F11ZUFP

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

F11ZUFP generates a multi-colour ordering for an n by n complex sparse matrix A with **symmetric sparsity pattern** (see Section 2.4.2 of the F11 Chapter Introduction) stored in coordinate storage format, distributed in cyclic row block form (see Section 2.5 of the F11 Chapter Introduction).

The colouring generated may alternatively be valid (i) for both the internal and the interface part of A or (ii) for the interface part of A only. The information generated by F11ZUFP is stored in the array IAINFO and can be used in subsequent calls to the (S)SOR preconditioning routine F11DRFP, or Black Box routine F11DSFP. F11ZUFP must be called before F11DRFP unless A has the same pattern of non-zero entries as a matrix previously processed by F11ZUFP. For such matrices only the non-zero entries of A need to be reordered, and F11YNFP should be used to perform this task.

The matrix A must be structurally symmetric, and the routine F11ZPFP must be called with SYMM = 'S' prior to F11ZUFP to set up auxiliary information about the sparse matrix A in the array IAINFO.

2 Specification

```
 SUBROUTINE F11ZUFP(ICNTXT, N, NNZ, A, IROW, ICOL, COLOR, NCOLOR,
1                   ICOLOR, IAINFO, LIA, IFAIL)
 INTEGER            ICNTXT, N, NNZ, IROW(*), ICOL(*), NCOLOR,
1                   ICOLOR(*), IAINFO(*), LIA, IFAIL
 COMPLEX*16         A(*)
 CHARACTER*1        COLOR
```

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.
m_l	–	the number of rows of the matrix assigned to the calling processor (m_l = IAINFO(3), 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: N, COLOR, IFAIL

Global output arguments: NCOLOR, IFAIL

The remaining arguments are local.

3.3 Distribution Strategy

The matrix A, of symmetric sparsity pattern, 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.

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 complex sparse matrices in cyclic row block form:

Complex sparse matrix generation: F01YPFP or F01YQFP

Complex sparse matrix distribution: F01XPFP

F11ZUFP must be called before using the (S)SOR preconditioning routine F11DRFP and Black Box routine F11DSFP.

3.5 Requisites

The complex sparse matrix A must have a symmetric sparsity pattern.

The complex sparse matrix A, of symmetric sparsity pattern, must have been preprocessed to set up the auxiliary information vector IAINFO by F11ZPFP with input parameter SYMM = 'S'.

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: 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 F11ZPFP in which the array IAINFO was initialised.

Constraint: N ≥ 1.

3: NNZ — INTEGER *Local Input*

On entry: the number of non-zero entries in the matrix A stored on the calling processor. It must contain the same value as the parameter NNZ returned from a prior call to F11ZPFP in which the array IAINFO was initialised.

Constraint: NNZ > 0.

4: 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 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 F11ZPFP.

On exit: the local non-zero entries reordered within each row to enable subsequent preconditioning operations using F11DRFP.

5: IROW(∗) — INTEGER array *Local Input*

6: ICOL(∗) — INTEGER array *Local Input/Local Output*

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 the array A.

On exit: the local row and column indices of the non-zero entries supplied in the array A. The contents of the arrays IROW and ICOL **must not** be changed between successive calls to library routines involving the matrix A.

7: COLOR — CHARACTER*1 *Global Input*

On entry: specifies the colouring strategy to be employed as follows:

if COLOR = 'B', then the colouring is performed automatically by F11ZUFP. The colouring generated is valid for the interface part of A only;
if COLOR = 'C', then the colouring is performed automatically by F11ZUFP. The colouring generated is valid for both the internal and the interface part of A;
if COLOR = 'U', then the colouring is performed according to the user-specified input values in ICOLOR.

Suggested value: COLOR = 'B' if F11ZUFP is used in conjunction with F11DRFP (see Section 6.1).

Constraint: COLOR = 'B', 'C' or 'U'.

8: NCOLOR — INTEGER *Global Output*

On exit: the number of colours used in the multi-colour ordering. If there are no coloured indices, i.e., if COLOR = 'B' and there are no interface indices, then NCOLOR is set to 1.

9: ICOLOR(∗) — INTEGER array *Local Input/Local Output*

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

On entry: if COLOR = 'U', then ICOLOR(i), for $i = 1, \ldots, m_l$, must specify an integer representing the colour of the local index i.

