NAG Library Routine Document

F04QAF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

F04QAF solves sparse nonsymmetric equations, sparse linear least squares problems and sparse damped linear least squares problems, using a Lanczos algorithm.

2 Specification

```
SUBROUTINE F04QAF (M, N, B, X, SE, APROD, DAMP, ATOL, BTOL, CONLIM,          &
                   ITNLIM, MSGLVL, ITN, ANORM, ACOND, RNORM, ARNORM,         &
                   XNORM, WORK, RUSER, LRUSER, IUSER, LIUSER, INFORM,        &
                   IFAIL)
INTEGER            M, N, ITNLIM, MSGLVL, ITN, LRUSER, IUSER(LIUSER),         &
                   LIUSER, INFORM, IFAIL
REAL (KIND=nag_wp) B(M), X(N), SE(N), DAMP, ATOL, BTOL, CONLIM, ANORM,       &
                   ACOND, RNORM, ARNORM, XNORM, WORK(N,2),                   &
                   RUSER(LRUSER)
EXTERNAL           APROD
```

3 Description

F04QAF can be used to solve a system of linear equations

$$Ax = b \tag{1}$$

where A is an n by n sparse nonsymmetric matrix, or can be used to solve linear least squares problems, so that F04QAF minimizes the value ρ given by

$$\rho = \|r\|, \quad r = b - Ax \tag{2}$$

where A is an m by n sparse matrix and $\|r\|$ denotes the Euclidean length of r so that $\|r\|^2 = r^{\mathrm{T}}r$. A damping parameter, λ, may be included in the least squares problem in which case F04QAF minimizes the value ρ given by

$$\rho^2 = \|r\|^2 + \lambda^2\|x\|^2. \tag{3}$$

λ is supplied as the parameter DAMP and should of course be zero if the solution to problems (1) or (2) is required. Minimizing ρ in (3) is often called ridge regression.

F04QAF is based upon algorithm LSQR (see Paige and Saunders (1982a) and Paige and Saunders (1982b)) and solves the problems by an algorithm based upon the Lanczos process. The routine does not require A explicitly, but A is specified via APROD which must perform the operations $(y + Ax)$ and $(x + A^{\mathrm{T}}y)$ for a given n-element vector x and m element vector y. A parameter to APROD specifies which of the two operations is required on a given entry.

The routine also returns estimates of the standard errors of the sample regression coefficients (x_i, for $i = 1, 2, \ldots, n$) given by the diagonal elements of the estimated variance-covariance matrix V. When problem (2) is being solved and A is of full rank, then V is given by

$$V = s^2\left(A^{\mathrm{T}}A\right)^{-1}, \quad s^2 = \rho^2/(m-n), \quad m > n$$

and when problem (3) is being solved then V is given by

$$V = s^2\left(A^{\mathrm{T}}A + \lambda^2 I\right)^{-1}, \quad s^2 = \rho^2/m, \quad \lambda \neq 0.$$

Let $\bar{A}$ denote the matrix

$$\bar{A} = A, \quad \lambda = 0; \quad \bar{A} = \begin{pmatrix} A \\ \lambda I \end{pmatrix}, \quad \lambda \neq 0, \tag{4}$$

let $\bar{r}$ denote the residual vector

$$\bar{r} = r, \quad \lambda = 0; \quad \bar{r} = \begin{pmatrix} b \\ 0 \end{pmatrix} - \bar{A}x, \quad \lambda \neq 0 \tag{5}$$

corresponding to an iterate x, so that $\rho = \|\bar{r}\|$ is the function being minimized, and let $\|A\|$ denote the Frobenius (Euclidean) norm of A. Then the routine accepts x as a solution if it is estimated that one of the following two conditions is satisfied:

$$\rho \leq tol_1 \|\bar{A}\| . \|x\| + tol_2 \|b\| \tag{6}$$

$$\|\bar{A}^{\mathrm{T}} \bar{r}\| \leq tol_1 \|\bar{A}\| \rho \tag{7}$$

where tol_1 and tol_2 are user-supplied tolerances which estimate the relative errors in A and b respectively. Condition (6) is appropriate for compatible problems where, in theory, we expect the residual to be zero and will be satisfied by an acceptable solution x to a compatible problem. Condition (7) is appropriate for incompatible systems where we do not expect the residual to be zero and is based on the observation that, in theory,

$$\bar{A}^{\mathrm{T}} \bar{r} = 0$$

when x is a solution to the least squares problem, and so (7) will be satisfied by an acceptable solution x to a linear least squares problem.

