F11GAFP

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. In particular, some of the notation and terminology used in this document was introduced in Section 2.1 of the F11 Chapter Introduction.

1 Description

The suite consisting of the routines F11GAFP, F11GBFP and F11GCFP is designed to solve the symmetric system of simultaneous linear equations $Ax = b$ of order n, where n is large and the coefficient matrix A is sparse.

F11GAFP is a set-up routine which must be called before F11GBFP, the iterative solver. The third routine in the suite, F11GCFP, can be used to return additional information about the computation. A choice of methods is available:

conjugate gradient method (CG), suitable for positive-definite symmetric matrices;

SYMMLQ, suitable for both positive-definite and indefinite symmetric matrices, although less efficient than the CG method when A is positive-definite.

It is recommended that the user should read Section 6, before proceeding to use this routine for the first time.

2 Specification

```
 SUBROUTINE F11GAFP(ICNTXT, METHOD, PRECON, SIGCMP, NORM, DISTR,
1                   WEIGHT, ITERM, N, NLOC, TOL, MAXITN, ANORM,
2                   SIGMAX, SIGTOL, MAXITS, MONIT, LWREQ, IFAIL)
 DOUBLE PRECISION   TOL, ANORM, SIGMAX, SIGTOL
 INTEGER            ICNTXT, ITERM, N, NLOC, MAXITN, MAXITS, MONIT,
1                   LWREQ, IFAIL
 CHARACTER*1        PRECON, SIGCMP, NORM, DISTR, WEIGHT
 CHARACTER*(*)      METHOD
```

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.
$n_l(i,j)$	–	the number of elements of the distributed vectors stored locally on the processor at location $\{i,j\}$ of the Library Grid.

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, SIGCMP, NORM, DISTR, WEIGHT, ITERM, N, TOL, MAXITN, ANORM, SIGMAX, SIGTOL, MAXITS, MONIT, IFAIL
Global output arguments:	IFAIL

The remaining arguments are local.

3.3 Distribution Strategy

Not applicable.

3.4 Related Routines

This is the first in a suite of three routines. The other two routines are:

F11GBFP:	to carry out the iterations
F11GCFP:	to return additional information about the computation

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: determines whether preconditioning is used. The possible choices are:

'N'	no preconditioning;
'P'	preconditioning.

Constraint: PRECON = 'N' or 'P'.

4: SIGCMP — CHARACTER*1 *Global Input*

On entry: determines whether an estimate of $\sigma_1(\bar{A}) = \|E^{-1}AE^{-T}\|_2$, the largest singular value of the preconditioned matrix, is to be computed using the bisection method on the sequence of symmetric tridiagonal matrices $\{T_k\}$ generated during the iteration (see Section 6 for further details). Note that $\bar{A} = A$ when a preconditioner is not used. If SIGMAX > 0.0 (see below), then SIGCMP is not referenced, i.e., when $\sigma_1(\bar{A})$ is supplied as input. The possible choices are:

'S'	$\sigma_1(\bar{A})$ is to be computed using the bisection method.
'N'	The bisection method is not used. If the termination criterion requires $\sigma_1(\bar{A})$, then a less expensive estimate is computed.

See Section 6 for further details.

Suggested value: SIGCMP = 'N'.

Constraint: SIGCMP = 'S' or 'N'.

5: NORM — CHARACTER*1 *Global Input*

On entry: defines the matrix and vector norm to be used in the termination criteria. The possible choices are:

'1'	l_1 norm;
'I'	l_∞ norm;
'2'	l_2 norm.

Suggested value:

NORM = 'I', if ITERM = 1;
NORM = '2', if ITERM = 2.

Constraints:

if ITERM = 1, then NORM = '1', 'I' or '2';
if ITERM = 2, then NORM = '2'.

6: DISTR — CHARACTER*1 *Global Input*

On entry: defines how vectors are distributed across the processors in the Library Grid (see Section 2.5.3 of the F11 Chapter Introduction). The possible choices are:

'A'	vectors are distributed across all processors in the Library Grid;
'C'	vectors are distributed by column;
'R'	vectors are distributed by row.

