F11BAFP

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 F11BAFP, F11BBFP and F11BCFP is designed to solve the real general (nonsymmetric) system of simultaneous linear equations $Ax = b$ of order n, where n is large and the coefficient matrix A is sparse.

F11BAFP is a set-up routine which must be called before F11BBFP, the iterative solver. The third routine in the suite, F11BCFP, can be used to return additional information about the computation. A choice of methods is available:

restarted generalized minimum residual method (RGMRES),

conjugate gradient squared method (CGS), or

stabilized bi-conjugate gradient method of order ℓ (Bi-CGSTAB(ℓ)).

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

2 Specification

```
 SUBROUTINE F11BAFP(ICNTXT, METHOD, PRECON, NORM, DISTR, WEIGHT,
1                   ITERM, N, NLOC, M, TOL, MAXITN, ANORM, SIGMAX,
2                   MONIT, LWREQ, IFAIL)
 DOUBLE PRECISION   TOL, ANORM, SIGMAX
 INTEGER            ICNTXT, ITERM, N, NLOC, M, MAXITN, MONIT,
1                   LWREQ, IFAIL
 CHARACTER*1        PRECON, 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, NORM, DISTR, WEIGHT, ITERM, N, M, TOL, MAXITN, ANORM, SIGMAX, 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:

F11BBFP: to carry out the iterations;

F11BCFP: 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:

'RGMRES' restarted generalized minimum residual method;
'CGS' conjugate gradient squared method;
'BICGSTAB' stabilized bi-conjugate gradient method or order ℓ;

Constraint: METHOD = 'RGMRES', 'CGS' or 'BICGSTAB'.

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: 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'.

5: 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'.

6: 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 6.1.4. (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 6.1.4 is used. The possible choices are:

'W' user-supplied weights are to be used and must be supplied on initial entry to F11BBFP.
'N' all weights are implicitly set equal to one. Weights do not need to be supplied on initial entry to F11BBFP.

Suggested value: WEIGHT = 'N'.

Constraints:

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

7: ITERM — INTEGER *Global Input*

On entry: defines the termination criterion to be used:

if ITERM = 1, the termination criterion defined in (2) of Section 6.1.4 is used;
if ITERM = 2, the termination criterion defined in (3) of Section 6.1.4 is used.

Suggested value: ITERM = 1.

Constraint: ITERM = 1 or 2.

8: N — INTEGER *Global Input*

On entry: the order n of the matrix A.

Constraint: N > 0.

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

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
$j = 0, \ldots, n_p - 1.$

10: M — INTEGER *Global Input*

On entry: if METHOD = 'RGMRES', M is the dimension m of the restart subspace.

If METHOD = 'BICGSTAB', M is the order ℓ of the polynomial Bi-CGSTAB method. Otherwise, M is not referenced.

Constraints:

if METHOD = 'RGMRES', $0 < \text{M} \leq \min(\text{N},50)$;
if METHOD = 'BICGSTAB', $0 < \text{M} \leq \min(\text{N},10)$.

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 6.1.4 (ITERM = 1).

If ANORM ≤ 0.0, ITERM = 1 and NORM = '1' or 'I', then $\|A\|_1$ or $\|A\|_\infty$ is estimated internally by F11BBFP.

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 ITERM = 2, the largest singular value σ_1 of the preconditioned iteration matrix; otherwise, SIGMAX is not referenced.

If SIGMAX ≤ 0.0, ITERM = 2 and METHOD = 'RGMRES', then the value of σ_1 will be estimated internally.

Constraint: if METHOD = 'CGS' or 'BICGSTAB' and ITERM = 2, then SIGMAX > 0.0.

15: MONIT — INTEGER *Global Input*

On entry: if MONIT > 0, the frequency at which a monitoring step is executed by F11BBFP: if METHOD = 'CGS' a monitoring step is executed every MONIT iterations; otherwise, a monitoring step is executed every MONIT super-iterations (groups of up to m or ℓ iterations for RGMRES or Bi-CGSTAB(ℓ), respectively).

There are some additional computational costs involved in monitoring the solution and residual vectors when the Bi-CGSTAB(ℓ) method is used with $\ell > 1$.

Constraint: MONIT $\leq$ MAXITN.

16: LWREQ — INTEGER *Local Output*

On exit: the minimum amount of workspace required by F11BBFP. (See also Section 4 of the document for F11BBFP.)

17: 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

F11BAFP has been called out of sequence. Either F11BAFP has been called twice without calling F11BBFP in between, or F11BBFP has not completed its current task.

6 Further Comments

6.1 Algorithmic Detail

6.1.1 Restarted Generalized Minimum Residual Method (RGMRES)

