

# F07HRFP (PZPBTRF)

## 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

F07HRFP (PZPBTRF) computes a permuted Cholesky factorization of an  $n$  by  $n$  complex banded Hermitian positive-definite matrix  $A_s$ , where  $A_s$  is a submatrix of a larger  $n$  by  $n_A$  matrix  $A$ , i.e.,

$$A_s(1:n, 1:n) \equiv A(1:n, j_A:j_A+n-1).$$

**Note:** if  $j_A = 1$  and  $n = n_A$ , then  $A_s = A$ .

The factorization may be formed either as  $A_s = PU^HUP^T$  or  $A_s = PLL^HP^T$ , where  $P$  is a permutation matrix,  $U$  is a banded upper triangular matrix with the same number of super-diagonals as  $A_s$  and  $L$  is banded lower triangular with the same number of sub-diagonals as  $A_s$ . This routine is best suited to narrow band matrices.

### 2 Specification

```

SUBROUTINE F07HRFP(UPLO, N, BW, A, JA, IDESCA, AF, LAF, WORK,
1                LWORK, INFO)
ENTRY          PZPBTRF(UPLO, N, BW, A, JA, IDESCA, AF, LAF, WORK,
1                LWORK, INFO)
INTEGER       N, BW, JA, IDESCA(*), LAF, LWORK, INFO
COMPLEX*16    A(*), AF(*), WORK(*)
CHARACTER*1   UPLO

```

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:

- $b_w$  – the half band width of the banded Hermitian matrix;
- $m_p$  – the number of rows in the Library Grid, for this routine  $m_p = 1$  or  $m_p = p$ ;
- $n_p$  – the number of columns in the Library Grid, for this routine  $n_p = 1$  or  $n_p = p$ .
- $p$  –  $m_p \times n_p$ , the total number of processors in the Library Grid.
- $M_b^X$  – 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  $X$ .

#### 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: UPLO, N, BW, JA, some elements of IDESCA (see Section 4 for a description of IDESCA).

Global output arguments: INFO.

The remaining arguments are local.

### 3.3 Distribution Strategy

The matrix  $A$  should be distributed over a one-dimensional array of processors, assuming a column block distribution, and is stored in compact format (see the F07 Chapter Introduction). **It is important that**  $p \times N_b^A \geq \text{mod}(j_A - 1, N_b^A) + n$  and  $N_b^A \geq 2 \times b_w$ . The first restriction states that the mapping for matrices must be blocked, reflecting the nature of the **divide and conquer algorithm** as a task-parallel algorithm. The second restriction is to ensure a better efficiency of this algorithm (see Section 6.2). This means that no processor may store more than one block of the matrix.

### 3.4 Related Routines

The Library provides many support routines for the generation/distribution and input/output of data in column or row block form. The following routines may be used in conjunction with F07HRFP (PZPBTRF):

Complex matrix generation:      column block distribution: F01YWFP

## 4 Arguments

1: UPLO — CHARACTER\*1 *Global Input*

*On entry:* indicates whether the upper or lower triangular part of  $A_s$  is stored and how  $A_s$  is factorized, as follows:

if UPLO = 'U', then the upper triangular part of  $A_s$  is stored and  $A_s$  is factorized as  $PU^HUP^T$ , where  $U$  is upper triangular and  $P$  is a permutation matrix;

if UPLO = 'L', then the lower triangular part of  $A_s$  is stored and  $A_s$  is factorized as  $PLL^HP^T$ , where  $L$  is lower triangular and  $P$  is a permutation matrix.

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

2: N — INTEGER *Global Input*

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

*Constraint:*  $N \geq 0$ .

3: BW — INTEGER *Global Input*

*On entry:*  $b_w$ , the number of super-diagonals or sub-diagonals of the matrix  $A_s$ .

*Constraints:*  $1 \leq BW \leq N-1$ .

**Note:** F07HRFP is suitable for the factorization of 'narrow' banded matrices. Hence, for large  $N$ ,  $BW$  should be much smaller than  $N-1$ .

