

# F01XAFP

## 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

F01XAFP distributes an  $n$  by  $n$  real sparse matrix  $A$ , stored in coordinate storage format on one or more processors, using a cyclic row block distribution (see Section 2.5 of the F11 Chapter Introduction). Depending on the value of the input parameter WHAT, F01XAFP distributes either (i) both the numerical values and the row and column coordinates of the non-zero entries of the matrix  $A$  or (ii) only the numerical values of the non-zero entries. The latter option should be used if the matrix  $A$  has the same pattern of non-zero entries as a previously generated matrix.

This routine distributes matrices in the form required by a number of routines in Chapter F11.

### 2 Specification

```

SUBROUTINE F01XAFP(ICNTXT, WHAT, IS, JS, N, MB, NNZ, A, LA, IROW,
1          ICOL, IWORK, IFAIL)
DOUBLE PRECISION A(LA)
CHARACTER*1     WHAT
INTEGER         ICNTXT, IS, JS, N, MB, NNZ, LA, IROW(LA),
1          ICOL(LA), IWORK(*), IFAIL

```

### 3 Usage

#### 3.1 Definitions

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

- $m_p$  – the number of rows in the Library Grid.
- $n_p$  – the number of columns in the Library Grid.
- $M_b$  – the blocking factor for the distribution of the rows of the matrix.

#### 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:       WHAT, IS, JS, N, MB, IFAIL

Global output arguments:     IFAIL

The remaining arguments are local.

#### 3.3 Distribution Strategy

On entry to F01XAFP, the entire matrix  $A$  must be stored in coordinate storage format on the 'source' processors, specified by the input parameters IS and JS.

On exit from F01XAFP, 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.

## 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:** WHAT — CHARACTER\*1 *Global Input*  
*On entry:* specifies which parts of the coordinate storage representation of  $A$  are to be distributed:  
 if WHAT = 'C', both the numerical values and the row and column coordinates of the non-zero entries of  $A$  are to be distributed;  
 if WHAT = 'N', only the numerical values of the non-zero entries of  $A$  are to be distributed.
- Constraint:* WHAT = 'C' or 'N'.
- 3:** IS — INTEGER *Global Input*
- 4:** JS — INTEGER *Global Input*  
*On entry:* the Library Grid coordinates of the source processor(s) on which  $A$  is stored.
- If JS = -1, then all processors in row IS of the Library Grid must store on entry a copy of  $A$ . Similarly, if IS = -1, then all processors in column JS of the Library Grid must store on entry a copy of  $A$ .
- If IS = JS = -1, then all processors in the Library Grid must store on entry a copy of  $A$ .
- Constraint:*  $-1 \leq IS \leq m_p - 1$  and  $-1 \leq JS \leq n_p - 1$ .
- 5:** N — INTEGER *Global Input*  
*On entry:*  $n$ , the order of the matrix  $A$ .
- Constraint:*  $N \geq 1$ .
- 6:** MB — INTEGER *Global Input*  
*On entry:*  $M_b$ , the blocking factor used to distribute the rows of the matrix  $A$ .
- Constraint:*  $MB \geq 1$ .
- 7:** NNZ — INTEGER *Local Input/Local Output*  
*On entry:* on the source processors, as specified by the input parameters IS and JS, the number of non-zero entries in  $A$ ; on all other processors NNZ is not used.
- Constraint:*  $NNZ \geq 0$  on the source processors; NNZ must have the same value on all source processors.
- On exit:* the number of non-zero entries in the blocks of the matrix  $A$  to be stored locally.
- 8:** A(LA) — DOUBLE PRECISION array *Local Input/Local Output*  
*On entry:* on the source processors, as specified by the input parameters IS and JS, the first NNZ elements of  $A$  must contain the non-zero entries of the matrix  $A$ ; on all other processors the input values of  $A$  are not used.
- Constraint:* A(1:NNZ) must be identical on all source processors.
- On exit:* if IFAIL = 0, the first NNZ elements of  $A$  contain the non-zero entries of the row blocks of the matrix  $A$  stored locally.

**9: LA — INTEGER***Local Input*

*On entry:* the dimension of the arrays A, IROW and ICOL as declared in the (sub)program from which F01XAFP is called.

*Constraint:* on the source processors, as specified by IS and JS,  $LA \geq \max(1, NNZ)$ ; on all other processors,  $LA \geq 1$  and LA must be large enough to store the non-zero entries of the row blocks of the matrix A assigned locally: in other words LA must be larger than or equal to the value returned in NNZ on exit.

