

F11JHFP

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.

Note: you should read the the F11 Chapter Introduction before using this routine.

1 Description

F11JHFP solves the real symmetric sparse system of linear equations,

$$Ax = b,$$

where the matrix A is represented in coordinate storage format and distributed in cyclic row block form (see Sections 2.4 and 2.5 of the F11 Chapter Introduction).

F11JHFP uses one of the following iterative methods:

conjugate gradient iterative method (CG) or
SYMMLQ iterative method

and employs a block Jacobi preconditioner. Details of the solution methods and of the block Jacobi preconditioner can be found in Sections 2.3 and 2.7 of the F11 Chapter Introduction.

A call to F11JHFP must always be preceded by a call to F11ZBFP to set up auxiliary information about the matrix A in the array IAINFO, and by a call to F11DFFP, to generate the preconditioner. It is important to note that although F11JHFP is designed to solve *symmetric* systems, the preconditioner set-up routine F11DFFP is for general *unsymmetric* systems. It is therefore possible for F11DFFP to generate an unsymmetric preconditioner, in which case F11JHFP may fail with IFAIL = 6. If A is positive-definite this problem is unlikely to occur, but for indefinite cases there is no such guarantee. In any case it is advisable to inhibit pivoting in the call to F11DFFP.

F11JHFP is a Black Box routine which calls the iterative solver routines F11GAFP, F11GBFP, F11GCFP, the preconditioning routine F11DGFP and the matrix-vector multiplication routine F11XBFP. In order to use an alternative storage scheme, preconditioner, matrix-vector multiplication, or termination criterion; or if additional diagnostic information is required, the underlying routines could be used directly.

2 Specification

```

SUBROUTINE F11JHFP(ICNTXT, METHOD, N, NNZ, A, IROW, ICOL, NNZC, C,
1          IROWC, ICOLC, IPIVP, IPIVQ, B, TOL, MAXITN, X,
2          RNORM, ITN, IAINFO, WORK, LWORK, IFAIL)
  INTEGER  ICNTXT, N, NNZ, IROW(*), ICOL(*),
1          NNZC, IROWC(*), ICOLC(*), IPIVP(*), IPIVQ(*),
2          MAXITN, ITN, IAINFO(*), LWORK, IFAIL
  DOUBLE PRECISION A(*), C(*), B(*), TOL, X(*), RNORM, WORK(LWORK)
  CHARACTER*(*)  METHOD

```

3 Usage

3.1 Definitions

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

- p – $m_p \times n_p$, the total number of processors in the Library Grid.
- m_p – the number of rows in the Library Grid.
- n_p – the number of columns in the Library Grid.
- M_b – the blocking factor used in the cyclic row block distribution.
- m_l – the number of rows of the matrix A (and M) assigned to the calling processor ($m_l = \text{IAINFO}(3)$, see IAINFO).

- m_i^o – the overall number of rows (and columns) in the diagonal blocks A_k (and M_k), $k = 1, 2, \dots, n_{LB}$, assigned to the calling processor ($m_i^o = \text{IAINFO}(4)$, see IAINFO).
- n_{int}^i – the number of internal interface indices (see Section 2.6.1 of the F11 Chapter Introduction) for the calling processor ($n_{int}^i = \text{IAINFO}(6)$, see IAINFO).
- n_{int}^e – the number of external interface indices (see Section 2.6.1 of the F11 Chapter Introduction) for the calling processor ($n_{int}^e = \text{IAINFO}(7)$, see IAINFO).
- n_{LB} – the number of row blocks assigned to the calling processor ($n_{LB} = \text{IAINFO}(8)$, see IAINFO).

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: METHOD, N, TOL, MAXITN, IFAIL

Global input arguments: RNORM, ITN, IFAIL

The remaining arguments are local.

3.3 Distribution Strategy

The matrix A must be distributed in cyclic row block form.

When A is distributed in cyclic row block form, blocks of M_b contiguous rows of the matrix A are stored in coordinate storage format on the Library Grid cyclically row by row (i.e., in the row major ordering of the grid) starting from the $\{0,0\}$ logical processor. The distribution of the preconditioner M is identical to the distribution of A , and M is automatically generated in the appropriate distributed form by F11DFFP.