Constraint: if COLOR = 'U', then ICOLOR(i) ≥ 1, for $i = 1, \ldots, m_l$, and the colouring represented by ICOLOR must be valid for both the internal and the interface part of A, i.e.,

$$\text{ICOLOR}(\text{IROW}(l)) \neq \text{ICOLOR}(\text{ICOL}(l)) \quad \text{if} \quad \text{IROW}(l) \neq \text{ICOL}(l)$$

for $l = 1, \ldots,$NNZ.

On exit: if COLOR = 'B' or 'C', ICOLOR represents the colouring generated by F11ZUFP; otherwise, ICOLOR is not changed. Depending on the value of the argument COLOR, valid colours may be assigned (i) to both internal and interface indices or (ii) to interface indices only. If ICOLOR(i) ≥ 1, for $i = 1, \ldots, m_l$, the ith index has been assigned a valid colour; ICOLOR(i) ≤ 0 indicates that the ith index has not been assigned a valid colour.

10: IAINFO(∗) — INTEGER array *Local Input/Local Output*

Note: the dimension of the array IAINFO must be at least max(200,IAINFO(2),LIA).

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 F11ZPFP.

On exit: auxiliary information about the matrix A including information about the multi-colour ordering of A. IAINFO(1) contains the minimum required value of LIA. The first IAINFO(2) elements of IAINFO contain information required by other library routines and therefore must not be changed between successive calls to library routines involving the matrix A. See Section 3.3 of the F11 Chapter Introduction

Note: if, on exit, IFAIL = 3, IAINFO(1) may be **smaller** than the minimum required value of LIA (when LIA > 0), and modifying the code by setting LIA to the returned value of IAINFO(1) may still cause failure; if, on exit, IFAIL = 4 (when LIA = −1), then an error occurred in the internal memory allocation mechanism.

11: LIA — INTEGER *Local Input*

Note: if LIA is set to -1, F11ZUFP will allocate the memory required for storing the auxiliary information about the matrix A (see Section 6.2). In this case, LIA is treated as a **global** input, i.e., LIA must be equal to -1 on all processors.

On entry: if LIA $\neq -1$, the dimension of the array IAINFO as declared in the (sub)program from which F11ZUFP is called.

Suggested value: LIA $= -1$ unless the required value of LIA was determined by a prior call to F11ZUFP.

Constraint: LIA $= -1$ or LIA $\geq$ max(200,IAINFO(2)). The choice between these two options is determined by the choice made at the call to F11ZPFP.

12: 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 initialised by a prior call to F11ZPFP.

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 initialised by a call to F11ZPFP, at least one of these arguments was changed between successive calls to library routines.

IFAIL $= 3$

LIA is too small, resulting in insufficient space to store the required auxiliary information in the array IAINFO.

IFAIL $= 5$

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

5.2 Any Error Checking Mode

IFAIL = 4

An error in the internal memory allocation mechanism occurred. This is usually due to insufficent memory resources.

6 Further Comments

6.1 Colouring Strategies

It is anticipated that the colourings generated by F11ZUFP will be used mainly in conjunction with the (S)SOR preconditioning routine F11DRFP. The reorderings P of the matrix A, of symmetric sparsity pattern, induced by valid colourings allow the (S)SOR iterative method for the permuted system PAP^T to proceed in NCOLOR consecutive computational steps, each of which can be performed in parallel. Colourings generated with COLOR = 'C' **maximize** the degree of parallelism within each step; whereas with COLOR = 'B', the degree of parallelism within each step is chosen to be **equal to the number of processors** in the Library Grid. Colourings generated using COLOR = 'B' are usually preferable to colourings generated using COLOR = 'C' since they generally require smaller numbers of colours, and hence smaller numbers of consecutive computational steps, and often lead to preconditioners of better quality.

6.2 Releasing Internally Allocated Resources

F11ZUFP will allocate the memory required for storing the auxiliary information about the matrix A if LIA is globally set to -1 on entry. When this information is no longer needed, the allocated memory should be released by a call to F11ZZFP.

