

NAG Toolbox for MATLAB

f04qa

1 Purpose

f04qa solves sparse nonsymmetric equations, sparse linear least-squares problems and sparse damped linear least-squares problems, using a Lanczos algorithm.

2 Syntax

```
[b, x, se, itnlim, itn, anorm, acond, rnorm, arnorm, xnorm, user,
inform, ifail] = f04qa(n, b, aprod, damp, atol, btol, conlim, itnlim,
msglvl, lruser, liuser, 'm', m, 'user', user)
```

3 Description

f04qa can be used to solve a system of linear equations

$$Ax = b \quad (1)$$

where A is an n by n sparse nonsymmetric matrix, or can be used to solve linear least-squares problems, so that f04qa minimizes the value ρ given by

$$\rho = \|r\|, \quad r = b - Ax \quad (2)$$

where A is an m by n sparse matrix and $\|r\|$ denotes the Euclidean length of r so that $\|r\|^2 = r^T r$. A damping parameter, λ , may be included in the least-squares problem in which case f04qa minimizes the value ρ given by

$$\rho^2 = \|r\|^2 + \lambda^2 \|x\|^2. \quad (3)$$

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.

f04qa 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 function does not require A explicitly, but A is specified via a user-supplied (sub)program **aprod** which must perform the operations $(y + Ax)$ and $(x + A^T y)$ 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 function also returns estimates of the standard errors of the sample regression coefficients (x_i , for $i = 1, 2, \dots, n$) 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

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

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

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

Let \bar{A} denote the matrix

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

let \bar{r} denote the residual vector

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

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 function accepts x as a solution if it is estimated that one of

the following two conditions is satisfied:

$$\rho \leq tol_1 \|\bar{A}\| \cdot \|x\| + tol_2 \|b\| \quad (6)$$

$$\|\bar{A}^T \bar{r}\| \leq tol_1 \|\bar{A}\| \rho \quad (7)$$

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,

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

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 function 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

$$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n \geq 0$$

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

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

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 f04qa 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 function provides for this possibility by terminating if

$$\text{cond}(\bar{A}) \geq c_{\text{lim}} \quad (8)$$

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 $\|r\|$, $\|A^T r\|$ and $\|x\|$, which are all returned by the function, 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 ‘regularise’ 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 f04qa 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

5.1 Compulsory Input Parameters

1: **n – int32 scalar**

n , the number of columns of the matrix A .

Constraint: $n \geq 1$.

2: **b(m) – double array**

The right-hand side vector b .

3: **aprod – string containing name of m-file**

aprod must perform the operations $y := y + Ax$ and $x := x + A^T y$ for given vectors x and y .

Its specification is:

```
[mode, x, y, user] = aprod(mode, m, n, x, y, user, lruser, liuser)
```

Input Parameters

1: **mode – int32 scalar**

Specifies which operation is to be performed.

mode = 1

aprod must compute $y + Ax$.

mode = 2

aprod must compute $x + A^T y$.

May be used as a flag to indicate a failure in the computation of $y + Ax$ or $x + A^T y$. If **mode** is negative on exit from **aprod**, f04qa will exit immediately with **ifail** set to **mode**.

2: **m – int32 scalar**

m , the number of rows of A .

3: **n – int32 scalar**

n , the number of columns of A .

4: **x(n) – double array**

The vector x .

If **mode** = 1, **x** must be unchanged.

If **mode** = 2, **x** must contain $x + A^T y$.

5: **y(m) – double array**

The vector y .

If **mode** = 1, **y** must contain $y + Ax$.

If **mode** = 2, **y** must be unchanged.

- 6: **user** – Any MATLAB object
 7: **lruser** – int32 scalar
 8: **liuser** – int32 scalar
- aprod** is called from f04qa with **user** as supplied to f04qa

Output Parameters

- 1: **mode** – int32 scalar
 Specifies which operation is to be performed.
mode = 1
aprod must compute $y + Ax$.
mode = 2
aprod must compute $x + A^T y$.
 May be used as a flag to indicate a failure in the computation of $y + Ax$ or $x + A^T y$. If **mode** is negative on exit from **aprod**, f04qa will exit immediately with **ifail** set to **mode**.
- 2: **x(n)** – double array
 The vector x .
 If **mode** = 1, **x** must be unchanged.
 If **mode** = 2, **x** must contain $x + A^T y$.
- 3: **y(m)** – double array
 The vector y .
 If **mode** = 1, **y** must contain $y + Ax$.
 If **mode** = 2, **y** must be unchanged.
- 4: **user** – Any MATLAB object
aprod is called from f04qa with **user** as supplied to f04qa

- 4: **damp** – double scalar
 The value λ . If either problem (1) or problem (2) is to be solved, then **damp** must be supplied as zero.
- 5: **atol** – double scalar
 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**.
- 6: **btol** – double scalar
 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**.

7: **conlim – double scalar**

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 $1.0/\text{atol}$. If **conlim** is supplied as zero, then the value $1.0/\epsilon$ is used instead of **conlim**.

8: **itnlim – int32 scalar**

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.

9: **msglvl – int32 scalar**

The level of printing from f04qa. If **msglvl** ≤ 0 , then no printing occurs, but otherwise messages will be output on the advisory message channel (see x04ab). A description of the printed output is given in Section 8.1. The level of printing is determined as follows:

msglvl ≤ 0

No printing.

msglvl = 1

A brief summary is printed just prior to return from f04qa.

msglvl ≥ 2

A summary line is printed periodically to monitor the progress of f04qa, together with a brief summary just prior to return from f04qa.