The routine also includes a test to prevent convergence to solutions, x, with unacceptably large elements. This can happen if A is nearly singular or is nearly rank deficient. If we let the singular values of $\bar{A}$ be

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$$

then the condition number of $\bar{A}$ is defined as

$$\mathrm{cond}(\bar{A}) = \sigma_1 / \sigma_k$$

where σ_k is the smallest nonzero singular value of $\bar{A}$ and hence k is the rank of $\bar{A}$. When $k < n$, then $\bar{A}$ is rank deficient, the least squares solution is not unique and F04QAF will normally converge to the minimal length solution. In practice $\bar{A}$ will not have exactly zero singular values, but may instead have small singular values that we wish to regard as zero.

The routine provides for this possibility by terminating if

$$\mathrm{cond}(\bar{A}) \geq c_{\mathrm{lim}} \tag{8}$$

where c_{lim} is a user-supplied limit on the condition number of $\bar{A}$. For problem (1) termination with this condition indicates that A is nearly singular and for problem (2) indicates that A is nearly rank deficient and so has near linear dependencies in its columns. In this case inspection of $\|r\|$, $\|A^{\mathrm{T}} r\|$ and $\|x\|$, which are all returned by the routine, will indicate whether or not an acceptable solution has been found. Condition (8), perhaps in conjunction with $\lambda \neq 0$, can be used to try and 'regularize' least squares solutions. A full discussion of the stopping criteria is given in Section 6 of Paige and Saunders (1982a).

Introduction of a nonzero damping parameter λ tends to reduce the size of the computed solution and to make its components less sensitive to changes in the data, and F04QAF is applicable when a value of λ is known *a priori*. To have an effect, λ should normally be at least $\sqrt{\epsilon}\|A\|$ where ϵ is the ***machine precision***. For further discussion see Paige and Saunders (1982b) and the references given there.

Whenever possible the matrix A should be scaled so that the relative errors in the elements of A are all of comparable size. Such a scaling helps to prevent the least squares problem from being unnecessarily sensitive to data errors and will normally reduce the number of iterations required. At the very least, in the absence of better information, the columns of A should be scaled to have roughly equal column length.

4 References

Paige C C and Saunders M A (1982a) LSQR: An algorithm for sparse linear equations and sparse least squares *ACM Trans. Math. Software* **8** 43–71

Paige C C and Saunders M A (1982b) Algorithm 583 LSQR: Sparse linear equations and least squares problems *ACM Trans. Math. Software* **8** 195–209

5 Parameters

1: M – INTEGER *Input*

On entry: m, the number of rows of the matrix A.

Constraint: M ≥ 1.

2: N – INTEGER *Input*

On entry: n, the number of columns of the matrix A.

Constraint: N ≥ 1.

3: B(M) – REAL (KIND=nag_wp) array *Input/Output*

On entry: the right-hand side vector b.

On exit: B is overwritten.

4: X(N) – REAL (KIND=nag_wp) array *Output*

On exit: the solution vector x.

5: SE(N) – REAL (KIND=nag_wp) array *Output*

On exit: the estimates of the standard errors of the components of x. Thus SE(i) contains an estimate of $\sqrt{\nu_{ii}}$, where ν_{ii} is the ith diagonal element of the estimated variance-covariance matrix V. The estimates returned in SE will be the lower bounds on the actual estimated standard errors, but will usually be correct to at least one significant figure.

6: APROD – SUBROUTINE, supplied by the user. *External Procedure*

APROD must perform the operations $y := y + Ax$ and $x := x + A^{\mathrm{T}}y$ for given vectors x and y.

The specification of APROD is:

```
SUBROUTINE APROD (MODE, M, N, X, Y, RUSER, LRUSER, IUSER, LIUSER)
INTEGER            MODE, M, N, LRUSER, IUSER(LIUSER), LIUSER
REAL (KIND=nag_wp) X(N), Y(M), RUSER(LRUSER)
```

1: MODE – INTEGER *Input/Output*

On entry: specifies which operation is to be performed.

