NAG Fortran Library Manual
Mark 19

Volume 8

F08F – F11

F08F – Least-squares and Eigenvalue Problems (LAPACK) (cont'd from Volume 7)
F11 – Sparse Linear Algebra

NAG Fortran Library Manual, Mark 19

The copyright owner gives no warranties and makes no representations about the contents of this manual and specifically disclaims any implied warranties or merchantability or fitness for any purpose.

The copyright owner reserves the right to revise this manual and to make changes from time to time in its contents without notifying any person of such revisions or changes.

September 1999 ISBN 1-85206-169-3

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

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

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

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

NAG GmbH
Schleißheimerstraße 5
85748 Garching
Deutschland

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

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

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

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

[NP3390/19]

F08FCF (SSYEVD/DSYEVD) – 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

F08FCF (SSYEVD/DSYEVD) computes all the eigenvalues, and optionally all the eigenvectors, of a real symmetric matrix. If the eigenvectors are requested, then it uses a divide and conquer algorithm to compute eigenvalues and eigenvectors. However, if only eigenvalues are required, then it uses the Pal–Walker–Kahan variant of the QL or QR algorithm.

2 Specification

```
 SUBROUTINE F08FCF(JOB, UPLO, N, A, LDA, W, WORK, LWORK, IWORK,
1                  LIWORK, INFO)
 ENTRY      ssyevd(JOB, UPLO, N, A, LDA, W, WORK, LWORK, IWORK,
1                  LIWORK, INFO)
 INTEGER           N, LDA, LWORK, IWORK(*), LIWORK, INFO
 real              A(LDA,*), W(*), WORK(*)
 CHARACTER*1       JOB, UPLO
```

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 matrix A. In other words, it can compute the spectral factorization of A as

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

where Λ is a diagonal matrix whose diagonal elements are the eigenvalues λ_i, and Z is the orthogonal matrix whose columns are the eigenvectors z_i. Thus

$$Az_i = \lambda_i z_i \quad \text{for } i = 1, 2, \ldots, n.$$

4 References

[1] Golub G H and Van Loan C F (1996) *Matrix Computations* Johns Hopkins University Press (3rd Edition), Baltimore

5 Parameters

1: JOB — CHARACTER*1 *Input*

On entry: indicates whether eigenvectors are computed as follows:

if JOB = 'N', then only eigenvalues are computed;
if JOB = 'V', then eigenvalues and eigenvectors are computed.

Constraint: JOB = 'N' or 'V'.

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: 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 triangular part of A must be stored and the elements of the array below the diagonal are not referenced; if UPLO = 'L', the lower triangular part of A must be stored and the elements of the array above the diagonal are not referenced.

On exit: if JOB = 'V', then this is overwritten by the orthogonal matrix Z which contains the eigenvectors of A.

5: LDA — INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F08FCF (SSYEVD/DSYEVD) is called.

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

6: W(∗) — ***real*** array *Output*

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

On exit: the eigenvalues of the matrix A in ascending order.

7: WORK(∗) — ***real*** array *Output*

Note: the dimension of the array WORK must be at least LWORK.

On exit: if LWORK > 0, then WORK(1) contains the required minimal size of LWORK.

8: LWORK — INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08FCF (SSYEVD/DSYEVD) is called.

Constraints:

if $N \leq 1$, then LWORK ≥ 1;
if JOB = 'N' and $N > 1$, then LWORK $\geq 2 \times N + 1$;
if JOB = 'V' and $N > 1$, then LWORK $\geq 3 \times N^2 + (5+2k) \times N + 1$ where k is the smallest integer which satisfies $2^k \geq N$.

9: IWORK(∗) — INTEGER array *Input*

Note: the dimension of the array IWORK must be at least LIWORK.

On exit: if LIWORK > 0, then IWORK(1) contains the required minimal size of LIWORK.

10: LIWORK — INTEGER *Input*

On entry: the dimension of the array IWORK as declared in the (sub)program from which F08FCF (SSYEVD/DSYEVD) is called.

Constraints:

if $N \leq 1$, then LIWORK ≥ 1;
if JOB = 'N' and $N > 1$, then LIWORK ≥ 1;
if JOB = 'V' and $N > 1$, then LIWORK $\geq 5 \times N + 2$.

11: INFO — INTEGER *Output*

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

6 Error Indicators and Warnings

INFO < 0

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

INFO > 0

If INFO $= i$, then the algorithm failed to converge; i indicates the number of elements of an intermediate tridiagonal form which did not converge to zero.

7 Accuracy

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

$$||E||_2 = O(\epsilon)||A||_2,$$

and ϵ is the ***machine precision***.

8 Further Comments

The complex analogue of this routine is F08FQF (CHEEVD/ZHEEVD).

9 Example

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

$$A = \begin{pmatrix} 1.0 & 2.0 & 3.0 & 4.0 \\ 2.0 & 2.0 & 3.0 & 4.0 \\ 3.0 & 3.0 & 3.0 & 4.0 \\ 4.0 & 4.0 & 4.0 & 4.0 \end{pmatrix}.$$

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.

```
*     F08FCF Example Program Text.
*     Mark 19 Release. NAG Copyright 1999.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA
      PARAMETER        (NMAX=8,LDA=NMAX)
      INTEGER          LWORK, LIWORK
      PARAMETER        (LWORK=4*NMAX*NMAX,LIWORK=5*NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
      CHARACTER        JOB, UPLO
*     .. Local Arrays ..
      real             A(LDA,NMAX), W(NMAX), WORK(LWORK)
      INTEGER          IWORK(LIWORK)
*     .. External Subroutines ..
      EXTERNAL         ssyevd, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08FCF 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
*
            READ (NIN,*) JOB
*
*           Calculate all the eigenvalues and eigenvectors of A
*
            CALL ssyevd(JOB,UPLO,N,A,LDA,W,WORK,LWORK,IWORK,LIWORK,INFO)
*
            WRITE (NOUT,*)
            IF (INFO.GT.0) THEN
               WRITE (NOUT,*) 'Failure to converge.'
            ELSE
*
*              Print eigenvalues and eigenvectors
*
               WRITE (NOUT,*) 'Eigenvalues'
               WRITE (NOUT,99999) (W(I),I=1,N)
               WRITE (NOUT,*)
               IFAIL = 0
*
               CALL X04CAF('General',' ',N,N,A,LDA,'Eigenvectors',IFAIL)
*
            END IF
         END IF
         STOP
*
99999 FORMAT (3X,(8F8.4))
      END
```

9.2 Program Data

```
F08FCF Example Program Data
  4                           :Value of N
  'L'                         :Value of UPLO
  1.0
  2.0   2.0
  3.0   3.0   3.0
  4.0   4.0   4.0   4.0     :End of matrix A
  'V'                         :Value of JOB
```

9.3 Program Results

```
 F08FCF Example Program Results

 Eigenvalues
    -2.0531 -0.5146 -0.2943 12.8621

 Eigenvectors
          1       2       3       4
 1   0.7003 -0.5144 -0.2767 -0.4103
 2   0.3592  0.4851  0.6634 -0.4422
 3  -0.1569  0.5420 -0.6504 -0.5085
 4  -0.5965 -0.4543  0.2457 -0.6144
```

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 $\geq$ 0.

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

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

On entry: the n by n symmetric matrix A. If UPLO = 'U', the upper triangle of A must be stored and the elements of the array below the diagonal are not referenced; if UPLO = 'L', the lower triangle of A must be stored and the elements of the array above the diagonal are not referenced.

On exit: 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 ith 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 ε 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 ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

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

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

where ε is the ***machine precision***.

8. Further Comments

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

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 $\geq$ 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 $\geq$ 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 $\geq$ max(1,M) if SIDE = 'L',
LDA $\geq$ 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 $\geq$ 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 ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed result differs from the exact result by a matrix E such that

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

where ε is the ***machine precision***.

8. Further Comments

The total number of floating-point operations is approximately $2m^2n$ 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
```

F08FQF (CHEEVD/ZHEEVD) – 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

F08FQF (CHEEVD/ZHEEVD) computes all the eigenvalues, and optionally all the eigenvectors, of a complex Hermitian matrix. If the eigenvectors are requested, then it uses a divide and conquer algorithm to compute eigenvalues and eigenvectors. However, if only eigenvalues are required, then it uses the Pal–Walker–Kahan variant of the QL or QR algorithm.

2 Specification

```
      SUBROUTINE F08FQF(JOB, UPLO, N, A, LDA, W, WORK, LWORK, RWORK,
     1                  LRWORK, IWORK, LIWORK, INFO)
      ENTRY      cheevd(JOB, UPLO, N, A, LDA, W, WORK, LWORK, RWORK,
     1                  LRWORK, IWORK, LIWORK, INFO)
      INTEGER           N, LDA, LWORK, LRWORK, IWORK(*), LIWORK, INFO
      real              W(*), RWORK(*)
      complex           A(LDA,*), WORK(*)
      CHARACTER*1       JOB, UPLO
```

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 complex Hermitian matrix A. In other words, it can compute the spectral factorization of A as

$$A = Z\Lambda Z^H,$$

where Λ is a real diagonal matrix whose diagonal elements are the eigenvalues λ_i, and Z is the (complex) unitary matrix whose columns are the eigenvectors z_i. Thus

$$Az_i = \lambda_i z_i \quad \text{for } i = 1, 2, \ldots, n.$$

4 References

[1] Golub G H and Van Loan C F (1996) *Matrix Computations* Johns Hopkins University Press (3rd Edition), Baltimore

5 Parameters

1: JOB — CHARACTER*1 *Input*

On entry: indicates whether eigenvectors are computed as follows:

if JOB = 'N', then only eigenvalues are computed;
if JOB = 'V', then eigenvalues and eigenvectors are computed.

Constraint: JOB = 'N' or 'V'.

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: 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 triangular part of A must be stored and the elements of the array below the diagonal are not referenced; if UPLO = 'L', the lower triangular part of A must be stored and the elements of the array above the diagonal are not referenced.

On exit: if JOB = 'V', then this is overwritten by the unitary matrix Z which contains the eigenvectors of A.

5: LDA — INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F08FQF (CHEEVD/ZHEEVD) is called.

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

6: W($*$) — ***real*** array *Output*

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

On exit: the eigenvalues of the matrix A in ascending order.

7: WORK($*$) — ***complex*** array *Output*

Note: the dimension of the array WORK must be at least LWORK.

On exit: if LWORK > 0, then the real part of WORK(1) contains the required minimal size of LWORK.

8: LWORK — INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08FQF (CHEEVD/ZHEEVD) is called.

Constraints:

if $N \leq 1$, then LWORK ≥ 1;
if JOB = 'N' and $N > 1$, then LWORK $\geq N + 1$;
if JOB = 'V' and $N > 1$, then LWORK $\geq N^2 + 2 \times N$.

9: RWORK($*$) — ***real*** array *Input*

Note: the dimension of the array RWORK must be at least LRWORK.

On exit: if LRWORK > 0, then RWORK(1) contains the required minimal size of LRWORK.

10: LRWORK — INTEGER *Input*

On entry: the dimension of the array LRWORK as declared in the (sub)program from which F08FQF (CHEEVD/ZHEEVD) is called.

Constraints:

if $N \leq 1$, then LWORK ≥ 1;
if JOB = 'N' and $N > 1$, then LWORK $\geq N$;
if JOB = 'V' and $N > 1$, then LWORK $\geq 3 \times N^2 + (4{+}2k) \times N + 1$ where k is the smallest integer which satisfies $2^k \geq N$.

11: IWORK(*) — INTEGER array *Input*

Note: the dimension of the array IWORK must be at least LIWORK.

On exit: if LIWORK > 0, then IWORK(1) contains the required minimal size of LIWORK.

12: LIWORK — INTEGER *Input*

On entry: the dimension of the array IWORK as declared in the (sub)program from which F08FQF (CHEEVD/ZHEEVD) is called.

Constraints:

if N $\leq$ 1, then LIWORK $\geq$ 1,
if JOB = 'N' and N > 1, then LIWORK $\geq$ 1,
if JOB = 'V' and N > 1, then LIWORK $\geq 5 \times$ N + 2.

13: INFO — INTEGER *Output*

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

6 Error Indicators and Warnings

INFO < 0

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

INFO > 0

If INFO = i, then the algorithm failed to converge; i indicates the number of elements of an intermediate tridiagonal form which did not converge to zero.

7 Accuracy

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

$$\|E\|_2 = O(\epsilon)\|A\|_2,$$

and ϵ is the ***machine precision***.

8 Further Comments

The real analogue of this routine is F08FCF (SSYEVD/DSYEVD).

9 Example

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

$$A = \begin{pmatrix} 1.0+0.0i & 2.0+1.0i & 3.0+1.0i & 4.0+1.0i \\ 2.0-1.0i & 2.0+0.0i & 3.0+2.0i & 4.0+2.0i \\ 3.0-1.0i & 3.0-2.0i & 3.0+0.0i & 4.0+3.0i \\ 4.0-1.0i & 4.0-2.0i & 4.0-3.0i & 4.0+0.0i \end{pmatrix}.$$

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.

```
*     F08FQF Example Program Text.
*     Mark 19 Release. NAG Copyright 1999.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDA
      PARAMETER        (NMAX=8,LDA=NMAX)
      INTEGER          LWORK, LIWORK, LRWORK
      PARAMETER        (LWORK=NMAX*NMAX+2*NMAX,LIWORK=2+5*NMAX,
     +                 LRWORK=4*NMAX*NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
      CHARACTER        JOB, UPLO
*     .. Local Arrays ..
      complex          A(LDA,NMAX), WORK(LWORK)
      real             RWORK(LRWORK), W(NMAX)
      INTEGER          IWORK(LIWORK)
*     .. External Subroutines ..
      EXTERNAL         X04DAF, cheevd
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08FQF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
         READ (NIN,*) UPLO
*
*        Read A from data file
*
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
         END IF
*
         READ (NIN,*) JOB
*
*        Calculate all the eigenvalues and eigenvectors of A
*
         CALL cheevd(JOB,UPLO,N,A,LDA,W,WORK,LWORK,RWORK,LRWORK,IWORK,
     +               LIWORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.GT.0) THEN
            WRITE (NOUT,*) 'Failure to converge.'
         ELSE
*
*           Print eigenvalues and eigenvectors
*
            WRITE (NOUT,*) 'Eigenvalues'
            DO 20 I = 1, N
               WRITE (NOUT,99999) I, W(I)
   20       CONTINUE
```

```
               WRITE (NOUT,*)
               IFAIL = 0
*
               CALL X04DAF('General',' ',N,N,A,LDA,'Eigenvectors',IFAIL)
*
            END IF
         END IF
         STOP
*
99999 FORMAT (3X,I5,5X,2F8.4)
         END
```

9.2 Program Data

```
F08FQF Example Program Data
  4                                                      :Value of N
  'L'                                                    :Value of UPLO
  (1.0, 0.0)
  (2.0, 1.0)  (2.0, 0.0)
  (3.0, 1.0)  (3.0, 2.0)  (3.0, 0.0)
  (4.0, 1.0)  (4.0, 2.0)  (4.0, 3.0)  (4.0, 0.0)        :End of matrix A
  'V'                                                    :Value of JOB
```

9.3 Program Results

```
 F08FQF Example Program Results

 Eigenvalues
        1      -4.2443
        2      -0.6886
        3       1.1412
        4      13.7916

 Eigenvectors
           1       2       3       4
 1    0.4836  0.6470 -0.4456 -0.3859
      0.0000  0.0000  0.0000  0.0000

 2    0.2912 -0.4984 -0.0230 -0.4441
     -0.3618 -0.1130 -0.5702  0.0156

 3   -0.3163  0.2949  0.5331 -0.5173
     -0.3696  0.3165  0.1317 -0.0844

 4   -0.4447 -0.2241 -0.3510 -0.5277
      0.3406 -0.2878  0.2261 -0.3168
```

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 $\geq$ 0.

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

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

On entry: the n by n Hermitian 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 ith 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

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

$c(n)$ is a modestly increasing function of n, and ε 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 ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

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

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

where ε is the ***machine precision***.

8. Further Comments

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

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 $\geq$ 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 $\geq$ 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 $\geq$ max(1,M) if SIDE = 'L',
LDA $\geq$ 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 $\geq$ 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 ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed result differs from the exact result by a matrix E such that

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

where ε is the ***machine precision***.

8. Further Comments

The total number of real floating-point operations is approximately $8m^2n$ 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)
```

F08GCF (SSPEVD/DSPEVD) – 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

F08GCF (SSPEVD/DSPEVD) computes all the eigenvalues, and optionally all the eigenvectors, of a real symmetric matrix held in packed storage. If the eigenvectors are requested, then it uses a divide and conquer algorithm to compute eigenvalues and eigenvectors. However, if only eigenvalues are required, then it uses the Pal–Walker–Kahan variant of the QL or QR algorithm.

2 Specification

```
      SUBROUTINE F08GCF(JOB, UPLO, N, AP, W, Z, LDZ, WORK, LWORK, IWORK,
     1                  LIWORK, INFO)
      ENTRY      sspevd(JOB, UPLO, N, AP, W, Z, LDZ, WORK, LWORK, IWORK,
     1                  LIWORK, INFO)
      INTEGER           N, LDZ, LWORK, IWORK(*), LIWORK, INFO
      real              AP(*), W(*), Z(LDZ,*), WORK(*)
      CHARACTER*1       JOB, UPLO
```

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 matrix A (held in packed storage). In other words, it can compute the spectral factorization of A as

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

where Λ is a diagonal matrix whose diagonal elements are the eigenvalues λ_i, and Z is the orthogonal matrix whose columns are the eigenvectors z_i. Thus

$$Az_i = \lambda_i z_i \quad \text{for } i = 1, 2, \ldots, n.$$

4 References

[1] Golub G H and Van Loan C F (1996) *Matrix Computations* Johns Hopkins University Press (3rd Edition), Baltimore

5 Parameters

1: JOB — CHARACTER*1 *Input*

On entry: indicates whether eigenvectors are computed as follows:

if JOB = 'N', then only eigenvalues are computed;
if JOB = 'V', then eigenvalues and eigenvectors are computed.

Constraint: JOB = 'N' or 'V'.

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: AP(∗) — ***real*** array *Input/Output*

Note: the dimension of the array AP must be at least max(1,N∗(N+1)/2).

On entry: the n by n symmetric matrix A, packed by columns. More precisely, if UPLO = 'U', the upper triangle of A must be stored with element a_{ij} in AP$(i + j(j-1)/2)$ for $i \leq j$; if UPLO = 'L', the lower triangle of A must be stored with element a_{ij} in AP$(i + (2n-j)(j-1)/2)$ for $i \geq j$.

On exit: A is overwritten by the values generated during the reduction to tridiagonal form. The elements of the diagonal and the off-diagonal of the tridiagonal matrix overwrites the corresponding elements of A.

5: W(∗) — ***real*** array *Output*

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

On exit: the eigenvalues of the matrix A in ascending order.

6: Z(LDZ, ∗) — ***real*** array *Output*

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

On exit: if JOB = 'V', then this is overwritten by the orthogonal matrix Z which contains the eigenvectors of A.

Z is not referenced if JOB = 'N'.

7: LDZ — INTEGER *Input*

On entry: the leading dimension of the array Z as declared in the (sub)program from which F08GCF (SSPEVD/DSPEVD) is called.

Constraints:

LDZ $\geq$ max(1,N) if JOB = 'V';
LDZ $\geq$ 1 if JOB = 'N'.

8: WORK(∗) — ***real*** array *Workspace*

Note: the dimension of the array WORK must be at least LWORK.

On exit: if LWORK > 0, then WORK(1) contains the required minimal size of LWORK.

9: LWORK — INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08GCF (SSPEVD/DSPEVD) is called.

Constraints:

if N $\leq$ 1, then LWORK $\geq$ 1;
if JOB = 'N' and N > 1, then LWORK $\geq 2 \times$ N;
if JOB = 'V' and N > 1, then LWORK $\geq 2 \times \text{N}^2 + (5{+}2k) \times \text{N} + 1$ where k is the smallest integer which satisfies $2^k \geq$ N.

10: IWORK(∗) — INTEGER array *Workspace*

Note: the dimension of the array IWORK must be at least LIWORK.

On exit: if LIWORK > 0, then IWORK(1) contains the required minimal size of LIWORK.

11: LIWORK — INTEGER *Input*

On entry: the dimension of the array IWORK as declared in the (sub)program from which F08GCF (SSPEVD/DSPEVD) is called.

Constraints:

if N $\leq$ 1, then LIWORK $\geq$ 1;
if JOB = 'N' and N > 1, then LIWORK $\geq$ 1;
if JOB = 'V' and N > 1, then LIWORK $\geq 5 \times$ N + 2.

12: INFO — INTEGER *Output*

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

6 Error Indicators and Warnings

INFO < 0

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

INFO > 0

If INFO $= i$, then the algorithm failed to converge; i indicates the number of elements of an intermediate tridiagonal form which did not converge to zero.

7 Accuracy

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

$$||E||_2 = O(\epsilon)||A||_2,$$

and ϵ is the ***machine precision***.

8 Further Comments

The complex analogue of this routine is F08GQF (CHPEVD/ZHPEVD).

9 Example

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

$$A = \begin{pmatrix} 1.0 & 2.0 & 3.0 & 4.0 \\ 2.0 & 2.0 & 3.0 & 4.0 \\ 3.0 & 3.0 & 3.0 & 4.0 \\ 4.0 & 4.0 & 4.0 & 4.0 \end{pmatrix}.$$

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.

```
*     F08GCF Example Program Text.
*     Mark 19 Release. NAG Copyright 1999.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDZ
      PARAMETER        (NMAX=8,LDZ=NMAX)
      INTEGER          LWORK, LIWORK
      PARAMETER        (LWORK=3*NMAX*NMAX,LIWORK=5*NMAX+2)
      INTEGER          MMAX
      PARAMETER        (MMAX=NMAX*(NMAX+1)/2)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, K, N
      CHARACTER        JOB, UPLO
*     .. Local Arrays ..
      real             AP(MMAX), W(NMAX), WORK(LWORK), Z(LDZ,NMAX)
      INTEGER          IWORK(LIWORK)
*     .. External Subroutines ..
      EXTERNAL         sspevd, X04CAF
*     .. Intrinsic Functions ..
      INTRINSIC        MAX
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08GCF 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

         K = 1
         IF (UPLO.EQ.'U') THEN
            DO 20 J = 1, N
               DO 20 I = 1, J
                  AP(K) = MAX(I,J)
                  K = K + 1
   20       CONTINUE
         ELSE IF (UPLO.EQ.'L') THEN
            DO 40 J = 1, N
               DO 40 I = J, N
                  AP(K) = MAX(I,J)
                  K = K + 1
   40       CONTINUE
         END IF
*
         READ (NIN,*) JOB
*
*        Calculate all the eigenvalues and eigenvectors of A
*
         CALL sspevd(JOB,UPLO,N,AP,W,Z,LDZ,WORK,LWORK,IWORK,LIWORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.GT.0) THEN
            WRITE (NOUT,*) 'Failure to converge.'
         ELSE
*
*           Print eigenvalues and eigenvectors
```

```
*
            WRITE (NOUT,*) 'Eigenvalues'
            WRITE (NOUT,99999) (W(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

```
F08GCF Example Program Data
  4                           :Value of N
  'L'                         :Value of UPLO
  1.0
  2.0   2.0
  3.0   3.0   3.0
  4.0   4.0   4.0   4.0     :End of matrix A
  'V'                         :Value of JOB
```

9.3 Program Results

```
 F08GCF Example Program Results

 Eigenvalues
   -2.0531 -0.5146 -0.2943 12.8621

 Eigenvectors
          1       2       3       4
 1   0.7003 -0.5144 -0.2767 -0.4103
 2   0.3592  0.4851  0.6634 -0.4422
 3  -0.1569  0.5420 -0.6504 -0.5085
 4  -0.5965 -0.4543  0.2457 -0.6144
```

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($i+(2n-j)(j-1)/2$) 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 ith 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 ε 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 ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

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

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

where ε is the ***machine precision***.

8. Further Comments

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

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 $\geq$ 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 $\geq$ 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 $\geq$ 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 ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed result differs from the exact result by a matrix E such that

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

where ε is the ***machine precision***.

8. Further Comments

The total number of floating-point operations is approximately $2m^2n$ 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
```

F08GQF (CHPEVD/ZHPEVD) – 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

F08GQF (CHPEVD/ZHPEVD) computes all the eigenvalues, and optionally all the eigenvectors, of a complex Hermitian matrix held in packed storage. If the eigenvectors are requested, then it uses a divide and conquer algorithm to compute eigenvalues and eigenvectors. However, if only eigenvalues are required, then it uses the Pal–Walker–Kahan variant of the QL or QR algorithm.

2 Specification

```
   SUBROUTINE F08GQF(JOB, UPLO, N, AP, W, Z, LDZ, WORK, LWORK, RWORK,
  1                  LRWORK, IWORK, LIWORK, INFO)
   ENTRY       chpevd(JOB, UPLO, N, AP, W, Z, LDZ, WORK, LWORK, RWORK,
  1                  LRWORK, IWORK, LIWORK, INFO)
   INTEGER           N, LDZ, LWORK, LRWORK, IWORK(*), LIWORK, INFO
   real              W(*), RWORK(*)
   complex           AP(*), Z(LDZ,*), WORK(*)
   CHARACTER*1       JOB, UPLO
```

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 complex Hermitian matrix A (held in packed storage). In other words, it can compute the spectral factorization of A as

$$A = Z\Lambda Z^H,$$

where Λ is a real diagonal matrix whose diagonal elements are the eigenvalues λ_i, and Z is the (complex) unitary matrix whose columns are the eigenvectors z_i. Thus

$$Az_i = \lambda_i z_i \quad \text{for } i = 1, 2, \ldots, n.$$

4 References

[1] Golub G H and Van Loan C F (1996) *Matrix Computations* Johns Hopkins University Press (3rd Edition), Baltimore

5 Parameters

1: JOB — CHARACTER*1 *Input*

On entry: indicates whether eigenvectors are computed as follows:

if JOB = 'N', then only eigenvalues are computed;
if JOB = 'V', then eigenvalues and eigenvectors are computed.

Constraint: JOB = 'N' or 'V'.

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: AP(*) — ***complex*** array *Input/Output*

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

On entry: the n by n Hermitian matrix A, packed by columns. More precisely, if UPLO = 'U', the upper triangle of A must be stored with element a_{ij} in AP$(i+j(j-1)/2)$ for $i \leq j$; if UPLO = 'L', the lower triangle of A must be stored with element a_{ij} in AP$(i+(2n-j)(j-1)/2)$ for $i \geq j$.

On exit: A is overwritten by the values generated during the reduction to tridiagonal form. The elements of the diagonal and the off-diagonal of the tridiagonal matrix overwrites the corresponding elements of A.

5: W(*) — ***real*** array *Output*

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

On exit: the eigenvalues of the matrix A in ascending order.

6: Z(LDZ, *) — ***complex*** array *Output*

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

On exit: if JOB = 'V', then this is overwritten by the unitary matrix Z which contains the eigenvectors of A.

Z is not referenced if JOB = 'N'.

7: LDZ — INTEGER *Input*

On entry: the leading dimension of the array Z as declared in the (sub)program from which F08GQF (CHPEVD/ZHPEVD) is called.

Constraints:

LDZ $\geq$ max(1,N) if JOB = 'V';
LDZ $\geq$ 1 if JOB = 'N'.

8: WORK(*) — ***complex*** array *Workspace*

Note: the dimension of the array WORK must be at least LWORK.

On exit: if LWORK > 0, then WORK(1) contains the required minimal size of LWORK.

9: LWORK — INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08GQF (CHPEVD/ZHPEVD) is called.

Constraints:

if N $\leq$ 1, then LWORK $\geq$ 1;
if JOB = 'N' and N > 1, then LWORK $\geq$ N;
if JOB = 'V' and N > 1, then LWORK $\geq$ 2 × N.

10: RWORK(*) — ***real*** array *Workspace*

Note: the dimension of the array RWORK must be at least LRWORK.

On exit: if LRWORK > 0, then RWORK(1) contains the required minimal size of LRWORK.

11: LRWORK — INTEGER *Input*

On entry: the dimension of the array RWORK as declared in the (sub)program from which F08GQF (CHPEVD/ZHPEVD) is called.

Constraints:

if N $\leq$ 1, then LRWORK $\geq$ 1;
if JOB = 'N' and N > 1, then LWORK $\geq$ N;
if JOB = 'V' and N > 1, then LRWORK $\geq 3 \times \mathrm{N}^2 + (4+2k) \times \mathrm{N} + 1$ where k is the smallest integer which satisfies $2^k \geq \mathrm{N}$.

12: IWORK(*) — INTEGER array *Workspace*

Note: the dimension of the array IWORK must be at least LIWORK.

On exit: if LIWORK > 0, then IWORK(1) contains the required minimal size of LIWORK.

13: LIWORK — INTEGER *Input*

On entry: the dimension of the array IWORK as declared in the (sub)program from which F08GQF (CHPEVD/ZHPEVD) is called.

Constraints:

if N $\leq$ 1, then LIWORK $\geq$ 1;
if JOB = 'N' and N > 1, then LIWORK $\geq$ 1;
if JOB = 'V' and N > 1, then LIWORK $\geq 5 \times \mathrm{N} + 2$.

14: INFO — INTEGER *Output*

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

6 Error Indicators and Warnings

INFO < 0

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

INFO > 0

If INFO $= i$, then the algorithm failed to converge; i indicates the number of elements of an intermediate tridiagonal form which did not converge to zero.

7 Accuracy

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

$$\|E\|_2 = O(\epsilon)\|A\|_2,$$

and ϵ is the ***machine precision***.

8 Further Comments

The real analogue of this routine is F08GCF (SSPEVD/DSPEVD).

9 Example

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

$$A = \begin{pmatrix} 1.0+0.0i & 2.0+1.0i & 3.0+1.0i & 4.0+1.0i \\ 2.0-1.0i & 2.0+0.0i & 3.0+2.0i & 4.0+2.0i \\ 3.0-1.0i & 3.0-2.0i & 3.0+0.0i & 4.0+3.0i \\ 4.0-1.0i & 4.0-2.0i & 4.0-3.0i & 4.0+0.0i \end{pmatrix}.$$

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.

```
*     F08GQF Example Program Text.
*     Mark 19 Release. NAG Copyright 1999.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDZ
      PARAMETER        (NMAX=8,LDZ=NMAX)
      INTEGER          LWORK, LIWORK, LRWORK, MMAX
      PARAMETER        (LWORK=NMAX*NMAX+2*NMAX,LIWORK=2+5*NMAX,
     +                 LRWORK=4*NMAX*NMAX,MMAX=NMAX*(NMAX+1)/2)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N
      CHARACTER        JOB, UPLO
*     .. Local Arrays ..
      complex          AP(MMAX), WORK(LWORK), Z(LDZ,NMAX)
      real             RWORK(LRWORK), W(NMAX)
      INTEGER          IWORK(LIWORK)
*     .. External Subroutines ..
      EXTERNAL         X04DAF, chpevd
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08GQF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
         READ (NIN,*) UPLO
*
*        Read A from data file
*
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
*
         READ (NIN,*) JOB
*
*        Calculate all the eigenvalues and eigenvectors of A
*
         CALL chpevd(JOB,UPLO,N,AP,W,Z,LDZ,WORK,LWORK,RWORK,LRWORK,
     +               IWORK,LIWORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.GT.0) THEN
            WRITE (NOUT,*) 'Failure to converge.'
         ELSE
*
*           Print eigenvalues and eigenvectors
*
            WRITE (NOUT,*) 'Eigenvalues'
            DO 20 I = 1, N
               WRITE (NOUT,99999) I, W(I)
   20       CONTINUE
```

```
            WRITE (NOUT,*)
            IFAIL = 0
*
            CALL X04DAF('General',' ',N,N,Z,LDZ,'Eigenvectors',IFAIL)
*
         END IF
      END IF
      STOP
*
99999 FORMAT (3X,I5,5X,2F8.4)
      END
```

9.2 Program Data

```
F08GQF Example Program Data
  4                                                   :Value of N
  'L'                                                 :Value of UPLO
  (1.0, 0.0)
  (2.0, 1.0)  (2.0, 0.0)
  (3.0, 1.0)  (3.0, 2.0)  (3.0, 0.0)
  (4.0, 1.0)  (4.0, 2.0)  (4.0, 3.0)  (4.0, 0.0)      :End of matrix A
  'V'                                                 :Value of JOB
```

9.3 Program Results

```
F08GQF Example Program Results

Eigenvalues
      1      -4.2443
      2      -0.6886
      3       1.1412
      4      13.7916

Eigenvectors
          1       2       3       4
1    0.4836  0.6470 -0.4456 -0.3859
     0.0000  0.0000  0.0000  0.0000

2    0.2912 -0.4984 -0.0230 -0.4441
    -0.3618 -0.1130 -0.5702  0.0156

3   -0.3163  0.2949  0.5331 -0.5173
    -0.3696  0.3165  0.1317 -0.0844

4   -0.4447 -0.2241 -0.3510 -0.5277
     0.3406 -0.2878  0.2261 -0.3168
```

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 $\geq$ 0.