The vectors b and x are distributed conformally to the sparse matrix A , i.e., b and x are distributed across the Library Grid in the same way as each of the columns of the matrix A .

These data distributions are described in more detail in Section 2.5 of the F11 Chapter Introduction.

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

3.4 Related Routines

Some Library routines can be used to generate or distribute real sparse matrices in cyclic row block form, or to generate or distribute vectors conformally to a given sparse matrix:

Real sparse matrix generation: F01YAFP or F01YBFP

Real sparse matrix distribution: F01XAFP

Real vector generation: F01YEFP

Real vector scatter: F01XEFP

3.5 Requisites

The real symmetric sparse matrix A must have been preprocessed to set up the auxiliary information vector IAINFO by F11ZBFP. The preconditioner must have been generated by calling F11DFFP.

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: METHOD — CHARACTER*(*) *Global Input*
On entry: specifies the iterative method to be used. The possible choices are:

'CG' conjugate gradient;
 'SYMMLQ' SYMMLQ method.

Constraint: METHOD = 'CG' or 'SYMMLQ'.

Note: only the first character is checked, and this may be of either case.

- 3:** N — INTEGER *Global Input*
On entry: n , the order of the matrix A . It must contain the same value as the parameter N used in a prior call to F11ZBFP in which the array IAINFO was initialised.
Constraint: $N \geq 1$.
- 4:** NNZ — INTEGER *Local Input*
On entry: the number of non-zero elements of matrix A stored on the calling processor. It must contain the same value as the parameter NNZ returned from a prior call to F11ZBFP in which the array IAINFO was initialised.
Constraint: $NNZ > 0$.
- 5:** A(*) — DOUBLE PRECISION array *Local Input*
Note: the dimension of the array A must be at least $\max(1, NNZ)$.
On entry: the non-zero elements in the blocks of the matrix A assigned to the calling processor. The local non-zero elements must have been reordered by a prior call to F11ZBFP.
- 6:** IROW(*) — INTEGER array *Local Input*
7: ICOL(*) — INTEGER array *Local Input*
Note: the dimension of the arrays IROW and ICOL must be at least $\max(1, NNZ)$.
On entry: the local row and column indices of the non-zero elements supplied in A. The contents of the arrays IROW and ICOL **must not** be changed between successive calls to library routines involving the matrix A .
- 8:** NNZC — INTEGER array *Local Input*
On entry: the number of non-zero elements in the matrices C_k , $k = 1, 2, \dots, n_{LB}$, assigned to the calling processor as returned by a prior call to F11DFFP.
Constraint: $NNZC \geq 0$.
- 9:** C(*) — DOUBLE PRECISION array *Local Input*
Note: the dimension of the array C must be at least $\max(1, NNZC)$.
On entry: the values returned in the array C by a prior call to F11DFFP.
- 10:** IROWC(*) — INTEGER array *Local Input*
11: ICOLC(*) — INTEGER array *Local Input*
Note: the dimension of the arrays IROWC and ICOLC must be at least $\max(1, NNZC)$.
On entry: the local row and column indices of the non-zero elements supplied in C as returned by a prior call to F11DFFP.
- 12:** IPIVP(*) — INTEGER array *Local Input*
13: IPIVQ(*) — INTEGER array *Local Input*
Note: the dimension of the arrays IPIVP and IPIVQ must be at least $\max(1, m_l^o)$.
On entry: the local indices of pivot elements values as returned by a prior call to F11DFFP.
- 14:** B(*) — DOUBLE PRECISION array *Local Input*
Note: the dimension of the array B must be at least $\max(1, m_l)$.
On entry: the local part of the vector b .

15: TOL — DOUBLE PRECISION *Global Input*

On entry: the required tolerance. Let x_l denote the approximate solution at iteration l , and r_l the corresponding residual. The algorithm is considered to have converged at iteration l if:

$$\|r_l\|_\infty \leq \tau \times (\|b\|_\infty + \|A\|_\infty \|x_l\|_\infty).$$

If $\text{TOL} \leq 0.0$, $\tau = \max(\sqrt{\epsilon}, \sqrt{n}\epsilon)$ is used, where ϵ is the *machine precision*. Otherwise $\tau = \max(\text{TOL}, 10\epsilon, \sqrt{n}\epsilon)$ is used.