MODE $= 1$
APROD must compute $y + Ax$.

MODE $= 2$
APROD must compute $x + A^{\mathrm{T}}y$.

On exit: may be used as a flag to indicate a failure in the computation of $y + Ax$ or $x + A^{\mathrm{T}}y$. If MODE is negative on exit from APROD, F04QAF will exit immediately with IFAIL set to MODE.

2: M – INTEGER *Input*

On entry: m, the number of rows of A.

3: N – INTEGER *Input*

On entry: n, the number of columns of A.

4: X(N) – REAL (KIND=nag_wp) array *Input/Output*

On entry: the vector x.

On exit: if MODE $= 1$, X must be unchanged.

If MODE $= 2$, X must contain $x + A^{\mathrm{T}}y$.

5: Y(M) – REAL (KIND=nag_wp) array *Input/Output*

On entry: the vector y.

On exit: if MODE $= 1$, Y must contain $y + Ax$.

If MODE $= 2$, Y must be unchanged.

6: RUSER(LRUSER) – REAL (KIND=nag_wp) array *User Workspace*

7: LRUSER – INTEGER *Input*

8: IUSER(LIUSER) – INTEGER array *User Workspace*

9: LIUSER – INTEGER *Input*

APROD is called with the parameters RUSER, LRUSER, IUSER and LIUSER as supplied to F04QAF. You are free to use the arrays RUSER, LRUSER, IUSER and LIUSER to supply information to APROD as an alternative to using COMMON global variables.

APROD must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which F04QAF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

7: DAMP – REAL (KIND=nag_wp) *Input*

On entry: the value λ. If either problem (1) or problem (2) is to be solved, then DAMP must be supplied as zero.

8: ATOL – REAL (KIND=nag_wp) *Input*

On entry: the tolerance, tol_1, of the convergence criteria (6) and (7); it should be an estimate of the largest relative error in the elements of A. For example, if the elements of A are correct to about 4 significant figures, then ATOL should be set to about 5×10^{-4}. If ATOL is supplied as less than ϵ, where ϵ is the ***machine precision***, then the value ϵ is used instead of ATOL.

9: BTOL – REAL (KIND=nag_wp) *Input*

On entry: the tolerance, tol_2, of the convergence criterion (6); it should be an estimate of the largest relative error in the elements of B. For example, if the elements of B are correct to about 4 significant figures, then BTOL should be set to about 5×10^{-4}. If BTOL is supplied as less than ϵ, then the value ϵ is used instead of BTOL.

10: CONLIM – REAL (KIND=nag_wp) *Input*

On entry: the value c_{lim} of equation (8); it should be an upper limit on the condition number of $\bar{A}$. CONLIM should not normally be chosen much larger than $1.0/\mathrm{ATOL}$. If CONLIM is supplied as zero, then the value $1.0/\epsilon$ is used instead of CONLIM.

11: ITNLIM – INTEGER *Input/Output*

On entry: an upper limit on the number of iterations. If ITNLIM ≤ 0, then the value N is used in place of ITNLIM, but for ill-conditioned problems a higher value of ITNLIM is likely to be necessary.

On exit: unchanged unless ITNLIM ≤ 0 on entry, in which case it is set to N.

12: MSGLVL – INTEGER *Input*

On entry: the level of printing from F04QAF. If MSGLVL ≤ 0, then no printing occurs, but otherwise messages will be output on the advisory message channel (see X04ABF). A description of the printed output is given in Section 9.1. The level of printing is determined as follows:

MSGLVL ≤ 0
No printing.

MSGLVL $= 1$
A brief summary is printed just prior to return from F04QAF.

MSGLVL ≥ 2
A summary line is printed periodically to monitor the progress of F04QAF, together with a brief summary just prior to return from F04QAF.

13: ITN – INTEGER *Output*

On exit: the number of iterations performed.

14: ANORM – REAL (KIND=nag_wp) *Output*

On exit: an estimate of $\left\|\bar{A}\right\|$ for the matrix $\bar{A}$ of (4).

15: ACOND – REAL (KIND=nag_wp) *Output*

On exit: an estimate of $\mathrm{cond}\left(\bar{A}\right)$ which is a lower bound.

