F08FUFP (PZUNMTR)

NAG Parallel Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check for implementation-dependent details. You are advised to enclose any calls to NAG Parallel Library routines between calls to Z01AAFP and Z01ABFP.

1 Description

F08FUFP (PZUNMTR) multiplies an m by n complex matrix C_s by a complex unitary matrix Q, where C_s is a submatrix of a larger m_C by n_C matrix C, i.e.,

$$C_s(1:m, 1:n) \equiv C(i_C : i_C + m - 1, j_C : j_C + n - 1).$$

Note: if $i_C = j_C = 1$, $m = m_C$ and $n = n_C$, then $C_s \equiv C$.

The matrix Q is defined by the factorization, $A_s = QTQ^H$ with A_s complex Hermitian, Q unitary and T real symmetric tridiagonal, as computed by the routine F08FSFP (PZHETRD).

This routine may be used to form one of the products

$$QC_s, Q^H C_s, C_s Q \quad \text{or} \quad C_s Q^H$$

without explicitly forming Q. The result is overwriteen on C_s

F08FUFP (PZUNMTR) can be used to compute the eigenvectors of A_s following computation of the eigenvectors of T by F08JXFP (PZSTEIN).

2 Specification

```
 SUBROUTINE F08FUFP(SIDE, UPLO, TRANS, M, N, A, IA, JA, IDESCA,
1                   TAU, C, IC, JC, IDESCC, WORK, LWORK, INFO)
 ENTRY      PZUNMTR(SIDE, UPLO, TRANS, M, N, A, IA, JA, IDESCA,
1                   TAU, C, IC, JC, IDESCC, WORK, LWORK, INFO)
 COMPLEX*16         A(*), TAU(*), C(*), WORK(*)
 INTEGER            M, N, IA, JA, IDESCA(*), IC, JC, IDESCC(*),
1                   LWORK, INFO
 CHARACTER*1        SIDE, UPLO, TRANS
```

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

3 Usage

3.1 Definitions

The following definitions are used in describing the data distribution within this document:

m_p	–	the number of rows in the Library Grid.
n_p	–	the number of columns in the Library Grid.
p_r	–	the row grid coordinate of the calling processor.
p_c	–	the column grid coordinate of the calling processor.
M_b^X	–	the blocking factor for the distribution of the rows of a matrix X.
N_b^X	–	the blocking factor for the distribution of the columns of a matrix X.
s_r^X	–	the row coordinate of the processor that possesses the first row of a distributed matrix X.
s_c^X	–	the column coordinate of the processor that possesses the first column of a distributed matrix X.

numroc($\hat{\ell}$,L_b^X,p,s^X,ℓ_p) – a function which gives the **num**ber of **r**ows **o**r **c**olumns of a distributed matrix X owned by the processor with the row or column coordinate p (p_r or p_c), where $\hat{\ell}$ is the total number of rows or columns of the matrix, L_b^X is the blocking factor used (M_b^X or N_b^X), s^X is the row or column coordinate (s_r^X or s_c^X) of the processor that possesses the first row or column of the distributed matrix and ℓ_p is either m_p or n_p. The Library provides the function Z01CAFP (NUMROC) for the evaluation of this function.

indxg2p(k,L_b^X,p,s^X,ℓ_p) – a function which gives the processor row or column coordinate which possess the row or column index k of a distributed matrix. The arguments L_b^X, s^X and ℓ_p have the same meaning as in the function numroc. However, the argument p is a dummy integer. The Library provides the utility function Z01CDFP (INDXG2P) for the evaluation of this function.

iclm(i,j) – a function which gives the least common multiple (LCM) of the two integers i and j.

3.2 Global and Local Arguments

The following global **input** arguments must have the same value on entry to the routine on each processor and the global **output** arguments will have the same value on exit from the routine on each processor:

Global input arguments: SIDE, UPLO, TRANS, M, N, IA, JA, IC, JC, the array elements IDESCA(1), IDESCA(3:8), IDESCC(1) and IDESCC(3:8)

Global output arguments: INFO

The remaining arguments are local.

3.3 Distribution Strategy

On entry, the matrix C must be stored in the cyclic two-dimensional block format and its descriptor array IDESCC must contain the details of the distributed matrix. The indices i_C and j_C and the dimensions m and n identify the submatrix C_s. See the F08 Chapter Introduction for further details.