4: A(\*) — COMPLEX\*16 array *Local Input/Local Output*

**Note:** the 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  $(\ell_A, \gamma)$  where  $\ell_A = \text{IDESCA}(9)$  if  $\text{IDESCA}(1) = 1$ , or  $\ell_A = \text{IDESCA}(6)$  if  $\text{IDESCA}(1) = 501$ ; and  $\gamma \geq N_b^A$ .

*On entry:* the local part of the distributed matrix  $A$  which may contain the upper part if UPLO = 'U' or the lower part if UPLO = 'L'. See the F07 Chapter Introduction for further detail.

*On exit:* information containing the factors of the matrix  $A_s$ . See Section 6.2.

5: JA — INTEGER *Global Input*

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

*Constraints:*  $1 \leq JA \leq n_A - N + 1$ .

**6:** IDESCA(\*) — INTEGER array *Local Input*

**Note:** the dimension of the array IDESCA must be at least 6 when IDESCA(1) = 501 and must be at least 9 when IDESCA(1) = 1.

*Distribution:* if IDESCA(1) = 1, the array elements IDESCA(3:8) must be global to the processor grid. If IDESCA(1) = 501, then only the array elements IDESCA(3:5) must be global. In all cases IDESCA(2) is 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.

If IDESCA(1) = 1, then  $p = 1 \times n_p$  and:

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  $A$  (since the compact storage is used here, IDESCA(5) = BW+1);

IDESCA(6), the blocking factor,  $N_b^A$ , used to distribute the columns of the matrix  $A$ ;

IDESCA(7), the processor row index over which the first row of the matrix  $A$  is distributed (since the logical grid is one-dimensional, IDESCA(7) = 0);

IDESCA(8), the processor column index over which the first column of the matrix  $A$  is distributed;

IDESCA(9), the leading dimension of the conceptual two-dimensional array  $A$ .

If IDESCA(1) = 501, then  $p = 1 \times n_p$  or  $p = m_p \times 1$ , and:

IDESCA(2), the Library context, usually returned by a call to the Library Grid initialisation routine Z01AAFP;

IDESCA(3), the size  $n_A$ , of the matrix  $A$ ;

IDESCA(4), the blocking factor,  $N_b^A$ , used to distribute the matrix  $A$ ;

IDESCA(5), the processor column index over which the first element of the matrix  $A$  is distributed;

IDESCA(6), the leading dimension of the conceptual two-dimensional array  $A$ .

IDESCA(7:9) are not referenced.

*Suggested value:* IDESCA(1) = 501 and  $p = 1 \times n_p$ .

*Constraints:*

IDESCA(1) = 1 or 501;

if IDESCA(1) = 1, then  $p = 1 \times n_p$ ;

if IDESCA(1) = 501; then  $p = m_p \times 1$  or  $p = 1 \times n_p$ ;

if IDESCA(1) = 501, then

$1 \leq \text{IDESCA}(3) \leq N + \text{JA} - 1$ ;

$\text{IDESCA}(4) \geq 2 \times \text{BW}$  and  $p \times \text{IDESCA}(4) \geq \text{mod}(\text{JA} - 1, \text{IDESCA}(4)) + N$ ;

$\text{IDESCA}(5) \geq 0$ ;

$\text{IDESCA}(6) \geq \text{BW} + 1$ ;

if IDESCA(1) = 1, then

$1 \leq \text{IDESCA}(4) \leq N + \text{JA} - 1$ ;

$\text{IDESCA}(6) \geq 2 \times \text{BW}$  and  $p \times \text{IDESCA}(6) \geq \text{mod}(\text{JA} - 1, \text{IDESCA}(6)) + N$ ;

$\text{IDESCA}(8) \geq 0$ ;

$\text{IDESCA}(9) \geq \text{BW} + 1$ .

**7:** AF(\*) — COMPLEX\*16 array*Local Output*

*On exit:* the auxiliary fill-in space. Fill-in is created and stored during the factorization. If LAF is not large enough, after an unsuccessful exit, INT(AF(1)) will contain the minimum acceptable size of AF.

