F04HBFP

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

F04HBFP solves a real symmetric positive-definite band system of linear equations with multiple right-hand sides,

$$AX = B,$$

using a Cholesky factorization, where A and B are n by n and n by r matrices, respectively.

The routine first computes a Cholesky factorization of A as $A = PLL^TP^T$ (if UPLO = 'L') or $A = PU^TUP^T$ (if UPLO = 'U'), where P is a permutation matrix and U and L are banded upper and lower triangular matrices respectively. Then forward and backward substitutions are used to compute X.

2 Specification

```
 SUBROUTINE F04HBFP(ICNTXT, UPLO, N, NCR, BW, A, LDA, NRHS, B, LDB,
1                   AF, LAF, WORK, LWORK, IFAIL)
 DOUBLE PRECISION   A(LDA,*), B(LDB,NRHS), AF(LAF), WORK(LWORK)
 INTEGER            ICNTXT, N, NCR, BW, LDA, NRHS, LDB, LAF, LWORK,
1                   IFAIL
 CHARACTER*1        UPLO
```

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.
N_b	–	the maximum number of columns of the matrix A, or the maximum number of rows of the matrix B held locally on a logical processor;
N_x	–	the number of columns of the matrix A and the number of rows of the matrix B held locally on a logical processor, where $0 \leq N_x \leq N_b$.
$\lceil x \rceil$	–	the ceiling function of x, which gives the smallest integer which is not less than x.

3.2 Global and Local Arguments

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

Global input arguments: UPLO, N, BW, NRHS, IFAIL.

Global output arguments: IFAIL.

The remaining arguments are local.

3.3 Distribution Strategy

The matrix A is stored in a two-dimensional array A, which should be distributed over a one-dimensional array of processors, assuming a column block distribution, and is stored in compact format (see the F04 Chapter Introduction). The right-hand sides of the equation are stored in the array B in a row block distribution. Each logical processor contains at most $N_b = \lceil n/p \rceil$ columns of the matrix A and rows of the right-hand side B. Some logical processor may not contain any columns of A or rows of B if n is not large relative to p, but if $n > (p-1)^2$ then all processors will certainly contain some columns or rows of A or B respectively. The mapping for matrices is blocked, reflecting the nature of the **divide and conquer algorithm** as a task-parallel algorithm, see Section 6.2.

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 F04HBFP :

Real matrix generation:	column block distribution : F01ZRFP
Real matrix generation:	row block distribution : F01ZMFP
Real matrix output:	row block distribution : X04BFFP

4 Arguments

1: ICNTXT — INTEGER *Local Input*

On entry: the Library context, usually returned by a call to the Library Grid initialisation routine Z01AAFP.

Note: the value of ICNTXT **must not** be changed.

2: UPLO — CHARACTER*1 *Global Input*

On entry: indicates whether the upper or lower triangular part of A is stored and how A is factorized :

if UPLO = 'U', then the upper triangular part of A should be stored and A is factorized as PU^TUP^T, where P is a permutation matrix and U is upper triangular;
if UPLO = 'L', then the lower triangular part of A is stored and A is factorized as PLL^TP^T, where P is a permutation matrix and L is lower triangular.

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

3: N — INTEGER *Global Input*

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

Constraint: N ≥ 1.

4: NCR — INTEGER *Local Output*

On exit: N_x, the number of columns and rows of the matrix A and B respectively, held on the processor in which F04HBFP called.

5: BW — INTEGER *Global Input*

On entry: b_w, the number of super-diagonals or sub-diagonals of the matrix A.

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

Note: F04HBFP is suitable for the solution of 'narrow' band linear equations. Hence, for large N, BW should be much smaller than N−1.

6: A(LDA,∗) — DOUBLE PRECISION array *Local Input/Local Output*

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

On entry: the local part of the distributed matrix A (stored in compact storage format) which may contain the upper part if UPLO = 'U' or the lower part if UPLO = 'L'. See the F07 Chapter Introduction for further detail as how A should be stored.

