0 < OMEGA < 2.0.

**9:** B(N) — ***real*** array *Input*

*On entry:* the right-hand side vector $b$.

**10:** M — INTEGER *Input*

*On entry:* if METHOD = 'RGMRES', M is the dimension of the restart subspace;
if METHOD = 'BICGSTAB', M is the order $\ell$ of the polynomial Bi-CGSTAB method;
otherwise, M is not referenced.

*Constraints:*

if METHOD = 'RGMRES', $0 < \mathrm{M} \leq \min(\mathrm{N},50)$;
if METHOD = 'BICGSTAB', $0 < \mathrm{M} \leq \min(\mathrm{N},10)$.

**11:** TOL — ***real*** *Input*

*On entry:* the required tolerance. Let $x_k$ denote the approximate solution at iteration $k$, and $r_k$ the corresponding residual. The algorithm is considered to have converged at iteration $k$ if:

$$\|r_k\|_\infty \leq \tau \times (\|b\|_\infty + \|A\|_\infty \|x_k\|_\infty).$$

If TOL $\leq 0.0$, $\tau = \max(\sqrt{\epsilon}, \sqrt{n}\,\epsilon)$ is used, where $\epsilon$ is the ***machine precision***. Otherwise $\tau = \max(\mathrm{TOL}, 10\epsilon, \sqrt{n}\,\epsilon)$ is used.

*Constraint:* TOL $< 1.0$.

**12:** MAXITN — INTEGER *Input*

*On entry:* the maximum number of iterations allowed.

*Constraint:* MAXITN $\geq 1$.

**13:** X(N) — ***real*** array *Input/Output*

*On entry:* an initial approximation to the solution vector $x$.

*On exit:* an improved approximation to the solution vector $x$.

**14:** RNORM — ***real*** *Output*

*On exit:* the final value of the residual norm $\|r_k\|_\infty$,where $k$ is the output value of ITN.

**15:** ITN — INTEGER *Output*

*On exit:* the number of iterations carried out.

**16:** WORK(LWORK) — ***real*** array *Workspace*

**17:** LWORK — INTEGER *Input*

*On entry:* the dimension of the array WORK as declared in the (sub)program from which F11DEF is called.

*Constraint:*

if METHOD = 'RGMRES' then LWORK $\geq 4 \times \mathrm{N} + \mathrm{M} \times (\mathrm{M}+\mathrm{N}+4) + \nu + 1$;
if METHOD = 'CGS' then LWORK $\geq 8 \times \mathrm{N} + \nu$;
if METHOD = 'BICGSTAB' then LWORK $\geq 2 \times \mathrm{N} \times (\mathrm{M}+2) + \mathrm{M} \times (\mathrm{M}+2) + \mathrm{N} + 2\,\nu$;

where $\nu = \mathrm{N}$ for PRECON = 'J' or 'S', and 0 otherwise.

**18:** IWORK(2*N+1) — INTEGER array *Workspace*

**19:** IFAIL — INTEGER *Input/Output*

*On entry:* IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

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

# 6 Errors and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL = 1

On entry, METHOD ≠ 'RGMRES', 'CGS', or 'BICGSTAB',
or PRECON ≠ 'N', 'J' or 'S',
or N < 1,
or NNZ < 1,
or NNZ > $N^2$,
or PRECON = 'S' and OMEGA lies outside the interval (0.0,2.0),
or M < 1,
or M > min(N,50), with METHOD = 'RGMRES',
or M > min(N,10), with METHOD = 'BICGSTAB',
or TOL ≥ 1.0,
or MAXITN < 1,
or LWORK too small.

IFAIL = 2

On entry, the arrays IROW and ICOL fail to satisfy the following constraints:

$1 \leq \mathrm{IROW}(i) \leq \mathrm{N}$ and $1 \leq \mathrm{ICOL}(i) \leq \mathrm{N}$, for $i = 1, 2, \ldots,$NNZ.
$\mathrm{IROW}(i-1) < \mathrm{IROW}(i)$, or
$\mathrm{IROW}(i-1) = \mathrm{IROW}(i)$ and $\mathrm{ICOL}(i-1) < \mathrm{ICOL}(i)$, for $i = 2, 3, \ldots,$NNZ.

Therefore a non-zero element has been supplied which does not lie within the matrix $A$, is out of order, or has duplicate row and column indices. Call F11ZAF to reorder and sum or remove duplicates.

IFAIL = 3

On entry, the matrix $A$ has a zero diagonal element. Jacobi and SSOR preconditioners are not appropriate for this problem.

IFAIL = 4

The required accuracy could not be obtained. However, a reasonable accuracy may have been obtained, and further iterations could not improve the result. You should check the output value of RNORM for acceptability. This error code usually implies that your problem has been fully and satisfactorily solved to within or close to the accuracy available on your system. Further iterations are unlikely to improve on this situation.

IFAIL = 5

Required accuracy not obtained in MAXITN iterations.

IFAIL = 6

A serious error has occurred in an internal call to F11BAF, F11BBF or F11BCF. Check all subroutine calls and array sizes. Seek expert help.

# 7 Accuracy

On successful termination, the final residual $r_k = b - Ax_k$, where $k =$ ITN, satisfies the termination criterion

$$\|r_k\|_\infty \leq \tau \times (\|b\|_\infty + \|A\|_\infty \|x_k\|_\infty).$$

The value of the final residual norm is returned in RNORM.

# 8 Further Comments

The time taken by F11DEF for each iteration is roughly proportional to NNZ.

The number of iterations required to achieve a prescribed accuracy cannot be easily determined a priori, as it can depend dramatically on the conditioning and spectrum of the preconditioned coefficient matrix $\bar{A} = M^{-1}A$.

# 9 Example

This example program solves a sparse nonsymmetric system of equations using the RGMRES method, with SSOR preconditioning.

## 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.

```
*     F11DEF Example Program Text
*     Mark 18 Release. NAG Copyright 1997.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LA, LWORK
      PARAMETER        (NMAX=1000,LA=10000,LWORK=10000)
*     .. Local Scalars ..
      real             OMEGA, RNORM, TOL
      INTEGER          I, IFAIL, ITN, LWREQ, M, MAXITN, N, NNZ
      CHARACTER        PRECON
      CHARACTER*8      METHOD
*     .. Local Arrays ..
      real             A(LA), B(NMAX), WORK(LWORK), X(NMAX)
      INTEGER          ICOL(LA), IROW(LA), IWORK(2*NMAX+1)
*     .. External Subroutines ..
      EXTERNAL         F11DEF
*     .. Intrinsic Functions ..
      INTRINSIC        MAX
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11DEF Example Program Results'
      WRITE (NOUT,*)
*     Skip heading in data file
      READ (NIN,*)
*
*     Read algorithmic parameters
*
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
         READ (NIN,*) NNZ
         READ (NIN,*) METHOD, PRECON
         READ (NIN,*) OMEGA
         READ (NIN,*) M, TOL, MAXITN
*
```

```
*        Check size of workspace
*
         LWREQ = MAX(5*N+M*(M+N+4)+1,9*N,2*N*(M+2)+M*(M+2)+3*N)
         IF (LWORK.LT.LWREQ) THEN
            WRITE (NOUT,*) 'LWORK must be at least', LWREQ
            STOP
         END IF
*
*        Read the matrix A
*
         DO 20 I = 1, NNZ
            READ (NIN,*) A(I), IROW(I), ICOL(I)
   20    CONTINUE
*
*        Read right-hand side vector b and initial approximate solution x
*
         READ (NIN,*) (B(I),I=1,N)
         READ (NIN,*) (X(I),I=1,N)
*
*        Solve Ax = b using F11DEF
*
         IFAIL = 0
         CALL F11DEF(METHOD,PRECON,N,NNZ,A,IROW,ICOL,OMEGA,B,M,TOL,
     +               MAXITN,X,RNORM,ITN,WORK,LWORK,IWORK,IFAIL)
*
         WRITE (NOUT,'(A,I10,A)') 'Converged in', ITN, ' iterations'
         WRITE (NOUT,'(A,1P,e16.3)') 'Final residual norm =', RNORM
         WRITE (NOUT,*)
*
*        Output x
*
         WRITE (NOUT,*) '           X'
         DO 40 I = 1, N
            WRITE (NOUT,'(1X,1P,e16.4)') X(I)
   40    CONTINUE
*
      END IF
      STOP
      END
```

## 9.2 Program Data

```
F11DEF Example Program Data
  5                    N
 16                    NNZ
  'RGMRES' 'S'         METHOD, PRECON
  1.05                  OMEGA
  1 1.e-10 1000        M, TOL, MAXITN
  2.   1    1
  1.   1    2
 -1.   1    4
 -3.   2    2
 -2.   2    3
  1.   2    5
  1.   3    1
  5.   3    3
  3.   3    4
```

```
 1.   3    5
-2.   4    1
-3.   4    4
-1.   4    5
 4.   5    2
-2.   5    3
-6.   5    5          A(I), IROW(I), ICOL(I), I=1,...,NNZ
 0.  -7.  33.
-19. -28.             B(I), I=1,...,N
 0.   0.   0.
 0.   0.              X(I), I=1,...,N
```

## 9.3 Program Results

```
 F11DEF Example Program Results

Converged in         13 iterations
Final residual norm =       5.087E-09

            X
      1.0000E+00
      2.0000E+00
      3.0000E+00
      4.0000E+00
      5.0000E+00
```

# F11GAF – 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.

## 1 Purpose

F11GAF is a set-up routine, the first in a suite of three routines for the iterative solution of a symmetric system of simultaneous linear equations. F11GAF must be called before F11GBF, the iterative solver. The third routine in the suite, F11GCF, can be used to return additional information about the computation.

These three routines are suitable for the solution of large sparse symmetric systems of equations.

## 2 Specification

```
      SUBROUTINE F11GAF(METHOD, PRECON, SIGCMP, NORM, WEIGHT, ITERM, N,
     1                  TOL, MAXITN, ANORM, SIGMAX, SIGTOL, MAXITS,
     2                  MONIT, LWREQ, IFAIL)
      INTEGER           ITERM, N, MAXITN, MAXITS, MONIT, LWREQ, IFAIL
      real              TOL, ANORM, SIGMAX, SIGTOL
      CHARACTER*1       PRECON, SIGCMP, NORM, WEIGHT
      CHARACTER*(*)     METHOD
```

## 3 Description

The suite consisting of the routines F11GAF, F11GBF, F11GCF is designed to solve the symmetric system of simultaneous linear equations $Ax = b$ of order $n$, where $n$ is large and the matrix of the coefficients $A$ is sparse.

F11GAF is a set-up routine which must be called before F11GBF, the iterative solver. The third routine in the suite, F11GCF can be used to return additional information about the computation. Either of two methods can be used:

**Conjugate Gradient Method** For this method (Hestenes and Stiefel [1], Golub and Van Loan [2], Barrett *et al.* [3], Dias da Cunha and Hopkins [4]), the matrix $A$ should ideally be positive-definite. The application of the Conjugate Gradient method to indefinite matrices may lead to failure or to lack of convergence.

**Lanczos Method (SYMMLQ)** This method, based upon the algorithm SYMMLQ (Paige and Saunders [5], Barrett *et al.* [3]), is suitable for both positive-definite and indefinite matrices. It is more robust than the Conjugate Gradient method but less efficient when $A$ is positive-definite.

Both conjugate gradient and Lanczos (SYMMLQ) methods start from the residual $r_0 = b - Ax_0$, where $x_0$ is an initial estimate for the solution (often $x_0 = 0$), and generate an orthogonal basis for the Krylov subspace span$\{A^k r_0\}$, for $k = 0, 1, \ldots$, by means of three-term recurrence relations (Golub and Van Loan [2]). A sequence of symmetric tridiagonal matrices $\{T_k\}$ is also generated. Here and in the following, the index $k$ denotes the iteration count. The resulting symmetric tridiagonal systems of equations are usually more easily solved than the original problem. A sequence of solution iterates $\{x_k\}$ is thus generated such that the sequence of the norms of the residuals $\{\|r_k\|\}$ converges to a required tolerance. Note that, in general, the convergence is not monotonic.

In exact arithmetic, after $n$ iterations, this process is equivalent to an orthogonal reduction of $A$ to symmetric tridiagonal form, $T_n = Q^T AQ$; the solution $x_n$ would thus achieve exact convergence. In finite-precision arithmetic, cancellation and round-off errors accumulate causing loss of orthogonality. These methods must therefore be viewed as genuinely iterative methods, able to converge to a solution **within a prescribed tolerance**.

The orthogonal basis is not formed explicitly in either method. The basic difference between the two methods lies in the method of solution of the resulting symmetric tridiagonal systems of equations: the

conjugate gradient method is equivalent to carrying out an $LDL^T$ (Cholesky) factorization whereas the Lanczos method (SYMMLQ) uses an $LQ$ factorization.

Faster convergence can be achieved using a **preconditioner** (Golub and Van Loan [2], Barrett *et al.* [3]). A preconditioner maps the original system of equations onto a different system, say

$$\bar{A}\bar{x} = \bar{b}, \tag{1}$$

with, hopefully, better characteristics with respect to its speed of convergence: for example, the condition number of the matrix of the coefficients can be improved or eigenvalues in its spectrum can be made to coalesce. An orthogonal basis for the Krylov subspace span$\{\bar{A}^k\bar{r}_0\}$, for $k = 0, 1, \ldots$, is generated and the solution proceeds as outlined above. The algorithms used are such that the solution and residual iterates of the original system are produced, not their preconditioned counterparts. Note that an unsuitable preconditioner or no preconditioning at all may result in a very slow rate, or lack, of convergence. However, preconditioning involves a trade-off between the reduction in the number of iterations required for convergence and the additional computational costs per iteration. Also, setting up a preconditioner may involve non-negligible overheads.

A preconditioner must be **symmetric and positive-definite**, i.e. representable by $M = EE^T$, where $M$ is non-singular, and such that $\bar{A} = E^{-1}AE^{-T} \sim I_n$ in (1), where $I_n$ is the identity matrix of order $n$. Also, we can define $\bar{r} = E^{-1}r$ and $\bar{x} = E^Tx$. These are formal definitions, used only in the design of the algorithms; in practice, only the means to compute the matrix-vector products $v = Au$ and to solve the preconditioning equations $Mv = u$ are required, that is, explicit information about $M$, $E$ or their inverses in not required at any stage.

The first termination criterion

$$\|r_k\|_p \leq \tau\,(\|b\|_p + \|A\|_p\,\|x_k\|_p) \tag{2}$$

is available for both conjugate gradient and Lanczos (SYMMLQ) methods. In (2), $p = 1, \infty$ or 2 and $\tau$ denotes a user-specified tolerance subject to $\max(10, \sqrt{n})\,\epsilon \leq \tau < 1$, where $\epsilon$ is the ***machine precision***. Facilities are provided for the estimation of the norm of the matrix of the coefficients $\|A\|_1 = \|A\|_\infty$, when this is not known in advance, used in (2), by applying Higham's method (Higham [6]). Note that $\|A\|_2$ cannot be estimated internally. This criterion uses an error bound derived from **backward** error analysis to ensure that the computed solution is the exact solution of a problem as close to the original as the termination tolerance requires. Termination criteria employing bounds derived from **forward** error analysis could be used, but any such criteria would require information about the condition number $\kappa(A)$ which is not easily obtainable.

The second termination criterion

$$\|\bar{r}_k\|_2 \leq \tau\,\max(1.0,\, \|b\|_2/\|r_0\|_2)\,(\|\bar{r}_0\|_2 + \sigma_1(\bar{A})\,\|\Delta\bar{x}_k\|_2) \tag{3}$$

is available only for the Lanczos method (SYMMLQ). In (3), $\sigma_1(\bar{A}) = \|\bar{A}\|_2$ is the largest singular value of the (preconditioned) iteration matrix $\bar{A}$. This termination criterion monitors the progress of the solution of the preconditioned system of equations and is less expensive to apply than criterion (2). When $\sigma_1(\bar{A})$ is not supplied, facilities are provided for its estimation by $\sigma_1(\bar{A}) \sim \max_k \sigma_1(T_k)$. The interlacing property $\sigma_1(T_{k-1}) \leq \sigma_1(T_k)$ and Gerschgorin's theorem provide lower and upper bounds from which $\sigma_1(T_k)$ can be easily computed by bisection. Alternatively, the less expensive estimate $\sigma_1(\bar{A}) \sim \max_k \|T_k\|_1$ can be used, where $\sigma_1(\bar{A}) \leq \|T_k\|_1$ by Gerschgorin's theorem. Note that only order of magnitude estimates are required by the termination criterion.

Termination criterion (2) is the recommended choice, despite its (small) additional costs per iteration when using the Lanczos method (SYMMLQ). Also, if the norm of the initial estimate is much larger than the norm of the solution, that is, if $\|x_0\| \gg \|x\|$, a dramatic loss of significant digits could result in complete lack of convergence. The use of criterion (2) will enable the detection of such a situation, and the iteration will be restarted at a suitable point. No such restart facilities are provided for criterion (3).

Optionally, a vector $w$ of user-specified weights can be used in the computation of the vector norms in termination criterion (2), i.e. $\|v\|_p^{(w)} = \|v^{(w)}\|_p$, where $(v^{(w)})_i = w_i\,v_i$, for $i = 1, 2, \ldots, n$. Note that the use of weights increases the computational costs.

The sequence of calls to the routines comprising the suite is enforced: first, the set-up routine F11GAF must be called, followed by the solver F11GBF. F11GCF can be called either when F11GBF is carrying out a monitoring step or after F11GBF has completed its tasks. Incorrect sequencing will raise an error condition.

# 4 References

[1] Hestenes M and Stiefel E (1952) Methods of conjugate gradients for solving linear systems *J. Res. Nat. Bur. Stand.* **49** 409–436

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

[3] Barrett R, Berry M, Chan T F, Demmel J, Donato J, Dongarra J, Eijkhout V, Pozo R, Romine C and van der Vorst H (1994) *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods* SIAM, Philadelphia

[4] Dias da Cunha R and Hopkins T (1994) PIM 1.1 — the the parallel iterative method package for systems of linear equations user's guide — Fortran 77 version *Technical Report* Computing Laboratory, University of Kent at Canterbury, Kent CT2 7NZ, UK

[5] Paige C C and Saunders M A (1975) Solution of sparse indefinite systems of linear equations *SIAM J. Numer. Anal.* **12** 617–629

[6] Higham N J (1988) 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

# 5 Parameters

**1:** METHOD — CHARACTER*(*) *Input*

*On entry:* the iterative method to be used. The possible choices are:

'CG' conjugate gradient method;
'SYMMLQ' Lanczos method (SYMMLQ).