8: LAF — INTEGER *Local Input*

*On entry:* the dimension of the array AF .

*Constraint:*  $LAF \geq (N_b^A + 2 \times BW) \times BW$ .

9: WORK(\*) — COMPLEX\*16 array *Local Workspace*

**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.

10: 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:*

either  $LWORK = -1$ ,  
or  $LWORK \geq BW \times BW$ .

11: 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  $INFO \neq 0$  explanatory error messages are output from the root processor (or processor  $\{0, 0\}$  when the root processor is not available) on the current error message unit (as defined by X04AAF).

$INFO = -i$

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,  $INFO = -(100 \times k + j)$ .

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.

$INFO > 0$

If  $INFO = k \leq p$  the submatrix stored and factored locally on processor  $\{0, k\}$  or  $\{k, 0\}$  was not positive definite, and the factorization was not completed. If  $INFO = k \geq p$  the submatrix stored on processor  $\{0, (k - p)\}$  or  $\{(k - p), 0\}$  representing interactions with other processors was not positive definite and the factorization was not completed.

## 6 Further Comments

The total number of floating-point operations is approximately  $4n(b_w + 1)^2$ , assuming  $n \gg b_w$ . A call to this routine may be followed by a call to F07HSFP (PZPBTRS), to solve  $A_s X = B_s$ .

Because of permutation performed by F07HRFP (PZPBTRF) to achieve better parallelism, the factors produced (which are stored in the arrays A and AF) will be **different** to those produced by equivalent sequential codes. However, they can be used in a subsequent call to the corresponding solver routine, F07HSFP (PZPBTRS).

Users should not access the factorized matrix directly, and the arrays A and AF **must not** be altered between calls to the factorize and the solver routines.

## 6.1 Algorithmic Detail

None.

## 6.2 Parallelism Detail

This routine uses a divide and conquer algorithm. This algorithm is well suited for narrow banded matrices. The matrix is distributed one-dimensionally, with columns divided amongst the processes. Hence the matrix is divided into a few pieces (usually  $p$ , with one stored on each process) formed by some of its columns and then the algorithm proceeds in two phrases:

- (1) Local phase : The individual pieces (in fact only the diagonal blocks of the matrix) are factorized independently and in parallel. These factors are applied to the matrix creating fill-in, which is stored in a non-inspectable way in the array AF. Mathematically, this is equivalent to reordering the matrix  $A$  as  $PAP^T$  and then factorizing the principal leading submatrix of size equal to the sum of the sizes of the matrices factorized on each process.
- (2) Reduced system phase : A small  $(b_w \times (p - 1)) \times (b_w \times (p - 1))$  system is formed representing interaction of the larger blocks, and is stored (as are its factors) in the array AF. A parallel **block cyclic reduction** algorithm is then used to complete the factorization.

It is also important to note that the block size  $N_b^A$  should not be too small ( $N_b^A \geq 2 \times b_w$ ). Otherwise the divide and conquer algorithm performs poorly.

## 6.3 Accuracy

If UPLO = 'U', the computed factor  $U$  is the exact factor of a perturbed matrix  $A + E$ , where

$$E = (e_{i,j})_{1 \leq i,j \leq n}, \quad |E| \leq c(b_w + 1)\epsilon|U^H| \cdot |U|,$$

$c(b_w + 1)$  is a modest linear function of  $b_w + 1$ , and  $\epsilon$  is the *machine precision*. If UPLO = 'L', a similar statement holds for the computed factor  $L$ . So it follows that the matrix perturbation  $E$  verify  $|e_{ij}| \leq c(b_w + 1)\epsilon\sqrt{a_{ii}a_{jj}}$ .