10: **lruser – int32 scalar**

Constraint: **lruser** ≥ 1 .

11: **liuser – int32 scalar**

Constraint: **liuser** ≥ 1 .

5.2 Optional Input Parameters1: **m – int32 scalar**

Default: The dimension of the array **b**.

m , the number of rows of the matrix A .

Constraint: **m** ≥ 1 .

2: **user – Any MATLAB object**

user is not used by f04qa, but is passed to **aproduct**. Note that for large objects it may be more efficient to use a global variable which is accessible from the m-files than to use **user**.

5.3 Input Parameters Omitted from the MATLAB Interface

work

5.4 Output Parameters1: **b(m) – double array**

The array is overwritten.

2: **x(n) – double array**

The solution vector x .

3: **se(n) – double array**

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 i th 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.

4: **itnlim – int32 scalar**

Unchanged unless **itnlim** ≤ 0 on entry, in which case it is set to **n**.

5: **itn – int32 scalar**

The number of iterations performed.

6: **anorm – double scalar**

An estimate of $\|\bar{A}\|$ for the matrix \bar{A} of (4).

7: **acond – double scalar**

An estimate of $\text{cond}(\bar{A})$ which is a lower bound.

8: **rnorm – double scalar**

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.

9: **arnorm – double scalar**

An estimate of the $\|\bar{A}^T \bar{r}\|$ corresponding to the solution x returned in **x**.

10: **xnorm – double scalar**

An estimate of $\|x\|$ for the solution x returned in **x**.

11: **user – Any MATLAB object**

user is not used by f04qa, but is passed to **aproduct**. Note that for large objects it may be more efficient to use a global variable which is accessible from the m-files than to use **user**.

12: **inform – int32 scalar**

The reason for termination of f04qa.

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.

13: **ifail** – int32 scalar

0 unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Errors detected by the function:

ifail < 0

A negative value of **ifail** indicates an exit from f04qa because you have set **mode** negative in the user-supplied (sub)program **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 user-supplied (sub)program **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 $c_{lim} = 1.0/\epsilon$, 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 user-supplied (sub)program **aprod** is working correctly.

ifail = 4

The limit on the number of iterations has been reached. The number of iterations required by f04qa 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.

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 **norm**. When the problem is incompatible, the computed solution x will satisfy the equation

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

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

8 Further Comments

The time taken by f04qa is likely to be principally determined by the time taken in the user-supplied (sub)program **aproduct**, 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 f04qa 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 f04qa, problem (1) is equivalent to

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

and problem (2) is equivalent to minimizing

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

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

Pre-conditioning can be incorporated into f04qa simply by coding the user-supplied (sub)program **aproduct** to compute $y + AM^{-1}x$ and $x + M^{-T}A^T y$ in place of $y + Ax$ and $x + A^T y$ respectively, and then solving the equations $Mx = y$ for x on return from f04qa. The quantity $y + AM^{-1}x$ should be computed by solving $Mz = x$ for z and then computing $y + Az$, and $x + M^{-T}A^T y$ should be computed by solving $M^T z = A^T y$ for z and then forming $x + z$.

8.1 Description of the Printed Output

When **msglvl** > 0, then f04qa will produce output (except in the case where the function fails with **ifail** = 1) on the advisory message channel (see x04ab).

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 $\ \bar{A}^T \bar{r}_k\ /(\ \bar{A}\ \ \bar{r}_k\)$ 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 $\ \bar{A}\ $. |
| COND(ABAR) | A monotonically increasing estimate of the condition number $\text{cond}(\bar{A})$. |

9 Example

```

f04qa_aprod.m

function [mode, x, y, user] = aprod(mode, m, n, x, y, user)

    % a is passed as user

    if (mode == 1)
        y = y + user*x;
    else
        x = x + transpose(user)*y;
    end

n = int32(12);
b = [0;
     0;
     0;
     -0.01;
     -0.01;
     0;
     0;
     -0.01;
     -0.01;
     0;
     0;
     0;
     10];
damp = 0;
atol = 1e-05;
btol = 0.0001;
conlim = 99999.999999999999;
itnlim = int32(100);
msglvl = int32(1);
a = [ [ 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, 0, -1, 0, 0, 1 ],
      [ 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 1 ],
      [ 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1 ] ];

[bOut, x, se, itnlimOut, itn, anorm, acond, rnorm, arnorm, xnorm, ruser,
inform, ifail] = ...
    f04qa(n, b, 'f04qa_aprod', damp, atol, btol, conlim, itnlim, msglvl,
'user', a)

```

OUTPUT FROM F04QAF.

```

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 )          1.31E-14
ESTIMATES OF   NORM( ABAR ), COND( ABAR )      4.12E+00    2.45E+00
bOut =
-0.2883
-0.2883
-0.2883
-0.2894
-0.2894
-0.2883
-0.2883
-0.2894
-0.2894
-0.2883
-0.2883
-0.2883
-0.0012
x =
1.2500
1.2500
1.2500
1.2467
1.2467
1.2500
1.2500
1.2467
1.2467
1.2500
1.2500
1.2500
1.2500
se =
0.0014
0.0014
0.0014
0.0028
0.0028
0.0014
0.0014
0.0028
0.0028
0.0014
0.0014
0.0014
itnlimOut =
100
itn =
2
anorm =
4.1231
acond =
2.4537
rnorm =
0.0115
arnorm =
1.3090e-14
xnorm =
4.3263
ruser =
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 |
| inform = | | | | | | | | | | | | |
| | | 2 | | | | | | | | | | |
| ifail = | | | | | | | | | | | | |
| | | 0 | | | | | | | | | | |
