F11JEFP

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

F11JEFP 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).

F11JEFP uses one of the following iterative methods:

conjugate gradient iterative method (CG) or

SYMMLQ iterative method.

F11JEFP provides the following choice for preconditioning:

no preconditioning,

relaxed Jacobi preconditioning or

symmetric successive-over-relaxation (SSOR) preconditioning.

Details of the solution methods and of the preconditioners can be found in Sections 2.2 and 2.7 of the F11 Chapter Introduction)

A call to F11JEFP must always be preceded by a call to F11ZBFP to set up auxiliary information about the matrix A in the array IAINFO. If SSOR preconditioning is adopted, F11ZGFP must also be called to set up a multi-colour ordering scheme before calling F11JEF.

F11JEFP is a Black Box routine which calls the iterative solver routines F11GAFP, F11GBFP, F11GCFP, the preconditioning routine F11DDFP or F11DKFP 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 F11JEFP(ICNTXT, METHOD, PRECON, N, NNZ, A, IROW, ICOL,
     1                   OMEGA, ITNP, B, TOL, MAXITN, X, RNORM, ITN,
     2                   IAINFO, WORK, LWORK, IWORK, IFAIL)
      INTEGER            ICNTXT, N, NNZ, IROW(*), ICOL(*), ITNP, MAXITN,
     1                   ITN, IAINFO(*), LWORK, IWORK(*), IFAIL
      DOUBLE PRECISION   A(*), OMEGA, B(*), TOL, X(*), RNORM, WORK(LWORK)
      CHARACTER*(*)      METHOD
      CHARACTER*1        PRECON
```

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 assigned to the calling processor (m_l = IAINFO(3), see IAINFO).
n^i_{int}	–	the number of internal interface indices (see Section 2.6.1 of the F11 Chapter Introduction) for the calling processor (n^i_{int} = IAINFO(6), see IAINFO).
n^e_{int}	–	the number of external interface indices (see Section 2.6.1 of the F11 Chapter Introduction) for the calling processor (n^e_{int} = IAINFO(7), see IAINFO).
n_{LB}	–	the number of row blocks assigned to the calling processor (n_{LB} = 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, PRECON, N, OMEGA, ITNP, TOL, MAXITN, IFAIL
Global output 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 vectors b and x are distributed conformally to the 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. If SSOR preconditioning is adopted, F11ZGFP must also have been called to set up a multi-colour ordering scheme before calling F11JEFP.

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: PRECON — CHARACTER*1 *Global Input*

On entry: specifies the type of preconditioning to be used. The possible choices are:

'N'	no preconditioning;
'J'	Jacobi preconditioning;
'S'	symmetric successive-over-relaxation preconditioning.

Constraint: PRECON = 'N', 'J' or 'S'.

4: 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.

5: NNZ — INTEGER *Local Input*

On entry: the number of non-zero entries of the 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.

6: 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 entries in the blocks of the matrix A assigned to the calling processor. The local non-zero entries must have been reordered by a prior call to F11YAFP or F11ZBFP.

7: IROW($*$) — INTEGER array *Local Input*

8: 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 entries 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.

9: OMEGA — DOUBLE PRECISION *Global Input*

On entry: if PRECON = 'J' or 'S', OMEGA is the relaxation parameter ω in the Jacobi or SSOR method; otherwise, OMEGA is not referenced.

Constraint: 0.0 $<$ OMEGA $<$ 2.0.

10: ITNP — INTEGER *Global Input*

On entry: if PRECON = 'J' or 'S', ITNP is the number of Jacobi or successive-over-relaxation iterations to be performed in each preconditioning step; otherwise ITNP is not referenced.

Constraint: ITNP $\geq$ 1.

11: 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.

12: TOL — DOUBLE PRECISION *Global Input*

On entry: the required tolerance. Let x_l denote the approximate solution at iteration l, and r_l, $r_l = b - Ax_l$, 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 TOL $\leq$ 0.0, then $\tau = \max(\sqrt{\epsilon}, \sqrt{n}\,\epsilon)$ is used, where ϵ is the ***machine precision***. Otherwise $\tau = \max(\text{TOL}, 10\epsilon, \sqrt{n}\,\epsilon)$ is used.

Constraint: TOL < 1.0.

13: MAXITN — INTEGER *Global Input*

On entry: the maximum number of iterations allowed.

Constraint: MAXITN ≥ 1.

14: 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: $x_{\texttt{ITN}}$, the final approximation to the solution vector x.

15: RNORM — DOUBLE PRECISION *Global Output*

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

16: ITN — INTEGER *Global Output*

On exit: the number of iterations carried out.

17: IAINFO($*$) — INTEGER array *Local Input*

Note: the dimension of the array IAINFO must be at least max(200,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 F11ZGFP if PRECON = 'S'. The first IAINFO(2) elements of IAINFO must not be changed between successive calls to library routines involving the matrix A.

18: WORK(LWORK) — DOUBLE PRECISION array *Local Workspace*
19: LWORK — INTEGER *Local Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F11JEFP is called.

Constraint:

Method	**Requirements**
CG	LWORK = $6m_l + \max(n^i_{int}, n^e_{int}) + r$
SYMMLQ	LWORK = $7m_l + \max(n^i_{int}, n^e_{int}) + r$.

where

$r = 0$ if PRECON = 'N';
$r = 2m_l$ if PRECON = 'J'; and
$r = m_l + n^e_{int}$ if PRECON = 'S'.

20: IWORK($*$) — INTEGER array *Workspace*

Note: the dimension of the array IWORK must be at least 1. IWORK is not referenced in this release of the Library.

21: 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 was invalid. This error occured 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 a prior call to F11ZBFP. If PRECON = 'S', this error may also be caused by F11ZGFP not being called prior to F11JEFP.

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 F11JEFP.

IFAIL = 3

On entry, the matrix A has a zero diagonal element. Jacobi and SSOR preconditioners are not appropriate for this problem.

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 = 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 NNZ. 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 easily be determined *a priori*, as it depends dramatically on the conditioning and spectrum of the preconditioned iteration matrix.

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, $i = 1, \ldots, 3$ 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