Suggested value: DISTR = 'A'.

Constraint: DISTR = 'A', 'C' or 'R'.

7: WEIGHT — CHARACTER*1 *Global Input*

On entry: specifies whether a vector w of user-supplied weights is to be used in the computation of the vector norms required in termination criterion (2) of Section 6.1. (ITERM = 1): $\|v\|_p^{(w)} = \|v^{(w)}\|_p$, where $v_i^{(w)} = w_i\, v_i$, for $i = 1, 2, \ldots, n$. The suffix $p = 1, 2, \infty$ denotes the vector norm used, as specified by the parameter NORM. Note that weights cannot be used when ITERM = 2, i.e., when criterion (3) of Section 6.1 is used. The possible choices are:

'W'	user-supplied weights are to be used and must be supplied on initial entry to F11GBFP.
'N'	all weights are implicitly set equal to one. Weights do not need to be supplied on initial entry to F11GBFP.

Suggested value: WEIGHT = 'N'.

Constraints:

if ITERM = 1, then WEIGHT = 'W' or 'N';
if ITERM = 2, then WEIGHT = 'N'.

8: ITERM — INTEGER *Global Input*

On entry: defines the termination criterion to be used:

if ITERM = 1, the termination criterion defined in (2) of Section 6.1 is used (both CG and SYMMLQ);
if ITERM = 2, the termination criterion defined in (3) of Section 6.1 is used (SYMMLQ method only).

Suggested value: ITERM = 1.

Constraints: ITERM = 1 or 2.

9: N — INTEGER *Global Input*

On entry: the order n of the matrix A.

Constraint: N > 0.

10: NLOC — INTEGER *Local Input*

On entry: the number of vector elements stored locally, i.e., NLOC = $n_l(i,j)$, where i, j are the row and column indices, respectively, of the calling processor. Note that information about the distribution pattern is not required: only the number of vector elements stored locally must be supplied.

Constraint: NLOC ≥ 0 and, according to the value of DISTR:

DISTR = 'A': $\sum_{i=0}^{m_p-1}\sum_{j=0}^{n_p-1} n_l(i,j) = n;$

DISTR = 'C': $\sum_{i=0}^{m_p-1} n_l(i,j) = n,$

for $j = 0,\ldots,n_p-1,$
$n_l(i,0) = n_l(i,1) = \ldots = n_l(i,n_p-1)$ and
$i = 0,\ldots,m_p-1;$

DISTR = 'R': $\sum_{j=0}^{n_p-1} n_l(i,j) = n,$

for $i = 0,\ldots,m_p-1,$
$n_l(0,j) = n_l(1,j) = \ldots = n_l(m_p-1,j)$ and
$j = 0,\ldots,n_p-1.$

11: TOL — DOUBLE PRECISION *Global Input*

On entry: the tolerance τ for the termination criterion. If TOL ≤ 0.0, $\tau = \max(\sqrt{\epsilon},\sqrt{n}\,\epsilon)$ is used, where ϵ is the ***machine precision***. Otherwise $\tau = \max(\text{TOL},10\epsilon,\sqrt{n}\,\epsilon)$ is used.

Constraint: TOL < 1.0.

12: MAXITN — INTEGER *Global Input*

On entry: the maximum number of iterations.

Constraint: MAXITN > 0.

13: ANORM — DOUBLE PRECISION *Global Input*

On entry: if ANORM > 0.0, the value of $\|A\|_p$ to be used in the termination criterion (2) of Section 6.1 (ITERM = 1).

If ANORM ≤ 0.0, ITERM = 1 and NORM = '1' or 'I', then $\|A\|_1$ or $\|A\|_\infty$ is estimated internally by F11GBFP.

If ITERM = 2, then ANORM is not referenced.

Constraint: if ITERM = 1 and NORM = '2', then ANORM > 0.0.

14: SIGMAX — DOUBLE PRECISION *Global Input*

On entry: if SIGMAX > 0.0, the value of the largest singular value of the preconditioned matrix: $\sigma_1(\bar{A}) = \|E^{-1}AE^{-T}\|_2$.