16: RNORM – REAL (KIND=nag_wp) *Output*

On exit: an estimate of $\|\bar{r}\|$ for the residual, $\bar{r}$, of (5) corresponding to the solution x returned in X. Note that $\|\bar{r}\|$ is the function being minimized.

17: ARNORM – REAL (KIND=nag_wp) *Output*

On exit: an estimate of the $\left\|\bar{A}^{\mathrm{T}}\bar{r}\right\|$ corresponding to the solution x returned in X.

18: XNORM – REAL (KIND=nag_wp) *Output*

On exit: an estimate of $\|x\|$ for the solution x returned in X.

19: WORK$(\mathrm{N}, 2)$ – REAL (KIND=nag_wp) array *Workspace*

20: RUSER(LRUSER) – REAL (KIND=nag_wp) array *User Workspace*

RUSER is not used by F04QAF, but is passed directly to APROD and may be used to pass information to this routine as an alternative to using COMMON global variables.

21: LRUSER – INTEGER *Input*

On entry: the dimension of the array RUSER as declared in the (sub)program from which F04QAF is called.

Constraint: LRUSER ≥ 1.

22: IUSER(LIUSER) – INTEGER array *User Workspace*

IUSER is not used by F04QAF, but is passed directly to APROD and may be used to pass information to this routine as an alternative to using COMMON global variables.

23: LIUSER – INTEGER *Input*

On entry: the dimension of the array IUSER as declared in the (sub)program from which F04QAF is called.

Constraint: LIUSER ≥ 1.

24: INFORM – INTEGER *Output*

On exit: the reason for termination of F04QAF.

INFORM $= 0$
The exact solution is $x = 0$. No iterations are performed in this case.

INFORM $= 1$
The termination criterion of (6) has been satisfied with tol_1 and tol_2 as the values supplied in ATOL and BTOL respectively.

INFORM $= 2$
The termination criterion of (7) has been satisfied with tol_1 as the value supplied in ATOL.

INFORM $= 3$
The termination criterion of (6) has been satisfied with tol_1 and/or tol_2 as the value ϵ, where ϵ is the ***machine precision***. One or both of the values supplied in ATOL and BTOL must have been less than ϵ and was too small for this machine.

INFORM $= 4$
The termination criterion of (7) has been satisfied with tol_1 as the value ϵ. The value supplied in ATOL must have been less than ϵ and was too small for this machine.

The values INFORM $= 5$, 6 and 7 correspond to failure with IFAIL $= 2$, 3 and 4 respectively (see Section 6) and when IFAIL is negative INFORM will be set to the same negative value.

25: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this parameter, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: IFAIL $= 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL $= 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

IFAIL < 0

A negative value of IFAIL indicates an exit from F04QAF because you have set MODE negative in APROD. The value of IFAIL will be the same as your setting of MODE.

IFAIL $= 1$

On entry, M < 1,
or N < 1,
or LRUSER < 1,
or LIUSER < 1.

IFAIL $= 2$

The condition of (8) has been satisfied for the value of c_{lim} supplied in CONLIM. If this failure is unexpected you should check that APROD is working correctly. Although conditions (6) or (7) have not been satisfied, the values returned in RNORM, ARNORM and XNORM may nevertheless indicate that an acceptable solution has been reached.

IFAIL $= 3$

The condition of (8) has been satisfied for the value $c_{\text{lim}} = 1.0/\epsilon$, where ϵ is the ***machine precision***. The matrix $\bar{A}$ is nearly singular or rank deficient and the problem is too ill-conditioned for this machine. If this failure is unexpected, you should check that APROD is working correctly.

IFAIL $= 4$

The limit on the number of iterations has been reached. The number of iterations required by F04QAF and the condition of the matrix $\bar{A}$ can depend strongly on the scaling of the problem. Poor scaling of the rows and columns of A should be avoided whenever possible.

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.8 in the Essential Introduction for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.7 in the Essential Introduction for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.6 in the Essential Introduction for further information.

7 Accuracy

When the problem is compatible, the computed solution x will satisfy the equation

$$r = b - Ax,$$

where an estimate of $\|r\|$ is returned in the parameter RNORM. When the problem is incompatible, the computed solution x will satisfy the equation

$$\bar{A}^{\mathrm{T}}\bar{r} = e,$$

where an estimate of $\|e\|$ is returned in the parameter ARNORM. See also Section 6.2 of Paige and Saunders (1982b).