```
*     F11JEFP 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
      PARAMETER        (LA=5*MLMAX)
      INTEGER          LIA, LWORK
      PARAMETER        (LIA=-1,LWORK=20*MLMAX)
*     .. Scalars in Common ..
      DOUBLE PRECISION C1, C2, C3
      INTEGER          NX
*     .. Local Scalars ..
      DOUBLE PRECISION ENORM, ENORML, OMEGA, RNORM, TOL
      INTEGER          I, ICNTXT, IFAIL, ITN, ITNP, J, LEVEL, MAXITN,
     +                 MB, ML, MP, MYCOL, MYROW, N, NCOLOR, NNZ, NP
      LOGICAL          ROOT, ZGRID
      CHARACTER        COLOR, DUP, KIND, PRECON, SYMM, ZERO
      CHARACTER*10     METHOD
      CHARACTER*80     FORMAT
*     .. Local Arrays ..
      DOUBLE PRECISION A(LA), B(MLMAX), TS(MLMAX), WORK(LWORK), X(MLMAX)
      INTEGER          CA(1), IAINFO(200), ICOL(LA), ICOLOR(MLMAX),
     +                 IDUMMY(1), IERR(1), IROW(LA), RA(1)
*     .. External Functions ..
      LOGICAL          Z01ACFP
      EXTERNAL         Z01ACFP
*     .. External Subroutines ..
      EXTERNAL         DGERV2D, DGESD2D, F01CPFP, F01YAFP, F01YEFP,
     +                 F11JEFP, F11ZBFP, F11ZGFP, F11ZZFP, GMAT, GSOL,
     +                 GVEC, PRINTI, X04YAFP, Z01AAFP, Z01ABFP, Z01BBFP,
     +                 Z01ZAFP, Z02EAFP
*     .. Intrinsic Functions ..
      INTRINSIC        ABS, MAX
*     .. Common blocks ..
      COMMON           /PROB/C1, C2, C3, NX
*     .. Executable Statements ..
      ROOT = Z01ACFP()
      IF (ROOT) WRITE (NOUT,*) 'F11JEFP Example Program Results'
*
*     Open input file on all processors
*
      OPEN (NIN,FILE='f11jefpe.d')
*
*     Skip heading in data file
*     Read processor grid size
*
      READ (NIN,*)
      READ (NIN,*) MP, NP
*
*     Read problem parameters
*
      READ (NIN,*) NX
      N = NX**2
```