It is assumed that the data has already been correctly distributed, and if this is not the case, then this routine will fail to produce correct results.

3.4 Related Routines

The Library provides many support routines for the generation/distribution and input/output of data. The following routines may be used in conjunction with F08FUFP (PZUNMTR):

Real matrix output: X04BFFP

Complex matrix generation: F01ZVFP

Real vector gather: F01ZPFP

Complex matrix gather: F01WGFP

4 Arguments

1: SIDE — CHARACTER*1 *Global Input*

On entry: indicates how Q or Q^H is to be applied to C_s as follows:

if SIDE = 'L', Q or Q^H is applied to C_s from the left;
if SIDE = 'R', Q or Q^H is applied to C_s from the right.

Constraint: SIDE = 'L' or 'R'.

2: UPLO — CHARACTER*1 *Global Input*

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

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

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

3: TRANS — CHARACTER*1 *Global Input*

On entry: indicates whether Q or Q^H is to be applied to C_s as follows:

if TRANS = 'N', Q is applied to C_s;
if TRANS = 'C', Q^H is applied to C_s.

Constraint: TRANS = 'N' or 'C'.

4: M — INTEGER *Global Input*

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

Constraint: M ≥ 0

5: N — INTEGER *Global Input*

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

Constraint: N ≥ 0

6: A($*$) — COMPLEX*16 array *Local Input*

Note: array A is formally defined as a vector. However, you may find it more convenient to consider A as a two-dimensional array of dimension (IDESCA(9),γ), where

if SIDE = 'L', $\gamma \geq$ numroc(JA+M−1,IDESCA(6),p_c,IDESCA(8),n_p),
if SIDE = 'R', $\gamma \geq$ numroc(JA+N−1,IDESCA(6),p_c,IDESCA(8),n_p).

On entry: details of the vectors which define the elementary reflectors as returned by a call to F08FSFP (PZHETRD).

7: IA — INTEGER *Global Input*

On entry: i_A, the row index of the matrix A that identifies the first row of the submatrix A_s to be factorized.

8: JA — INTEGER *Global Input*

On entry: j_A, the column index of the matrix A that identifies the first column of the submatrix A_s to be factorized.

Constraint: mod(IA−1, IDESCA(5)) = mod(JA−1, IDESCA(6)) = 0.

9: IDESCA($*$) — INTEGER array *Local Input*

Note: the dimension of the array IDESCA must be at least 9.

Distribution: the array elements IDESCA(1) and IDESCA(3),...,IDESCA(8) must be global to the processor grid and the elements IDESCA(2) and IDESCA(9) are local to each processor.

On entry: the description array for the matrix A. This array must contain details of the distribution of the matrix A and the logical processor grid.

IDESCA(1), the descriptor type. For this routine, which uses a cyclic two-dimensional block distribution, IDESCA(1) = 1;
IDESCA(2), the Library context, usually returned by a call to the Library Grid initialisation routine Z01AAFP;
IDESCA(3), the number of rows, m_A, of the matrix A;

IDESCA(4), the number of columns, n_A, of the matrix A;
IDESCA(5), the blocking factor, M_b^A, used to distribute the rows of the matrix A;
IDESCA(6), the blocking factor, N_b^A, used to distribute the columns of the matrix A;
IDESCA(7), the processor row index, s_r^A, over which the first row of the matrix A is distributed;
IDESCA(8), the processor column index, s_c^A, over which the first column of the matrix A is distributed;
IDESCA(9), the leading dimension of the conceptual two-dimensional array A.

Constraints:

IDESCA(1) = 1;
IDESCA(3) $\geq$ 0; IDESCA(4) $\geq$ 0;
IDESCA(5) = IDESCA(6) $\geq$ 1;
$0 \leq$ IDESCA(7) $\leq m_p - 1$; $0 \leq$ IDESCA(8) $\leq n_p - 1$;
if SIDE = 'L', IDESCA(9) $\geq$ max(1, numroc(IA+M−1,IDESCA(5),p_r,IDESCA(7),m_p));
if SIDE = 'R', IDESCA(9) $\geq$ max(1,numroc(IA+N−1,IDESCA(5),p_r,IDESCA(7),m_p)).