On exit: the permutated triangular factor U or L from the Cholesky factorization of $A = PU^TUP^T$ or $A = PLL^TP^T$, using the same distribution strategy and storage format.

7: LDA — INTEGER *Local Input*

On entry: the size of the first dimension of the array A as declared in the (sub)program from which F04HBFP is called.

Constraint: LDA $\geq$ BW + 1.

8: NRHS — INTEGER *Global Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS $\geq$ 1.

9: B(LDB, NRHS) — DOUBLE PRECISION array *Local Input/Local Output*

On entry: the local part of the right-hand side B which is stored in row block fashion.

On exit: the n by r solution matrix X distributed in the same row block distribution.

10: LDB — INTEGER *Local Input*

On entry: the size of the first dimension of the array B as declared in the (sub)program from which F04HBFP is called.

Constraint: LDB $\geq N_b$.

11: AF(LAF) — DOUBLE PRECISION array *Local Output*

On exit: the auxiliary fill-in space. Fill-in is created and stored during the factorisation. If LAF is not large enough, after an unsuccesful exit, INT(AF(1)) will contain the minimum acceptable size of AF. (For this purpose, IFAIL should be set to -1 prior to the first call to F04HBFP.)

12: LAF — INTEGER *Local Input*

On entry: the dimension of the array AF .

Constraint: LAF $\geq$ (N_b+2×BW)×BW.

13: WORK(LWORK) — DOUBLE PRECISION array *Local Workspace*

On exit: if LWORK is not large enough, after an unsuccesful exit, INT(WORK(1)) will contain the minimum acceptable size of WORK. (For this purpose, IFAIL should be set to -1 prior to the first call to F04HBFP.)

14: LWORK — INTEGER *Local Input*

Note: this routine provides a facility to obtain the minimal size of the array WORK by setting LWORK = 1. In that case LWORK must be treated as a **global** input and INT(WORK(1)), on exit provides the optimal size of the array WORK. However no other useful computations are performed by the routine.

On entry: size of the workspace array WORK.

Constraints:

LWORK $\geq$ max(BW×BW,BW×NRHS),
or LWORK = 1.

15: IFAIL — INTEGER *Global Input/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 entry: IFAIL must be set to 0, −1 or 1. For users not familiar with this argument (described in the Essential Introduction) the recommended values are:

IFAIL = 0, if multigridding is **not** employed;
IFAIL = −1, if multigridding is employed.

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

5 Errors and Warnings

If on entry IFAIL = 0 or −1, 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).

5.1 Full Error Checking Mode Only

IFAIL = −2000

The routine has been called with an invalid value of ICNTXT on one or more processors.

IFAIL = −1000

The logical processor grid and library mechanism (Library Grid) have not been correctly defined, see Z01AAFP.

IFAIL = $-i$

On entry, the ith argument is invalid:

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.

5.2 Any Error Checking Mode

IFAIL > 0

If IFAIL = $k \leq p$ the submatrix stored and factored locally on processor $\{0, k\}$ or $\{k, 0\}$ was not positive definite, and the factorization was not completed. If IFAIL = $k \geq p$ 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 $n(b_w+1)^2 + 4b_w nr$, assuming $n >> b_w$.

6.1 Algorithmic Detail

The matrix A is first factorized using the Cholesky algorithms. Then forward and backward substitutions are used to calculate the solution. Assuming the decomposition of the matrix $A = PU^TUP^T = PLL^TP^T$, where P is a permutation matrix and U is upper triangular, L is lower triangular. Then the solution X is computed by solving $PU^TY = B$ and then $UP^TX = Y$, or $PLY = B$ and then $L^TP^TX = Y$.

6.2 Parallelism Detail

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

(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 should not be too small ($N_b \geq 2 \times b_w$). Otherwise the divide and conquer algorithm performs poorly.

The Level-3 BLAS operations used in this routine are carried out in parallel.