The restarted generalized minimum residual method (RGMRES) (Saad and Schultz [4], Barrett *et al.* [2], Dias da Cunha and Hopkins [3]) starts from the residual $r_0 = b - Ax_0$, where x_0 is an initial estimate for the solution (often $x_0 = 0$). An orthogonal basis for the Krylov subspace span$\{A^k r_0\}$, for $k = 0, 1, 2, \ldots$, is generated explicitly: this is referred to as Arnoldi's method (Arnoldi [8]). The solution is then expanded onto the orthogonal basis so as to minimize the residual norm $\|b - Ax\|_2$. The lack of symmetry of A implies that the orthogonal basis is generated by applying a 'long' recurrence relation, whose length increases linearly with the iteration count. For all but the most trivial problems, computational and storage costs can quickly become prohibitive as the iteration count increases. RGMRES limits these costs by employing a restart strategy: every m iterations at most, the Arnoldi process is restarted from $r_l = b - Ax_l$, where the subscript l denotes the last available iterate. Each group of m iterations is referred to as a 'super-iteration'. The value of m is chosen in advance and is fixed throughout the computation. Unfortunately, an optimum value of m cannot easily be predicted.

6.1.2 Conjugate Gradient Squared Method (CGS)

The conjugate gradient squared method (CGS) (Sonneveld [5], Barrett *et al.* [2], Dias da Cunha and Hopkins [3]) is a development of the bi-conjugate gradient method where the nonsymmetric Lanczos method is applied to reduce the coefficients matrix to real tridiagonal form: two bi-orthogonal sequences of vectors are generated starting from the residual $r_0 = b - Ax_0$, where x_0 is an initial estimate for the solution (often $x_0 = 0$) and from the *shadow residual* $\hat{r}_0$ corresponding to the arbitrary problem $A^T\hat{x} = \hat{b}$, where $\hat{b}$ can be any vector, but in practice is chosen so that $r_0 = \hat{r}_0$. In the course of the iteration, the residual and shadow residual $r_i = P_i(A)r_0$ and $\hat{r}_i = P_i(A^T)\hat{r}_0$ are generated, where P_i is a polynomial of order i, and bi-orthogonality is exploited by computing the vector product $\rho_i = (\hat{r}_i, r_i) = (P_i(A^T)\hat{r}_0, P_i(A)r_0) = (\hat{r}_0, P_i^2(A)r_0)$. Applying the 'contraction' operator $P_i(A)$ twice, the iteration coefficients can still be recovered without advancing the solution of the shadow problem, which is of no interest. The CGS method often provides fast convergence; however, there is no reason why the contraction operator should also reduce the once reduced vector $P_i(A)r_0$: this may well lead to a highly irregular convergence which may result in large cancellation errors.

6.1.3 Stabilized Bi-Conjugate Gradient Method of Order ℓ (Bi-CGSTAB(ℓ))

The stabilized bi-conjugate gradient method of order ℓ (Bi-CGSTAB(ℓ)) (van der Vorst [6], Sleijpen and Fokkema [7], Dias da Cunha and Hopkins [3]) is similar to the CGS method above. However, instead of generating the sequence $\{P_i^2(A)r_0\}$, it generates the sequence $\{Q_i(A)P_i(A)r_0\}$ where the $Q_i(A)$ are polynomials chosen to minimize the residual *after* the application of the contraction operator $P_i(A)$. Two main steps can be identified for each iteration: an OR (Orthogonal Residuals) step where a basis of order ℓ is generated by a Bi-CG iteration and an MR (Minimum Residuals) step where the residual is minimized over the basis generated, by a method akin to GMRES. For $\ell = 1$, the method corresponds to the Bi-CGSTAB method of van der Vorst [6]. For $\ell > 1$ more information about complex eigenvalues of the iteration matrix can be taken into account, and this may lead to improved convergence and robustness. However, as ℓ increases, numerical instabilities may arise. For this reason, a maximum value of $\ell = 10$ is imposed, but probably $\ell = 4$ is sufficient in many cases.

6.1.4 General considerations

For each method, a sequence of solution iterates $\{x_i\}$ is generated such that, hopefully, the sequence of the residual norms $\{\|r_i\|\}$ converges to a required tolerance. Note that, in general, convergence, when it occurs, is not monotonic.

In the RGMRES and Bi-CGSTAB(ℓ) methods above, the user's program must provide the **maximum** number of basis vectors used, m or ℓ, respectively; however, a **smaller** number of basis vectors may be generated and used when the stability of the solution process requires this.

Faster convergence can be achieved using a **preconditioner** (Golub and van Loan [1], Barrett *et al.* [2]). 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 coefficients matrix 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 *left* preconditioner M^{-1} can be used by the RGMRES and CGS methods, such that $\bar{A} = M^{-1}A \sim I_n$ in (1), where I_n is the identity matrix of order n; a *right* preconditioner M^{-1} can be used by the Bi-CGSTAB(ℓ) method, such that $\bar{A} = AM^{-1} \sim I_n$. 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 $v = A^T u$ (the latter only being required when an estimate of $\|A\|_1$ or $\|A\|_\infty$ is computed internally), and to solve the preconditioning equations $Mv = u$ are required, that is, explicit information about M, or its inverse is 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 all three methods. 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 coefficients matrix $\|A\|_1$ or $\|A\|_\infty$, when this is not known in advance, by applying Higham's method (Higham [9]). 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 are not used because any such criteria would require information about the condition number $\kappa(A)$ which is not easily obtainable.