*Constraint:* METHOD = 'CG' or 'SYMMLQ'.

**2:** PRECON — CHARACTER*1 *Input*

*On entry:* determines whether preconditioning is used. The possible choices are:

'N' no preconditioning;
'P' preconditioning.

*Constraint:* PRECON = 'N' or 'P'.

**3:** SIGCMP — CHARACTER*1 *Input*

*On entry:* determines whether an estimate of $\sigma_1(\bar{A}) = \|E^{-1}AE^{-T}\|_2$, the largest singular value of the preconditioned matrix of the coefficients, is to be computed using the bisection method on the sequence of tridiagonal matrices $\{T_k\}$ generated during the iteration. Note that $\bar{A} = A$ when a preconditioner is not used.
If SIGMAX > 0.0 (see below), i.e. when $\sigma_1(\bar{A})$ is supplied, the value of SIGCMP is ignored.
The possible choices are:

'S' $\sigma_1(\bar{A})$ is to be computed using the bisection method.
'N' The bisection method is not used.
If the termination criterion (3) is used, requiring $\sigma_1(\bar{A})$, an inexpensive estimate is computed and used (see Section 3).

*Suggested value:* SIGCMP = 'N'.

*Constraint:* SIGCMP = 'S' or 'N'.

**4:** NORM — CHARACTER*1 *Input*

*On entry:* defines the matrix and vector norm to be used in the termination criteria. The possible choices are:

'1' use the $l_1$ norm;

'I' use the $l_\infty$ norm;

'2' use the $l_2$ norm.

*Suggested value:* NORM = 'I', if ITERM = 1; NORM = '2', if ITERM = 2 (see below).

*Constraints:* if ITERM = 1, then NORM = '1', 'I' or '2'; if ITERM = 2, then NORM = '2'.

**5:** WEIGHT — CHARACTER*1 *Input*

*On entry:* specifies whether a vector $w$ of user-supplied weights is to be used in the vector norms used in the computation of termination criterion (2) (ITERM = 1): $\|v\|_p^{(w)} = \|v^{(w)}\|_p$, where $v_i^{(w)} = w_i\, v_i$, for $i = 1, 2, \ldots, n$. The suffix $p = 1, 2, \infty$ denotes the vector norm used, as specified by the parameter NORM. Note that weights cannot be used when ITERM = 2, i.e. when criterion (3) is used. The possible choices are:

'W' user-supplied weights are to be used and must be supplied on initial entry to F11GBF.

'N' all weights are implicitly set equal to one. Weights do not need to be supplied on initial entry to F11GBF.

*Suggested value:* WEIGHT = 'N'.

*Constraints:* if ITERM = 1, then WEIGHT = 'W' or 'N'; if ITERM = 2, then WEIGHT = 'N'.

**6:** ITERM — INTEGER *Input*

*On entry:* defines the termination criterion to be used. The possible choices are:

1 use the termination criterion defined in (2) (both conjugate gradient and Lanczos (SYMMLQ) methods);

2 use the termination criterion defined in (3) (Lanczos method (SYMMLQ) only).

*Suggested value:* ITERM = 1.

*Constraints:* if METHOD = 'CG', then ITERM = 1; if METHOD = 'SYMMLQ', then ITERM = 1 or 2.

**7:** N — INTEGER *Input*

*On entry:* the order $n$ of the matrix $A$.

*Constraint:* N > 0.

**8:** TOL — ***real*** *Input*

*On entry:* the tolerance $\tau$ for the termination criterion. If TOL $\leq 0.0$, $\tau = \max(\sqrt{\epsilon}, \sqrt{n}\,\epsilon)$ is used, where $\epsilon$ is the ***machine precision***. Otherwise $\tau = \max(\text{TOL}, 10\epsilon, \sqrt{n}\,\epsilon)$ is used.

*Constraint:* TOL < 1.0.

**9:** MAXITN — INTEGER *Input*

*On entry:* the maximum number of iterations.

*Constraint:* MAXITN > 0.

**10:** ANORM — ***real*** *Input*

*On entry:* if ANORM > 0.0, the value of $\|A\|_p$ to be used in the termination criterion (2) (ITERM = 1).
If ANORM $\leq$ 0.0, ITERM = 1 and NORM = '1' or 'I', then $\|A\|_1 = \|A\|_\infty$ is estimated internally by F11GBF.
If ITERM = 2, then ANORM is not referenced.

*Constraint:* if ITERM = 1 and NORM = '2', then ANORM > 0.0.

**11:** SIGMAX — ***real*** *Input*

*On entry:* if SIGMAX > 0.0, the value of $\sigma_1(\bar{A}) = \|E^{-1}AE^{-T}\|_2$.
If SIGMAX $\leq$ 0.0, $\sigma_1(\bar{A})$ is estimated by F11GBF when either SIGCMP = 'S' or termination criterion (3) (ITERM = 2) is employed, though it will be used only in the latter case. Otherwise, SIGMAX is not referenced.

**12:** SIGTOL — ***real*** *Input*

*On entry:* the tolerance used in assessing the convergence of the estimate of $\sigma_1(\bar{A}) = \|\bar{A}\|_2$ when the bisection method is used. If SIGTOL $\leq$ 0.0, the default value SIGTOL = 0.01 is used. The actual value used is max(SIGTOL, $\epsilon$).
If SIGCMP = 'N' or SIGMAX > 0.0, then SIGTOL is not referenced.

*Suggested value:* SIGTOL = 0.01 should be sufficient in most cases.

*Constraint:* if SIGCMP = 'S' and SIGMAX $\leq$ 0.0, then SIGTOL < 1.0.

**13:** MAXITS — INTEGER *Input*

*On entry:* the maximum iteration number $k$ = MAXITS for which $\sigma_1(T_k)$ is computed by bisection (see also Section 3).
If SIGCMP = 'N' or SIGMAX > 0.0, then MAXITS is not referenced.

*Suggested value:* MAXITS = min(10, $n$) when SIGTOL is of the order of its default value (0.01).

*Constraint:* if SIGCMP = 'S' and SIGMAX $\leq$ 0.0, then 1 $\leq$ MAXITS $\leq$ MAXITN.

**14:** MONIT — INTEGER *Input*

*On entry:* if MONIT > 0, the frequency at which a monitoring step is executed by F11GBF: the current solution and residual iterates will be returned by F11GBF and a call to F11GCF made possible every MONIT iterations, starting from the MONIT-th. Otherwise, no monitoring takes place.
There are some additional computational costs involved in monitoring the solution and residual vectors when the Lanczos method (SYMMLQ) is used.

*Constraint:* MONIT $\leq$ MAXITN.

**15:** LWREQ — INTEGER *Output*

*On exit:* the minimum amount of workspace required by F11GBF. (See also Section 5 of the document for F11GBF.)

**16:** IFAIL — INTEGER *Input/Output*

*On entry:* IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

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

# 6 Errors and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL = $-i$

On entry, the $i$-th argument had an illegal value.

IFAIL = 1

F11GAF has been called out of sequence.

# 7 Accuracy

Not applicable.

# 8 Further Comments

When $\sigma_1(\bar{A})$ is not supplied (SIGMAX $\leq$ 0.0) but it is required, it is estimated by F11GBF using either of the two methods described in Section 3, as specified by the parameter SIGCMP. In particular, if SIGCMP = 'S', then the computation of $\sigma_1(\bar{A})$ is deemed to have converged when the differences between three successive values of $\sigma_1(T_k)$ differ, in a relative sense, by less than the tolerance SIGTOL, i.e. when $\max\left(\frac{|\sigma_1^{(k)}-\sigma_1^{(k-1)}|}{\sigma_1^{(k)}}, \frac{|\sigma_1^{(k)}-\sigma_1^{(k-2)}|}{\sigma_1^{(k)}}\right) \leq$ SIGTOL. The computation of $\sigma_1(\bar{A})$ is also terminated when the iteration count exceeds the maximum value allowed, i.e. $k \geq$ MAXITS.

Bisection is increasingly expensive with increasing iteration count. A reasonably large value of SIGTOL, of the order of the suggested value, is recommended and an excessive value of MAXITS should be avoided. Under these conditions, $\sigma_1(\bar{A})$ usually converges within very few iterations.

# 9 Example

The example solves a 20 × 20 symmetric system of simultaneous linear equations using the conjugate gradient method, where the matrix of the coefficients $A$, has random sparsity pattern. An incomplete Cholesky preconditioner is used (routines F11JAF and F11JBF).

## 9.1 Example 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.

```
*     F11GAF Example Program Text
*     Mark 17 Release. NAG Copyright 1995.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LA, LIWORK, LWORK
      PARAMETER        (NMAX=1000,LA=10000,LIWORK=2*LA+7*NMAX+1,
     +                 LWORK=6*NMAX)
*     .. Local Scalars ..
      real             ANORM, DSCALE, DTOL, SIGERR, SIGMAX, SIGTOL,
     +                 STPLHS, STPRHS, TOL
      INTEGER          I, IFAIL, IFAIL1, IREVCM, ITERM, ITN, ITS, LFILL,
     +                 LWREQ, MAXITN, MAXITS, MONIT, N, NNZ, NNZC, NPIVM
      LOGICAL          LOOP
      CHARACTER        MIC, NORM, PRECON, PSTRAT, SIGCMP, WEIGHT
```

```
      CHARACTER*6       METHOD
*     .. Local Arrays ..
      real              A(LA), B(NMAX), WORK(LWORK), X(NMAX)
      INTEGER           ICOL(LA), IPIV(NMAX), IROW(LA), ISTR(NMAX+1),
     +                  IWORK(LIWORK)
*     .. External Subroutines ..
      EXTERNAL          F11GAF, F11GBF, F11GCF, F11JAF, F11JBF, F11XEF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11GAF Example Program Results'
*
*     Skip heading in data file
*
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*     Read or initialize the parameters for the iterative solver
*
         READ (NIN,*) METHOD
         READ (NIN,*) PRECON, SIGCMP, NORM, WEIGHT, ITERM
         READ (NIN,*) TOL, MAXITN
         READ (NIN,*) MONIT
         ANORM = 0.0e0
         SIGMAX = 0.0e0
         SIGTOL = 1.0e-2
         MAXITS = N
*
*     Read the parameters for the preconditioner
*
         READ (NIN,*) LFILL, DTOL
         READ (NIN,*) MIC, DSCALE
         READ (NIN,*) PSTRAT
*
*     Read the number of non-zero elements of the matrix A, then read
*     the non-zero elements
*
         READ (NIN,*) NNZ
         DO 20 I = 1, NNZ
            READ (NIN,*) A(I), IROW(I), ICOL(I)
   20    CONTINUE
*
*     Read right-hand side vector b and initial approximate solution x
*
         READ (NIN,*) (B(I),I=1,N)
         READ (NIN,*) (X(I),I=1,N)
*
*     Initialize any other parameters
*
*
*     Calculate incomplete Cholesky factorization
*
         IFAIL = 0
         CALL F11JAF(N,NNZ,A,LA,IROW,ICOL,LFILL,DTOL,MIC,DSCALE,PSTRAT,
     +               IPIV,ISTR,NNZC,NPIVM,IWORK,LIWORK,IFAIL)
*
*     Call F11GAF to initialize the solver
*
         IFAIL = 0
```

```
      CALL F11GAF(METHOD,PRECON,SIGCMP,NORM,WEIGHT,ITERM,N,TOL,
     +            MAXITN,ANORM,SIGMAX,SIGTOL,MAXITS,MONIT,LWREQ,
     +            IFAIL)
*
*     Call repeatedly F11GBF to solve the equations
*     Note that the arrays B and X are overwritten
*
*     On final exit, X will contain the solution and B the residual
*     vector
*
      IFAIL = 0
      IREVCM = 0
      LOOP = .TRUE.
*
      LWREQ = LWORK
   40 CONTINUE
      CALL F11GBF(IREVCM,X,B,WORK,LWREQ,IFAIL)
      IF (IREVCM.EQ.1) THEN
         IFAIL1 = -1
         CALL F11XEF(N,NNZ,A,IROW,ICOL,'No checking',X,B,IFAIL1)
         IF (IFAIL1.NE.0) IREVCM = 6
      ELSE IF (IREVCM.EQ.2) THEN
         IFAIL1 = -1
         CALL F11JBF(N,A,LA,IROW,ICOL,IPIV,ISTR,'No checking',X,B,
     +               IFAIL1)
         IF (IFAIL1.NE.0) IREVCM = 6
      ELSE IF (IREVCM.EQ.3) THEN
         IFAIL1 = 0
         CALL F11GCF(ITN,STPLHS,STPRHS,ANORM,SIGMAX,ITS,SIGERR,
     +               IFAIL1)
         WRITE (NOUT,99999) ITN, STPLHS
         WRITE (NOUT,99998)
         WRITE (NOUT,99997) (X(I),B(I),I=1,N)
      ELSE IF (IREVCM.EQ.4) THEN
         LOOP = .FALSE.
      END IF
      IF (LOOP) GO TO 40
*
*     Obtain information about the computation
*
      IFAIL1 = 0
      CALL F11GCF(ITN,STPLHS,STPRHS,ANORM,SIGMAX,ITS,SIGERR,IFAIL1)
*
*     Print the output data
*
      WRITE (NOUT,99996)
      WRITE (NOUT,99995) 'Number of iterations for convergence:    ',
     +  ITN
      WRITE (NOUT,99994) 'Residual norm:                           ',
     +  STPLHS
      WRITE (NOUT,99994) 'Right-hand side of termination criterion:',
     +  STPRHS
      WRITE (NOUT,99994) '1-norm of matrix A:                      ',
     +  ANORM
      WRITE (NOUT,99994) 'Largest singular value of A_bar:         ',
     +  SIGMAX
*
*     Output x
```

```
*
        WRITE (NOUT,99998)
        WRITE (NOUT,99997) (X(I),B(I),I=1,N)
      END IF
      STOP
*
99999 FORMAT (/1X,'Monitoring at iteration no.',I4,/1X,1P,'residual no',
     +       'rm: ',e14.4)
99998 FORMAT (2X,'Solution vector',2X,'Residual vector')
99997 FORMAT (1X,1P,e16.4,1X,e16.4)
99996 FORMAT (/1X,'Final Results')
99995 FORMAT (1X,A,I4)
99994 FORMAT (1X,A,1P,e14.4)
      END
```

## 9.2 Example Data

```
F11GAF Example Program Data
  7                       N
  'CG'                    METHOD
  'P'  'S'  '1'  'N'  1   PRECON, SIGCMP, NORM, WEIGHT, ITERM
  1.0E-6   20             TOL, MAXITN
  2                       MONIT
  0  0.0                  LFILL, DTOL
  'N'  0.0                MIC, DSCALE
  'M'                     PSTRAT
 16                       NNZ
  4.   1    1
  1.   2    1
  5.   2    2
  2.   3    3
  2.   4    2
  3.   4    4
 -1.   5    1
  1.   5    4
  4.   5    5
  1.   6    2
 -2.   6    5
  3.   6    6
  2.   7    1
 -1.   7    2
 -2.   7    3
  5.   7    7          A(I), IROW(I), ICOL(I), I=1,...,NNZ
 15.  18.  -8.  21.
 11.  10.  29.          B(I), I=1,...,N
  0.   0.   0.   0.
  0.   0.   0.          X(I), I=1,...,N
```

## 9.3 Example Results

```
F11GAF Example Program Results

Monitoring at iteration no.   2
residual norm:     1.9938E+00
 Solution vector  Residual vector
      9.6320E-01      -2.2960E-01
      1.9934E+00       2.2254E-01
```

```
      3.0583E+00       9.5827E-02
      4.1453E+00      -2.5155E-01
      4.8289E+00      -1.7160E-01
      5.6630E+00       6.7533E-01
      7.1062E+00      -3.4737E-01

Monitoring at iteration no.   4
residual norm:     6.6574E-03
 Solution vector  Residual vector
      9.9940E-01      -1.0551E-03
      2.0011E+00      -2.4675E-03
      3.0008E+00      -1.7116E-05
      3.9996E+00       4.4929E-05
      4.9991E+00       2.1359E-03
      5.9993E+00      -8.7482E-04
      7.0007E+00       6.2045E-05

Final Results
Number of iterations for convergence:       5
Residual norm:                              2.0428E-14
Right-hand side of termination criterion:   3.9200E-04
1-norm of matrix A:                         1.0000E+01
Largest singular value of A_bar:            1.3596E+00
 Solution vector  Residual vector
      1.0000E+00       0.0000E+00
      2.0000E+00       0.0000E+00
      3.0000E+00      -2.6645E-15
      4.0000E+00      -3.5527E-15
      5.0000E+00      -5.3291E-15
      6.0000E+00       1.7764E-15
      7.0000E+00       7.1054E-15
```

# F11GBF – 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.

## 1 Purpose

F11GBF is an iterative solver for a symmetric system of simultaneous linear equations; F11GBF is the second in a suite of three routines, where the first routine, F11GAF, must be called prior to F11GBF to set-up the suite, and the third routine in the suite, F11GCF, can be used to return additional information about the computation.

These three routines are suitable for the solution of large sparse symmetric systems of equations.

## 2 Specification

```
SUBROUTINE F11GBF(IREVCM, U, V, WORK, LWORK, IFAIL)
INTEGER           IREVCM, LWORK, IFAIL
real              U(*), V(*), WORK(LWORK)
```

## 3 Description

F11GBF solves the symmetric system of linear simultaneous equations $Ax = b$ using either the preconditioned conjugate gradient method (Hestenes and Stiefel [1], Golub and Van Loan [2], Barrett *et al.* [3], Dias da Cunha and Hopkins [4]) or a preconditioned Lanczos method based upon the algorithm SYMMLQ (Paige and Saunders [5], Barrett *et al.* [3]).

For a general description of the methods employed you are referred to Section 3 of the document for F11GAF.

F11GBF can solve the system after the first routine in the suite, F11GAF, has been called to initialize the computation and specify the method of solution. The third routine in the suite, F11GCF, can be used to return additional information generated by the computation during monitoring steps and after F11GBF has completed its tasks.

F11GBF uses **reverse communication**, i.e., F11GBF returns repeatedly to the calling program with the parameter IREVCM (see Section 5) set to specified values which require the calling program to carry out a specific task: either to compute the matrix-vector product $v = Au$; to solve the preconditioning equation $Mv = u$; to notify the completion of the computation; or, to allow the calling program to monitor the solution. Through the parameter IREVCM the calling program can cause immediate or tidy termination of the execution. On final exit, the last iterates of the solution and of the residual vectors of the original system of equations are returned.

Reverse communication has the following advantages.