**10: IROW(LA) — INTEGER array***Local Input/Local Output***11: ICOL(LA) — INTEGER array***Local Input/Local Output*

*On entry:* on the source processors, as specified by IS and JS, IROW(1:NNZ) and ICOL(1:NNZ) contain the row and column indices of the non-zero entries of the matrix A; on all other processors, the input values of IROW and ICOL are not used.

*Constraint:* on the source processors, as specified by IS and JS,  $1 \leq IROW(i) \leq N$  and  $1 \leq ICOL(i) \leq N$ , for  $i = 1, 2, \dots, NNZ$ . Also, IROW(1:NNZ) and ICOL(1:NNZ) respectively must be identical on all source processors.

*On exit:* if IFAIL = 0, then:

- if WHAT = 'C', then IROW(1:NNZ) and ICOL(1:NNZ) contain the row and column indices, respectively, of the row blocks of the matrix A stored locally;
- if WHAT = 'N', the arrays IROW and ICOL are overwritten on exit, and their contents are indeterminate.

**12: IWORK(\*) — INTEGER array***Local Workspace*

**Note:** the dimension of the array IWORK must be at least N+1 on the source processors, as specified by IS and JS; on all other processors, IWORK is not referenced, but must have dimension at least 1.

**13: 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.

## 5.2 Any Error Checking Mode

IFAIL = 1

LA is too small on at least one processor: there is not enough space to store the non-zero entries of the row blocks of the matrix  $A$  assigned to it.

## 6 Further Comments

### 6.1 Algorithmic Detail

Each source processor is assigned a corresponding set of (non-source) destination processors. It extracts the blocks of  $A$  assigned to different destination processors from the input coordinate storage representation of  $A$  and sends them to each destination processor in turn. The set of destination processors assigned to a source processor is determined as follows:

if  $IS \geq 0$  and  $JS \geq 0$ , then any processor in the Library Grid other than that specified by  $IS$  and  $JS$  is a destination processor associated to it;

if  $IS \geq 0$  and  $JS = -1$ , then any processor in each column of the Library Grid other than that in row  $IS$  is a destination processor associated to it;

if  $IS = -1$  and  $JS \geq 0$ , then any processor in each row of the Library Grid other than that in column  $JS$  is a destination processor associated to it;

if  $IS = -1$  and  $JS = -1$ , then there are no associated destination processors.

Additionally, each source processor extracts the blocks of  $A$  assigned to it from the input coordinate storage representation of  $A$ .

### 6.2 Parallelism Detail

The degree to which the distribution of matrix blocks described in Section 6.1 can be performed in parallel depends on the specific set of source processors:

if  $IS \geq 0$  and  $JS \geq 0$ , then execution is essentially sequential, each destination processor being served in turn;

if  $IS \geq 0$  and  $JS = -1$ , then different columns of the Library Grid can operate in parallel, wholly independently of each other;

if  $IS = -1$  and  $JS \geq 0$ , then different rows of the Library Grid can operate in parallel, wholly independently of each other;

if  $IS = JS = -1$ , then all processors of the Library Grid can operate in parallel, wholly independently of each other.

## 7 References

None.

## 8 Example

To solve the linear system of equations  $Ax = b$  of order  $n = n_x^2$ , where  $n_x$  is a user-specified integer. The unsymmetric sparse coefficient matrix  $A$  is given by the block matrix

$$A = \begin{pmatrix} D & E & 0 & \cdots & \cdots & 0 \\ F & D & E & \ddots & & \vdots \\ 0 & F & D & \ddots & & \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ & & & \ddots & D & E & 0 \\ \vdots & & & \ddots & F & D & E \\ 0 & \cdots & \cdots & 0 & F & D \end{pmatrix},$$

where the matrix blocks  $D$ ,  $E$  and  $F$  of order  $n_x$  are defined in terms of the quantity  $h := (n_x + 1)^{-1}$  as follows:

$$D = \frac{1}{h^2} \begin{pmatrix} 4 & -h-1 & 0 & \cdots & \cdots & 0 \\ h-1 & 4 & -h-1 & \ddots & & \vdots \\ 0 & h-1 & 4 & \ddots & & \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ & & & \ddots & 4 & -h-1 & 0 \\ \vdots & & & \ddots & h-1 & 4 & -h-1 \\ 0 & \cdots & \cdots & 0 & h-1 & 4 \end{pmatrix},$$

$$E = -\frac{h+1}{h^2} I$$

and

$$F = \frac{h-1}{h^2} I,$$

where  $I$  is the identity matrix. The right-hand side vector is given by  $b = 10^2(1, 1, \dots, 1)^T$ .