The second termination criterion

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

is also available for all three methods. 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) for the Bi-CGSTAB(ℓ) method with $\ell > 1$. Only the RGMRES method provides facilities to estimate $\sigma_1(\bar{A})$ internally, when this is not supplied.

Termination criterion (2) is the recommended choice, despite its additional costs per iteration when using the Bi-CGSTAB(ℓ) method with $\ell > 1$. 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).

Optionally, a vector w of user-specified weights can be used in the computation of the vector norms in termination criterion (2), i.e., $\|v\|_p^{(w)} = \|v^{(w)}\|_p$, where $(v^{(w)})_i = w_i \, v_i$, for $i = 1, 2, \ldots, n$. Note that the use of weights increases the computational costs.

The sequence of calls to the routines comprising the suite is enforced: first, the set-up routine F11BAFP must be called, followed by the solver F11BBFP. F11BCFP can be called either when F11BBFP is carrying out a monitoring step or after F11BBFP has completed its tasks. Incorrect sequencing will raise an error condition.

In general, it is not possible to recommend one method in preference to another. RGMRES is often used in the solution of systems arising from partial differential equations. On the other hand, it can easily stagnate when the size m of the orthogonal basis is too small, or the preconditioner is not good enough. CGS can be the fastest method, but the computed residuals can exhibit instability which may greatly affect the convergence and quality of the solution. Bi-CGSTAB(ℓ) seems robust and reliable, but it can be slower than the other methods: if a preconditioner is used and $\ell > 1$, Bi-CGSTAB(ℓ) computes the solution of the preconditioned system $\bar{x}_k = Mx_k$: the preconditioning equations must be solved to obtain the required solution. The algorithm employed limits to 10% or less, when no intermediate monitoring

is requested, the number of times the preconditioner has to be thus applied compared with the total number of applications of the preconditioner. Also, when the termination criterion (2) is used, the CGS and Bi-CGSTAB(ℓ) methods will restart the iteration automatically when necessary in order to solve the given problem.

RGMRES can estimate internally the maximum singular value σ_1 of the iteration matrix, using $\sigma_1 \sim \|T\|_1$ where T is the upper triangular matrix obtained by QR factorization of the upper Hessenberg matrix generated by the Arnoldi process. The computational costs of this computation are negligible when compared to the overall costs.

Loss of orthogonality in the RGMRES method, or of bi-orthogonality in the Bi-CGSTAB(ℓ) method may degrade the solution and speed of convergence. For both methods, the algorithms employed include checks on the basis vectors so that the number of basis vectors used for a given super-iteration may be less than the value specified in the input parameter M. Also, the Bi-CGSTAB(ℓ) method will automatically restart the computation from the last available iterates, when the stability of the solution process requires it.

Termination criterion (3) involves only the residual (or norm of the residual) produced directly by the iteration process: this may differ from the norm of the true residual $\tilde{r}_k = b - Ax_k$, particularly when the norm of the residual is very small. Also, if the norm of the initial estimate of the solution is much larger than the norm of the exact solution, convergence can be achieved despite very large errors in the solution. On the other hand, termination criterion (3) is cheaper to use and inspects the progress of the actual iteration. Termination criterion (2) should be preferred in most cases, despite its slightly larger costs.

6.2 Parallelism Detail

Not applicable.

6.3 Accuracy

Not applicable.

6.4 Computational Costs

The computational costs of F11BAFP are negligible compared to the costs of F11BBFP.

7 References

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

[2] 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

[3] 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

[4] Saad Y and Schultz M (1986) GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems *SIAM J. Sci. Statist. Comput.* **7** 856–869

[5] Sonneveld P (1989) CGS, a fast Lanczos-type solver for nonsymmetric linear systems *SIAM J. Sci. Statist. Comput.* **10** 36–52

[6] van der Vorst H (1989) Bi–CGSTAB, A fast and smoothly converging variant of Bi–CG for the solution of nonsymmetric linear systems *SIAM J. Sci. Statist. Comput.* **13** 631–644

[7] Sleijpen G L G and Fokkema D R (1993) BiCGSTAB(ℓ) for linear equations involving matrices with complex spectrum *ETNA* **1** 11–32

[**8**] Arnoldi W (1951) The principle of minimized iterations in the solution of the matrix eigenvalue problem *Quart. Appl. Math.* **9** 17–29

[**9**] 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

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 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