## 7 References

- [1] 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>
- [2] 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
- [3] Golub G H and van Loan C F (1996) *Matrix Computations* Johns Hopkins University Press (3rd Edition), Baltimore

## 8 Example

This example is from the numerical solution of the Laplacian equation in two space dimensions. The differential equation has the form:

$$\begin{cases} -\Delta u &= (1 + 2i)\frac{5}{4}\pi^2 \sin(\pi x) \sin(\frac{\pi}{2}y) \text{ in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega; \end{cases}$$

where  $\Omega = (0; 1) \times (0; 2)$ , and  $\partial\Omega = ((0; 1) \times \{0\}) \cup (\{1\} \times (0; 2)) \cup ((0; 1) \times \{2\}) \cup (\{0\} \times (0; 2))$ . The true solution of this problem is  $u(x, y) = (1 + 2i) \sin(\pi x) \sin(\frac{\pi}{2}y)$ .

This problem is solved numerically using a five-point finite difference scheme. The matrix  $A$  has the form (numbering the point in the usual way, from left to right and from bottom to top):

$$\frac{1}{h^2} \times \begin{pmatrix} \mathbf{D} & \mathbf{E} & & & \\ \mathbf{E} & \mathbf{D} & \mathbf{E} & & \\ & & \ddots & \ddots & \ddots \\ & & & \mathbf{E} & \mathbf{D} & \mathbf{E} \\ & & & & \mathbf{E} & \mathbf{D} \end{pmatrix},$$

where the  $N_x$  by  $N_x$  matrices  $\mathbf{D}$  and  $\mathbf{E}$  have the following forms:

$$\mathbf{D} = \begin{pmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & & \ddots & \ddots & \ddots \\ & & & -1 & 4 & -1 \\ -1 & & & -1 & 4 \end{pmatrix} \quad \text{and} \quad \mathbf{E} = \begin{pmatrix} -1 & & & & \\ & \ddots & & & \\ & & & & \\ & & & & \\ & & & & -1 \end{pmatrix}$$

In this example the mesh has  $N = N_x \times N_y = 36$  inner points ( $N_x = 4$  and  $N_y = 9$  are the number of points in each direction), the half band width of the matrix  $A$  is  $b_w = N_x$  and its size is  $h = \frac{1}{N_x + 1} = \frac{2}{N_y + 1}$ . The precision of the computed solution is  $O(h^2)$  in the Euclidian norm.

## 8.1 Example Text

```
* F07HRFP Example Program Text
* NAG Parallel Library Release 3. NAG Copyright 1999.
* .. Parameters ..
INTEGER          NOUT
PARAMETER       (NOUT=6)
INTEGER          NMAX, BWMAX
PARAMETER       (NMAX=50,BWMAX=10)
INTEGER          MG, NG
PARAMETER       (MG=1,NG=4)
INTEGER          TDA, NBMAX, TDB, LDB, LDA
PARAMETER       (TDA=NMAX/(MG*NG),NBMAX=TDA,TDB=TDA,LDB=TDB,
+               LDA=BWMAX+1)
INTEGER          IAROW, LWORK, LAF, NRHSMAX
PARAMETER       (IAROW=0,LWORK=BWMAX*BWMAX,LAF=(NBMAX+2*BWMAX)
+               *BWMAX,NRHSMAX=5)
* .. Scalars in Common ..
INTEGER          NX, NY
* .. Local Scalars ..
DOUBLE PRECISION ERROR, ERROR0, ERRORP, ERRORPO
INTEGER          BW, I, IB, ICNTXT, IFAIL, INFO, JA, LEVEL, MP,
+               MYCOL, MYROW, N, NB, NP, NPCOL, NPROW, NRHS
LOGICAL          ROOT
CHARACTER        UPLO
CHARACTER*80     FORMAT
* .. Local Arrays ..
COMPLEX*16       A(LDA,TDA), AF(LAF), B(LDB,NRHSMAX),
+               SOL(LDB,NRHSMAX), WORK(LWORK)
INTEGER          IDESCA(9), IDESCB(9)
* .. External Functions ..
LOGICAL          Z01ACFP
EXTERNAL         Z01ACFP
* .. External Subroutines ..
EXTERNAL         DGERV2D, DGESD2D, EXACT, F01YWFP, F01YZFP,
+               F07HRFP, F07HSFP, GMAT, GRHS, X04BZFP, Z01AAF,
+               Z01ABFP, Z01ZAFP, Z02EAFP
```

```

*      .. Intrinsic Functions ..
INTRINSIC          DCONJG, DBLE, DSQRT
*      .. Common blocks ..
COMMON            /DIM/NX, NY
*      .. Executable Statements ..
ROOT = Z01ACFP()
IF (ROOT) WRITE (NOUT,*) 'F07HRFP Example Program Results '
*
*      Define the 2D processor grid 1-by-p type
*
MP = MG
NP = NG
IFAIL = 0
*
CALL Z01AAFP(ICNTXT,MP,NP,IFAIL)
*
CALL Z01ZAFP(ICNTXT,NPROW,NPCOL,MYROW,MYCOL)
*
*      Set error checking level
*
LEVEL = 1
CALL Z02EAFP(ICNTXT,LEVEL,IFAIL)
*
*      Initialization
*
NX = 4
NY = 9
N = NX*NY
BW = NX
UPLD = 'L'
NRHS = 1
NB = N/(MG*NG)
FORMAT = '(2F12.4)'
*
IF ((N.LE.NMAX) .AND. (BW.LE.BWMAX) .AND. (NB.LE.NBMAX)) THEN
*
*      Array Descriptor for A
*
      IDESCA(1) = 501
      IDESCA(2) = ICNTXT
      IDESCA(3) = N
      IDESCA(4) = NB
      IDESCA(5) = IAROW
      IDESCA(6) = LDA
      IDESCA(7) = 0
      JA = 1
*
*      Array Descriptor for B
*
      IDESCB(1) = 502
      IDESCB(2) = ICNTXT
      IDESCB(3) = N
      IDESCB(4) = NB
      IDESCB(5) = IAROW
      IDESCB(6) = LDB
      IDESCB(7) = 0
      IB = 1
*