Constraint: $\text{TOL} < 1.0$.

16: MAXITN — INTEGER *Global Input*

On entry: the maximum number of iterations allowed.

Constraint: $\text{MAXITN} \geq 1$.

17: X(*) — DOUBLE PRECISION array *Local Input/Local Output*

Note: the dimension of the array X must be at least $\max(1, m_l)$.

On entry: an initial approximation to the solution vector x .

On exit: x_{ITN} , the final approximation to the solution vector x .

18: RNORM — DOUBLE PRECISION *Global Output*

On exit: $\|r_{\text{ITN}}\|_\infty$, the final value of the residual norm.

19: ITN — INTEGER *Global Output*

On exit: the number of iterations carried out.

20: IAINFO(*) — INTEGER array *Local Input*

Note: the dimension of the array IAINFO must be at least $\max(200, \text{IAINFO}(2))$.

On entry: the first IAINFO(2) elements of IAINFO contain auxiliary information about the matrix A . The array IAINFO must be initialised by a prior call to F11ZBFP and additional information must be stored in IAINFO by a prior call to F11DFFP. The first IAINFO(2) elements of IAINFO must not be changed between successive calls to library routines involving the matrix A .

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

22: LWORK — INTEGER *Local Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F11JHFP is called.

Constraint:

Method	Requirements
CG	$\text{LWORK} = 6m_l + \max(2m_l^o, n_{int}^i, n_{int}^e)$
SYMMLQ	$\text{LWORK} = 7m_l + \max(2m_l^o, n_{int}^i, n_{int}^e)$.

23: 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 multigriding is **not** employed;
- IFAIL = -1, if multigriding 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 i th argument was 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. An explanatory message distinguishes between these two cases.

$IFAIL = 1$

IAINFO was not set up by prior calls of F11ZBFP and F11DFFP.

$IFAIL = 2$

On entry, the data stored in the arguments N, NNZ, IROW, ICOL and IAINFO is inconsistent. This indicates that, after the array IAINFO was set up by a call to F11ZBFP at least one of these arguments was changed before calling F11JHFP.

$IFAIL = 3$

On entry, the data stored in the arguments N, NNZC, IROWC, ICOLC, IPIVP, IPIVQ and IAINFO is inconsistent. This indicates that, after the array IAINFO was set up by a call to F11ZBFP and F11DFFP, at least one of these arguments was changed before calling F11JHFP.

5.2 Any Error Checking Mode

$IFAIL = 4$

The required accuracy could not be obtained. However, a reasonable accuracy may have been obtained, and further iterations could not improve the result. Check the output value of RNORM for acceptability. This error code usually implies that the problem has been fully and satisfactorily solved to within or close to the accuracy available on the system. Further iterations are unlikely to improve on this situation.

$IFAIL = 5$

The required accuracy could not be obtained in MAXITN iterations.

$IFAIL = 6$

The preconditioner appears not to be positive-definite.

$IFAIL = 7$

The matrix appears not to be positive-definite (conjugate gradient method (CG) only).

$IFAIL = 8$

A serious error has occurred in an internal call to an auxiliary routine. Check all subroutine calls and array sizes. Seek expert help.

6 Further Comments

6.1 Accuracy

On successful termination, the final residual $r_l = b - Ax_l$, where $l = \text{ITN}$, satisfies the termination criterion

$$\|r_l\|_\infty \leq \tau \times (\|b\|_\infty + \|A\|_\infty \|x_l\|_\infty).$$

The value of the final residual norm is returned in RNORM.

6.2 Computational Costs

The number of arithmetic operation performed by each processor in each iteration is roughly proportional to the value of NNZC. The number of communication operations depends on the sparsity pattern of the matrix A and the particular row block distribution used.

The number of iterations required to achieve a prescribed accuracy cannot be easily determined *a priori*, as it depends dramatically on the conditioning and spectrum of the preconditioned iteration matrix $\bar{A} = M^{-1}A$.