3: AP(*) – ***complex*** array. *Input/Output*

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

On entry: the n by n Hermitian matrix A, packed by columns. More precisely, if UPLO = 'U', the upper triangle of A must be stored with element a_{ij} in AP(i+j(j–1)/2) for $i \leq j$; if UPLO = 'L', the lower triangle of A must be stored with element a_{ij} in AP(i+(2n–j)(j–1)/2) for $i \geq j$.

On exit: A is overwritten by 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 ith 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 ε 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 $\geq$ 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 ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

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

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

where ε is the ***machine precision.***

8. Further Comments

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

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

7. Accuracy

The computed result differs from the exact result by a matrix E such that

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

where ε is the ***machine precision***.

8. Further Comments

The total number of real floating-point operations is approximately $8m^2n$ 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)
```

F08HCF (SSBEVD/DSBEVD) – 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

F08HCF (SSBEVD/DSBEVD) computes all the eigenvalues, and optionally all the eigenvectors, of a real symmetric band matrix. If the eigenvectors are requested, then it uses a divide and conquer algorithm to compute eigenvalues and eigenvectors. However, if only eigenvalues are required, then it uses the Pal–Walker–Kahan variant of the QL or QR algorithm.

2 Specification

```
      SUBROUTINE F08HCF(JOB, UPLO, N, KD, AB, LDAB, W, Z, LDZ, WORK,
     1                  LWORK, IWORK, LIWORK, INFO)
      ENTRY      ssbevd(JOB, UPLO, N, KD, AB, LDAB, W, Z, LDZ, WORK,
     1                  LWORK, IWORK, LIWORK, INFO)
      INTEGER           N, KD, LDAB, LDZ, LWORK, IWORK(*), LIWORK, INFO
      real              AB(LDAB,*), W(*), Z(LDZ,*), WORK(*)
      CHARACTER*1       JOB, UPLO
```

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 band matrix A. In other words, it can compute the spectral factorization of A as

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

where Λ is a diagonal matrix whose diagonal elements are the eigenvalues λ_i, and Z is the orthogonal matrix whose columns are the eigenvectors z_i. Thus

$$Az_i = \lambda_i z_i \quad \text{for } i = 1,2,\ldots,n.$$

4 References

[1] Golub G H and Van Loan C F (1996) *Matrix Computations* Johns Hopkins University Press (3rd Edition), Baltimore

5 Parameters

1: JOB — CHARACTER*1 *Input*

On entry: indicates whether eigenvectors are computed as follows:

if JOB = 'N', then only eigenvalues are computed;
if JOB = 'V', then eigenvalues and eigenvectors are computed.

Constraint: JOB = 'N' or 'V'.

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 $\geq$ 0.

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

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

On entry: the upper or the lower triangle of the n by n symmetric band matrix A, stored in the first KD+1 rows of the array AB. More precisely, the jth column of A is stored in the jth column of the array AB as follows:

if UPLO = 'U', then AB(KD+1+$i-j,j$) = a_{ij} for max(1,j−KD) $\leq i \leq j$;
if UPLO = 'L', then AB($1+i-j,j$) = a_{ij} for $j \leq i \leq$ min(n,j+KD).

On exit: A is overwritten by the values generated during the reduction to tridiagonal form. If UPLO = 'U', the first superdiagonal and the diagonal of the tridiagonal matrix are returned in rows KD and KD+1 of the array AB, respectively, and if UPLO = 'L' then the diagonal and the first subdiagonal of the tridiagonal matrix are returned in the first two rows of the array AB.

6: LDAB — INTEGER *Input*

On entry: the leading dimension of the array AB.

Constraint: LDAB $\geq$ KD+1.

7: W(*) — ***real*** array *Output*

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

On exit: the eigenvalues of the matrix A in ascending order.

8: Z(LDZ, *) — ***real*** array *Output*

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

On exit: if JOB = 'V', then this is overwritten by the orthogonal matrix Z which contains the eigenvectors of A. The ith column of Z contains the eigenvector which corresponds to the eigenvalue W(i).

If JOB = 'N', then Z is not referenced.

9: LDZ — INTEGER *Input*

On entry: the leading dimension of the array Z as declared in the (sub)program from which F08HCF (SSBEVD/DSBEVD) is called.

Constraints:

LDZ $\geq$ max(1,N) if JOB = 'V';
LDZ $\geq$ 1 if JOB = 'N'.

10: WORK(*) — ***real*** array *Workspace*

Note: the dimension of the array WORK must be at least LWORK.

On exit: if LWORK > 0, then WORK(1) contains the required minimal size of LWORK.

11: LWORK — INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08HCF (SSBEVD/DSBEVD) is called.

Constraints:

if N $\leq$ 1, then LWORK $\geq$ 1;
if JOB = 'N' and N > 1, then LWORK $\geq 2 \times$ N;
if JOB = 'V' and N > 1, then LWORK $\geq 3 \times \mathrm{N}^2 + (4+2k) \times \mathrm{N} + 1$ where k is the smallest integer which satisfies $2^k \geq \mathrm{N}$.

12: IWORK(*) — INTEGER array *Workspace*

Note: the dimension of the array IWORK must be at least LIWORK.

On exit: if LIWORK > 0, then IWORK(1) contains the required minimal size of LIWORK.

13: LIWORK — INTEGER *Input*

On entry: the dimension of the array IWORK as declared in the (sub)program from which F08HCF (SSBEVD/DSBEVD) is called.

Constraints:

if JOB = 'N' or N $\leq$ 1, then LIWORK $\geq$ 1;
if JOB = 'V' and N > 1, then LIWORK $\geq 5 \times$ N + 2.

14: INFO — INTEGER *Output*

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

6 Error Indicators and Warnings

INFO < 0

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

INFO > 0

If INFO = i, then the algorithm failed to converge; i indicates the number of elements of an intermediate tridiagonal form which did not converge to zero.

7 Accuracy

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

$$\|E\|_2 = O(\epsilon)\|A\|_2,$$

and ϵ is the ***machine precision***.

8 Further Comments

The complex analogue of this routine is F08HQF (CHBEVD/ZHBEVD).

9 Example

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

$$A = \begin{pmatrix} 1.0 & 2.0 & 3.0 & 0.0 & 0.0 \\ 2.0 & 2.0 & 3.0 & 4.0 & 0.0 \\ 3.0 & 3.0 & 3.0 & 4.0 & 5.0 \\ 0.0 & 4.0 & 4.0 & 4.0 & 5.0 \\ 0.0 & 0.0 & 5.0 & 5.0 & 5.0 \end{pmatrix}.$$

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.

```
*     F08HCF Example Program Text.
*     Mark 19 Release. NAG Copyright 1999.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, KDMAX, LDAB, LDZ
      PARAMETER        (NMAX=9,KDMAX=4,LDAB=KDMAX,LDZ=NMAX)
      INTEGER          LWORK, LIWORK
      PARAMETER        (LWORK=4*NMAX*NMAX,LIWORK=5*NMAX+2)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, KD, N
      CHARACTER        JOB, UPLO
*     .. Local Arrays ..
      real             AB(LDAB,NMAX), W(NMAX), WORK(LWORK), Z(LDZ,NMAX)
      INTEGER          IWORK(LIWORK)
*     .. External Subroutines ..
      EXTERNAL         ssbevd, X04CAF
*     .. Intrinsic Functions ..
      INTRINSIC        MAX, MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08HCF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, KD
      IF (N.LE.NMAX) 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
*
         READ (NIN,*) JOB
*
*        Calculate all the eigenvalues and eigenvectors of A
*
         CALL ssbevd(JOB,UPLO,N,KD,AB,LDAB,W,Z,LDZ,WORK,LWORK,IWORK,
     +               LIWORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.GT.0) THEN
            WRITE (NOUT,*) 'Failure to converge.'
         ELSE
*
*           Print eigenvalues and eigenvectors
*
```

```
            WRITE (NOUT,*) 'Eigenvalues'
            WRITE (NOUT,99999) (W(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

```
F08HCF Example Program Data
  5  2                         :Values of N and KD
  'L'                          :Value of UPLO
  1.0  2.0  3.0
  2.0  2.0  3.0  4.0
  3.0  3.0  3.0  4.0  5.0
       4.0  4.0  4.0  5.0
            5.0  5.0  5.0      :End of matrix A
  'V'                          :Value of JOB
```

9.3 Program Results

```
F08HCF Example Program Results

Eigenvalues
   -3.2474 -2.6633  1.7511  4.1599 14.9997

Eigenvectors
          1       2       3       4       5
1    0.0394 -0.6238 -0.5635  0.5165  0.1582
2    0.5721  0.2575  0.3896  0.5955  0.3161
3   -0.4372  0.5900 -0.4008  0.1470  0.5277
4   -0.4424 -0.4308  0.5581 -0.0470  0.5523
5    0.5332 -0.1039 -0.2421 -0.5956  0.5400
```

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.

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,
  1                  INFO)
   ENTRY      ssbtrd(VECT, UPLO, N, KD, AB, LDAB, D, E, Q, LDQ, WORK,
  1                  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: $T = Q^T AQ$. 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 (1984) Banded eigenvalue solvers on vector machines *ACM Trans. Math. Software* **10** 73–86

[2] Parlett B N (1980) *The Symmetric Eigenvalue Problem* Prentice–Hall

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 (and the array Q must contain a matrix on entry);
if VECT = 'U', then Q is updated (and the array Q must contain a matrix on entry);
if VECT = 'N', then Q is not required.

Constraint: VECT = 'V', 'U' 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 $\geq$ 0.

5: AB(LDAB,$*$) — ***real*** array *Input/Output*

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

On entry: the 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 $\max(1,j-k) \leq i \leq j$; if UPLO = 'L', the elements of the lower triangle of A within the band must be stored with element a_{ij} in AB$(1+i-j,j)$ for $j \leq i \leq \min(n,j+k)$.

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 $\geq$ 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 *Input/Output*

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

On entry: if VECT = 'U', Q must contain the matrix formed in a previous stage of the reduction (for example, the reduction of a banded symmetric-definite generalized eigenproblem); otherwise Q need not be set.

On exit: if VECT = 'V' or 'U', the n by n matrix Q.

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 $\geq$ max(1,N) if VECT = 'V' or 'U';
LDQ $\geq$ 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 ith 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

$$\| E \|_2 \le c(n)\epsilon \| A \|_2,$$

$c(n)$ is a modestly increasing function of n, and ϵ 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

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

where ϵ 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
```

F08HQF (CHBEVD/ZHBEVD) – 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

F08HQF (CHBEVD/ZHBEVD) computes all the eigenvalues, and optionally all the eigenvectors, of a complex Hermitian band matrix. If the eigenvectors are requested, then it uses a divide and conquer algorithm to compute eigenvalues and eigenvectors. However, if only eigenvalues are required, then it uses the Pal–Walker–Kahan variant of the QL or QR algorithm.

2 Specification

```
 SUBROUTINE F08HQF(JOB, UPLO, N, KD, AB, LDAB, W, Z, LDZ, WORK,
1                  LWORK, RWORK, LRWORK, IWORK, LIWORK, INFO)
 ENTRY      chbevd(JOB, UPLO, N, KD, AB, LDAB, W, Z, LDZ, WORK,
1                  LWORK, RWORK, LRWORK, IWORK, LIWORK, INFO)
 INTEGER           N, KD, LDAB, LDZ, LWORK, LRWORK, IWORK(*),
1                  LIWORK, INFO
 real              W(*), RWORK(*),
 complex           AB(LDAB,*), Z(LDZ,*), WORK(*)
 CHARACTER*1       JOB, UPLO
```

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 complex Hermitian band matrix A. In other words, it can compute the spectral factorization of A as

$$A = Z\Lambda Z^H,$$

where Λ is a real diagonal matrix whose diagonal elements are the eigenvalues λ_i, and Z is the (complex) unitary matrix whose columns are the eigenvectors z_i. Thus

$$Az_i = \lambda_i z_i \quad \text{for } i = 1,2,\ldots,n.$$

4 References

[1] Golub G H and Van Loan C F (1996) *Matrix Computations* Johns Hopkins University Press (3rd Edition), Baltimore

5 Parameters

1: JOB — CHARACTER*1 *Input*

On entry: indicates whether eigenvectors are computed as follows:

if JOB = 'N', then only eigenvalues are computed;
if JOB = 'V', then eigenvalues and eigenvectors are computed.

Constraint: JOB = 'N' or 'V'.

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 $\geq$ 0.

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

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

On entry: the upper or the lower triangle of the n by n Hermitian band matrix A, stored in the first KD+1 rows of the array AB. More precisely, the jth column of A is stored in the jth column of the array AB as follows:

if UPLO = 'U', AB(KD+1+$i-j$,j) = a_{ij} for max(1,j−KD) $\leq i \leq j$;
if UPLO = 'L', AB(1+$i-j$,j) = a_{ij} for $j \leq i \leq$ min(n,j+KD).

On exit: A is overwritten by the values generated during the reduction to tridiagonal form. If UPLO = 'U', the first superdiagonal and the diagonal of the tridiagonal matrix are returned in rows KD and KD + 1 of the array AB, respectively, and if UPLO = 'L' then the diagonal and the first subdiagonal of the tridiagonal matrix are returned in the first two rows of the array AB.

6: LDAB — INTEGER *Input*

On entry: the leading dimension of the array AB.

Constraint: LDAB $\geq$ KD+1.

7: W(*) — ***real*** array *Output*

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

On exit: the eigenvalues of the matrix A in ascending order.

8: Z(LDZ, *) — ***complex*** array *Output*

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

On exit: if JOB = 'V', then this is overwritten by the unitary matrix Z which contains the eigenvectors of A. The ith column of Z contains the eigenvector which corresponds to the eigenvalue W(i).

If JOB = 'N', then Z is not referenced.

9: LDZ — INTEGER *Input*

On entry: the leading dimension of the array Z as declared in the (sub)program from which F08HQF (CHBEVD/ZHBEVD) is called.

Constraints:

LDZ $\geq$ max(1,N) if JOB = 'V';
LDZ $\geq$ 1 if JOB = 'N'.

10: WORK(*) — ***complex*** array *Workspace*

Note: the dimension of the array WORK must be at least LWORK.

On exit: if LWORK > 0, then WORK(1) contains the required minimal size of LWORK.

11: LWORK — INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08HQF (CHBEVD/ZHBEVD) is called.

Constraints:

if N $\leq$ 1, then LWORK $\geq$ 1;
if JOB = 'N' and N > 1, then LWORK $\geq$ N;
if JOB = 'V' and N > 1, then LWORK $\geq 2 \times \mathrm{N}^2$.

12: RWORK(*) — ***real*** array *Workspace*

Note: the dimension of the array RWORK must be at least LRWORK.

On exit: if LRWORK > 0, then RWORK(1) contains the required minimal size of LRWORK.

13: LRWORK — INTEGER *Input*

On entry: the dimension of the array RWORK as declared in the (sub)program from which F08HQF (CHBEVD/ZHBEVD) is called.

Constraints:

if N $\leq$ 1, then LRWORK $\geq$ 1;
if JOB = 'N' and N > 1, then LWORK $\geq$ N;
if JOB = 'V' and N > 1, then LRWORK $\geq 3 \times \mathrm{N}^2 + (4{+}2k) \times \mathrm{N} + 1$ where k is the smallest integer which satisfies $2^k \geq \mathrm{N}$.

14: IWORK(*) — INTEGER array *Workspace*

Note: the dimension of the array IWORK must be at least LIWORK.

On exit: if LIWORK > 0, then IWORK(1) contains the required minimal size of LIWORK.

15: LIWORK — INTEGER *Input*

On entry: the dimension of the array IWORK as declared in the (sub)program from which F08HQF (CHBEVD/ZHBEVD) is called.

Constraints:

if JOB = 'N' or N $\leq$ 1, then LIWORK $\geq$ 1;
if JOB = 'V' and N > 1, then LIWORK $\geq 5 \times \mathrm{N} + 2$.

16: INFO — INTEGER *Output*

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

6 Error Indicators and Warnings

INFO < 0

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

INFO > 0

If INFO = i, then the algorithm failed to converge; i indicates the number of elements of an intermediate tridiagonal form which did not converge to zero.

7 Accuracy

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

$$\|E\|_2 = O(\epsilon)\|A\|_2,$$

and ϵ is the ***machine precision***.

8 Further Comments

The real analogue of this routine is F08HCF (SSBEVD/DSBEVD).

9 Example

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

$$A = \begin{pmatrix} 1.0+0.0i & 2.0+1.0i & 3.0+1.0i & 0.0+0.0i & 0.0+0.0i \\ 2.0-1.0i & 2.0+0.0i & 3.0+2.0i & 4.0+2.0i & 0.0+0.0i \\ 3.0-1.0i & 3.0-2.0i & 3.0+0.0i & 4.0+3.0i & 5.0+3.0i \\ 0.0+0.0i & 4.0-2.0i & 4.0-3.0i & 4.0+0.0i & 5.0+4.0i \\ 0.0+0.0i & 0.0+0.0i & 5.0-3.0i & 5.0+0.0i & 5.0+0.0i \end{pmatrix}.$$

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.

```
*     F08HQF Example Program Text.
*     Mark 19 Release. NAG Copyright 1999.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDZ, LDAB
      PARAMETER        (NMAX=8,LDZ=NMAX,LDAB=NMAX)
      INTEGER          LWORK, LIWORK, LRWORK
      PARAMETER        (LWORK=NMAX*NMAX+2*NMAX,LIWORK=2+5*NMAX,
     +                 LRWORK=4*NMAX*NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, KD, N
      CHARACTER        JOB, UPLO
*     .. Local Arrays ..
      complex          AB(LDAB,NMAX), WORK(LWORK), Z(LDZ,NMAX)
      real             RWORK(LRWORK), W(NMAX)
      INTEGER          IWORK(LIWORK)
*     .. External Subroutines ..
      EXTERNAL         X04DAF, chbevd
*     .. Intrinsic Functions ..
      INTRINSIC        MAX, MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08HQF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, KD
      IF (N.LE.NMAX) 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
*
         READ (NIN,*) JOB
*
*        Calculate all the eigenvalues and eigenvectors of A
*
         CALL chbevd(JOB,UPLO,N,KD,AB,LDAB,W,Z,LDZ,WORK,LWORK,RWORK,
     +               LRWORK,IWORK,LIWORK,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.GT.0) THEN
            WRITE (NOUT,*) 'Failure to converge.'
         ELSE
*
*           Print eigenvalues and eigenvectors
*
            WRITE (NOUT,*) 'Eigenvalues'
            DO 60 I = 1, N
               WRITE (NOUT,99999) I, W(I)
   60       CONTINUE
            WRITE (NOUT,*)
            IFAIL = 0
*
            CALL X04DAF('General',' ',N,N,Z,LDZ,'Eigenvectors',IFAIL)
*
         END IF
      END IF
      STOP
*
99999 FORMAT (3X,I5,5X,2F8.4)
      END
```

9.2 Program Data

```
F08HQF Example Program Data
  5  2                                                   :Values of N and KD
  'L'                                                    :Value of UPLO
  (1.0, 0.0)
  (2.0, 1.0) (2.0, 0.0)
  (3.0, 1.0) (3.0, 2.0) (3.0, 0.0)
             (4.0, 2.0) (4.0, 3.0) (4.0, 0.0)
                        (5.0, 3.0) (5.0, 4.0) (5.0, 0.0) :End of matrix A
  'V'                                                    :Value of JOB
```

9.3 Program Results

```
 F08HQF Example Program Results

 Eigenvalues
       1      -6.4185
       2      -1.4094
       3       1.4421
       4       4.4856
       5      16.9002
 Eigenvectors
          1       2       3       4       5
 1  -0.2591  0.6367  0.4516  0.5503  0.1439
     0.0000  0.0000  0.0000  0.0000  0.0000

 2   0.0245 -0.2578 -0.3029  0.4785  0.3060
     0.4344  0.2413 -0.4402  0.2759  0.0411

 3   0.5159 -0.3039  0.3160  0.2128  0.4681
    -0.1095 -0.3481  0.2978  0.0465  0.2306

 4   0.0004  0.3450 -0.4088 -0.1707  0.4098
    -0.5093 -0.0832 -0.3213  0.0200  0.3832

 5  -0.4333 -0.2469  0.0204  0.0175  0.1819
     0.1353  0.2634  0.2262 -0.5611  0.5136
```

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.

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,
1                  INFO)
 ENTRY      chbtrd(VECT, UPLO, N, KD, AB, LDAB, D, E, Q, LDQ, WORK,
1                  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: $T = Q^H AQ$. 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 (1984) Banded eigenvalue solvers on vector machines *ACM Trans. Math. Software* **10** 73–86

[2] Parlett B N (1980) *The Symmetric Eigenvalue Problem* Prentice–Hall

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 (and the array Q must contain a matrix on entry);
if VECT = 'U', then Q is updated (and the array Q must contain a matrix on entry);
if VECT = 'N', then Q is not required.

Constraint: VECT = 'V', 'U' 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 $\geq$ 0.

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

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

On entry: the 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 $\max(1,j-k) \leq i \leq j$; if UPLO = 'L', the elements of the lower triangle of A within the band must be stored with element a_{ij} in AB($1+i-j,j$) for $j \leq i \leq \min(n,j+k)$.

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 $\geq$ 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 *Input/Output*

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

On entry: if VECT = 'U', Q must contain the matrix formed in a previous stage of the reduction (for example, the reduction of a banded Hermitian-definite generalized eigenproblem); otherwise Q need not be set.

On exit: if VECT = 'V' or 'U', the n by n matrix Q.

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 $\geq$ max(1,N) if VECT = 'V' or 'U';
LDQ $\geq$ 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 ith 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

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

$c(n)$ is a modestly increasing function of n, and ϵ 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

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

where ϵ 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)
```

F08JCF (SSTEVD/DSTEVD) – 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

F08JCF (SSTEVD/DSTEVD) computes all the eigenvalues, and optionally all the eigenvectors, of a real symmetric tridiagonal matrix. If the eigenvectors are requested, then it uses a divide and conquer algorithm to compute eigenvalues and eigenvectors. However, if only eigenvalues are required, then it uses the Pal–Walker–Kahan variant of the QL or QR algorithm.

2 Specification

```
 SUBROUTINE F08JCF(JOB, N, D, E, Z, LDZ, WORK, LWORK, IWORK,
1                  LIWORK, INFO)
 ENTRY      sstevd(JOB, N, D, E, Z, LDZ, WORK, LWORK, IWORK,
1                  LIWORK, INFO)
 INTEGER           N, LDZ, LWORK, IWORK(*), LIWORK, INFO
 real              D(*), E(*), Z(LDZ,*), WORK(*)
 CHARACTER*1       JOB
```

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 Λ is a diagonal matrix whose diagonal elements are the eigenvalues λ_i, and Z is the orthogonal matrix whose columns are the eigenvectors z_i. Thus

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

4 References

[1] Golub G H and Van Loan C F (1996) *Matrix Computations* Johns Hopkins University Press (3rd Edition), Baltimore

5 Parameters

1: JOB — CHARACTER*1 *Input*

On entry: indicates whether eigenvectors are computed as follows:

if JOB = 'N', then only eigenvalues are computed;
if JOB = 'V', then eigenvalues and eigenvectors are computed.

Constraint: JOB = 'N' or 'V'.

2: N — INTEGER *Input*

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

Constraint: N $\geq$ 0.

3: D(∗) — ***real*** array *Input/Output*

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

On entry: the n diagonal elements of the tridiagonal matrix T.

On exit: the eigenvalues of the matrix T in ascending order.

4: E(∗) — ***real*** array *Input/Output*

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

On entry: the $n-1$ off-diagonal elements of the tridiagonal matrix T. The nth element of this array is used as workspace.

On exit: the array is overwritten with intermediate results.

5: Z(LDZ, ∗) — ***real*** array *Output*

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

On exit: if JOB = 'V', then this is overwritten by the orthogonal matrix Z which contains the eigenvectors of T.

If JOB = 'N', then Z is not referenced.

6: LDZ — INTEGER *Input*

On entry: the leading dimension of the array Z as declared in the (sub)program from which F08JCF (SSTEVD/DSTEVD) is called.

Constraints:

LDZ $\geq$ max(1,N) if JOB = 'V';
LDZ $\geq$ 1 if JOB = 'N'.

7: WORK(∗) — ***real*** array *Workspace*

Note: the dimension of the array WORK must be at least LWORK.

On exit: if LWORK > 0, then WORK(1) contains the required minimal size of LWORK.

8: LWORK — INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08JCF (SSTEVD/DSTEVD) is called.

Constraints:

if JOB = 'N' or N $\leq$ 1, then LWORK $\geq$ 1;
if JOB = 'V' and N > 1, then LWORK $\geq 2 \times \mathrm{N}^2 + (3+2k) \times \mathrm{N} + 1$ where k is the smallest integer which satisfies $2^k \geq \mathrm{N}$.

9: IWORK(∗) — INTEGER array *Workspace*

Note: the dimension of the array IWORK must be at least LIWORK.

On exit: if LIWORK > 0, then IWORK(1) contains the required minimal size of LIWORK.

10: LIWORK — INTEGER *Input*

On entry: the dimension of the array IWORK as declared in the (sub)program from which F08JCF (SSTEVD/DSTEVD) is called.

Constraints:

if JOB = 'N' or N $\leq$ 1, then LIWORK $\geq$ 1;
if JOB = 'V' and N > 1, then LIWORK $\geq 5 \times \mathrm{N} + 2$.

11: INFO — INTEGER *Output*

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

6 Error Indicators and Warnings

INFO < 0

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

INFO > 0

If INFO $= i$, then the algorithm failed to converge; i indicates the number of elements of an intermediate tridiagonal form which did not converge to zero.

7 Accuracy

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

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

and ϵ is the ***machine precision***.

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

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

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:

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

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

8 Further Comments

There is no complex analogue of this routine.

9 Example

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

$$T = \begin{pmatrix} 1.0 & 1.0 & 0.0 & 0.0 \\ 1.0 & 4.0 & 1.0 & 0.0 \\ 0.0 & 1.0 & 9.0 & 1.0 \\ 0.0 & 0.0 & 1.0 & 16.0 \end{pmatrix}.$$

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.

```
*     F08JCF Example Program Text.
*     Mark 19 Release. NAG Copyright 1999.
*     .. Parameters ..
```

```
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LDZ
      PARAMETER        (NMAX=8,LDZ=NMAX)
      INTEGER          LWORK, LIWORK
      PARAMETER        (LWORK=4*NMAX*NMAX,LIWORK=5*NMAX+2)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, N
      CHARACTER        JOB
*     .. Local Arrays ..
      real             D(NMAX), E(NMAX), WORK(LWORK), Z(LDZ,NMAX)
      INTEGER          IWORK(LIWORK)
*     .. External Subroutines ..
      EXTERNAL         sstevd, X04CAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08JCF 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)
*
         READ (NIN,*) JOB
*
*        Calculate all the eigenvalues and eigenvectors of T
*
         CALL sstevd(JOB,N,D,E,Z,LDZ,WORK,LWORK,IWORK,LIWORK,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

```
F08JCF Example Program Data
 4                             :Value of N
 1.0  4.0  9.0  16.0
 1.0  2.0  3.0                 :End of T
 'V'                           :Value of JOB
```

9.3 Program Results

```
 F08JCF Example Program Results

 Eigenvalues
     0.6476  3.5470  8.6578 17.1477

 Eigenvectors
          1       2       3       4
 1   0.9396  0.3388  0.0494  0.0034
 2  -0.3311  0.8628  0.3781  0.0545
 3   0.0853 -0.3648  0.8558  0.3568
 4  -0.0167  0.0879 -0.3497  0.9326
```

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 Λ is a diagonal matrix whose diagonal elements are the eigenvalues λ_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:

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

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 ±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 ith column corresponds to the ith 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 ith 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 ε is the ***machine precision.***

If λ_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:

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

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 ith 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 ε is the ***machine precision***.

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

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

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 Λ is a diagonal matrix whose diagonal elements are the eigenvalues λ_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:

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

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 ±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 $\geq$ 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 ith column corresponds to the ith 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 $\geq$ 1 if COMPZ = 'N',
LDZ $\geq$ 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 ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, the leading minor of order i is not positive-definite and the 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 λ_i is an exact eigenvalue of T and $\tilde{\lambda}_i$ is the corresponding computed value, then

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

where $c(n)$ is a modestly increasing function of n, ϵ 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:

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

where $relgap_i$ is the relative gap between λ_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:

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

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 ith 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 ith 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:

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

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: $0 \le \mathrm{M} \le \mathrm{N}$.

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 ith column corresponds to the ith 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 ith 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 ε 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 Λ is a diagonal matrix whose diagonal elements are the eigenvalues λ_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 ith column corresponds to the ith 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 ith 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 ε is the ***machine precision***.

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

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

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:

$$\theta(\tilde{z}_i, z_i) \le \frac{c(n)\varepsilon\|T\|_2}{\min\limits_{i \ne j}|\lambda_i - \lambda_j|}.$$

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 Λ is a diagonal matrix whose diagonal elements are the eigenvalues λ_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:

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

In this case, the matrix Q must be formed explicitly and passed to 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 $\geq$ 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 ith column corresponds to the ith 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 $\geq$ 1 if COMPZ = 'N',
LDZ $\geq$ 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 ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO > 0

If INFO = i, the leading minor of order i is not positive-definite and the 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 λ_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, ε 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 λ_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:

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

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) $\leq$ 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) $\leq$ 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 ith column corresponds to the ith 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 $\geq$ 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 ith 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 ε 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:

$$A = Q\begin{pmatrix} B_1 \\ 0 \end{pmatrix}P^T = Q_1B_1P^T,$$

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 ith 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 ε 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 $\frac{4}{3}n^2(3m-n)$ if $m \geq n$ or $\frac{4}{3}m^2(3n-m)$ 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 ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

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

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

where ε is the ***machine precision***. 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 ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed result differs from the exact result by a matrix E such that

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

where ε is the ***machine precision.***

8. Further Comments

The total number of floating-point operations is approximately

$2n_Ck(2m_C-k)$ if SIDE = 'L' and $m_C \geq k$;

$2m_Ck(2n_C-k)$ if SIDE = 'R' and $n_C \geq k$;

$2m_C^2n_C$ if SIDE = 'L' and $m_C < k$;

$2m_Cn_C^2$ 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 $A = Q_aR$ and then reduces the factor R to bidiagonal form B: $R = Q_bBP^T$. Finally it forms Q_a and calls F08KGF (SORMBR/DORMBR) to form $Q = Q_aQ_b$.

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:

$$A = Q\begin{pmatrix} B_1 \\ 0 \end{pmatrix}P^H = Q_1B_1P^H,$$

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 ith 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 ε 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 ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

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

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

where ε is the ***machine precision***. 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 $m > n$, $\dfrac{16m^3}{3}$ if $m \leq n$;
2. To form the n leading columns of Q when $m > n$: $\frac{8}{3}n^2(3m-n)$;
3. To form the whole of P^H: $\dfrac{16n^3}{3}$ 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 $m < n$: $\frac{8}{3}m^2(3n-m)$.

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 ≥ 0.

5: N – INTEGER. *Input*

On entry: n_C, the number of columns of the matrix C.

Constraint: N ≥ 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 ≥ 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 ≥ max(1,r) if VECT = 'Q',
LDA ≥ 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 ≥ 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 ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed result differs from the exact result by a matrix E such that

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

where ε is the ***machine precision***.

8. Further Comments

The total number of real floating-point operations is approximately

$8n_Ck(2m_C-k)$ if SIDE = 'L' and $m_C \geq k$;

$8m_Ck(2n_C-k)$ if SIDE = 'R' and $n_C \geq k$;

$8m_C^2n_C$ if SIDE = 'L' and $m_C < k$;

$8m_Cn_C^2$ 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 $A = Q_aR$ and then reduces the factor R to bidiagonal form B: $R = Q_bBP^H$. Finally it forms Q_a and calls F08KUF (CUNMBR/ZUNMBR) to form $Q = Q_aQ_b$.

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)
```

F08LEF (SGBBRD/DGBBRD) – 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

F08LEF (SGBBRD/DGBBRD) reduces a real m by n band matrix to upper bidiagonal form.

2 Specification

```
      SUBROUTINE F08LEF(VECT, M, N, NCC, KL, KU, AB, LDAB, D, E, Q, LDQ,
     1                  PT, LDPT, C, LDC, WORK, INFO)
      ENTRY      sgbbrd(VECT, M, N, NCC, KL, KU, AB, LDAB, D, E, Q, LDQ,
     1                  PT, LDPT, C, LDC, WORK, INFO)
      INTEGER           M, N, NCC, KL, KU, LDAB, LDQ, LDPT, LDC, INFO
      real              AB(LDAB,*), D(*), E(*), Q(LDQ,*), PT(LDPT,*),
     1                  C(LDC,*), WORK(*)
      CHARACTER*1       VECT
```

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

3 Description

This routine reduces a real m by n band matrix to upper bidiagonal form B by an orthogonal transformation: $A = QBP^T$. The orthogonal matrices Q and P^T, of order m and n respectively, are determined as a product of Givens rotation matrices, and may be formed explicitly by the routine if required. A matrix C may also be updated to give $\tilde{C} = Q^TC$.