7 References

[1] 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:

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

for $(x, y) \in \Omega = (0, 1)^2$, where c_i, $i = 1, \ldots, 5$ are given complex 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

```
*     F11ZUFP 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)
      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 ANORM, ENORM, ENORML, OMEGA, SIGMAX, STPLHS,
     +                 STPRHS, TOL
      INTEGER          I, ICNTXT, IFAIL, IREVCM, ITERM, ITN, IW, J, LW,
     +                 LWREQ, M, MAXITN, MB, ML, MLOMAX, MONIT, MP, N,
     +                 NCOLOR, NINTE, NINTI, NITS, NNZ, NP
      LOGICAL          LOOP, ROOT, ZGRID
      CHARACTER        CHECK, COLOR, DISTR, DUP, INVDIA, KIND, NORM,
     +                 PRECON, PRECON1, SYMM, WEIGHT, ZERO
      CHARACTER*10     METHOD
      CHARACTER*80     FORMAT
*     .. Local Arrays ..
      COMPLEX*16       A(LA), RDIAG(MLMAX), TS(MLMAX), U(MLMAX),
     +                 V(MLMAX), WORK(LWORK)
      DOUBLE PRECISION WGHTS(MLMAX)
      INTEGER          CA(1), IAINFO(200), ICOL(LA), ICOLOR(MLMAX),
     +                 IERR(1), IROW(LA), RA(1)
*     .. External Functions ..
      LOGICAL          Z01ACFP
      EXTERNAL         Z01ACFP
*     .. External Subroutines ..
      EXTERNAL         DGERV2D, DGESD2D, F01CPFP, F01YPFP, F01YTFP,
     +                 F11BRFP, F11BSFP, F11BTFP, F11DRFP, F11DXFP,
     +                 F11XPFP, F11ZPFP, F11ZUFP, F11ZZFP, GMAT, GSOL,
     +                 GVEC, PRINTI, X04YPFP, Z01AAFP, Z01ABFP, Z01BBFP
*     .. Intrinsic Functions ..
      INTRINSIC        ABS, MAX
*     .. Common blocks ..
      COMMON           /PROB/C1, C2, C3, C4, C5, NX
*     .. Executable Statements ..
      ROOT = Z01ACFP()
      IF (ROOT) WRITE (NOUT,*) 'F11ZUFP Example Program Results'
*
*     Open input file on all processors
*
      OPEN (NIN,FILE='f11zufpe.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, NORM, ITERM, MONIT
      READ (NIN,*) M
      READ (NIN,*) TOL, MAXITN
      READ (NIN,*) COLOR
      READ (NIN,*) NITS, OMEGA
      READ (NIN,*) FORMAT
*
*     Read complex coefficients in PDE
*
      READ (NIN,*) C1, C2, C3, C4, C5
*
*     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 140
*
*     Generate sparse matrix
*
      MB = (N+MP*NP-1)/(MP*NP)
      CALL F01YPFP(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 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,99996)
         GO TO 120
      END IF
*
*     Generate right-hand side vector
*
      CALL F01YTFP(ICNTXT,GVEC,N,V,IAINFO,IFAIL)
*
*     Generate multi-color ordering for (S)SOR preconditioner
```

```
*
      IF (PRECON.EQ.'F' .OR. PRECON.EQ.'B' .OR. PRECON.EQ.'S')
     +    CALL F11ZUFP(ICNTXT,N,NNZ,A,IROW,ICOL,COLOR,NCOLOR,ICOLOR,
     +                 IAINFO,LIA,IFAIL)
*
*     Initialize solver suite
*
      PRECON1 = 'P'
      IF (PRECON.EQ.'N') PRECON1 = 'N'
      WEIGHT = 'N'
      DISTR = 'A'
      ANORM = 0.D0
      SIGMAX = 0.D0
      CHECK = 'N'
      CALL F11BRFP(ICNTXT,METHOD,PRECON1,NORM,DISTR,WEIGHT,ITERM,N,ML,M,
     +             TOL,MAXITN,ANORM,SIGMAX,MONIT,LW,IFAIL)
*
*     Check workspace size
*
      NINTI = IAINFO(6)
      NINTE = IAINFO(7)
      MLOMAX = IAINFO(5)
      IERR(1) = 0
      LWREQ = LW + MAX(NINTI,NINTE)
      IF (ROOT) THEN
         LWREQ = MAX(LWREQ,LW+ML+MLOMAX)
      ELSE
         LWREQ = MAX(LWREQ,LW+2*ML)
      END IF
      IF (PRECON.EQ.'J') THEN
         LWREQ = MAX(LWREQ,LW+ML+MAX(NINTI,NINTE))
      ELSE IF (PRECON.EQ.'F' .OR. PRECON.EQ.'B' .OR. PRECON.EQ.'S') THEN
         LWREQ = MAX(LWREQ,LW+NINTE+MAX(NINTI,NINTE))
      END IF