10: TAU(∗) — COMPLEX*16 array *Local Input*

Note: the dimension of the array TAU must be at least α where

if SIDE = 'L' and UPLO = 'U', α = numroc(IDESCA(3),IDESCA(6),p_c,IDESCA(8),n_p);
if SIDE = 'L' and UPLO = 'L', α = numroc(JA+M−2,IDESCA(6),p_c,IDESCA(8),n_p);
if SIDE = 'R' and UPLO = 'U', α = numroc(IDESCA(4),IDESCA(6),p_c,IDESCA(8),n_p);
if SIDE = 'R' and UPLO = 'L', α = numroc(JA+N−2,IDESCA(6),p_c,IDESCA(8),n_p).

On entry: details of the elementary reflectors, as returned by a call to F08FSFP (PZHETRD).

11: C(∗) — COMPLEX*16 array *Local Input/Local Output*

Note: array C is formally defined as a vector. However, you may find it more convenient to consider C as a two-dimensional array of dimension (IDESCC(9),γ), where
$\gamma \geq$ numroc(JC +N−1,IDESCC(6),p_c,IDESCC(8),n_p).

On entry: the local part of the matrix C.

On exit: overwritten by QC_s, Q^HC_s, C_sQ or C_sQ^H.

12: IC — INTEGER *Global Input*

On entry: i_C, the row index of matrix C that identifies the first column of the submatrix C_s.

Constraint: 1 $\leq$ IC $\leq$ IDESCC(3) − M + 1.

13: JC — INTEGER *Global Input*

On entry: j_C, the column index of matrix C that identifies the first column of the submatrix C_s.

Constraint: 1 $\leq$ JC $\leq$ IDESCC(4) − N + 1.

14: IDESCC(∗) — INTEGER array *Local Input*

Note: the dimension of the array IDESCC must be at least 9.

Distribution: the array elements IDESCC(1) and IDESCC(3),. . .,IDESCC(8) must be global to the processor grid and the elements IDESCC(2) and IDESCC(9) are local to each processor.

On entry: the description array for the matrix C. This array must contain details of the distribution of the matrix C and the logical processor grid.

IDESCC(1), the descriptor type. For this routine, which uses a cyclic two-dimensional block distribution, IDESCC(1) = 1;
IDESCC(2), the Library context, usually returned by a call to the Library Grid initialisation routine Z01AAFP;

IDESCC(3), the number of rows, m_C, of the matrix C;
IDESCC(4), the number of columns, n_C, of the matrix C;
IDESCC(5), the blocking factor, M_b^C, used to distribute the rows of the matrix C;
IDESCC(6), the blocking factor, N_b^C, used to distribute the columns of the matrix C;
IDESCC(7), the processor row index, s_r^C, over which the first row of the matrix C is distributed;
IDESCC(8), the processor column index, s_c^C, over which the first column of the matrix C is distributed;
IDESCC(9), the leading dimension of the conceptual two-dimensional array C.

Constraints:

IDESCC(1) = 1;
IDESCC(3) $\geq$ 0; IDESCC(4) $\geq$ 0;
IDESCC(5) $\geq$ 1; IDESCC(6) $\geq$ 1;
$0 \leq$ IDESCC(7) $\leq m_p - 1$; $0 \leq$ IDESCC(8) $\leq n_p - 1$;
IDESCC(9) $\geq$ max(1,numroc(IDESCC(3),IDESCC(5),p_r,IDESCC(7),m_p)).

The following constraints are to ensure common alignments in distribution and blocking between the matrices A and C:

if SIDE = 'L', IDESCC(5) = IDESCA(5); $\alpha_{c1} = \alpha_{a1}$; $\beta_{c1} = \beta_{a1}$;
if SIDE = 'R', IDESCC(6) = IDESCA(5); $\alpha_{c2} = \alpha_{a1}$;

where

α_{a1} = mod(i_{aa}−1, IDESCA(5));
α_{c1} = mod(i_{cc}−1, IDESCC(5));
α_{c2} = mod(j_{cc}−1, IDESCC(6));
β_{a1} = indxg2p(i_{aa}, IDESCA(5), p_r, IDESCA(7), m_p);
β_{c1} = indxg2p(i_{cc}, IDESCC(5), p_r, IDESCC(7), m_p);

where

if UPLO = 'U', i_{aa} = IA; j_{aa} = JA +1; i_{cc} = IC; j_{cc} = JC;
if UPLO = 'L', i_{aa} = IA +1; j_{aa} = JA;
    if SIDE = 'L', i_{cc} = IC +1; j_{cc} = JC;
    if SIDE = 'R', i_{cc} = IC; j_{cc} = JC+1.