(1) Maximum flexibility in the representation and storage of sparse matrices. All matrix operations are performed outside the solver routine, thereby avoiding the need for a complicated interface with enough flexibility to cope with all types of storage schemes and sparsity patterns. This applies also to preconditioners.
(2) Enhanced user interaction: the progress of the solution can be closely monitored by the user and tidy or immediate termination can be requested. This is useful, for example, when alternative termination criteria are to be employed or in case of failure of the external routines used to perform matrix operations.

## 4 References

[1] Hestenes M and Stiefel E (1952) Methods of conjugate gradients for solving linear systems *J. Res. Nat. Bur. Stand.* **49** 409–436

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

[3] Barrett R, Berry M, Chan T F, Demmel J, Donato J, Dongarra J, Eijkhout V, Pozo R, Romine C and van der Vorst H (1994) *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods* SIAM, Philadelphia

[4] Dias da Cunha R and Hopkins T (1994) PIM 1.1 — the parallel iterative method package for systems of linear equations user's guide — Fortran 77 version *Technical Report* Computing Laboratory, University of Kent at Canterbury, Kent CT2 7NZ, UK

[5] Paige C C and Saunders M A (1975) Solution of sparse indefinite systems of linear equations *SIAM J. Numer. Anal.* **12** 617–629

[6] Higham N J (1988) 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

# 5 Parameters

**Note.**This routine uses **reverse communication**. Its use involves an initial entry, intermediate exits and re-entries, and a final exit, as indicated by the **parameter IREVCM**. Between intermediate exits and re-entries **all parameters other than IREVCM and V must remain unchanged**.

1: IREVCM — INTEGER *Input/Output*

*On initial entry:* IREVCM = 0, otherwise an error condition will be raised.

*On intermediate re-entry:* IREVCM must either be unchanged from its previous exit value, or can have one of the following values.

5 Tidy termination: the computation will terminate at the end of the current iteration. Further reverse communication exits may occur depending on when the termination request is issued. F11GBF will then return with the termination code IREVCM = 4. Note that before calling F11GBF with IREVCM = 5 the calling program must have performed the tasks required by the value of IREVCM returned by the previous call to F11GBF, otherwise subsequently returned values may be invalid.

6 Immediate termination: F11GBF will return immediately with termination code IREVCM = 4 and with any useful information available. This includes the last iterate of the solution and, for conjugate gradient only, the last iterate of the residual vector. The residual vector is generally not available when the Lanczos method (SYMMLQ) is used. F11GBF will then return with the termination code IREVCM = 4.
Immediate termination may be useful, for example, when errors are detected during matrix-vector multiplication or during the solution of the preconditioning equation.

Changing IREVCM to any other value between calls will result in an error.

*On intermediate exit:* IREVCM has the following meanings.

1 The calling program must compute the matrix-vector product $v = Au$, where $u$ and $v$ are stored in U and V, respectively;

2 The calling program must solve the preconditioning equation $Mv = u$, where $u$ and $v$ are stored in U and V, respectively;

3 Monitoring step: the solution and residual at the current iteration are returned in the arrays U and V, respectively. No action by the calling program is required. F11GCF can be called at this step to return additional information.

*On final exit:* IREVCM = 4: F11GBF has completed its tasks. The value of IFAIL determines whether the iteration has been successfully completed, errors have been detected or the calling program has requested termination.

*Constraints:* on initial entry, IREVCM = 0; on re-entry, either IREVCM must remain unchanged or be reset to 5 or 6.

**2:** U(*) — ***real*** array *Input/Output*

**Note:** the dimension of the array U must be at least $n$.

*On initial entry:* an initial estimate, $x_0$, of the solution of the system of equations $Ax = b$.

*On intermediate re-entry:* U must remain unchanged.

*On intermediate exit:* the returned value of IREVCM determines the contents of U in the following way:

| | |
|---|---|
| IREVCM = 1, 2 | U holds the vector $u$ on which the operation specified by IREVCM is to be carried out; |
| IREVCM = 3 | U holds the current iterate of the solution vector. |

*On final exit:* if IFAIL = 3 or < 0, the array U is unchanged from the initial entry to F11GBF. If IFAIL = 1, the array U is unchanged from the last entry to F11GBF. Otherwise, U holds the last iterate of the solution of the system of equations, for all returned values of IFAIL.

**3:** V(*) — ***real*** array *Input/Output*

**Note:** the dimension of the array V must be at least $n$.

*On initial entry:* the right-hand side $b$ of the system of equations $Ax = b$.

*On intermediate re-entry:* the returned value of IREVCM determines the contents of V in the following way:

| | |
|---|---|
| IREVCM = 1, 2 | V must store the vector $v$, the result of the operation specified by the value of IREVCM returned by the previous call to F11GBF; |
| IREVCM = 3 | V must remain unchanged. |

*On intermediate exit:* if IREVCM = 3, V holds the current iterate of the residual vector. Note that this is an approximation to the true residual vector. Otherwise, it does not contain any useful information.

*On final exit:* if IFAIL = 3 or < 0, the array V is unchanged from the last entry to F11GBF. If IFAIL = 1, the array V is unchanged from the initial entry to F11GBF. If IFAIL = 0 or 2, the array V contains the true residual vector of the system of equations (see also Section 6);
Otherwise, V stores the last iterate of the residual vector unless the Lanczos method (SYMMLQ) was used and IFAIL $\geq$ 5, in which case V is set to 0.0.

**4:** WORK(LWORK) — ***real*** array *Input/Output*

*On initial entry:* if user-supplied weights are used in the computation of the vector norms in the termination criterion (see Section 3 of the document for F11GAF and Section 5 of the document for F11GAF), these must be stored in WORK(1:$n$).
Otherwise, WORK need not be initialized.

*On intermediate re-entry:* WORK must remain unchanged.

*On final exit:* if weights are used, WORK(1:$n$) remains unchanged from the values supplied on initial entry.

**5:** LWORK — INTEGER *Input*

*On initial entry:* the dimension of the array WORK as declared in the (sub)program from which F11GBF was called. The required amount of workspace is as follows:

| **Method** | **Requirements** |
|---|---|
| conjugate gradient | LWORK= $5n$ |
| Lanczos (SYMMLQ) | LWORK= $6n$ |

to this must be added

| | |
|---|---|
| 2 × (MAXITS+1) | if the largest singular value of the iteration matrix is estimated by F11GBF using bisection (see Section 3 of the document for F11GAF, Section 5 of the document for F11GAF and Section 8 of the document for F11GAF); |

*n* if user-defined weights are used in the computation of vector norms for the termination criterion (see Section 3 of the document for F11GAF and Section 5 of the document for F11GAF).

*Constraint:* LWORK $\geq$ LWREQ, where LWREQ is returned by F11GAF.

**6:** IFAIL — INTEGER *Input*

*On entry:* IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

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

# 6 Errors and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL = $-i$

On entry, the $i$th argument had an illegal value.

IFAIL = 1

F11GBF has been called again after returning the termination code IREVCM = 4. No further computation has been carried out and all input data and data stored for access by F11GCF have remained unchanged.

IFAIL = 2

The required accuracy could not be obtained. However, F11GBF has terminated with reasonable accuracy: the last iterate of the residual satisfied the termination criterion but the exact residual $r = b - Ax$, did not. A small number of iterations have been carried out after the iterated residual satisfied the termination criterion, but were unable to improve on the accuracy. This error code usually implies that your problem has been fully and satisfactory solved to within or close to the accuracy available on your system. Further iterations are unlikely to improve on this situation. You should call F11GCF to check the values of the left-and right-hand side of the termination condition.

IFAIL = 3

F11GAF was either not called before calling F11GBF or it returned an error. The arguments U and V remain unchanged.

IFAIL = 4

The calling program requested a tidy termination before the solution had converged. The arrays U and V return the last iterates available of the solution and of the residual vector, respectively.

IFAIL = 5

The solution did not converge within the maximum number of iterations allowed. The arrays U and V return the last iterates available of the solution and of the residual vector, respectively.

IFAIL = 6

The preconditioner appears not to be positive-definite. It is likely that your results are meaningless: both methods require a positive-definite preconditioner (see also Section 3). However, the array U returns the last iterate of the solution, the array V returns the last iterate of the residual vector, for the conjugate gradient method only.

IFAIL = 7

The matrix of the coefficients appears not to be positive-definite (conjugate gradient method only). The arrays U and V return the last iterates of the solution and residual vector, respectively. However, you should be warned that the results returned can be be in error.

IFAIL = 8

The calling program requested an immediate termination. However, the array U returns the last iterate of the solution, the array V returns the last iterate of the residual vector, for the conjugate gradient method only.

# 7 Accuracy

On completion, i.e., IREVCM = 4 on exit, the arrays U and V will return the solution and residual vectors, $x_k$ and $r_k = b - Ax_k$, respectively, at the $k$th iteration, the last iteration performed, unless an immediate termination was requested and the Lanczos method (SYMMLQ) was used.

On successful completion, the termination criterion is satisfied to within the user-specified tolerance, as described in Section 3 of the document for F11GAF. The computed values of the left- and right-hand sides of the termination criterion selected can be obtained by a call to F11GCF.

# 8 Further Comments

The number of operations carried out by F11GBF for each iteration is likely to be principally determined by the computation of the matrix-vector products $v = Au$ and by the solution of the preconditioning equation $Mv = u$ in the calling program. Each of these operations is carried out once every iteration.

The number of the remaining operations in F11GBF for each iteration is approximately proportional to $n$. Note that the Lanczos method (SYMMLQ) requires a slightly larger number of operations than the conjugate gradient method.

The number of iterations required to achieve a prescribed accuracy cannot be easily determined at the onset, as it can depend dramatically on the conditioning and spectrum of the preconditioned matrix of the coefficients $\bar{A} = E^{-1}AE^{-T}$.

Additional matrix-vector products are required for the computation of $\|A\|_1 = \|A\|_\infty$, when this has not been supplied to F11GAF and is required by the termination criterion employed.

The number of operations required to compute $\sigma_1(\bar{A})$ is negligible for reasonable values of SIGTOL and MAXITS (see Section 5 of the document for F11GAF and Section 8 of the document for F11GAF).

If the termination criterion $\|r_k\|_p \leq \tau(\|b\|_p + \|A\|_p \|x_k\|_p)$ is used (see Section 3 of the document for F11GAF) and $\|x_0\| \gg \|x_k\|$, so that because of loss of significant digits the required accuracy could not be obtained, the iteration is restarted automatically at some suitable point: F11GBF sets $x_0 = x_k$ and the computation begins again. For particularly badly scaled problems, more than one restart may be necessary. Naturally, restarting adds to computational costs: it is recommended that the iteration should start from a value $x_0$ which is as close to the true solution $\tilde{x}$ as can be estimated. Otherwise, the iteration should start from $x_0 = 0$.

# 9 Example

See the example for F11GAF.

# F11GCF – 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.

## 1 Purpose

F11GCF is the third in a suite of three routines for the iterative solution of a symmetric system of simultaneous linear equations (Golub and Van Loan [1]). F11GCF returns information about the computations during an iteration and/or after this has been completed. The first routine of the suite, F11GAF, is a set-up routine, the second routine, F11GBF is the proper iterative solver.

These three routines are suitable for the solution of large sparse symmetric systems of equations.

## 2 Specification

```
SUBROUTINE F11GCF(ITN, STPLHS, STPRHS, ANORM, SIGMAX, ITS, SIGERR,
1                 IFAIL)
INTEGER           ITN, ITS, IFAIL
real              STPLHS, STPRHS, ANORM, SIGMAX, SIGERR
```

## 3 Description

F11GCF returns information about the solution process. It can be called either during a monitoring step of F11GBF or after F11GBF has completed its tasks. Calling F11GCF at any other time will result in an error condition being raised.

For further information you should read the documentation for F11GAF and F11GBF.

## 4 References

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

## 5 Parameters

**1:** ITN — INTEGER *Output*

*On exit:* the number of iterations carried out by F11GBF.

**2:** STPLHS — ***real*** *Output*

*On exit:* the current value of the left-hand side of the termination criterion used by F11GBF.

**3:** STPRHS — ***real*** *Output*

*On exit:* the current value of the right-hand side of the termination criterion used by F11GBF.

**4:** ANORM — ***real*** *Output*

*On exit:* the norm $\|A\|_1 = \|A\|_\infty$ when either it has been supplied to F11GAF or it has been estimated by F11GBF (see also Sections 3 and 5 of the document for F11GAF).
Otherwise, ANORM = 0.0 is returned.

**5:** SIGMAX — ***real*** *Output*

*On exit:* the current estimate of the largest singular value $\sigma_1(\bar{A})$ of the preconditioned iteration matrix $\bar{A} = E^{-1}AE^{-T}$, when either it has been supplied to F11GAF or it has been estimated by F11GBF (see also Sections 3 and 5 of the document for F11GAF). Note that if ITS < ITN then SIGMAX contains the final estimate. If, on final exit from F11GBF, ITS = ITN, then the estimation of $\sigma_1(\bar{A})$ may have not converged: in this case you should look at the value returned in SIGERR (see below).
Otherwise, SIGMAX = 0.0 is returned.

**6:** ITS — INTEGER *Output*

*On exit:* the number of iterations employed so far in the computation of the estimate of $\sigma_1(\bar{A})$, the largest singular value of the preconditioned matrix $\bar{A} = E^{-1}AE^{-T}$, when $\sigma_1(\bar{A})$ has been estimated by F11GBF using the bisection method (see also Sections 3, 5 and 8 of the document for F11GAF).
Otherwise, ITS = 0 is returned.

**7:** SIGERR — ***real*** *Output*

*On exit:* if $\sigma_1(\bar{A})$ has been estimated by F11GBF using bisection,

$$\text{SIGERR} = \max\left(\frac{|\sigma_1^{(k)} - \sigma_1^{(k-1)}|}{\sigma_1^{(k)}}, \frac{|\sigma_1^{(k)} - \sigma_1^{(k-2)}|}{\sigma_1^{(k)}}\right),$$

where $k$ = ITS denotes the iteration number. The estimation has converged if SIGERR $\leq$ SIGTOL where SIGTOL is an input parameter to F11GAF.
Otherwise, SIGERR = 0.0 is returned.

**8:** IFAIL — INTEGER *Input/Output*

*On entry:* IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

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

# 6 Errors and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL = 1

F11GCF has been called out of sequence. For example, the last call to F11GBF did not return IREVCM = 3 or 4.

# 7 Accuracy

Not applicable.

# 8 Further Comments

None.

# 9 Example

See the example for F11GAF.

# F11JAF – 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.

## 1 Purpose

F11JAF computes an incomplete Cholesky factorization of a real sparse symmetric matrix, represented in symmetric coordinate storage format. This factorization may be used as a preconditioner in combination with F11JCF or F11GBF.

## 2 Specification

```
 SUBROUTINE F11JAF(N, NNZ, A, LA, IROW, ICOL, LFILL, DTOL, MIC,
1                  DSCALE, PSTRAT, IPIV, ISTR, NNZC, NPIVM, IWORK,
2                  LIWORK, IFAIL)
 INTEGER           N, NNZ, LA, IROW(LA), ICOL(LA), LFILL, IPIV(N),
1                  ISTR(N+1), NNZC, NPIVM, IWORK(LIWORK), LIWORK,
2                  IFAIL
 real              A(LA), DTOL, DSCALE
 CHARACTER*1       MIC, PSTRAT
```

## 3 Description

This routine computes an incomplete Cholesky factorization [3] of a real sparse symmetric $n$ by $n$ matrix $A$. It is designed specifically for positive-definite matrices, but may also work for some mildly indefinite cases. The factorization is intended primarily for use as a preconditioner with one of the symmetric iterative solvers F11JCF or F11GBF.

The decomposition is written in the form

$$A = M + R$$

where

$$M = PLDL^T P^T$$

and $P$ is a permutation matrix, $L$ is lower triangular with unit diagonal elements, $D$ is diagonal and $R$ is a remainder matrix.

The amount of fill-in occurring in the factorization can vary from zero to complete fill, and can be controlled by specifying either the maximum level of fill LFILL, or the drop tolerance DTOL. The factorization may be modified in order to preserve row sums, and the diagonal elements may be perturbed to ensure that the preconditioner is positive-definite. Diagonal pivoting may optionally be employed, either with a user-defined ordering, or using the Markowitz strategy [2], which aims to minimize fill-in. For further details see Section 8.

The sparse matrix $A$ is represented in symmetric coordinate storage (SCS) format (see Section 2.1.1 of the Chapter Introduction). The array A stores all the non-zero elements of the lower triangular part of $A$, while arrays IROW and ICOL store the corresponding row and column indices respectively. Multiple non-zero elements may not be specified for the same row and column index.

The preconditioning matrix $M$ is returned in terms of the SCS representation of the lower triangular matrix

$$C = L + D^{-1} - I.$$

## 4 References

[1] Chan T F (1991) Fourier analysis of relaxed incomplete factorization preconditioners *SIAM J. Sci. Statist. Comput.* **12(2)** 668–680

[2] Markowitz H M (1957) The elimination form of the inverse and its application to linear programming *Management Sci.* **3** 255–269

[3] Meijerink J and van der Vorst H (1977) An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix *Math. Comput.* **31** 148–162

[4] Salvini S A and Shaw G J (1995) An evaluation of new NAG Library solvers for large sparse symmetric linear systems *NAG Technical Report TR1/95*

[5] van der Vorst H A (1990) The convergence behaviour of preconditioned CG and CG-S in the presence of rounding errors *Lecture Notes in Mathematics* (ed O Axelsson and L Y Kolotilina) **1457** Springer-Verlag

# 5 Parameters

**1:** N — INTEGER *Input*

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

*Constraint:* $\mathrm{N} \geq 1$.

**2:** NNZ — INTEGER *Input*

*On entry:* the number of non-zero elements in the lower triangular part of the matrix $A$.

*Constraint:* $1 \leq \mathrm{NNZ} \leq \mathrm{N} \times (\mathrm{N}+1)/2$.

**3:** A(LA) — ***real*** array *Input/Output*

*On entry:* the non-zero elements in the lower triangular part of the matrix $A$, ordered by increasing row index, and by increasing column index within each row. Multiple entries for the same row and column indices are not permitted. The routine F11ZBF may be used to order the elements in this way.

*On exit:* the first NNZ elements of A contain the non-zero elements of $A$ and the next NNZC elements contain the elements of the lower triangular matrix $C$. Matrix elements are ordered by increasing row index, and by increasing column index within each row.

**4:** LA — INTEGER *Input*

*On entry:* the dimension of the arrays A, IROW and ICOL as declared in the (sub)program from which F11JAF is called. These arrays must be of sufficient size to store both $A$ (NNZ elements) and $C$ (NNZC elements).