8 Parallelism and Performance

F04QAF is not threaded by NAG in any implementation.

F04QAF makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

The time taken by F04QAF is likely to be principally determined by the time taken in APROD, which is called twice on each iteration, once with MODE $= 1$ and once with MODE $= 2$. The time taken per iteration by the remaining operations in F04QAF is approximately proportional to $\max(m, n)$.

The Lanczos process will usually converge more quickly if A is pre-conditioned by a nonsingular matrix M that approximates A in some sense and is also chosen so that equations of the form $My = c$ can efficiently be solved for y. For a discussion of pre-conditioning, see the F11 Chapter Introduction. In the context of F04QAF, problem (1) is equivalent to

$$\left(AM^{-1}\right)y = b, \quad Mx = y$$

and problem (2) is equivalent to minimizing

$$\rho = \|r\|, \quad r = b - \left(AM^{-1}\right)y, \quad Mx = y.$$

Note that the normal matrix $\left(AM^{-1}\right)^{\mathrm{T}}\left(AM^{-1}\right) = M^{-\mathrm{T}}(A^{\mathrm{T}}A)M^{-1}$ so that the pre-conditioning AM^{-1} is equivalent to the pre-conditioning $M^{-\mathrm{T}}(A^{\mathrm{T}}A)M^{-1}$ of the normal matrix $A^{\mathrm{T}}A$.

Pre-conditioning can be incorporated into F04QAF simply by coding APROD to compute $y + AM^{-1}x$ and $x + M^{-\mathrm{T}}A^{\mathrm{T}}y$ in place of $y + Ax$ and $x + A^{\mathrm{T}}y$ respectively, and then solving the equations $Mx = y$ for x on return from F04QAF. The quantity $y + AM^{-1}x$ should be computed by solving $Mz = x$ for z and then computing $y + Az$, and $x + M^{-\mathrm{T}}A^{\mathrm{T}}y$ should be computed by solving $M^{\mathrm{T}}z = A^{\mathrm{T}}y$ for z and then forming $x + z$.

9.1 Description of the Printed Output

When MSGLVL > 0, then F04QAF will produce output (except in the case where the routine fails with IFAIL $= 1$) on the advisory message channel (see X04ABF).

When MSGLVL ≥ 2 then a summary line is printed periodically giving the following information:

Output	Meaning
`ITN`	Iteration number, k.
`X(1)`	The first element of the current iterate x_k.
`FUNCTION`	The current value of the function, ρ, being minimized.
`COMPAT`	An estimate of $\|\bar{r}_k\|/\|b\|$, where $\bar{r}_k$ is the residual corresponding to x_k. This value should converge to zero (in theory) if and only if the problem is compatible. `COMPAT` decreases monotonically.
`INCOMPAT`	An estimate of $\left\|\bar{A}^{\mathrm{T}}\bar{r}_k\right\|/\left(\left\|\bar{A}\right\|\|\bar{r}_k\|\right)$ which should converge to zero if and only if at the solution ρ is nonzero. `INCOMPAT` is not usually monotonic.
`NRM(ABAR)`	A monotonically increasing estimate of $\left\|\bar{A}\right\|$.
`COND(ABAR)`	A monotonically increasing estimate of the condition number $\mathrm{cond}\left(\bar{A}\right)$.

10 Example

This example solves the linear least squares problem

$$\min \rho = \|r\|, \quad r = b - Ax$$

where A is the 13 by 12 matrix and b is the 13 element vector given by

$$A = \begin{pmatrix} 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}, \quad b = -h^2 \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ -h^{-3} \end{pmatrix}$$

with $h = 0.1$.