15: WORK(∗) — COMPLEX*16 array *Local Workspace/Local Output*

Note: the dimension of WORK must be at least max(1,LWORK). The minimum value of LWORK required to successfully call this routine can be obtained by setting LWORK = −1. The required size is returned in the real part of array element WORK(1).

On exit: the real part of WORK(1) contains the minimum dimension of the array WORK required to successfully complete the task.

16: LWORK — INTEGER *Local Input*

On entry: either −1 (see WORK) or the dimension of the array WORK required to successfully complete the task. If LWORK is set to −1 on entry this routine simply performs some initial error checking and then, if these checks are successful, calculates the minimum size of LWORK required.

Constraints:

if SIDE = 'L',
    m_1 = M−1; n_1 = N;
    LWORK = max($N_b^A * (N_b^A - 1)/2$,$(\mu_{c1} + \mu_{c2}) * N_b^A$) + $N_b^A * N_b^A$ or LWORK = −1;

if SIDE = 'R',

m_1 = M; n_1 = N−1;

LWORK = $\max(N_b^A * (N_b^A - 1)/2, \gamma_3 * N_b^A) + N_b^A * N_b^A$ or LWORK = −1;

where

i_{aa}, i_{cc}, α_{a1}, α_{c1}, α_{c2} and β_{c1} are defined in the description of the argument IDESCC(9);
N_b^A = IDESCA(6);
α_{a2} = mod(j_{aa}−1, IDESCA(6));
μ_{a1} = numroc($n_1 + \beta_{a1}$,IDESCA(5),p_r,β_{a1},m_p);
β_{c2} = indxg2p(j_{cc},IDESCC(6),p_c,IDESCC(8),n_p);
μ_{c1} = numroc($m_1 + \alpha_{c1}$,IDESCC(5),p_r,β_{c1},m_p);
μ_{c2} = numroc($n_1 + \alpha_{c2}$,IDESCC(6),p_c,β_{c2},n_p);
δ = iclm(m_p,n_p)/n_p;
γ_1 = numroc($n_1 + \alpha_{c2}$,IDESCC(6),0,0,n_p);
γ_2 = numroc(γ_1,IDESCC(6),0,0,δ);
γ_3 = μ_{c2}+max($\mu_{a1} + \gamma_2$,μ_{c1}).

17: INFO — INTEGER *Global Output*

The NAG Parallel Library provides a mechanism, via the routine Z02EAFP, to reduce the amount of parameter validation performed by this routine. For a full description refer to the Z02 Chapter Introduction.

On exit: INFO = 0 (or −9999 if reduced error checking is enabled) unless the routine detects an error (see Section 5).

5 Errors and Warnings

If INFO < 0 an explanatory message is output and control returned to the calling program.

INFO = $-i$

On entry, one of the arguments was invalid:

if the kth argument is a scalar INFO = $-k$;
if the kth argument is an array and its jth element is invalid, INFO = $-(100\times k + j)$.

This error occurred either because a global argument did not have the same value on all logical processors, or because its value on one or more processors was incorrect.

6 Further Comments

6.1 Algorithmic Detail

See Anderson, *et al.* [1], Golub and Van Loan [2] and Blackford *et al.* [3].

6.2 Parallelism Detail

The Level-3 BLAS operations are carried out in parallel within the routine.

7 References

[1] Anderson E, Bai Z, Bischof C, Blackford S, Demmel J, Dongarra J J, Du Croz J J, Greenbaum A, Hammarling S, McKenney A and Sorensen D (1999) *LAPACK Users' Guide* (3rd Edition) SIAM, Philadelphia

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

[3] Blackford L S, Choi J, Cleary A, D'Azevedo E, Demmel J, Dhillon I, Dongarra J, Hammarling S, Henry G, Petitet A, Stanley K, Walker D and Whaley R C (1997) ScaLAPACK Users' Guide *SIAM* 3600 University City Science Center, Philadelpia, PA 19104-2688, USA. URL: http://www.netlib.org/scalapack/slug/scalapack_slug.html

8 Example

The example program illustrates the the computation of eigenvectors of a 9 by 9 complex Hermitian matrix A_s. The (i,j)th element of this matrix A_s is CMPLX(max(i,j), min(i,j)), CMPLX(max(i,j), $-$min(i,j)) or CMPLX(max(i,j), 0) if $i > j$, $i < j$ or $i = j$, respectively. This matrix is generated using the routine F01ZVFP.