If SIGMAX ≤ 0.0, $\sigma_1(\bar{A})$ is estimated by F11GBFP when either SIGCMP = 'S' or the termination criterion defined in (3) of Section 6.1 is used. Otherwise, SIGMAX is not referenced.

15: SIGTOL — DOUBLE PRECISION *Global Input*

On entry: the tolerance used in assessing the convergence of the estimate of $\sigma_1(\bar{A}) = \|\bar{A}\|_2$ when the bisection method is used. If SIGTOL ≤ 0.0, the default value SIGTOL $= 0.01$ is used. The actual value used is max (SIGTOL,ϵ). If SIGCMP = 'N' or SIGMAX > 0.0, then SIGTOL is not referenced.

Suggested value: SIGTOL $= 0.01$ should be sufficient in most cases.

Constraint: if SIGCMP = 'S' and SIGMAX ≤ 0.0, then SIGTOL < 1.0.

16: MAXITS — INTEGER *Global Input*

On entry: the maximum iteration number $k =$ MAXITS for which $\sigma_1(T_k)$ is computed by bisection (see also Section 6.1). If SIGCMP = 'N' or SIGMAX > 0.0, then MAXITS is not referenced.

Suggested value: MAXITS = min(10,n) when SIGTOL is of the order of its default value (0.01).

Constraint: if SIGCMP = 'S' and SIGMAX ≤ 0.0, then $1 \leq$ MAXITS $\leq$ MAXITN.

17: MONIT — INTEGER *Global Input*

On entry: if MONIT > 0, the frequency at which a monitoring step is executed by F11GBFP: the current solution and residual iterates will be returned by F11GBFP and a call to F11GCFP made possible every MONIT iterations, starting from iteration MONIT. Otherwise, no monitoring takes place.

There are some additional computational costs involved in monitoring the solution and residual vectors when the Lanczos method (SYMMLQ) is used.

Constraint: MONIT $\leq$ MAXITN.

18: LWREQ — INTEGER *Local Output*

On exit: the amount of workspace required by F11GBFP. See also Section 4 of the document for F11GBFP.

19: 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 had illegal value(s) on one or more processors. For global arguments, this may also be caused by the ith argument not having the same value on **all** processors (see also Section 3.2).

IFAIL $= 1$

F11GAFP has been called out of sequence. Either F11GAFP has been called twice without calling F11GBFP in between, or F11GBFP has not completed its current task.

6 Further Comments

6.1 Algorithmic Detail

The conjugate gradient method (CG) (see Hestenes and Stiefel [4], Golub and van Loan [3], Barrett *et al.*[1], Dias da Cunha and Hopkins [2]) should be ideally applied when the matrix A is positive-definite. Otherwise the Lanczos Method (SYMMLQ), based upon the algorithm SYMMLQ (Paige and Saunders [6], Barrett *et al.*[1]), should be used. It is more robust than the conjugate gradient method (CG) but less efficient when A is positive-definite.

Both the conjugate gradient method (CG) and Lanczos (SYMMLQ) method start from the residual $r_0 = b - Ax_0$, where x_0 is an initial estimate for the solution (often $x_0 = 0$), and generate an orthogonal basis for the Krylov subspace span$\{A^k r_0\}$, for $k = 0, 1, \ldots$, by means of three-term recurrence relations (Golub and van Loan [3]). A sequence of symmetric tridiagonal matrices $\{T_k\}$ is also generated. Here and in the following, the index k denotes the iteration count. The resulting symmetric tridiagonal systems of equations are usually more easily solved than the original problem. A sequence of solution iterates $\{x_k\}$ is thus generated such that the sequence of the norms of the residuals $\{\|r_k\|\}$ converges to a required tolerance. Note that, in general, the convergence is not monotonic.

In exact arithmetic, after n iterations, this process is equivalent to an orthogonal reduction of A to symmetric tridiagonal form, $T_n = Q^T AQ$; the solution x_n would thus achieve exact convergence. In finite-precision arithmetic, cancellation and round-off errors accumulate causing loss of orthogonality. These methods must therefore be viewed as genuinely iterative methods, able to converge to a solution **within a prescribed tolerance**.