The routine uses a vectorisable form of the reduction.

4 References

None.

5 Parameters

1: VECT — CHARACTER*1 *Input*

On entry: indicates whether the matrices Q and/or P^T are generated:

if VECT = 'N', then neither Q nor P^T is generated;
if VECT = 'Q', then Q is generated;
if VECT = 'P', then P^T is generated;
if VECT = 'B', then both Q and P^T are generated.

Constraint: VECT = 'N', 'Q', 'P' or 'B'.

2: M — INTEGER *Input*

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

Constraint: M $\geq$ 0.

3: N — INTEGER *Input*

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

Constraint: N $\geq$ 0.

4: NCC — INTEGER *Input*

On entry: n_C, the number of columns of the matrix C.

Constraint: NCC ≥ 0.

5: KL — INTEGER *Input*

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

Constraint: KL ≥ 0.

6: KU — INTEGER *Input*

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

Constraint: KU ≥ 0.

7: AB(LDAB,$*$) — ***real*** array *Input/Output*

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

On entry: the m by n band matrix A, stored in rows 1 to $k_l + k_u + 1$. More precisely, element a_{ij} must be stored in AB$(k_u + 1 + i - j, j)$ for $\max(1,j - k_u) \leq i \leq \min(m, j + k_l)$.

On exit: A is overwritten by values generated during the reduction.

8: LDAB — INTEGER *Input*

On entry: the first dimension of the array AB as declared in the (sub)program from which F08LEF (SGBBRD/DGBBRD) is called.

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

9: 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.

10: E($*$) — ***real*** array *Output*

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

On exit: the super-diagonal elements of the bidiagonal matrix B.

11: Q(LDQ,$*$) — ***real*** array *Output*

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

On exit: the m by m orthogonal matrix Q, if VECT = 'Q' or 'B'.

Q is not referenced if VECT = 'N' or 'P'.

12: LDQ — INTEGER *Input*

On entry: the first dimension of the array Q as declared in the (sub)program from which F08LEF (SGBBRD/DGBBRD) is called.

Constraints:

LDQ $\geq$ max(1,M) if VECT = 'Q' or 'B';
LDQ $\geq$ 1 otherwise.

13: PT(LDPT,$*$) — ***real*** array *Output*

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

On exit: the n by n orthogonal matrix P^T, if VECT = 'P' or 'B'.

PT is not referenced if VECT = 'N' or 'Q'.

14: LDPT — INTEGER *Input*

On entry: the first dimension of the array PT as declared in the (sub)program from which F08LEF (SGBBRD/DGBBRD) is called.

Constraints:

LDPT $\geq$ max(1,N) if VECT = 'P' or 'B';
LDPT $\geq$ 1 otherwise.

15: C(LDC,*) — ***real*** array *Input/Output*

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

On entry: an m by n_C matrix C.

On exit: C is overwritten by Q^TC.

C is not referenced if NCC = 0.

16: LDC — INTEGER *Input*

On entry: the first dimension of the array C as declared in the (sub)program from which F08LEF (SGBBRD/DGBBRD) is called.

Constraints:

LDC $\geq$ max(1,M) if NCC > 0;
LDC $\geq$ 1 if NCC = 0.

17: WORK(*) — ***real*** array *Workspace*

Note: the dimension of the array WORK must be at least 2*max(M,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 ith 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

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

$c(n)$ is a modestly increasing function of n, and ϵ 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.

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

$$\| F \|_2 = O(\epsilon).$$

A similar statement holds for the computed matrix P^T.

8 Further Comments

The total number of real floating-point operations is approximately the sum of:

$6n^2k$, if VECT = 'N' and NCC = 0, and:

$3n^2n_C(k-1)/k$, if C is updated, and:

$3n^3(k-1)/k$, if either Q or P^T is generated (double this if both),

where $k = k_l + k_u$, assuming $n \gg k$. For this section we assume that $m = n$.

The complex analogue of this routine is F08LSF (CGBBRD/ZGBBRD).

9 Example

To reduce the matrix A to upper bidiagonal form, where

$$A = \begin{pmatrix} -0.57 & -1.28 & 0.00 & 0.00 \\ -1.93 & 1.08 & -0.31 & 0.00 \\ 2.30 & 0.24 & 0.40 & -0.35 \\ 0.00 & 0.64 & -0.66 & 0.08 \\ 0.00 & 0.00 & 0.15 & -2.13 \\ -0.00 & 0.00 & 0.00 & 0.50 \end{pmatrix}.$$

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.

```
*     F08LEF Example Program Text.
*     Mark 19 Release. NAG Copyright 1999.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          MMAX, NMAX, NCCMAX, KLMAX, KUMAX, LDAB, LDQ,
     +                 LDPT, LDC
      PARAMETER        (MMAX=8,NMAX=8,NCCMAX=8,KLMAX=8,KUMAX=8,
     +                 LDAB=KLMAX+KUMAX+1,LDQ=MMAX,LDPT=NMAX,LDC=MMAX)
      CHARACTER        VECT
      PARAMETER        (VECT='N')
*     .. Local Scalars ..
      INTEGER          I, INFO, J, KL, KU, M, N, NCC
*     .. Local Arrays ..
      real             AB(LDAB,NMAX), C(MMAX,NCCMAX), D(NMAX),
     +                 E(NMAX-1), PT(LDPT,NMAX), Q(LDQ,MMAX),
     +                 WORK(2*MMAX+2*NMAX)
*     .. External Subroutines ..
      EXTERNAL         sgbbrd
*     .. Intrinsic Functions ..
      INTRINSIC        MAX, MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08LEF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) M, N, KL, KU, NCC
      IF (M.LE.MMAX .AND. N.LE.NMAX .AND. KL.LE.KLMAX .AND. KU.LE.
     +    KUMAX .AND. NCC.LE.NCCMAX) THEN
*
*        Read A from data file
*
```

```
         READ (NIN,*) ((AB(KU+1+I-J,J),J=MAX(I-KL,1),MIN(I+KU,N)),I=1,M)
*
*        Reduce A to upper bidiagonal form
*
         CALL sgbbrd(VECT,M,N,NCC,KL,KU,AB,LDAB,D,E,Q,LDQ,PT,LDPT,C,LDC,
     +               WORK,INFO)
*
*        Print bidiagonal form
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) 'Diagonal'
         WRITE (NOUT,99999) (D(I),I=1,MIN(M,N))
         WRITE (NOUT,*) 'Super-diagonal'
         WRITE (NOUT,99999) (E(I),I=1,MIN(M,N)-1)
      END IF
      STOP
*
99999 FORMAT (1X,8F9.4)
      END
```

9.2 Program Data

```
F08LEF Example Program Data
  6  4  2  1  0                 :Values of M, N, KL, KU and NCC
 -0.57  -1.28
 -1.93   1.08  -0.31
  2.30   0.24   0.40  -0.35
         0.64  -0.66   0.08
                0.15  -2.13
                       0.50     :End of matrix A
```

9.3 Program Results

```
F08LEF Example Program Results

Diagonal
   3.0561   1.5259   0.9690  -1.5685
Super-diagonal
   0.6206   1.2353  -1.1240
```

F08LSF (CGBBRD/ZGBBRD) – 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

F08LSF (CGBBRD/ZGBBRD) reduces a complex m by n band matrix to real upper bidiagonal form.

2 Specification

```
 SUBROUTINE F08LSF(VECT, M, N, NCC, KL, KU, AB, LDAB, D, E, Q, LDQ,
1                  PT, LDPT, C,LDC, WORK, RWORK, INFO)
 ENTRY       cgbbrd(VECT, M, N, NCC, KL, KU, AB, LDAB, D, E, Q, LDQ,
1                  PT, LDPT, C,LDC, WORK, RWORK, INFO)
 INTEGER           M, N, NCC, KL, KU, LDAB, LDQ, LDPT, LDC, INFO
 real              D(*), E(*), RWORK(*)
 complex           AB(LDAB,*), Q(LDQ,*), PT(LDPT,*), C(LDC,*),
1                  WORK(*)
 CHARACTER*1       VECT
```

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

3 Description

This routine reduces a complex m by n band matrix to real upper bidiagonal form B by a unitary transformation: $A = QBP^H$. The unitary matrices Q and P^H, of order m and n respectively, are determined as a product of Givens rotation matrices, and may be formed explicitly by the routine if required. A matrix C may also be updated to give $\tilde{C} = Q^H C$.

The routine uses a vectorisable form of the reduction.

4 References

None.

5 Parameters

1: VECT — CHARACTER*1 *Input*

On entry: indicates whether the matrices Q and/or P^H are generated:

if VECT = 'N', then neither Q nor P^H is generated;
if VECT = 'Q', then Q is generated;
if VECT = 'P', then P^H is generated;
if VECT = 'B', then both Q and P^H are generated.

Constraint: VECT = 'N', 'Q', 'P' or 'B'.

2: M — INTEGER *Input*

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

Constraint: M $\geq$ 0.

3: N — INTEGER *Input*

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

Constraint: N $\geq$ 0.

4: NCC — INTEGER *Input*

On entry: n_C, the number of columns of the matrix C.

Constraint: NCC $\geq$ 0.

5: KL — INTEGER *Input*

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

Constraint: KL $\geq$ 0.

6: KU — INTEGER *Input*

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

Constraint: KU $\geq$ 0.

7: AB(LDAB,*) — ***complex*** array *Input/Output*

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

On entry: the m by n band matrix A, stored in rows 1 to $k_l + k_u + 1$. More precisely, element a_{ij} must be stored in AB$(k_u + 1 + i - j, j)$ for $\max(1,j - k_u) \leq i \leq \min(m, j + k_l)$.

On exit: A is overwritten by values generated during the reduction.

8: LDAB — INTEGER *Input*

On entry: the first dimension of the array AB as declared in the (sub)program from which F08LSF (CGBBRD/ZGBBRD) is called.

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

9: 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.

10: E(*) — ***real*** array *Output*

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

On exit: the super-diagonal elements of the bidiagonal matrix B.

11: Q(LDQ,*) — ***complex*** array *Output*

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

On exit: the m by m unitary matrix Q, if VECT = 'Q' or 'B'.

Q is not referenced if VECT = 'N' or 'P'.

12: LDQ — INTEGER *Input*

On entry: the first dimension of the array Q as declared in the (sub)program from which F08LSF (CGBBRD/ZGBBRD) is called.

Constraints:

LDQ $\geq$ max(1,M) if VECT = 'Q' or 'B';
LDQ $\geq$ 1 otherwise.

13: PT(LDPT,*) — ***complex*** array *Output*

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

On exit: the n by n unitary matrix P^H, if VECT = 'P' or 'B'.

PT is not referenced if VECT = 'N' or 'Q'.

14: LDPT — INTEGER *Input*

On entry: the first dimension of the array PT as declared in the (sub)program from which F08LSF (CGBBRD/ZGBBRD) is called.

Constraints:

LDPT $\geq$ max(1,N) if VECT = 'P' or 'B';
LDPT $\geq$ 1 otherwise.

15: C(LDC,*) — ***complex*** array *Input/Output*

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

On entry: an m by n_C matrix C.

On exit: C is overwritten by $Q^H C$.

C is not referenced if NCC = 0.

16: LDC — INTEGER *Input*

On entry: the first dimension of the array C as declared in the (sub)program from which F08LSF (CGBBRD/ZGBBRD) is called.

Constraints:

LDC $\geq$ max(1,M) if NCC > 0;
LDC $\geq$ 1 if NCC = 0.

17: WORK(*) — ***complex*** array *Workspace*

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

18: RWORK(*) — ***real*** array *Workspace*

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

19: INFO — INTEGER *Output*

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

6 Error Indicators and Warnings

INFO < 0

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

7 Accuracy

The computed bidiagonal form B satisfies $QBP^H = A + E$, where

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

$c(n)$ is a modestly increasing function of n, and ϵ 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.

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

$$\|F\|_2 = O(\epsilon).$$

A similar statement holds for the computed matrix P^H.

8 Further Comments

The total number of real floating-point operations is approximately the sum of:

$20n^2k$, if VECT = 'N' and NCC = 0, and:

$10n^2n_C(k-1)/k$, if C is updated, and:

$10n^3(k-1)/k$ if either Q or P^H is generated (double this if both),

where $k = k_l + k_u$, assuming $n \gg k$. For this section we assume that $m = n$.

The real analogue of this routine is F08LEF (SGBBRD/DGBBRD).

9 Example

To reduce the matrix A to upper bidiagonal form, where

$$A = \begin{pmatrix} 0.96-0.81i & -0.03+0.96i & 0.00+0.00i & 0.00+0.00i \\ -0.98+1.98i & -1.20+0.19i & -0.66+0.42i & 0.00+0.00i \\ 0.62-0.46i & 1.01+0.02i & 0.63-0.17i & -1.11+0.60i \\ 0.00+0.00i & 0.19-0.54i & -0.98-0.36i & 0.22-0.20i \\ 0.00+0.00i & 0.00+0.00i & -0.17-0.46i & 1.47+1.59i \\ 0.00+0.00i & 0.00+0.00i & 0.00+0.00i & 0.26+0.26i \end{pmatrix}.$$

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.

```
*     F08LSF Example Program Text.
*     Mark 19 Release. NAG Copyright 1999.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          MMAX, NMAX, NCCMAX, KLMAX, KUMAX, LDAB, LDQ,
     +                 LDPT, LDC
      PARAMETER        (MMAX=8,NMAX=8,NCCMAX=8,KLMAX=8,KUMAX=8,
     +                 LDAB=KLMAX+KUMAX+1,LDQ=MMAX,LDPT=NMAX,LDC=MMAX)
      CHARACTER        VECT
      PARAMETER        (VECT='N')
*     .. Local Scalars ..
      INTEGER          I, INFO, J, KL, KU, M, N, NCC
*     .. Local Arrays ..
      complex          AB(LDAB,NMAX), C(MMAX,NCCMAX), PT(LDPT,NMAX),
     +                 Q(LDQ,MMAX), WORK(MMAX+NMAX)
      real             D(NMAX), E(NMAX-1), RWORK(MMAX+NMAX)
*     .. External Subroutines ..
      EXTERNAL         cgbbrd
*     .. Intrinsic Functions ..
      INTRINSIC        MAX, MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08LSF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) M, N, KL, KU, NCC
      IF (M.LE.MMAX .AND. N.LE.NMAX .AND. KL.LE.KLMAX .AND. KU.LE.
     +    KUMAX .AND. NCC.LE.NCCMAX) THEN
*
*        Read A from data file
*
```

```
            READ (NIN,*) ((AB(KU+1+I-J,J),J=MAX(I-KL,1),MIN(I+KU,N)),I=1,M)
*
*           Reduce A to upper bidiagonal form
*
            CALL cgbbrd(VECT,M,N,NCC,KL,KU,AB,LDAB,D,E,Q,LDQ,PT,LDPT,C,LDC,
     +                  WORK,RWORK,INFO)
*
*           Print bidiagonal form
*
            WRITE (NOUT,*)
            WRITE (NOUT,*) 'Diagonal'
            WRITE (NOUT,99999) (D(I),I=1,MIN(M,N))
            WRITE (NOUT,*) 'Super-diagonal'
            WRITE (NOUT,99999) (E(I),I=1,MIN(M,N)-1)
         END IF
         STOP
*
99999 FORMAT (1X,8F9.4)
      END
```

9.2 Program Data

```
F08LSF Example Program Data
  6  4  2  1  0                                   :Values of M, N, KL, KU and NCC
 ( 0.96,-0.81) (-0.03, 0.96)
 (-0.98, 1.98) (-1.20, 0.19) (-0.66, 0.42)
 ( 0.62,-0.46) ( 1.01, 0.02) ( 0.63,-0.17) (-1.11, 0.60)
               ( 0.19,-0.54) (-0.98,-0.36) ( 0.22,-0.20)
                             (-0.17,-0.46) ( 1.47, 1.59)
                                           ( 0.26, 0.26)   :End of matrix A
```

9.3 Program Results

```
F08LSF Example Program Results

Diagonal
   2.6560   1.7501   2.0607   0.8658
Super-diagonal
   1.7033   1.2800   0.1467
```

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 Σ is a diagonal matrix with real diagonal elements σ_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 ±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 ith 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 σ_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 ϵ 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 σ_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 Σ is a diagonal matrix with real diagonal elements σ_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 ith 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 σ_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 ϵ 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 σ_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: $N \geq 0$.

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: $1 \leq \text{ILO} \leq \text{IHI} \leq \text{N}$ if $\text{N} > 0$,
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 ith 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 ε 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 ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

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

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

where ε is the ***machine precision***.

8. Further Comments

The total number of floating-point operations is approximately $\frac{4}{3}q^3$, 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 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 ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed result differs from the exact result by a matrix E such that

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

where ε is the ***machine precision***.

8. Further Comments

The total number of floating-point operations is approximately $2nq^2$ 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 λ 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 $\ge$ 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 $\ge$ 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

$$\mathrm{SCALE}(j) = \begin{cases} p_j, & \text{for } j = 1,2,\ldots,i_{lo}-1 \\ d_j, & \text{for } j = i_{lo},i_{lo}+1,\ldots,i_{hi} \text{ and} \\ p_j, & \text{for } j = i_{hi}+1,i_{hi}+2,\ldots,n. \end{cases}$$

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 ith 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: $1 \leq$ ILO $\leq$ IHI $\leq$ N if 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: M ≥ 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: LDV $\geq$ max(1,N).

10: INFO – INTEGER. *Output*

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

6. Error Indicators and Warnings

INFO < 0

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

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: $1 \le$ ILO $\le$ IHI $\le$ 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 ith 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 ε 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 $\frac{8}{3}q^2(2q+3n)$, 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 $\geq$ 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: $1 \leq$ ILO $\leq$ IHI $\leq$ 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 ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

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

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

where ε is the ***machine precision***.

8. Further Comments

The total number of real floating-point operations is approximately $\frac{16q^3}{3}$, 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 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 $\geq$ 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 $\geq$ 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 $\leq$ ILO $\leq$ IHI $\leq$ M if SIDE = 'L' and M > 0;
ILO = 1 and IHI = 0 if SIDE = 'L' and M = 0;
1 $\leq$ ILO $\leq$ IHI $\leq$ 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 $\geq$ max(1,M) if SIDE = 'L',
LDA $\geq$ 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 $\geq$ 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 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 ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7. Accuracy

The computed result differs from the exact result by a matrix E such that

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

where ε is the ***machine precision***.

8. Further Comments

The total number of real floating-point operations is approximately $8nq^2$ 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 λ 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 $\geq$ 0.

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

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

On entry: the n by n 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 $\geq$ 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 $1 \leq j < i_{lo}$ or $i_{hi} < i \leq n$.

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 ith 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: $1 \leq$ ILO $\leq$ IHI $\leq$ N if 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: M ≥ 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: LDV $\geq$ max(1,N).

10: INFO – INTEGER. *Output*

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

6. Error Indicators and Warnings

INFO < 0

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

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 ±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 ith 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 ε is the ***machine precision***.

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

$$|\tilde{\lambda}_i - \lambda_i| \le \frac{c(n)\epsilon\|H\|_2}{s_i},$$

where $c(n)$ is a modestly increasing function of n, and s_i is the reciprocal condition number of λ_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$(i+1)$ = α; 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 λ, 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, λ, x and y may be complex. If x is an eigenvector corresponding to a complex eigenvalue λ, 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 jth eigenvalue can be assumed to be an eigenvalue of the block containing the jth 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); $0 \le m \le n$.

Constraint: MM $\ge m$.

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 ith column of VL (corresponding to the jth eigenvalue) failed to converge. If the ith 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 ith column of VR (corresponding to the jth eigenvalue) failed to converge. If the ith 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 ith 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 $\geq$ 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 $\geq$ 1 and min(ILO,N) $\leq$ IHI $\leq$ 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 $\geq$ 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 ith 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 ε is the ***machine precision***.

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

$$|\tilde{\lambda}_i-\lambda_i| \le \frac{c(n)\varepsilon\|H\|_2}{s_i},$$

where $c(n)$ is a modestly increasing function of n, and s_i is the reciprocal condition number of λ_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 λ, 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 jth eigenvalue can be assumed to be an eigenvalue of the block containing the jth 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); $0 \le m \le n$.

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 ith column of VL (corresponding to the jth 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 ith column of VR (corresponding to the jth 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 ith 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: $\tilde{T} = Z^TTZ$. 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 ith 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 ε 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 $r = |\text{IFST}-\text{ILST}|$.

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: $\tilde{T} = Z^TTZ$. 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 $AQ_1 = Q_1T_{11}$, 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 λ_j, SELECT(j) must be set .TRUE.. To select a complex conjugate pair of eigenvalues λ_j and λ_{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 $\geq$ 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 $\geq$ 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 $\geq$ max(1,N) if COMPQ = 'V',
LDQ $\geq$ 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); $0 \le m \le n$.

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 ≥ max(1,N),
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'.

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 ≥ 1,
if JOB = 'V' or 'B', then LIWORK ≥ 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 ith 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 ε 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; $\mathrm{WR}(i) = \tilde{t}_{ii}$ for $i = 1,2,\ldots,n$ and $\mathrm{WI}(i) = 0.0$.

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 $\alpha\pm\sqrt{\beta\gamma}$; $\mathrm{WR}(i) = \mathrm{WR}(i+1) = \alpha$; $\mathrm{WI}(i) = +\sqrt{|\beta\gamma|}$; $\mathrm{WI}(i+1) = -\mathrm{WI}(i)$.

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)); α is a scale factor (≤ 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 = '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 α.

13: INFO – INTEGER. *Output*

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

6. Error Indicators and Warnings

INFO < 0

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

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 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 $\tilde{X} = Q_1^TXQ_2$ and $\tilde{C} = Q_1^TCQ_2$. $\tilde{C}$ 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 λ, are defined by:

$$Tx = \lambda x \text{ and } y^H T = \lambda y^H \text{ (or } T^T y = \bar{\lambda} y\text{)}.$$

Note that even though T is real, λ, x and y may be complex. If x is an eigenvector corresponding to a complex eigenvalue λ, 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 λ_j, SELECT(j) must be set .TRUE.. To select the complex eigenvector corresponding to a complex conjugate pair of eigenvalues λ_j and λ_{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 $\geq$ 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 $\geq$ 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 $0 \le m \le n$.

Constraint: MM $\ge m$.

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 ith 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:

$$\theta(\tilde{x}_i,x_i) \leq \frac{c(n)\epsilon\|T\|_2}{sep_i}$$

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 λ_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 λ_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 λ_i is then given by

$$\frac{\epsilon\|T\|}{s_i},$$

where ϵ is the ***machine precision.***

To estimate the reciprocal of the condition number of the right eigenvector corresponding to λ_i, the routine first calls F08QFF (STREXC/DTREXC) to reorder the eigenvalues so that λ_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 λ_i is then given by

$$\frac{\epsilon\|T\|}{sep_i}.$$

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 λ_j, SELECT(j) must be set .TRUE.. To select condition numbers corresponding to a complex conjugate pair of eigenvalues λ_j and λ_{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 $\geq$ 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 $\geq$ 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 jth columns of VL and VR all correspond to the same eigenpair (but not in general the jth 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 $0 \le m \le n$.

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 ith 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: $\tilde{T} = Z^HTZ$. 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 ith 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 ε 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: $\tilde{T} = Z^HTZ$. 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 $AQ_1 = Q_1T_{11}$, 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 λ_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$(1,m\times(\mathrm{N}-m))$,
if JOB = 'V' or 'B', then LWORK ≥ max$(1,2\times m\times(\mathrm{N}-m))$.

The actual amount of workspace required cannot exceed $\mathrm{N}^2/4$ if JOB = 'E' or $\mathrm{N}^2/2$ 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 ith 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 ε 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; α is a scale factor (≤ 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 α.

13: INFO – INTEGER. *Output*

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

6. Error Indicators and Warnings

INFO < 0

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

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:

$$\|\tilde{X} - X\|_F \le \frac{\|R\|_F}{sep(A,B)}$$

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 $\tilde{X} = Q_1^HXQ_2$ and $\tilde{C} = Q_1^HCQ_2$. $\tilde{C}$ 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 λ, 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 λ_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 $\geq$ 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 F08QXF (CTREVC/ZTREVC) is called.

Constraint: LDT $\geq$ 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 $\geq$ max(1,N) if JOB = 'L' or 'B',
LDVL $\geq$ 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 ith 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 λ_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 λ_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 λ_i is then given by

$$\frac{\varepsilon\|T\|}{s_i},$$

where ε is the ***machine precision***.

To estimate the reciprocal of the condition number of the right eigenvector corresponding to λ_i, the routine first calls F08QTF (CTREXC/ZTREXC) to reorder the eigenvalues so that λ_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 λ_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 λ_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 ≥ 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 ≥ 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 jth columns of VL and VR all correspond to the same eigenpair (but not in general the jth 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 $0 \le m \le n$.

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 ith 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',
$C = L^TAL$ if UPLO = 'L'.

Constraint: $1 \le$ 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 $B = U^TU$;

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 ≥ 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 $\ge$ 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 $\ge$ max(1,N).

8: INFO – INTEGER. *Output*

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

6. Error Indicators and Warnings

INFO < 0

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

7. Accuracy

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: $1 \le \text{ITYPE} \le 3$.

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 ith 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' 'L'	U^TU LL^T	$U^{-T}AU^{-1}$ $L^{-1}AL^{-T}$	$U^{-1}y$ $L^{-T}y$
2	$ABz = \lambda z$	'U' 'L'	U^TU LL^T	UAU^T L^TAL	$U^{-1}y$ $L^{-T}y$
3	$BAz = \lambda z$	'U' 'L'	U^TU LL^T	UAU^T L^TAL	U^Ty 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',
$C = L^TAL$ if UPLO = 'L'.

Constraint: $1 \le$ 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 $B = U^TU$;

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

On exit: 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 ith 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',
$C = L^HAL$ if UPLO = 'L'.

Constraint: $1 \leq$ 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 $B = U^HU$;

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$(i+(2n-j)(j-1)/2)$ 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 ith 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
```

F08UEF (SSBGST/DSBGST) – 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

F08UEF (SSBGST/DSBGST) reduces a real symmetric-definite generalized eigenproblem $Az = \lambda Bz$ to the standard form $Cy = \lambda y$, where A and B are band matrices, A is a real symmetric matrix, and B has been factorized by F08UFF (SPBSTF/DPBSTF).

2 Specification

```
      SUBROUTINE F08UEF(VECT, UPLO, N, KA, KB, AB, LDAB, BB, LDBB, X,
     1                  LDX, WORK, INFO)
      ENTRY      ssbgst(VECT, UPLO, N, KA, KB, AB, LDAB, BB, LDBB, X,
     1                  LDX, WORK, INFO)
      INTEGER           N, KA, KB, LDAB, LDBB, LDX, INFO
      real              AB(LDAB,*), BB(LDBB,*), X(LDX,*), WORK(*)
      CHARACTER*1       VECT, UPLO
```

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

3 Description

To reduce the real symmetric-definite generalized eigenproblem $Az = \lambda Bz$ to the standard form $Cy = \lambda y$, where A, B and C are banded, this routine must be preceded by a call to F08UFF (SPBSTF/DPBSTF) which computes the split Cholesky factorization of the positive-definite matrix B: $B = S^T S$. The split Cholesky factorization, compared with the ordinary Cholesky factorization, allows the work to be approximately halved.

This routine overwrites A with $C = X^T AX$, where $X = S^{-1}Q$ and Q is a orthogonal matrix chosen (implicitly) to preserve the bandwidth of A. The routine also has an option to allow the accumulation of X, and then, if z is an eigenvector of C, Xz is an eigenvector of the original system.

4 References

[1] Crawford C R (1973) Reduction of a band-symmetric generalized eigenvalue problem *Comm. ACM* **16** 41–44

[2] Kaufman L (1984) Banded eigenvalue solvers on vector machines *ACM Trans. Math. Software* **10** 73–86

5 Parameters

1: VECT — CHARACTER*1 *Input*

On entry: indicates whether X is to be returned as follows:

if VECT = 'N', then X is not returned;
if VECT = 'V', then X is returned.

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

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 matrices A and B.

Constraint: N $\geq$ 0.

4: KA — INTEGER *Input*

On entry: k_A, the number of super-diagonals of the matrix A if UPLO = 'U', or the number of sub-diagonals if UPLO = 'L'.

Constraint: KA $\geq$ 0.

5: KB — INTEGER *Input*

On entry: k_B, the number of super-diagonals of the matrix B if UPLO = 'U', or the number of sub-diagonals if UPLO = 'L'.

Constraint: KA $\geq$ KB $\geq$ 0.

6: 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_A+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($k_A+1+i-j,j$) for max(1,$j-k_A$) $\leq i \leq j$; if UPLO = 'L', the elements of the lower triangle of A within the band must be stored with element a_{ij} in AB($1+i-j,j$) for $j \leq i \leq \min(n, j+k_A)$.

On exit: the upper or lower triangle of A is overwritten by the corresponding upper or lower triangle of C as specified by UPLO.

7: LDAB — INTEGER *Input*

On entry: the first dimension of the array AB as declared in the (sub)program from which F08UEF (SSBGST/DSBGST) is called.

Constraint: LDAB $\geq$ KA+1.

8: BB(LDBB,*) — ***real*** array *Input*

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

On entry: the banded split Cholesky factor of B as specified by UPLO, N and KB and returned by F08UFF (SPBSTF/DPBSTF).

9: LDBB — INTEGER *Input*

On entry: the first dimension of the array BB as declared in the (sub)program from which F08UEF (SSBGST/DSBGST) is called.

Constraint: LDBB $\geq$ KB+1.

10: X(LDX,*) — ***real*** array *Output*

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

On exit: the n by n matrix $X = S^{-1}Q$, if VECT = 'V'.

X is not referenced if VECT = 'N'.

11: LDX — INTEGER *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which F08UEF (SSBGST/DSBGST) is called.

Constraints:

LDX $\geq$ max(1,N) if VECT = 'V',
LDX $\geq$ 1 if VECT = 'N'.

12: WORK(*) — ***real*** array *Workspace*

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

13: INFO — INTEGER *Output*

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

6 Error Indicators and Warnings

INFO < 0

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

7 Accuracy

Forming the reduced matrix C is a stable procedure. However it involves implicit multiplication by B^{-1}. 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.

8 Further Comments

The total number of floating-point operations is approximately $6n^2k_B$, when VECT = 'N', assuming $n \gg k_A, k_B$; there are an additional $(3/2)n^3(k_B/k_A)$ operations when VECT = 'V'.

The complex analogue of this routine is F08USF (CHBGST/ZHBGST).

9 Example

To compute all the eigenvalues of $Az = \lambda Bz$, where

$$A = \begin{pmatrix} 0.24 & 0.39 & 0.42 & 0.00 \\ 0.39 & -0.11 & 0.79 & 0.63 \\ 0.42 & 0.79 & -0.25 & 0.48 \\ 0.00 & 0.63 & 0.48 & -0.03 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 2.07 & 0.95 & 0.00 & 0.00 \\ 0.95 & 1.69 & -0.29 & 0.00 \\ 0.00 & -0.29 & 0.65 & -0.33 \\ 0.00 & 0.00 & -0.33 & 1.17 \end{pmatrix}.$$

Here A is symmetric, B is symmetric positive-definite, and A and B are treated as band matrices. B must first be factorized by F08UFF (SPBSTF/DPBSTF). The program calls F08UEF (SSBGST/DSBGST) to reduce the problem to the standard form $Cy = \lambda y$, then F08HEF (SSBTRD/DSBTRD) 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.

