

F11YNFP

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

F11YNFP reorders the non-zero entries of an n by n complex sparse matrix A , distributed in cyclic row block form, on each processor in such a way that subsequent operations involving the matrix A can be performed efficiently. Auxiliary information about another matrix, \hat{A} , with the same pattern of non-zero entries and the same data distribution as A must have been stored in the array IAINFO by a prior call to F11ZPFP with KIND = 'R'. Additionally, the non-zero entries of A must be stored in the array A in exactly the same order as the non-zero entries of \hat{A} were stored on entry to F11ZPFP.

Depending on the values of the arguments DUP and ZERO in the prior call to F11ZPFP, entries with duplicate row and column indices may be removed, or their values may be summed; and any entries with zero values may be removed.

2 Specification

```
SUBROUTINE F11YNFP(ICNTXT, NNZ, A, IAINFO, IWORK, IFAIL)
  COMPLEX*16      A(*)
  INTEGER        ICNTXT, NNZ, IAINFO(*), IWORK(*), IFAIL
```

3 Usage

3.1 Definitions

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

M_b – the blocking factor used in the cyclic row block distribution.

3.2 Global and Local Arguments

The input argument IFAIL is global, and so must have the same value on entry to the routine on each processor. The output argument IFAIL is global and so will have the same value on exit from the routine on each processor. 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.

This data distribution is 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 complex sparse matrices in cyclic row block form:

Complex sparse matrix generation: F01YPFP or F01YQFP

Complex sparse matrix distribution: F01XPFP

3.5 Requisites

A complex sparse matrix of same sparsity pattern must have been preprocessed to set up the auxiliary information vector IAINFO by F11ZPFP, with input parameter KIND = 'R' before calling F11YNFP.

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:** NNZ — INTEGER *Local Input/Local Output*
On entry: the number of non-zero entries of the matrix A stored on the calling processor. It must be equal to the **entry** value of the argument NNZ in the prior call to F11ZPFP in which the array IAINFO was initialised.
Constraint: $NNZ > 0$.
On exit: the number of local non-zero entries with pairwise distinct row and column indices. Because of the removal of duplicates and zero entries, the value of NNZ on exit may be less than its value on entry.
- 3:** A(*) — COMPLEX*16 array *Local Input/Local Output*
Note: the dimension of the array A must be at least $\max(1, NNZ)$.
On entry: the numerical values of the non-zero entries in the blocks of the matrix A assigned to the calling processor. They must be stored in exactly the same order as the numerical values of the non-zero entries of \tilde{A} on entry to F11ZPFP.
On exit: the numerical values of the local non-zero entries reordered to enable subsequent operations involving the matrix A .
- 4:** 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 \tilde{A} . The array IAINFO must be initialised by a prior call to F11ZPFP with KIND = 'R'. The first IAINFO(2) elements of IAINFO **must not** be changed between successive calls to library routines involving the matrices A or \tilde{A} .
- 5:** IWORK(*) — INTEGER array *Local Workspace*
Note: the dimension of the array IWORK must be at least $\max(1, NNZE)$, where NNZE is the value of the argument NNZ **on exit** from F11ZPFP, if DUP = 'S' in the prior call to F11ZPFP; otherwise, IWORK is not referenced and its dimension must be at least 1.
- 6:** 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 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 initialised by a prior call to F11ZPFP.

$IFAIL = 2$

The argument KIND was not set to 'R' in the prior call to F11ZPFP in which IAINFO was initialized.

$IFAIL = 3$

On entry, the data stored in the arguments NNZ and IAINFO is inconsistent. This may indicate that the array IAINFO has been changed since it was set up by a call to F11ZPFP.

6 Further Comments

If the argument ZERO was set to 'R' in the prior call to F11ZPFP, any zero or non-zero entry of A at the location of a removed zero entry of \tilde{A} will be removed.

7 References

None.