The complex matrix A_s is reduced to the real tridiagonal form T using F08FSFP (PZHETRD). The argument UPLO is set to 'L' and hence only the lower triangular part of the matrix A_s is used.

The diagonal vector d and the off-diagonal vector e are gathered to every logical processor using the routine F01ZPFP. The vector d (denoted locally by the array DL) and the the vector e (denoted locally by the array EL) are printed on the root processor.

The eigenvalues of the real tridiagonal matrix T are computed using the routine F08JJFP (PDSTEBZ). Routine F08JXFP (PZSTEIN) gives the eigenvectors of the real tridiagonal matrix.

Finally, the eigenvectors of the original complex matrix A_s are obtained by calling the routine F08FUFP (PZUNMTR). The first two computed eigenvectors are gathered to the root processor and are printed.

8.1 Example Text

```
*     F08FUFP Example Program Text
*     NAG Parallel Library Release 3. NAG Copyright 1999.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          N
      PARAMETER        (N=9)
      INTEGER          NA
      PARAMETER        (NA=20)
      INTEGER          NB
      PARAMETER        (NB=2)
      INTEGER          MP, NP
      PARAMETER        (MP=2,NP=2)
      INTEGER          LDA, TDA
      PARAMETER        (LDA=NA/MP+NB,TDA=NA/NP+NB)
      INTEGER          LWORK, LRWORK, LIWORK
      PARAMETER        (LWORK=100,LRWORK=100,LIWORK=50)
*     .. Local Scalars ..
      DOUBLE PRECISION ORFAC
      INTEGER          I, IA, ICNTXT, ID, IFAIL, INFO, IZ, J, JA, JD,
     +                 JZ, M, NSPLIT, MPROC, NPROC
      LOGICAL          ROOT
      CHARACTER        SIDE, TRANS, UPLO
*     .. Local Arrays ..
      COMPLEX*16       A(LDA,TDA), TAU(TDA), WORK(LWORK), X(N,2),
     +                 Z(LDA,TDA)
      DOUBLE PRECISION D(TDA), DL(N), E(TDA), EL(N), GAP(MP*NP),
     +                 RWORK(LRWORK), W(N)
      INTEGER          IBLOCK(N), ICLUSTR(2*MP*NP), IDESCA(9),
     +                 IDESCZ(9), ISPLIT(N), IWORK(LIWORK), JFAIL(NA)
*     .. External Functions ..
      LOGICAL          Z01ACFP
      EXTERNAL         Z01ACFP
```

```
*     .. External Subroutines ..
      EXTERNAL         F01WGFP, F01ZPFP, F01ZVFP, F08FSFP, F08FUFP,
     +                 F08JJFP, F08JXFP, GMATA, Z01AAFP, Z01ABFP
*     .. Executable Statements ..
      ROOT = Z01ACFP()
      IF (ROOT) THEN
         WRITE (NOUT,*) 'F08FUFP Example Program Results'
         WRITE (NOUT,*)
      END IF
*
      MPROC = MP
      NPROC = NP
      ID = 0
      JD = 0
      IFAIL = 0
      CALL Z01AAFP(ICNTXT,MPROC,NPROC,IFAIL)
*
*     Set the starting address and array descriptor for A
*
      IA = 1
      JA = 1
      IDESCA(1) = 1
      IDESCA(2) = ICNTXT
      IDESCA(3) = NA
      IDESCA(4) = NA
      IDESCA(5) = NB
      IDESCA(6) = NB
      IDESCA(7) = 0
      IDESCA(8) = 0
      IDESCA(9) = LDA
*
*     Generate the matrix A
*
      IFAIL = 0
      CALL F01ZVFP(GMATA,N,N,A,IA,JA,IDESCA,IFAIL)
*
*     Reduce A to tridiagonal form
*
      UPLO = 'L'
      CALL F08FSFP(UPLO,N,A,IA,JA,IDESCA,D,E,TAU,WORK,LWORK,INFO)
*
*     Gather the diagonal D to each logical processor
*
      IFAIL = 0
      CALL F01ZPFP(N,IA,JA,IDESCA,D,DL,RWORK,LRWORK,IFAIL)
*
*     Gather the off-diagonal E to each logical processor
*
      IFAIL = 0
      CALL F01ZPFP(N,IA,JA,IDESCA,E,EL,RWORK,LRWORK,IFAIL)
*
*     Print the diagonal and off-diagonal elements of the tridiagonal
*     matrix on the root processor
*
      IF (ROOT) THEN
         WRITE (NOUT,*) 'Elements of the tridiagonal matrix T'
         WRITE (NOUT,*)
         IF (UPLO.EQ.'U') THEN
```

```
            WRITE (NOUT,'(3X,''I'',12X,''DL(I)'',9X,''EL(I)''/)')
            WRITE (NOUT,'(1X,I3,5X,2(F12.4,2X))') 1, DL(1)