```
*
*     Read algorithmic parameters
*
      READ (NIN,*) METHOD
      READ (NIN,*) PRECON
      READ (NIN,*) TOL, MAXITN
      READ (NIN,*) COLOR
      READ (NIN,*) OMEGA, ITNP
      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)
      CALL Z01ZAFP(ICNTXT,MP,NP,MYROW,MYCOL)
*
*     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)
*
*     Perform multi-coloring if SSOR preconditioner is to be used
*
      IF (PRECON.EQ.'S') CALL F11ZGFP(ICNTXT,N,NNZ,A,IROW,ICOL,COLOR,
     +                                NCOLOR,ICOLOR,IAINFO,LIA,IFAIL)
*
*     Print summary of input parameters and options
*
      IF (ROOT) CALL PRINTI(NOUT,METHOD,PRECON,N,MAXITN,TOL,COLOR,OMEGA,
     +                      ITNP,MP,NP,MB)
*
*     Set initial approximation to solution
*
      DO 20 I = 1, ML
         X(I) = 0.D0
   20 CONTINUE
*
*     Solve the equations
*
      CALL F11JEFP(ICNTXT,METHOD,PRECON,N,NNZ,A,IROW,ICOL,OMEGA,ITNP,B,
     +             TOL,MAXITN,X,RNORM,ITN,IAINFO,WORK,LWORK,IDUMMY,
     +             IFAIL)
*
*     Generate true solution TS and error on local part of mesh
*
      CALL F01YEFP(ICNTXT,GSOL,N,TS,IAINFO,IFAIL)
      ENORML = 0.D0
      DO 40 I = 1, IAINFO(3)
         ENORML = MAX(ENORML,ABS(TS(I)-X(I)))
   40 CONTINUE
      IF ( .NOT. ROOT) CALL DGESD2D(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)
*
*        Completion
*
  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.D0/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.D0/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.D0,Y,W,WX,WY,WXX,WYY)
            F(IND) = F(IND) - D3*W
         ELSE IF (IX.EQ.NX) THEN
            CALL TSOL(1.D0,Y,W,WX,WY,WXX,WYY)
            F(IND) = F(IND) - D2*W
         END IF
         IF (IY.EQ.1) THEN
            CALL TSOL(X,0.D0,W,WX,WY,WXX,WYY)
            F(IND) = F(IND) - D5*W
         ELSE IF (IY.EQ.NX) THEN
            CALL TSOL(X,1.D0,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,PRECON,N,MAXITN,TOL,COLOR,OMEGA,
     +                  ITNP,MP,NP,MB)
*
*     Prints a summary of the input parameters and options.
*
*
*     .. Scalar Arguments ..
      DOUBLE PRECISION  OMEGA, TOL
      INTEGER           ITNP, MAXITN, MB, MP, N, NOUT, NP
      CHARACTER         COLOR, PRECON
      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,99998)
     +  'Use the preconditioner (PRECON)                      -', PRECON
      WRITE (NOUT,99996)
     +  'Tolerance (TOL)                                      -', TOL
      WRITE (NOUT,99997)
     +  'Maximum number of iterations allowed (MAXITN)        -', MAXITN
      IF (PRECON.EQ.'S') WRITE (NOUT,99998)
     +    'Use the coloring (COLOR)                             -',
     +    COLOR
      IF (PRECON.EQ.'J' .OR. PRECON.EQ.'S') THEN
         WRITE (NOUT,99996)
     +     'Relaxation parameter (OMEGA)                        -',
     +     OMEGA
         WRITE (NOUT,99997)
     +     'Number of preconditioner iterations (ITNP)          -',
     +     ITNP
      END IF
*
*     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

```
F11JEFP Example Program Data
   2    2                         :  MP, NP
   8                              :  NX
  'CG'                            :  METHOD
  'S'                             :  PRECON
   1.0D-09  100                   :  TOL, MAXITN
  'B'                             :  COLOR
   1.0D+0 1                       :  OMEGA, ITNP
   '(8F8.4)'                      :  FORMAT
   0                              :  LEVEL
  -1.0 -2.0  0.2                  :  C1, C2, C3
```

8.3 Example Results

```
F11JEFP 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
Use the preconditioner (PRECON)                      -    S
Tolerance (TOL)                                      -    1.00D-09
Maximum number of iterations allowed (MAXITN)        -  100
Use the coloring (COLOR)                             -    B
Relaxation parameter (OMEGA)                         -    1.00D+00
Number of preconditioner iterations (ITNP)           -    1

Summary of results
------------------

Number of iterations carried out (ITN)               -   16
Residual norm (RNORM)                                -    1.04D-06

Error norm =    3.44D-09

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 -0.2840 -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
```