```
*     F08UEF Example Program Text.
*     Mark 19 Release. NAG Copyright 1999.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, KMAX, LDAB, LDBB, LDX
      PARAMETER        (NMAX=8,KMAX=8,LDAB=KMAX-1,LDBB=KMAX-1,LDX=NMAX)
*     .. Local Scalars ..
      INTEGER          I, INFO, J, KA, KB, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      real             AB(LDAB,NMAX), BB(LDBB,NMAX), D(NMAX), E(NMAX-1),
     +                 WORK(2*NMAX), X(LDX,NMAX)
*     .. External Subroutines ..
      EXTERNAL         spbstf, ssbgst, ssbtrd, ssterf
*     .. Intrinsic Functions ..
      INTRINSIC        MAX, MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08UEF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, KA, KB
      IF (N.LE.NMAX .AND. KA.LE.KMAX .AND. KB.LE.KA) THEN
*
*        Read A and B from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            DO 20 I = 1, N
               READ (NIN,*) (AB(KA+1+I-J,J),J=I,MIN(N,I+KA))
   20       CONTINUE
            DO 40 I = 1, N
               READ (NIN,*) (BB(KB+1+I-J,J),J=I,MIN(N,I+KB))
   40       CONTINUE
         ELSE IF (UPLO.EQ.'L') THEN
            DO 60 I = 1, N
               READ (NIN,*) (AB(1+I-J,J),J=MAX(1,I-KA),I)
   60       CONTINUE
            DO 80 I = 1, N
               READ (NIN,*) (BB(1+I-J,J),J=MAX(1,I-KB),I)
   80       CONTINUE
         END IF
*
*        Compute the split Cholesky factorization of B
*
         CALL spbstf(UPLO,N,KB,BB,LDBB,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 ssbgst('N',UPLO,N,KA,KB,AB,LDAB,BB,LDBB,X,LDX,WORK,
     +                  INFO)
*
*           Reduce C to tridiagonal form T = (Q**T)*C*Q
*
            CALL ssbtrd('N',UPLO,N,KA,AB,LDAB,D,E,X,LDX,WORK,INFO)
*
*           Calclate 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,(8F8.4))
      END
```

9.2 Program Data

```
F08UEF Example Program Data
   4  2  1                        :Values of N, KA and KB
   'L'                            :Value of UPLO
   0.24
   0.39  -0.11
   0.42   0.79  -0.25
          0.63   0.48  -0.03      :End of matrix A
   2.07
   0.95   1.69
         -0.29   0.65
                -0.33   1.17      :End of matrix B
```

9.3 Program Results

```
F08UEF Example Program Results

Eigenvalues
   -0.8305 -0.6401  0.0992  1.8525
```

F08UFF (SPBSTF/DPBSTF) – 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

F08UFF (SPBSTF/DPBSTF) computes a split Cholesky factorization of a real symmetric positive-definite band matrix.

2 Specification

```
SUBROUTINE F08UFF(UPLO, N, KB, BB, LDBB, INFO)
ENTRY      spbstf(UPLO, N, KB, BB, LDBB, INFO)
INTEGER          N, KB, LDBB, INFO
real             BB(LDBB,*)
CHARACTER*1      UPLO
```

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

3 Description

This routine computes a split Cholesky factorization of a real symmetric positive-definite band matrix B. It is designed to be used in conjunction with F08UEF (SSBGST/DSBGST).

The factorization has the form $B = S^TS$, where S is a band matrix of the same bandwidth as B and the following structure: S is upper triangular in the first $(n+k)/2$ rows, and transposed — hence, lower triangular — in the remaining rows. For example, if $n = 9$ and $k = 2$, then

$$S = \begin{pmatrix} s_{11} & s_{12} & s_{13} & & & & & & \\ & s_{22} & s_{23} & s_{24} & & & & & \\ & & s_{33} & s_{34} & s_{35} & & & & \\ & & & s_{44} & s_{45} & & & & \\ & & & & s_{55} & & & & \\ & & & s_{64} & s_{65} & s_{66} & & & \\ & & & & s_{75} & s_{76} & s_{77} & & \\ & & & & & s_{86} & s_{87} & s_{88} & \\ & & & & & & s_{97} & s_{98} & s_{99} \end{pmatrix}.$$

4 References

None.

5 Parameters

1: UPLO — CHARACTER*1 *Input*

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

if UPLO = 'U', then the upper triangular part of B is stored;
if UPLO = 'L', then the lower triangular part of B is stored.

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

2: N — INTEGER *Input*

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

Constraint: N $\geq$ 0.

3: KB — INTEGER *Input*

On entry: k, the number of super-diagonals of the matrix B if UPLO = 'U', or the number of sub-diagonals if UPLO = 'L'.

Constraint: KB $\geq$ 0.

4: BB(LDBB,*) — ***real*** array *Input/Output*

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

On entry: the n by n symmetric positive-definite band matrix B, stored in rows 1 to $k + 1$. More precisely, if UPLO = 'U', the elements of the upper triangle of B within the band must be stored with element b_{ij} in BB$(k+1+i-j,j)$ for max$(1,j-k) \leq i \leq j$; if UPLO = 'L', the elements of the lower triangle of B within the band must be stored with element b_{ij} in BB$(1+i-j,j)$ for $j \leq i \leq \min(n,j+k)$.

On exit: B is overwritten by the elements of its split Cholesky factor S.

5: LDBB — INTEGER *Input*

On entry: the first dimension of the array BB as declared in the (sub)program from which F08UFF (SPBSTF/DPBSTF) is called.

Constraint: LDBB $\geq$ KB+1.

6: INFO — INTEGER *Output*

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

6 Error Indicators and Warnings

INFO < 0

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

INFO > 0

If INFO = i, the factorization could not be completed, because the updated element b_{ii} would be the square root of a negative number. Hence B is not positive-definite. This may indicate an error in forming the matrix B.

7 Accuracy

The computed factor S is the exact factor of a perturbed matrix $B + E$, where

$$|E| \leq c(k+1)\varepsilon|S^T||S|,$$

$c(k+1)$ is a modest linear function of k+1, and ε is the ***machine precision***. It follows that $|e_{ij}| \leq c(k+1)\varepsilon\sqrt{(b_{ii}b_{jj})}$.

8 Further Comments

The total number of floating-point operations is approximately $n(k+1)^2$, assuming $n \gg k$.

A call to this routine may be followed by a call to F08UEF (SSBGST/DSBGST) to solve the generalized eigenproblem $Az = \lambda Bz$, where A and B are banded and B is positive-definite.

The complex analogue of this routine is F08UTF (CPBSTF/ZPBSTF).

9 Example

See the example for F08UEF (SSBGST/DSBGST).

F08USF (CHBGST/ZHBGST) – 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

F08USF (CHBGST/ZHBGST) reduces a complex Hermitian-definite generalized eigenproblem $Az = \lambda Bz$ to the standard form $Cy = \lambda y$, where A and B are band matrices, A is a complex Hermitian matrix, and B has been factorized by F08UTF (CPBSTF/ZPBSTF).

2 Specification

```
      SUBROUTINE F08USF(VECT, UPLO, N, KA, KB, AB, LDAB, BB, LDBB, X,
     1                  LDX, WORK, RWORK,INFO)
      ENTRY      chbgst(VECT, UPLO, N, KA, KB, AB, LDAB, BB, LDBB, X,
     1                  LDX, WORK, RWORK,INFO)
      INTEGER           N, KA, KB, LDAB, LDBB, LDX, INFO
      complex           AB(LDAB,*), BB(LDBB,*), X(LDX,*), WORK(*)
      real              RWORK(*)
      CHARACTER*1       VECT, UPLO
```

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

3 Description

To reduce the complex Hermitian-definite generalized eigenproblem $Az = \lambda Bz$ to the standard form $Cy = \lambda y$, where A, B and C are banded, this routine must be preceded by a call to F08UTF (CPBSTF/ZPBSTF) which computes the split Cholesky factorization of the positive-definite matrix B: $B = S^H S$. The split Cholesky factorization, compared with the ordinary Cholesky factorization, allows the work to be approximately halved.

This routine overwrites A with $C = X^H AX$, where $X = S^{-1}Q$ and Q is a unitary matrix chosen (implicitly) to preserve the bandwidth of A. The routine also has an option to allow the accumulation of X, and then, if z is an eigenvector of C, Xz is an eigenvector of the original system.

4 References

[1] Crawford C R (1973) Reduction of a band-symmetric generalized eigenvalue problem *Comm. ACM* **16** 41–44

[2] Kaufman L (1984) Banded eigenvalue solvers on vector machines *ACM Trans. Math. Software* **10** 73–86

5 Parameters

1: VECT — CHARACTER*1 *Input*

On entry: indicates whether X is to be returned as follows:

if VECT = 'N', then X is not returned;
if VECT = 'V', then X is returned.

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

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 matrices A and B.

Constraint: N $\geq$ 0.

4: KA — INTEGER *Input*

On entry: k_A, the number of super-diagonals of the matrix A if UPLO = 'U', or the number of sub-diagonals if UPLO = 'L'.

Constraint: KA $\geq$ 0.

5: KB — INTEGER *Input*

On entry: k_B, the number of super-diagonals of the matrix B if UPLO = 'U', or the number of sub-diagonals if UPLO = 'L'.

Constraint: KA $\geq$ KB $\geq$ 0.

6: 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 $k_A + 1$. 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($k_A + 1 + i - j, j$) for max(1,$j - k_A$) $\leq i \leq j$; if UPLO = 'L', the elements of the lower triangle of A within the band must be stored with element a_{ij} in AB($1 + i - j, j$) for $j \leq i \leq \min(n, j + k_A)$.

On exit: the upper or lower triangle of A is overwritten by the corresponding upper or lower triangle of C as specified by UPLO.

7: LDAB — INTEGER *Input*

On entry: the first dimension of the array AB as declared in the (sub)program from which F08USF (CHBGST/ZHBGST) is called.

Constraint: LDAB $\geq$ KA+1.

8: BB(LDBB,*) — ***complex*** array *Input*

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

On entry: the banded split Cholesky factor of B as specified by UPLO, N and KB and returned by F08UTF (CPBSTF/ZPBSTF).

9: LDBB — INTEGER *Input*

On entry: the first dimension of the array BB as declared in the (sub)program from which F08USF (CHBGST/ZHBGST) is called.

Constraint: LDBB $\geq$ KB+1.

10: X(LDX,*) — ***complex*** array *Output*

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

On exit: the n by n matrix $X = S^{-1}Q$, if VECT = 'V'.

X is not referenced if VECT = 'N'.

11: LDX — INTEGER *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which F08USF (CHBGST/ZHBGST) is called.

Constraints:

LDX $\geq$ max(1,N) if VECT = 'V',
LDX $\geq$ 1 if VECT = 'N'.

12: WORK(*) — ***complex*** array *Workspace*

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

13: RWORK(*) — ***real*** array *Workspace*

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

14: INFO — INTEGER *Output*

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

6 Error Indicators and Warnings

INFO < 0

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

7 Accuracy

Forming the reduced matrix C is a stable procedure. However it involves implicit multiplication by B^{-1}. 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.

8 Further Comments

The total number of real floating-point operations is approximately $20n^2k_B$, when VECT = 'N', assuming $n \gg k_A, k_B$; there are an additional $5n^3(k_B/k_A)$ operations when VECT = 'V'.

The real analogue of this routine is F08UEF (SSBGST/DSBGST).

9 Example

To compute all the eigenvalues of $Az = \lambda Bz$, where

$$A = \begin{pmatrix} -1.13+0.00i & 1.94-2.10i & -1.40+0.25i & 0.00+0.00i \\ 1.94+2.10i & -1.91+0.00i & -0.82-0.89i & -0.67+0.34i \\ -1.40-0.25i & -0.82+0.89i & -1.87+0.00i & -1.10-0.16i \\ 0.00+0.00i & -0.67-0.34i & -1.10+0.16i & 0.50+0.00i \end{pmatrix}$$

and

$$B = \begin{pmatrix} 9.89+0.00i & 1.08-1.73i & 0.00+0.00i & 0.00+0.00i \\ 1.08+1.73i & 1.69+0.00i & -0.04+0.29i & 0.00+0.00i \\ 0.00+0.00i & -0.04-0.29i & 2.65+0.00i & -0.33+2.24i \\ 0.00+0.00i & 0.00+0.00i & -0.33-2.24i & 2.17+0.00i \end{pmatrix}.$$

Here A is Hermitian, B is Hermitian positive-definite, and A and B are treated as band matrices. B must first be factorized by F08UTF (CPBSTF/ZPBSTF). The program calls F08USF (CHBGST/ZHBGST) to reduce the problem to the standard form $Cy = \lambda y$, then F08HSF (CHBTRD/ZHBTRD) 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.

```
*     F08USF Example Program Text.
*     Mark 19 Release. NAG Copyright 1999.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, KMAX, LDAB, LDBB, LDX
      PARAMETER        (NMAX=8,KMAX=8,LDAB=KMAX-1,LDBB=KMAX-1,LDX=NMAX)
*     .. Local Scalars ..
      INTEGER          I, INFO, J, KA, KB, N
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          AB(LDAB,NMAX), BB(LDBB,NMAX), WORK(NMAX),
     +                 X(LDX,NMAX)
      real             D(NMAX), E(NMAX-1), RWORK(NMAX)
*     .. External Subroutines ..
      EXTERNAL         ssterf, chbgst, chbtrd, cpbstf
*     .. Intrinsic Functions ..
      INTRINSIC        MAX, MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F08USF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, KA, KB
      IF (N.LE.NMAX .AND. KA.LE.KMAX .AND. KB.LE.KA) THEN
*
*        Read A and B from data file
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            DO 20 I = 1, N
               READ (NIN,*) (AB(KA+1+I-J,J),J=I,MIN(N,I+KA))
   20       CONTINUE
            DO 40 I = 1, N
               READ (NIN,*) (BB(KB+1+I-J,J),J=I,MIN(N,I+KB))
   40       CONTINUE
         ELSE IF (UPLO.EQ.'L') THEN
            DO 60 I = 1, N
               READ (NIN,*) (AB(1+I-J,J),J=MAX(1,I-KA),I)
   60       CONTINUE
            DO 80 I = 1, N
               READ (NIN,*) (BB(1+I-J,J),J=MAX(1,I-KB),I)
   80       CONTINUE
         END IF
*
*        Compute the split Cholesky factorization of B
*
         CALL cpbstf(UPLO,N,KB,BB,LDBB,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 chbgst('N',UPLO,N,KA,KB,AB,LDAB,BB,LDBB,X,LDX,WORK,
     +                 RWORK,INFO)
*
*          Reduce C to tridiagonal form T = (Q**H)*C*Q
*
           CALL chbtrd('N',UPLO,N,KA,AB,LDAB,D,E,X,LDX,WORK,INFO)
*
*          Calclate 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,(8F8.4))
      END
```

9.2 Program Data

```
F08USF Example Program Data
  4  2  1                                            :Values of N, KA and KB
 'L'                                                 :Value of UPLO
 (-1.13, 0.00)
 ( 1.94, 2.10) (-1.91, 0.00)
 (-1.40,-0.25) (-0.82, 0.89) (-1.87, 0.00)
               (-0.67,-0.34) (-1.10, 0.16) ( 0.50, 0.00)    :End of matrix A
 ( 9.89, 0.00)
 ( 1.08, 1.73) ( 1.69, 0.00)
               (-0.04,-0.29) ( 2.65, 0.00)
                             (-0.33,-2.24) ( 2.17, 0.00)    :End of matrix B
```

9.3 Program Results

```
F08USF Example Program Results

Eigenvalues
   -6.6089 -2.0416  0.1603  1.7712
```

F08UTF (CPBSTF/ZPBSTF) – 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

F08UTF (CPBSTF/ZPBSTF) computes a split Cholesky factorization of a complex Hermitian positive-definite band matrix.

2 Specification

```
SUBROUTINE F08UTF(UPLO, N, KB, BB, LDBB, INFO)
ENTRY      cpbstf(UPLO, N, KB, BB, LDBB, INFO)
INTEGER          N, KB, LDBB, INFO
complex          BB(LDBB,*)
CHARACTER*1      UPLO
```

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

3 Description

This routine computes a split Cholesky factorization of a complex Hermitian positive-definite band matrix B. It is designed to be used in conjunction with F08USF (CHBGST/ZHBGST).

The factorization has the form $B = S^H S$, where S is a band matrix of the same bandwidth as B and the following structure: S is upper triangular in the first $(n+k)/2$ rows, and transposed — hence, lower triangular — in the remaining rows. For example, if $n = 9$ and $k = 2$, then

$$S = \begin{pmatrix} s_{11} & s_{12} & s_{13} & & & & & & \\ & s_{22} & s_{23} & s_{24} & & & & & \\ & & s_{33} & s_{34} & s_{35} & & & & \\ & & & s_{44} & s_{45} & & & & \\ & & & & s_{55} & & & & \\ & & & s_{64} & s_{65} & s_{66} & & & \\ & & & & s_{75} & s_{76} & s_{77} & & \\ & & & & & s_{86} & s_{87} & s_{88} & \\ & & & & & & s_{97} & s_{98} & s_{99} \end{pmatrix}.$$

4 References

None.

5 Parameters

1: UPLO — CHARACTER*1 *Input*

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

if UPLO = 'U', then the upper triangular part of B is stored;
if UPLO = 'L', then the lower triangular part of B is stored.

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

2: N — INTEGER *Input*

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

Constraint: N $\geq$ 0.

3: KB — INTEGER *Input*

On entry: k, the number of super-diagonals of the matrix B if UPLO = 'U', or the number of sub-diagonals if UPLO = 'L'.

Constraint: KB $\geq$ 0.

4: BB(LDBB,*) — ***complex*** array *Input/Output*

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

On entry: the n by n Hermitian positive-definite band matrix B, stored in rows 1 to $k+1$. More precisely, if UPLO = 'U', the elements of the upper triangle of B within the band must be stored with element b_{ij} in BB$(k+1+i-j,j)$ for $\max(1,j-k) \leq i \leq j$; if UPLO = 'L', the elements of the lower triangle of B within the band must be stored with element b_{ij} in BB$(1+i-j,j)$ for $j \leq i \leq \min(n,j+k)$.

On exit: B is overwritten by the elements of its split Cholesky factor S.

5: LDBB — INTEGER *Input*

On entry: the first dimension of the array BB as declared in the (sub)program from which F08UTF (CPBSTF/ZPBSTF) is called.

Constraint: LDBB $\geq$ KB+1.

6: INFO — INTEGER *Output*

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

6 Error Indicators and Warnings

INFO < 0

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

INFO > 0

If INFO = i, the factorization could not be completed, because the updated element b_{ii} would be the square root of a negative number. Hence B is not positive-definite. This may indicate an error in forming the matrix B.

7 Accuracy

The computed factor S is the exact factor of a perturbed matrix $B+E$, where

$$|E| \leq c(k+1)\varepsilon|S^H||S|,$$

$c(k+1)$ is a modest linear function of k+1, and ε is the ***machine precision***. It follows that $|e_{ij}| \leq c(k+1)\varepsilon\sqrt{(b_{ii}b_{jj})}$.

8 Further Comments

The total number of floating-point operations is approximately $4n(k+1)^2$, assuming $n \gg k$.

A call to this routine may be followed by a call to F08USF (CHBGST/ZHBGST) to solve the generalized eigenproblem $Az = \lambda Bz$, where A and B are banded and B is positive-definite.

The real analogue of this routine is F08UFF (SPBSTF/DPBSTF).

9 Example

See the example for F08USF (CHBGST/ZHBGST).

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
F11BDF	19	Real sparse nonsymmetric linear systems, set-up for F11BEF
F11BEF	19	Real sparse nonsymmetric linear systems, preconditioned RGMRES, CGS, Bi-CGSTAB or TFQMR method
F11BFF	19	Real sparse nonsymmetric linear systems, diagnostic for F11BEF
F11BRF	19	Complex sparse non-Hermitian linear systems, set-up for F11BSF
F11BSF	19	Complex sparse non-Hermitian linear systems, preconditioned RGMRES, CGS, Bi-CGSTAB or TFQMR method
F11BTF	19	Complex sparse non-Hermitian linear systems, diagnostic for F11BSF
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 preconditioning matrix generated 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)
F11DNF	19	Complex sparse non-Hermitian linear systems, incomplete *LU* factorization
F11DPF	19	Solution of complex linear system involving incomplete *LU* preconditioning matrix generated by F11DNF
F11DQF	19	Solution of complex sparse non-Hermitian linear system, RGMRES, CGS, Bi-CGSTAB or TFQMR method, preconditioner computed by F11DNF (Black Box)
F11DRF	19	Solution of linear system involving preconditioning matrix generated by applying SSOR to complex sparse non-Hermitian matrix
F11DSF	19	Solution of complex sparse non-Hermitian linear system, RGMRES, CGS, Bi-CGSTAB or TFQMR 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, preconditioned 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 Cholesky 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)
F11JNF	19	Complex sparse Hermitian matrix, incomplete Cholesky factorization
F11JPF	19	Solution of complex linear system involving incomplete Cholesky preconditioning matrix generated by F11JNF

F11JQF	19	Solution of complex sparse Hermitian linear system, conjugate gradient/Lanczos method, preconditioner computed by F11JNF (Black Box)
F11JRF	19	Solution of linear system involving preconditioning matrix generated by applying SSOR to complex sparse Hermitian matrix
F11JSF	19	Solution of complex sparse Hermitian 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
F11XNF	19	Complex sparse non-Hermitian matrix vector multiply
F11XSF	19	Complex sparse Hermitian matrix vector multiply
F11ZAF	18	Real sparse nonsymmetric matrix reorder routine
F11ZBF	17	Real sparse symmetric matrix reorder routine
F11ZNF	19	Complex sparse non-Hermitian matrix reorder routine
F11ZPF	19	Complex sparse Hermitian matrix reorder routine

** This routine has been superseded, although it will be retained in the Library until at least Mark 21. See the document 'Advice on Replacement Calls for Withdrawn/Superseded Routines' for details of the recommended replacement 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, complex non-Hermitian and Hermitian 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 the 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 users to choose their own preferred format, or to avoid storing the matrix altogether. Other routines are the so-called **black-boxes**, which are easier to use, but are based on fixed storage formats. Two such formats are currently provided. 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 three one-dimensional arrays — a real or complex array A and two integer arrays IROW and ICOL. These arrays are all 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 and Hermitian 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, complex non-Hermitian, real symmetric positive-definite and complex Hermitian 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 reached. 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 [5], 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 real nonsymmetric, complex non-Hermitian, real symmetric and complex Hermitian systems of equations is further considered in Section 2.4 and Section 2.5.

2.4 Iterative Methods for Real Nonsymmetric and Complex Non-Hermitian 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 real nonsymmetric and complex non-Hermitian matrices this class includes:

the restarted generalized minimum residual (RGMRES) method [10];

the conjugate gradient squared (CGS) method [12];

the polynomial stabilized bi-conjugate gradient (Bi-CGSTAB (ℓ)) method [13], [11];

the transpose-free quasi-minimal residual method (TFQMR) [3], [4].

Here we just give a brief overview of these algorithms as implemented in Chapter F11. For full details see the routine documents for F11BDF and F11BRF.

RGMRES is based on the Arnoldi method, which explicitly generates an orthogonal basis for the Krylov subspace $\operatorname{span}\{A^k r_0\}$, $k = 0, 1, 2, \ldots$, where r_0 is the initial residual. The solution is then expanded onto the orthogonal basis so as to minimize the residual norm. For real nonsymmetric and complex non-Hermitian 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 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 $A^H\hat{x} = \hat{b}$, 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 $\hat{r}_i = P_i(A^H)\hat{r}_0$ are generated, where P_i is a polynomial of order i, and bi-orthogonality is exploited by computing the vector product $\rho_i = (\hat{r}_i, r_i) = (P_i(A^H)\hat{r}_0, P_i(A)r_0) = (\hat{r}_0, P_i^2(A)r_0)$. 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 (ℓ)) 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 ℓ 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 [13]. 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 ℓ increases, numerical instabilities may arise.

The transpose-free quasi-minimal residual method (TFQMR) (Freund and Nachtigal [3], Freund [4]) is conceptually derived from the CGS method. The residual is minimised over the space of the residual vectors generated by the CGS iterations under the simplifying assumption that residuals are almost orthogonal. In practice, this is not the case but theoretical analysis has proved the validity of the method. This has the effect of remedying the rather irregular convergence behaviour with wild oscillations in the residual norm that can degrade the numerical performance and robustness of the CGS method. In general, the TFQMR method can be expected to converge at least as fast as the CGS method, in terms of number of iterations, although each iteration involves a higher operation count. When the CGS method exhibits irregular convergence, the TFQMR method can produce much smoother, almost monotonic convergence curves. However, the close relationship between the CGS and TFQMR method implies that the *overall* speed of convergence is similar for both methods. In some cases, the TFQMR method may converge faster than the CGS method.

Faster convergence can usually be achieved by using a **preconditioner**. A *left* preconditioner M^{-1} can be used by the RGMRES, CGS and TFQMR 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 (ℓ) 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 $v = A^H u$ (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 [8], or on the approximate inverses occurring in stationary iterative methods (see Young [14]). 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 Real Symmetric and Complex Hermitian Linear Systems

Two of the best known iterative methods applicable to real symmetric and complex Hermitian linear systems are the conjugate gradient (CG) method [6], [5] and a Lanczos type method based on SYMMLQ [9]. The description of these methods given below is for the real symmetric case. The generalization to complex Hermitian matrices is straightforward.

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 [5]). 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 [7], or on the approximate inverses occurring in stationary iterative methods (see Young [14]). A common example is the **incomplete**

Cholesky factorization

$$M = PLDL^TP^T = A - R$$

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-box routines.

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 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 also applies 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 suites of basic routines for real symmetric and nonsymmetric systems, and for complex non-Hermitian systems.

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 routine 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 Real Nonsymmetric and Complex Non-Hermitian Linear Systems

The suite of basic routines F11BDF, F11BEF and F11BFF implements either RGMRES, CGS, Bi-CGSTAB(ℓ), or TFQMR, for the iterative solution of the real 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(ℓ) seems robust and reliable, but it can be slower than the other methods. TFQMR can be viewed as a more robust variant of the CGS method: it shares the CGS method speed but avoids the CGS fluctuations in the residual, which may give, rise to instability. Some further discussion of the relative merits of these methods can be found in [1].

The utility routines provided for real 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 ω must currently be supplied by the user. Automatic procedures for choosing ω will be included in the chapter at a future mark.

F11XAF computes matrix-vector products for real nonsymmetric matrices stored in ordered CS format. An additional utility routine F11ZAF orders the non-zero elements of a real sparse nonsymmetric matrix stored in general CS format.

The black-box routine F11DCF makes calls to F11BDF, F11BEF, F11BFF, F11DBF and F11XAF, to solve a real sparse nonsymmetric linear system, represented in CS format, using RGMRES, CGS, Bi-CGSTAB(ℓ), or TFQMR, with incomplete LU preconditioning. F11DEF is similar, but has options for no preconditioning, Jacobi preconditioning or SSOR preconditioning.

For complex non-Hermitian sparse matrices there is an equivalent suite of routines. F11BRF, F11BSF and F11BTF are the basic routines which implement the same methods used for real nonsymmetric systems, namely RGMRES, CGS, Bi-CGSTAB(ℓ) and TFQMR, for the solution of complex sparse non-Hermitian linear systems. F11DNF and F11DPF are the complex equivalents of F11DAF and F11DBF, respectively, providing facilities for implementing ILU preconditioning. F11DRF implements a complex version of the SSOR preconditioner. Utility routines F11XNF and F11ZNF are provided for computing matrix-vector products and sorting the elements of complex sparse non-Hermitian matrices, respectively. Finally, the black-box routines F11DQF and F11DSF are complex equivalents of F11DCF and F11DEF.

3.3 Iterative Methods for Real Symmetric and Complex Hermitian 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 real 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 real 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 ω must currently be supplied by the user. Automatic procedures for choosing ω will be included in the chapter at a future mark.

F11XEF computes matrix-vector products for real symmetric matrices stored in ordered SCS format. An additional utility routine F11ZBF orders the non-zero elements of a real 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 real 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.

For complex Hermitian sparse matrices there is an equivalent suite of utility and black-box routines, although the basic routines are not available at this Mark. F11JNF and F11JPF are the complex equivalents of F11JAF and F11JBF, respectively, providing facilities for implementing incomplete Cholesky preconditioning. F11JRF implements a complex version of the SSOR preconditioner. Utility routines F11XSF and F11ZPF are provided for computing matrix-vector products and sorting the elements of complex sparse Hermitian matrices, respectively. Finally, the black-box routines F11JQF and F11JSF provide easy-to-use implementations of the CG and SYMMLQ methods for complex Hermitian linear systems.

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 the various preconditioner generation routines can be chosen in such a way as to produce a direct solution.

For example, 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.

Complex non-Hermitian systems can be solved directly in the same way using F11DNF and F11DPF, while for complex Hermitian systems F11JNF and F11JPF may be used.

Some 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

Basic routines for real nonsymmetric linear systems	
set-up routine	**F11BDF**
reverse communication RGMRES, CGS, Bi-CGSTAB(ℓ) or TFQMR solver routine	**F11BEF**
diagnostic routine	**F11BFF**
Basic routines for complex non-Hermitian linear systems	
set-up routine	**F11BRF**
reverse communication RGMRES, CGS, Bi-CGSTAB(ℓ) or TFQMR solver routine	**F11BSF**
diagnostic routine	**F11BTF**
Black-box routines for real nonsymmetric linear systems	
RGMRES, CGS, Bi-CGSTAB(ℓ) or TFQMR solver with incomplete *LU* preconditioning	**F11DCF**
RGMRES, CGS, Bi-CGSTAB(ℓ) or TFQMR solver with no preconditioning, Jacobi, or SSOR preconditioning	**F11DEF**
Black-box routines for complex non-Hermitian linear systems	
RGMRES, CGS, Bi-CGSTAB(ℓ) or TFQMR solver with incomplete *LU* preconditioning	**F11DQF**
RGMRES, CGS, Bi-CGSTAB(ℓ) or TFQMR solver with no preconditioning, Jacobi, or SSOR preconditioning	**F11DSF**
Utility routines for real nonsymmetric linear systems	
incomplete *LU* factorization	**F11DAF**
solver for linear systems involving preconditioning matrix from F11DAF	**F11DBF**
solver for linear systems involving SSOR preconditioning matrix	**F11DDF**
matrix-vector multiplier for real nonsymmetric matrices in CS format	**F11XAF**
sort routine for real nonsymmetric matrices in CS format	**F11ZAF**
Utility routines for complex non-Hermitian linear systems	
incomplete *LU* factorization	**F11DNF**
solver for linear systems involving preconditioning matrix from F11DNF	**F11DPF**
solver for linear systems involving SSOR preconditioning matrix	**F11DRF**
matrix-vector multiplier for complex non-Hermitian matrices in CS format	**F11XNF**
sort routine for complex non-Hermitian matrices in CS format	**F11ZNF**
Basic routines for real symmetric linear systems	
set-up routine	**F11GAF**
reverse communication CG or SYMMLQ solver routine	**F11GBF**
diagnostic routine	**F11GCF**
Black-box routines for real symmetric linear systems	
CG or SYMMLQ solver with incomplete Cholesky preconditioning	**F11JCF**
CG or SYMMLQ solver with no preconditioning, Jacobi, or SSOR preconditioning	**F11JEF**
Black-box routines for complex Hermitian linear systems	
CG or SYMMLQ solver with incomplete Cholesky preconditioning	**F11JQF**
CG or SYMMLQ solver with no preconditioning, Jacobi, or SSOR preconditioning	**F11JSF**
Utility routines for real symmetric linear systems	
incomplete Cholesky factorization	**F11JAF**
solver for linear systems involving preconditioning matrix from F11JAF	**F11JBF**
solver for linear systems involving SSOR preconditioning matrix	**F11JDF**
matrix-vector multiplier for real symmetric matrices in SCS format	**F11XEF**
sort routine for real symmetric matrices in SCS format	**F11ZBF**
Utility routines for complex Hermitian linear systems	
incomplete Cholesky factorization	**F11JNF**
solver for linear systems involving preconditioning matrix from F11JNF	**F11JPF**
solver for linear systems involving SSOR preconditioning matrix	**F11JRF**
matrix-vector multiplier for complex Hermitian matrices in SCS format	**F11XSF**
sort routine for complex Hermitian matrices in SCS format	**F11ZPF**

5 Routines Withdrawn or Scheduled for Withdrawal

Since Mark 13 the following routines have either been withdrawn or superseded. Those routines indicated by a dagger are still present at Mark 19, but will be omitted at a future date. Advice on replacing calls to these routines is given in the document 'Advice on Replacement Calls for Withdrawn/Superseded Routines'.

F11BAF† F11BBF† F11BCF†

6 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] Freund R W and Nachtigal N (1991) QMR: a Quasi-Minimal Residual Method for Non-Hermitian Linear Systems *Numer. Math.* **60** 315–339

[4] Freund R W (1993) A Transpose-Free Quasi-Mimimal Residual Algorithm for Non-Hermitian Linear Sytems *SIAM J. Sci. Comput.* **14** 470–482

[5] Golub G H and Van Loan C F (1996) *Matrix Computations* Johns Hopkins University Press (3rd Edition), Baltimore

[6] Hestenes M and Stiefel E (1952) Methods of conjugate gradients for solving linear systems *J. Res. Nat. Bur. Stand.* **49** 409–436

[7] 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

[8] 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

[9] Paige C C and Saunders M A (1975) Solution of sparse indefinite systems of linear equations *SIAM J. Numer. Anal.* **12** 617–629

[10] 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

[11] Sleijpen G L G and Fokkema D R (1993) BiCGSTAB(ℓ) for linear equations involving matrices with complex spectrum *ETNA* **1** 11–32

[12] Sonneveld P (1989) CGS, a fast Lanczos-type solver for nonsymmetric linear systems *SIAM J. Sci. Statist. Comput.* **10** 36–52

[13] 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

[14] 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 (ℓ) method (Bi-CGSTAB (ℓ)).

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 (ℓ) Method (Bi-CGSTAB (ℓ))

The bi-conjugate gradient stabilized (ℓ) method (Bi-CGSTAB (ℓ)) (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 ℓ 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 ℓ 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 (ℓ) methods above, the user's program must provide the **maximum** number of basis vectors used, m or ℓ, 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 (ℓ) 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

$$\|r_k\|_p \leq \tau\left(\|b\|_p + \|A\|_p\,\|x_k\|_p\right) \tag{2}$$

is also available for all three methods. In (2), $p = 1,\infty$ or 2 and τ denotes a user-specified tolerance subject to $\max(10,\sqrt{n}),\epsilon \leq \tau < 1$, where ϵ 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 (ℓ) 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 (ℓ) 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 (ℓ) seems robust and reliable, but it can be slower than the other methods: if a preconditioner is used and $\ell > 1$, Bi-CGSTAB (ℓ) 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 (ℓ) 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(ℓ) 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 (ℓ) 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_∞ 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 ℓ 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', $0 < \mathrm{M} \leq \min(\mathrm{N},10)$.

8: TOL — ***real*** *Input*

On entry: the tolerance τ for the termination criterion. If TOL $\leq$ 0.0, $\tau = \max(\sqrt{\epsilon},\sqrt{n}\,\epsilon)$ is used, where ϵ is the ***machine precision***. Otherwise $\tau = \max(\mathrm{TOL},10\epsilon,\sqrt{n}\,\epsilon)$ 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 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 σ_1 of the preconditioned iteration matrix; otherwise, SIGMAX is not referenced.

If SIGMAX $\leq$ 0.0, ITERM = 2 and METHOD = 'RGMRES', then the value of σ_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 ℓ iterations for RGMRES or Bi-CGSTAB (ℓ), respectively).

There are some additional computational costs involved in monitoring the solution and residual vectors when the Bi-CGSTAB (ℓ) 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 ith 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 σ_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 (ℓ) 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 (ℓ) 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 × 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 (ℓ), the bi-conjugate gradient stabilized method of order ℓ (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(ℓ) 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 (ℓ) 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 (ℓ)	LWORK $= 2n\,(\ell + 2) + \ell(\ell + 2) + p + q$, where ℓ 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 ith 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, x_k and $r_k = b - Ax_k$, respectively, at the kth 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 (ℓ) 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 $\|r_k\|_p \leq \tau\left(\|b\|_p + \|A\|_p\,\|x_k\|_p\right)$ 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.

F11BDF – 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

F11BDF is a set-up routine, the first in a suite of three routines for the iterative solution of a real general (nonsymmetric) system of simultaneous linear equations. F11BDF must be called before F11BEF, the iterative solver. The third routine in the suite, F11BFF, 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 F11BDF(METHOD, PRECON, NORM, WEIGHT, ITERM, N, M, TOL,
1                  MAXITN, ANORM, SIGMAX, MONIT, LWREQ, WORK,
2                  LWORK, IFAIL)
 INTEGER           ITERM, N, M, MAXITN, MONIT, LWREQ, LWORK, IFAIL
 real              TOL, ANORM, SIGMAX
 CHARACTER*1       PRECON, NORM, WEIGHT
 CHARACTER*(*)     METHOD
 real              WORK(LWORK)
```

3 Description

The suite consisting of the routines F11BDF, F11BEF, F11BFF is designed to solve the general (nonsymmetric) system of simultaneous linear equations $Ax = b$ of order n, where n is large and the coefficient matrix A is sparse.