7 References

- [1] Barrett R, Berry M, Chan T F, Demmel J, Donato J, Dongarra J, Eijkhout V, Pozo R, Romine C and van der Vorst H (1994) *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods* SIAM, Philadelphia
- [2] Paige C C and Saunders M A (1975) Solution of sparse indefinite systems of linear equations *SIAM J. Numer. Anal.* **12** 617–629
- [3] Saad Y (1996) *Iterative Methods for Sparse Linear Systems* PWS Publishing Company, Boston, MA

8 Example

This example solves a linear system of equations $Ax = b$ representing the five-point finite-difference approximation to the partial differential equation:

$$c_1 \frac{\partial^2 w}{\partial x^2} + c_2 \frac{\partial^2 w}{\partial y^2} + c_3 w = f$$

for $(x, y) \in \Omega = (0, 1)^2$, where c_i , $i = 1, \dots, 3$ are given real constants. The problem is discretised using central differences on a uniform $n_x \times n_x$ mesh and Dirichlet boundary conditions are prescribed on the entire boundary of Ω . The right-hand side and Dirichlet boundary values are obtained from the known true solution. The example also computes the infinity norm of the error between the approximate and true solutions.

Note that this example cannot be expected to work correctly for arbitrary choices of the coefficients c_i , since the mathematical problem is not always well-posed. However, it should generally work satisfactorily for elliptic problems.

8.1 Example Text

```
*      F11JHFP Example Program Text
*      NAG Parallel Library Release 3. NAG Copyright 1999.
*      .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5, NOUT=6)
      INTEGER          MLMAX
      PARAMETER        (MLMAX=1000)
      INTEGER          LA, LC
      PARAMETER        (LA=5*MLMAX, LC=2*LA)
```

```

INTEGER          LIA, LWORK
PARAMETER        (LIA=-1,LWORK=20*MLMAX)
DOUBLE PRECISION DZERO
PARAMETER        (DZERO=0.DO)
*
.. Scalars in Common ..
DOUBLE PRECISION C1, C2, C3
INTEGER          NX
*
.. Local Scalars ..
DOUBLE PRECISION ENORM, ENORML, RNORM, TOL
INTEGER          I, ICNTXT, IFAIL, ITN, J, LEVEL, MAXITN, MB, ML,
+               MP, N, NNZ, NNZC, NP, NVRLP
LOGICAL          ROOT, ZGRID
CHARACTER        DUP, KIND, SYMM, ZERO
CHARACTER*10     METHOD
CHARACTER*80     FORMAT
*
.. Local Arrays ..
DOUBLE PRECISION A(LA), B(MLMAX), C(LC), DTOL(1), TS(MLMAX),
+               WORK(LWORK), X(MLMAX)
INTEGER          CA(1), IAINFO(200), ICOL(LA), ICOLC(LC), IERR(1),
+               IPIVP(MLMAX), IPIVQ(MLMAX), IROW(LA), IROWC(LC),
+               LFILL(1), NPIVM(1), RA(1)
CHARACTER        MILU(1), PSTRAT(1)
*
.. External Functions ..
LOGICAL          Z01ACFP
EXTERNAL         Z01ACFP
*
.. External Subroutines ..
EXTERNAL         DGERV2D, DGESD2D, F01CPFP, F01YAFP, F01YEFP,
+               F11DFFP, F11JHFP, F11ZBFP, F11ZZFP, GMAT, GSOL,
+               GVEC, PRINTI, X04YAFP, Z01AAFP, Z01ABFP, Z01BBFP,
+               Z02EAFP
*
.. Intrinsic Functions ..
INTRINSIC        ABS, MAX
*
.. Common blocks ..
COMMON           /PROB/C1, C2, C3, NX
*
.. Executable Statements ..
ROOT = Z01ACFP()
IF (ROOT) WRITE (NOUT,*) 'F11JHFP Example Program Results'
*
*
*   Open input file on all processors
*
*   OPEN (NIN,FILE='f11jhfp.d')
*
*   Skip heading in data file
*
*   READ (NIN,*)
*   READ (NIN,*) MP, NP
*
*   Read problem parameters
*
*   READ (NIN,*) NX
*   N = NX**2
*
*   Read algorithmic parameters
*
*   READ (NIN,*) METHOD
*   READ (NIN,*) TOL, MAXITN
*   READ (NIN,*) FORMAT
*   READ (NIN,*) LEVEL