```

```

        IFAIL = 0
        INFO = 0
*
*   Generation of the matrix A
*
        CALL F01YWFP(GMAT,UPL0,N,BW,A,IDESCA,IFAIL)
*
*   Generation of the vector B
*
        CALL F01YZFP(GRHS,N,NRHS,B,IDESCB,IFAIL)
*
*   Generation of the exact solution
*
        CALL F01YZFP(EXACT,N,NRHS,SOL,IDESCB,IFAIL)
*
*   Factorization of the matrix and solving of the linear system
*
        CALL F07HRFP(UPL0,N,BW,A,JA,IDESCA,AF,LAF,WORK,LWORK,INFO)
*
        IF (INFO.EQ.0) THEN
*
*   Solving the linear System
*
        CALL F07HSFP(UPL0,N,BW,NRHS,A,JA,IDESCA,B,IB,IDESCB,AF,LAF,
+           WORK,LWORK,INFO)
*
*   Compute the error
*
        ERROR = 0.DO
        ERROR0 = 0.DO
        DO 20 I = 1, NB
            ERROR = ERROR + (B(I,1)-SOL(I,1))*DCONJG(B(I,1)-SOL(I,1))
            ERROR0 = ERROR0 + SOL(I,1)*DCONJG(SOL(I,1))
20        CONTINUE
*
        IF (MYCOL.GT.0) THEN
            CALL DGEDSD2D(ICNTXT,1,1,ERROR,1,0,0)
            CALL DGEDSD2D(ICNTXT,1,1,ERROR0,1,0,0)
        END IF
*
*   Print solution
*
        IF (ROOT) THEN
            DO 40 I = 1, NP - 1
                CALL DGERV2D(ICNTXT,1,1,ERRORP,1,0,I)
                CALL DGERV2D(ICNTXT,1,1,ERRORPO,1,0,I)
                ERROR = ERROR + ERRORP
                ERROR0 = ERROR0 + ERRORPO
40            CONTINUE
*
            ERROR = ERROR/ERROR0
            ERROR = DSQRT(ERROR)
*
            WRITE (NOUT,*)
            WRITE (NOUT,*) ' Solutions '
            WRITE (NOUT,*)
            WRITE (NOUT,*) ' Computed '
            WRITE (NOUT,*)

