

F08FGFP (PDORMTR)

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

F08FGFP (PDORMTR) multiplies an m by n real matrix C_s by a real orthogonal 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, $A_s = QTQ^T$ with A_s real symmetric, Q orthogonal and T real symmetric tridiagonal, as computed by the routine F08FEFP (PDSYTRD).

This routine may be used to form one of the products

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

without explicitly forming Q . The result is overwritten on C_s .

F08FGFP (PDORMTR) can be used to compute the eigenvectors of A following the computation of the eigenvectors of T by F08JKFP (PDSSTEIN).

2 Specification

```

SUBROUTINE F08FGFP(SIDE, UPLO, TRANS, M, N, A, IA, JA, IDESCA,
1          TAU, C, IC, JC, IDESCC, WORK, LWORK, INFO)
ENTRY      PDORMTR(SIDE, UPLO, TRANS, M, N, A, IA, JA, IDESCA,
1          TAU, C, IC, JC, IDESCC, WORK, LWORK, INFO)
DOUBLE PRECISION 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($\hat{\ell}, L_b^X, p, s^X, \ell_p$) – a function which gives the **number of rows or columns** 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 F08FGFP (PDORMTR):

Real matrix output: X04BFFP
Real matrix generation: F01ZQFP
Real vector scatter: F01ZPFP
Real matrix gather: F01WAFP

4 Arguments

1: SIDE — CHARACTER*1

Global Input

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

- if SIDE = 'L', Q or Q^T is applied to C_s from the left;
- if SIDE = 'R', Q or Q^T 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^T is to be applied to C_s as follows:
 if TRANS = 'N', Q is applied to C_s ;
 if TRANS = 'T', Q^T is applied to C_s .
Constraint: TRANS = 'N' or 'T'.
- 4:** M — INTEGER *Global Input*
On entry: m , the number of rows of the matrix C_s .
Constraint: $M \geq 0$
- 5:** N — INTEGER *Global Input*
On entry: n , the number of columns of the matrix C_s .
Constraint: $N \geq 0$
- 6:** A(*) — DOUBLE PRECISION 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', $\gamma \geq \text{numroc}(\text{JA}+M-1, \text{IDESCA}(6), p_c, \text{IDESCA}(8), n_p)$
 if SIDE = 'R', $\gamma \geq \text{numroc}(\text{JA}+N-1, \text{IDESCA}(6), p_c, \text{IDESCA}(8), n_p)$.
On entry: details of the vectors which define the elementary reflectors as returned by a call to F08FEFP (PDSYTRD).
- 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: $\text{mod}(\text{IA}-1, \text{IDESCA}(5)) = \text{mod}(\text{JA}-1, \text{IDESCA}(6)) = 0$.
- 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 Library 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(*) — DOUBLE PRECISION array *Local Input*

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

- if SIDE = 'L' and UPLO = 'U', $\alpha =$ numroc(IDESCA(3),IDESCA(6), p_c ,IDESCA(8), n_p);
- if SIDE = 'L' and UPLO = 'L', $\alpha =$ numroc(JA+M-2,IDESCA(6), p_c ,IDESCA(8), n_p);
- if SIDE = 'R' and UPLO = 'U', $\alpha =$ numroc(IDESCA(4),IDESCA(6), p_c ,IDESCA(8), n_p);
- if SIDE = 'R' and UPLO = 'L' $\alpha =$ 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 F08FEFP (PDSYTRD).

11: C(*) — DOUBLE PRECISION 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 , $Q^T C_s$, $C_s Q$ or $C_s Q^T$.

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(2) = IDESCA(2);
 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 $\max(1, \text{numroc}(\text{IDESSC}(3), \text{IDESSC}(5), p_r, \text{IDESSC}(7), m_p))$.
 if UPLO = 'U', $i_{aa} = \text{IA}$; $j_{aa} = \text{JA} + 1$; $i_{cc} = \text{IC}$; $j_{cc} = \text{JC}$;

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

$\alpha_{a1} = \text{mod}(i_{aa} - 1, \text{IDESSC}(5))$;
 $\alpha_{c1} = \text{mod}(i_{cc} - 1, \text{IDESSC}(5))$;
 $\alpha_{c2} = \text{mod}(j_{cc} - 1, \text{IDESSC}(6))$;
 $\beta_{a1} = \text{indxg2p}(i_{aa}, \text{IDESSC}(5), p_r, \text{IDESSC}(7), m_p)$;
 $\beta_{c1} = \text{indxg2p}(i_{cc}, \text{IDESSC}(5), p_r, \text{IDESSC}(7), m_p)$;

where

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

15: WORK(*) — DOUBLE PRECISION array*Local Workspace/Local Output*

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

On exit: 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',
 $m_1 = M - 1$; $n_1 = N$;
 $\text{LWORK} = \max(N_b^A * (N_b^A - 1) / 2, (\mu_{c1} + \mu_{c2}) * N_b^A) + N_b^A * N_b^A$ or LWORK = -1;

if SIDE = 'R',

$$m_1 = M; n_1 = N-1;$$

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

where

$i_{aa}, i_{cc}, \alpha_{a1}, \alpha_{c1}, \alpha_{c2}$ and β_{c1} are defined in the description of the argument IDESCC(9);

$$N_b^A = \text{IDESCA}(6);$$

$$\alpha_{a2} = \text{mod}(j_{aa}-1, \text{IDESCA}(6));$$

$$\mu_{a1} = \text{numroc}(n_1 + \beta_{a1}, \text{IDESCA}(5), p_r, \beta_{a1}, m_p);$$

$$\beta_{c2} = \text{indxg2p}(j_{cc}, \text{IDESCC}(8), n_p);$$

$$\mu_{c1} = \text{numroc}(m_1 + \alpha_{c1}, \text{IDESCC}(5), p_r, \beta_{c1}, m_p);$$

$$\mu_{c2} = \text{numroc}(n_1 + \alpha_{c2}, \text{IDESCC}(6), p_c, \beta_{c2}, n_p);$$

$$\delta = \text{iclm}(m_p, n_p)/n_p;$$

$$\gamma_1 = \text{numroc}(n_1 + \alpha_{c2}, \text{IDESCC}(6), 0, 0, n_p);$$

$$\gamma_2 = \text{numroc}(\gamma_1, \text{IDESCC}(6), 0, 0, \delta);$$

$$\gamma_3 = \mu_{c2} + \max(\mu_{a1} + \gamma_2, \mu_{c1}).$$

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 k th argument is a scalar INFO = - k ;

if the k th argument is an array and its j th element is invalid, INFO = -(100× 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, Philadelphia, 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 10 by 10 real symmetric matrix A_s . The (i,j) th element of this matrix A_s is $\max(i,j)$. This matrix is generated using routine F01ZQFP.

The real matrix A is reduced to the real tridiagonal form T using F08FEFP (PDSYTRD). The argument UPLO is set to ‘U’ and hence only the upper 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 F08JKFP (PDSTEIN) gives the eigenvectors of the real tridiagonal matrix.

Finally, the eigenvectors of the original matrix A_s are obtained by calling routine F08FGFP (PDORMTR). The first four computed eigenvectors are gathered to the root processor and printed.

8.1 Example Text