8 Example

This example solves a sequence of linear systems of equations $Ax = b$ representing five-point finite-difference approximations to the partial differential equations:

$$c_1 \frac{\partial^2 w}{\partial x^2} + c_2 \frac{\partial^2 w}{\partial y^2} + c_3 \frac{\partial w}{\partial x} + c_4 \frac{\partial w}{\partial y} + c_5 w = f$$

for $(x, y) \in \Omega = (0, 1)^2$, where c_i , $i = 1, \dots, 5$ are given complex constants, read from file for each problem in the sequence. Each 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

```

*   F11YNFP 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, LIWORK
PARAMETER       (LIA=-1,LWORK=20*MLMAX,LIWORK=LA)
COMPLEX*16      ZZERO
PARAMETER       (ZZERO=(0.D0,0.D0))
*   .. Scalars in Common ..
COMPLEX*16      C1, C2, C3, C4, C5
INTEGER          NX
*   .. Local Scalars ..
DOUBLE PRECISION ENORM, ENORML, OMEGA, RNORM, TOL
INTEGER          I, ICNTXT, IEQNS, IFAIL, ITN, ITNP, J, LEVEL, M,
+               MAXITN, MB, ML, MP, N, NEQNS, NNZ, NNZE, NP
LOGICAL          ROOT, ZGRID
CHARACTER        DUP, KIND, PRECON, SYMM, WHAT, ZERO
CHARACTER*10     METHOD
CHARACTER*80     FORMAT
*   .. Local Arrays ..
COMPLEX*16      A(LA), B(MLMAX), TS(MLMAX), WORK(LWORK), X(MLMAX)
INTEGER          CA(1), IAINFO(200), ICOL(LA), IERR(1), IROW(LA),
+               IWORK(LIWORK), RA(1)
*   .. External Functions ..
LOGICAL          Z01ACFP
EXTERNAL         Z01ACFP
*   .. External Subroutines ..
EXTERNAL         DGERV2D, DGESD2D, F01CPFP, F01YQFP, F01YTFP,
+               F11DSFP, F11YNFP, F11ZPFP, F11ZZFP, GMAT, GSOL,
+               GVEC, PRINTI, X04YPFP, Z01AAFP, Z01ABFP, Z01BBFP,
+               Z02EAFP
*   .. Intrinsic Functions ..
INTRINSIC        ABS, MAX
*   .. Common blocks ..
COMMON          /PROB/C1, C2, C3, C4, C5, NX
*   .. Executable Statements ..
ROOT = Z01ACFP()
IF (ROOT) WRITE (NOUT,*) 'F11YNFP Example Program Results'
*
*   Open input file on all processors
*
OPEN (NIN,FILE='f11ynfpe.d')
*
*   Skip heading in data file
*   Read size of processor grid
*
READ (NIN,*)
READ (NIN,*) MP, NP
*
*   Read problem parameters
*

```

```

      READ (NIN,*) NX, NEQNS
      N = NX**2
      MB = (N+MP*NP-1)/(MP*NP)
*
*   Read algorithmic parameters
*
      READ (NIN,*) METHOD
      READ (NIN,*) PRECON, OMEGA, ITNP
      READ (NIN,*) M
      READ (NIN,*) TOL, MAXITN
      READ (NIN,*) FORMAT
      READ (NIN,*) LEVEL
*
*   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 140
*
*   Set error checking level
*
      CALL Z02EAFP(ICNTXT,LEVEL,IFAIL)
*
      DO 100 IEQNS = 1, NEQNS
*
*   Read complex coefficients in PDE
*
          READ (NIN,*) C1, C2, C3, C4, C5
*
*   Switch off generation of row and column indices after first
*   equation
*
          WHAT = 'N'
          IF (IEQNS.EQ.1) WHAT = 'C'
*
*   Generate sparse matrix
*
          CALL F01YQFP(ICNTXT,GMAT,WHAT,N,MB,NNZE,A,LA,IROW,ICOL,IFAIL)
*
*   Deal with first system of equations
*
          IF (IEQNS.EQ.1) THEN
*
*   Print summary of input parameters and options
*
              IF (ROOT) CALL PRINTI(NOUT,METHOD,N,MAXITN,TOL,M,MP,NP,MB)
*
*   Set up auxiliary data for subsequent operations
*
              NNZ = NNZE
              DUP = 'F'
              ZERO = 'R'
              SYMM = 'S'
              KIND = 'R'