```
*     F11BAFP 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)
      DOUBLE PRECISION DZERO
      PARAMETER        (DZERO=0.D0)
*     .. Scalars in Common ..
      DOUBLE PRECISION C1, C2, C3, C4, C5
      INTEGER          NX
*     .. Local Scalars ..
      DOUBLE PRECISION ANORM, ENORM, ENORML, SIGMAX, STPLHS, STPRHS, TOL
      INTEGER          I, ICNTXT, IFAIL, IREVCM, ITERM, ITN, IW, J,
     +                 LEVEL, LW, LWREQ, M, MAXITN, MB, ML, MLO, MLOMAX,
     +                 MONIT, MP, N, NB, NINTE, NINTI, NNZ, NNZC, NOVER,
     +                 NP
      LOGICAL          LOOP, ROOT, ZGRID
      CHARACTER        CHECK, DISTR, DUP, KIND, NORM, PRECON, 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,
     +                 F11BAFP, F11BBFP, F11BCFP, F11DFFP, F11DGFP,
     +                 F11XBFP, F11ZBFP, F11ZZFP, GMAT, GSOL, GVEC,
     +                 PRINTI, X04YAFP, Z01AAFP, Z01ABFP, Z01BBFP,
     +                 Z02EAFP
*     .. Intrinsic Functions ..
      INTRINSIC        ABS, MAX
*     .. Common blocks ..
      COMMON           /PROB/C1, C2, C3, C4, C5, NX
*     .. Executable Statements ..
      ROOT = Z01ACFP()
      IF (ROOT) WRITE (NOUT,*) 'F11BAFP Example Program Results'
*
*     Open input file on all processors
*
      OPEN (NIN,FILE='f11bafpe.d')
*
*     Skip heading in data file
*     Read size of processor grid
*
      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,*) FORMAT
      READ (NIN,*) LEVEL
*
*     Read 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 160
*
*     Set error checking level
*
      CALL Z02EAFP(ICNTXT,LEVEL,IFAIL)
*
*     Generate sparse matrix
*     Create block size for distribution
*
      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
      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 140
      END IF
*
*     Generate right-hand side vector
*
      CALL F01YEFP(ICNTXT,GVEC,N,V,IAINFO,IFAIL)
*
*     Set up block Jacobi preconditioner
*     Initialize 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
      CALL F11DFFP(ICNTXT,N,NNZ,A,IROW,ICOL,NOVER,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 F11BAFP(ICNTXT,METHOD,PRECON,NORM,DISTR,WEIGHT,ITERM,N,ML,M,
     +             TOL,MAXITN,ANORM,SIGMAX,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,NORM,DISTR,ITERM,N,
     +                      MAXITN,TOL,M,MONIT,MP,NP,MB)
*
*     Set initial approximation to solution
*
      DO 40 I = 1, ML
         U(I) = DZERO
   40 CONTINUE
*
*     Solve equations using reverse communication scheme
*
      LW = LWREQ
      IW = LWREQ + 1
      IREVCM = 0
      LOOP = .TRUE.
*
   60 CONTINUE
      CALL F11BBFP(ICNTXT,IREVCM,U,V,WORK,LW,IFAIL)
*
      IF (IREVCM.LE.-1) THEN
*
*     Compute  v = A**T * u (used only to compute ||A||)
*
         CALL F11XBFP(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 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 F11BCFP(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,''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
*
*     Get information about final solution
*
      CALL F11BCFP(ICNTXT,ITN,STPLHS,STPRHS,ANORM,SIGMAX,IFAIL)
*
*     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)
*
*     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 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_x + c4*w_y + c5*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, C4, C5
      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, 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 the 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, C4, C5
      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, 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 ..
      DOUBLE PRECISION TS(*)
*     .. Scalars in Common ..
      DOUBLE PRECISION C1, C2, C3, C4, C5
      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, 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) = 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,NORM,DISTR,ITERM,N,MAXITN,
     +                  TOL,M,MONIT,MP,NP,MB)
*
*     Prints a summary of the input parameters and options
*
*     .. Scalar Arguments ..
      DOUBLE PRECISION  TOL
      INTEGER           ITERM, M, MAXITN, MB, MONIT, MP, N, NOUT, NP
      CHARACTER         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
*
*     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

```
F11BAFP Example Program Data
  2     2                        :  MP, NP
    10                           :  NX
  'BICGSTAB'                     :  METHOD
  'N'  'I'    1    0             :  PRECON, NORM, ITERM, MONIT
    2                            :  M
   1.0D-09  10000                :  TOL, MAXITN
   '(8F8.4)'                     :  FORMAT
    0                            :  LEVEL
   1.0  2.0  1.0  -1.0  0.0      :  C1,...,C5
```