```
*      F08FGFP Example Program Text
*      NAG Parallel Library Release 3. NAG Copyright 1999.
*      .. Parameters ..
INTEGER          NOUT
PARAMETER       (NOUT=6)
INTEGER          N
PARAMETER       (N=10)
INTEGER          NA
PARAMETER       (NA=20)
INTEGER          MP, NP
PARAMETER       (MP=2, NP=2)
INTEGER          NB
PARAMETER       (NB=2)
INTEGER          LDA, TDA
PARAMETER       (LDA=NA/MP+NB, TDA=NA/NP+NB)
INTEGER          LWORK, LIWORK
PARAMETER       (LWORK=100, LIWORK=100)
*      .. Local Scalars ..
DOUBLE PRECISION ORFAC
INTEGER          I, IA, ICNTXT, IFAIL, INFO, IZ, J, JA, JZ, M,
+              NSPLIT, MPROC, NPROC
LOGICAL         ROOT
CHARACTER       SIDE, TRANS, UPLO
*      .. Local Arrays ..
DOUBLE PRECISION A(LDA, TDA), D(TDA), DL(N), E(TDA), EL(N),
+              GAP(MP*NP), TAU(TDA), W(NA), WORK(LWORK), X(N,4),
+              Z(LDA, TDA)
INTEGER          IBLOCK(N), ICLUSTR(MP*NP*2), IDESCA(9),
+              IDESCZ(9), ISPLIT(N), IWORK(LIWORK), JFAIL(NA)
*      .. External Functions ..
LOGICAL         Z01ACFP
EXTERNAL        Z01ACFP
*      .. External Subroutines ..
EXTERNAL        F01WAFP, F01ZPFP, F01ZQFP, F08FEFP, F08FGFP,
+              F08JJFP, F08JKFP, GMATA, Z01AAFP, Z01ABFP
```