            DO 20 I = 2, N
               WRITE (NOUT,'(1X,I3,5X,2(F12.4,2X))') I, DL(I), EL(I)
   20       CONTINUE
         ELSE IF (UPLO.EQ.'L') THEN
            WRITE (NOUT,'(3X,''I'',12X,''DL(I)'',9X,''EL(I)''/)')
            DO 40 I = 1, N - 1
               WRITE (NOUT,'(1X,I3,5X,2(F12.4,2X))') I, DL(I), EL(I)
   40       CONTINUE
            WRITE (NOUT,'(1X,I3,5X,2(F12.4,2X))') N, DL(N)
         END IF
      END IF
*
*     Compute the eigenvalues of T (and of A)
*
      IF (UPLO.EQ.'L') THEN
         CALL F08JJFP(ICNTXT,'A','B',N,0.0D0,0.0D0,0,0,-1.0D0,DL,EL,M,
     +                NSPLIT,W,IBLOCK,ISPLIT,RWORK,LRWORK,IWORK,LIWORK,
     +                INFO)
      ELSE IF (UPLO.EQ.'U') THEN
         CALL F08JJFP(ICNTXT,'A','B',N,0.0D0,0.0D0,0,0,-1.0D0,DL,EL(2),
     +                M,NSPLIT,W,IBLOCK,ISPLIT,RWORK,LRWORK,IWORK,
     +                LIWORK,INFO)
      END IF
*
*     Print the computed eigenvalues
*
      IF (ROOT) THEN
         WRITE (NOUT,*) 'Eigenvalues'
         WRITE (NOUT,*)
         DO 60 I = 1, N
            WRITE (NOUT,'(1X,I3,5X,F12.4)') I, W(I)
   60    CONTINUE
      END IF
*
*     Set the starting address and array descriptor for the matrix of
*     eigenvectors, Z
*
      IZ = 1
      JZ = 1
      IDESCZ(1) = 1
      IDESCZ(2) = ICNTXT
      IDESCZ(3) = NA
      IDESCZ(4) = NA
      IDESCZ(5) = NB
      IDESCZ(6) = NB
      IDESCZ(7) = 0
      IDESCZ(8) = 0
      IDESCZ(9) = LDA
*
*     Compute the eigenvectors of T
*
      ORFAC = 1.0D-06
      IF (UPLO.EQ.'L') THEN
         CALL F08JXFP(N,DL,EL,M,W,IBLOCK,ISPLIT,ORFAC,Z,IZ,JZ,IDESCZ,
     +                RWORK,LRWORK,IWORK,LIWORK,JFAIL,ICLUSTR,GAP,INFO)
      ELSE IF (UPLO.EQ.'U') THEN
```

```
         CALL F08JXFP(N,DL,EL(2),M,W,IBLOCK,ISPLIT,ORFAC,Z,IZ,JZ,IDESCZ,
     +                RWORK,LRWORK,IWORK,LIWORK,JFAIL,ICLUSTR,GAP,INFO)
      END IF
*
*     Transform the eigenvectors of T to those of A
*
      SIDE = 'L'
      TRANS = 'N'
      CALL F08FUFP(SIDE,UPLO,TRANS,N,N,A,IA,JA,IDESCA,TAU,Z,IZ,JZ,
     +             IDESCZ,WORK,LWORK,INFO)
*
*     Gather the first two eigenvectors to the (root) processor (0,0)
*     as the matrix X
*
      IFAIL = 0
      CALL F01WGFP(N,4,Z,IZ,JZ,IDESCZ,ID,JD,X,N,WORK,LWORK,IFAIL)
*
*     Print these two eigenvectors
*
      IF (ROOT) THEN
         WRITE (NOUT,*) 'The first two eigenvectors'
         WRITE (NOUT,*)
         DO 80 I = 1, N
            WRITE (NOUT,
     +        '(1X,2(''(''',D12.5,1X,''',''',1X,D12.5,''')''',2X) )') (X(I,J)
     +        ,J=1,2)
   80    CONTINUE
      END IF
*
      IFAIL = 0
      CALL Z01ABFP(ICNTXT,'N',IFAIL)
*
      STOP
      END
*
      SUBROUTINE GMATA(I1,I2,J1,J2,AL,LDAL)
*
*     GMATA generates the block A(I1: I2, J1: J2) of the lower
*     triangular part of the complex Hermitian matrix A such that
*
*        a(i,j) = cmplx( max(i,j),  min(i,j) )  if i .GT. j  ;
*        a(i,j) = cmplx( max(i,j), -min(i,j) )  if i .LT. j  ;
*        a(i,j) = cmplx( max(i,j),  0        )  if i .EQ. j  .
*
*     in the array AL.
*
*     .. Scalar Arguments ..
      INTEGER          I1, I2, J1, J2, LDAL
*     .. Array Arguments ..
      COMPLEX*16       AL(LDAL,*)
*     .. Local Scalars ..
      INTEGER          I, J, K, L
*     .. Intrinsic Functions ..
      INTRINSIC        CMPLX, DBLE, MAX, MIN
*     .. Executable Statements ..
      L = 1
      DO 40 J = J1, J2
         K = 1
```

```
         DO 20 I = I1, I2
            IF (I.GT.J) THEN
               AL(K,L) = CMPLX(DBLE(MAX(I,J)),+DBLE(MIN(I,J)))
            ELSE IF (I.EQ.J) THEN
               AL(K,L) = CMPLX(DBLE(MAX(I,J)),0.0D0)
            END IF
            K = K + 1
   20    CONTINUE
         L = L + 1
   40 CONTINUE
      RETURN
      END
```