F11BDF is a set-up routine which must be called before F11BEF, the iterative solver. The third routine in the suite, F11BFF, 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 (ℓ) method (Bi-CGSTAB (ℓ));

transpose-free quasi-minimal residual method (TFQMR).

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 (ℓ) Method (Bi-CGSTAB (ℓ))

The bi-conjugate gradient stabilized (ℓ) method (Bi-CGSTAB (ℓ)) (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 ℓ 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 ℓ increases, numerical instabilities may arise. For this reason, a maximum value of $\ell = 10$ is imposed, but probably $\ell = 4$ is sufficient in most cases.

3.4 Transpose-Free Quasi-Minimal Residual Method (TFQMR)

The transpose-free quasi-minimal residual method (TFQMR) (Freund and Nachtigal [10], Freund [11]) is conceptually derived from the CGS method. The residual is minimised over the space of the residual vectors generated by the CGS iterations under the simplifying assumption that residuals are almost orthogonal. In practice, this is not the case but theoretical analysis has proved the validity of the method. This has the effect of remedying the rather irregular convergence behaviour with wild oscillations in the residual norm that can degrade the numerical performance and robustness of the CGS method. In general, the TFQMR method can be expected to converge at least as fast as the CGS method, in terms of number of iterations, although each iteration involves a higher operation count. When the CGS method exhibits irregular convergence, the TFQMR method can produce much smoother, almost monotonic convergence curves. However, the close relationship between the CGS and TFQMR method implies that the *overall* speed of convergence is similar for both methods. In some cases, the TFQMR method may converge faster than the CGS method.

3.5 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 (ℓ) methods above, the user's program must provide the **maximum** number of basis vectors used, m or ℓ, 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 the 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, CGS and TFQMR 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 (ℓ) 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, i.e., explicit information about M, or its inverse is not required at any stage.

The first termination criterion

$$\|r_k\|_p \leq \tau \left(\|b\|_p + \|A\|_p \, \|x_k\|_p\right) \tag{2}$$

is also available for all three methods. In (2), $p = 1$, ∞ or 2 and τ denotes a user-specified tolerance subject to max$(10,\sqrt{n})$, $\epsilon \leq \tau < 1$, where ϵ 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

$$\|\bar{r}_k\|_2 \leq \tau \left(\|\bar{r}_0\|_2 + \sigma_1(\bar{A}) \, \|\Delta \bar{x}_k\|_2\right) \tag{3}$$

is available for all methods except TFQMR. 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 (ℓ) 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 (ℓ) 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 F11BDF must be called, followed by the solver F11BEF. F11BFF can be called either when F11BEF is carrying out a monitoring step or after F11BEF 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 (ℓ) seems robust and reliable, but it can be slower than the other methods: if a preconditioner is used and $\ell > 1$, Bi-CGSTAB (ℓ) 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. TFQMR can be viewed as a more robust variant of the CGS method: it shares the

CGS method speed but avoids the CGS fluctuations in the residual, which may give rise to instability. Also, when the termination criterion (2) is used, the CGS, Bi-CGSTAB (ℓ) and TFQMR 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 (1996) *Matrix Computations* Johns Hopkins University Press (3rd 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(ℓ) 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

[10] Freund R W and Nachtigal N (1991) QMR: a Quasi-Minimal Residual Method for Non-Hermitian Linear Systems *Numer. Math.* **60** 315–339

[11] Freund R W (1993) A Transpose-Free Quasi-Mimimal Residual Algorithm for Non-Hermitian Linear Systems *SIAM J. Sci. Comput.* **14** 470–482

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 (ℓ) method;
'TFQMR'	transpose-free quasi-minimal residual method.

Constraint: METHOD = 'RGMRES', 'CGS', 'BICGSTAB' or 'TFQMR'.

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 inthe termination criteria. The possible choices are:

'1' l_1 norm;
'I' l_∞ norm;
'2' 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 F11BEF.
'N' all weights are implicitly set equal to one. Weights do not need to be supplied on initial entry to F11BEF.

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: if METHOD = 'RGMRES', 'CGS' or 'BICGSTAB', ITERM = 1 or 2;

if METHOD = 'TFQMR', ITERM = 1.

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 ℓ 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', $0 < \mathrm{M} \leq \min(\mathrm{N},10)$.

8: TOL — ***real*** *Input*

On entry: the tolerance τ for the termination criterion. If TOL ≤ 0.0, $\tau = \max(\sqrt{\epsilon}, \sqrt{n}\,\epsilon)$ is used, where ϵ is the ***machine precision***. Otherwise $\tau = \max(\mathrm{TOL}, 10\epsilon, \sqrt{n}\,\epsilon)$ 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 F11BEF.

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 σ_1 of the preconditioned iteration matrix; otherwise, SIGMAX is not referenced.

If SIGMAX $\leq$ 0.0, ITERM = 2 and METHOD = 'RGMRES', then the value of σ_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 F11BEF: if METHOD = 'CGS' or 'TFQMR', a monitoring step is executed every MONIT iterations; otherwise, a monitoring step is executed every MONIT super-iterations (groups of up to m or ℓ iterations for RGMRES or Bi-CGSTAB (ℓ), respectively).

There are some additional computational costs involved in monitoring the solution and residual vectors when the Bi-CGSTAB (ℓ) method is used with $\ell > 1$.

Constraint: MONIT $\leq$ MAXITN.

13: LWREQ — INTEGER *Output*

On exit: the minimum amount of workspace required by F11BEF. (See also Section 5 of the document for F11BEF.)

14: WORK (LWORK) — ***real*** array *Output*

On exit: the workspace WORK is initialised by F11BDF. It must **not** be modified before calling the next routine in the suite, namely F11BEF.

15: LWORK — INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F11BDF was called.

Constraint: LWORK $\geq$ 100.

Note: although the minimum value of LWORK ensures the correct functioning of F11BDF, a larger value is required by the other routines in the suite, namely F11BEF and F11BFF. The required value is as follows:

Method	**Requirements**
RGMRES	LWORK $= 100 + n(m+3) + m(m+5) + 1$, where m is the dimension of the basis;
CGS	LWORK $= 100 + 7n$;
Bi-CGSTAB (ℓ)	LWORK $= 100 + (2n+\ell)(\ell+2) + p$, where ℓ is the order of the method;
TFQMR	LWORK $= 100 + 10n$.

where

$p = 2n$ if $\ell > 1$ and ITERM = 2 was supplied;

$p = n$ if $\ell > 1$ and a preconditioner is used or ITERM = 2 was supplied;

$p = 0$ otherwise.

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 ith argument had an illegal value.

IFAIL = 1

F11BDF has been called out of sequence.

7 Accuracy

Not applicable.

8 Further Comments

RGMRES can estimate internally the maximum singular value σ_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 (ℓ) 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 CGS, Bi-CGSTAB (ℓ) and TFQMR methods will restart automatically the computation from the last available iterates, when the stability of the solution process requires it.

Termination criterion (3), when available, 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 × 8 nonsymmetric system of simultaneous linear equations using the bi-conjugate gradient stabilized method of order $\ell = 2$, where the coefficients matrix A has a 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.

```
*     F11BDF Example Program Text.
*     Mark 19 Release. NAG Copyright 1999.
*     .. 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), WGT(NMAX), WORK(LWORK), X(NMAX)
      INTEGER          ICOL(LA), IDIAG(NMAX), IPIVP(NMAX), IPIVQ(NMAX),
     +                 IROW(LA), ISTR(NMAX+1), IWORK(LIWORK)
*     .. External Subroutines ..
      EXTERNAL         F11BDF, F11BEF, F11BFF, F11DAF, F11DBF, F11XAF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11BDF 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
*
```

```
      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 F11BDF to initialize the solver
*
      IFAIL = 0
      CALL F11BDF(METHOD,PRECON,NORM,WEIGHT,ITERM,N,M,TOL,MAXITN,
     +            ANORM,SIGMAX,MONIT,LWREQ,WORK,LWORK,IFAIL)
*
*     Call repeatedly F11BEF 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 F11BEF(IREVCM,X,B,WGT,WORK,LWREQ,IFAIL)
      IF (IREVCM.EQ.-1) THEN
         IFAILX = -1
         CALL F11XAF('Transpose',N,NNZ,A,IROW,ICOL,'No checking',X,B,
     +               IFAILX)
      ELSE IF (IREVCM.EQ.1) THEN
         IFAILX = -1
         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 F11BFF(ITN,STPLHS,STPRHS,ANORM,SIGMAX,WORK,LWREQ,
     +               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 F11BFF(ITN,STPLHS,STPRHS,ANORM,SIGMAX,WORK,LWREQ,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 (1X,A,I4)
99994 FORMAT (1X,A,1P,e14.4)
      END
```

9.2 Example Data

```
F11BDF Example Program Data
  8                        N
  'BICGSTAB'               METHOD
  'P'    '1'    'N'   1    PRECON, 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

```
F11BDF 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
```

F11BEF – 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

F11BEF is an iterative solver for a real general (nonsymmetric) system of simultaneous linear equations; F11BEF is the second in a suite of three routines, where the first routine, F11BDF, must be called prior to F11BEF to set-up the suite, and the third routine in the suite, F11BFF, 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 F11BEF(IREVCM, U, V, WGT, WORK, LWORK, IFAIL)
INTEGER           IREVCM, LWORK, IFAIL
real              U(*), V(*), WGT(*), WORK(LWORK)
```

3 Description

F11BEF solves the general (nonsymmetric) system of linear simultaneous equations $Ax = b$ using one of four available methods: RGMRES, the preconditioned restarted generalized minimum residual method (Saad and Schultz [1]); CGS, the preconditioned conjugate gradient squared method (Sonneveld [2]); Bi-CGSTAB (ℓ), the bi-conjugate gradient stabilized method of order ℓ (Van der Vorst [3], Sleijpen and Fokkema [4]); or TFQMR, the quasi-minimal transpose-free quasi-minimum residual method (Freund and Nachtigal [6], Freund [7]).

For a general description of the methods employed you are referred to Section 3 of the document for F11BDF.

F11BEF can solve the system after the first routine in the suite, F11BDF, has been called to initialize the computation and specify the method of solution. The third routine in the suite, F11BFF, can be used to return additional information generated by the computation during monitoring steps and after F11BEF has completed its tasks.

F11BEF 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 four 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(ℓ) 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

[6] Freund R W and Nachtigal N (1991) QMR: a Quasi-Minimal Residual Method for Non-Hermitian Linear Systems *Numer. Math.* **60** 315–339

[7] Freund R W (1993) A Transpose-Free Quasi-Mimimal Residual Algorithm for Non-Hermitian Linear Sytems *SIAM J. Sci. Comput.* **14** 470–482

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. F11BEF will then return with the termination code IREVCM = 4. Note that before calling F11BEF with IREVCM = 5 the calling program must have performed the tasks required by the value of IREVCM returned by the previous call to F11BEF, otherwise subsequently returned values may be invalid.

6 Immediate termination: F11BEF 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. F11BEF 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 (ℓ) 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 F11BDF.

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. F11BFF can be called at this step to return additional information.

On final exit: IREVCM = 4: F11BEF 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 F11BEF. If IFAIL = 1, the array U is unchanged from the last entry to F11BEF. 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 F11BEF;

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 F11BEF. If IFAIL = 1, the array V is unchanged from the last entry to F11BEF. 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: WGT($*$) — ***real*** array *Input*

Note: the dimension of the array WGT must be at least max$(1, n)$ if weights are used, 1 otherwise.

On entry: the user-supplied weights, if these are to be used in the computation of the vector norms in the termination criterion (see Sections 3 and 5 of the document for F11BDF).

5: WORK (LWORK) — ***real*** array *Input/Output*

On initial entry: the workspace WORK as returned by F11BDF (see also Section 5 of the document for F11BDF).

On intermediate re-entry: WORK must remain unchanged.

6: LWORK — INTEGER *Input*

On initial entry: the dimension of the array WORK as declared in the (sub)program from which F11BEF was called (see also Sections 3 and 5 of the document for F11BDF). The required amount of workspace is as follows:

Method	**Requirements**
RGMRES	LWORK $= 100 + n(m+3) + m(m+5) + 1$, where m is the dimension of the basis;
CGS	LWORK $= 100 + 7n$;
Bi-CGSTAB (ℓ)	LWORK $= 100 + (2n+\ell)(\ell+2) + p$, where ℓ is the order of the method;
TFQMR	LWORK $= 100 + 10n$;

where

$p = 2n$ if $\ell > 1$ and ITERM $= 2$ was supplied;
$p = n$ if $\ell > 1$ and a preconditioner is used or ITERM $= 2$ was supplied;
$p = 0$ otherwise.

Constraint: LWORK $\geq$ LWREQ, where LWREQ is returned by F11BDF.

7: 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 ith argument had an illegal value.

IFAIL $= 1$

F11BEF 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 F11BFF have remained unchanged.

IFAIL $= 2$

The required accuracy could not be obtained. However, F11BEF 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 F11BEF 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 F11BFF to check the values of the left-and right-hand side of the termination condition.

IFAIL $= 3$

F11BDF was either not called before calling F11BEF 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 and TFQMR methods 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 kth 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 F11BDF. The computed values of the left- and right-hand sides of the termination criterion selected can be obtained by a call to F11BFF.

8 Further Comments

The number of operations carried out by F11BEF 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 F11BEF 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, CGS and TFQMR methods) or $\bar{A} = AM^{-1}$ (Bi-CGSTAB (ℓ) method).

Additional matrix-vector products are required for the computation of $\|A\|_1$ or $\|A\|_\infty$, when this has not been supplied to F11BDF 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 F11BDF) 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: F11BEF 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 F11BDF.

F11BFF – 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

F11BFF is the third in a suite of three routines for the iterative solution of a real general (nonsymmetric) system of simultaneous linear equations (Golub and Van Loan [1]). F11BFF returns information about the computations during an iteration and/or after this has been completed. The first routine of the suite, F11BDF, is a set-up routine; the second routine, F11BEF is the iterative solver itself.

These three routines are suitable for the solution of large sparse general (nonsymmetric) systems of equations.

2 Specification

```
 SUBROUTINE F11BFF(ITN, STPLHS, STPRHS, ANORM, SIGMAX, WORK,
1                  LWORK, IFAIL)
 INTEGER           ITN, IFAIL, LWORK
 real              STPLHS, STPRHS, ANORM, SIGMAX
 real              WORK(LWORK)
```

3 Description

F11BFF returns information about the solution process. It can be called either during a monitoring step of F11BEF or after F11BEF has completed its tasks. Calling F11BFF at any other time will result in an error condition being raised.

For further information you should read the documentation for F11BDF and F11BEF.

4 References

[1] Golub G H and Van Loan C F (1996) *Matrix Computations* Johns Hopkins University Press (3rd Edition), Baltimore

5 Parameters

1: ITN — INTEGER *Output*

On exit: the number of iterations carried out by F11BEF.

2: STPLHS — ***real*** *Output*

On exit: the current value of the left-hand side of the termination criterion used by F11BEF.

3: STPRHS — ***real*** *Output*

On exit: the current value of the right-hand side of the termination criterion used by F11BEF.

4: ANORM — ***real*** *Output*

On exit: the norm $\|A\|_1$ or $\|A\|_\infty$ when it is used by the termination criterion in F11BEF, either when it has been supplied to F11BDF or it has been estimated by F11BEF (see also Sections 3 and 5 of the document for F11BDF); 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 F11BEF, either when it has been supplied to F11BDF or it has been estimated by F11BEF (see also Sections 3 and 5 of the document for F11BDF); otherwise, SIGMAX = 0.0 is returned.

6: WORK (LWORK) — ***real*** array *Input*

On entry: the workspace WORK as returned by F11BEF (see also Sections 3 and 5 of the document for F11BEF).

7: LWORK — INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F11BFF was called (see also Section 5 of the document for F11BDF).

Constraint: LWORK $\geq$ 100.

Note: although the minimum value of LWORK ensures the correct functioning of F11BFF, a larger value is required by the iterative solver F11BEF (see also Section 5 of the document for F11BDF).

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 = $-i$

On entry, the ith argument had an illegal value.

IFAIL = 1

F11BFF has been called out of sequence. For example, the last call to F11BEF did not return IREVCM = 3 or 4.

7 Accuracy

Not applicable.

8 Further Comments

None.

9 Example

See the example for F11BDF.

F11BRF – 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

F11BRF is a set-up routine, the first in a suite of three routines for the iterative solution of a complex general (non-Hermitian) system of simultaneous linear equations. F11BRF must be called before F11BSF, the iterative solver. The third routine in the suite, F11BTF, can be used to return additional information about the computation.

These three routines are suitable for the solution of large sparse general (non-Hermitian) systems of equations.

2 Specification

```
      SUBROUTINE F11BRF(METHOD, PRECON, NORM, WEIGHT, ITERM, N, M, TOL,
     1                  MAXITN, ANORM, SIGMAX, MONIT, LWREQ, WORK,
     2                  LWORK, IFAIL)
      INTEGER           ITERM, N, M, MAXITN, MONIT, LWREQ, LWORK, IFAIL
      real              TOL, ANORM, SIGMAX
      CHARACTER*1       PRECON, NORM, WEIGHT
      CHARACTER*(*)     METHOD
      complex           WORK(LWORK)
```

3 Description

The suite consisting of the routines F11BRF, F11BSF, F11BTF is designed to solve the general (non-Hermitian) system of simultaneous linear equations $Ax = b$ of order n, where n is large and the coefficient matrix A is sparse.

F11BRF is a set-up routine which must be called before F11BSF, the iterative solver. The third routine in the suite, F11BTF, 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 (ℓ) method (Bi-CGSTAB (ℓ));

transpose-free quasi-minimal residual method (TFQMR).

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 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 $A^H\hat{x} = \hat{b}$, 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 $\hat{r}_i = P_i(A^H)\hat{r}_0$ are generated, where P_i is a polynomial of order i, and bi-orthogonality is exploited by computing the vector product $\rho_i = (\hat{r}_i, r_i) = (P_i(A^H)\hat{r}_0, P_i(A)r_0) = (\hat{r}_0, P_i^2(A)r_0)$. 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 (ℓ) Method (Bi-CGSTAB (ℓ))

The bi-conjugate gradient stabilized (ℓ) method (Bi-CGSTAB (ℓ)) (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 ℓ 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 ℓ increases, numerical instabilities may arise. For this reason, a maximum value of $\ell = 10$ is imposed, but probably $\ell = 4$ is sufficient in most cases.

3.4 Transpose-Free Quasi-Minimal Residual Method (TFQMR)

The transpose-free quasi-minimal residual method (TFQMR) (Freund and Nachtigal [10], Freund [11]) is conceptually derived from the CGS method. The residual is minimised over the space of the residual vectors generated by the CGS iterations under the simplifying assumption that residuals are almost orthogonal. In practice, this is not the case but theoretical analysis has proved the validity of the method. This has the effect of remedying the rather irregular convergence behaviour with wild oscillations in the residual norm that can degrade the numerical performance and robustness of the CGS method. In general, the TFQMR method can be expected to converge at least as fast as the CGS method, in terms of number of iterations, although each iteration involves a higher operation count. When the CGS method exhibits irregular convergence, the TFQMR method can produce much smoother, almost monotonic convergence curves. However, the close relationship between the CGS and TFQMR method implies that the *overall* speed of convergence is similar for both methods. In some cases, the TFQMR method may converge faster than the CGS method.

3.5 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 (ℓ) methods above, the user's program must provide the **maximum** number of basis vectors used, m or ℓ, 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 the 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, CGS and TFQMR 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 (ℓ) 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 $v = A^H u$ (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, i.e., 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), $p = 1$, ∞ or 2 and τ denotes a user-specified tolerance subject to $\max(10,\sqrt{n})$, $\epsilon \leq \tau < 1$, where ϵ 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 available for all methods except TFQMR. 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 (ℓ) 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 (ℓ) 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 F11BRF must be called, followed by the solver F11BSF. F11BTF can be called either when F11BSF is carrying out a monitoring step or after F11BSF 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 (ℓ) seems robust and reliable, but it can be slower than the other methods: if a preconditioner is used and $\ell > 1$, Bi-CGSTAB (ℓ) 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. TFQMR can be viewed as a more robust variant of the CGS method: it shares the

CGS method speed but avoids the CGS fluctuations in the residual, which may give rise to instability. Also, when the termination criterion (2) is used, the CGS, Bi-CGSTAB (ℓ) and TFQMR 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 (1996) *Matrix Computations* Johns Hopkins University Press (3rd 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(ℓ) 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

[10] Freund R W and Nachtigal N (1991) QMR: a Quasi-Minimal Residual Method for Non-Hermitian Linear Systems *Numer. Math.* **60** 315–339

[11] Freund R W (1993) A Transpose-Free Quasi-Mimimal Residual Algorithm for Non-Hermitian Linear Systems *SIAM J. Sci. Comput.* **14** 470–482

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 (ℓ) method;
'TFQMR'	transpose-free quasi-minimal residual method.

Constraint: METHOD = 'RGMRES', 'CGS', 'BICGSTAB' or 'TFQMR'.

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 inthe termination criteria. The possible choices are:

'1' l_1 norm;
'I' l_∞ norm;
'2' 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 F11BSF.
'N' all weights are implicitly set equal to one. Weights do not need to be supplied on initial entry to F11BSF.

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: if METHOD = 'RGMRES', 'CGS' or 'BICGSTAB', ITERM = 1 or 2;

if METHOD = 'TFQMR', ITERM = 1.

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 ℓ 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 τ for the termination criterion. If TOL ≤ 0.0, $\tau = \max(\sqrt{\epsilon},\ \sqrt{n}\,\epsilon)$ is used, where ϵ is the ***machine precision***. Otherwise $\tau = \max(\mathrm{TOL},\ 10\epsilon,\ \sqrt{n}\,\epsilon)$ 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 F11BSF.

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 σ_1 of the preconditioned iteration matrix; otherwise, SIGMAX is not referenced.

If SIGMAX ≤ 0.0, ITERM = 2 and METHOD = 'RGMRES', then the value of σ_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 F11BSF: if METHOD = 'CGS' or 'TFQMR', a monitoring step is executed every MONIT iterations; otherwise, a monitoring step is executed every MONIT super-iterations (groups of up to m or ℓ iterations for RGMRES or Bi-CGSTAB (ℓ), respectively).

There are some additional computational costs involved in monitoring the solution and residual vectors when the Bi-CGSTAB (ℓ) method is used with $\ell > 1$.

Constraint: MONIT $\leq$ MAXITN.

13: LWREQ — INTEGER *Output*

On exit: the minimum amount of workspace required by F11BSF. (See also Section 5 of the document for F11BSF.)

14: WORK (LWORK) — ***complex*** array *Output*

On exit: the workspace WORK is initialised by F11BRF. It must **not** be modified before calling the next routine in the suite, namely F11BSF.

15: LWORK — INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F11BRF was called.

Constraint: LWORK ≥ 120.

Note: although the minimum value of LWORK ensures the correct functioning of F11BRF, a larger value is required by the other routines in the suite, namely F11BSF and F11BTF. The required value is as follows:

Method	**Requirements**
RGMRES	LWORK $= 120 + n(m+3) + m(m+5) + 1$, where m is the dimension of the basis;
CGS	LWORK $= 120 + 7n$;
Bi-CGSTAB (ℓ)	LWORK $= 120 + (2n+\ell)(\ell+2) + p$, where ℓ is the order of the method;
TFQMR	LWORK $= 120 + 10n$;

where

$p = 2n$ if $\ell > 1$ and ITERM $= 2$ was supplied;
$p = n$ if $\ell > 1$ and a preconditioner is used or ITERM $= 2$ was supplied;
$p = 0$ otherwise.

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 ith argument had an illegal value.

IFAIL $= 1$

F11BRF has been called out of sequence.

7 Accuracy

Not applicable.

8 Further Comments

RGMRES can estimate internally the maximum singular value σ_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 (ℓ) 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 CGS, Bi-CGSTAB (ℓ) and TFQMR methods will restart automatically the computation from the last available iterates, when the stability of the solution process requires it.

Termination criterion (3), when available, 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 8×8 nonsymmetric system of simultaneous linear equations using the bi-conjugate gradient stabilized method of order $\ell = 2$, where the coefficients matrix A has a 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.

```
*     F11BRF Example Program Text
*     Mark 19 Release. NAG Copyright 1999.
*     .. 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 ..
      complex          A(LA), B(NMAX), WORK(LWORK), X(NMAX)
      real             WGT(NMAX)
      INTEGER          ICOL(LA), IDIAG(NMAX), IPIVP(NMAX), IPIVQ(NMAX),
     +                 IROW(LA), ISTR(NMAX+1), IWORK(LIWORK)
*     .. External Subroutines ..
      EXTERNAL         F11BRF, F11BSF, F11BTF, F11DNF, F11DPF, F11XNF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11BRF 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
      CALL F11DNF(N,NNZ,A,LA,IROW,ICOL,LFILL,DTOL,PSTRAT,MILU,IPIVP,
     +            IPIVQ,ISTR,IDIAG,NNZC,NPIVM,IWORK,LIWORK,IFAIL)
*
*     Call F11BRF to initialize the solver
*
      IFAIL = 0
      CALL F11BRF(METHOD,PRECON,NORM,WEIGHT,ITERM,N,M,TOL,MAXITN,
     +            ANORM,SIGMAX,MONIT,LWREQ,WORK,LWORK,IFAIL)
*
*     Call repeatedly F11BSF 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 F11BSF(IREVCM,X,B,WGT,WORK,LWREQ,IFAIL)
      IF (IREVCM.EQ.-1) THEN
         IFAILX = -1
         CALL F11XNF('Transpose',N,NNZ,A,IROW,ICOL,'No checking',X,B,
     +               IFAILX)
      ELSE IF (IREVCM.EQ.1) THEN
         IFAILX = -1
         CALL F11XNF('No transpose',N,NNZ,A,IROW,ICOL,'No checking',
     +               X,B,IFAILX)
      ELSE IF (IREVCM.EQ.2) THEN
         IFAIL1 = -1
         CALL F11DPF('No transpose',N,A,LA,IROW,ICOL,IPIVP,IPIVQ,
     +               ISTR,IDIAG,'No checking',X,B,IFAILX)
      ELSE IF (IREVCM.EQ.3) THEN
         IFAIL1 = 0
         CALL F11BTF(ITN,STPLHS,STPRHS,ANORM,SIGMAX,WORK,LWREQ,
     +               IFAIL1)
         WRITE (NOUT,99999) ITN, STPLHS
         WRITE (NOUT,99998)
         WRITE (NOUT,99996) (X(I),I=1,N)
         WRITE (NOUT,99997)
         WRITE (NOUT,99996) (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 F11BTF(ITN,STPLHS,STPRHS,ANORM,SIGMAX,WORK,LWREQ,IFAIL1)
*
*           Print the output data
*
            WRITE (NOUT,99995)
            WRITE (NOUT,99994) 'Number of iterations for convergence:    ',
     +        ITN
            WRITE (NOUT,99993) 'Residual norm:                           ',
     +        STPLHS
            WRITE (NOUT,99993) 'Right-hand side of termination criterion:',
     +        STPRHS
            WRITE (NOUT,99993) '1-norm of matrix A:                      ',
     +        ANORM
*
*           Output x
*
            WRITE (NOUT,99998)
            WRITE (NOUT,99996) (X(I),I=1,N)
            WRITE (NOUT,99997)
            WRITE (NOUT,99996) (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')
99997 FORMAT (/2X,'  Residual vector')
99996 FORMAT (1X,'(',1P,e16.4,',',1P,e16.4,')')
99995 FORMAT (/1X,'Final Results')
99994 FORMAT (1X,A,I4)
99993 FORMAT (1X,A,1P,e14.4)
      END
```

9.2 Example Data

```
F11BRF Example Program Data
  8                         N
  'TFQMR'                   METHOD
  'P'  '1'  'N'  1          PRECON, NORM, WEIGHT, ITERM
  1  1.0E-8   20            M, TOL, MAXITN
  2                         MONIT
  0  0.0                    LFILL, DTOL
  'N'  'C'                  MILU, PSTRAT
  24                        NNZ
 ( 2., 1.)   1   1
 (-1., 1.)   1   4
 ( 1.,-3.)   1   8
 ( 4., 7.)   2   1
 (-3., 0.)   2   2
 ( 2., 4.)   2   5
 (-7.,-5.)   3   3
 ( 2., 1.)   3   6
 ( 3., 2.)   4   1
 (-4., 2.)   4   3
 ( 0., 1.)   4   4
 ( 5.,-3.)   4   7
 (-1., 2.)   5   2
```

```
( 8., 6.)   5   5
(-3.,-4.)   5   7
(-6.,-2.)   6   1
( 5.,-2.)   6   3
( 2., 0.)   6   6
( 0.,-5.)   7   3
(-1., 5.)   7   5
( 6., 2.)   7   7
(-1., 4.)   8   2
( 2., 0.)   8   6
( 3., 3.)   8   8      A(I), IROW(I), ICOL(I), I=1,...,NNZ
(  7., 11.)
(  1., 24.)
(-13.,-18.)
(-10.,  3.)
( 23., 14.)
( 17., -7.)
( 15., -3.)
( -3., 20.)            B(I), I=1,...,N
( 0., 0.)
( 0., 0.)
( 0., 0.)
( 0., 0.)
( 0., 0.)
( 0., 0.)
( 0., 0.)
( 0., 0.)              X(I), I=1,...,N
```

9.3 Example Results

```
F11BRF Example Program Results

Monitoring at iteration no.   2
residual norm:     8.2345E+01

   Solution vector
(     6.9055E-01,      1.4236E+00)
(     7.3931E-02,     -1.1880E+00)
(     1.4778E+00,      4.7846E-01)
(     5.6572E+00,     -3.0786E+00)
(     1.4243E+00,     -1.1246E+00)
(     1.0374E-01,      1.9740E+00)
(     4.4985E-01,     -1.2715E+00)
(     2.5704E+00,      1.7578E+00)

   Residual vector
(     1.7772E+00,      4.6797E+00)
(     1.0774E+00,      6.4600E+00)
(    -3.2812E+00,     -1.1314E+01)
(    -3.8698E+00,     -1.6438E+00)
(     8.9912E+00,      1.1100E+01)
(     9.7428E+00,     -4.6218E-01)
(     3.1668E+00,      2.8721E+00)
(    -1.0323E+01,      1.5837E+00)
```

```
 Final Results
 Number of iterations for convergence:      4
 Residual norm:                              1.2260E-11
 Right-hand side of termination criterion:   8.9100E-06
 1-norm of matrix A:                         2.7000E+01

    Solution vector
 (      1.0000E+00,       1.0000E+00)
 (      2.0000E+00,      -1.0000E+00)
 (      3.0000E+00,       1.0000E+00)
 (      4.0000E+00,      -1.0000E+00)
 (      3.0000E+00,      -1.0000E+00)
 (      2.0000E+00,       1.0000E+00)
 (      1.0000E+00,      -1.0000E+00)
 (     -1.0283E-14,       3.0000E+00)

    Residual vector
 (      3.2152E-13,      -7.7094E-13)
 (      2.2187E-12,      -5.4712E-13)
 (      4.9205E-13,       9.8055E-13)
 (      1.0552E-12,       2.2737E-13)
 (     -1.0090E-12,      -9.2015E-13)
 (     -9.0594E-13,       1.7586E-13)
 (     -2.5935E-13,       1.1902E-13)
 (      4.4587E-13,       1.8119E-12)
```

F11BSF – 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

F11BSF is an iterative solver for a complex general (non-Hermitian) system of simultaneous linear equations; F11BSF is the second in a suite of three routines, where the first routine, F11BRF, must be called prior to F11BSF to set-up the suite, and the third routine in the suite, F11BTF, can be used to return additional information about the computation.