The orthogonal basis is not formed explicitly in either method. The basic difference between the two methods lies in the method of solution of the resulting symmetric tridiagonal systems of equations: the conjugate gradient method (CG) is equivalent to carrying out an LDL^T (Cholesky) factorization whereas the Lanczos method (SYMMLQ) uses an LQ factorization.

Faster convergence can be achieved using a **preconditioner** (Golub and van Loan [3], Barrett *et al.*[1]). A preconditioner maps the original system of equations onto a different system, say

$$\bar{A}\bar{x} = \bar{b}, \tag{1}$$

with, hopefully, better characteristics with respect to its speed of convergence: for example, the condition number of the matrix of the coefficients can be improved or eigenvalues in its spectrum can be made to coalesce. An orthogonal basis for the Krylov subspace span$\{\bar{A}^k \bar{r}_0\}$, for $k = 0, 1, \ldots$, is generated and the solution proceeds as outlined above. The algorithms used are such that the solution and residual iterates of the original system are produced, not their preconditioned counterparts. Note that an unsuitable preconditioner or no preconditioning at all may result in a very slow rate, or lack, of convergence. However, preconditioning involves a trade-off between the reduction in the number of iterations required for convergence and the additional computational costs per iteration. Also, setting up a preconditioner may involve non-negligible overheads.

A preconditioner must be **symmetric and positive-definite**, i.e. representable by $M = EE^T$, where M is non-singular, and such that $\bar{A} = E^{-1}AE^{-T} \sim I_n$ in (1), where I_n is the identity matrix of order n. Also, we can define $\bar{r} = E^{-1}r$ and $\bar{x} = E^T x$. These are formal definitions, used only in the design of the algorithms; in practice, only the means to compute the matrix-vector products $v = Au$ and to solve the preconditioning equations $Mv = u$ are required, that is, explicit information about M, E or their inverses in not required at any stage.

The first termination criterion

$$\|r_k\|_p \leq \tau \left(\|b\|_p + \|A\|_p \, \|x_k\|_p\right) \tag{2}$$

is available for both the conjugate gradient method (CG) and the Lanczos method (SYMMLQ). In (2), $p = 1, \infty$ or 2 and τ denotes a user-specified tolerance subject to $\max(10, \sqrt{n})\ \epsilon \leq \tau < 1$, where ϵ is the ***machine precision***. Facilities are provided for the estimation of the norm of the matrix of the coefficients $\|A\|_1 = \|A\|_\infty$, when this is not known in advance, used in (2), by applying Higham's method (Higham [5]). Note that $\|A\|_2$ cannot be estimated internally. This criterion uses an error bound derived from **backward** error analysis to ensure that the computed solution is the exact solution of a problem as close to the original as the termination tolerance requires. Termination criteria employing bounds derived from **forward** error analysis could be used, but any such criteria would require information about the condition number $\kappa(A)$ which is not easily obtainable.

The second termination criterion

$$\|\bar{r}_k\|_2 \leq \tau \max(1.0,\ \|b\|_2/\|r_0\|_2)\left(\|\bar{r}_0\|_2 + \sigma_1(\bar{A})\ \|\Delta\bar{x}_k\|_2\right) \tag{3}$$

is available only for the Lanczos method (SYMMLQ). In (3), $\sigma_1(\bar{A}) = \|\bar{A}\|_2$ is the largest singular value of the (preconditioned) iteration matrix $\bar{A}$. This termination criterion monitors the progress of the solution of the preconditioned system of equations and is less expensive to apply than criterion (2). When $\sigma_1(\bar{A})$ is not supplied, facilities are provided for its estimation by $\sigma_1(\bar{A}) \sim \max_k \sigma_1(T_k)$. The interlacing property $\sigma_1(T_{k-1}) \leq \sigma_1(T_k)$ and Gerschgorin's theorem provide lower and upper bounds from which $\sigma_1(T_k)$ can be easily computed by bisection. Alternatively, the less expensive estimate $\sigma_1(\bar{A}) \sim \max_k \|T_k\|_1$ can be used, where $\sigma_1(\bar{A}) \leq \|T_k\|_1$ by Gerschgorin's theorem. Note that only order of magnitude estimates are required by the termination criterion.