6.3 Accuracy

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

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

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

For each right-hand side vector b, the computed solution x is the exact solution of a perturbed system of equations $(A + E)x = b$, where

$$\begin{array}{ll} |E| \leq c(b_w + 1)\epsilon |U^T| \cdot |U| & \text{if UPLO = 'U',} \\ |E| \leq c(b_w + 1)\epsilon |L| \cdot |L^T| & \text{if UPLO = 'L',} \end{array}$$

$c(b_w + 1)$ is a modest linear function of $b_w + 1$ and ϵ is the ***machine precision***. If x is the true solution, then the computed solution $\hat{x}$ satisfies a forward error bound of the form

$$\frac{\|x - \hat{x}\|_\infty}{\|x\|_\infty} \leq \epsilon c(b_w + 1)\kappa(A),$$

where $\kappa(A)$ is the condition number of A (see the F07 Chapter Introduction).

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, Philadelpia, 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 :

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

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 $u(x) = sin(\pi x) sin(\frac{\pi}{2}y)$.

This problem is solved numerically using a five-point finite difference scheme. The matrix A has the form (numbering the points 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 :

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

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 matrix nalf band width 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. The example uses a 1 by 4 logical processor grid and a block size of 9.

8.1 Example Text

```
*     F04HBFP 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          LWORK, LAF, NRHSMAX
      PARAMETER        (LWORK=BWMAX*BWMAX,LAF=(NBMAX+2*BWMAX)*BWMAX,
     +                 NRHSMAX=5)
*     .. Scalars in Common ..
      INTEGER          NX, NY
*     .. Local Scalars ..
      DOUBLE PRECISION ERROR, ERROR0, ERRORP, ERRORP0
      INTEGER          BW, BWP1, I, ICNTXT, ICOFF, IFAIL, INFO, LEVEL,
     +                 MP, MX, MYCOL, MYROW, N, NB, NCR, NCX, NP, NRHS
      LOGICAL          ROOT
```

```
      CHARACTER        CHARL, CHARU, CNUMOP, TITOP, UPLO
      CHARACTER*80     FORMAT
*     .. Local Arrays ..
      DOUBLE PRECISION A(LDA,TDA), AF(LAF), B(LDB,NRHSMAX),
     +                 SOL(LDB,NRHSMAX), W(LDB,NRHSMAX), WORK(LWORK)
*     .. External Functions ..
      LOGICAL          Z01ACFP
      EXTERNAL         Z01ACFP
*     .. External Subroutines ..
      EXTERNAL         DGERV2D, DGESD2D, EXACT, F01ZMFP, F01ZRFP,
     +                 F04HBFP, GMATL, GMATU, GRHS, X04BFFP, Z01AAFP,
     +                 Z01ABFP, Z01ZAFP, Z02EAFP
*     .. Intrinsic Functions ..
      INTRINSIC        DBLE, DSQRT
*     .. Common blocks ..
      COMMON           /DIM/NX, NY
*     .. Executable Statements ..
      ROOT = Z01ACFP()
      IF (ROOT) WRITE (NOUT,*) 'F04HBFP Example Program Results '
*
*     Define the 2D processor grid 1-by-p type
*
      MP = MG
      NP = NG
      IFAIL = 0
      INFO = 0
*
      CALL Z01AAFP(ICNTXT,MP,NP,IFAIL)
*
*     Obtain information about this processor
*     Ensure all computations are within Library Grid
*
      CALL Z01ZAFP(ICNTXT,MP,NP,MYROW,MYCOL)
      IF (MYROW.LT.MP .AND. MYCOL.LT.NP) THEN
*
*     Set error checking level
*
         LEVEL = 1
         CALL Z02EAFP(ICNTXT,LEVEL,IFAIL)
*
*     Initialization
*
         NX = 4
         NY = 9
         N = NX*NY
         NB = N/(MG*NG)
         BW = NX
         UPLO = 'L'
         NRHS = 1
         FORMAT = 'F12.4'
         BWP1 = BW + 1
         CHARL = 'L'
         CHARU = 'U'
*
         IF ((N.LE.NMAX) .AND. (BW.LE.BWMAX)) THEN
*
*        Generation of the banded matrix A
*
```

```
            IF (UPLO.EQ.CHARL) THEN
*
               CALL F01ZRFP(ICNTXT,GMATL,BWP1,N,A,LDA,NCX,IFAIL)
*
            ELSE IF (UPLO.EQ.CHARU) THEN
*
               CALL F01ZRFP(ICNTXT,GMATU,BWP1,N,A,LDA,NCX,IFAIL)
*
            END IF
*
*        Generation of the vector B
*
            IFAIL = 0
            CALL F01ZMFP(ICNTXT,GRHS,N,NRHS,B,LDB,MX,IFAIL)
*
*        Generation of the exact solution
*
            IFAIL = 0
            CALL F01ZMFP(ICNTXT,EXACT,N,NRHS,SOL,LDB,MX,IFAIL)
*
*        Solve of AX = B
*
            CALL F04HBFP(ICNTXT,UPLO,N,NCR,BW,A,LDA,NRHS,B,LDB,AF,LAF,
     +                   WORK,LWORK,INFO)
*
            IF (INFO.EQ.0) THEN