      IF (LWREQ.GT.LWORK) 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
         WRITE (NOUT,99995)
         GO TO 120
      END IF
*
*     Print summary of input parameters and options
*
      IF (ROOT) CALL PRINTI(NOUT,METHOD,PRECON,NORM,DISTR,ITERM,N,
     +                      MAXITN,TOL,M,MONIT,COLOR,OMEGA,NITS,NCOLOR,
     +                      MP,NP,MB)
*
*     Set initial approximation to solution
*
      DO 20 I = 1, ML
         U(I) = ZZERO
   20 CONTINUE
*
*     Solve equations using reverse communication scheme
*
      INVDIA = 'C'
      IW = LW + 1
      IREVCM = 0
```

```
      LOOP = .TRUE.
*
   40 CONTINUE
      CALL F11BSFP(ICNTXT,IREVCM,U,V,WGHTS,WORK,LW,IFAIL)
*
      IF (IREVCM.LE.-1) THEN
*
*     Compute  v = A**T * u (used only to compute ||A||)
*
         CALL F11XPFP(ICNTXT,'Transpose',N,NNZ,A,IROW,ICOL,CHECK,U,V,
     +               IAINFO,WORK(IW),IFAIL)
*
      ELSE IF (IREVCM.EQ.1) THEN
*
*     Compute  v = A * u
*
         CALL F11XPFP(ICNTXT,'No transpose',N,NNZ,A,IROW,ICOL,CHECK,U,V,
     +               IAINFO,WORK(IW),IFAIL)
*
      ELSE IF (IREVCM.EQ.2) THEN
*
*     Perform preconditioning step
*
         IF (PRECON.EQ.'J') THEN
            CALL F11DXFP(ICNTXT,NITS,N,NNZ,A,IROW,ICOL,INVDIA,RDIAG,
     +                   OMEGA,U,V,IAINFO,WORK(IW),IFAIL)
         ELSE
            CALL F11DRFP(ICNTXT,PRECON,NITS,N,NNZ,A,IROW,ICOL,INVDIA,
     +                   RDIAG,OMEGA,U,V,IAINFO,WORK(IW),IFAIL)
         END IF
         INVDIA = 'U'
*
      ELSE IF (IREVCM.EQ.3) THEN
*
*     Monitoring
*
         CALL F11BTFP(ICNTXT,ITN,STPLHS,STPRHS,ANORM,SIGMAX,IFAIL)
         IF (ROOT) THEN
            WRITE (NOUT,'(/1X,''Monitoring step''/1X,15(''-'')/)')
            WRITE (NOUT,99999)
     +        'Number of iterations carried out (ITN)              -',
     +        ITN
            WRITE (NOUT,99997)
     +        'Left-hand side of termination criterion (STPLHS)    -',
     +        STPLHS
            WRITE (NOUT,99997)
     +        'Right-hand side of termination criterion (STPRHS)   -',
     +        STPRHS
            IF (ITERM.EQ.2) WRITE (NOUT,99998)
     +          'Largest singular value of (precond.) matrix (SIGMAX) -'
     +          , SIGMAX
            WRITE (NOUT,'(/1X,''Monitoring at iteration no. '',I3)')
     +        ITN
            WRITE (NOUT,
     +        '(/1X,''Solution vector (last iterate)''/1X,30(''-'')/)')
         END IF
         CALL X04YPFP(ICNTXT,NOUT,N,U,FORMAT,IAINFO,WORK(IW),IFAIL)
         IF (ROOT) WRITE (NOUT,
```

```
     +       '(/1X,''Residual vector (last iterate)''/1X,30(''-'')/)')
          CALL X04YPFP(ICNTXT,NOUT,N,V,FORMAT,IAINFO,WORK(IW),IFAIL)
*
      ELSE IF (IREVCM.EQ.4) THEN
*
*     Termination
*
         LOOP = .FALSE.
      END IF
      IF (LOOP) GO TO 40
*
*     Generate true solution TS and error on local part of mesh
*
      CALL F01YTFP(ICNTXT,GSOL,N,TS,IAINFO,IFAIL)
      ENORML = 0.D0
      DO 60 I = 1, IAINFO(3)
         ENORML = MAX(ENORML,ABS(TS(I)-U(I)))
   60 CONTINUE
      IF ( .NOT. ROOT) CALL DGESD2D(ICNTXT,1,1,ENORML,1,0,0)
*
*     Get information about final solution
*
      CALL F11BTFP(ICNTXT,ITN,STPLHS,STPRHS,ANORM,SIGMAX,IFAIL)
*
*     Produce final report
*
      IF (ROOT) THEN
         WRITE (NOUT,'(/1X,''Summary of results''/1X,18(''-'')/)')
         WRITE (NOUT,99999)
     +     'Number of iterations carried out (ITN)               -', ITN
         WRITE (NOUT,99997)
     +     'Left-hand side of termination criterion (STPLHS)     -',
     +     STPLHS
         WRITE (NOUT,99997)
     +     'Right-hand side of termination criterion (STPRHS)    -',
     +     STPRHS
         IF (ITERM.EQ.1) THEN
            WRITE (NOUT,99998)