Termination criterion (2) is the recommended choice, despite its (small) additional costs per iteration when using the Lanczos method (SYMMLQ). Also, if the norm of the initial estimate is much larger than the norm of the solution, that is, if $\|x_0\| \gg \|x\|$, a dramatic loss of significant digits could result in complete lack of convergence. The use of criterion (2) will enable the detection of such a situation, and the iteration will be restarted at a suitable point. No such restart facilities are provided for criterion (3).

When $\sigma_1(\bar{A})$ is not supplied (SIGMAX $\leq$ 0.0) but it is required, it is estimated by F11GBFP using either of the two methods described above, as specified by the parameter SIGCMP. In particular, if SIGCMP = 'S', then the computation of $\sigma_1(\bar{A})$ is deemed to have converged when the differences between three successive values of $\sigma_1(T_k)$ differ, in a relative sense, by less than the tolerance SIGTOL, i.e. when

$$\max\left(\frac{|\sigma_1^{(k)} - \sigma_1^{(k-1)}|}{\sigma_1^{(k)}}, \frac{|\sigma_1^{(k)} - \sigma_1^{(k-2)}|}{\sigma_1^{(k)}}\right) \leq \text{SIGTOL}.$$

The computation of $\sigma_1(\bar{A})$ is also terminated when the iteration count exceeds the maximum value allowed, i.e. $k \geq$ MAXITS.

Bisection is increasingly expensive with increasing iteration count. A reasonably large value of SIGTOL, of the order of the suggested value, is recommended and an excessive value of MAXITS should be avoided. Under these conditions, $\sigma_1(\bar{A})$ usually converges within very few iterations.

6.2 Parallelism Detail

Not applicable.

6.3 Accuracy

Not applicable.

6.4 Computational Costs

The computational costs of F11GAFP are negligible compared to the costs of F11GBFP.

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] Dias da Cunha R and Hopkins T (1994) PIM 1.1 — the parallel iterative method package for systems of linear equations user's guide — Fortran 77 version *Technical Report* Computing Laboratory, University of Kent at Canterbury, Kent CT2 7NZ, UK

[3] Golub G H and van Loan C F (1996) *Matrix Computations* Johns Hopkins University Press (3rd Edition), Baltimore

[4] Hestenes M and Stiefel E (1952) Methods of conjugate gradients for solving linear systems *J. Res. Nat. Bur. Stand.* **49** 409–436

[5] Higham N J (1988) FORTRAN codes for estimating the one-norm of a real or complex matrix, with applications to condition estimation *ACM Trans. Math. Software* **14** 381–396

[6] Paige C C and Saunders M A (1975) Solution of sparse indefinite systems of linear equations *SIAM J. Numer. Anal.* **12** 617–629