```

*      .. Executable Statements ..
ROOT = Z01ACFP()
IF (ROOT) THEN
    WRITE (NOUT,*) 'F08FGFP Example Program Results'
    WRITE (NOUT,*)
END IF
*
MPROC = MP
NPROC = NP
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 F01ZQFP(GMATA,N,N,A,IA,JA,IDESCA,IFAIL)
*
*      Reduce A to tridiagonal form
*
UPLO = 'U'
CALL F08FEFP(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,WORK,LWORK,IFAIL)
*
*      Gather the off-diagonal E to each logical processor
*
IFAIL = 0
CALL F01ZPFP(N,IA,JA,IDESCA,E,EL,WORK,LWORK,IFAIL)
*
*      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,WORK,LWORK,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,WORK,LWORK,IWORK,LIWORK,
+             INFO)
END IF
*

```



```

*      Print the computed eigenvalues
*
      IF (ROOT) THEN
        WRITE (NOUT,*) 'Eigenvalues'
        WRITE (NOUT,*)
        DO 20 I = 1, N
          WRITE (NOUT,'(1X,I3,5X,F12.4)') I, W(I)
20      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 F08JKFP(N,DL,EL,M,W,IBLOCK,ISPLIT,ORFAC,Z,IZ,JZ,DESCZ,
+          WORK,LWORK,IWORK,LIWORK,JFAIL,ICLUSTR,GAP,INFO)
      ELSE IF (UPLO.EQ.'U') THEN
        CALL F08JKFP(N,DL,EL(2),M,W,IBLOCK,ISPLIT,ORFAC,Z,IZ,JZ,DESCZ,
+          WORK,LWORK,IWORK,LIWORK,JFAIL,ICLUSTR,GAP,INFO)
      END IF
*
*      Transform the eigenvectors of T to those of A
*
      SIDE = 'L'
      TRANS = 'N'
      CALL F08FGFP(SIDE,UPLO,TRANS,N,N,A,IA,JA,DESCA,TAU,Z,IZ,JZ,
+        IDESCZ,WORK,LWORK,INFO)
*
*      Gather the first four eigenvectors to the (root) processor (0,0)
*      as the matrix X
*
      IFAIL = 0
      CALL F01WAFP(N,4,Z,IZ,JZ,DESCZ,0,0,X,N,WORK,LWORK,IFAIL)
*
*      Print these four eigenvectors
*
      IF (ROOT) THEN
        WRITE (NOUT,*) 'The first four eigenvectors'
        WRITE (NOUT,*)
        DO 40 I = 1, N
          WRITE (NOUT,'(1X,4(D15.5,1X))') (X(I,J),J=1,4)
40      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 upper
  triangular part of the symmetric matrix A such
  that
*
   $a(i,j) = \max(i,j)$ 
*
  in the array AL.
*
  .. Scalar Arguments ..
  INTEGER          I1, I2, J1, J2, LDAL
*
  .. Array Arguments ..
  DOUBLE PRECISION AL(LDAL,*)
*
  .. Local Scalars ..
  INTEGER          I, J, K, L
*
  .. Intrinsic Functions ..
  INTRINSIC        MAX
*
  .. Executable Statements ..
  L = 1
  DO 40 J = J1, J2
    K = 1
    DO 20 I = I1, I2
      IF (I.LE.J) THEN
        AL(K,L) = MAX(I,J)
      END IF
      K = K + 1
    20  CONTINUE
    L = L + 1
  40  CONTINUE
*
  RETURN
*
  END

```

8.2 Example Data

None.

8.3 Example Results

F08FGFP Example Program Results

Eigenvalues

1	-12.6794
2	-2.7427
3	-1.2357
4	-0.7302
5	-0.5024
6	-0.3829
7	-0.3153
8	-0.2765
9	-0.2563
10	74.1216

The first four eigenvectors

0.46896D+00	0.43190D+00	0.40177D+00	0.36393D+00
0.43197D+00	0.27443D+00	0.76647D-01	-0.13445D+00
0.36092D+00	0.16900D-01	-0.31050D+00	-0.44871D+00
0.26140D+00	-0.24679D+00	-0.44639D+00	-0.14848D+00
0.14126D+00	-0.42050D+00	-0.22104D+00	0.35508D+00
0.99868D-02	-0.44089D+00	0.18318D+00	0.37238D+00
-0.12208D+00	-0.30054D+00	0.43916D+00	-0.12027D+00
-0.24451D+00	-0.50605D-01	0.33977D+00	-0.44822D+00
-0.34767D+00	0.21778D+00	-0.34579D-01	-0.16236D+00
-0.42340D+00	0.40676D+00	-0.38094D+00	0.34585D+00