```



```

        END IF
*
        IFAIL = 0
        CALL X04BZFP(NOUT,N,NRHS,B,IDESCB,FORMAT,WORK,IFAIL)
*
        IF (ROOT) THEN
            WRITE (NOUT,*)
            WRITE (NOUT,*) ' Exact '
            WRITE (NOUT,*)
        END IF
*
        IFAIL = 0
        CALL X04BZFP(NOUT,N,NRHS,SOL,IDESCB,FORMAT,WORK,IFAIL)
*
        IF (ROOT) THEN
            WRITE (NOUT,*)
            WRITE (NOUT,*) ' L2 error and H^2 ', ERROR,
+             1.DO/(DBLE(NX+1)*DBLE(NX+1))
        END IF
*
        ELSE IF (INFO.GT.0) THEN
            IF (ROOT) WRITE (NOUT,*)
+             'Matrix is not positive-definite'
        END IF
    END IF
*
    IFAIL = 0
    CALL Z01ABFP(ICNTXT,'N',IFAIL)
*
    STOP
    END
*
    SUBROUTINE GMAT(J1,JL,BW,UPLO,AL,LDAL)
*
    .. Scalar Arguments ..
    INTEGER          BW, J1, JL, LDAL
    CHARACTER        UPLO
*
    .. Array Arguments ..
    COMPLEX*16      AL(LDAL,*)
*
    .. Scalars in Common ..
    INTEGER          NX, NY
*
    .. Local Scalars ..
    INTEGER          BWP1, I, J, K
    CHARACTER        CHARL, CHARU
*
    .. Intrinsic Functions ..
    INTRINSIC        MOD
*
    .. Common blocks ..
    COMMON           /DIM/NX, NY
*
    .. Executable Statements ..
    BWP1 = BW + 1
    CHARL = 'L'
    CHARU = 'U'
*
    IF (UPLO.EQ.CHARL) THEN
*
        K = 1
        DO 20 J = J1, JL
*
            DO I = 1, LDAL

```

```

        AL(I,K) = 0.DO
      END DO
*
      AL(1,K) = 4.DO
      AL(2,K) = -1DO
      AL(BWP1,K) = -1.DO
*
      IF (MOD(J,NX).EQ.0) AL(2,K) = 0.DO
*
      K = K + 1
20    CONTINUE
*
      END IF
*
      IF (UPLO.EQ.CHARU) THEN
*
        K = 1
        DO 40 J = J1, JL
*
          DO I = 1, LDAL
            AL(I,K) = 0.DO
          END DO

          AL(1,K) = -1.DO
          AL(BW,K) = -1.DO
          AL(BWP1,K) = 4.DO
*
          IF (MOD(J,NX).EQ.1) AL(BW,K) = 0.DO
*
          K = K + 1
40    CONTINUE
*
        END IF
*
        End of GMAT
*
        RETURN
      END
*
      SUBROUTINE GRHS(J1,JL,NRHS,BL,LDBL)
*
      .. Scalar Arguments ..
      INTEGER          J1, JL, LDBL, NRHS
*
      .. Array Arguments ..
      COMPLEX*16      BL(LDBL,*)
*
      .. Scalars in Common ..
      INTEGER          NX, NY
*
      .. Local Scalars ..
      DOUBLE PRECISION FQ, H, HX, HY, PI, PI2
      INTEGER          J, JX, JY, K
*
      .. External Functions ..
      DOUBLE PRECISION X01AAF
      EXTERNAL        X01AAF
*
      .. Intrinsic Functions ..
      INTRINSIC       DBLE, DSIN, MOD
*
      .. Common blocks ..
      COMMON          /DIM/NX, NY
*
      .. Executable Statements ..
      K = 1