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

```
*     F11GAFP Example Program Text
*     NAG Parallel Library Release 3 Revised. NAG Copyright 1999.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          MLMAX, NBLKS
      PARAMETER        (MLMAX=1000,NBLKS=4)
      INTEGER          LA, LC
      PARAMETER        (LA=5*MLMAX,LC=2*LA)
      INTEGER          LIA, LWORK
      PARAMETER        (LIA=-1,LWORK=20*MLMAX)
*     .. Scalars in Common ..
      DOUBLE PRECISION C1, C2, C3
      INTEGER          NX
*     .. Local Scalars ..
      DOUBLE PRECISION ANORM, ENORM, ENORML, SIGERR, SIGMAX, SIGTOL,
     +                 STPLHS, STPRHS, TOL
      INTEGER          I, ICNTXT, IFAIL, IREVCM, ITERM, ITN, ITS, IW, J,
     +                 LEVEL, LW, LWREQ, MAXITN, MAXITS, MB, ML, MLO,
     +                 MLOMAX, MONIT, MP, N, NB, NINTE, NINTI, NNZ,
     +                 NNZC, NOVRLP, NP
      LOGICAL          LOOP, ROOT, ZGRID
      CHARACTER        CHECK, DISTR, DUP, KIND, NORM, PRECON, SIGCMP,
     +                 SYMM, WEIGHT, ZERO
```

```
      CHARACTER*10     METHOD
      CHARACTER*80     FORMAT
*     .. Local Arrays ..
      DOUBLE PRECISION A(LA), C(LC), DTOL(NBLKS), TS(MLMAX), U(MLMAX),
     +                 V(MLMAX), WORK(LWORK)
      INTEGER          CA(1), IAINFO(200), ICOL(LA), ICOLC(LC), IERR(1),
     +                 IPIVP(MLMAX), IPIVQ(MLMAX), IROW(LA), IROWC(LC),
     +                 LFILL(NBLKS), NPIVM(NBLKS), RA(1)
      CHARACTER        MILU(NBLKS), PSTRAT(NBLKS)
*     .. External Functions ..
      LOGICAL          Z01ACFP
      EXTERNAL         Z01ACFP
*     .. External Subroutines ..
      EXTERNAL         DGERV2D, DGESD2D, F01CPFP, F01YAFP, F01YEFP,
     +                 F11DFFP, F11DGFP, F11GAFP, F11GBFP, F11GCFP,
     +                 F11XBFP, F11ZBFP, F11ZZFP, GMAT, GSOL, GVEC,
     +                 PRINTI, X04YAFP, Z01AAFP, Z01ABFP, Z01BBFP,
     +                 Z02EAFP
*     .. Intrinsic Functions ..
      INTRINSIC        ABS, MAX
*     .. Common blocks ..
      COMMON           /PROB/C1, C2, C3, NX
*     .. Executable Statements ..
      ROOT = Z01ACFP()
      IF (ROOT) WRITE (NOUT,*) 'F11GAFP Example Program Results'
*
*     Open input file on all processors
*
      OPEN (NIN,FILE='f11gafpe.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, SIGCMP, NORM, ITERM, MONIT
      READ (NIN,*) TOL, MAXITN
      READ (NIN,*) SIGTOL, MAXITS
      READ (NIN,*) FORMAT
      READ (NIN,*) LEVEL
*
*     Read coefficients in PDE
*
      READ (NIN,*) C1, C2, C3
*
*     Close input file
*
      CLOSE (NIN)
*
```

```
*     Initialize Library Grid
*
      IFAIL = 0
      CALL Z01AAFP(ICNTXT,MP,NP,IFAIL)
*
*     Check whether processor is part of the Library Grid
*
      CALL Z01BBFP(ICNTXT,ZGRID,IFAIL)
      IF ( .NOT. ZGRID) GO TO 160
*
*     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)
      NB = ML/MB
*
*     Check number of blocks is less than max allowed
*
      IERR(1) = 0
      IF (NB.GT.NBLKS) IERR(1) = 1
      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 140
      END IF
*
*     Generate right-hand side vector
*
      CALL F01YEFP(ICNTXT,GVEC,N,V,IAINFO,IFAIL)
*
*     Set up block Jacobi preconditioner
*     Initialise parameters for each block on processor
*
      DO 20 J = 1, NB
         LFILL(J) = 0
         DTOL(J) = 1.D-1
         PSTRAT(J) = 'N'
         MILU(J) = 'N'
   20 CONTINUE
```

```
      NOVRLP = 0
      CALL F11DFFP(ICNTXT,N,NNZ,A,IROW,ICOL,NOVRLP,LFILL,DTOL,PSTRAT,
     +             MILU,IPIVP,IPIVQ,NNZC,C,LC,IROWC,ICOLC,NPIVM,IAINFO,
     +             LIA,IFAIL)
*
*     Initialize solver suite
*
      WEIGHT = 'N'
      DISTR = 'A'
      ANORM = 0.D0
      SIGMAX = 0.D0
      CHECK = 'N'
      CALL F11GAFP(ICNTXT,METHOD,PRECON,SIGCMP,NORM,DISTR,WEIGHT,ITERM,
     +             N,ML,TOL,MAXITN,ANORM,SIGMAX,SIGTOL,MAXITS,MONIT,
     +             LWREQ,IFAIL)
*
*     Check workspace size
*
      NINTI = IAINFO(6)
      NINTE = IAINFO(7)
      MLO = IAINFO(4)
      MLOMAX = IAINFO(5)
      IERR(1) = 0
      IF (ROOT) THEN
         IF ((LWREQ+MAX(NINTI,NINTE,2*MLO,ML+MLOMAX)).GT.LWORK) IERR(1)
     +       = 1
      ELSE
         IF ((LWREQ+MAX(NINTI,NINTE,2*MLO)).GT.LWORK) IERR(1) = 1
      END IF
*
      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 140
      END IF
*
*     Print summary of input parameters and options
*
      IF (ROOT) CALL PRINTI(NOUT,METHOD,PRECON,SIGCMP,NORM,DISTR,ITERM,
     +                      N,MAXITN,TOL,SIGTOL,MONIT,MP,NP,MB)
*
*     Set initial approximation to solution