**Note:** the listing of the Example Program presented below does not give a full pathname for the data file being opened, but in general the user must give the full pathname in this and any other OPEN statement.

### 8.1 Example Text

```
*      F01XAFP 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)
DOUBLE PRECISION DZERO
PARAMETER       (DZERO=0.DO)
*      .. Local Scalars ..
DOUBLE PRECISION OMEGA, RNORM, TOL
INTEGER          I, ICNTXT, IFAIL, IS, ITN, ITNP, JS, LEVEL, M,
+               MAXITN, MB, ML, MP, MYCOL, MYPROC, MYROW, N, NNZ,
+               NP, NUMPROCS, NX
LOGICAL          ROOT, ZGRID
CHARACTER        DUP, KIND, PRECON, SYMM, WHAT, ZERO
```

```

CHARACTER*10    METHOD
CHARACTER*80    FORMAT
*
.. Local Arrays ..
DOUBLE PRECISION A(LA), B(MLMAX), WORK(LWORK), X(MLMAX)
INTEGER         CA(1), IAINFO(200), ICOL(LA), IERR(1), IROW(LA),
+              IWORK(3*MLMAX), RA(1)
*
.. External Functions ..
INTEGER         Z01CFFP
LOGICAL        Z01ACFP
EXTERNAL       Z01CFFP, Z01ACFP
*
.. External Subroutines ..
EXTERNAL       F01CPFP, F01XAFP, F01XEFP, F01XFFP, F11DEFP,
+             F11ZBFP, F11ZZFP, GMAT, GVEC, PRINTI, Z01AAFP,
+             Z01ABFP, Z01BBFP, Z01ZAFP, Z02EAFP
*
.. Executable Statements ..
ROOT = Z01ACFP()
IF (ROOT) WRITE (NOUT,*) 'F01XAFP Example Program Results'
*
*
*   Open input file on all processors
*
*   OPEN (NIN,FILE='f01xafpe.d')
*
*   Skip heading in data file
*
*   READ (NIN,*)
*   READ (NIN,*) MP, NP
*
*   Read the problem parameters
*
*   READ (NIN,*) NX
*
*   Read the algorithmic parameters
*
*   READ (NIN,*) METHOD
*   READ (NIN,*) PRECON, OMEGA, ITNP
*   READ (NIN,*) M
*   READ (NIN,*) TOL, MAXITN
*   READ (NIN,*) FORMAT
*   READ (NIN,*) LEVEL
*
*   Close input file
*
*   CLOSE (NIN)
*
*   Set up remaining problem parameters
*   N - Order of matrix A
*   MB - Block size for distribution
*
*   N = NX*NX
*   IF (N.GT.MLMAX) THEN
*       IERR(1) = 1
*       GO TO 20
*   END IF
*   MB = (N+MP*NP-1)/(MP*NP)
*
*   Initialize Library Grid
*
*   IFAIL = 0

```

```

CALL Z01AAFP(ICNTXT,MP,NP,IFAIL)
*
* Check whether processor is in Library Grid
*
CALL Z01BBFP(ICNTXT,ZGRID,IFAIL)
IF ( .NOT. ZGRID) GO TO 60
*
* Calculate myproc number for later use
*
CALL Z01ZAFP(ICNTXT,MP,NP,MYROW,MYCOL)
NUMPROCS = MP*NP
MYPROC = MYROW*NP + MYCOL
*
* Set error checking level
*
CALL Z02EAFP(ICNTXT,LEVEL,IFAIL)
*
IS = 0
JS = 0
*
* Generate sparse matrix on the (IS,JS) processor
*
IERR(1) = 0
IF (MYROW.EQ.IS .AND. MYCOL.EQ.JS) THEN
  CALL GMAT(N,NX,NNZ,A,LA,IROW,ICOL)
*
* Check whether number of non-zero entries is less than LA
*
  IF (NNZ.GT.LA) THEN
    IERR(1) = 2
    GO TO 20
  END IF
*
* Generate right-hand side vector on the (IS,JS) processor
*
  CALL GVEC(N,X)

END IF

20 IFAIL = 0
*
* Check that IERR is 0 on all processors
*
CALL F01CPFP(ICNTXT,'X','All',1,1,IERR,1,RA,CA,1,0,-1,-1,IFAIL)
IF (IERR(1).NE.0) THEN
  IF (ROOT) THEN
    IF (IERR(1).EQ.1) WRITE (NOUT,99996)
    IF (IERR(1).EQ.2) WRITE (NOUT,99995)
  END IF
  GO TO 60
END IF
*
* Distribute sparse matrix
* Set WHAT to C so that numerical values and co-ordinates are
* distributed.
*
WHAT = 'C'
CALL F01XAFP(ICNTXT,WHAT,IS,JS,N,MB,NNZ,A,LA,IROW,ICOL,IWORK,