```

```

      CALL F11ZPFP(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 120
      END IF
*
*   Deal with all following systems of equations
*
      ELSE
*
*   Permute elements in A appropriately
*
      CALL F11YNFP(ICNTXT,NNZE,A,IAINFO,IWORK,IFAIL)
*
      END IF
*
*   Generate right-hand side vector
*
      CALL F01YTFP(ICNTXT,GVEC,N,B,IAINFO,IFAIL)
*
*   Set initial approximation to solution
*
      DO 20 I = 1, ML
        X(I) = ZZERO
20    CONTINUE
*
*   Solve equations
*
      CALL F11DSFP(ICNTXT,METHOD,PRECON,N,NNZ,A,IROW,ICOL,OMEGA,ITNP,
+           B,M,TOL,MAXITN,X,RNORM,ITN,IAINFO,WORK,LWORK,
+           IWORK,IFAIL)
*
*   Generate true solution TS and error on local part of mesh
*
      CALL F01YTFP(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 X04YFPF(ICNTXT,NOUT,N,X,FORMAT,IAINFO,WORK,IFAIL)
100 CONTINUE
*
*      Close input file
*
      CLOSE (NIN)
*
*      Release internally allocated memory if necessary
*
120 IF (LIA.EQ.-1) CALL F11ZZFP(ICNTXT,IAINFO,IFAIL)
*
*      Completion
*
140 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(WHAT,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:
*
*       $c_1 w_{xx} + c_2 w_{yy} + c_3 w_x + c_4 w_y + c_5 w = f$ 
*
*      where the  $c_i$  are complex coefficients.
*      The right-hand side vector is set up in the
*      routine GVEC.
*

```

```

*      .. Scalar Arguments ..
      INTEGER      I1, I2, LAL, N, NNZL
      CHARACTER    WHAT
*      .. Array Arguments ..
      COMPLEX*16   AL(LAL)
      INTEGER      ICOLL(LAL), IROWL(LAL)
*      .. Scalars in Common ..
      COMPLEX*16   C1, C2, C3, C4, C5
      INTEGER      NX
*      .. Local Scalars ..
      COMPLEX*16   D1, D2, D3, D4, D5
      DOUBLE PRECISION H, RH, RH2
      INTEGER      I, IX, IY
      LOGICAL      LCOORD
*      .. Intrinsic Functions ..
      INTRINSIC    DBLE, MOD
*      .. Common blocks ..
      COMMON       /PROB/C1, C2, C3, C4, C5, NX
*      .. Executable Statements ..

      LCOORD = WHAT .EQ. 'C' .OR. WHAT .EQ. 'c'

*
*      Calculate details of mesh
*
      H = 1/DBLE(NX+1)
      RH = 1.DO/H
      RH2 = RH*RH

*
*      Define stencil coefficient
*
      D1 = -2*RH2*(C1+C2) + C5
      D2 = RH2*C1 + 0.5*RH*C3
      D3 = RH2*C1 - 0.5*RH*C3
      D4 = RH2*C2 + 0.5*RH*C4
      D5 = RH2*C2 - 0.5*RH*C4

*
*      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
         IF (LCOORD) THEN
            IROWL(NNZL) = I
            ICOLL(NNZL) = I
         END IF

```



```

        AL(NNZL) = D1
*
*   Now add off-diagonal elements where necessary
*
        IF (IX.GT.1) THEN
            NNZL = NNZL + 1
            IF (LCOORD) THEN
                IROWL(NNZL) = I
                ICOLL(NNZL) = I - 1
            END IF
            AL(NNZL) = D3
        END IF
*
        IF (IX.LT.NX) THEN
            NNZL = NNZL + 1
            IF (LCOORD) THEN
                IROWL(NNZL) = I
                ICOLL(NNZL) = I + 1
            END IF
            AL(NNZL) = D2
        END IF
*
        IF (IY.GT.1) THEN
            NNZL = NNZL + 1
            IF (LCOORD) THEN
                IROWL(NNZL) = I
                ICOLL(NNZL) = I - NX
            END IF
            AL(NNZL) = D5
        END IF
        IF (IY.LT.NX) THEN
            NNZL = NNZL + 1
            IF (LCOORD) THEN
                IROWL(NNZL) = I
                ICOLL(NNZL) = I + NX
            END IF
            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 ..
COMPLEX*16      F(*)
*   .. Scalars in Common ..
COMPLEX*16      C1, C2, C3, C4, C5
INTEGER          NX