```

```

    PI = X01AAF(0.0D0)
    PI2 = PI*PI
    HX = 1.DO/DBLE(NX+1)
    HY = 2.DO/DBLE(NY+1)
    H = HX*HY
    FQ = 5.DO/4.DO
*
    DO 20 J = J1, JL
        JX = MOD(J,NX)
        JY = J/NX + 1
        IF (JX.EQ.0) THEN
            JX = NX
            JY = JY - 1
        END IF
        BL(K,1) = FQ*H*PI2*DSIN(DBLE(JX)*PI*HX)*DSIN(DBLE(JY)
+
            *PI*HY/2.DO)*(1.DO,2.DO)
        K = K + 1
    20 CONTINUE
*
*   End of GRHS
*
    RETURN
    END
*
    SUBROUTINE EXACT(J1,JL,NRHS,BL,LDBL)
*   .. Scalar Arguments ..
    INTEGER          J1, JL, LDBL, NRHS
*   .. Array Arguments ..
    COMPLEX*16       BL(LDBL,*)
*   .. Scalars in Common ..
    INTEGER          NX, NY
*   .. Local Scalars ..
    DOUBLE PRECISION HX, HY, PI
    INTEGER          J, JX, JY, K
*   .. External Functions ..
    DOUBLE PRECISION X01AAF
    EXTERNAL         X01AAF
*   .. Intrinsic Functions ..
    INTRINSIC        DBLE, DSIN, MOD
*   .. Common blocks ..
    COMMON           /DIM/NX, NY
*   .. Executable Statements ..
    K = 1
    PI = X01AAF(0.0D0)
    HX = 1.DO/DBLE(NX+1)
    HY = 2.DO/DBLE(NY+1)
*
    DO 20 J = J1, JL
        JX = MOD(J,NX)
        JY = J/NX + 1
        IF (JX.EQ.0) THEN
            JX = NX
            JY = JY - 1
        END IF
        BL(K,1) = DSIN(DBLE(JX)*PI*HX)*DSIN(DBLE(JY)*PI*HY/2.DO)*(1.DO,
+
            2.DO)
        K = K + 1
    20 CONTINUE

```

```

*
*   End of EXACT
*
*   RETURN
*   END

```

## 8.2 Example Data

None.

## 8.3 Example Results

F07HRFP Example Program Results

Solutions

Computed

0.1868	0.3736
0.3022	0.6045
0.3022	0.6045
0.1868	0.3736
0.3553	0.7106
0.5749	1.1498
0.5749	1.1498
0.3553	0.7106
0.4890	0.9781
0.7913	1.5825
0.7913	1.5825
0.4890	0.9781
0.5749	1.1498
0.9302	1.8604
0.9302	1.8604
0.5749	1.1498
0.6045	1.2090
0.9781	1.9561
0.9781	1.9561
0.6045	1.2090
0.5749	1.1498
0.9302	1.8604
0.9302	1.8604
0.5749	1.1498
0.4890	0.9781
0.7913	1.5825
0.7913	1.5825
0.4890	0.9781
0.3553	0.7106
0.5749	1.1498
0.5749	1.1498
0.3553	0.7106
0.1868	0.3736
0.3022	0.6045
0.3022	0.6045
0.1868	0.3736

Exact

0.1816	0.3633
0.2939	0.5878
0.2939	0.5878
0.1816	0.3633
0.3455	0.6910
0.5590	1.1180
0.5590	1.1180
0.3455	0.6910
0.4755	0.9511
0.7694	1.5388
0.7694	1.5388
0.4755	0.9511
0.5590	1.1180
0.9045	1.8090
0.9045	1.8090
0.5590	1.1180
0.5878	1.1756
0.9511	1.9021
0.9511	1.9021
0.5878	1.1756
0.5590	1.1180
0.9045	1.8090
0.9045	1.8090
0.5590	1.1180
0.4755	0.9511
0.7694	1.5388
0.7694	1.5388
0.4755	0.9511
0.3455	0.6910
0.5590	1.1180
0.5590	1.1180
0.3455	0.6910
0.1816	0.3633
0.2939	0.5878
0.2939	0.5878
0.1816	0.3633

L2 error and  $H^2$  2.839878462926276E-002 4.000000000000000E-002

---