These three routines are suitable for the solution of large sparse general (non-Hermitian) systems of equations.

2 Specification

```
SUBROUTINE F11BSF(IREVCM, U, V, WGT, WORK, LWORK, IFAIL)
INTEGER           IREVCM, LWORK, IFAIL
real              WGT(*)
complex           U(*), V(*), WORK(LWORK)
```

3 Description

F11BSF solves the general (non-Hermitian) system of linear simultaneous equations $Ax = b$ using one of four available methods: RGMRES, the preconditioned restarted generalized minimum residual method (Saad and Schultz [1]); CGS, the preconditioned conjugate gradient squared method (Sonneveld [2]); Bi-CGSTAB (ℓ), the bi-conjugate gradient stabilized method of order ℓ (Van der Vorst [3], Sleijpen and Fokkema [4]); or TFQMR, the quasi-minimal transpose-free quasi-minimum residual method (Freund and Nachtigal [6], Freund [7]).

For a general description of the methods employed you are referred to Section 3 of the document for F11BRF.

F11BSF can solve the system after the first routine in the suite, F11BRF, has been called to initialize the computation and specify the method of solution. The third routine in the suite, F11BTF, can be used to return additional information generated by the computation during monitoring steps and after F11BSF has completed its tasks.

F11BSF 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 $v = A^H u$ (the four 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(ℓ) 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

[6] Freund R W and Nachtigal N (1991) QMR: a Quasi-Minimal Residual Method for Non-Hermitian Linear Systems *Numer. Math.* **60** 315–339

[7] Freund R W (1993) A Transpose-Free Quasi-Mimimal Residual Algorithm for Non-Hermitian Linear Systems *SIAM J. Sci. Comput.* **14** 470–482

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. F11BSF will then return with the termination code IREVCM = 4. Note that before calling F11BSF with IREVCM = 5 the calling program must have performed the tasks required by the value of IREVCM returned by the previous call to F11BSF, otherwise subsequently returned values may be invalid.

6 Immediate termination: F11BSF 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. F11BSF 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 $v = A^H u$, where u and v are stored in U and V, respectively; RGMRES, CGS and Bi-CGSTAB (ℓ) 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 F11BRF.

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. F11BTF can be called at this step to return additional information.

On final exit: IREVCM = 4: F11BSF 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(∗) — ***complex*** 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 F11BSF. If IFAIL = 1, the array U is unchanged from the last entry to F11BSF. Otherwise, U holds the last available iterate of the solution of the system of equations, for all returned values of IFAIL.

3: V(∗) — ***complex*** 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 F11BSF;

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 F11BSF. If IFAIL = 1, the array V is unchanged from the last entry to F11BSF. 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: WGT(∗) — ***real*** array *Input*

Note: the dimension of the array WGT must be at least max$(1, n)$ if weights are used, 1 otherwise.

On entry: the user-supplied weights, if these are to be used in the computation of the vector norms in the termination criterion (see Sections 3 and 5 of the document for F11BRF).

5: WORK (LWORK) — ***complex*** array *Input/Output*

On initial entry: the workspace WORK as returned by F11BRF (see also Section 5 of the document for F11BRF).

On intermediate re-entry: WORK must remain unchanged.

6: LWORK — INTEGER *Input*

On initial entry: the dimension of the array WORK as declared in the (sub)program from which F11BSF was called (see also Sections 3 and 5 of the document for F11BRF). The required amount of workspace is as follows:

Method	**Requirements**
RGMRES	LWORK $= 120 + n(m + 3) + m(m + 5) + 1$, where m is the dimension of the basis;
CGS	LWORK $= 120 + 7n$;
Bi-CGSTAB (ℓ)	LWORK $= 120 + (2n + \ell)(\ell + 2) + p$, where ℓ is the order of the method;
TFQMR	LWORK $= 120 + 10n$;

where

$p = 2n$	if $\ell > 1$ and ITERM = 2 was supplied;
$p = n$	if $\ell > 1$ and a preconditioner is used or ITERM = 2 was supplied;
$p = 0$	otherwise.

Constraint: LWORK $\geq$ LWREQ, where LWREQ is returned by F11BRF.

7: 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 ith argument had an illegal value.

IFAIL = 1

F11BSF 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 F11BTF have remained unchanged.

IFAIL = 2

The required accuracy could not be obtained. However, F11BSF 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 F11BSF 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 F11BTF to check the values of the left-and right-hand side of the termination condition.

IFAIL = 3

F11BRF was either not called before calling F11BSF 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 and TFQMR methods 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 kth 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 F11BRF. The computed values of the left- and right-hand sides of the termination criterion selected can be obtained by a call to F11BTF.

8 Further Comments

The number of operations carried out by F11BSF 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 F11BSF 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, CGS and TFQMR methods) or $\bar{A} = AM^{-1}$ (Bi-CGSTAB (ℓ) method).

Additional matrix-vector products are required for the computation of $\|A\|_1$ or $\|A\|_\infty$, when this has not been supplied to F11BRF 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 F11BRF) 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: F11BSF 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 F11BRF.

F11BTF – 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

F11BTF is the third in a suite of three routines for the iterative solution of a complex general (non-Hermitian) system of simultaneous linear equations (Golub and Van Loan [1]). F11BTF returns information about the computations during an iteration and/or after this has been completed. The first routine of the suite, F11BRF, is a set-up routine; the second routine, F11BSF is the iterative solver itself.

These three routines are suitable for the solution of large sparse general (non-Hermitian) systems of equations.

2 Specification

```
      SUBROUTINE F11BTF(ITN, STPLHS, STPRHS, ANORM, SIGMAX, WORK,
     1                  LWORK, IFAIL)
      INTEGER           ITN, IFAIL, LWORK
      real              STPLHS, STPRHS, ANORM, SIGMAX
      complex           WORK(LWORK)
```

3 Description

F11BTF returns information about the solution process. It can be called either during a monitoring step of F11BSF or after F11BSF has completed its tasks. Calling F11BTF at any other time will result in an error condition being raised.

For further information you should read the documentation for F11BRF and F11BSF.

4 References

[1] Golub G H and Van Loan C F (1996) *Matrix Computations* Johns Hopkins University Press (3rd Edition), Baltimore

5 Parameters

1: ITN — INTEGER *Output*

On exit: the number of iterations carried out by F11BSF.

2: STPLHS — ***real*** *Output*

On exit: the current value of the left-hand side of the termination criterion used by F11BSF.

3: STPRHS — ***real*** *Output*

On exit: the current value of the right-hand side of the termination criterion used by F11BSF.

4: ANORM — ***real*** *Output*

On exit: the norm $\|A\|_1$ or $\|A\|_\infty$ when it is used by the termination criterion in F11BSF, either when it has been supplied to F11BRF or it has been estimated by F11BSF (see also Sections 3 and 5 of the document for F11BRF); 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 F11BSF, either when it has been supplied to F11BRF or it has been estimated by F11BSF (see also Sections 3 and 5 of the document for F11BRF); otherwise, SIGMAX = 0.0 is returned.

6: WORK (LWORK) — ***complex*** array *Input*

On entry: the workspace WORK as returned by F11BSF (see also Sections 3 and 5 of the document for F11BSF).

7: LWORK — INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F11BTF was called (see also Section 5 of the document for F11BRF).

Constraint: LWORK $\geq$ 120.

Note: although the minimum value of LWORK ensures the correct functioning of F11BTF, a larger value is required by the iterative solver F11BSF (see also Section 5 of the document for F11BRF).

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 = $-i$

On entry, the ith argument had an illegal value.

IFAIL = 1

F11BTF has been called out of sequence. For example, the last call to F11BSF did not return IREVCM = 3 or 4.

7 Accuracy

Not applicable.

8 Further Comments

None.

9 Example

See the example for F11BRF.

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: $\mathrm{N} \geq 1$.

2: NNZ — INTEGER *Input*

On entry: the number of non-zero elements in the matrix A.

Constraint: $1 \leq \mathrm{NNZ} \leq \mathrm{N}^2$.

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: $\mathrm{LA} \geq 2 \times \mathrm{NNZ}$.

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.

Constraint: IROW and ICOL must satisfy the following constraints (which may be imposed by a call to F11ZAF):

$1 \leq \mathrm{IROW}(i) \leq \mathrm{N}$ and $1 \leq \mathrm{ICOL}(i) \leq \mathrm{N}$, 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 $\mathrm{LFILL} < 0$ then DTOL is used as a drop tolerance to control the fill-in (see Section 8.2); otherwise DTOL is not referenced.

Constraint: $\mathrm{DTOL} \geq 0.0$ if $\mathrm{LFILL} < 0$.

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 $> \mathrm{N}^2$,
or LA $< 2 \times$ NNZ,
or LFILL < 0 and DTOL < 0.0,
or PSTRAT $\neq$ 'N', 'U', 'P', or 'C',
or MILU $\neq$ 'M' or 'N',
or LIWORK $< 7 \times \mathrm{N} + 2$.

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 $i = 1,2,\ldots,\mathrm{NNZ}$.
$\mathrm{IROW}(i-1) < \mathrm{IROW}(i)$, 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 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 ≥ 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 α 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 ≤ 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 ϵ 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), stabilized bi-conjugate gradient (Bi-CGSTAB), or transpose-free quasi-minimal residual (TFQMR) 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], Bi-CGSTAB(ℓ) [5], [3], or TFQMR [6], [7] method.

F11DCF uses the incomplete *LU* factorization determined by F11DAF as the preconditioning matrix. A call to F11DCF must always be preceded 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 F11BDF, F11BEF and F11BFF. 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(ℓ) 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

[6] Freund R W and Nachtigal N (1991) QMR: a Quasi-Minimal Residual Method for Non-Hermitian Linear Systems *Numer. Math.* **60** 315–339

[7] Freund R W (1993) A Transpose-Free Quasi-Mimimal Residual Algorithm for Non-Hermitian Linear Systems *SIAM J. Sci. Comput.* **14** 470–482

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 (ℓ) method;
'TFQMR'	transpose-free quasi-minimal residual method.

Constraint: METHOD = 'RGMRES', 'CGS', 'BICGSTAB' or 'TFQMR'.

2: N — INTEGER *Input*

On entry: n, the order of the matrix A. This **must** be the same value as was supplied in the preceding call to F11DAF.

Constraint: 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 preceding 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 preceding call to F11DAF.

Constraint: LA $\geq 2 \times$ 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 ℓ of the polynomial Bi-CGSTAB method; otherwise, M is not referenced.

Constraints:

if METHOD = 'RGMRES', $0 < \text{M} \leq \min(\text{N},50)$;
if METHOD = 'BICGSTAB', $0 < \text{M} \leq \min(\text{N},10)$.

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 ≤ 0.0, $\tau = \max(\sqrt{\epsilon}, \sqrt{n}\,\epsilon)$ is used, where ϵ 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 ≥ 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 $\geq 4 \times \text{N} + \text{M} \times (\text{M}+\text{N}+5) + 101$;
if METHOD = 'CGS' then LWORK $\geq 8 \times \text{N} + 100$;
if METHOD = 'BICGSTAB' then LWORK $\geq 2 \times \text{N} \times (\text{M}+3) + \text{M} \times (\text{M}+2) + 100$;
if METHOD = 'TFQMR' then LWORK $\geq 11 \times \text{N} + 100$.

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 ≠ 'RGMRES', 'CGS', 'BICGSTAB', or 'TFQMR',
or N < 1,
or NNZ < 1,
or NNZ > N^2,
or LA < 2 × 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 ≥ 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 preceded 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 preceded 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 F11BDF, F11BEF or F11BFF. 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 preceding 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 19 Revised. NAG Copyright 1999.
*     .. 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+5)+101,8*N+100,2*N*(M+3)+M*(M+2)+100,
     +           11*N+100)
         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,D16.3)') ' Final residual norm =', RNORM
         WRITE (NOUT,*)
*
*        Output x
*
         WRITE (NOUT,*) '          X'
         DO 40 I = 1, N
            WRITE (NOUT,'(1X,1P,D16.4)') 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 ω 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: $1 \le \mathrm{NNZ} \le \mathrm{N}^2$.

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):

> $1 \le \mathrm{IROW}(i) \le \mathrm{N}$, $1 \le \mathrm{ICOL}(i) \le \mathrm{N}$, 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 $i = 2, 3, \ldots, \mathrm{NNZ}$.

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 ω.

Constraint: $0.0 < \mathrm{OMEGA} < 2.0$.

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:

$1 \leq \mathrm{IROW}(i) \leq \mathrm{N}$ and $1 \leq \mathrm{ICOL}(i) \leq \mathrm{N}$, 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 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 ϵ 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 19 Revised. NAG Copyright 1999.
*     .. 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), WGT(NMAX),
     +                 WORK(LWORK), X(NMAX)
      INTEGER          ICOL(LA), IROW(LA), IWORK(LIWORK)
*     .. External Subroutines ..
      EXTERNAL         F11BDF, F11BEF, F11BFF, 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(N*(M+3)+M*(M+5)+101,7*N+100,(2*N+M)*(M+2)+N+100,
     +           10*N+100)
         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 F11BDF to initialize solver
*
         WEIGHT = 'N'
         MONIT = 0
         IFAIL = 0
         CALL F11BDF(METHOD,PRECON,NORM,WEIGHT,ITERM,N,M,TOL,MAXITN,
     +               ANORM,SIGMAX,MONIT,LWNEED,WORK,LWORK,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 F11BEF to solve the linear system
*
         IREVCM = 0
         CKXAF = 'C'
         CKDDF = 'C'
*
  100    CONTINUE
*
         CALL F11BEF(IREVCM,X,B,WGT,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 F11BFF(ITN,STPLHS,STPRHS,ANORM,SIGMAX,WORK,LWORK,IFAIL)
*
           WRITE (NOUT,'(A,I10,A)') ' Converged in', ITN, ' iterations'
           WRITE (NOUT,'(A,1P,D16.3)') ' Matrix norm =', ANORM
           WRITE (NOUT,'(A,1P,D16.3)') ' Final residual norm =', STPLHS
           WRITE (NOUT,*)
*
*          Output x
*
           WRITE (NOUT,*) '           X'
           DO 120 I = 1, N
              WRITE (NOUT,'(1X,1P,D16.4)') 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), stabilized bi-conjugate gradient (Bi-CGSTAB), or transpose-free quasi-minimal residual (TFQMR) 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], Bi-CGSTAB(ℓ) [4], [2], or TFQMR [6], [7] 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 F11BDF, F11BEF and F11BFF. 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(ℓ) 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

[6] Freund R W and Nachtigal N (1991) QMR: a Quasi-Minimal Residual Method for Non-Hermitian Linear Systems *Numer. Math.* **60** 315–339

[7] Freund R W (1993) A Transpose-Free Quasi-Mimimal Residual Algorithm for Non-Hermitian Linear Systems *SIAM J. Sci. Comput.* **14** 470–482

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 (ℓ) method;
'TFQMR'	transpose-free quasi-minimal residual method.

Constraint: METHOD = 'RGMRES', 'CGS', 'BICGSTAB', or 'TFQMR'.

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 $\leq$ NNZ $\leq$ 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 $\leq$ 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.

8: OMEGA — ***real*** *Input*

On entry: if PRECON = 'S', OMEGA is the relaxation parameter ω 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 ℓ 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).

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 ≤ 0.0, $\tau = \max(\sqrt{\epsilon}, \sqrt{n}\,\epsilon)$ is used, where ϵ 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 ≥ 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 ≥ 4 × N + M × (M+N+5) + ν + 101;
if METHOD = 'CGS' then LWORK ≥ 8 × N + ν + 100;
if METHOD = 'BICGSTAB' then LWORK ≥ 2 × N × (M+3) + M × (M+2) + ν + 100;
if METHOD = 'TFQMR' then LWORK ≥ 11 × N + ν + 100;

where ν = 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', 'BICGSTAB', or 'TFQMR',
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:

$1 \leq \text{IROW}(i) \leq \text{N}$ 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.

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 F11BDF, F11BEF or F11BFF. 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 19 Revised. NAG Copyright 1999.
*     .. 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, L, 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
*
         L = N
         IF (PRECON.EQ.'N' .OR. PRECON.EQ.'n') L = 0
         LWREQ = MAX(4*N+M*(M+N+5)+L+101,8*N+L+100,2*N*(M+3)+M*(M+2)
     +           +L+100,11*N+L+100)
         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,D16.3)') ' Final residual norm =', RNORM
         WRITE (NOUT,*)
*
*        Output x
*
         WRITE (NOUT,*) '           X'
         DO 40 I = 1, N
            WRITE (NOUT,'(1X,1P,D16.4)') 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
```

F11DNF – 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

F11DNF computes an incomplete LU factorization of a complex sparse non-Hermitian matrix, represented in coordinate storage format. This factorization may be used as a preconditioner in combination with F11DQF or F11BSF.

2 Specification

```
      SUBROUTINE F11DNF(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
      complex           A(LA)
      real              DTOL
      CHARACTER*1       PSTRAT, MILU
```

3 Description

This routine computes an incomplete LU factorization [1], [2] of a complex sparse non-Hermitian n by n matrix A. The factorization is intended primarily for use as a preconditioner with one of the iterative solvers F11DQF or F11BSF.

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

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) — ***complex*** 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 F11ZNF 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 F11DNF 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 F11ZNF):

$1 \leq \mathrm{IROW}(i) \leq \mathrm{N}$ and $1 \leq \mathrm{ICOL}(i) \leq \mathrm{N}$, 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 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 F11DNF 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 F11DNF 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 $LA < 2 \times NNZ$,
or LFILL < 0 and DTOL < 0.0,
or PSTRAT ≠ 'N', 'U', 'P', or 'C',
or MILU ≠ 'M' or 'N',
or $LIWORK < 7 \times N + 2$.

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 F11ZNF 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 F11ZNF. 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 F11DNF is used in combination with F11BSF, or F11DQF, 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 F11DNF is roughly proportional to $\mathrm{NNZC}^2/\mathrm{N}$.

8.2 Control of Fill-in

If LFILL ≥ 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 α is the maximum modulus 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. F11DNF 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. The recommended approach is to 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.

For certain classes of matrices (typically those arising from the discretisation of elliptic or parabolic partial differential equations) the convergence rate of the preconditioned iterative solver can sometimes be significantly improved by using an incomplete factorization which preserves the row-sums of the original matrix. In these cases try setting MILU = 'M'.

8.5 Direct Solution of Sparse Linear Systems

Although it is not their primary purpose F11DNF and F11DPF may be used together to obtain a **direct** solution to a non-singular sparse complex non-Hermitian linear system. To achieve this the call to F11DPF should be preceded by a **complete** LU factorization

$$A = PLDUQ = M.$$

A complete factorization is obtained from a call to F11DNF 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 F11DNF and F11DPF as a direct method is illustrated in Section 9 of the document for F11DPF.

9 Example

This example program reads in a complex sparse non-Hermitian matrix A and calls F11DNF 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 F11DNF 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.

```
*     F11DNF Example Program Text.
*     Mark 19 Release. NAG Copyright 1999.
*     .. 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 ..
      complex          A(LA)
      INTEGER          ICOL(LA), IDIAG(NMAX), IPIVP(NMAX), IPIVQ(NMAX),
     +                 IROW(LA), ISTR(NMAX+1), IWORK(LIWORK)
*     .. External Subroutines ..
      EXTERNAL         F11DNF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11DNF 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 F11DNF(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,'(I8,5X,''('',D16.4,'','',D16.4,'')'',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,'(I8,5X,''('',D16.4,'','',D16.4,'')'',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

```
F11DNF Example Program Data
 4                 N
 11                NNZ
 0 0.0             LFILL, DTOL
 'C'               PSTRAT
 'N'               MILU
( 1., 3.)  1   2
( 1., 0.)  1   3
(-1.,-2.)  2   1
( 2.,-2.)  2   3
( 2., 1.)  2   4
( 0., 5.)  3   1
(-2., 0.)  3   4
( 1., 1.)  4   1
(-2., 4.)  4   2
( 1.,-3.)  4   3
( 0., 7.)  4   4   A(I), IROW(I), ICOL(I), I=1,...,NNZ
```

9.3 Program Results

```
F11DNF Example Program Results

 Original Matrix
N     =   4
NNZ   =  11
      1     (      0.1000E+01,      0.3000E+01)       1       2
      2     (      0.1000E+01,      0.0000E+00)       1       3
      3     (     -0.1000E+01,     -0.2000E+01)       2       1
      4     (      0.2000E+01,     -0.2000E+01)       2       3
      5     (      0.2000E+01,      0.1000E+01)       2       4
      6     (      0.0000E+00,      0.5000E+01)       3       1
      7     (     -0.2000E+01,      0.0000E+00)       3       4
      8     (      0.1000E+01,      0.1000E+01)       4       1
      9     (     -0.2000E+01,      0.4000E+01)       4       2
     10     (      0.1000E+01,     -0.3000E+01)       4       3
     11     (      0.0000E+00,      0.7000E+01)       4       4

 Factorization
N     =   4
NNZ   =  11
NPIVM =   0
     12     (      0.1000E+00,     -0.3000E+00)       1       1
     13     (      0.1000E+00,     -0.3000E+00)       1       3
     14     (      0.0000E+00,     -0.2000E+00)       2       2
     15     (      0.0000E+00,      0.4000E+00)       2       4
     16     (     -0.4000E+00,      0.2000E+00)       3       2
     17     (      0.2500E+00,      0.2500E+00)       3       3
     18     (     -0.5000E-01,      0.6500E+00)       3       4
     19     (      0.1000E+01,      0.1000E+01)       4       1
     20     (      0.2000E+00,     -0.2000E+00)       4       2
     21     (      0.1000E+01,     -0.1000E+01)       4       3
     22     (     -0.4803E-01,     -0.1397E+00)       4       4

          I       IPIVP(I)  IPIVQ(I)
          1         1         2
     2      3      1
     3      2      3
     4      4      4
```

F11DPF – 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

F11DPF solves a system of complex linear equations involving the incomplete LU preconditioning matrix generated by F11DNF.

2 Specification

```
 SUBROUTINE F11DPF(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
 complex           A(LA), Y(N), X(N)
 CHARACTER*1       TRANS, CHECK
```

3 Description

This routine solves a system of complex 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 complex sparse matrix stored in coordinate storage (CS) format (see Section 2.1.1 of the Chapter Introduction), as generated by F11DNF.

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 F11DPF through the matrix

$$C = L + D^{-1} + U - 2I$$

which is an N by N sparse matrix, stored in CS format, as returned by F11DNF. The permutation matrices P and Q are returned from F11DNF via the arrays IPIVP and IPIVQ.

It is envisaged that a common use of F11DPF will be to carry out the preconditioning step required in the application of F11BSF to sparse complex linear systems. F11DPF is used for this purpose by the black-box routine F11DQF.

F11DPF may also be used in combination with F11DNF to solve a sparse system of complex linear equations directly (see Section 8.5 of the document for F11DNF). This use of F11DPF is illustrated 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 preceding call to F11DNF.

Constraint: N $\geq$ 1.

3: A(LA) — ***complex*** array *Input*

On entry: the values returned in the array A by a previous call to F11DNF.

4: LA — INTEGER *Input*

On entry: the dimension of the arrays A, IROW and ICOL as declared in the (sub)program from which F11DPF is called. This **must** be the same value as was supplied in the preceding call to F11DNF.

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) — ***complex*** array *Input*

On entry: the right-hand side vector y.

13: X(N) — ***complex*** 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 $\neq$ 'N' or 'T',
or CHECK $\neq$ '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 F11DPF has been preceded by a valid call to F11DNF 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 ϵ is the ***machine precision***. An equivalent result holds when TRANS = 'T'.

8 Further Comments

8.1 Timing

The time taken for a call to F11DPF is proportional to the value of NNZC returned from F11DNF.

8.2 Use of CHECK

It is expected that a common use of F11DPF will be to carry out the preconditioning step required in the application of F11BSF to sparse complex linear systems. In this situation F11DPF 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 complex sparse non-Hermitian matrix A and a vector y. It then calls F11DNF, with LFILL = −1 and DTOL = 0.0, to compute the **complete** LU decomposition

$$A = PLDUQ.$$

Finally it calls F11DPF 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.

```
*     F11DPF Example Program Text.
*     Mark 19 Release. NAG Copyright 1999.
*     .. 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 ..
      complex          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         F11DNF, F11DPF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11DPF 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 = 0.0e0
         PSTRAT = 'C'
         MILU = 'N'
         IFAIL = 0
*
         CALL F11DNF(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 F11DPF(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,'(1X''('',D16.4,'','',D16.4,'')'')') X(I)
   40       CONTINUE
*
         END IF
*
      END IF
      STOP
      END
```

9.2 Program Data

```
F11DPF Example Program Data
  4                     N
 11                     NNZ
 ( 1., 2.)   1    2
 ( 1., 3.)   1    3
 (-1.,-3.)   2    1
 ( 2., 0.)   2    3
 ( 0., 4.)   2    4
 ( 3., 4.)   3    1
 (-2., 0.)   3    4
 ( 1.,-1.)   4    1
 (-2.,-1.)   4    2
 ( 1., 0.)   4    3
 ( 1., 3.)   4    4     A(I), IROW(I), ICOL(I), I=1,...,NNZ
 (  5.0, 14.0)
 ( 21.0,  5.0)
 (-21.0, 18.0)
 ( 14.0,  4.0)          Y(I), I=1,...,N
```

9.3 Program Results

```
 F11DPF Example Program Results

 Solution of linear system
 (      0.1000E+01,      0.4000E+01)
 (      0.2000E+01,      0.3000E+01)
 (      0.3000E+01,     -0.2000E+01)
 (      0.4000E+01,     -0.1000E+01)
```

F11DQF – 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

F11DQF solves a complex sparse non-Hermitian system of linear equations, represented in coordinate storage format, using a restarted generalized minimal residual (RGMRES), conjugate gradient squared (CGS), stabilized bi-conjugate gradient (Bi-CGSTAB), or transpose-free quasi-minimal residual (TFQMR) method, with incomplete *LU* preconditioning.

2 Specification

```
      SUBROUTINE F11DQF(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
      complex           A(LA), B(N), X(N), WORK(LWORK)
      real              TOL, RNORM
      CHARACTER*(*)     METHOD
```

3 Description

This routine solves a complex sparse non-Hermitian linear system of equations:

$$Ax = b,$$

using a preconditioned RGMRES [1], CGS [3], Bi-CGSTAB(ℓ) [4], [2], or TFQMR [5], [6] method.

F11DQF uses the incomplete *LU* factorization determined by F11DNF as the preconditioning matrix. A call to F11DQF must always be preceded by a call to F11DNF. Alternative preconditioners for the same storage scheme are available by calling F11DSF.

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 F11DNF. The array A holds the non-zero entries in these matrices, while IROW and ICOL hold the corresponding row and column indices.

F11DQF is a black-box routine which calls F11BRF, F11BSF and F11BTF. 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(ℓ) 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] Freund R W and Nachtigal N (1991) QMR: a Quasi-Minimal Residual Method for Non-Hermitian Linear Systems *Numer. Math.* **60** 315–339

[6] Freund R W (1993) A Transpose-Free Quasi-Mimimal Residual Algorithm for Non-Hermitian Linear Systems *SIAM J. Sci. Comput.* **14** 470–482

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 (ℓ) method;
'TFQMR'	transpose-free quasi-minimal residual method.

Constraint: METHOD = 'RGMRES', 'CGS', 'BICGSTAB' or 'TFQMR'.

2: N — INTEGER *Input*

On entry: n, the order of the matrix A. This **must** be the same value as was supplied in the preceding call to F11DNF.

Constraint: 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 preceding call to F11DNF.

Constraint: $1 \leq \text{NNZ} \leq \text{N}^2$.

4: A(LA) — ***complex*** array *Input*

On entry: the values returned in array A by a previous call to F11DNF.

5: LA — INTEGER *Input*

On entry: the dimension of the arrays A, IROW and ICOL as declared in the (sub)program from which F11DQF is called. This **must** be the same value as was supplied in the preceding call to F11DNF.

Constraint: LA $\geq$ 2 × 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 F11DNF.

12: B(N) — ***complex*** 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 ℓ 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 ≤ 0.0, $\tau = \max(\sqrt{\epsilon}, \sqrt{n}\,\epsilon)$ is used, where ϵ is the ***machine precision***. Otherwise $\tau = \max(\mathrm{TOL}, 10\epsilon, \sqrt{n}\,\epsilon)$ is used.

Constraint: TOL < 1.0.

15: MAXITN — INTEGER *Input*

On entry: the maximum number of iterations allowed.

Constraint: MAXITN ≥ 1.

16: X(N) — ***complex*** 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) — ***complex*** array *Workspace*

20: LWORK — INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F11DQF is called.

Constraints:

if METHOD = 'RGMRES' then LWORK $\geq 4 \times \mathrm{N} + \mathrm{M} \times (\mathrm{M}+\mathrm{N}+5) + 121$;
if METHOD = 'CGS' then LWORK $\geq 8 \times \mathrm{N} + 120$;
if METHOD = 'BICGSTAB' then LWORK $\geq 2 \times \mathrm{N} \times (\mathrm{M}+3) + \mathrm{M} \times (\mathrm{M}+2) + 120$;
if METHOD = 'TFQMR' then LWORK $\geq 11 \times \mathrm{N} + 120$.

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 ≠ 'RGMRES', 'CGS', 'BICGSTAB', or 'TFQMR',
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 F11DQF has been preceded by a valid call to F11DNF, 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 F11DQF has been preceded by a valid call to F11DNF, 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

Algorithmic breakdown. A solution is returned, although it is possible that it is completely inaccurate.

IFAIL = 7

A serious error has occurred in an internal call to F11BRF, F11BSF or F11BTF. 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 F11DQF for each iteration is roughly proportional to the value of NNZC returned from the preceding call to F11DNF.

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 complex sparse non-Hermitian 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.

```
*     F11DQF Example Program Text.
*     Mark 19 Release. NAG Copyright 1999.
*     .. 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 ..
      complex          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         F11DNF, F11DQF
*     .. Intrinsic Functions ..
      INTRINSIC        MAX
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11DQF 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+5)+121,8*N+120,2*N*(M+3)+M*(M+2)+120,
     +           11*N+120)
*
         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 rhs 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 F11DNF(N,NNZ,A,LA,IROW,ICOL,LFILL,DTOL,PSTRAT,MILU,IPIVP,
     +               IPIVQ,ISTR,IDIAG,NNZC,NPIVM,IWORK,LIWORK,IFAIL)
*
*        Solve Ax = b using F11DQF
*
         CALL F11DQF(METHOD,N,NNZ,A,LA,IROW,ICOL,IPIVP,IPIVQ,ISTR,IDIAG,
     +               B,M,TOL,MAXITN,X,RNORM,ITN,WORK,LWORK,IFAIL)
*
         WRITE (NOUT,'(1X,A,I10,A)') 'Converged in', ITN, ' iterations'
         WRITE (NOUT,'(1X,A,1P,D16.3)') 'Final residual norm =', RNORM
         WRITE (NOUT,*)
*
*        Output x
*
         WRITE (NOUT,*) '                   X'
         DO 40 I = 1, N
            WRITE (NOUT,'(1X,''('',1P,D16.4,'','',1P,D16.4,'')'')') X(I)
   40    CONTINUE
      END IF
*
      STOP
      END
```

9.2 Program Data

```
F11DQF 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., 1.)  1  4
( 1.,-3.)  1  8
( 4., 7.)  2  1
(-3., 0.)  2  2
( 2., 4.)  2  5
(-7.,-5.)  3  3
( 2., 1.)  3  6
( 3., 2.)  4  1
(-4., 2.)  4  3
( 0., 1.)  4  4
( 5.,-3.)  4  7
(-1., 2.)  5  2
( 8., 6.)  5  5
(-3.,-4.)  5  7
(-6.,-2.)  6  1
( 5.,-2.)  6  3
( 2., 0.)  6  6
( 0.,-5.)  7  3
(-1., 5.)  7  5
( 6., 2.)  7  7
(-1., 4.)  8  2
( 2., 0.)  8  6
( 3., 3.)  8  8         A(I), IROW(I), ICOL(I), I=1,...,NNZ
(  7., 11.)
(  1., 24.)
(-13.,-18.)
(-10.,  3.)
( 23., 14.)
( 17., -7.)
( 15., -3.)
( -3., 20.)             B(I), I=1,...,N
( 0., 0.)
( 0., 0.)
( 0., 0.)
( 0., 0.)
( 0., 0.)
( 0., 0.)
( 0., 0.)
( 0., 0.)               X(I), I=1,...,N
```

9.3 Program Results

```
F11DQF Example Program Results

Converged in         4 iterations
Final residual norm =       1.325E-11

                   X
(      1.0000E+00,      1.0000E+00)
(      2.0000E+00,     -1.0000E+00)
(      3.0000E+00,      1.0000E+00)
(      4.0000E+00,     -1.0000E+00)
(      3.0000E+00,     -1.0000E+00)
(      2.0000E+00,      1.0000E+00)
(      1.0000E+00,     -1.0000E+00)
(     -1.7115E-12,      3.0000E+00)
```

F11DRF – 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

F11DRF solves a system of linear equations involving the preconditioning matrix corresponding to SSOR applied to a complex sparse non-Hermitian matrix, represented in coordinate storage format.