```

```

*      .. Local Scalars ..
COMPLEX*16      D1, D2, D3, D4, D5, W, WX, WXX, WY, WYY
DOUBLE PRECISION H, RH, RH2, X, Y
INTEGER      I, IND, IX, IY
*      .. External Subroutines ..
EXTERNAL      TSOL
*      .. Intrinsic Functions ..
INTRINSIC      DBLE, MOD
*      .. Common blocks ..
COMMON      /PROB/C1, C2, C3, C4, C5, 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) + C5
      D2 = RH2*C1 + 0.5*RH*C3
      D3 = RH2*C1 - 0.5*RH*C3
      D4 = RH2*C2 + 0.5*RH*C4
      D5 = RH2*C2 - 0.5*RH*C4
*
      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*WX + C4*WY + C5*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 ..
COMPLEX*16      TS(*)
*
*   .. Scalars in Common ..
COMPLEX*16      C1, C2, C3, C4, C5
INTEGER         NX
*
*   .. Local Scalars ..
COMPLEX*16      W, WX, WXX, WY, WYY
DOUBLE PRECISION H, X, Y
INTEGER         I, IND, IX, IY
*
*   .. External Subroutines ..
EXTERNAL        TSOL
*
*   .. Intrinsic Functions ..
INTRINSIC       DBLE, MOD
*
*   .. Common blocks ..
COMMON          /PROB/C1, C2, C3, C4, C5, 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) = sin(x) + i*(x*x-2*y*y)
*
*   .. Scalar Arguments ..
*   COMPLEX*16      W, WX, WXX, WY, WYY
*   DOUBLE PRECISION X, Y
*   .. Intrinsic Functions ..
*   INTRINSIC      COS, DCMLPX, SIN
*   .. Executable Statements ..
*   W = DCMLPX(SIN(X),X*X-2*Y*Y)
*   WX = DCMLPX(COS(X),2*X)
*   WY = DCMLPX(0.0D0,-4*Y)
*   WXX = DCMLPX(-SIN(X),2.0D0)
*   WYY = DCMLPX(0.0D0,-4.0D0)
*
*   RETURN
*   END

SUBROUTINE PRINTI(NOUT,METHOD,N,MAXITN,TOL,M,MP,NP,MB)
*
*   Prints a summary of the input parameters and options.
*
*   .. Scalar Arguments ..
*   DOUBLE PRECISION  TOL
*   INTEGER           M, 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
*   IF (METHOD.EQ.'RGMRES') THEN
*       WRITE (NOUT,99997)
*   + 'Dimension of RGMRES orthogonal basis (M)            -', M
*   ELSE IF (METHOD.EQ.'BICGSTAB') THEN
*       WRITE (NOUT,99997)
*   + 'Order of BICGSTAB method (M)                        -', M
*   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

```

F11YNFP Example Program Data
  2      2      : MP, NP
  4      2      : NX, NEQNS
'BICGSTAB'      : METHOD
'J', 1.0D0, 2   : PRECON, OMEGA, ITNP
  2           : M
1.0D-09 100     : TOL, MAXITN
'(4(:,' ' ('',F7.4,'',',',F7.4,'')'))' : FORMAT
  0           : LEVEL
(1.0, 2.0) (1.0,-1.0)
(0.0, 3.0) (1.0, 0.0)
(1.3,-2.2)      : C1, C2, C3, C4, C5
(2.0, 0.0) (3.0, 2.0)
(1.0, 0.0) (0.0, 2.0)
(3.0,-2.0)      : C1, C2, C3, C4, C5
    
```

8.3 Example Results

F11YNFP 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) - 16
Block size used in the data distribution (MB) - 4
Method used (METHOD) - BICGSTAB
Tolerance (TOL) - 1.00D-09
Maximum number of iterations allowed (MAXITN) - 100
Order of BICGSTAB method (M) - 2
    
```

Summary of results

```

Number of iterations carried out (ITN) - 8
Residual norm (RNORM) - 2.99D-10
    
```

Error norm = 9.35D-04

Solution vector

```

( 0.1985,-0.0405) ( 0.3892, 0.0794) ( 0.5644, 0.2795) ( 0.7172, 0.5597)
( 0.1983,-0.2806) ( 0.3889,-0.1608) ( 0.5642, 0.0393) ( 0.7171, 0.3196)
    
```

(0.1983,-0.6806) (0.3890,-0.5608) (0.5642,-0.3607) (0.7171,-0.0804)
(0.1985,-1.2404) (0.3892,-1.1205) (0.5644,-0.9204) (0.7172,-0.6402)

Summary of results

Number of iterations carried out (ITN) - 8
Residual norm (RNORM) - 1.58D-13

Error norm = 8.05D-05

Solution vector

(0.1986,-0.0400) (0.3894, 0.0800) (0.5646, 0.2800) (0.7173, 0.5600)
(0.1986,-0.2800) (0.3893,-0.1600) (0.5646, 0.0400) (0.7173, 0.3200)
(0.1986,-0.6800) (0.3893,-0.5600) (0.5646,-0.3600) (0.7173,-0.0800)
(0.1986,-1.2400) (0.3894,-1.1200) (0.5646,-0.9200) (0.7173,-0.6400)