*Constraint:* $\mathrm{LA} \geq 2 \times \mathrm{NNZ}$.

**5:** IROW(LA) — INTEGER array *Input/Output*

**6:** ICOL(LA) — INTEGER array *Input/Output*

*On entry:* the row and column indices of the non-zero elements supplied in A.

*Constraints:* IROW and ICOL must satisfy the following constraints (which may be imposed by a call to F11ZBF):

$1 \leq \mathrm{IROW}(i) \leq \mathrm{N}$ and $1 \leq \mathrm{ICOL}(i) \leq \mathrm{IROW}(i)$, for $i = 1, 2, \ldots, \mathrm{NNZ}$.
$\mathrm{IROW}(i-1) < \mathrm{IROW}(i)$, or
$\mathrm{IROW}(i-1) = \mathrm{IROW}(i)$ and $\mathrm{ICOL}(i-1) < \mathrm{ICOL}(i)$, for $i = 2, 3, \ldots, \mathrm{NNZ}$.

*On exit:* the row and column indices of the non-zero elements returned in A.

**7:** LFILL — INTEGER *Input*

*On entry:* if $\mathrm{LFILL} \geq 0$ its value is the maximum level of fill allowed in the decomposition (see Section 8.2). A negative value of LFILL indicates that DTOL will be used to control the fill instead.

**8:** DTOL — ***real*** *Input*

*On entry:* if LFILL $< 0$ then DTOL is used as a drop tolerance to control the fill-in (see Section 8.2). Otherwise DTOL is not referenced.

*Constraint:* DTOL $\geq 0.0$ if LFILL $< 0$.

**9:** MIC — CHARACTER*1 *Input*

*On entry:* indicates whether or not the factorization should be modified to preserve row sums (see Section 8.3):

if MIC = 'M', the factorization is modified (MIC);
if MIC = 'N', the factorization is not modified.

*Constraint:* MIC = 'M' or 'N'.

**10:** DSCALE — ***real*** *Input*

*On entry:* the diagonal scaling parameter. All diagonal elements are multiplied by the factor (1 + DSCALE) at the start of the factorization. This can be used to ensure that the preconditioner is positive-definite. See Section 8.3.

**11:** PSTRAT — CHARACTER*1 *Input*

*On entry:* specifies the pivoting strategy to be adopted as follows:

if PSTRAT = 'N', then no pivoting is carried out;
if PSTRAT = 'M', then diagonal pivoting aimed at minimizing fill-in is carried out, using the Markowitz strategy;
if PSTRAT = 'U', then diagonal pivoting is carried out according to the user-defined input value of IPIV.

*Suggested value:* PSTRAT = 'M'.

*Constraint:* PSTRAT = 'N', 'M' or 'U'.

**12:** IPIV(N) — INTEGER array *Input/Output*

*On entry:* if PSTRAT = 'U', then IPIV($i$) must specify the row index of the diagonal element used as a pivot at elimination stage $i$. Otherwise IPIV need not be initialized.

*Constraint:* if PSTRAT = 'U', then IPIV must contain a valid permutation of the integers on [1,N].

*On exit:* the pivot indices. If IPIV($i$) = $j$ then the diagonal element in row $j$ was used as the pivot at elimination stage $i$.

**13:** ISTR(N + 1) — INTEGER array *Output*

*On exit:* ISTR($i$), for $i = 1, 2, \ldots,$ N holds the starting address in the arrays A, IROW and ICOL of row $i$ of the matrix $C$. ISTR(N + 1) holds the address of the last non-zero element in $C$ plus one.

**14:** NNZC — INTEGER *Output*

*On exit:* the number of non-zero elements in the lower triangular matrix $C$.

**15:** NPIVM — INTEGER *Output*

*On exit:* the number of pivots which were modified during the factorization to ensure that $M$ was positive-definite. The quality of the preconditioner will generally depend on the returned value of NPIVM. If NPIVM is large the preconditioner may not be satisfactory. In this case it may be advantageous to call F11JAF again with an increased value of either LFILL or DSCALE. See also Section 8.4.

**16:** IWORK(LIWORK) — INTEGER array *Workspace*

**17:** LIWORK — INTEGER *Input*

*On entry:* the dimension of the array IWORK as declared in the (sub)program from which F11JAF is called.

*Constraints:* the minimum permissible value of LIWORK depends on LFILL as follows:

$\text{LIWORK} \geq 2 \times \text{LA} - 3 \times \text{NNZ} + 7 \times \text{N} + 1$, for $\text{LFILL} \geq 0$, or
$\text{LIWORK} \geq \text{LA} - \text{NNZ} + 7 \times \text{N} + 1$, for $\text{LFILL} < 0$.

**18:** IFAIL — INTEGER *Input/Output*

*On entry:* IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

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

# 6 Errors and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL = 1

On entry, $\text{N} < 1$,
or $\text{NNZ} < 1$,
or $\text{NNZ} > \text{N} \times (\text{N}+1)/2$,
or $\text{LA} < 2 \times \text{NNZ}$,
or $\text{DTOL} < 0.0$,
or MIC $\neq$ 'M' or 'N',
or PSTRAT $\neq$ 'N' 'M' or 'U',
or $\text{LIWORK} < 2 \times \text{LA} - 3 \times \text{NNZ} + 7 \times \text{N} + 1$, and $\text{LFILL} \geq 0$,
or $\text{LIWORK} < \text{LA} - \text{NNZ} + 7 \times \text{N} + 1$, and $\text{LFILL} < 0$.

IFAIL = 2

On entry, the arrays IROW and ICOL fail to satisfy the following constraints:

$1 \leq \text{IROW}(i) \leq \text{N}$ and $1 \leq \text{ICOL}(i) \leq \text{IROW}(i)$, for $i = 1, 2, \ldots, \text{NNZ}$.
$\text{IROW}(i-1) < \text{IROW}(i)$, or
$\text{IROW}(i-1) = \text{IROW}(i)$ and $\text{ICOL}(i-1) < \text{ICOL}(i)$, for $i = 2, 3, \ldots, \text{NNZ}$.

Therefore a non-zero element has been supplied which does not lie in the lower triangular part of $A$, is out of order, or has duplicate row and column indices. Call F11ZBF to reorder and sum or remove duplicates.

IFAIL = 3

On entry, PSTRAT = 'U', but IPIV does not represent a valid permutation of the integers in $[1, \text{N}]$. An input value of IPIV is either out of range or repeated.

IFAIL = 4

LA is too small, resulting in insufficient storage space for fill-in elements. The decomposition has been terminated before completion. Either increase LA or reduce the amount of fill by setting PSTRAT = 'M', reducing LFILL, or increasing DTOL.

IFAIL = 5

A serious error has occurred in an internal call to F11ZBF. Check all subroutine calls and array sizes. Seek expert help.

# 7 Accuracy

The accuracy of the factorization will be determined by the size of the elements that are dropped and the size of any modifications made to the diagonal elements. If these sizes are small then the computed factors will correspond to a matrix close to $A$. The factorization can generally be made more accurate by increasing LFILL, or by reducing DTOL with LFILL $< 0$.

If F11JAF is used in combination with F11GBF, or F11JCF, the more accurate the factorization the fewer iterations will be required. However, the cost of the decomposition will also generally increase.

# 8 Further Comments

## 8.1 Timing

The time taken for a call to F11JAF is roughly proportional to NNZC$^2$/N.

## 8.2 Control of Fill-in

If LFILL $\geq 0$ the amount of fill-in occurring in the incomplete factorization is controlled by limiting the maximum **level** of fill-in to LFILL. The original non-zero elements of $A$ are defined to be of level 0. The fill level of a new non-zero location occurring during the factorization is defined as:

$$k = \max(k_e, k_c) + 1,$$

where $k_e$ is the level of fill of the element being eliminated, and $k_c$ is the level of fill of the element causing the fill-in.

If LFILL $< 0$ the fill-in is controlled by means of the **drop tolerance** DTOL. A potential fill-in element $a_{ij}$ occurring in row $i$ and column $j$ will not be included if:

$$|a_{ij}| < \text{DTOL} \times \sqrt{|a_{ii}a_{jj}|}.$$

For either method of control, any elements which are not included are discarded if MIC = 'N', or subtracted from the diagonal element in the elimination row if MIC = 'M'.

## 8.3 Choice of Parameters

There is unfortunately no choice of the various algorithmic parameters which is optimal for all types of symmetric matrix, and some experimentation will generally be required for each new type of matrix encountered.

If the matrix $A$ is not known to have any particular special properties the following strategy is recommended. Start with LFILL = 0, MIC = 'N' and DSCALE = 0.0. If the value returned for NPIVM is significantly larger than zero, i.e., a large number of pivot modifications were required to ensure that $M$ was positive-definite, the preconditioner is not likely to be satisfactory. In this case increase either LFILL or DSCALE until NPIVM falls to a value close to zero. Once suitable values of LFILL and DSCALE have been found try setting MIC = 'M' to see if any improvement can be obtained by using **modified** incomplete Cholesky.

F11JAF is primarily designed for positive-definite matrices, but may work for some mildly indefinite problems. If NPIVM cannot be satisfactorily reduced by increasing LFILL or DSCALE then $A$ is probably too indefinite for this routine.

If $A$ has non-positive off-diagonal elements, is non-singular, and has only non-negative elements in its inverse, it is called an 'M-matrix'. It can be shown that no pivot modifications are required in the incomplete Cholesky factorization of an M-matrix [3]. In this case a good preconditioner can generally be expected by setting LFILL = 0, MIC = 'M' and DSCALE = 0.0.

For certain mesh-based problems involving M-matrices it can be shown in theory that setting MIC = 'M', and choosing DSCALE appropriately can reduce the order of magnitude of the condition number of the preconditioned matrix as a function of the mesh steplength [1]. In practise this property often holds even with DSCALE = 0.0, although an improvement in condition can result from increasing DSCALE slightly [5].

Some illustrations of the application of F11JAF to linear systems arising from the discretization of two-dimensional elliptic partial differential equations, and to random-valued randomly structured symmetric positive-definite linear systems, can be found in [4].

### 8.4 Direct Solution of Positive-Definite Systems

Although it is not their primary purpose F11JAF and F11JBF may be used together to obtain a **direct** solution to a symmetric positive-definite linear system. To achieve this the call to F11JBF should be preceded by a **complete** Cholesky factorization

$$A = PLDL^T P^T = M.$$

A complete factorization is obtained from a call to F11JAF with LFILL $< 0$ and DTOL $= 0.0$, provided NPIVM $= 0$ on exit. A non-zero value of NPIVM indicates that $A$ is not positive-definite, or is ill-conditioned. A factorization with non-zero NPIVM may serve as a preconditioner, but will not result in a direct solution. It is therefore **essential** to check the output value of NPIVM if a direct solution is required.

The use of F11JAF and F11JBF as a direct method is illustrated in Section 9 of the routine document for F11JBF.

## 9 Example

This example program reads in a symmetric sparse matrix $A$ and calls F11JAF to compute an incomplete Cholesky factorization. It then outputs the non-zero elements of both $A$ and $C = L + D^{-1} - I$.

The call to F11JAF has LFILL $= 0$, MIC $=$ 'N', DSCALE $= 0.0$ and PSTRAT $=$ 'M', giving an unmodified zero-fill factorization of an unperturbed matrix, with Markowitz diagonal pivoting.

### 9.1 Example 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.

```
*     F11JAF Example Program Text
*     Mark 17 Release. NAG Copyright 1995.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LA, LIWORK
      PARAMETER        (NMAX=1000,LA=10000,LIWORK=2*LA+7*NMAX+1)
*     .. Local Scalars ..
      real             DSCALE, DTOL
      INTEGER          I, IFAIL, LFILL, N, NNZ, NNZC, NPIVM
      CHARACTER        MIC, PSTRAT
*     .. Local Arrays ..
      real             A(LA)
      INTEGER          ICOL(LA), IPIV(NMAX), IROW(LA), ISTR(NMAX+1),
     +                 IWORK(LIWORK)
*     .. External Subroutines ..
      EXTERNAL         F11JAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11JAF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
*
*     Read algorithmic parameters
*
      READ (NIN,*) N
```

```
      IF (N.LE.NMAX) THEN
         READ (NIN,*) NNZ
         READ (NIN,*) LFILL, DTOL
         READ (NIN,*) MIC, DSCALE
         READ (NIN,*) PSTRAT
*
*     Read the matrix A
*
         DO 20 I = 1, NNZ
            READ (NIN,*) A(I), IROW(I), ICOL(I)
   20    CONTINUE
*
*     Calculate incomplete Cholesky factorization
*
         IFAIL = 0
         CALL F11JAF(N,NNZ,A,LA,IROW,ICOL,LFILL,DTOL,MIC,DSCALE,PSTRAT,
     +               IPIV,ISTR,NNZC,NPIVM,IWORK,LIWORK,IFAIL)
*
*     Output original matrix
*
         WRITE (NOUT,*) ' Original Matrix'
         WRITE (NOUT,*) ' N     =', N
         WRITE (NOUT,*) ' NNZ   =', NNZ
         DO 40 I = 1, NNZ
            WRITE (NOUT,99999) I, A(I), IROW(I), ICOL(I)
   40    CONTINUE
         WRITE (NOUT,*)
*
*     Output details of the factorization
*
         WRITE (NOUT,*) ' Factorization'
         WRITE (NOUT,*) ' N     =', N
         WRITE (NOUT,*) ' NNZ   =', NNZC
         WRITE (NOUT,*) ' NPIVM =', NPIVM
         DO 60 I = NNZ + 1, NNZ + NNZC
            WRITE (NOUT,99999) I, A(I), IROW(I), ICOL(I)
   60    CONTINUE
         WRITE (NOUT,*)
*
         WRITE (NOUT,*) '      I     IPIV(I)'
         DO 80 I = 1, N
            WRITE (NOUT,99998) I, IPIV(I)
   80    CONTINUE
*
      END IF
      STOP
*
99999 FORMAT (1X,I8,e16.4,2I8)
99998 FORMAT (1X,2I8)
      END
```

## 9.2 Example Data

```
F11JAF Example Program Data
  7                  N
 16                  NNZ
  0 0.0              LFILL, DTOL
  'N' 0.0            MIC, DSCALE
```

```
 'M'                 PSTRAT
 4.   1    1
 1.   2    1
 5.   2    2
 2.   3    3
 2.   4    2
 3.   4    4
-1.   5    1
 1.   5    4
 4.   5    5
 1.   6    2
-2.   6    5
 3.   6    6
 2.   7    1
-1.   7    2
-2.   7    3
 5.   7    7        A(I), IROW(I), ICOL(I), I=1,...,NNZ
```

## 9.3 Example Results

```
F11JAF Example Program Results
 Original Matrix
 N     =           7
 NNZ   =          16
       1      0.4000E+01       1       1
       2      0.1000E+01       2       1
       3      0.5000E+01       2       2
       4      0.2000E+01       3       3
       5      0.2000E+01       4       2
       6      0.3000E+01       4       4
       7     -0.1000E+01       5       1
       8      0.1000E+01       5       4
       9      0.4000E+01       5       5
      10      0.1000E+01       6       2
      11     -0.2000E+01       6       5
      12      0.3000E+01       6       6
      13      0.2000E+01       7       1
      14     -0.1000E+01       7       2
      15     -0.2000E+01       7       3
      16      0.5000E+01       7       7

 Factorization
 N     =           7
 NNZ   =          16
 NPIVM =           0
      17      0.5000E+00       1       1
      18      0.3333E+00       2       2
      19      0.3333E+00       3       2
      20      0.2727E+00       3       3
      21     -0.5455E+00       4       3
      22      0.5238E+00       4       4
      23     -0.2727E+00       5       3
      24      0.2683E+00       5       5
      25      0.6667E+00       6       2
      26      0.5238E+00       6       4
      27      0.2683E+00       6       5
      28      0.3479E+00       6       6
      29     -0.1000E+01       7       1
```

```
 30      0.5366E+00      7       5
 31     -0.5345E+00      7       6
 32      0.9046E+00      7       7

 I     IPIV(I)
 1        3
 2        4
 3        5
 4        6
 5        1
 6        2
 7        7
```

# F11JBF – 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.

## 1 Purpose

F11JBF solves a system of linear equations involving the incomplete Cholesky preconditioning matrix generated by F11JAF.

## 2 Specification

```
 SUBROUTINE F11JBF(N, A, LA, IROW, ICOL, IPIV, ISTR, CHECK, Y, X,
1                  IFAIL)
 INTEGER           N, LA, IROW(LA), ICOL(LA), IPIV(N), ISTR(N+1),
1                  IFAIL
 real              A(LA), Y(N), X(N)
 CHARACTER*1       CHECK
```

## 3 Description

This routine solves a system of linear equations

$$Mx = y$$

involving the preconditioning matrix $M = PLDL^TP^T$, corresponding to an incomplete Cholesky decomposition of a sparse symmetric matrix stored in symmetric coordinate storage (SCS) format (see Section 2.1.1 of the Chapter Introduction), as generated by F11JAF.

In the above decomposition $L$ is a lower triangular sparse matrix with unit diagonal, $D$ is a diagonal matrix and $P$ is a permutation matrix. $L$ and $D$ are supplied to F11JBF through the matrix

$$C = L + D^{-1} - I$$

which is a lower triangular N by N sparse matrix, stored in SCS format, as returned by F11JAF. The permutation matrix $P$ is returned from F11JAF via the array IPIV.

It is envisaged that a common use of F11JBF will be to carry out the preconditioning step required in the application of F11GBF to sparse symmetric linear systems. F11JBF is used for this purpose by the black-box routine F11JCF.

F11JBF may also be used in combination with F11JAF to solve a sparse symmetric positive-definite system of linear equations directly (see Section 8.4 of the routine document for F11JAF). This use of F11JBF is demonstrated in Section 9.

## 4 References

None.

## 5 Parameters

**1:** N — INTEGER *Input*

*On entry:* $n$, the order of the matrix $M$. This **must** be the same value as was supplied in the preceeding call to F11JAF.

*Constraint:* $\mathrm{N} \geq 1$.

**2:** A(LA) — ***real*** array *Input*

*On entry:* the values returned in array A by a previous call to F11JAF.

**3:** LA — INTEGER *Input*

*On entry:* the dimension of the arrays A, IROW and ICOL as declared in the (sub)program from which F11JBF is called. This **must** be the same value as was supplied in the preceeding call to F11JAF.