2 Specification

```
SUBROUTINE F11DRF(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
complex           A(NNZ), RDIAG(N), Y(N), X(N)
real              OMEGA
CHARACTER*1       TRANS, CHECK
```

3 Description

This routine solves a system of linear equations

$$Mx = y, \quad \text{or} \quad M^H x = y,$$

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 complex sparse non-Hermitian 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 ω is a user-defined relaxation parameter.

It is envisaged that a common use of F11DRF will be to carry out the preconditioning step required in the application of F11BSF to sparse linear systems. For an illustration of this use of F11DRF see the example program given in Section 9. F11DRF is also used for this purpose by the black-box routine F11DSF.

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 $M^H x = y$ 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) — ***complex*** 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 F11ZNF 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.

Constraint: IROW and ICOL must satisfy the following constraints (which may be imposed by a call to F11ZNF):

$1 \leq \text{IROW}(i) \leq \text{N}$ 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 $i = 2, 3, \ldots, \text{NNZ}$.

7: RDIAG(N) — ***complex*** 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 ω.

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) — ***complex*** array *Input*

On entry: the right-hand side vector y.

11: X(N) — ***complex*** 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:

$1 \leq \mathrm{IROW}(i) \leq \mathrm{N}$ 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 in the matrix A, is out of order, or has duplicate row and column indices. Call F11ZNF 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 ϵ is the ***machine precision***. An equivalent result holds when TRANS = 'T'.

8 Further Comments

8.1 Timing

The time taken for a call to F11DRF is proportional to NNZ.

8.2 Use of CHECK

It is expected that a common use of F11DRF will be to carry out the preconditioning step required in the application of F11BSF to sparse linear systems. In this situation F11DRF 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 complex sparse linear system of equations:

$$Ax = b,$$

using RGMRES with SSOR preconditioning.

The RGMRES algorithm itself is implemented by the reverse communication routine F11BSF, 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 F11XNF.
>
> If IREVCM = -1 a conjugate transposed matrix-vector product $v = A^H u$ is required in the estimation of the norm of A. This is implemented by a call to F11XNF.
>
> If IREVCM = 2 a solution of the preconditioning equation $Mv = u$ is required. This is achieved by a call to F11DRF.
>
> If IREVCM = 4 F11BSF has completed its tasks. Either the iteration has terminated, or an error condition has arisen.

For further details see the routine document for F11BSF.

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.

```
*     F11DRF Example Program Text.
*     Mark 19 Release. NAG Copyright 1999.
*     .. 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        CKDRF, CKXNF, NORM, PRECON, TRANS, WEIGHT
      CHARACTER*8      METHOD
*     .. Local Arrays ..
      complex          A(LA), B(NMAX), RDIAG(NMAX), WORK(LWORK), X(NMAX)
      real             WGT(NMAX)
      INTEGER          ICOL(LA), IROW(LA), IWORK(LIWORK)
*     .. External Subroutines ..
      EXTERNAL         F11BRF, F11BSF, F11BTF, F11DRF, F11XNF
*     .. Intrinsic Functions ..
      INTRINSIC        MAX
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11DRF 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(121+N*(3+M)+M*(M+5),120+7*N,120+(2*N+M)*(M+2)+2*N,
     +           120+10*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 rhs vector b and initial approximate solution x
*
         READ (NIN,*) (B(I),I=1,N)
         READ (NIN,*) (X(I),I=1,N)
*
*        Call F11BRF to initialize solver
*
         WEIGHT = 'N'
         MONIT = 0
         IFAIL = 0
         CALL F11BRF(METHOD,PRECON,NORM,WEIGHT,ITERM,N,M,TOL,MAXITN,
     +               ANORM,SIGMAX,MONIT,LWNEED,WORK,LWORK,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.0.0e0) 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
*
      END IF
*
*     Call F11BSF to solve the linear system
*
      IREVCM = 0
      CKXNF = 'C'
      CKDRF = 'C'
*
  100 CONTINUE
*
      CALL F11BSF(IREVCM,X,B,WGT,WORK,LWORK,IFAIL)
*
      IF (IREVCM.EQ.1) THEN
*
*        Compute matrix-vector product
*
         TRANS = 'Not transposed'
         CALL F11XNF(TRANS,N,NNZ,A,IROW,ICOL,CKXNF,X,B,IFAIL)
         CKXNF = 'N'
         GO TO 100
*
      ELSE IF (IREVCM.EQ.-1) THEN
*
*        Compute conjugate transposed matrix-vector product
*
         TRANS = 'Transposed'
         CALL F11XNF(TRANS,N,NNZ,A,IROW,ICOL,CKXNF,X,B,IFAIL)
         CKXNF = 'N'
         GO TO 100
*
      ELSE IF (IREVCM.EQ.2) THEN
*
*        SSOR preconditioning
*
         TRANS = 'Not transposed'
         CALL F11DRF(TRANS,N,NNZ,A,IROW,ICOL,RDIAG,OMEGA,CKDRF,X,B,
     +               IWORK,IFAIL)
         CKDRF = 'N'
         GO TO 100
*
      ELSE IF (IREVCM.EQ.4) THEN
*
*        Termination
*
         CALL F11BTF(ITN,STPLHS,STPRHS,ANORM,SIGMAX,WORK,LWORK,IFAIL)
*
         WRITE (NOUT,'(1X,A,I10,A)') 'Converged in', ITN,
     +     ' iterations'
         WRITE (NOUT,'(1X,A,1P,D16.3)') 'Matrix norm =', ANORM
```

```
            WRITE (NOUT,'(1X,A,1P,D16.3)') 'Final residual norm =',
     +        STPLHS
            WRITE (NOUT,*)
*
*           Output x
*
            WRITE (NOUT,*) '                   X'
            DO 120 I = 1, N
               WRITE (NOUT,'(1X,''('',1P,D16.4,'','',1P,D16.4,'')'')')
     +           X(I)
  120       CONTINUE
*
         END IF
*
  140    CONTINUE
*
      END IF
      STOP
      END
```

9.2 Program Data

```
F11DRF Example Program Data
  5                      N
 16                      NNZ
 'CGS'                   METHOD
 'P' 'I' 1               PRECON, NORM, ITERM
 2 1.E-10 1000           M, TOL, MAXITN
 0.E0 0.E0               ANORM, SIGMAX
 1.4E0                   OMEGA
 ( 2., 3.)   1    1
 ( 1.,-1.)   1    2
 (-1., 0.)   1    4
 ( 0., 2.)   2    2
 (-2., 1.)   2    3
 ( 1., 0.)   2    5
 ( 0.,-1.)   3    1
 ( 5., 4.)   3    3
 ( 3.,-1.)   3    4
 ( 1., 0.)   3    5
 (-2., 2.)   4    1
 (-3., 1.)   4    4
 ( 0., 3.)   4    5
 ( 4.,-2.)   5    2
 (-2., 0.)   5    3
 (-6., 1.)   5    5      A(I), IROW(I), ICOL(I), I=1,...,NNZ
 ( -3., 3.)
 (-11., 5.)
 ( 23.,48.)
 (-41., 2.)
 (-28.,-31.)             B(I), I=1,...,N
 ( 0., 0.)
 ( 0., 0.)
 ( 0., 0.)
 ( 0., 0.)
 ( 0., 0.)               X(I), I=1,...,N
```

9.3 Program Results

```
F11DRF Example Program Results

Converged in          5 iterations
Matrix norm =        1.500E+01
Final residual norm =        1.776E-14

                   X
(       1.0000E+00,       2.0000E+00)
(       2.0000E+00,       3.0000E+00)
(       3.0000E+00,       4.0000E+00)
(       4.0000E+00,       5.0000E+00)
(       5.0000E+00,       6.0000E+00)
```

F11DSF – 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

F11DSF solves a complex sparse non-Hermitian system of linear equations, represented in coordinate storage format, using a restarted generalized minimal residual (RGMRES), conjugate gradient squared (CGS), stabilized bi-conjugate gradient (Bi-CGSTAB), or transpose-free quasi-minimal residual (TFQMR) method, without preconditioning, with Jacobi, or with SSOR preconditioning.

2 Specification

```
      SUBROUTINE F11DSF(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
      complex           A(NNZ), B(N), X(N), WORK(LWORK)
      real              OMEGA, TOL, RNORM
      CHARACTER*(*)     METHOD
      CHARACTER*1       PRECON
```

3 Description

This routine solves a complex sparse non-Hermitian system of linear equations:

$$Ax = b,$$

using an RGMRES [1], CGS [3], Bi-CGSTAB(ℓ) [4], [2], or TFQMR [6], [7] method.

F11DSF 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 F11DQF.

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.

F11DSF is a black-box routine which calls F11BRF, F11BSF and F11BTF. 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(ℓ) 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

[6] Freund R W and Nachtigal N (1991) QMR: a Quasi-Minimal Residual Method for Non-Hermitian Linear Systems *Numer. Math.* **60** 315–339

[7] Freund R W (1993) A Transpose-Free Quasi-Mimimal Residual Algorithm for Non-Hermitian Linear Systems *SIAM J. Sci. Comput.* **14** 470–482

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 (ℓ) method;
'TFQMR'	transpose-free quasi-minimal residual method.

Constraint: METHOD = 'RGMRES', 'CGS', 'BICGSTAB' or 'TFQMR'.

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 (SSOR).

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: $1 \leq$ NNZ $\leq$ N^2.

5: A(NNZ) — ***complex*** 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 F11ZNF 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 F11ZNF):

$1 \leq$ 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.

8: OMEGA — ***real*** *Input*

On entry: if PRECON = 'S', OMEGA is the relaxation parameter ω 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) — ***complex*** 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 ℓ 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).

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 ≤ 0.0, $\tau = \max(\sqrt{\epsilon}, \sqrt{n}\,\epsilon)$ is used, where ϵ 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 ≥ 1.

13: X(N) — ***complex*** 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) — ***complex*** array *Workspace*

17: LWORK — INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F11DSF is called.

Constraint:

if METHOD = 'RGMRES', LWORK ≥ 4 × N + M × (M+N+5) + ν + 121;
if METHOD = 'CGS', LWORK ≥ 8 × N + ν + 120;
if METHOD = 'BICGSTAB', LWORK ≥ 2 × N × (M+3) + M × (M+2) + ν + 120;
if METHOD = 'TFQMR', LWORK ≥ 11 × N + ν + 120;

where ν = 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', 'BICGSTAB' or 'TFQMR',
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), when METHOD = 'RGMRES',
or M > min(N,10), when METHOD = 'BICGSTAB',
or TOL ≥ 1.0,
or MAXITN < 1,
or LWORK is too small.

IFAIL = 2

On entry, the arrays IROW and ICOL fail to satisfy the following constraints:

$1 \le \mathrm{IROW}(i) \le \mathrm{N}$ and $1 \le \mathrm{ICOL}(i) \le \mathrm{N}$, 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 within the matrix A, is out of order, or has duplicate row and column indices. Call F11ZNF to reorder and sum or remove duplicates.

IFAIL = 3

On entry, the matrix A has a zero diagonal element. Jacobi and SSOR preconditioners are therefore 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

Algorithmic breakdown. A solution is returned, although it is possible that it is completely inaccurate.

IFAIL = 7

A serious error has occurred in an internal call to F11BRF, F11BSF or F11BTF. 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 F11DSF for each iteration is roughly proportional to NNZ.

The number of iterations required to achieve a prescribed accuracy cannot easily be 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 complex sparse non-Hermitian system of equations using the CGS method, with no 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.

```
*     F11DSF Example Program Text.
*     Mark 19 Release. NAG Copyright 1999.
*     .. 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, L, LWREQ, M, MAXITN, N, NNZ
      CHARACTER        PRECON
      CHARACTER*8      METHOD
*     .. Local Arrays ..
      complex          A(LA), B(NMAX), WORK(LWORK), X(NMAX)
      INTEGER          ICOL(LA), IROW(LA), IWORK(2*NMAX+1)
*     .. External Subroutines ..
      EXTERNAL         F11DSF
*     .. Intrinsic Functions ..
      INTRINSIC        MAX
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11DSF 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
*
         L = N
         IF (PRECON.EQ.'N' .OR. PRECON.EQ.'n') L = 0
         LWREQ = MAX(4*N+M*(M+N+5)+L+121,8*N+L+120,2*N*(M+3)+M*(M+2)
     +           +L+120,11*N+L+120)
         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 rhs 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 F11DSF
*
         IFAIL = 0
         CALL F11DSF(METHOD,PRECON,N,NNZ,A,IROW,ICOL,OMEGA,B,M,TOL,
     +               MAXITN,X,RNORM,ITN,WORK,LWORK,IWORK,IFAIL)
*
         WRITE (NOUT,'(1X,A,I10,A)') 'Converged in', ITN, ' iterations'
         WRITE (NOUT,'(1X,A,1P,D16.3)') 'Final residual norm =', RNORM
         WRITE (NOUT,*)
*
*        Output x
*
         WRITE (NOUT,*) '                   X'
         DO 40 I = 1, N
            WRITE (NOUT,'(1X,''('',1P,D16.4,'','',1P,D16.4,'')'')') X(I)
   40    CONTINUE
*
      END IF
      STOP
      END
```

9.2 Program Data

```
F11DSF Example Program Data
 5                     N
16                     NNZ
 'CGS' 'N'             METHOD, PRECON
 1.05                  OMEGA
 1 1.E-10 1000         M, TOL, MAXITN
( 2., 3.)  1   1
( 1.,-1.)  1   2
(-1., 0.)  1   4
( 0., 2.)  2   2
(-2., 1.)  2   3
( 1., 0.)  2   5
( 0.,-1.)  3   1
( 5., 4.)  3   3
( 3.,-1.)  3   4
( 1., 0.)  3   5
(-2., 2.)  4   1
(-3., 1.)  4   4
( 0., 3.)  4   5
( 4.,-2.)  5   2
(-2., 0.)  5   3
(-6., 1.)  5   5      A(I), IROW(I), ICOL(I), I=1,...,NNZ
( -3., 3.)
(-11., 5.)
( 23.,48.)
(-41., 2.)
(-28.,-31.)           B(I), I=1,...,N
( 0., 0.)
( 0., 0.)
( 0., 0.)
( 0., 0.)
( 0., 0.)             X(I), I=1,...,N
```

9.3 Program Results

```
F11DSF Example Program Results

Converged in         5 iterations
Final residual norm =      1.052E-10

                   X
(      1.0000E+00,      2.0000E+00)
(      2.0000E+00,      3.0000E+00)
(      3.0000E+00,      4.0000E+00)
(      4.0000E+00,      5.0000E+00)
(      5.0000E+00,      6.0000E+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 $\bar{x} = E^T x$. 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 τ denotes a user-specified tolerance subject to $\max(10, \sqrt{n})\,\epsilon \leq \tau < 1$, where ϵ 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_∞ 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 τ for the termination criterion. If TOL ≤ 0.0, $\tau = \max(\sqrt{\epsilon}, \sqrt{n}\,\epsilon)$ is used, where ϵ is the ***machine precision***. Otherwise $\tau = \max(\mathrm{TOL}, 10\epsilon, \sqrt{n}\,\epsilon)$ 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, ϵ).
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 ≥ 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 ith 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 kth 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 ≤ 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):

$$1 \leq \mathrm{IROW}(i) \leq \mathrm{N} \text{ and } 1 \leq \mathrm{ICOL}(i) \leq \mathrm{IROW}(i), \text{ for } i = 1,2,\ldots,\mathrm{NNZ}.$$
$$\mathrm{IROW}(i-1) < \mathrm{IROW}(i), \text{ or}$$
$$\mathrm{IROW}(i-1) = \mathrm{IROW}(i) \text{ and } \mathrm{ICOL}(i-1) < \mathrm{ICOL}(i), \text{ for } i = 2,3,\ldots,\mathrm{NNZ}.$$

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 ≠ 'M' or 'N',
or PSTRAT ≠ '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, 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 $\mathrm{NNZC}^2/\mathrm{N}$.

8.2 Control of Fill-in

If LFILL ≥ 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:

$$|a_{ij}| < \mathrm{DTOL} \times \sqrt{|a_{ii}a_{jj}|}.$$

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

$$A = PLDL^TP^T = M.$$

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: 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 ϵ 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:

$$A = PLDL^TP^T.$$

Finally it calls F11JBF to solve the system

$$PLDL^TP^Tx = y.$$

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, 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 preceded 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 2.1.2 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 preceding 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 preceding 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 preceding 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 ϵ 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 + 120;
if METHOD = 'SYMMLQ' then LWORK $\geq 7 \times$ N + 120.

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 preceded 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 preceded 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 preceding 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 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.

```
*     F11JCF Example Program Text
*     Mark 19 Revised. NAG Copyright 1999.
*     .. 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+120)
```

```
*     .. 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 Program 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 Program 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 2.1.2 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 ω 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: N $\geq$ 1.

2: NNZ — INTEGER *Input*

On entry: the number of non-zero elements in the lower triangular part of A.

Constraint: 1 $\leq$ NNZ $\leq$ N $\times$ (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 $i = 1, 2, \ldots$,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.

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 ω.

Constraint: $0.0 \leq \mathrm{OMEGA} \leq 2.0$.

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 $i = 1, 2, \ldots,$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 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 ϵ 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 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.

```
*     F11JDF Example Program Text
*     Mark 19 Revised. NAG Copyright 1999.
*     .. 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+120)
*     .. 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), WGT(NMAX),
     +                 WORK(LWORK), X(NMAX)
      INTEGER          ICOL(LA), IROW(LA), IWORK(LIWORK)
*     .. External Subroutines ..
      EXTERNAL         F11GDF, F11GEF, F11GFF, 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 F11GDF to initialize solver
*
        WEIGHT = 'N'
        MONIT = 0
        IFAIL = 0
        CALL F11GDF(METHOD,PRECON,SIGCMP,NORM,WEIGHT,ITERM,N,TOL,
     +              MAXITN,ANORM,SIGMAX,SIGTOL,MAXITS,MONIT,LWNEED,
     +              WORK,LWORK,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 F11GEF to solve the linear system
*
        IREVCM = 0
        CKXEF = 'C'
        CKJDF = 'C'
*
 100    CONTINUE
*
        CALL F11GEF(IREVCM,X,B,WGT,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 F11GFF(ITN,STPLHS,STPRHS,ANORM,SIGMAX,ITS,SIGERR,WORK,
     +                  LWORK,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 Program 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 Program 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 2.1.2 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 (SSOR).

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 lower triangular part of the matrix A.

Constraint: 1 $\leq$ NNZ $\leq$ N $\times$ (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):

$1 \leq \text{IROW}(i) \leq \text{N}$ and $1 \leq \text{ICOL}(i) \leq \text{IROW}(i)$, 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.

8: OMEGA — ***real*** *Input*

On entry: if PRECON = 'S', OMEGA is the relaxation parameter ω to be used in the SSOR method. Otherwise OMEGA need not be initialized.

Constraint: 0.0 $\leq$ 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 ϵ 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 $\geq 6 \times$ N + ν + 120;
if METHOD = 'SYMMLQ' then LWORK $\geq 7 \times$ N + ν + 120;

where ν = 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:

$1 \le \mathrm{IROW}(i) \le \mathrm{N}$ and $1 \le \mathrm{ICOL}(i) \le \mathrm{IROW}(i)$, 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 $k = \mathrm{ITN}$, satisfies the termination criterion

$$\|r_k\|_\infty \le \tau \times (\|b\|_\infty + \|A\|_\infty \|x_k\|_\infty).$$

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 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.

```
*     F11JEF Example Program Text
*     Mark 19 Revised. NAG Copyright 1999.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LA, LWORK
      PARAMETER        (NMAX=1000,LA=10000,LWORK=7*NMAX+120)
*     .. 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 Program 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 Program 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
```

F11JNF – 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

F11JNF computes an incomplete Cholesky factorization of a complex sparse Hermitian matrix, represented in symmetric coordinate storage format. This factorization may be used as a preconditioner in combination with F11JQF.

2 Specification

```
  SUBROUTINE F11JNF(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
  complex           A(LA)
  real              DTOL, DSCALE
  CHARACTER*1       MIC, PSTRAT
```

3 Description

This routine computes an incomplete Cholesky factorization [3] of a complex sparse Hermitian 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 the complex Hermitian iterative solver F11JQF.

The decomposition is written in the form

$$A = M + R$$

where

$$M = PLDL^H P^T$$

and P is a permutation matrix, L is lower triangular complex with unit diagonal elements, D is real 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 2.1.2 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: 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 $\leq$ NNZ $\leq$ N $\times$ (N+1)/2.

3: A(LA) — ***complex*** 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 F11ZPF 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 F11JNF 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.

Constraints: IROW and ICOL must satisfy the following constraints (which may be imposed by a call to F11ZPF):

$1 \leq \mathrm{IROW}(i) \leq \mathrm{N}$ and $1 \leq \mathrm{ICOL}(i) \leq \mathrm{IROW}(i)$, 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.

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: 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 (1.0+DSCALE) at the start of the factorization. This can be used to ensure that the preconditioner is positive-definite. See also 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 [2];
if PSTRAT = 'U', then diagonal pivoting is carried out according to the user-defined input array 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 to be 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 F11JNF 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 F11JNF is called.

Constraints: the minimum permissible value of LIWORK depends on LFILL as follows:

LIWORK $\geq$ 2 × LA − 3 × NNZ + 7 × N + 1, if LFILL $\geq$ 0, and
LIWORK $\geq$ LA − NNZ + 7 × N + 1, otherwise.

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, N < 1,
or NNZ < 1,
or NNZ > N × (N+1)/2,
or LA < 2 × NNZ,
or DTOL < 0.0,
or MIC $\neq$ 'M' or 'N',
or PSTRAT $\neq$ 'N' 'M' or 'U',
or LIWORK is 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 F11ZPF 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,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 F11ZPF. 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 F11JNF is used in combination with F11JQF, 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 F11JNF is roughly proportional to NNZC^2/N.

8.2 Control of Fill-in

If LFILL ≥ 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:

$$|a_{ij}| < \mathrm{DTOL} \times \sqrt{|a_{ii}a_{jj}|}.$$

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 complex Hermitian 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.

F11JNF 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.

For certain classes of matrices (typically those arising from the discretisation of elliptic or parabolic partial differential equations), the convergence rate of the preconditioned iterative solver can sometimes be significantly improved by using an incomplete factorization which preserves the row-sums of the original matrix. In these cases try setting MIC = 'M'.

8.4 Direct Solution of Positive-Definite Systems

Although it is not their primary purpose F11JNF and F11JPF may be used together to obtain a **direct** solution to a complex Hermitian positive-definite linear system. To achieve this the call to F11JPF should be preceded by a **complete** Cholesky factorization

$$A = PLDL^{H}P^{T} = M.$$

A complete factorization is obtained from a call to F11JNF 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 F11JNF and F11JPF as a direct method is illustrated in Section 9 of the document for F11JPF.

9 Example

This example program reads in a complex sparse Hermitian matrix A and calls F11JNF 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 F11JNF 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 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.

```
*     F11JNF Example Program Text.
*     Mark 19 Release. NAG Copyright 1999.
*     .. 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 ..
      complex          A(LA)
      INTEGER          ICOL(LA), IPIV(NMAX), IROW(LA), ISTR(NMAX+1),
     +                 IWORK(LIWORK)
*     .. External Subroutines ..
      EXTERNAL         F11JNF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11JNF 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 F11JNF(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 (I8,5X,'(',e16.4,',',e16.4,')',2I8)
99998 FORMAT (1X,2I8)
      END
```

9.2 Program Data

```
F11JNF Example Program Data
  7                   N
 16                   NNZ
  0 0.0               LFILL, DTOL
  'N' 0.0             MIC, DSCALE
  'M'                 PSTRAT
  ( 6., 0.)   1    1
  ( 1.,-2.)   2    1
  ( 9., 0.)   2    2
  ( 4., 0.)   3    3
( 2., 2.)  4   2
( 5., 0.)  4   4
( 0.,-1.)  5   1
( 1., 0.)  5   4
( 4., 0.)  5   5
( 1., 3.)  6   2
( 0.,-2.)  6   5
( 3., 0.)  6   6
( 2., 1.)  7   1
(-1., 0.)  7   2
(-3.,-1.)  7   3
( 5., 0.)  7   7  A(I), IROW(I), ICOL(I), I=1,...,NNZ
```

9.3 Program Results

```
F11JNF Example Program Results
 Original Matrix
 N     =          7
 NNZ   =         16
      1     (     0.6000E+01,     0.0000E+00)      1       1
      2     (     0.1000E+01,    -0.2000E+01)      2       1
      3     (     0.9000E+01,     0.0000E+00)      2       2
      4     (     0.4000E+01,     0.0000E+00)      3       3
      5     (     0.2000E+01,     0.2000E+01)      4       2
      6     (     0.5000E+01,     0.0000E+00)      4       4
      7     (     0.0000E+00,    -0.1000E+01)      5       1
      8     (     0.1000E+01,     0.0000E+00)      5       4
      9     (     0.4000E+01,     0.0000E+00)      5       5
     10     (     0.1000E+01,     0.3000E+01)      6       2
     11     (     0.0000E+00,    -0.2000E+01)      6       5
     12     (     0.3000E+01,     0.0000E+00)      6       6
     13     (     0.2000E+01,     0.1000E+01)      7       1
     14     (    -0.1000E+01,     0.0000E+00)      7       2
     15     (    -0.3000E+01,    -0.1000E+01)      7       3
     16     (     0.5000E+01,     0.0000E+00)      7       7

 Factorization
 N     =          7
 NNZ   =         16
 NPIVM =          0
     17     (     0.2500E+00,     0.0000E+00)      1       1
     18     (     0.2000E+00,     0.0000E+00)      2       2
     19     (     0.2000E+00,     0.0000E+00)      3       2
     20     (     0.2632E+00,     0.0000E+00)      3       3
     21     (     0.0000E+00,    -0.5263E+00)      4       3
     22     (     0.5135E+00,     0.0000E+00)      4       4
     23     (     0.0000E+00,     0.2632E+00)      5       3
     24     (     0.1743E+00,     0.0000E+00)      5       5
     25     (    -0.7500E+00,    -0.2500E+00)      6       1
     26     (     0.3486E+00,     0.1743E+00)      6       5
     27     (     0.6141E+00,     0.0000E+00)      6       6
     28     (     0.4000E+00,    -0.4000E+00)      7       2
     29     (     0.5135E+00,    -0.1541E+01)      7       4
     30     (     0.1743E+00,    -0.3486E+00)      7       5
     31     (    -0.6141E+00,     0.5352E+00)      7       6
     32     (     0.3197E+01,     0.0000E+00)      7       7
```

```
I     IPIV(I)
 1       3
 2       4
 3       5
 4       6
 5       1
 6       7
 7       2
```

F11JPF – 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

F11JPF solves a system of complex linear equations involving the incomplete Cholesky preconditioning matrix generated by F11JNF.

2 Specification

```
 SUBROUTINE F11JPF(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
 complex           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 $M = PLDL^HP^T$, corresponding to an incomplete Cholesky decomposition of a complex sparse Hermitian matrix stored in symmetric coordinate storage (SCS) format (see Section 2.1.2 of the Chapter Introduction), as generated by F11JNF.

In the above decomposition L is a complex lower triangular sparse matrix with unit diagonal, D is a real diagonal matrix and P is a permutation matrix. L and D are supplied to F11JPF through the matrix

$$C = L + D^{-1} - I$$

which is a lower triangular n by n complex sparse matrix, stored in SCS format, as returned by F11JNF. The permutation matrix P is returned from F11JNF via the array IPIV.

F11JPF may also be used in combination with F11JNF to solve a sparse complex Hermitian positive-definite system of linear equations directly (see Section 8.4 of the document for F11JNF). This is illustrated 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 preceding call to F11JNF.

Constraint: N $\geq$ 1.

2: A(LA) — ***complex*** array *Input*

On entry: the values returned in array A by a previous call to F11JNF.

3: LA — INTEGER *Input*

On entry: the dimension of the arrays A, IROW and ICOL as declared in the (sub)program from which F11JPF is called. This **must** be the same value as was supplied in the preceding call to F11JNF.

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 F11JNF.

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.

Constraint: CHECK = 'C' or 'N'.

9: Y(N) — ***complex*** array *Input*

On entry: the right-hand side vector y.

10: X(N) — ***complex*** 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 $\neq$ '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 F11JPF has been preceded by a valid call to F11JNF 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

$$|\delta M| \leq c(n)\epsilon P|L||D||L^H|P^T,$$

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

8 Further Comments

8.1 Timing

The time taken for a call to F11JPF is proportional to the value of NNZC returned from F11JNF.

9 Example

This example program reads in a complex sparse Hermitian positive-definite matrix A and a vector y. It then calls F11JNF, with LFILL $= -1$ and DTOL $= 0.0$, to compute the **complete** Cholesky decomposition of A:

$$A = PLDL^H P^T.$$

Finally it calls F11JPF to solve the system

$$PLDL^H P^T x = y.$$

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.

```
*     F11JPF Example Program Text.
*     Mark 19 Release. NAG Copyright 1999.
*     .. 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 ..
      complex          A(LA), X(NMAX), Y(NMAX)
      INTEGER          ICOL(LA), IPIV(NMAX), IROW(LA), ISTR(NMAX+1),
     +                 IWORK(LIWORK)
*     .. External Subroutines ..
      EXTERNAL         F11JNF, F11JPF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11JPF 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 F11JNF(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
*                           H T
*           Solve P L D L P x = y
*
            CHECK = 'C'
*
            CALL F11JPF(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,',',e16.4,')')
      END
```

9.2 Program Data

```
F11JPF Example Program Data
  9                   N
 23                   NNZ
 ( 6., 0.)    1    1
 (-1., 1.)    2    1
 ( 6., 0.)    2    2
 ( 0., 1.)    3    2
 ( 5., 0.)    3    3
 ( 5., 0.)    4    4
 ( 2.,-2.)    5    1
 ( 4., 0.)    5    5
 ( 1., 1.)    6    3
 ( 2., 0.)    6    4
 ( 6., 0.)    6    6
```

```
(-4., 3.)   7    2
( 0., 1.)   7    5
(-1., 0.)   7    6
( 6., 0.)   7    7
(-1.,-1.)   8    4
( 0.,-1.)   8    6
( 9., 0.)   8    8
( 1., 3.)   9    1
( 1., 2.)   9    5
(-1., 0.)   9    6
( 1., 4.)   9    8
( 9., 0.)   9    9    A(I), IROW(I), ICOL(I), I=1,...,NNZ
( 8.,54.) (-10.,-92.)
(25.,27.) (26., -28.)
(54.,12.) (26.,-22.)
(47.,65.) (71.,-57.)
(60.,70.)            Y(I), I=1,...,N
```

9.3 Program Results

```
F11JPF Example Program Results
Solution of linear system
(      0.1000E+01,      0.9000E+01)
(      0.2000E+01,     -0.8000E+01)
(      0.3000E+01,      0.7000E+01)
(      0.4000E+01,     -0.6000E+01)
(      0.5000E+01,      0.5000E+01)
(      0.6000E+01,     -0.4000E+01)
(      0.7000E+01,      0.3000E+01)
(      0.8000E+01,     -0.2000E+01)
(      0.9000E+01,      0.1000E+01)
```

F11JQF – 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

F11JQF solves a complex sparse Hermitian system of linear equations, represented in symmetric coordinate storage format, using a conjugate gradient or Lanczos method, with incomplete Cholesky preconditioning.

2 Specification

```
SUBROUTINE F11JQF(METHOD, N, NNZ, A, LA, IROW, ICOL, IPIV, ISTR,
1                 B, TOL, MAXITN, X, RNORM, ITN, WORK, LWORK, IFAIL)
INTEGER           N, NNZ, LA, IROW(LA), ICOL(LA), IPIV(N),
1                 ISTR(N+1), MAXITN, ITN, LWORK, IFAIL
complex           A(LA), B(N), X(N), WORK(LWORK)
real              TOL, RNORM
CHARACTER*(*)     METHOD
```

3 Description

This routine solves a complex sparse Hermitian 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].