     +        'Norm of the matrix of the coefficients (ANORM)       -',
     +        ANORM
         ELSE
            WRITE (NOUT,99998)
     +        'Largest singular value of (precond.) matrix (SIGMAX) -',
     +        SIGMAX
         END IF
*
*     Receive local error norms and calculate global error norm
*
         ENORM = ENORML
         DO 100 I = 1, MP
            DO 80 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
   80       CONTINUE
  100    CONTINUE
         WRITE (NOUT,*)
```

```
         WRITE (NOUT,99998) 'Error norm =', ENORM
*
         WRITE (NOUT,'(/1X,''Solution vector''/1X,15(''-'')/)')
      END IF
      CALL X04YPFP(ICNTXT,NOUT,N,U,FORMAT,IAINFO,WORK(IW),IFAIL)
*
*     Release internally allocated memory if necessary
*
  120 IF (LIA.EQ.-1) CALL F11ZZFP(ICNTXT,IAINFO,IFAIL)
*
*     Finalize Library Grid
*
  140 CALL Z01ABFP(ICNTXT,'N',IFAIL)
*
*     End of example program
*
      STOP
*
99999 FORMAT (1X,A,I5)
99998 FORMAT (1X,A,2X,1P,E12.4)
99997 FORMAT (1X,A,3X,1P,D9.2)
99996 FORMAT (1X,'** ERROR: Number of rows per processor too large')
99995 FORMAT (1X,'** ERROR: LWORK too small')
      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_x + c4*w_y + c5*w = f
*
*     where the ci are complex coefficients.
*     The right-hand side vector is set up in the
*     routine GVEC.
*
*     .. Scalar Arguments ..
      INTEGER          I1, I2, LAL, N, NNZL
*     .. 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
*     .. 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.D0/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
         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 ..
      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.D0/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.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 ..
      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, DCMPLX, SIN
*     .. Executable Statements ..
      W = DCMPLX(SIN(X),X*X-2*Y*Y)
      WX = DCMPLX(COS(X),2*X)
      WY = DCMPLX(0.0D0,-4*Y)
      WXX = DCMPLX(-SIN(X),2.0D0)
      WYY = DCMPLX(0.0D0,-4.0D0)
*
      RETURN
      END


      SUBROUTINE PRINTI(NOUT,METHOD,PRECON,NORM,DISTR,ITERM,N,MAXITN,
     +                  TOL,M,MONIT,COLOR,OMEGA,NITS,NCOLOR,MP,NP,MB)
*
*     Prints a summary of the input parameters and options.
*
*
*     .. Scalar Arguments ..
      DOUBLE PRECISION  OMEGA, TOL
      INTEGER           ITERM, M, MAXITN, MB, MONIT, MP, N, NCOLOR,
     +                  NITS, NOUT, NP
      CHARACTER         COLOR, DISTR, NORM, 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,99998)
     +  'Matrix and vector norm in use (NORM)                 -', NORM
      WRITE (NOUT,99998)
     +  'Distribution of vectors (DISTR)                      -', DISTR
      WRITE (NOUT,99997)
     +  'Termination criterion (ITERM)                        -', ITERM
      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
      WRITE (NOUT,99997)
     +  'Monitoring frequency (MONIT)                         -', MONIT
      IF (PRECON.EQ.'F' .OR. PRECON.EQ.'B' .OR. PRECON.EQ.'S') THEN
         WRITE (NOUT,99998)
     +     'Use the coloring (COLOR)                             -',
     +     COLOR
         WRITE (NOUT,99997)
     +     'Number of colors (NCOLOR)                            -',
     +     NCOLOR
      END IF
      IF (PRECON.EQ.'J' .OR. PRECON.EQ.'F' .OR. PRECON.EQ.'B' .OR.
     +    PRECON.EQ.'S') THEN
         WRITE (NOUT,99996)
     +     'Relaxation parameter (OMEGA)                         -',
     +     OMEGA
         WRITE (NOUT,99997)
     +     'Number of preconditioner iterations (NITS)           -',
     +     NITS
      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