**4:** IROW(LA) — INTEGER array *Input*

**5:** ICOL(LA) — INTEGER array *Input*

**6:** IPIV(N) — INTEGER array *Input*

**7:** ISTR(N+1) — INTEGER array *Input*

*On entry:* the values returned in arrays IROW, ICOL, IPIV and ISTR by a previous call to F11JAF.

**8:** CHECK — CHARACTER*1 *Input*

*On entry:* specifies whether or not the input data should be checked:

if CHECK = 'C', checks are carried out on the values of N, IROW, ICOL, IPIV and ISTR;
if CHECK = 'N', none of these checks are carried out.

See also Section 8.2.

*Constraint:* CHECK = 'C' or 'N'.

**9:** Y(N) — ***real*** array *Input*

*On entry:* the right-hand side vector $y$.

**10:** X(N) — ***real*** array *Output*

*On exit:* the solution vector $x$.

**11:** IFAIL — INTEGER *Input/Output*

*On entry:* IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

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

# 6 Errors and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL = 1

On entry, CHECK ≠ 'C' or 'N'.

IFAIL = 2

On entry, N < 1.

IFAIL = 3

On entry, the SCS representation of the preconditioning matrix $M$ is invalid. Further details are given in the error message. Check that the call to F11JBF has been preceeded by a valid call to F11JAF and that the arrays A, IROW, ICOL, IPIV and ISTR have not been corrupted between the two calls.

# 7 Accuracy

The computed solution $x$ is the exact solution of a perturbed system of equations $(M + \delta M)x = y$, where

$$|\delta M| \leq c(n)\epsilon P|L||D||L^T|P^T,$$

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

# 8 Further Comments

## 8.1 Timing

The time taken for a call to F11JBF is proportional to the value of NNZC returned from F11JAF.

## 8.2 Use of CHECK

It is expected that a common use of F11JBF will be to carry out the preconditioning step required in the application of F11GBF to sparse symmetric linear systems. In this situation F11JBF is likely to be called many times with the same matrix $M$. In the interests of both reliability and efficiency, you are recommended to set CHECK to 'C' for the first of such calls, and to 'N' for all subsequent calls.

# 9 Example

This example program reads in a symmetric positive-definite sparse matrix $A$ and a vector $y$. It then calls F11JAF, with LFILL = $-1$ and DTOL = 0.0, to compute the **complete** Cholesky decomposition of $A$:

$$A = PLDL^T P^T.$$

Finally it calls F11JBF to solve the system

$$PLDL^T P^T x = y.$$

## 9.1 Example 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.

```
*     F11JBF Example Program Text
*     Mark 17 Release. NAG Copyright 1995.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LA, LIWORK
      PARAMETER        (NMAX=1000,LA=10000,LIWORK=2*LA+7*NMAX+1)
*     .. Local Scalars ..
      real             DSCALE, DTOL
      INTEGER          I, IFAIL, LFILL, N, NNZ, NNZC, NPIVM
      CHARACTER        CHECK, MIC, PSTRAT
*     ... Local Arrays ..
      real             A(LA), X(NMAX), Y(NMAX)
      INTEGER          ICOL(LA), IPIV(NMAX), IROW(LA), ISTR(NMAX+1),
     +                 IWORK(LIWORK)
*     .. External Subroutines ..
      EXTERNAL         F11JAF, F11JBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11JBF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
*
*     Read order of matrix and number of non-zero entries
*
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
         READ (NIN,*) NNZ
*
*     Read the matrix A
*
```

```
         DO 20 I = 1, NNZ
            READ (NIN,*) A(I), IROW(I), ICOL(I)
   20    CONTINUE
*
*     Read the vector y
*
         READ (NIN,*) (Y(I),I=1,N)
*
*     Calculate Cholesky factorization
*
         LFILL = -1
         DTOL = 0.0e0
         MIC = 'N'
         DSCALE = 0.0e0
         PSTRAT = 'M'
         IFAIL = 0
*
         CALL F11JAF(N,NNZ,A,LA,IROW,ICOL,LFILL,DTOL,MIC,DSCALE,PSTRAT,
     +               IPIV,ISTR,NNZC,NPIVM,IWORK,LIWORK,IFAIL)
*
*     Check the output value of NPIVM
*
         IF (NPIVM.NE.0) THEN
*
            WRITE (NOUT,*) 'Factorization is not complete'
*
         ELSE
*                    T T
*     Solve P L D L P x = y
*
            CHECK = 'C'
*
            CALL F11JBF(N,A,LA,IROW,ICOL,IPIV,ISTR,CHECK,Y,X,IFAIL)
*
*     Output results
*
            WRITE (NOUT,*) ' Solution of linear system'
            DO 40 I = 1, N
               WRITE (NOUT,99999) X(I)
   40       CONTINUE
*
         END IF
      END IF
      STOP
*
99999 FORMAT (1X,e16.4)
      END
```

## 9.2 Example Data

```
F11JBF Example Program Data
  9                    N
 23                    NNZ
  4.   1    1
 -1.   2    1
  6.   2    2
  1.   3    2
  2.   3    3
```

```
 3.   4    4
 2.   5    1
 4.   5    5
 1.   6    3
 2.   6    4
 6.   6    6
-4.   7    2
 1.   7    5
-1.   7    6
 6.   7    7
-1.   8    4
-1.   8    6
 3.   8    8
 1.   9    1
 1.   9    5
-1.   9    6
 1.   9    8
 4.   9    9          A(I), IROW(I), ICOL(I), I=1,...,NNZ
 4.10 -2.94  1.41
 2.53  4.35  1.29
 5.01  0.52  4.57     Y(I), I=1,...,N
```

## 9.3 Example Results

```
F11JBF Example Program Results
 Solution of linear system
      0.7000E+00
      0.1600E+00
      0.5200E+00
      0.7700E+00
      0.2800E+00
      0.2100E+00
      0.9300E+00
      0.2000E+00
      0.9000E+00
```

# F11JCF – 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.

## 1 Purpose

F11JCF solves a real sparse symmetric system of linear equations, represented in symmetric coordinate storage format, using a conjugate gradient or Lanczos method, with incomplete Cholesky preconditioning.

## 2 Specification

```
      SUBROUTINE F11JCF(METHOD, N, NNZ, A, LA, IROW, ICOL, IPIV, ISTR,
     1                  B, TOL, MAXITN, X, RNORM, ITN, WORK, LWORK,
     2                  IFAIL)
      INTEGER           N, NNZ, LA, IROW(LA), ICOL(LA), IPIV(N),
     1                  ISTR(N+1), MAXITN, ITN, LWORK, IFAIL
      real              A(LA), B(N), TOL, X(N), RNORM, WORK(LWORK)
      CHARACTER*(*)     METHOD
```

## 3 Description

This routine solves a real sparse symmetric linear system of equations:

$$Ax = b,$$

using a preconditioned conjugate gradient method [2], or a preconditioned Lanczos method based on the algorithm SYMMLQ [3]. The conjugate gradient method is more efficient if $A$ is positive-definite, but may fail to converge for indefinite matrices. In this case the Lanczos method should be used instead. For further details see [1].

F11JCF uses the incomplete Cholesky factorization determined by F11JAF as the preconditioning matrix. A call to F11JCF must always be preceeded by a call to F11JAF. Alternative preconditioners for the same storage scheme are available by calling F11JEF.

The matrix $A$, and the preconditioning matrix $M$, are represented in symmetric coordinate storage (SCS) format (see Section 2.1.1 of the Chapter Introduction) in the arrays A, IROW and ICOL, as returned from F11JAF. The array A holds the non-zero entries in the lower triangular parts of these matrices, while IROW and ICOL hold the corresponding row and column indices.

## 4 References


[1] Barrett R, Berry M, Chan T F, Demmel J, Donato J, Dongarra J, Eijkhout V, Pozo R, Romine C and van der Vorst H (1994) *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods* SIAM, Philadelphia

[2] Meijerink J and van der Vorst H (1977) An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix *Math. Comput.* **31** 148–162

[3] Paige C C and Saunders M A (1975) Solution of sparse indefinite systems of linear equations *SIAM J. Numer. Anal.* **12** 617–629

[4] Salvini S A and Shaw G J (1995) An evaluation of new NAG Library solvers for large sparse symmetric linear systems *NAG Technical Report TR1/95*

# 5 Parameters

**1:** METHOD — CHARACTER*(*) *Input*

*On entry:* specifies the iterative method to be used. The possible choices are:

'CG' Conjugate Gradient method;
'SYMMLQ' Lanczos method (SYMMLQ).

*Constraint:* METHOD = 'CG' or 'SYMMLQ'.

**2:** N — INTEGER *Input*

*On entry:* $n$, the order of the matrix $A$. This **must** be the same value as was supplied in the preceeding call to F11JAF.

*Constraint:* N $\geq$ 1.

**3:** NNZ — INTEGER *Input*

*On entry:* the number of non-zero elements in the lower triangular part of the matrix $A$. This **must** be the same value as was supplied in the preceeding call to F11JAF.

*Constraint:* 1 $\leq$ NNZ $\leq$ N $\times$ (N + 1)/2.

**4:** A(LA) — ***real*** array *Input*

*On entry:* the values returned in array A by a previous call to F11JAF.

**5:** LA — INTEGER *Input*

*On entry:* the dimension of the arrays A, IROW and ICOL as declared in the (sub)program from which F11JCF is called. This **must** be the same value as was supplied in the preceeding call to F11JAF.

*Constraint:* LA $\geq$ 2 $\times$ NNZ.

**6:** IROW(LA) — INTEGER array *Input*

**7:** ICOL(LA) — INTEGER array *Input*

**8:** IPIV(N) — INTEGER array *Input*

**9:** ISTR(N+1) — INTEGER array *Input*

*On entry:* the values returned in arrays IROW, ICOL, IPIV and ISTR by a previous call to F11JAF.

**10:** B(N) — ***real*** array *Input*

*On entry:* the right-hand side vector $b$.

**11:** TOL — ***real*** *Input*

*On entry:* the required tolerance. Let $x_k$ denote the approximate solution at iteration $k$, and $r_k$ the corresponding residual. The algorithm is considered to have converged at iteration $k$ if:

$$\|r_k\|_\infty \leq \tau \times (\|b\|_\infty + \|A\|_\infty \|x_k\|_\infty).$$

If TOL $\leq$ 0.0, $\tau = \max(\sqrt{\epsilon}, \sqrt{n}\,\epsilon)$ is used, where $\epsilon$ is the ***machine precision***. Otherwise $\tau = \max(\text{TOL}, 10\epsilon, \sqrt{n}\,\epsilon)$ is used.

*Constraint:* TOL < 1.0

**12:** MAXITN — INTEGER *Input*

*On entry:* the maximum number of iterations allowed.

*Constraint:* MAXITN $\geq$ 1.

**13:** X(N) — ***real*** array *Input/Output*

*On entry:* an initial approximation to the solution vector $x$.

*On exit:* an improved approximation to the solution vector $x$.

**14:** RNORM — ***real*** *Output*

*On exit:* the final value of the residual norm $\|r_k\|_\infty$, where $k$ is the output value of ITN.

**15:** ITN — INTEGER *Output*

*On exit:* the number of iterations carried out.

**16:** WORK(LWORK) — ***real*** array *Workspace*

**17:** LWORK — INTEGER *Input*

*On entry:* the dimension of the array WORK as declared in the (sub)program from which F11JCF is called.

*Constraints:*

if METHOD = 'CG' then LWORK $\geq 6 \times$ N;
if METHOD = 'SYMMLQ' then LWORK $\geq 7 \times$ N.

**18:** IFAIL — INTEGER *Input/Output*

*On entry:* IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

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

# 6 Errors and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL = 1

On entry, METHOD ≠ 'CG' or 'SYMMLQ',
or N < 1,
or NNZ < 1,
or NNZ > N × (N + 1)/2,
or LA too small,
or TOL ≥ 1.0,
or MAXITN < 1,
or LWORK too small.

IFAIL = 2

On entry, the SCS representation of $A$ is invalid. Further details are given in the error message. Check that the call to F11JCF has been preceeded by a valid call to F11JAF, and that the arrays A, IROW, and ICOL have not been corrupted between the two calls.

IFAIL = 3

On entry, the SCS representation of the preconditioning matrix $M$ is invalid. Further details are given in the error message. Check that the call to F11JCF has been preceeded by a valid call to F11JAF, and that the arrays A, IROW, ICOL, IPIV and ISTR have not been corrupted between the two calls.

IFAIL = 4

The required accuracy could not be obtained. However, a reasonable accuracy has been obtained and further iterations could not improve the result.

IFAIL = 5

Required accuracy not obtained in MAXITN iterations.

IFAIL = 6

The preconditioner appears not to be positive-definite.

IFAIL = 7

The matrix of the coefficients appears not to be positive-definite (conjugate gradient method only).

IFAIL = 8

A serious error has occurred in an internal call to F11GAF, F11GBF or F11GCF. Check all subroutine calls and array sizes. Seek expert help.

# 7 Accuracy

On successful termination, the final residual $r_k = b - Ax_k$, where $k$ = ITN, satisfies the termination criterion

$$\|r_k\|_\infty \leq \tau \times (\|b\|_\infty + \|A\|_\infty \|x_k\|_\infty).$$

The value of the final residual norm is returned in RNORM.

# 8 Further Comments

The time taken by F11JCF for each iteration is roughly proportional to the value of NNZC returned from the preceeding call to F11JAF. One iteration with the Lanczos method (SYMMLQ) requires a slightly larger number of operations than one iteration with the conjugate gradient method.

The number of iterations required to achieve a prescribed accuracy cannot be easily determined a priori, as it can depend dramatically on the conditioning and spectrum of the preconditioned matrix of the coefficients $\bar{A} = M^{-1}A$.

Some illustrations of the application of F11JCF to linear systems arising from the discretization of two-dimensional elliptic partial differential equations, and to random-valued randomly structured symmetric positive-definite linear systems, can be found in [4].

# 9 Example

This example program solves a symmetric positive-definite system of equations using the conjugate gradient method, with incomplete Cholesky preconditioning.

## 9.1 Example 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.

```
*     F11JCF Example Program Text
*     Mark 17 Release. NAG Copyright 1995.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LA, LIWORK, LWORK
      PARAMETER        (NMAX=1000,LA=10000,LIWORK=2*LA+7*NMAX+1,
     +                 LWORK=6*NMAX)
```

```
*     .. Local Scalars ..
      real             DSCALE, DTOL, RNORM, TOL
      INTEGER          I, IFAIL, ITN, LFILL, MAXITN, N, NNZ, NNZC, NPIVM
      CHARACTER        MIC, PSTRAT
      CHARACTER*6      METHOD
*     .. Local Arrays ..
      real             A(LA), B(NMAX), WORK(LWORK), X(NMAX)
      INTEGER          ICOL(LA), IPIV(NMAX), IROW(LA), ISTR(NMAX+1),
     +                 IWORK(LIWORK)
*     .. External Subroutines ..
      EXTERNAL         F11JAF, F11JCF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11JCF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
*
*     Read algorithmic parameters
*
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
         READ (NIN,*) NNZ
         READ (NIN,*) METHOD
         READ (NIN,*) LFILL, DTOL
         READ (NIN,*) MIC, DSCALE
         READ (NIN,*) PSTRAT
         READ (NIN,*) TOL, MAXITN
*
*     Read the matrix A
*
         DO 20 I = 1, NNZ
            READ (NIN,*) A(I), IROW(I), ICOL(I)
   20    CONTINUE
*
*     Read right-hand side vector b and initial approximate solution x
*
         READ (NIN,*) (B(I),I=1,N)
         READ (NIN,*) (X(I),I=1,N)
*
*     Calculate incomplete Cholesky factorization
*
         IFAIL = 0
         CALL F11JAF(N,NNZ,A,LA,IROW,ICOL,LFILL,DTOL,MIC,DSCALE,PSTRAT,
     +               IPIV,ISTR,NNZC,NPIVM,IWORK,LIWORK,IFAIL)
*
*     Solve Ax = b using F11JCF
*
         CALL F11JCF(METHOD,N,NNZ,A,LA,IROW,ICOL,IPIV,ISTR,B,TOL,MAXITN,
     +               X,RNORM,ITN,WORK,LWORK,IFAIL)
*
         WRITE (NOUT,99999) 'Converged in', ITN, ' iterations'
         WRITE (NOUT,99998) 'Final residual norm =', RNORM
*
*     Output x
*
         DO 40 I = 1, N
            WRITE (NOUT,99997) X(I)
   40    CONTINUE
      END IF
```

```
      STOP
*
99999 FORMAT (1X,A,I10,A)
99998 FORMAT (1X,A,1P,e16.3)
99997 FORMAT (1X,1P,e16.4)
      END
```

## 9.2 Example Data

```
F11JCF Example Program Data
  7                    N
  16                   NNZ
 'CG'                  METHOD
  1 0.0                LFILL, DTOL
  'N' 0.0              MIC, DSCALE
  'M'                  PSTRAT
  1.0E-6 100           TOL, MAXITN
  4.    1    1
  1.    2    1
  5.    2    2
  2.    3    3
  2.    4    2
  3.    4    4
 -1.    5    1
  1.    5    4
  4.    5    5
  1.    6    2
 -2.    6    5
  3.    6    6
  2.    7    1
 -1.    7    2
 -2.    7    3
  5.    7    7          A(I), IROW(I), ICOL(I), I=1,...,NNZ
 15.   18.   -8.   21.
 11.   10.   29.         B(I), I=1,...,N
  0.    0.    0.    0.
  0.    0.    0.         X(I), I=1,...,N
```

## 9.3 Example Results

```
F11JCF Example Program Results
Converged in          1 iterations
Final residual norm =       7.105E-15
      1.0000E+00
      2.0000E+00
      3.0000E+00
      4.0000E+00
      5.0000E+00
      6.0000E+00
      7.0000E+00
```

# F11JDF – 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.

## 1 Purpose

F11JDF solves a system of linear equations involving the preconditioning matrix corresponding to SSOR applied to a real sparse symmetric matrix, represented in symmetric coordinate storage format.

## 2 Specification

```
SUBROUTINE F11JDF(N, NNZ, A, IROW, ICOL, RDIAG, OMEGA, CHECK, Y,
1                 X, IWORK, IFAIL)
INTEGER           N, NNZ, IROW(NNZ), ICOL(NNZ), IWORK(N+1), IFAIL
real              A(NNZ), RDIAG(N), OMEGA, Y(N), X(N)
CHARACTER*1       CHECK
```

## 3 Description

This routine solves a system of equations:

$$Mx = y$$

involving the preconditioning matrix:

$$M = \frac{1}{\omega(2-\omega)}(D+\omega L)D^{-1}(D+\omega L)^T$$

corresponding to symmetric successive-over-relaxation (SSOR) [1] on a linear system $Ax = b$, where $A$ is a sparse symmetric matrix stored in symmetric coordinate storage (SCS) format (see Section 2.1.1 of the Chapter Introduction).