*
      DO 40 I = 1, ML
         U(I) = 0.D0
   40 CONTINUE
*
*     Solve equations using reverse communication scheme
*
      LW = LWREQ
      IW = LWREQ + 1
      IREVCM = 0
      LOOP = .TRUE.
*
   60 CONTINUE
*
      CALL F11GBFP(ICNTXT,IREVCM,U,V,WORK,LW,IFAIL)
*
```

```
      IF (IREVCM.EQ.1) THEN
*
*     Compute  v = A * u
*
         CALL F11XBFP(ICNTXT,'No transpose',N,NNZ,A,IROW,ICOL,CHECK,U,V,
     +                IAINFO,WORK(IW),IFAIL)
*
      ELSE IF (IREVCM.EQ.2) THEN
*
*     Solve  M * v = u
*
         CALL F11DGFP(ICNTXT,'No transpose',N,NNZC,C,IROWC,ICOLC,IPIVP,
     +                IPIVQ,CHECK,U,V,IAINFO,WORK(IW),IFAIL)
*
*
      ELSE IF (IREVCM.EQ.3) THEN
*
*     Monitoring
*
         CALL F11GCFP(ICNTXT,ITN,STPLHS,STPRHS,ANORM,SIGMAX,ITS,SIGERR,
     +                IFAIL)
         IF (ROOT) THEN
            WRITE (NOUT,'(/1X,''Monitoring step''/1X,15(''-'')/)')
            WRITE (NOUT,99999)
     +        'Number of iterations carried out                     -',
     +        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 .OR. SIGCMP.EQ.'S') WRITE (NOUT,99998)
     +          'Largest singular value of (precond.) matrix (SIGMAX) -'
     +          , SIGMAX
            IF (SIGCMP.EQ.'S') THEN
               WRITE (NOUT,99999)
     +          'Number of iterations used to compute SIGMAX (ITS)    -'
     +           , ITS
               WRITE (NOUT,99998)
     +          'Relative error in largest singular value (SIGERR)    -'
     +           , SIGERR
            END IF
            WRITE (NOUT,
     +        '(/1X,''Solution vector (last iterate)''/1X,30(''-'')/)')
         END IF
         CALL X04YAFP(ICNTXT,NOUT,N,U,FORMAT,IAINFO,WORK(IW),IFAIL)
         IF (ROOT) WRITE (NOUT,
     +       '(/1X,''Residual vector (last iterate)''/1X,30(''-'')/)')
         CALL X04YAFP(ICNTXT,NOUT,N,V,FORMAT,IAINFO,WORK(IW),IFAIL)
*
      ELSE IF (IREVCM.EQ.4) THEN
*
*     Termination
*
         LOOP = .FALSE.
      END IF
      IF (LOOP) GO TO 60
```

```
*
*     Generate true solution TS and error on local part of mesh
*
      CALL F01YEFP(ICNTXT,GSOL,N,TS,IAINFO,IFAIL)
      ENORML = 0.D0
      DO 80 I = 1, IAINFO(3)
         ENORML = MAX(ENORML,ABS(TS(I)-U(I)))
   80 CONTINUE
      IF ( .NOT. ROOT) CALL DGESD2D(ICNTXT,1,1,ENORML,1,0,0)
*
*     Get information about final solution
*
      CALL F11GCFP(ICNTXT,ITN,STPLHS,STPRHS,ANORM,SIGMAX,ITS,SIGERR,
     +             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) WRITE (NOUT,99998)
     +       'Norm of the matrix of the coefficients (ANORM)       -',
     +       ANORM
         IF (ITERM.EQ.2 .OR. SIGCMP.EQ.'S') WRITE (NOUT,99998)
     +       'Largest singular value of (precond.) matrix (SIGMAX) -',
     +       SIGMAX
         IF (SIGCMP.EQ.'S') THEN
            WRITE (NOUT,99999)
     +        'Number of iterations used to compute SIGMAX (ITS)    -',
     +        ITS
            WRITE (NOUT,99998)
     +        'Relative error in largest singular value (SIGERR)    -',
     +        SIGERR
         END IF
*
*     Receive local error norms and calculate global error norm
*
         ENORM = ENORML
         DO 120 I = 1, MP
            DO 100 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
  100       CONTINUE
  120    CONTINUE
         WRITE (NOUT,*)
         WRITE (NOUT,99998) 'Error norm =', ENORM
         WRITE (NOUT,'(/1X,''Solution vector''/1X,15(''-'')/)')
      END IF
      CALL X04YAFP(ICNTXT,NOUT,N,U,FORMAT,IAINFO,WORK(IW),IFAIL)
```

```
*
*     Release internally allocated memory if necessary
*
  140 IF (LIA.EQ.-1) CALL F11ZZFP(ICNTXT,IAINFO,IFAIL)
*
*     Finalize Library Grid
*
  160 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 = 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,SIGCMP,NORM,DISTR,ITERM,N,
     +                  MAXITN,TOL,SIGTOL,MONIT,MP,NP,MB)
*
*     Prints a summary of the input parameters and options.
*
*
*     .. Scalar Arguments ..
      DOUBLE PRECISION  SIGTOL, TOL
      INTEGER           ITERM, MAXITN, MB, MONIT, MP, N, NOUT, NP
      CHARACTER         DISTR, NORM, PRECON, SIGCMP
      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)
     +  'Use bisection for largest singular value (SIGCMP)    -', SIGCMP
      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
      WRITE (NOUT,99996)
     +  'Tolerance for the largest singular value (SIGTOL)    -', SIGTOL
      WRITE (NOUT,99997)
     +  'Monitoring frequency (MONIT)                         -', MONIT
*
*     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