Such a problem can arise by considering the Neumann problem on a rectangle

$$\frac{\delta u}{\delta n} = 0$$

$$\frac{\delta u}{\delta n} = 0 \quad \boxed{\nabla^2 u = g(x, y)} \quad \frac{\delta u}{\delta n} = 0 \quad \int_c u = 1$$

$$\frac{\delta u}{\delta n} = 0$$

where C is the boundary of the rectangle, and discretizing as illustrated below with the square mesh

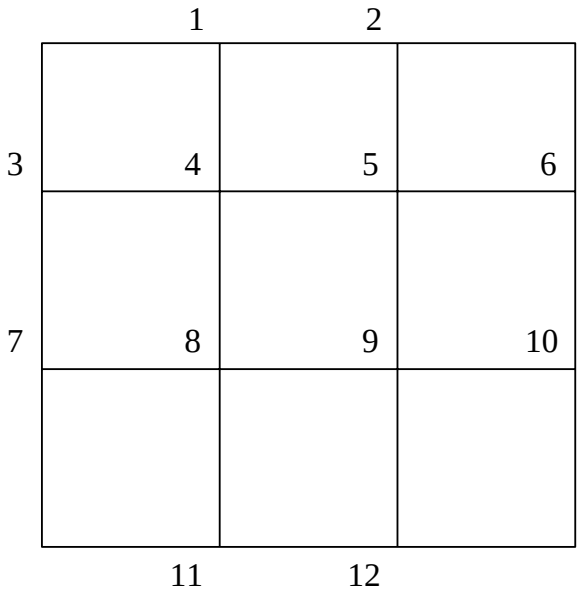

Figure 1

The 12 by 12 symmetric part of A represents the difference equations and the final row comes from the normalizing condition. The example program has $g(x, y) = 1$ at all the internal mesh points, but apart from this is written in a general manner so that the number of rows (NROWS) and columns (NCOLS) in the grid can readily be altered.