In the definition of $M$ given above $D$ is the diagonal part of $A$, $L$ is the strictly lower triangular part of $A$, and $\omega$ is a user-defined relaxation parameter.

It is envisaged that a common use of F11JDF will be to carry out the preconditioning step required in the application of F11GBF to sparse linear systems. For an illustration of this use of F11JDF see the example program given in Section 9.1. F11JDF is also used for this purpose by the black-box routine F11JEF.

## 4 References

[1] Young D (1971) *Iterative Solution of Large Linear Systems* Academic Press, New York

## 5 Parameters

**1:** N — INTEGER *Input*

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

*Constraint:* $\mathrm{N} \geq 1$.

**2:** NNZ — INTEGER *Input*

*On entry:* the number of non-zero elements in the lower triangular part of $A$.

*Constraint:* $1 \leq \mathrm{NNZ} \leq \mathrm{N} \times (\mathrm{N}+1)/2$.

**3:** A(NNZ) — ***real*** array *Input*

*On entry:* the non-zero elements in the lower triangular part of the matrix $A$, ordered by increasing row index, and by increasing column index within each row. Multiple entries for the same row and column indices are not permitted. The routine F11ZBF may be used to order the elements in this way.

**4:** IROW(NNZ) — INTEGER array *Input*

**5:** ICOL(NNZ) — INTEGER array *Input*

*On entry:* the row and column indices of the non-zero elements supplied in A.

*Constraints:* IROW and ICOL must satisfy the following constraints (which may be imposed by a call to F11ZBF):

$1 \leq \text{IROW}(i) \leq \text{N}$, $1 \leq \text{ICOL}(i) \leq \text{IROW}(i)$, for $i = 1, 2, \ldots, \text{NNZ}$.
$\text{IROW}(i-1) < \text{IROW}(i)$, or
$\text{IROW}(i-1) = \text{IROW}(i)$ and $\text{ICOL}(i-1) < \text{ICOL}(i)$, for $i = 2, 3, \ldots, \text{NNZ}$.

**6:** RDIAG(N) — ***real*** array *Input*

*On entry:* the elements of the diagonal matrix $D^{-1}$, where $D$ is the diagonal part of $A$.

**7:** OMEGA — ***real*** *Input*

*On entry:* the relaxation parameter $\omega$.

*Constraint:* $0.0 \leq \text{OMEGA} \leq 2.0$.

**8:** CHECK — CHARACTER*1 *Input*

*On entry:* specifies whether or not the input data should be checked:

if CHECK = 'C', checks are carried out on the values of N, NNZ, IROW, ICOL and OMEGA;
if CHECK = 'N', none of these checks are carried out.

See also Section 8.2.

*Constraint:* CHECK = 'C' or 'N'.

**9:** Y(N) — ***real*** array *Input*

*On entry:* the right-hand side vector $y$.

**10:** X(N) — ***real*** array *Output*

*On exit:* the solution vector $x$.

**11:** IWORK(N+1) — INTEGER array *Workspace*

**12:** IFAIL — INTEGER *Input/Output*

*On entry:* IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

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

# 6 Errors and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL = 1

On entry, CHECK ≠ 'C' or 'N'.

IFAIL = 2

On entry, N < 1,
or NNZ < 1,
or NNZ > N × (N + 1)/2,
or OMEGA lies outside the interval [0.0, 2.0],

IFAIL = 3

On entry, the arrays IROW and ICOL fail to satisfy the following constraints:

$1 \leq \mathrm{IROW}(i) \leq \mathrm{N}$ and $1 \leq \mathrm{ICOL}(i) \leq \mathrm{IROW}(i)$, for $i = 1, 2, \ldots, \mathrm{NNZ}$.
$\mathrm{IROW}(i-1) < \mathrm{IROW}(i)$, or
$\mathrm{IROW}(i-1) = \mathrm{IROW}(i)$ and $\mathrm{ICOL}(i-1) < \mathrm{ICOL}(i)$, for $i = 2, 3, \ldots, \mathrm{NNZ}$.

Therefore a non-zero element has been supplied which does not lie in the lower triangular part of $A$, is out of order, or has duplicate row and column indices. Call F11ZBF to reorder and sum or remove duplicates.

# 7 Accuracy

The computed solution $x$ is the exact solution of a perturbed system of equations $(M + \delta M)x = y$, where

$$|\delta M| \leq c(n)\epsilon |D + \omega L||D^{-1}||(D + \omega L)^T|,$$

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

# 8 Further Comments

## 8.1 Timing

The time taken for a call to F11JDF is proportional to NNZ.

## 8.2 Use of CHECK

It is expected that a common use of F11JDF will be to carry out the preconditioning step required in the application of F11GBF to sparse symmetric linear systems. In this situation F11JDF is likely to be called many times with the same matrix $M$. In the interests of both reliability and efficiency, you are recommended to set CHECK to 'C' for the first of such calls, and to 'N' for all subsequent calls.

# 9 Example

This example program solves a sparse symmetric linear system of equations:

$$Ax = b,$$

using the conjugate-gradient (CG) method with SSOR preconditioning.

The CG algorithm itself is implemented by the reverse communication routine F11GBF, which returns repeatedly to the calling program with various values of the parameter IREVCM. This parameter indicates the action to be taken by the calling program.

If IREVCM = 1 a matrix-vector product $v = Au$ is required. This is implemented by a call to F11XEF.

If IREVCM = 2 a solution of the preconditioning equation $Mv = u$ is required. This is achieved by a call to F11JDF.

If IREVCM = 4 F11GBF has completed its tasks. Either the iteration has terminated, or an error condition has arisen.

For further details see the routine document for F11GBF.

## 9.1 Example 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.

```
*     F11JDF Example Program Text
*     Mark 17 Release. NAG Copyright 1995.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LA, LIWORK, LWORK
      PARAMETER        (NMAX=1000,LA=10000,LIWORK=NMAX+1,LWORK=6*NMAX)
*     .. Local Scalars ..
      real             ANORM, OMEGA, SIGERR, SIGMAX, SIGTOL, STPLHS,
     +                 STPRHS, TOL
      INTEGER          I, IFAIL, IREVCM, ITERM, ITN, ITS, LWNEED,
     +                 MAXITN, MAXITS, MONIT, N, NNZ
      CHARACTER        CKJDF, CKXEF, NORM, PRECON, SIGCMP, WEIGHT
      CHARACTER*6      METHOD
*     .. Local Arrays ..
      real             A(LA), B(NMAX), RDIAG(NMAX), WORK(LWORK), X(NMAX)
      INTEGER          ICOL(LA), IROW(LA), IWORK(LIWORK)
*     .. External Subroutines ..
      EXTERNAL         F11GAF, F11GBF, F11GCF, F11JDF, F11XEF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11JDF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
*
*     Read algorithmic parameters
*
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
         READ (NIN,*) NNZ
         READ (NIN,*) METHOD
         READ (NIN,*) PRECON, SIGCMP, NORM, ITERM
         READ (NIN,*) TOL, MAXITN
         READ (NIN,*) ANORM, SIGMAX
         READ (NIN,*) SIGTOL, MAXITS
         READ (NIN,*) OMEGA
*
*     Read the matrix A
*
         DO 20 I = 1, NNZ
            READ (NIN,*) A(I), IROW(I), ICOL(I)
   20    CONTINUE
*
*     Read right-hand side vector b and initial approximate solution x
*
```

```
            READ (NIN,*) (B(I),I=1,N)
            READ (NIN,*) (X(I),I=1,N)
*
*     Call F11GAF to initialize solver
*
            WEIGHT = 'N'
            MONIT = 0
            IFAIL = 0
            CALL F11GAF(METHOD,PRECON,SIGCMP,NORM,WEIGHT,ITERM,N,TOL,
     +                  MAXITN,ANORM,SIGMAX,SIGTOL,MAXITS,MONIT,LWNEED,
     +                  IFAIL)
*
*     Calculate reciprocal diagonal matrix elements.
*
            DO 40 I = 1, N
               IWORK(I) = 0
   40       CONTINUE
*
            DO 60 I = 1, NNZ
               IF (IROW(I).EQ.ICOL(I)) THEN
                  IWORK(IROW(I)) = IWORK(IROW(I)) + 1
                  IF (A(I).NE.0.0e0) THEN
                     RDIAG(IROW(I)) = 1.0e0/A(I)
                  ELSE
                     WRITE (NOUT,*) 'Matrix has a zero diagonal element'
                     GO TO 140
                  END IF
               END IF
   60       CONTINUE
*
            DO 80 I = 1, N
               IF (IWORK(I).EQ.0) THEN
                  WRITE (NOUT,*) 'Matrix has a missing diagonal element'
                  GO TO 140
               END IF
               IF (IWORK(I).GE.2) THEN
                  WRITE (NOUT,*) 'Matrix has a multiple diagonal element'
                  GO TO 140
               END IF
   80       CONTINUE
*
*     Call F11GBF to solve the linear system
*
            IREVCM = 0
            CKXEF = 'C'
            CKJDF = 'C'
*
  100       CONTINUE
*
            CALL F11GBF(IREVCM,X,B,WORK,LWORK,IFAIL)
*
            IF (IREVCM.EQ.1) THEN
*
*           Compute matrix vector product
*
              CALL F11XEF(N,NNZ,A,IROW,ICOL,CKXEF,X,B,IFAIL)
              CKXEF = 'N'
              GO TO 100
```

```
*
      ELSE IF (IREVCM.EQ.2) THEN
*
*       SSOR preconditioning
*
          CALL F11JDF(N,NNZ,A,IROW,ICOL,RDIAG,OMEGA,CKJDF,X,B,IWORK,
     +                IFAIL)
          CKJDF = 'N'
          GO TO 100
*
      ELSE IF (IREVCM.EQ.4) THEN
*
*       Termination
*
          CALL F11GCF(ITN,STPLHS,STPRHS,ANORM,SIGMAX,ITS,SIGERR,IFAIL)
*
          WRITE (NOUT,99999) 'Converged in', ITN, ' iterations'
          WRITE (NOUT,99998) 'Final residual norm =', STPLHS
*
*       Output x
*
          DO 120 I = 1, N
             WRITE (NOUT,99997) X(I)
  120     CONTINUE
*
      END IF
*
  140 CONTINUE
    END IF
    STOP
*
99999 FORMAT (1X,A,I10,A)
99998 FORMAT (1X,A,1P,e16.3)
99997 FORMAT (1X,1P,e16.4)
      END
```

## 9.2 Example Data

```
F11JDF Example Program Data
 7                    N
16                    NNZ
'CG'                  METHOD
'P' 'N' 'I' 1         PRECON, SIGCMP, NORM, ITERM
1.0E-6 100            TOL, MAXITN
0.0E0 0.0E0           ANORM, SIGMAX
0.0E0 10              SIGTOL, MAXITS
1.0E0                 OMEGA
 4.   1    1
 1.   2    1
 5.   2    2
 2.   3    3
 2.   4    2
 3.   4    4
-1.   5    1
 1.   5    4
 4.   5    5
 1.   6    2
-2.   6    5
```

```
 3.   6    6
 2.   7    1
-1.   7    2
-2.   7    3
 5.   7    7          A(I), IROW(I), ICOL(I), I=1,...,NNZ
15.  18.  -8.   21.
11.  10.   29.        B(I), I=1,...,N
 0.   0.   0.    0.
 0.   0.   0.        X(I), I=1,...,N
```

## 9.3 Example Results

```
F11JDF Example Program Results
Converged in         6 iterations
Final residual norm =       7.105E-15
      1.0000E+00
      2.0000E+00
      3.0000E+00
      4.0000E+00
      5.0000E+00
      6.0000E+00
      7.0000E+00
```

# F11JEF – 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.

## 1 Purpose

F11JEF solves a real sparse symmetric system of linear equations, represented in symmetric coordinate storage format, using a conjugate gradient or Lanczos method, without preconditioning, with Jacobi or with SSOR preconditioning.

## 2 Specification

```
 SUBROUTINE F11JEF(METHOD, PRECON, N, NNZ, A, IROW, ICOL, OMEGA, B,
1                  TOL, MAXITN, X, RNORM, ITN, WORK, LWORK, IWORK,
2                  IFAIL)
 INTEGER           N, NNZ, IROW(NNZ), ICOL(NNZ), MAXITN, ITN,
1                  IWORK(N+1), LWORK, IFAIL
 real              A(NNZ), OMEGA, B(N), TOL, X(N), RNORM,
1                  WORK(LWORK)
 CHARACTER*(*)     METHOD
 CHARACTER*1       PRECON
```

## 3 Description

This routine solves a real sparse symmetric linear system of equations:

$$Ax = b,$$

using a preconditioned conjugate gradient method [1], or a preconditioned Lanczos method based on the algorithm SYMMLQ [2]. The conjugate gradient method is more efficient if $A$ is positive-definite, but may fail to converge for indefinite matrices. In this case the Lanczos method should be used instead. For further details see [1].

The routine allows the following choices for the preconditioner:

no preconditioning;

Jacobi preconditioning [3];

symmetric successive-over-relaxation (SSOR) preconditioning [3].

For incomplete Cholesky (IC) preconditioning see F11JCF.

The matrix $A$ is represented in symmetric coordinate storage (SCS) format (see Section 2.1.1 of the Chapter Introduction) in the arrays A, IROW and ICOL. The array A holds the non-zero entries in the lower triangular part of the matrix, while IROW and ICOL hold the corresponding row and column indices.

## 4 References

[1] Barrett R, Berry M, Chan T F, Demmel J, Donato J, Dongarra J, Eijkhout V, Pozo R, Romine C and van der Vorst H (1994) *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods* SIAM, Philadelphia

[2] Paige C C and Saunders M A (1975) Solution of sparse indefinite systems of linear equations *SIAM J. Numer. Anal.* **12** 617–629

[3] Young D (1971) *Iterative Solution of Large Linear Systems* Academic Press, New York

# 5 Parameters

**1:** METHOD — CHARACTER*(*) *Input*

*On entry:* specifies the iterative method to be used. The possible choices are:

'CG' conjugate gradient method;
'SYMMLQ' Lanczos method (SYMMLQ).

*Constraint:* METHOD = 'CG' or 'SYMMLQ'.

**2:** PRECON — CHARACTER*1 *Input*

*On entry:* specifies the type of preconditioning to be used. The possible choices are:

'N' no preconditioning;
'J' Jacobi;
'S' symmetric successive-over-relaxation.

*Constraint:* PRECON = 'N', 'J' or 'S'.

**3:** N — INTEGER *Input*

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

*Constraint:* $\mathrm{N} \geq 1$.

**4:** NNZ — INTEGER *Input*

*On entry:* the number of non-zero elements in the lower triangular part of the matrix $A$.

*Constraint:* $1 \leq \mathrm{NNZ} \leq \mathrm{N} \times (\mathrm{N}+1)/2$.

**5:** A(NNZ) — ***real*** array *Input*

*On entry:* the non-zero elements of the lower triangular part of the matrix $A$, ordered by increasing row index, and by increasing column index within each row. Multiple entries for the same row and column indices are not permitted. The routine F11ZBF may be used to order the elements in this way.

**6:** IROW(NNZ) — INTEGER array *Input*

**7:** ICOL(NNZ) — INTEGER array *Input*

*On entry:* the row and column indices of the non-zero elements supplied in A.

*Constraints:* IROW and ICOL must satisfy the following constraints (which may be imposed by a call to F11ZBF):

$1 \leq \mathrm{IROW}(i) \leq \mathrm{N}$ and $1 \leq \mathrm{ICOL}(i) \leq \mathrm{IROW}(i)$, for $i = 1, 2, \ldots, \mathrm{NNZ}$.
$\mathrm{IROW}(i-1) < \mathrm{IROW}(i)$, or
$\mathrm{IROW}(i-1) = \mathrm{IROW}(i)$ and $\mathrm{ICOL}(i-1) < \mathrm{ICOL}(i)$, for $i = 2, 3, \ldots, \mathrm{NNZ}$.

**8:** OMEGA — ***real*** *Input*

*On entry:* if PRECON = 'S', OMEGA is the relaxation parameter $\omega$ to be used in the SSOR method. Otherwise PRECON need not be initialized.

*Constraint:* $0.0 \leq \mathrm{OMEGA} \leq 2.0$.

**9:** B(N) — ***real*** array *Input*

*On entry:* the right-hand side vector $b$.

**10:** TOL — ***real*** *Input*

*On entry:* the required tolerance. Let $x_k$ denote the approximate solution at iteration $k$, and $r_k$ the corresponding residual. The algorithm is considered to have converged at iteration $k$ if:

$$\|r_k\|_\infty \leq \tau \times (\|b\|_\infty + \|A\|_\infty \|x_k\|_\infty).$$

If TOL $\leq 0.0$, $\tau = \max(\sqrt{\epsilon}, \sqrt{n}\,\epsilon)$ is used, where $\epsilon$ is the ***machine precision***. Otherwise $\tau = \max(\text{TOL}, 10\epsilon, \sqrt{n}\,\epsilon)$ is used.

*Constraint:* TOL $< 1.0$

**11:** MAXITN — INTEGER *Input*

*On entry:* the maximum number of iterations allowed.

*Constraint:* MAXITN $\geq 1$.

**12:** X(N) — ***real*** array *Input/Output*

*On entry:* an initial approximation to the solution vector $x$.

*On exit:* an improved approximation to the solution vector $x$.

**13:** RNORM — ***real*** *Output*

*On exit:* the final value of the residual norm $\|r_k\|_\infty$, where $k$ is the output value of ITN.

**14:** ITN — INTEGER *Output*

*On exit:* the number of iterations carried out.

**15:** WORK(LWORK) — ***real*** array *Workspace*

**16:** LWORK — INTEGER *Input*

*On entry:* the dimension of the array WORK as declared in the (sub)program from which F11JEF is called.

*Constraints:*

if METHOD = 'CG' then LWORK $\geq 6 \times \text{N} + \nu$;
if METHOD = 'SYMMLQ' then LWORK $\geq 7 \times \text{N} + \nu$,

where $\nu = \text{N}$ for PRECON = 'J' or 'S', and 0 otherwise.