8.2 Example Data

None.

8.3 Example Results

```
F08FUFP Example Program Results

Elements of the tridiagonal matrix T

  I            DL(I)         EL(I)

  1           1.0000      -17.0880
  2          50.0411       26.6938
  3          -6.5579       -7.7489
  4          -0.8542       -7.0696
  5           1.0154        4.2385
  6          -0.1027       -3.0785
  7           0.2649       -2.0244
  8           0.0992        1.1771
  9           0.0942
Eigenvalues

  1         -21.8293
  2          -7.5810
  3          -3.4375
  4          -1.3673
  5          -0.0854
  6           1.3253
  7           3.7903
  8           9.4385
  9          64.7465
The first two eigenvectors

(-0.27535D+00 ,  0.00000D+00)  ( 0.30190D+00 ,  0.00000D+00)
(-0.28206D+00 ,  0.53372D-01)  ( 0.25788D+00 , -0.13278D+00)
(-0.25422D+00 ,  0.13954D+00)  ( 0.57464D-01 , -0.26498D+00)
(-0.16995D+00 ,  0.23627D+00)  (-0.24224D+00 , -0.17396D+00)
(-0.17737D-01 ,  0.29908D+00)  (-0.30075D+00 ,  0.19306D+00)
( 0.17950D+00 ,  0.26852D+00)  ( 0.92859D-01 ,  0.37297D+00)
( 0.34524D+00 ,  0.10268D+00)  ( 0.35988D+00 , -0.39213D-01)
( 0.36451D+00 , -0.16794D+00)  (-0.77765D-01 , -0.33814D+00)
( 0.16033D+00 , -0.40296D+00)  (-0.33628D+00 ,  0.14932D+00)
```