```

```

*
*   Read coefficients in PDE
*
*   READ (NIN,*) C1, C2, C3
*
*   Close input file
*
*   CLOSE (NIN)
*
*   Initialize Library Grid
*
*   IFAIL = 0
*   CALL Z01AAFP(ICNTXT,MP,NP,IFAIL)
*
*   Check whether processor is part of the Library Grid
*
*   CALL Z01BBFP(ICNTXT,ZGRID,IFAIL)
*   IF ( .NOT. ZGRID) GO TO 120
*
*   Set error checking level
*
*   CALL Z02EAFP(ICNTXT,LEVEL,IFAIL)
*
*   Generate sparse matrix
*
*   MB = (N+MP*NP-1)/(MP*NP)
*   CALL F01YAFP(ICNTXT,GMAT,N,MB,NNZ,A,LA,IROW,ICOL,IFAIL)
*
*   Set up auxiliary data for subsequent operations
*
*   DUP = 'F'
*   ZERO = 'R'
*   SYMM = 'S'
*   KIND = 'N'
*   CALL F11ZBFP(ICNTXT,N,MB,NNZ,A,IROW,ICOL,DUP,ZERO,SYMM,KIND,
+             IAINFO,LIA,IFAIL)
*
*   Check whether number of rows is less than the corresponding
*   maximum possible value determined by MLMAX
*
*   ML = IAINFO(3)
*   IERR(1) = 0
*   IF (ML.GT.MLMAX) IERR(1) = 1
*   CALL F01CPFP(ICNTXT,'X','All',1,1,IERR,1,RA,CA,1,0,-1,-1,IFAIL)
*   IF (IERR(1).NE.0) THEN
*       IF (ROOT) WRITE (NOUT,99997)
*       GO TO 100
*   END IF
*
*   Generate right-hand side vector
*
*   CALL F01YEFP(ICNTXT,GVEC,N,B,IAINFO,IFAIL)
*
*   Set up block Jacobi preconditioner
*
*   LFILL(1) = 0
*   DTOL(1) = 1.D-1
*   PSTRAT(1) = 'N'

```



```

MILU(1) = 'N'
NVRLP = 0
CALL F11DFFP(ICNTXT,N,NNZ,A,IROW,ICOL,NVRLP,LFILL,DTOL,PSTRAT,
+           MILU,IPIVP,IPIVQ,NNZC,C,LC,IROWC,ICOLC,NPIVM,IAINFO,
+           LIA,IFAIL)
*
*   Print summary of input parameters and options
*
IF (ROOT) CALL PRINTI(NOUT,METHOD,N,MAXITN,TOL,MP,NP,MB)
*
*   Set initial approximation to solution
*
DO 20 I = 1, ML
  X(I) = DZERO
20 CONTINUE
*
*   Solve equations
*
CALL F11JHFP(ICNTXT,METHOD,N,NNZ,A,IROW,ICOL,NNZC,C,IROWC,ICOLC,
+           IPIVP,IPIVQ,B,TOL,MAXITN,X,RNORM,ITN,IAINFO,WORK,
+           LWORK,IFAIL)
*
*   Generate true solution TS and error on local part of mesh
*
CALL F01YFEP(ICNTXT,GSOL,N,TS,IAINFO,IFAIL)
ENORML = 0.DO
DO 40 I = 1, IAINFO(3)
  ENORML = MAX(ENORML,ABS(TS(I)-X(I)))
40 CONTINUE
IF (.NOT. ROOT) CALL DGEDS2D(ICNTXT,1,1,ENORML,1,0,0)
*
*   Produce report
*
IF (ROOT) THEN
  WRITE (NOUT,'(/1X,'Summary of results'/1X,18(''-'))')
  WRITE (NOUT,99999)
+   'Number of iterations carried out (ITN)           -', ITN
  WRITE (NOUT,99998)
+   'Residual norm (RNORM)                           -',
+   RNORM
*
*   Receive local error norms and calculate global error norm
*
ENORM = ENORML
DO 80 I = 1, MP
  DO 60 J = 1, NP
    IF (I*.J.GT.1) THEN
      CALL DGERV2D(ICNTXT,1,1,ENORML,1,I-1,J-1)
      ENORM = MAX(ENORM,ENORML)
    END IF
60   CONTINUE
80   CONTINUE
  WRITE (NOUT,*)
  WRITE (NOUT,99998) 'Error norm =', ENORM

  WRITE (NOUT,'(/1X,'Solution vector'/1X,15(''-'))')
END IF
CALL X04YAFP(ICNTXT,NOUT,N,X,FORMAT,IAINFO,WORK,IFAIL)