8.3 Example Results

```
F11BAFP 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)                 -  100
Block size used in the data distribution (MB)        -   25
Method used (METHOD)                                 -    BICGSTAB
Use the preconditioner (PRECON)                      -    N
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)        -10000
Order of BICGSTAB method (M)                         -    2
Monitoring frequency (MONIT)                         -    0

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

Number of iterations carried out (ITN)               -   26
```

```
Left-hand side of termination criterion (STPLHS)     -    9.12D-07
Right-hand side of termination criterion (STPRHS)    -    3.05D-06
Norm of the matrix of the coefficients (ANORM)       -    1.4520E+03

Error norm =    1.3565E-08

Solution vector
---------------

-0.0083  0.0165  0.0579  0.1157  0.1901  0.2810  0.3884  0.5124
 0.6529  0.8099 -0.0579 -0.0331  0.0083  0.0661  0.1405  0.2314
 0.3388  0.4628  0.6033  0.7603 -0.1405 -0.1157 -0.0744 -0.0165
 0.0579
 0.1488  0.2562  0.3802  0.5207  0.6777 -0.2562 -0.2314 -0.1901
-0.1322 -0.0579  0.0331  0.1405  0.2645  0.4050  0.5620 -0.4050
-0.3802 -0.3388 -0.2810 -0.2066 -0.1157 -0.0083  0.1157  0.2562
 0.4132
-0.5868 -0.5620 -0.5207 -0.4628 -0.3884 -0.2975 -0.1901 -0.0661
 0.0744  0.2314 -0.8017 -0.7769 -0.7355 -0.6777 -0.6033 -0.5124
-0.4050 -0.2810 -0.1405  0.0165 -1.0496 -1.0248 -0.9835 -0.9256
-0.8512
-0.7603 -0.6529 -0.5289 -0.3884 -0.2314 -1.3306 -1.3058 -1.2645
-1.2066 -1.1322 -1.0413 -0.9339 -0.8099 -0.6694 -0.5124 -1.6446
-1.6198 -1.5785 -1.5207 -1.4463 -1.3554 -1.2479 -1.1240 -0.9835
-0.8264
```