*
*     Compute the error
*

               ERROR = 0.D0
               ERROR0 = 0.D0
               DO 20 I = 1, NB
                  ERROR = ERROR + (B(I,1)-SOL(I,1))*(B(I,1)-SOL(I,1))
                  ERROR0 = ERROR0 + SOL(I,1)*SOL(I,1)
   20          CONTINUE
*
               IF (MYCOL.GT.0) THEN
                  CALL DGESD2D(ICNTXT,1,1,ERROR,1,0,0)
                  CALL DGESD2D(ICNTXT,1,1,ERROR0,1,0,0)
               END IF
*
*           Print solution and the error
*
               IF (ROOT) THEN
                  DO 40 I = 1, NP - 1
                     CALL DGERV2D(ICNTXT,1,1,ERRORP,1,0,I)
                     CALL DGERV2D(ICNTXT,1,1,ERRORP0,1,0,I)
                     ERROR = ERROR + ERRORP
                     ERROR0 = ERROR0 + ERRORP0
   40             CONTINUE
*
                  ERROR = ERROR/ERROR0
                  ERROR = DSQRT(ERROR)
*
                  WRITE (NOUT,*)
                  WRITE (NOUT,*) ' Solutions '
                  WRITE (NOUT,*)
```

```
                  WRITE (NOUT,*) ' Computed '
                  WRITE (NOUT,*)
*
                  TITOP = 'Y'
                  CNUMOP = 'L'
               END IF
*
               IFAIL = 0
               ICOFF = 0
*
               CALL X04BFFP(ICNTXT,NOUT,MX,NRHS,B,LDB,FORMAT,TITOP,
     +                      CNUMOP,ICOFF,W,LDB,IFAIL)
*
               IF (ROOT) THEN
                  WRITE (NOUT,*)
                  WRITE (NOUT,*) ' Exact '
                  WRITE (NOUT,*)
               END IF
*
               CALL X04BFFP(ICNTXT,NOUT,MX,NRHS,SOL,LDB,FORMAT,TITOP,
     +                      CNUMOP,ICOFF,W,LDB,IFAIL)
*
               IF (ROOT) THEN
                  WRITE (NOUT,*)
                  WRITE (NOUT,*) ' L2 error and H^2 ', ERROR,
     +              1.D0/(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
      END IF
*
      IFAIL = 0
      CALL Z01ABFP(ICNTXT,'N',IFAIL)
*
      STOP
      END
*
      SUBROUTINE GMATL(BWP1,J1,JL,AL,LDAL)
*
*     This routine generates the matrix A
*     assuming lower triangular storage
*
*     .. Scalar Arguments ..
      INTEGER          BWP1, J1, JL, LDAL
*     .. Array Arguments ..
      DOUBLE PRECISION AL(LDAL,*)
*     .. Scalars in Common ..
      INTEGER          NX, NY
*     .. Local Scalars ..
      INTEGER          I, J, K
*     .. Intrinsic Functions ..
      INTRINSIC        MOD
*     .. Common blocks ..
      COMMON           /DIM/NX, NY
```

```
*     .. Executable Statements ..
      K = 1
      DO 20 J = J1, JL

         DO I = 1, LDAL
            AL(I,K) = 0.D0
         END DO

         AL(1,K) = 4.D0
         AL(2,K) = -1D0
         AL(BWP1,K) = -1.D0
*
         IF (MOD(J,NX).EQ.0) AL(2,K) = 0.D0
*
         K = K + 1
   20 CONTINUE
*
*     End of GMATL
*
      RETURN
      END
*
      SUBROUTINE GMATU(BWP1,J1,JL,AL,LDAL)
*
*     This routine generates the matrix A
*     assuming upper triangular storage
*
*     .. Scalar Arguments ..
      INTEGER          BWP1, J1, JL, LDAL
*     .. Array Arguments ..
      DOUBLE PRECISION AL(LDAL,*)
*     .. Scalars in Common ..
      INTEGER          NX, NY
*     .. Local Scalars ..
      INTEGER          I, J, K
*     .. Intrinsic Functions ..
      INTRINSIC        MOD
*     .. Common blocks ..
      COMMON           /DIM/NX, NY
*     .. Executable Statements ..
      K = 1
      DO 20 J = J1, JL

         DO I = 1, LDAL
            AL(I,K) = 0.D0
         END DO

         AL(1,K) = -1.D0
         AL(BWP1-1,K) = -1.D0
         AL(BWP1,K) = 4.D0
*
         IF (MOD(J,NX).EQ.1) AL(BWP1-1,K) = 0.D0
*
         K = K + 1
   20 CONTINUE
*
*     End of GMATU
*
```

```
      RETURN
      END
*
      SUBROUTINE GRHS(J1,JL,NRHS,BL,LDBL)
*
*     This routine generates the right hand side
*
*     .. Scalar Arguments ..
      INTEGER          J1, JL, LDBL, NRHS
*     .. Array Arguments ..
      DOUBLE PRECISION 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.D0/DBLE(NX+1)
      HY = 2.D0/DBLE(NY+1)
      H = HX*HY
      FQ = 5.D0/4.D0
*
      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.D0)
         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 ..
      DOUBLE PRECISION 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.D0/DBLE(NX+1)
      HY = 2.D0/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.D0)
         K = K + 1
   20 CONTINUE
*
*     End of EXACT
*
      RETURN
      END
```