```

```

*
*   Release internally allocated memory if necessary
*
100 IF (LIA.EQ.-1) CALL F11ZZFP(ICNTXT,IAINFO,IFAIL)
*
*   Finalize Library Grid
*
120 CALL Z01ABFP(ICNTXT,'N',IFAIL)
*
*   End of example program
*
STOP
*
99999 FORMAT (1X,A,I5)
99998 FORMAT (1X,A,3X,1P,D9.2)
99997 FORMAT (1X,'** ERROR: Number of rows per processor too large')
END

```

```

SUBROUTINE GMAT(I1,I2,N,NNZL,AL,LAL,IROWL,ICOLL)
*
*   This routine generates a block tridiagonal matrix
*   representing the five-point finite difference
*   approximation to the equation:
*
*    $c1*w_{xx} + c2*w_{yy} + c3*w = f$ 
*
*   where the ci are real coefficients.
*   The right-hand side vector is set up in the
*   routine GVEC.
*
*   .. Scalar Arguments ..
INTEGER      I1, I2, LAL, N, NNZL
*   .. Array Arguments ..
DOUBLE PRECISION AL(LAL)
INTEGER      ICOLL(LAL), IROWL(LAL)
*   .. Scalars in Common ..
DOUBLE PRECISION C1, C2, C3
INTEGER      NX
*   .. Local Scalars ..
DOUBLE PRECISION D1, D2, D3, D4, D5, H, RH, RH2
INTEGER      I, IX, IY
*   .. Intrinsic Functions ..
INTRINSIC    DBLE, MOD
*   .. Common blocks ..
COMMON      /PROB/C1, C2, C3, NX
*   .. Executable Statements ..
*
*   Calculate details of mesh
*
H = 1/DBLE(NX+1)
RH = 1.DO/H
RH2 = RH*RH
*
*   Define stencil coefficient
*
D1 = -2*RH2*(C1+C2) + C3
D2 = RH2*C1

```

```

D3 = RH2*C1
D4 = RH2*C2
D5 = RH2*C2
*
* Check whether there is sufficient storage space
*
IF (LAL.LT.5*(I2-I1+1)) THEN
  NNZL = -1
  RETURN
END IF
*
NNZL = 0
DO 20 I = I1, I2
*
* Calculate indices of mesh node

  IX = 1 + MOD(I-1,NX)
  IY = 1 + (I-1)/NX
*
* Set up diagonal elements of matrix first
*
  NNZL = NNZL + 1
  IROWL(NNZL) = I
  ICOLL(NNZL) = I
  AL(NNZL) = D1
*
* Now add off-diagonal elements where necessary
*
  IF (IX.GT.1) THEN
    NNZL = NNZL + 1
    IROWL(NNZL) = I
    ICOLL(NNZL) = I - 1
    AL(NNZL) = D3
  END IF
*
  IF (IX.LT.NX) THEN
    NNZL = NNZL + 1
    IROWL(NNZL) = I
    ICOLL(NNZL) = I + 1
    AL(NNZL) = D2
  END IF
*
  IF (IY.GT.1) THEN
    NNZL = NNZL + 1
    IROWL(NNZL) = I
    ICOLL(NNZL) = I - NX
    AL(NNZL) = D5
  END IF
  IF (IY.LT.NX) THEN
    NNZL = NNZL + 1
    IROWL(NNZL) = I
    ICOLL(NNZL) = I + NX
    AL(NNZL) = D4
  END IF
*
20 CONTINUE
*
RETURN