F11JQF uses the incomplete Cholesky factorization determined by F11JNF as the preconditioning matrix. A call to F11JQF must always be preceeded by a call to F11JNF. Alternative preconditioners for the same storage scheme are available by calling F11JSF.

The matrix A and the preconditioning matrix M are represented in symmetric coordinate storage (SCS) format (see Section 2.1.2 of the Chapter Introduction) in the arrays A, IROW and ICOL, as returned from F11JNF. 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

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 preceding call to F11JNF.

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 preceding call to F11JNF.

Constraint: 1 $\leq$ NNZ $\leq$ N × (N+1)/2.

4: A(LA) — ***complex*** array *Input*

On entry: the values returned in array A by a previous call to F11JNF.

5: LA — INTEGER *Input*

On entry: the dimension of the arrays A, IROW and ICOL as declared in the (sub)program from which F11JQF is called. This **must** be the same value as was supplied in the preceding call to F11JNF.

Constraint: LA $\geq$ 2 × 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 F11JNF.

10: B(N) — ***complex*** 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 ϵ 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) — ***complex*** 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) — ***complex*** array *Workspace*

17: LWORK — INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F11JQF is called.

Constraints:

if METHOD = 'CG', LWORK $\geq 6 \times$ N + 120;
if METHOD = 'SYMMLQ', LWORK $\geq 7 \times$ N+ 120.

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 N < 1,
or NNZ < 1,
or NNZ > N × (N+1)/2,
or LA too small,
or TOL $\geq$ 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 F11JQF has been preceded by a valid call to F11JNF, and that the arrays A, IROW and ICOL have not been corrupted between the two calls.

IFAIL = 3

On entry, the SCS representation of M is invalid. Further details are given in the error message. Check that the call to F11JQF has been preceded by a valid call to F11JNF, 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 an auxiliary routine. 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 F11JQF for each iteration is roughly proportional to the value of NNZC returned from the preceding call to F11JNF. 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 easily be 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 complex sparse Hermitian positive-definite system of equations using the conjugate gradient method, with incomplete Cholesky 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.

```
*     F11JQF Example Program Text.
*     Mark 19 Release. NAG Copyright 1999.
*     .. 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=7*NMAX+120)
*     .. 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 ..
      complex          A(LA), B(NMAX), WORK(LWORK), X(NMAX)
      INTEGER          ICOL(LA), IPIV(NMAX), IROW(LA), ISTR(NMAX+1),
     +                 IWORK(LIWORK)
*     .. External Subroutines ..
      EXTERNAL         F11JNF, F11JQF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11JQF 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 rhs 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 F11JNF(N,NNZ,A,LA,IROW,ICOL,LFILL,DTOL,MIC,DSCALE,PSTRAT,
     +               IPIV,ISTR,NNZC,NPIVM,IWORK,LIWORK,IFAIL)
*
*        Solve Ax = b using F11JQF
*
         CALL F11JQF(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,'(',e16.4,',',e16.4,')')
      END
```

9.2 Program Data

```
F11JQF Example Program Data
  9                         N
  23                        NNZ
 'CG'                       METHOD
  0 0.0                     LFILL, DTOL
  'N' 0.0                   MIC, DSCALE
  'M'                       PSTRAT
  1.0E-6 100                TOL, MAXITN
 ( 6., 0.)  1   1
 (-1., 1.)  2   1
 ( 6., 0.)  2   2
 ( 0., 1.)  3   2
 ( 5., 0.)  3   3
 ( 5., 0.)  4   4
 ( 2.,-2.)  5   1
 ( 4., 0.)  5   5
 ( 1., 1.)  6   3
 ( 2., 0.)  6   4
 ( 6., 0.)  6   6
 (-4., 3.)  7   2
 ( 0., 1.)  7   5
 (-1., 0.)  7   6
 ( 6., 0.)  7   7
 (-1.,-1.)  8   4
 ( 0.,-1.)  8   6
 ( 9., 0.)  8   8
 ( 1., 3.)  9   1
 ( 1., 2.)  9   5
 (-1., 0.)  9   6
 ( 1., 4.)  9   8
 ( 9., 0.)  9   9       A(I), IROW(I), ICOL(I), I=1,...,NNZ
 (  8.,  54.)
 (-10., -92.)
 ( 25.,  27.)
 ( 26., -28.)
 ( 54.,  12.)
 ( 26., -22.)
 ( 47.,  65.)
 ( 71., -57.)
 ( 60.,  70.)            B(I), I=1,...,N
 (  0.,   0.)
 (  0.,   0.)
 (  0.,   0.)
 (  0.,   0.)
 (  0.,   0.)
 (  0.,   0.)
 (  0.,   0.)
 (  0.,   0.)
 (  0.,   0.)            X(I), I=1,...,N
```

9.3 Program Results

```
F11JQF Example Program Results
Converged in         5 iterations
Final residual norm =       3.197E-14
(      0.1000E+01,       0.9000E+01)
(      0.2000E+01,      -0.8000E+01)
(      0.3000E+01,       0.7000E+01)
(      0.4000E+01,      -0.6000E+01)
(      0.5000E+01,       0.5000E+01)
(      0.6000E+01,      -0.4000E+01)
(      0.7000E+01,       0.3000E+01)
(      0.8000E+01,      -0.2000E+01)
(      0.9000E+01,       0.1000E+01)
```

F11JRF – 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

F11JRF solves a system of linear equations involving the preconditioning matrix corresponding to SSOR applied to a complex sparse Hermitian matrix, represented in symmetric coordinate storage format.

2 Specification

```
 SUBROUTINE F11JRF(N, NNZ, A, IROW, ICOL, RDIAG, OMEGA, CHECK, Y,
1                  X, IWORK, IFAIL)
 INTEGER           N, NNZ, IROW(NNZ), ICOL(NNZ), IWORK(N+1), IFAIL
 complex           A(NNZ), Y(N), X(N)
 real              RDIAG(N), OMEGA
 CHARACTER*1       CHECK
```

3 Description

This routine solves a system of equations:

$$Mx = y$$

involving the preconditioning matrix:

$$M = \frac{1}{\omega(2-\omega)}(D+\omega L)D^{-1}(D+\omega L)^H$$

corresponding to symmetric successive-over-relaxation (SSOR) [1] on a linear system $Ax = b$, where A is a sparse complex Hermitian matrix stored in symmetric coordinate storage (SCS) format (see Section 2.1.2 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 ω is a user-defined relaxation parameter. Note that since A is Hermitian the matrix D is necessarily real.

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: 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 $\leq$ NNZ $\leq$ N × (N+1)/2.

3: A(NNZ) — ***complex*** 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 F11ZPF 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 F11ZPF):

$1 \leq \mathrm{IROW}(i) \leq \mathrm{N}$, $1 \leq \mathrm{ICOL}(i) \leq \mathrm{IROW}(i)$, 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: RDIAG(N) — ***real*** array *Input*

On entry: the elements of the diagonal matrix D^{-1}, where D is the diagonal part of A. Note that since A is Hermitian the elements of D^{-1} are necessarily real.

7: OMEGA — ***real*** *Input*

On entry: the relaxation parameter ω.

Constraint: $0.0 \leq \mathrm{OMEGA} \leq 2.0$.

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.

Constraint: CHECK = 'C' or 'N'.

9: Y(N) — ***complex*** array *Input*

On entry: the right-hand side vector y.

10: X(N) — ***complex*** 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:

$1 \leq$ IROW(i) $\leq$ N and $1 \leq$ ICOL(i) $\leq$ IROW(i), 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.

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 F11ZPF to reorder and sum or remove duplicates.

IFAIL = 4

On entry, a row of A has no diagonal entry.

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 ϵ is the ***machine precision***.

8 Further Comments

8.1 Timing

The time taken for a call to F11JRF is proportional to NNZ.

9 Example

None.

F11JSF – 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

F11JSF solves a complex sparse Hermitian 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 F11JSF(METHOD, PRECON, N, NNZ, A, IROW, ICOL, OMEGA, B,
1                  TOL, MAXITN, X, RNORM, ITN, RDIAG, WORK, LWORK,
2                  IWORK, IFAIL)
 INTEGER           N, NNZ, IROW(NNZ), ICOL(NNZ), MAXITN, ITN,
1                  IWORK(N+1), LWORK, IFAIL
 complex           A(NNZ), B(N), X(N), WORK(LWORK)
 real              RDIAG(N), OMEGA, TOL, RNORM
 CHARACTER*(*)     METHOD
 CHARACTER*1       PRECON
```

3 Description

This routine solves a complex sparse Hermitian 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].

F11JSF 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 F11JQF.

The matrix A is represented in symmetric coordinate storage (SCS) format (see Section 2.1.2 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 (SSOR).

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 lower triangular part of the matrix A.

Constraint: $1 \leq$ NNZ $\leq$ N $\times$ (N+1)/2.

5: A(NNZ) — ***complex*** 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 F11ZPF 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 F11ZPF):

$1 \leq$ IROW(i) $\leq$ N and $1 \leq$ ICOL(i) $\leq$ IROW(i), 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.

8: OMEGA — ***real*** *Input*

On entry: if PRECON = 'S', OMEGA is the relaxation parameter ω to be used in the SSOR method. Otherwise OMEGA need not be initialized.

Constraint: 0.0 $\leq$ OMEGA $\leq$ 2.0.

9: B(N) — ***complex*** 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 ϵ is the ***machine precision***. Otherwise $\tau = \max(\mathrm{TOL}, 10\epsilon, \sqrt{n}\,\epsilon)$ 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) — ***complex*** 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: RDIAG(N) — ***real*** array *Output*

On exit: the elements of the diagonal matrix D^{-1}, where D is the diagonal part of A. Note that since A is Hermitian the elements of D^{-1} are necessarily real.

16: WORK(LWORK) — ***complex*** array *Workspace*

17: LWORK — INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F11JSF is called.

Constraints:

if METHOD = 'CG', LWORK $\geq$ 6 × N + 120;
if METHOD = 'SYMMLQ', LWORK $\geq$ 7 × N + 120.

18: IWORK(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 ≠ 'CG' or 'SYMMLQ',
or PRECON ≠ '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 is 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 F11ZPF 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 an auxiliary routine. Check all subroutine calls and array sizes. Seek expert help.

IFAIL = 9

The matrix of the coefficients has a non-real diagonal entry, and is therefore not Hermitian.

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 F11JSF 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 easily be 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 complex sparse Hermitian positive-definite system of equations using the conjugate gradient 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.

```
*     F11JSF Example Program Text.
*     Mark 19 Release. NAG Copyright 1999.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, LA, LWORK
      PARAMETER        (NMAX=1000,LA=10000,LWORK=7*NMAX+120)
*     .. Local Scalars ..
      real             OMEGA, RNORM, TOL
      INTEGER          I, IFAIL, ITN, MAXITN, N, NNZ
      CHARACTER        PRECON
      CHARACTER*6      METHOD
*     .. Local Arrays ..
      complex          A(LA), B(NMAX), WORK(LWORK), X(NMAX)
      real             RDIAG(NMAX)
      INTEGER          ICOL(LA), IROW(LA), IWORK(NMAX+1)
*     .. External Subroutines ..
      EXTERNAL         F11JSF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11JSF 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 rhs 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 F11JSF
*
         IFAIL = 0
         CALL F11JSF(METHOD,PRECON,N,NNZ,A,IROW,ICOL,OMEGA,B,TOL,MAXITN,
     +               X,RNORM,ITN,RDIAG,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,'(',e16.4,',',e16.4,')')
      END
```

9.2 Program Data

```
F11JSF Example Program Data
  9                      N
  23                     NNZ
 'CG' 'SSOR'             METHOD, PRECON
  1.1                    OMEGA
  1.0E-6 100             TOL, MAXITN
 ( 6., 0.)   1    1
 (-1., 1.)   2    1
 ( 6., 0.)   2    2
 ( 0., 1.)   3    2
 ( 5., 0.)   3    3
 ( 5., 0.)   4    4
 ( 2.,-2.)   5    1
 ( 4., 0.)   5    5
 ( 1., 1.)   6    3
 ( 2., 0.)   6    4
 ( 6., 0.)   6    6
 (-4., 3.)   7    2
 ( 0., 1.)   7    5
 (-1., 0.)   7    6
```

```
( 6., 0.)   7   7
(-1.,-1.)   8   4
( 0.,-1.)   8   6
( 9., 0.)   8   8
( 1., 3.)   9   1
( 1., 2.)   9   5
(-1., 0.)   9   6
( 1., 4.)   9   8
( 9., 0.)   9   9     A(I), IROW(I), ICOL(I), I=1,...,NNZ
(  8.,  54.)
(-10., -92.)
( 25.,  27.)
( 26., -28.)
( 54.,  12.)
( 26., -22.)
( 47.,  65.)
( 71., -57.)
( 60.,  70.)          B(I), I=1,...,N
(  0.,   0.)
(  0.,   0.)
(  0.,   0.)
(  0.,   0.)
(  0.,   0.)
(  0.,   0.)
(  0.,   0.)
(  0.,   0.)
(  0.,   0.)          X(I), I=1,...,N
```

9.3 Program Results

```
F11JSF Example Program Results
Converged in          7 iterations
Final residual norm =      1.477E-05
(      0.1000E+01,     0.9000E+01)
(      0.2000E+01,    -0.8000E+01)
(      0.3000E+01,     0.7000E+01)
(      0.4000E+01,    -0.6000E+01)
(      0.5000E+01,     0.5000E+01)
(      0.6000E+01,    -0.4000E+01)
(      0.7000E+01,     0.3000E+01)
(      0.8000E+01,    -0.2000E+01)
(      0.9000E+01,     0.1000E+01)
```

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):

$1 \leq \mathrm{IROW}(i) \leq \mathrm{N}$, $1 \leq \mathrm{ICOL}(i) \leq \mathrm{N}$, 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.

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 $\mathrm{NNZ} > \mathrm{N}^2$.

IFAIL = 3

On entry, the arrays IROW and ICOL fail to satisfy the following constraints:

$1 \leq \mathrm{IROW}(i) \leq \mathrm{N}$ and $1 \leq \mathrm{ICOL}(i) \leq \mathrm{N}$, 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.

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

$$||y - A^Tx||_\infty \leq c(n)\epsilon||A^T||_\infty||x||_\infty,$$

if TRANS = 'T', where $c(n)$ is a modest linear function of n, and ϵ 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 $y = A^Tx$.

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 ϵ 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
```

F11XNF – 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 conjugate transposed matrix-vector product involving a complex sparse non-Hermitian matrix stored in coordinate storage format.

2 Specification

```
SUBROUTINE F11XNF(TRANS, N, NNZ, A, IROW, ICOL, CHECK, X, Y, IFAIL)
INTEGER          N, NNZ, IROW(NNZ), ICOL(NNZ), IFAIL
complex          A(NNZ), X(N), Y(N)
CHARACTER*1      TRANS, CHECK
```

3 Description

F11XNF computes either the matrix-vector product $y = Ax$, or the conjugate transposed matrix-vector product $y = A^H x$, according to the value of the argument TRANS, where A is a complex n by n sparse non-Hermitian 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 the 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 F11XNF will be to compute the matrix-vector product required in the application of F11BSF to sparse complex linear systems. This is illustrated in Section 9 of the document for F11DRF.

4 References

None.

5 Parameters

1: TRANS — CHARACTER*1 *Input*

On entry: specifies whether or not the matrix A is conjugate transposed:

if TRANS = 'N', then $y = Ax$ is computed;
if TRANS = 'T', then $y = A^H x$ 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) — ***complex*** 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 F11ZNF 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.

Constraint: IROW and ICOL must satisfy the following constraints (which may be imposed by a call to F11ZNF):

$1 \leq \text{IROW}(i) \leq \text{N}$ 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.

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) — ***complex*** array *Input*

On entry: the vector x.

9: Y(N) — ***complex*** 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, $\text{N} < 1$,
or $\text{NNZ} < 1$,
or $\text{NNZ} > \text{N}^2$.

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 $i = 1, 2, \ldots$,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 F11ZNF to reorder and sum or remove duplicates.

7 Accuracy

The computed vector y satisfies the error bound:

$$\|y - Ax\|_\infty \le c(n)\epsilon\|A\|_\infty\|x\|_\infty,$$

if TRANS = 'N', or

$$\|y - A^Tx\|_\infty \le c(n)\epsilon\|A^T\|_\infty\|x\|_\infty,$$

if TRANS = 'T', where $c(n)$ is a modest linear function of n, and ϵ is the ***machine precision***.

8 Further Comments

8.1 Timing

The time taken for a call to F11XNF is proportional to NNZ.

8.2 Use of CHECK

It is expected that a common use of F11XNF will be to compute the matrix-vector product required in the application of F11BSF to sparse complex linear systems. In this situation F11XNF 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 complex sparse matrix A and a vector x. It then calls F11XNF to compute the matrix-vector product $y = Ax$ and the conjugate transposed matrix-vector product $y = A^Hx$.

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.

```
*     F11XNF Example Program Text.
*     Mark 19 Release. NAG Copyright 1999.
*     .. 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 ..
      complex         A(LA), X(NMAX), Y(NMAX)
      INTEGER         ICOL(LA), IROW(LA)
*     .. External Subroutines ..
      EXTERNAL        F11XNF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11XNF 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 F11XNF(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,'(1X,''('',D16.4,'','',D16.4,'')'')') Y(I)
   40    CONTINUE
*
*        Calculate conjugate transposed matrix-vector product
*
         TRANS = 'Transposed'
         CHECK = 'N'
         IFAIL = 0
         CALL F11XNF(TRANS,N,NNZ,A,IROW,ICOL,CHECK,X,Y,IFAIL)
*
*        Output results
*
         WRITE (NOUT,*)
         WRITE (NOUT,*) ' Conjugate transposed matrix-vector product'
         DO 60 I = 1, N
            WRITE (NOUT,'(1X,''('',D16.4,'','',D16.4,'')'')') Y(I)
   60    CONTINUE
*
      END IF
      STOP
      END
```

9.2 Program Data

```
F11XNF Example Program Data
  5                      N
 11                      NNZ
  ( 2., 3.)   1    1
  ( 1.,-4.)   1    2
  ( 1., 0.)   2    3
  (-1.,-2.)   2    4
  ( 4., 1.)   3    1
  ( 0., 1.)   3    3
 ( 1., 3.)  3    5
 ( 0.,-1.)  4    4
 ( 2.,-6.)  4    5
 (-2., 0.)  5    2
 ( 3., 1.)  5    5      A(I), IROW(I), ICOL(I), I=1,...,NNZ
( 0.70, 0.21)
( 0.16,-0.43)
( 0.52, 0.97)
( 0.77, 0.00)
( 0.28,-0.64)           X(I), I=1,...,N
```

9.3 Program Results

```
F11XNF Example Program Results

 Matrix-vector product
(     -0.7900E+00,      0.1450E+01)
(     -0.2500E+00,     -0.5700E+00)
(      0.3820E+01,      0.2260E+01)
(     -0.3280E+01,     -0.3730E+01)
(      0.1160E+01,     -0.7800E+00)

 Conjugate transposed matrix-vector product
(      0.5080E+01,      0.1680E+01)
(     -0.7000E+00,      0.4290E+01)
(      0.1130E+01,     -0.9500E+00)
(      0.7000E+00,      0.1520E+01)
(      0.5170E+01,      0.1830E+01)
```

F11XSF – 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 complex sparse Hermitian matrix stored in symmetric coordinate storage format.

2 Specification

```
SUBROUTINE F11XSF(N, NNZ, A, IROW, ICOL, CHECK, X, Y, IFAIL)
INTEGER          N, NNZ, IROW(NNZ), ICOL(NNZ), IFAIL
complex          A(NNZ), X(N), Y(N)
CHARACTER*1      CHECK
```

3 Description

F11XSF computes the matrix-vector product

$$y = Ax$$

where A is an n by n complex Hermitian sparse matrix, of arbitrary sparsity pattern, stored in symmetric coordinate storage (SCS) format (see Section 2.1.2 of the Chapter Introduction). The array A stores all the non-zero elements in the lower triangular part of A, while arrays IROW and ICOL store the corresponding row and column indices respectively.

4 References

None.

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 lower triangular part of the matrix A.

Constraint: 1 $\leq$ NNZ $\leq$ N $\times$ (N+1)/2.

3: A(NNZ) — ***complex*** 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 F11ZPF 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 F11ZPF):

$1 \leq \text{IROW}(i) \leq \text{N}$, $1 \leq \text{ICOL}(i) \leq \text{IROW}(i)$, 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.

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.

Constraint: CHECK = 'C' or 'N'.

7: X(N) — ***complex*** array *Input*

On entry: the vector x.

8: Y(N) — ***complex*** 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:

$1 \leq \text{IROW}(i) \leq \text{N}$ and $1 \leq \text{ICOL}(i) \leq \text{IROW}(i)$, 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 lower triangular part of A, is out of order, or has duplicate row and column indices. Call F11ZPF 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 ϵ is the ***machine precision***.

8 Further Comments

8.1 Timing

The time taken for a call to F11XSF is proportional to NNZ.

9 Example

This example program reads in a complex sparse Hermitian positive-definite matrix A and a vector x. It then calls F11XSF to compute the matrix-vector product $y = Ax$.

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.

```
*     F11XSF Example Program Text.
*     Mark 19 Release. NAG Copyright 1999.
*     .. 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 ..
      complex          A(LA), X(NMAX), Y(NMAX)
      INTEGER          ICOL(LA), IROW(LA)
*     .. External Subroutines ..
      EXTERNAL         F11XSF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11XSF 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 F11XSF(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,',',e16.4,')')
      END
```

9.2 Program Data

```
F11XSF Example Program Data
  9                    N
 23                    NNZ
 ( 6., 0.)  1   1
 (-1., 1.)  2   1
 ( 6., 0.)  2   2
 ( 0., 1.)  3   2
 ( 5., 0.)  3   3
 ( 5., 0.)  4   4
 ( 2.,-2.)  5   1
 ( 4., 0.)  5   5
 ( 1., 1.)  6   3
 ( 2., 0.)  6   4
 ( 6., 0.)  6   6
 (-4., 3.)  7   2
 ( 0., 1.)  7   5
 (-1., 0.)  7   6
 ( 6., 0.)  7   7
 (-1.,-1.)  8   4
 ( 0.,-1.)  8   6
 ( 9., 0.)  8   8
 ( 1., 3.)  9   1
 ( 1., 2.)  9   5
 (-1., 0.)  9   6
 ( 1., 4.)  9   8
 ( 9., 0.)  9   9  A(I), IROW(I), ICOL(I), I=1,...,NNZ
 (1., 9.) (2.,-8.) (3., 7.)
 (4.,-6.) (5., 5.) (6.,-4.)
 (7., 3.) (8.,-2.) (9., 1.)  X(I), I=1,...,N
```

9.3 Program Results

```
F11XSF Example Program Results
 Matrix-vector product
(      0.8000E+01,      0.5400E+02)
(     -0.1000E+02,     -0.9200E+02)
(      0.2500E+02,      0.2700E+02)
(      0.2600E+02,     -0.2800E+02)
(      0.5400E+02,      0.1200E+02)
(      0.2600E+02,     -0.2200E+02)
(      0.4700E+02,      0.6500E+02)
(      0.7100E+02,     -0.5700E+02)
(      0.6000E+02,      0.7000E+02)
```

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: N $\geq$ 1.

2: NNZ — INTEGER *Input/Output*

On entry: the number of non-zero elements in the matrix A.

Constraint: 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 $i = 1,2,\ldots,$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 $i = 1,2,\ldots,$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 $i = 1,2,\ldots,$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 19 Revised. NAG Copyright 1999.
*     .. 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,D16.4,2I8)') 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,D16.4,2I8)') 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: 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: 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: $1 \leq \mathrm{IROW}(i) \leq \mathrm{N}$, for $i = 1, 2, \ldots$,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: $1 \leq \mathrm{ICOL}(i) \leq \mathrm{IROW}(i)$, for $i = 1, 2, \ldots$,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 $i = 1, 2, \ldots$,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:

$$1 \leq \mathrm{IROW}(i) \leq \mathrm{N},$$
$$1 \leq \mathrm{ICOL}(i) \leq \mathrm{IROW}(i),$$

for $i = 1, 2, \ldots, \mathrm{NNZ}$.

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 $\mathrm{ISTR}(i) = \mathrm{ISTR}(i+1)$.

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
```

F11ZNF – 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 complex sparse non-Hermitian matrix, represented in coordinate storage format.

2 Specification

```
      SUBROUTINE F11ZNF(N, NNZ, A, IROW, ICOL, DUP, ZERO, ISTR, IWORK,
     1                  IFAIL)
      INTEGER           N, NNZ, IROW(*), ICOL(*), ISTR(N+1), IWORK(N),
     1                  IFAIL
      complex           A(*)
      CHARACTER*1       DUP, ZERO
```

3 Description

F11ZNF takes a coordinate storage (CS) representation (see Section 2.1.1 of the Chapter Introduction) of a complex n by n sparse non-Hermitian 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.

The routine also returns a pointer array 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: N $\geq$ 1.

2: NNZ — INTEGER *Input/Output*

On entry: the number of non-zero elements in the matrix A.

Constraint: NNZ $\geq$ 0.

On exit: the number of non-zero elements with unique row and column indices.

3: A(*) — ***complex*** 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 $i = 1, 2, \ldots$,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 $i = 1, 2, \ldots$,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 with IFAIL = 3 on detecting a duplicate.

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 with IFAIL = 4 on detecting a zero.

Constraint: ZERO = 'R', 'K' or 'F'.

8: ISTR(N+1) — INTEGER array *Output*

On exit: ISTR(i), for $i = 1, 2, \ldots$,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. See also Section 8.

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 F11ZNF 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 complex sparse matrix A, calls F11ZNF 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.

```
*     F11ZNF Example Program Text.
*     Mark 19 Release. NAG Copyright 1999.
*     .. 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 ..
      complex          A(LA)
      INTEGER          ICOL(LA), IROW(LA), ISTR(NMAX+1), IWORK(NMAX)
*     .. External Subroutines ..
      EXTERNAL         F11ZNF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11ZNF 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,'(I8,5X,''('',D16.4,'','',D16.4,'')'',2I8)') I,
     +        A(I), IROW(I), ICOL(I)
   40    CONTINUE
*
*        Reorder, sum duplicates and remove zeros
*
         DUP = 'S'
         ZERO = 'R'
         IFAIL = 0
*
         CALL F11ZNF(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,'(I8,5X,''('',D16.4,'','',D16.4,'')'',2I8)') I,
     +        A(I), IROW(I), ICOL(I)
   60    CONTINUE
*
      END IF
      STOP
      END
```

9.2 Example Data

```
F11ZNF Example Program Data
  5                    N
  15                   NNZ
  ( 4., 1.)   3    1
  (-2., 6.)   5    2
  ( 1.,-3.)   4    4
  (-2.,-1.)   4    2
  (-3., 0.)   5    5
  ( 1., 2.)   1    2
  ( 0., 0.)   1    5
  ( 1., 3.)   3    5
  (-1.,-1.)   2    4
  ( 6.,-3.)   5    5
  ( 2., 6.)   1    1
  ( 2., 1.)   4    2
  ( 1., 0.)   2    3
  ( 0.,-3.)   3    3
  ( 2., 2.)   4    5    A(I), IROW(I), ICOL(I), I=1,...,NNZ
```

9.3 Example Results

```
F11ZNF Example Program Results

Original elements
NNZ =    15
      1    (      0.4000E+01,      0.1000E+01)      3       1
      2    (     -0.2000E+01,      0.6000E+01)      5       2
      3    (      0.1000E+01,     -0.3000E+01)      4       4
      4    (     -0.2000E+01,     -0.1000E+01)      4       2
      5    (     -0.3000E+01,      0.0000E+00)      5       5
      6    (      0.1000E+01,      0.2000E+01)      1       2
      7    (      0.0000E+00,      0.0000E+00)      1       5
      8    (      0.1000E+01,      0.3000E+01)      3       5
      9    (     -0.1000E+01,     -0.1000E+01)      2       4
     10    (      0.6000E+01,     -0.3000E+01)      5       5
     11    (      0.2000E+01,      0.6000E+01)      1       1
     12    (      0.2000E+01,      0.1000E+01)      4       2
     13    (      0.1000E+01,      0.0000E+00)      2       3
     14    (      0.0000E+00,     -0.3000E+01)      3       3
     15    (      0.2000E+01,      0.2000E+01)      4       5
Reordered elements
NNZ =    11
      1    (      0.2000E+01,      0.6000E+01)      1       1
      2    (      0.1000E+01,      0.2000E+01)      1       2
      3    (      0.1000E+01,      0.0000E+00)      2       3
      4    (     -0.1000E+01,     -0.1000E+01)      2       4
      5    (      0.4000E+01,      0.1000E+01)      3       1
      6    (      0.0000E+00,     -0.3000E+01)      3       3
      7    (      0.1000E+01,      0.3000E+01)      3       5
      8    (      0.1000E+01,     -0.3000E+01)      4       4
      9    (      0.2000E+01,      0.2000E+01)      4       5
     10    (     -0.2000E+01,      0.6000E+01)      5       2
     11    (      0.3000E+01,     -0.3000E+01)      5       5
```

F11ZPF – 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 sparse complex Hermitian matrix, represented in symmetric coordinate storage format.

2 Specification

```
      SUBROUTINE F11ZPF(N, NNZ, A, IROW, ICOL, DUP, ZERO, ISTR, IWORK,
     1                  IFAIL)
      INTEGER           N, NNZ, IROW(*), ICOL(*), ISTR(N+1), IWORK(N),
     1                  IFAIL
      complex           A(*)
      CHARACTER*1       DUP, ZERO
```

3 Description

F11ZPF takes a symmetric coordinate storage (SCS) representation (see Section 2.1.2 of the Chapter Introduction) of a sparse n by n complex Hermitian 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.

The routine also returns a pointer array 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: 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: NNZ $\geq$ 0.

On exit: the number of lower triangular non-zero elements with unique row and column indices.

3: A(∗) — ***complex*** 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: $1 \leq \mathrm{IROW}(i) \leq \mathrm{N}$, for $i = 1, 2, \ldots, \mathrm{NNZ}$.

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: $1 \leq \mathrm{ICOL}(i) \leq \mathrm{IROW}(i)$, for $i = 1, 2, \ldots, \mathrm{NNZ}$.

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;
if DUP = 'F', the routine fails with IFAIL = 3 on detecting a duplicate.

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 with IFAIL = 4 on detecting a zero.

Constraint: ZERO = 'R', 'K' or 'F'.

8: ISTR(N+1) — INTEGER array *Output*

On exit: ISTR(i), for $i = 1, 2, \ldots, \mathrm{N}$, 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. See also Section 8.

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 in the lower triangular part of A, i.e., one or more of the following constraints has been violated:

$$1 \le \text{IROW}(i) \le \text{N},$$
$$1 \le \text{ICOL}(i) \le \text{IROW}(i),$$

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 F11ZPF 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 SCS representation of a complex sparse Hermitian matrix A, calls F11ZPF 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.**

```
*     F11ZPF Example Program Text.
*     Mark 19 Release. NAG Copyright 1999.
*     .. 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 ..
      complex          A(LA)
      INTEGER          ICOL(LA), IROW(LA), ISTR(NMAX+1), IWORK(NMAX)
*     .. External Subroutines ..
      EXTERNAL         F11ZPF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F11ZPF 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 F11ZPF(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 (I8,5X,'(',e16.4,',',e16.4,')',2I8)
      END
```

9.2 Program Data

```
F11ZPF Example Program Data
  4                    N
  9                    NNZ
  (1., 2.)   3    2
  (0., 0.)   2    1
  (0., 3.)   3    2
  (3.,-5.)   4    4
  (4., 2.)   1    1
  (0., 3.)   2    2
  (2., 4.)   3    3
  (1.,-1.)   3    2
  (1., 3.)   3    2    A(I), IROW(I), ICOL(I), I=1,...,NNZ
```

9.3 Program Results

```
F11ZPF Example Program Results
Original elements
NNZ =          9
     1    (      0.1000E+01,     0.2000E+01)    3      2
     2    (      0.0000E+00,     0.0000E+00)    2      1
     3    (      0.0000E+00,     0.3000E+01)    3      2
     4    (      0.3000E+01,    -0.5000E+01)    4      4
     5    (      0.4000E+01,     0.2000E+01)    1      1
     6    (      0.0000E+00,     0.3000E+01)    2      2
     7    (      0.2000E+01,     0.4000E+01)    3      3
     8    (      0.1000E+01,    -0.1000E+01)    3      2
     9    (      0.1000E+01,     0.3000E+01)    3      2
Reordered elements
NNZ =    5
     1    (      0.4000E+01,     0.2000E+01)    1      1
     2    (      0.0000E+00,     0.3000E+01)    2      2
     3    (      0.3000E+01,     0.7000E+01)    3      2
     4    (      0.2000E+01,     0.4000E+01)    3      3
     5    (      0.3000E+01,    -0.5000E+01)    4      4
```