8.2 Example Data

None.

8.3 Example Results

```
 F04HBFP Example Program Results

  Solutions

  Computed

   Array from logical processor    0,   0

             1
        0.1868
        0.3022
        0.3022
        0.1868
        0.3553
        0.5749
        0.5749
        0.3553
        0.4890

   Array from logical processor    0,   1
```

```
             1
        0.7913
        0.7913
        0.4890
        0.5749
        0.9302
        0.9302
        0.5749
        0.6045
        0.9781

 Array from logical processor    0,   2

             1
        0.9781
        0.6045
        0.5749
        0.9302
        0.9302
        0.5749
        0.4890
        0.7913
        0.7913

 Array from logical processor    0,   3

             1
        0.4890
        0.3553
        0.5749
        0.5749
        0.3553
        0.1868
        0.3022
        0.3022
        0.1868


Exact

 Array from logical processor    0,   0

             1
        0.1816
        0.2939
        0.2939
        0.1816
        0.3455
        0.5590
        0.5590
        0.3455
        0.4755

 Array from logical processor    0,   1

             1
        0.7694
        0.7694
```

```
          0.4755
          0.5590
          0.9045
          0.9045
          0.5590
          0.5878
          0.9511

     Array from logical processor    0,   2

               1
          0.9511
          0.5878
          0.5590
          0.9045
          0.9045
          0.5590
          0.4755
          0.7694
          0.7694

     Array from logical processor    0,   3

               1
          0.4755
          0.3455
          0.5590
          0.5590
          0.3455
          0.1816
          0.2939
          0.2939
          0.1816


   L2 error and H^2   2.839878462926260E-002  4.000000000000000E-002
```