```

```

END

SUBROUTINE GVEC(I1,I2,F)
*
*   Computes the processor piece of the right-hand side vector
*   F of the linear system described in the subroutine GMAT.
*   It is based on the true solution defined in TSOL.
*
*   .. Scalar Arguments ..
INTEGER          I1, I2
*
*   .. Array Arguments ..
DOUBLE PRECISION F(*)
*
*   .. Scalars in Common ..
DOUBLE PRECISION C1, C2, C3
INTEGER          NX
*
*   .. Local Scalars ..
DOUBLE PRECISION D1, D2, D3, D4, D5, H, RH, RH2, W, WX, WXX, WY,
+               WYY, X, Y
INTEGER          I, IND, IX, IY
*
*   .. External Subroutines ..
EXTERNAL         TSOL
*
*   .. Intrinsic Functions ..
INTRINSIC       DBLE, MOD
*
*   .. Common blocks ..
COMMON          /PROB/C1, C2, C3, NX
*
*   .. Executable Statements ..
*
*   Calculate details of mesh
*
H = 1/DBLE(NX+1)
RH = 1.DO/H
RH2 = RH*RH
*
*   Define stencil coefficients
*
D1 = -2*RH2*(C1+C2) + C3
D2 = RH2*C1
D3 = RH2*C1
D4 = RH2*C2
D5 = RH2*C2
*
DO 20 I = I1, I2
*
*   Calculate coordinates (X,Y) of mesh point
*
IX = 1 + MOD(I-1,NX)
IY = 1 + (I-1)/NX
X = IX*H
Y = IY*H
*
*   Calculate true solution and its derivatives
*
CALL TSOL(X,Y,W,WX,WY,WXX,WYY)
*
*   Set right-hand side at interior points
*
IND = I - I1 + 1

```

```

      F(IND) = C1*WXX + C2*WYY + C3*W
*
*   Modify right-hand side near boundaries
*
      IF (IX.EQ.1) THEN
        CALL TSOL(0.DO,Y,W,WX,WY,WXX,WYY)
        F(IND) = F(IND) - D3*W
      ELSE IF (IX.EQ.NX) THEN
        CALL TSOL(1.DO,Y,W,WX,WY,WXX,WYY)
        F(IND) = F(IND) - D2*W
      END IF
      IF (IY.EQ.1) THEN
        CALL TSOL(X,0.DO,W,WX,WY,WXX,WYY)
        F(IND) = F(IND) - D5*W
      ELSE IF (IY.EQ.NX) THEN
        CALL TSOL(X,1.DO,W,WX,WY,WXX,WYY)
        F(IND) = F(IND) - D4*W
      END IF
*
20 CONTINUE
*
      RETURN
      END

      SUBROUTINE GSOL(I1,I2,TS)
*
*   Computes the processor piece of the true solution.
*
*   .. Scalar Arguments ..
      INTEGER          I1, I2
*   .. Array Arguments ..
      DOUBLE PRECISION TS(*)
*   .. Scalars in Common ..
      DOUBLE PRECISION C1, C2, C3
      INTEGER          NX
*   .. Local Scalars ..
      DOUBLE PRECISION H, W, WX, WXX, WY, WYY, X, Y
      INTEGER          I, IND, IX, IY
*   .. External Subroutines ..
      EXTERNAL        TSOL
*   .. Intrinsic Functions ..
      INTRINSIC       DBLE, MOD
*   .. Common blocks ..
      COMMON          /PROB/C1, C2, C3, NX
*   .. Executable Statements ..
*
*   Calculate details of mesh
*
      H = 1/DBLE(NX+1)

      DO 20 I = I1, I2
*
*   Calculate coordinates (X,Y) of mesh point
*
        IX = 1 + MOD(I-1,NX)
        IY = 1 + (I-1)/NX
        X = IX*H