```

```

+           IFAIL)
*
*   Set up auxiliary data for subsequent operations
*
  SYMM = 'S'
  KIND = 'N'
  DUP = 'F'
  ZERO = 'R'
  CALL F11ZBFP(ICNTXT,N,MB,NNZ,A,IROW,ICOL,DUP,ZERO,SYMM,KIND,
+           IAINFO,LIA,IFAIL)
*
*   Find how many local rows on a processor
*
  ML = Z01CFFP(NUMPROCS,N,MYPROC)
*
*   Distribute right-hand side vector
*
  CALL F01XEFP(ICNTXT,IS,JS,N,X,B,IAINFO,IWORK,WORK,IFAIL)
*
*   Print a summary of input parameters and options
*
  IF (ROOT) CALL PRINTI(NOUT,METHOD,PRECON,N,MAXITN,TOL,M,OMEGA,
+           ITNP,MP,NP,MB)
*
*   Set initial approximation to solution locally
*
  DO 40 I = 1, ML
    X(I) = DZERO
40 CONTINUE
*
*   Solve equations
*
  CALL F11DEFP(ICNTXT,METHOD,PRECON,N,NNZ,A,IROW,ICOL,OMEGA,ITNP,B,
+           M,TOL,MAXITN,X,RNORM,ITN,IAINFO,WORK,LWORK,IWORK,
+           IFAIL)
*
*   Gather solution vector to (IS,JS) processor
*
  CALL F01XFFP(ICNTXT,IS,JS,N,X,B,IAINFO,IWORK,WORK,IFAIL)
*
*   Produce a report
*
  IF (MYROW.EQ.0 .AND. MYCOL.EQ.0) 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
    WRITE (NOUT,'(/1X,'Solution vector'/1X,15(' '-'))')
    WRITE (NOUT,FORMAT) (B(I),I=1,N)
  END IF
*
*   Release internally allocated memory
*
  CALL F11ZZFP(ICNTXT,IAINFO,IFAIL)
*
*   Finalize Library Grid

```



```

*
60 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')
99996 FORMAT (1X,'** ERROR: Number of rows on (IS,JS) processor too ',
+           'large')
99995 FORMAT (1X,'** ERROR: Number of non-zero entries on (IS,JS) ',
+           'processor too large')
END
*
*****
*****
*
SUBROUTINE PRINTI(NOUT,METHOD,PRECON,N,MAXITN,TOL,M,OMEGA,ITNP,MP,
+               NP,MB)
*
Prints a summary of the input parameters and options.
*
*
*   .. Scalar Arguments ..
DOUBLE PRECISION OMEGA, TOL
INTEGER          ITNP, M, MAXITN, MB, MP, N, NOUT, NP
CHARACTER        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)
+ 'Smoothing Parameter (OMEGA)                         -', OMEGA
WRITE (NOUT,99997)
+ 'Number of inner iterations                          -', ITNP
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
*
*****
*****
*
      SUBROUTINE GMAT(N,NX,NNZL,AL,LAL,IROWL,ICOLL)
*
*
*   .. Scalar Arguments ..
      INTEGER          LAL, N, NNZL, NX
*
*   .. Array Arguments ..
      DOUBLE PRECISION AL(LAL)
      INTEGER          ICOLL(LAL), IROWL(LAL)
*
*   .. Local Scalars ..
      DOUBLE PRECISION E, F, H, H2, H24
      INTEGER          BLOCK, I, INZL, J, MATRIX
      LOGICAL          BLKBND, MATBND
*
*   .. Intrinsic Functions ..
      INTRINSIC       DBLE, MOD
*
*   .. Executable Statements ..
*
      This routine creates the matrix A
      where A is defined as
*
*           | D E 0 0 0 0 0 |
*           | F D E 0 0 0 0 |
*           | 0 F D E 0 0 0 |
*           | 0 0 F D E 0 0 |
*           | 0 0 0 F D E 0 |
*           | 0 0 0 0 F D E |
*           | 0 0 0 0 0 F D |
*
*
*   F and E are diagonal nx x nx matrices, and
*   D is a tridiagonal nx x nx matrix.
*
*
      NNZL = 0
      H = 1.0D0/DBLE(NX+1)
      H2 = H*H
      H24 = 4/H2
      E = (H-1.D+0)/H2
      F = -(H+1.D+0)/H2
      DO 40 I = 1, N
*
*           INZL = NNZL
*
*   Calculate number of non-zero elements in I-th row