10.1 Program Text

```
!   F04QAF Example Program Text
!   Mark 25 Release. NAG Copyright 2014.

    Module f04qafe_mod
!     F04QAF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Accessibility Statements ..
      Private
      Public                                :: aprod
!     .. Parameters ..
      Integer, Parameter, Public            :: iset = 1, liuser = 1,            &
                                               lruser = 1, nin = 5, nout = 6
!     .. Local Scalars ..
      Integer, Public, Save                 :: ncols, nrows
    Contains
      Subroutine atimes(n,x,y)
!       Called by routine aprod. Returns Y = Y + A*X,
!       where A is not stored explicitly.

!       .. Scalar Arguments ..
        Integer, Intent (In)                  :: n
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (In)       :: x(n)
        Real (Kind=nag_wp), Intent (Inout)    :: y(n)
!       .. Local Scalars ..
        Integer                               :: i, i1, i2, i3, il, j
!       .. Executable Statements ..
        Do j = 1, nrows - 2
          y(j) = y(j) + x(j) - x(j+nrows-1)
        End Do
        Do j = 1, ncols - 2
          i = j*nrows - 1
          y(i) = y(i) + x(i) - x(i+1)
          i1 = i + 1
          il = i1 + nrows - 3
          Do i = i1, il
            i2 = i - nrows
            If (j==1) i2 = i2 + 1
            i3 = i + nrows
            If (j==ncols-2) i3 = i3 - 1
            y(i) = y(i) - x(i2) - x(i-1) + 4.0_nag_wp*x(i) - x(i+1) - x(i3)
          End Do
          i = il + 1
          y(i) = y(i) - x(i-1) + x(i)
        End Do
        Do j = n - nrows + 3, n
          y(j) = y(j) - x(j-nrows+1) + x(j)
        End Do
        Return
      End Subroutine atimes
      Subroutine aprod(mode,m,n,x,y,ruser,lruser,iuser,liuser)

!       APROD returns
!       Y = Y + A*X          when MODE = 1
!       X = X + ( A**T )*Y   when MODE = 2
!       for a given X and Y.

!       .. Scalar Arguments ..
        Integer, Intent (In)                  :: liuser, lruser, m, n
        Integer, Intent (Inout)               :: mode
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Inout)    :: ruser(lruser), x(n), y(m)
        Integer, Intent (Inout)               :: iuser(liuser)
!       .. Local Scalars ..
```

```
        Integer                               :: j, j1, j2
!       .. Executable Statements ..
        If (mode/=2) Then
          Call atimes(n,x,y)
          Do j = 1, nrows - 2
            y(m) = y(m) + x(j)
          End Do
          Do j = 1, ncols - 2
            y(m) = y(m) + x(j*nrows-1) + x(j*nrows+nrows-2)
          End Do
          Do j = m - nrows + 2, n
            y(m) = y(m) + x(j)
          End Do
        Else
          Call atimes(n,y,x)
          Do j = 1, nrows - 2
            x(j) = x(j) + y(m)
          End Do
          Do j = 1, ncols - 2
            j1 = j*nrows - 1
            j2 = j1 + nrows - 1
            x(j1) = x(j1) + y(m)
            x(j2) = x(j2) + y(m)
          End Do
          Do j = m - nrows + 2, n
            x(j) = x(j) + y(m)
          End Do
        End If
        Return
      End Subroutine aprod
    End Module f04qafe_mod
    Program f04qafe

!     F04QAF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: f04qaf, nag_wp, x04abf
      Use f04qafe_mod, Only: aprod, iset, liuser, lruser, ncols, nin, nout,    &
                             nrows
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)                    :: acond, anorm, arnorm, atol,     &
                                               btol, c, conlim, damp, h, rnorm, &
                                               xnorm
      Integer                               :: i1, ifail, inform, itn, itnlim, &
                                               k, m, msglvl, n, outchn
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable       :: b(:), se(:), work(:,:), x(:)
      Real (Kind=nag_wp)                    :: ruser(lruser)
      Integer                               :: iuser(liuser)
!     .. Executable Statements ..
      Write (nout,*) 'F04QAF Example Program Results'
      Write (nout,*)
      Flush (nout)
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) nrows, ncols
      n = ncols*nrows - 4
      m = n + 1
      Allocate (b(m),se(n),work(n,2),x(n))
      outchn = nout
      Call x04abf(iset,outchn)

      h = 0.1_nag_wp
!     Initialize rhs and other quantities required by F04QAF.
!     Convergence will be sooner if we do not regard A as exact,
!     so atol is not set to zero.
      b(1:n) = 0.0_nag_wp
      c = -h**2
      i1 = nrows
```

```
      Do k = 3, ncols
        b(i1:(i1+nrows-3)) = c
        i1 = i1 + nrows
      End Do
      b(m) = 1.0_nag_wp/h
      damp = 0.0_nag_wp
      atol = 1.0E-5_nag_wp
      btol = 1.0E-4_nag_wp
      conlim = 1.0_nag_wp/atol
      itnlim = 100
!     * Set msglvl to 2 to get output at each iteration *
      msglvl = 1

!     ifail: behaviour on error exit
!             =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
      ifail = 0
      Call f04qaf(m,n,b,x,se,aprod,damp,atol,btol,conlim,itnlim,msglvl,itn, &
        anorm,acond,rnorm,arnorm,xnorm,work,ruser,lruser,iuser,liuser,inform, &
        ifail)

      Write (nout,*)
      Write (nout,*) 'Solution returned by F04QAF'
      Write (nout,99999) x(1:n)
      Write (nout,*)
      Write (nout,99998) 'Norm of the residual = ', rnorm

99999 Format (1X,5F9.3)
99998 Format (1X,A,1P,E12.2)
    End Program f04qafe
```

10.2 Program Data

```
F04QAF Example Program Data
   4   4                : nrows, ncols
```

10.3 Program Results

```
 F04QAF Example Program Results

 OUTPUT FROM SPARSE LINEAR LEAST-SQUARES SOLVER.

 LEAST-SQUARES SOLUTION OF  A*X = B

 THE MATRIX A HAS     13 ROWS AND     12 COLS
 THE DAMPING PARAMETER IS  DAMP =      0.00E+00

 ATOL =     1.00E-05     CONLIM =      1.00E+05
 BTOL =     1.00E-04     ITNLIM =    100

 NO. OF ITERATIONS  =      2
 STOPPING CONDITION =      2
 ( THE LEAST-SQRS SOLN IS GOOD ENOUGH, GIVEN ATOL )


 ACTUAL        NORM( RBAR ), NORM( X )        1.15E-02    4.33E+00
 NORM( TRANSPOSE( ABAR )*RBAR )               9.16E-15
 ESTIMATES OF  NORM( ABAR ), COND( ABAR )     4.12E+00    2.45E+00

 Solution returned by F04QAF
     1.250    1.250    1.250    1.247    1.247
     1.250    1.250    1.247    1.247    1.250
     1.250    1.250

 Norm of the residual =     1.15E-02
```