```

```

        Y = IY*H
*
*   Calculate true solution and store in TS
*
        CALL TSOL(X,Y,W,WX,WY,WXX,WYY)
        IND = I - I1 + 1
        TS(IND) = W
*
20 CONTINUE
*
        RETURN
        END

SUBROUTINE TSOL(X,Y,W,WX,WY,WXX,WYY)
*
*   Defines a true solution W and its derivatives.
*   This example is for the function:
*
        w(x,y) = x*x-2*y*y
*
*   .. Scalar Arguments ..
        DOUBLE PRECISION W, WX, WXX, WY, WYY, X, Y
*   .. Executable Statements ..
        W = X*X - 2*Y*Y
        WX = 2*X
        WY = -4*Y
        WXX = 2.0
        WYY = -4.0
*
        RETURN
        END

SUBROUTINE PRINTI(NOUT,METHOD,N,MAXITN,TOL,MP,NP,MB)
*
*   Prints a summary of the input parameters and options.
*
*   .. Scalar Arguments ..
        DOUBLE PRECISION  TOL
        INTEGER           MAXITN, MB, MP, N, NOUT, NP
        CHARACTER*10      METHOD
*   .. Executable Statements ..
        WRITE (NOUT,99999)
        WRITE (NOUT,99997)
        + 'Number of processor rows in the Library grid (MP)    -', MP
        WRITE (NOUT,99997)
        + 'Number of processor columns in the Library grid (NP) -', NP
        WRITE (NOUT,99997)
        + 'Order of the system of equations (N)                -', N
        WRITE (NOUT,99997)
        + 'Block size used in the data distribution (MB)        -', MB
        WRITE (NOUT,99998)
        + 'Method used (METHOD)                                  -', METHOD
        WRITE (NOUT,99996)
        + 'Tolerance (TOL)                                       -', TOL
        WRITE (NOUT,99997)

```

```

+ 'Maximum number of iterations allowed (MAXITN)      -', MAXITN
*
*   End of subroutine PRINTI
*
*   RETURN
*
*
99999 FORMAT (/1X,'Summary of input parameters and options',/1X,39('-'),
+           /)
99998 FORMAT (1X,A,4X,A)
99997 FORMAT (1X,A,I5)
99996 FORMAT (1X,A,3X,1P,D9.2)
      END

```

8.2 Example Data

F11JHFP Example Program Data

```

  2   2           : MP, NP
  8           : NX
'CG'         : METHOD
1.0D-06 100    : TOL, MAXITN
'(8F8.4)'     : FORMAT
  0           : LEVEL
-1.0 -2.0 0.2  : C1, C2, C3

```

8.3 Example Results

F11JHFP Example Program Results

Summary of input parameters and options

```

Number of processor rows in the Library grid (MP) - 2
Number of processor columns in the Library grid (NP) - 2
Order of the system of equations (N) - 64
Block size used in the data distribution (MB) - 16
Method used (METHOD) - CG
Tolerance (TOL) - 1.00D-06
Maximum number of iterations allowed (MAXITN) - 100

```

Summary of results

```

Number of iterations carried out (ITN) - 14
Residual norm (RNORM) - 1.24D-03

```

Error norm = 2.92D-06

Solution vector

```

-0.0123  0.0247  0.0864  0.1728  0.2840  0.4198  0.5802  0.7654
-0.0864 -0.0494  0.0123  0.0988  0.2099  0.3457  0.5062  0.6914
-0.2099 -0.1728 -0.1111 -0.0247  0.0864  0.2222  0.3827  0.5679
-0.3827 -0.3457 -0.2839 -0.1975 -0.0864  0.0494  0.2099  0.3951

```

-0.6049 -0.5679 -0.5062 -0.4198 -0.3086 -0.1728 -0.0123 0.1728
-0.8765 -0.8395 -0.7778 -0.6914 -0.5802 -0.4444 -0.2840 -0.0988
-1.1975 -1.1605 -1.0988 -1.0123 -0.9012 -0.7654 -0.6049 -0.4198
-1.5679 -1.5309 -1.4691 -1.3827 -1.2716 -1.1358 -0.9753 -0.7901