**17:** IWORK(N+1) — INTEGER array *Workspace*

**18:** IFAIL — INTEGER *Input/Output*

*On entry:* IFAIL must be set to 0, $-1$ or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

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

# 6 Errors and Warnings

If on entry IFAIL = 0 or $-1$, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL = 1

On entry, METHOD $\neq$ 'CG' or 'SYMMLQ',
or PRECON $\neq$ 'N', 'J' or 'S',
or N $< 1$,
or NNZ $< 1$,

or NNZ > N × (N + 1)/2,

or OMEGA lies outside the interval [0.0, 2.0],

or TOL ≥ 1.0,

or MAXITN < 1,

or LWORK too small.

IFAIL = 2

On entry, the arrays IROW and ICOL fail to satisfy the following constraints:

$1 \leq \mathrm{IROW}(i) \leq \mathrm{N}$ and $1 \leq \mathrm{ICOL}(i) \leq \mathrm{IROW}(i)$, for $i = 1, 2, \ldots, \mathrm{NNZ}$.
$\mathrm{IROW}(i-1) < \mathrm{IROW}(i)$, or
$\mathrm{IROW}(i-1) = \mathrm{IROW}(i)$ and $\mathrm{ICOL}(i-1) < \mathrm{ICOL}(i)$, for $i = 2, 3, \ldots, \mathrm{NNZ}$.

Therefore a non-zero element has been supplied which does not lie in the lower triangular part of $A$, is out of order, or has duplicate row and column indices. Call F11ZBF to reorder and sum or remove duplicates.

IFAIL = 3

On entry, the matrix $A$ has a zero diagonal element. Jacobi and SSOR preconditioners are not appropriate for this problem.

IFAIL = 4

The required accuracy could not be obtained. However, a reasonable accuracy has been obtained and further iterations could not improve the result.

IFAIL = 5

Required accuracy not obtained in MAXITN iterations.

IFAIL = 6

The preconditioner appears not to be positive-definite.

IFAIL = 7

The matrix of the coefficients appears not to be positive-definite (conjugate gradient method only).

IFAIL = 8

A serious error has occurred in an internal call to F11GAF, F11GBF or F11GCF. Check all subroutine calls and array sizes. Seek expert help.

# 7 Accuracy

On successful termination, the final residual $r_k = b - Ax_k$, where $k$ = ITN, satisfies the termination criterion

$$\|r_k\|_\infty \leq \tau \times (\|b\|_\infty + \|A\|_\infty \|x_k\|_\infty).$$

The value of the final residual norm is returned in RNORM.

# 8 Further Comments

The time taken by F11JEF for each iteration is roughly proportional to NNZ. One iteration with the Lanczos method (SYMMLQ) requires a slightly larger number of operations than one iteration with the conjugate gradient method.

The number of iterations required to achieve a prescribed accuracy cannot be easily determined a priori, as it can depend dramatically on the conditioning and spectrum of the preconditioned matrix of the coefficients $\bar{A} = M^{-1}A$.

# 9 Example

This example program solves a symmetric positive-definite system of equations using the conjugate gradient method, with SSOR preconditioning.

## 9.1 Example 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.

```
*     F11JEF Example Program Text
*     Mark 17 Release. NAG Copyright 1995.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LA, LWORK
      PARAMETER        (NMAX=1000,LA=10000,LWORK=6*NMAX)
*     .. Local Scalars ..
      real             OMEGA, RNORM, TOL
      INTEGER          I, IFAIL, ITN, MAXITN, N, NNZ
      CHARACTER        PRECON
      CHARACTER*6      METHOD
*     .. Local Arrays ..
      real             A(LA), B(NMAX), WORK(LWORK), X(NMAX)
      INTEGER          ICOL(LA), IROW(LA), IWORK(NMAX+1)
*     .. External Subroutines ..
      EXTERNAL         F11JEF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11JEF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
*
*     Read algorithmic parameters
*
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
         READ (NIN,*) NNZ
         READ (NIN,*) METHOD, PRECON
         READ (NIN,*) OMEGA
         READ (NIN,*) TOL, MAXITN
*
*     Read the matrix A
*
         DO 20 I = 1, NNZ
            READ (NIN,*) A(I), IROW(I), ICOL(I)
   20    CONTINUE
*
*     Read right-hand side vector b and initial approximate solution x
*
         READ (NIN,*) (B(I),I=1,N)
         READ (NIN,*) (X(I),I=1,N)
*
*     Solve Ax = b using F11JEF
*
         IFAIL = 0
         CALL F11JEF(METHOD,PRECON,N,NNZ,A,IROW,ICOL,OMEGA,B,TOL,MAXITN,
     +               X,RNORM,ITN,WORK,LWORK,IWORK,IFAIL)
*
```

```
         WRITE (NOUT,99999) 'Converged in', ITN, ' iterations'
         WRITE (NOUT,99998) 'Final residual norm =', RNORM
*
*     Output x
*
         DO 40 I = 1, N
            WRITE (NOUT,99997) X(I)
   40    CONTINUE
      END IF
      STOP
*
99999 FORMAT (1X,A,I10,A)
99998 FORMAT (1X,A,1P,e16.3)
99997 FORMAT (1X,1P,e16.4)
      END
```

## 9.2 Example Data

```
F11JEF Example Program Data
  7                   N
  16                  NNZ
 'CG' 'SSOR'          METHOD, PRECON
  1.1                 OMEGA
  1.0E-6 100          TOL, MAXITN
  4.   1    1
  1.   2    1
  5.   2    2
  2.   3    3
  2.   4    2
  3.   4    4
 -1.   5    1
  1.   5    4
  4.   5    5
  1.   6    2
 -2.   6    5
  3.   6    6
  2.   7    1
 -1.   7    2
 -2.   7    3
  5.   7    7          A(I), IROW(I), ICOL(I), I=1,...,NNZ
 15.  18.  -8.  21.
 11.  10.  29.         B(I), I=1,...,N
  0.   0.   0.   0.
  0.   0.   0.         X(I), I=1,...,N
```

## 9.3 Example Results

```
F11JEF Example Program Results
Converged in          6 iterations
Final residual norm =       5.026E-06
      1.0000E+00
      2.0000E+00
      3.0000E+00
      4.0000E+00
      5.0000E+00
      6.0000E+00
      7.0000E+00
```

# F11XAF – 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.

## 1 Purpose

Computes a matrix-vector or transposed matrix-vector product involving a real sparse nonsymmetric matrix stored in coordinate storage format.

## 2 Specification

```
SUBROUTINE F11XAF(TRANS, N, NNZ, A, IROW, ICOL, CHECK, X, Y, IFAIL)
INTEGER           N, NNZ, IROW(NNZ), ICOL(NNZ), IFAIL
real              A(NNZ), X(N), Y(N)
CHARACTER*1       TRANS, CHECK
```

## 3 Description

F11XAF computes either the matrix-vector product $y = Ax$, or the transposed matrix-vector product $y = A^T x$, according to the value of the argument TRANS, where $A$ is an $n$ by $n$ sparse nonsymmetric matrix, of arbitrary sparsity pattern. The matrix $A$ is stored in coordinate storage (CS) format (see Section 2.1.1 of the Chapter Introduction). The array A stores all non-zero elements of $A$, while arrays IROW and ICOL store the corresponding row and column indices respectively.

It is envisaged that a common use of F11XAF will be to compute the matrix-vector product required in the application of F11BBF to sparse linear systems. An illustration of this usage appears in Section 9 of the routine document for F11DDF.

## 4 References

None.

## 5 Parameters

**1:** TRANS — CHARACTER*1 *Input*

*On entry:* specifies whether or not the matrix $A$ is transposed:

if TRANS = 'N', then $y = Ax$ is computed;
if TRANS = 'T', then $y = A^T x$ is computed.

*Constraint:* TRANS = 'N' or 'T'.

**2:** N — INTEGER *Input*

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

*Constraint:* N $\geq$ 1.

**3:** NNZ — INTEGER *Input*

*On entry:* the number of non-zero elements in the matrix $A$.

*Constraint:* $1 \leq$ NNZ $\leq$ N$^2$.

**4:** A(NNZ) — ***real*** array *Input*

*On entry:* the non-zero elements in the matrix $A$, ordered by increasing row index, and by increasing column index within each row. Multiple entries for the same row and column indices are not permitted. The routine F11ZAF may be used to order the elements in this way.

**5:** IROW(NNZ) — INTEGER array *Input*

**6:** ICOL(NNZ) — INTEGER array *Input*

*On entry:* the row and column indices of the non-zero elements supplied in A.

*Constraints:* IROW and ICOL must satisfy the following constraints (which may be imposed by a call to F11ZAF):

$1 \leq \text{IROW}(i) \leq \text{N}$, $1 \leq \text{ICOL}(i) \leq \text{N}$, for $i = 1, 2, \ldots, \text{NNZ}$.
$\text{IROW}(i-1) < \text{IROW}(i)$, or
$\text{IROW}(i-1) = \text{IROW}(i)$ and $\text{ICOL}(i-1) < \text{ICOL}(i)$, for $i = 2, 3, \ldots, \text{NNZ}$.

**7:** CHECK — CHARACTER*1 *Input*

*On entry:* specifies whether or not the CS representation of the matrix $A$ should be checked:

if CHECK = 'C', checks are carried on the values of N, NNZ, IROW and ICOL;
if CHECK = 'N', none of these checks are carried out.

See also Section 8.2.

*Constraint:* CHECK = 'C' or 'N'.

**8:** X(N) — ***real*** array *Input*

*On entry:* the vector $x$.

**9:** Y(N) — ***real*** array *Output*

*On exit:* the vector $y$.

**10:** IFAIL — INTEGER *Input/Output*

*On entry:* IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

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

# 6 Errors and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL = 1

On entry, TRANS ≠ 'N' or 'T',
or CHECK ≠ 'C' or 'N'.

IFAIL = 2

On entry, N < 1,
or NNZ < 1,
or NNZ > $\text{N}^2$.

IFAIL = 3

On entry, the arrays IROW and ICOL fail to satisfy the following constraints:

$1 \leq \text{IROW}(i) \leq \text{N}$ and $1 \leq \text{ICOL}(i) \leq \text{N}$, for $i = 1, 2, \ldots, \text{NNZ}$.
$\text{IROW}(i-1) < \text{IROW}(i)$, or
$\text{IROW}(i-1) = \text{IROW}(i)$ and $\text{ICOL}(i-1) < \text{ICOL}(i)$, for $i = 2, 3, \ldots, \text{NNZ}$.

Therefore a non-zero element has been supplied which does not lie within the matrix $A$, is out of order, or has duplicate row and column indices. Call F11ZAF to reorder and sum or remove duplicates.

# 7 Accuracy

The computed vector $y$ satisfies the error bound:

$$||y - Ax||_\infty \leq c(n)\epsilon||A||_\infty||x||_\infty,$$

if TRANS = 'N', or

$$||y - A^T x||_\infty \leq c(n)\epsilon||A^T||_\infty||x||_\infty,$$

if TRANS = 'T', where $c(n)$ is a modest linear function of $n$, and $\epsilon$ is the ***machine precision***.

# 8 Further Comments

## 8.1 Timing

The time taken for a call to F11XAF is proportional to NNZ.

## 8.2 Use of CHECK

It is expected that a common use of F11XAF will be to compute the matrix-vector product required in the application of F11BBF to sparse linear systems. In this situation F11XAF is likely to be called many times with the same matrix $A$. In the interests of both reliability and efficiency you are recommended to set CHECK to 'C' for the first of such calls, and to 'N' for all subsequent calls.

# 9 Example

This example program reads in a sparse matrix $A$ and a vector $x$. It then calls F11XAF to compute the matrix-vector product $y = Ax$ and the transposed matrix-vector product $y = A^T x$.

## 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.

```
*     F11XAF Example Program Text
*     Mark 18 Release. NAG Copyright 1997.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          LA, NMAX
      PARAMETER        (LA=10000,NMAX=1000)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, N, NNZ
      CHARACTER        CHECK, TRANS
*     .. Local Arrays ..
      real             A(LA), X(NMAX), Y(NMAX)
      INTEGER          ICOL(LA), IROW(LA)
*     .. External Subroutines ..
      EXTERNAL         F11XAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11XAF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
*
*     Read order of matrix and number of non-zero entries
*
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
         READ (NIN,*) NNZ
```

```
*
*        Read the matrix A
*
         DO 20 I = 1, NNZ
            READ (NIN,*) A(I), IROW(I), ICOL(I)
   20    CONTINUE
*
*        Read the vector x
*
         READ (NIN,*) (X(I),I=1,N)
*
*        Calculate matrix-vector product
*
         TRANS = 'Not transposed'
         CHECK = 'C'
         IFAIL = 0
         CALL F11XAF(TRANS,N,NNZ,A,IROW,ICOL,CHECK,X,Y,IFAIL)
*
*        Output results
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) ' Matrix-vector product'
         DO 40 I = 1, N
            WRITE (NOUT,'(e16.4)') Y(I)
   40    CONTINUE
*
*        Calculate transposed matrix-vector product
*
         TRANS = 'Transposed'
         CHECK = 'N'
         IFAIL = 0
         CALL F11XAF(TRANS,N,NNZ,A,IROW,ICOL,CHECK,X,Y,IFAIL)
*
*        Output results
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) ' Transposed matrix-vector product'
         DO 60 I = 1, N
            WRITE (NOUT,'(e16.4)') Y(I)
   60    CONTINUE
*
      END IF
      STOP
      END
```

## 9.2 Program Data

```
F11XAF Example Program Data
  5                    N
 11                    NNZ
  2.   1    1
  1.   1    2
  1.   2    3
 -1.   2    4
  4.   3    1
  1.   3    3
  1.   3    5
  1.   4    4
```

```
 2.   4    5
-2.   5    2
 3.   5    5          A(I), IROW(I), ICOL(I), I=1,...,NNZ
0.70 0.16 0.52
0.77 0.28             X(I), I=1,...,N
```

## 9.3 Program Results

```
F11XAF Example Program Results

 Matrix-vector product
     0.1560E+01
    -0.2500E+00
     0.3600E+01
     0.1330E+01
     0.5200E+00

 Transposed matrix-vector product
     0.3480E+01
     0.1400E+00
     0.6800E+00
     0.6100E+00
     0.2900E+01
```

# F11XEF – 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.

## 1 Purpose

Computes a matrix-vector product involving a real sparse symmetric matrix stored in symmetric coordinate storage format.

## 2 Specification

```
SUBROUTINE F11XEF(N, NNZ, A, IROW, ICOL, CHECK, X, Y, IFAIL)
INTEGER           N, NNZ, IROW(NNZ), ICOL(NNZ), IFAIL
real              A(NNZ), X(N), Y(N)
CHARACTER*1       CHECK
```

## 3 Description

F11XEF computes the matrix-vector product

$$y = Ax$$

where A is an N by N symmetric sparse matrix, of arbitrary sparsity pattern, stored in symmetric coordinate storage (SCS) format (see Section 2.1.1 of the Chapter Introduction). The array A stores all non-zero elements in the lower triangular part of $A$, while arrays IROW and ICOL store the corresponding row and column indices respectively.

It is envisaged that a common use of F11XEF will be to compute the matrix-vector product required in the application of F11GBF to sparse symmetric linear systems. An illustration of this usage appears in Section 9 of the routine document for F11JDF.

## 4 References

None.

## 5 Parameters

**1:** N — INTEGER *Input*

*On entry:* the order of the matrix $A$.

*Constraint:* $\mathrm{N} \geq 1$.

**2:** NNZ — INTEGER *Input*

*On entry:* the number of non-zero elements in the lower triangular part of $A$.

*Constraint:* $1 \leq \mathrm{NNZ} \leq \mathrm{N} \times (\mathrm{N}+1)/2$.

**3:** A(NNZ) — ***real*** array *Input*

*On entry:* the non-zero elements in the lower triangular part of the matrix $A$, ordered by increasing row index, and by increasing column index within each row. Multiple entries for the same row and column indices are not permitted. The routine F11ZBF may be used to order the elements in this way.

**4:** IROW(NNZ) — INTEGER array *Input*

**5:** ICOL(NNZ) — INTEGER array *Input*

*On entry:* the row and column indices of the non-zero elements supplied in A.

*Constraints:* IROW and ICOL must satisfy the following constraints (which may be imposed by a call to F11ZBF):

$1 \leq \mathrm{IROW}(i) \leq \mathrm{N}$, $1 \leq \mathrm{ICOL}(i) \leq \mathrm{IROW}(i)$, for $i = 1, 2, \ldots, \mathrm{NNZ}$.
$\mathrm{IROW}(i-1) < \mathrm{IROW}(i)$, or
$\mathrm{IROW}(i-1) = \mathrm{IROW}(i)$ and $\mathrm{ICOL}(i-1) < \mathrm{ICOL}(i)$, for $i = 2, 3, \ldots, \mathrm{NNZ}$.

**6:** CHECK — CHARACTER*1 *Input*

*On entry:* specifies whether or not the input data should be checked:

if CHECK = 'C', checks are carried out on the values of N, NNZ, IROW and ICOL;
if CHECK = 'N', none of these checks are carried out.

See also Section 8.2.

*Constraint:* CHECK = 'C' or 'N'.

**7:** X(N) — ***real*** array *Input*

*On entry:* the vector $x$.

**8:** Y(N) — ***real*** array *Output*

*On exit:* the vector $y$.

**9:** IFAIL — INTEGER *Input/Output*

*On entry:* IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

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

# 6 Errors and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL = 1

On entry, CHECK ≠ 'C' or 'N'.

IFAIL = 2

On entry, N < 1,
or NNZ < 1,
or NNZ > N × (N + 1)/2.

IFAIL = 3

On entry, the arrays IROW and ICOL fail to satisfy the following constraints:

$1 \leq \mathrm{IROW}(i) \leq \mathrm{N}$ and $1 \leq \mathrm{ICOL}(i) \leq \mathrm{IROW}(i)$, for $i = 1, 2, \ldots, \mathrm{NNZ}$.
$\mathrm{IROW}(i-1) < \mathrm{IROW}(i)$, or
$\mathrm{IROW}(i-1) = \mathrm{IROW}(i)$ and $\mathrm{ICOL}(i-1) < \mathrm{ICOL}(i)$, for $i = 2, 3, \ldots, \mathrm{NNZ}$.

Therefore a non-zero element has been supplied which does not lie in the lower triangular part of $A$, is out of order, or has duplicate row and column indices. Call F11ZBF to reorder and sum or remove duplicates.

# 7 Accuracy

The computed vector $y$ satisfies the error bound:

$$\|y - Ax\|_\infty \leq c(n)\epsilon\|A\|_\infty\|x\|_\infty,$$

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

# 8 Further Comments

## 8.1 Timing

The time taken for a call to F11XEF is proportional to NNZ.