```
F11ZUFP Example Program Data
    2    2                        :  MP, NP
    8                             :  NX
  'BICGSTAB'                        :  METHOD
  'S'  'I'    1    0                :  PRECON, NORM, ITERM, MONIT
    2                             :  M
   1.0D-9   100                   :  TOL, MAXITN
  'B'                             :  COLOR
    1   1.0D+0                    :  NITS, OMEGA
'(4(:,'' ('',F7.4,'','',F7.4,'')''))'  :  FORMAT
  (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
```

8.3 Example Results

```
 F11ZUFP 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)                      -    S
 Matrix and vector norm in use (NORM)                 -    I
 Distribution of vectors (DISTR)                      -    A
 Termination criterion (ITERM)                        -    1
 Tolerance (TOL)                                      -    1.00D-09
 Maximum number of iterations allowed (MAXITN)        -  100
 Order of BICGSTAB method (M)                         -    2
 Monitoring frequency (MONIT)                         -    0
 Use the coloring (COLOR)                             -    B
 Number of colors (NCOLOR)                            -    2
 Relaxation parameter (OMEGA)                         -    1.00D+00
 Number of preconditioner iterations (NITS)           -    1

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

 Number of iterations carried out (ITN)               -   31
 Left-hand side of termination criterion (STPLHS)     -    4.07D-08
 Right-hand side of termination criterion (STPRHS)    -    2.46D-06
 Norm of the matrix of the coefficients (ANORM)       -    1.1394E+03

 Error norm =    3.0066E-04
```

```
Solution vector
---------------

( 0.1109,-0.0124) ( 0.2204, 0.0246) ( 0.3272, 0.0863) ( 0.4299, 0.1727)
( 0.5274, 0.2838) ( 0.6183, 0.4197) ( 0.7017, 0.5802) ( 0.7764, 0.7654)
( 0.1108,-0.0865) ( 0.2203,-0.0495) ( 0.3271, 0.0122) ( 0.4299, 0.0986)
( 0.5273, 0.2097) ( 0.6183, 0.3455) ( 0.7016, 0.5061) ( 0.7763, 0.6913)
( 0.1108,-0.2100) ( 0.2203,-0.1730) ( 0.3271,-0.1113) ( 0.4298,-0.0249)
( 0.5273, 0.0862) ( 0.6183, 0.2220) ( 0.7016, 0.3826) ( 0.7763, 0.5678)
( 0.1108,-0.3828) ( 0.2203,-0.3459) ( 0.3270,-0.2842) ( 0.4298,-0.1978)
( 0.5273,-0.0867) ( 0.6182, 0.0492) ( 0.7016, 0.2097) ( 0.7763, 0.3950)
( 0.1108,-0.6051) ( 0.2203,-0.5681) ( 0.3270,-0.5064) ( 0.4298,-0.4200)
( 0.5273,-0.3089) ( 0.6182,-0.1730) ( 0.7016,-0.0125) ( 0.7763, 0.1728)
( 0.1108,-0.8766) ( 0.2203,-0.8397) ( 0.3271,-0.7780) ( 0.4298,-0.6916)
( 0.5273,-0.5804) ( 0.6183,-0.4446) ( 0.7016,-0.2841) ( 0.7763,-0.0988)
( 0.1108,-1.1976) ( 0.2203,-1.1606) ( 0.3271,-1.0989) ( 0.4299,-1.0125)
( 0.5273,-0.9014) ( 0.6183,-0.7656) ( 0.7017,-0.6050) ( 0.7763,-0.4198)
( 0.1109,-1.5680) ( 0.2204,-1.5309) ( 0.3272,-1.4692) ( 0.4299,-1.3828)
( 0.5274,-1.2717) ( 0.6183,-1.1359) ( 0.7017,-0.9754) ( 0.7764,-0.7902)
```