```

```

*      Depends if on boundary of a block and/or of matrix
*
      MATRIX = 1 + MOD(I-1,NX)
      BLOCK = 1 + (I-1)/NX
      MATBND = (MATRIX.EQ.1) .OR. (MATRIX.EQ.NX)
      BLKBND = (BLOCK.EQ.1) .OR. (BLOCK.EQ.NX)
      IF (MATBND .AND. BLKBND) THEN
          NNZL = NNZL + 3
      ELSE IF ( .NOT. (MATBND .OR. BLKBND)) THEN
          NNZL = NNZL + 5
      ELSE
          NNZL = NNZL + 4
      END IF

*
*      Check whether there is sufficient storage space
*
      IF (NNZL.GT.LAL) GO TO 40

*
*      Set non-zero elements in I-th row
*
      DO 20 J = INZL + 1, NNZL
          IROWL(J) = I
20    CONTINUE

*
*      Set up diagonal elements of D first
*
      INZL = INZL + 1
      ICOLL(INZL) = I
      AL(INZL) = H24

*
*      Now off-diagonal elements of D
*
      IF (MATRIX.GT.1) THEN
          INZL = INZL + 1
          ICOLL(INZL) = I - 1
          AL(INZL) = E
      END IF
      IF (MATRIX.LT.NX) THEN
          INZL = INZL + 1
          ICOLL(INZL) = I + 1
          AL(INZL) = F
      END IF

*
*      Now add the E and F diagonal blocks
*
      IF (BLOCK.GT.1) THEN
          INZL = INZL + 1
          ICOLL(INZL) = I - NX
          AL(INZL) = E
      END IF
      IF (BLOCK.LT.NX) THEN
          INZL = INZL + 1
          ICOLL(INZL) = I + NX
          AL(INZL) = F
      END IF

*
      NNZL = INZL
*

```

```

40 CONTINUE
*
*   End of subroutine GMAT
*
      RETURN
      END
*
*****
*****
*
      SUBROUTINE GVEC(N,X)
*
*
*   .. Scalar Arguments ..
      INTEGER          N
*
*   .. Array Arguments ..
      DOUBLE PRECISION X(*)
*
*   .. Local Scalars ..
      INTEGER          I
*
*   .. Executable Statements ..
      DO 20 I = 1, N
          X(I) = 1.D+2
20 CONTINUE
*
*   End of subroutine GVEC
*
      RETURN
      END

```

## 8.2 Example Data

```

F01XAFP Example Program Data
  2    2                :  MP, NP
  8                :  NX
'BICGSTAB'         :  METHOD
'J', 0.2d0, 3      :  PRECON, OMEGA, ITNP
  2                :  M
1.0D-06 100        :  TOL, MAXITN
'(8F8.4)'          :  FORMAT
  0                :  LEVEL

```

## 8.3 Example Results

F01XAFP 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) - BICGSTAB
Use the preconditioner (PRECON) - J
Smoothing Parameter (OMEGA) - 2.00D-01

```

Number of inner iterations	-	3
Tolerance (TOL)	-	1.00D-06
Maximum number of iterations allowed (MAXITN)	-	100
Order of BICGSTAB method (M)	-	2

## Summary of results

-----

Number of iterations carried out (ITN)	-	10
Residual norm (RNORM)	-	3.97D-04

## Solution vector

-----

2.0231	3.0860	3.5805	3.6951	3.5227	3.1002	2.4225	1.4364
3.0860	4.7996	5.6149	5.8043	5.5143	4.8090	3.6933	2.1219
3.5805	5.6149	6.5941	6.8219	6.4696	5.6165	4.2777	2.4241
3.6951	5.8043	6.8219	7.0595	6.6941	5.8076	4.4167	2.4959
3.5227	5.5143	6.4696	6.6941	6.3566	5.5311	4.2250	2.4014
3.1002	4.8090	5.6165	5.8076	5.5311	4.8443	3.7394	2.1571
2.4225	3.6933	4.2777	4.4167	4.2250	3.7394	2.9382	1.7417
1.4364	2.1219	2.4241	2.4959	2.4014	2.1571	1.7417	1.0827