## 8.2 Use of CHECK

It is expected that a common use of F11XEF will be to compute the matrix-vector product required in the application of F11GBF to sparse symmetric linear systems. In this situation F11XEF is likely to be called many times with the same matrix $A$. In the interests of both reliability and efficiency you are recommended to set CHECK to 'C' for the first of such calls, and to 'N' for all subsequent calls.

# 9 Example

This example program reads in a symmetric positive-definite sparse matrix $A$ and a vector $x$. It then calls F11XEF to compute the matrix-vector product $y = Ax$.

## 9.1 Example 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.

```
*     F11XEF Example Program Text
*     Mark 17 Release. NAG Copyright 1995.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          LA, NMAX
      PARAMETER        (LA=10000,NMAX=1000)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, N, NNZ
      CHARACTER        CHECK
*     .. Local Arrays ..
      real             A(LA), X(NMAX), Y(NMAX)
      INTEGER          ICOL(LA), IROW(LA)
*     .. External Subroutines ..
      EXTERNAL         F11XEF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11XEF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
*
*     Read order of matrix and number of non-zero entries
*
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
         READ (NIN,*) NNZ
*
*     Read the matrix A
*
```

```
         DO 20 I = 1, NNZ
            READ (NIN,*) A(I), IROW(I), ICOL(I)
   20    CONTINUE
*
*     Read the vector x
*
         READ (NIN,*) (X(I),I=1,N)
*
*     Calculate matrix-vector product
*
         CHECK = 'C'
         IFAIL = 0
         CALL F11XEF(N,NNZ,A,IROW,ICOL,CHECK,X,Y,IFAIL)
*
*     Output results
*
         WRITE (NOUT,*) ' Matrix-vector product'
         DO 40 I = 1, N
            WRITE (NOUT,99999) Y(I)
   40    CONTINUE
      END IF
      STOP
*
99999 FORMAT (1X,e16.4)
      END
```

## 9.2 Example Data

```
F11XEF Example Program Data
  9                    N
 23                    NNZ
  4.   1    1
 -1.   2    1
  6.   2    2
  1.   3    2
  2.   3    3
  3.   4    4
  2.   5    1
  4.   5    5
  1.   6    3
  2.   6    4
  6.   6    6
 -4.   7    2
  1.   7    5
 -1.   7    6
  6.   7    7
 -1.   8    4
 -1.   8    6
  3.   8    8
  1.   9    1
  1.   9    5
 -1.   9    6
  1.   9    8
  4.   9    9          A(I), IROW(I), ICOL(I), I=1,...,NNZ
 0.70 0.16 0.52
 0.77 0.28 0.21
 0.93 0.20 0.90        X(I), I=1,...,N
```

## 9.3 Example Results

```
F11XEF Example Program Results
 Matrix-vector product
      0.4100E+01
     -0.2940E+01
      0.1410E+01
      0.2530E+01
      0.4350E+01
      0.1290E+01
      0.5010E+01
      0.5200E+00
      0.4570E+01
```

# F11ZAF – 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.

## 1 Purpose

Sorts the non-zero elements of a real sparse nonsymmetric matrix, represented in coordinate storage format.

## 2 Specification

```
      SUBROUTINE F11ZAF(N, NNZ, A, IROW, ICOL, DUP, ZERO, ISTR, IWORK,
     1                  IFAIL)
      INTEGER           N, NNZ, IROW(*), ICOL(*), ISTR(N+1), IWORK(N),
     1                  IFAIL
      real              A(*)
      CHARACTER*1       DUP, ZERO
```

## 3 Description

F11ZAF takes a coordinate storage (CS) representation (see Section 2.1.1 of the Chapter Introduction) of a real $n$ by $n$ sparse nonsymmetric matrix $A$, and reorders the non-zero elements by increasing row index and increasing column index within each row. Entries with duplicate row and column indices may be removed, or the values may be summed. Any entries with zero values may optionally be removed.

F11ZAF also returns a pointer ISTR to the starting address of each row in $A$.

## 4 References

None.

## 5 Parameters

**1:** N — INTEGER *Input*

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

*Constraint:* $\mathrm{N} \geq 1$.

**2:** NNZ — INTEGER *Input/Output*

*On entry:* the number of non-zero elements in the matrix $A$.

*Constraint:* $\mathrm{NNZ} \geq 0$.

*On exit:* the number of non-zero elements with unique row and column indices.

**3:** A(*) — ***real*** array *Input/Output*

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

*On entry:* the non-zero elements of the matrix $A$. These may be in any order and there may be multiple non-zero elements with the same row and column indices.

*On exit:* the non-zero elements ordered by increasing row index, and by increasing column index within each row. Each non-zero element has a unique row and column index.

**4:** IROW(*) — INTEGER array *Input/Output*

**Note:** the dimension of the array IROW must be at least max(1,NNZ).

*On entry:* the row indices corresponding to the non-zero elements supplied in the array A.

*Constraint:* $1 \leq \text{IROW}(i) \leq \text{N}$, for $i = 1, 2, \ldots, \text{NNZ}$.

*On exit:* the first NNZ elements contain the row indices corresponding to the non-zero elements returned in the array A.

**5:** ICOL(*) — INTEGER array *Input/Output*

**Note:** the dimension of the array ICOL must be at least max(1,NNZ).

*On entry:* the column indices corresponding to the non-zero elements supplied in the array A.

*Constraint:* $1 \leq \text{ICOL}(i) \leq \text{N}$, for $i = 1, 2, \ldots, \text{NNZ}$.

*On exit:* the first NNZ elements contain the row indices corresponding to the non-zero elements returned in the array A.

**6:** DUP — CHARACTER*1 *Input*

*On entry:* indicates how any non-zero elements with duplicate row and column indices are to be treated:

if DUP = 'R' the entries are removed;
if DUP = 'S' the relevant values in A are summed;
if DUP = 'F' the routine fails on detecting a duplicate, with IFAIL = 3.

*Constraint:* DUP = 'R', 'S', or 'F'.

**7:** ZERO — CHARACTER*1 *Input*

*On entry:* indicates how any elements with zero values in A are to be treated:

if ZERO = 'R' the entries are removed;
if ZERO = 'K' the entries are kept;
if ZERO = 'F' the routine fails on detecting a zero, with IFAIL = 4.

*Constraint:* ZERO = 'R', 'K', or 'F'.

**8:** ISTR(N+1) — INTEGER array *Output*

*On exit:* ISTR($i$), for $i = 1, 2, \ldots, \text{N}$, contains the index of arrays A, IROW and ICOL where row $i$ of the matrix $A$ starts. ISTR(N+1) contains the index + 1 of the last non-zero element in $A$.

**9:** IWORK(N) — INTEGER array *Workspace*

**10:** IFAIL — INTEGER *Input/Output*

*On entry:* IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

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

# 6 Errors and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL = 1

On entry, N < 1,
or NNZ < 0,
or DUP ≠ 'R', 'S', or 'F',
or ZERO ≠ 'R', 'K', or 'F'.

IFAIL = 2

On entry, a non-zero element has been supplied which does not lie within the matrix $A$, i.e., one or more of the following constraints has been violated:

$1 \leq \mathrm{IROW}(i) \leq \mathrm{N}$,
$1 \leq \mathrm{ICOL}(i) \leq \mathrm{N}$,

for $i = 1, 2, \ldots,$NNZ.

IFAIL = 3

On entry, DUP = 'F', and non-zero elements have been supplied which have duplicate row and column indices.

IFAIL = 4

On entry, ZERO = 'F', and at least one matrix element has been supplied with a zero coefficient value.

# 7 Accuracy

Not applicable.

# 8 Further Comments

The time taken for a call to F11ZAF is proportional to NNZ.

Note that the resulting matrix may have either rows or columns with no entries. If row $i$ has no entries then ISTR($i$) = ISTR($i + 1$).

# 9 Example

This example program reads the CS representation of a real sparse matrix $A$, calls F11ZAF to reorder the non-zero elements, and outputs the original and the reordered representations.

## 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.

```
*     F11ZAF Example Program Text
*     Mark 18 Release. NAG Copyright 1997.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          LA, NMAX
```

```
      PARAMETER        (LA=10000,NMAX=1000)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, N, NNZ
      CHARACTER        DUP, ZERO
*     .. Local Arrays ..
      real             A(LA)
      INTEGER          ICOL(LA), IROW(LA), ISTR(NMAX+1), IWORK(NMAX)
*     .. External Subroutines ..
      EXTERNAL         F11ZAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11ZAF Example Program Results'
      WRITE (NOUT,*)
*     Skip heading in data file
      READ (NIN,*)
*
*     Read order of matrix and number of non-zero entries
*
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
         READ (NIN,*) NNZ
*
*        Read and output the original non-zero elements
*
         DO 20 I = 1, NNZ
            READ (NIN,*) A(I), IROW(I), ICOL(I)
   20    CONTINUE
         WRITE (NOUT,*) 'Original elements'
         WRITE (NOUT,'(A,I4)') 'NNZ = ', NNZ
         DO 40 I = 1, NNZ
            WRITE (NOUT,'(I8,e16.4,2I8)') I, A(I), IROW(I), ICOL(I)
   40    CONTINUE
*
*        Reorder, sum duplicates and remove zeros
*
         DUP = 'S'
         ZERO = 'R'
         IFAIL = 0
*
         CALL F11ZAF(N,NNZ,A,IROW,ICOL,DUP,ZERO,ISTR,IWORK,IFAIL)
*
*        Output results
*
         WRITE (NOUT,*) 'Reordered elements'
         WRITE (NOUT,'(A,I4)') 'NNZ = ', NNZ
         DO 60 I = 1, NNZ
            WRITE (NOUT,'(I8,e16.4,2I8)') I, A(I), IROW(I), ICOL(I)
   60    CONTINUE
*
      END IF
      STOP
      END
```

## 9.2 Program Data

```
F11ZAF Example Program Data
  5                    N
  15                   NNZ
  4.   3    1
 -2.   5    2
  1.   4    4
 -2    4    2
 -3    5    5
  1.   1    2
  0.   1    5
  1.   3    5
 -1.   2    4
  6.   5    5
  2.   1    1
  2.   4    2
  1.   2    3
  1.   3    3
  2.   4    5          A(I), IROW(I), ICOL(I), I=1,...,NNZ
```

## 9.3 Program Results

```
 F11ZAF Example Program Results

 Original elements
NNZ =   15
       1      0.4000E+01       3       1
       2     -0.2000E+01       5       2
       3      0.1000E+01       4       4
       4     -0.2000E+01       4       2
       5     -0.3000E+01       5       5
       6      0.1000E+01       1       2
       7      0.0000E+00       1       5
       8      0.1000E+01       3       5
       9     -0.1000E+01       2       4
      10      0.6000E+01       5       5
      11      0.2000E+01       1       1
      12      0.2000E+01       4       2
      13      0.1000E+01       2       3
      14      0.1000E+01       3       3
      15      0.2000E+01       4       5
 Reordered elements
NNZ =   11
       1      0.2000E+01       1       1
       2      0.1000E+01       1       2
       3      0.1000E+01       2       3
       4     -0.1000E+01       2       4
       5      0.4000E+01       3       1
       6      0.1000E+01       3       3
       7      0.1000E+01       3       5
       8      0.1000E+01       4       4
       9      0.2000E+01       4       5
      10     -0.2000E+01       5       2
      11      0.3000E+01       5       5
```

# F11ZBF – 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.

## 1 Purpose

Sorts the non-zero elements of a real sparse symmetric matrix, represented in symmetric coordinate storage format.

## 2 Specification

```
SUBROUTINE F11ZBF(N, NNZ, A, IROW, ICOL, DUP, ZERO, ISTR, IWORK,
1                 IFAIL)
INTEGER           N, NNZ, IROW(*), ICOL(*), ISTR(N+1), IWORK(N),
1                 IFAIL
real              A(*)
CHARACTER*1       DUP, ZERO
```

## 3 Description

F11ZBF takes a symmetric coordinate storage (SCS) representation (see Section 2.1.2 of the Chapter Introduction) of a real $n$ by $n$ sparse symmetric matrix $A$, and reorders the non-zero elements by increasing row index and increasing column index within each row. Entries with duplicate row and column indices may be removed, or the values may be summed. Any entries with zero values may optionally be removed.

F11ZBF also returns a pointer ISTR to the starting address of each row in $A$.

## 4 References

None.

## 5 Parameters

**1:** N — INTEGER *Input*

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

*Constraint:* $\mathrm{N} \geq 1$.

**2:** NNZ — INTEGER *Input/Output*

*On entry:* the number of non-zero elements in the lower triangular part of the matrix $A$.

*Constraint:* $\mathrm{NNZ} \geq 0$.

*On exit:* the number of lower triangular non-zero elements with unique row and column indices.

**3:** A(*) — ***real*** array *Input/Output*

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

*On entry:* the non-zero elements of the lower triangular part of the matrix $A$. These may be in any order and there may be multiple non-zero elements with the same row and column indices.

*On exit:* the lower triangular non-zero elements ordered by increasing row index, and by increasing column index within each row. Each non-zero element has a unique row and column index.

**4:** IROW(*) — INTEGER array *Input/Output*

**Note:** the dimension of the array IROW must be at least max(1,NNZ).

*On entry:* the row indices corresponding to the non-zero elements supplied in the array A.

*Constraint:* $1 \leq \text{IROW}(i) \leq \text{N}$, for $i = 1, 2, \ldots, \text{NNZ}$.

*On exit:* the first NNZ elements contain the row indices corresponding to the non-zero elements returned in the array A.

**5:** ICOL(*) — INTEGER array *Input/Output*

**Note:** the dimension of the array ICOL must be at least max(1,NNZ).

*On entry:* the column indices corresponding to the non-zero elements supplied in the array A.

*Constraint:* $1 \leq \text{ICOL}(i) \leq \text{IROW}(i)$, for $i = 1, 2, \ldots, \text{NNZ}$.

*On exit:* the first NNZ elements contain the column indices corresponding to the non-zero elements returned in the array A.

**6:** DUP — CHARACTER*1 *Input*

*On entry:* indicates how any non-zero elements with duplicate row and column indices are to be treated:

if DUP = 'R' the entries are removed;
if DUP = 'S' the relevant values in A are summed.

*Constraint:* DUP = 'R' or 'S'.

**7:** ZERO — CHARACTER*1 *Input*

*On entry:* indicates how any elements with zero values in A are to be treated:

if ZERO = 'R' the entries are removed;
if ZERO = 'K' the entries are kept.

*Constraint:* ZERO = 'R' or 'K'.

**8:** ISTR(N+1) — INTEGER array *Output*

*On exit:* ISTR($i$), for $i = 1, 2, \ldots, \text{N}$, contains the starting address in the arrays A, IROW and ICOL of row $i$ of the matrix $A$. ISTR(N+1) contains the address of the last non-zero element in $A$ plus one.

**9:** IWORK(N) — INTEGER array *Workspace*

**10:** IFAIL — INTEGER *Input/Output*

*On entry:* IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

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

# 6 Errors and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL = 1

On entry, N < 1,
or NNZ < 0,
or DUP ≠ 'R' or 'S'.
or ZERO ≠ 'R' or 'K'.

IFAIL = 2

On entry, a non-zero element has been supplied which does not lie in the lower triangular part of $A$, i.e., one or more of the following constraints has been violated:

$$1 \leq \text{IROW}(i) \leq \text{N},$$
$$1 \leq \text{ICOL}(i) \leq \text{IROW}(i),$$

for $i = 1, 2, \ldots, \text{NNZ}$.

# 7 Accuracy

Not applicable.

# 8 Further Comments

The time taken for a call to F11ZBF is proportional to NNZ.

Note that the resulting matrix may have either rows or columns with no entries. If row $i$ has no entries then $\text{ISTR}(i) = \text{ISTR}(i+1)$.

# 9 Example

This example program reads the SCS representation of a real sparse symmetric matrix $A$, calls F11ZBF to reorder the non-zero elements, and outputs the original and the reordered representations.

## 9.1 Example 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.

```
*     F11ZBF Example Program Text
*     Mark 17 Release. NAG Copyright 1995.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          LA, NMAX
      PARAMETER        (LA=10000,NMAX=1000)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, N, NNZ
      CHARACTER        DUP, ZERO
*     .. Local Arrays ..
      real             A(LA)
      INTEGER          ICOL(LA), IROW(LA), ISTR(NMAX+1), IWORK(NMAX)
*     .. External Subroutines ..
      EXTERNAL         F11ZBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11ZBF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
*
*     Read order of matrix and number of non-zero entries
*
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
         READ (NIN,*) NNZ
*
*     Read and output the original non-zero elements
```

```
*
         DO 20 I = 1, NNZ
            READ (NIN,*) A(I), IROW(I), ICOL(I)
   20    CONTINUE
         WRITE (NOUT,*) 'Original elements'
         WRITE (NOUT,*) 'NNZ = ', NNZ
         DO 40 I = 1, NNZ
            WRITE (NOUT,99998) I, A(I), IROW(I), ICOL(I)
   40    CONTINUE
*
*     Reorder, sum duplicates and remove zeros
*
         DUP = 'S'
         ZERO = 'R'
         IFAIL = 0
*
         CALL F11ZBF(N,NNZ,A,IROW,ICOL,DUP,ZERO,ISTR,IWORK,IFAIL)
*
*     Output results
*
         WRITE (NOUT,*) 'Reordered elements'
         WRITE (NOUT,99999) 'NNZ = ', NNZ
         DO 60 I = 1, NNZ
            WRITE (NOUT,99998) I, A(I), IROW(I), ICOL(I)
   60    CONTINUE
      END IF
      STOP
*
99999 FORMAT (1X,A,I4)
99998 FORMAT (1X,I8,e16.4,2I8)
      END
```

## 9.2 Example Data

```
F11ZBF Example Program Data
  4                    N
  9                    NNZ
  1.   3    2
  0.   2    1
  1.   3    2
  3.   4    4
  4.   1    1
  6.   2    2
  2.   3    3
  1.   3    2
  1.   3    2          A(I), IROW(I), ICOL(I), I=1,...,NNZ
```

## 9.3 Example Results

```
F11ZBF Example Program Results
Original elements
NNZ =            9
       1      0.1000E+01       3       2
       2      0.0000E+00       2       1
       3      0.1000E+01       3       2
       4      0.3000E+01       4       4
       5      0.4000E+01       1       1
```

```
         6      0.6000E+01      2       2
         7      0.2000E+01      3       3
         8      0.1000E+01      3       2
         9      0.1000E+01      3       2
Reordered elements
NNZ =    5
         1      0.4000E+01      1       1
         2      0.6000E+01      2       2
         3      0.4000E+01      3       2
         4      0.2000E+01      3       3
         5      0.3000E+01      4       4
```