```
F11GAFP Example Program Data
    2    2                         :  MP, NP
    8                              :  NX
  'CG'                             :  METHOD
  'P'  'S' 'I'    1    0           :  PRECON, SIGCMP, NORM, ITERM, MONIT
   1.0D-09   100                   :  TOL, MAXITN
   1.0D-02    6                    :  SIGTOL, MAXITS
   '(8F8.4)'                       :  FORMAT
    0                              :  LEVEL
  -1.0 -3.0  0.1                   :  C1, C2, C3
```

8.3 Example Results

```
F11GAFP 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)                      -    P
Use bisection for largest singular value (SIGCMP)    -    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
Tolerance for the largest singular value (SIGTOL)    -    1.00D-02
Monitoring frequency (MONIT)                         -    0

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

Number of iterations carried out (ITN)               -   22
Left-hand side of termination criterion (STPLHS)     -    5.81D-07
Right-hand side of termination criterion (STPRHS)    -    2.63D-06
Norm of the matrix of the coefficients (ANORM)       -    1.2961E+03
Largest singular value of (precond.) matrix (SIGMAX) -    1.6844E+00
Number of iterations used to compute SIGMAX (ITS)    -    6
Relative error in largest singular value (SIGERR)    -    1.3439E-02

Error norm =    1.0098E-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
```