NAG Toolbox

nag_linsys_real_gen_norm_rcomm (f04yd)

1 Purpose

nag_linsys_real_gen_norm_rcomm (f04yd) estimates the 1-norm of a real rectangular matrix without accessing the matrix explicitly. It uses reverse communication for evaluating matrix products. The function may be used for estimating condition numbers of square matrices.

2 Syntax

```
[irevcm, x, y, estnrm, work, iwork, ifail] = nag_linsys_real_gen_norm_rcomm
(irevcm, x, y, estnrm, seed, work, iwork, 'm', m, 'n', n, 't', t)
[irevcm, x, y, estnrm, work, iwork, ifail] = f04yd(irevcm, x, y, estnrm, seed,
work, iwork, 'm', m, 'n', n, 't', t)
```

3 Description

nag_linsys_real_gen_norm_rcomm (f04yd) computes an estimate (a lower bound) for the 1-norm

$$\|A\|_1 = \max_{1\le j\le n}\sum_{i=1}^{m}\left|a_{ij}\right| \tag{1}$$

of an m by n real matrix $A = \left(a_{ij}\right)$. The function regards the matrix A as being defined by a user-supplied 'Black Box' which, given an $n \times t$ matrix X (with $t \ll n$) or an $m \times t$ matrix Y, can return AX or $A^{\mathrm{T}}Y$. A reverse communication interface is used; thus control is returned to the calling program whenever a matrix product is required.

Note: this function is **not recommended** for use when the elements of A are known explicitly; it is then more efficient to compute the 1-norm directly from formula (1) above.

The **main use** of the function is for estimating $\left\|B^{-1}\right\|_1$ for a square matrix, B, and hence the **condition number** $\kappa_1(B) = \|B\|_1\left\|B^{-1}\right\|_1$, without forming B^{-1} explicitly ($A = B^{-1}$ above).

If, for example, an LU factorization of B is available, the matrix products $B^{-1}X$ and $B^{-\mathrm{T}}Y$ required by nag_linsys_real_gen_norm_rcomm (f04yd) may be computed by back- and forward-substitutions, without computing B^{-1}.

The function can also be used to estimate 1-norms of matrix products such as $A^{-1}B$ and ABC, without forming the products explicitly. Further applications are described by Higham (1988).

Since $\|A\|_\infty = \|A^{\mathrm{T}}\|_1$, nag_linsys_real_gen_norm_rcomm (f04yd) can be used to estimate the ∞-norm of A by working with A^{T} instead of A.

The algorithm used is described in Higham and Tisseur (2000).

4 References

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

Higham N J and Tisseur F (2000) A block algorithm for matrix 1-norm estimation, with an application to 1-norm pseudospectra *SIAM J. Matrix. Anal. Appl.* **21** 1185–1201

5 Parameters

Note: this function uses **reverse communication.** Its use involves an initial entry, intermediate exits and re-entries, and a final exit, as indicated by the argument **irevcm**. Between intermediate exits and re-entries, **all arguments other than x and y must remain unchanged**.

5.1 Compulsory Input Parameters

1: **irevcm** – INTEGER

On initial entry: must be set to 0.

On intermediate re-entry: **irevcm** must be unchanged.

2: $\mathbf{x}(ldx,:)$ – REAL (KIND=nag_wp) array

The first dimension of the array **x** must be at least **n**.

The second dimension of the array **x** must be at least $\max(1,\mathbf{t})$.

On initial entry: need not be set.

On intermediate re-entry: if $\mathbf{irevcm} = 2$, must contain $A^{\mathrm{T}}Y$.

3: $\mathbf{y}(ldy,:)$ – REAL (KIND=nag_wp) array

The first dimension of the array **y** must be at least **m**.

The second dimension of the array **y** must be at least $\max(1,\mathbf{t})$.

On initial entry: need not be set.

On intermediate re-entry: if $\mathbf{irevcm} = 1$, must contain AX.

4: **estnrm** – REAL (KIND=nag_wp)

On initial entry: need not be set.

On intermediate re-entry: must not be changed.

5: **seed** – INTEGER

The seed used for random number generation.

If $\mathbf{t} = 1$, **seed** is not used.

Constraint: if $\mathbf{t} > 1$, $\mathbf{seed} \geq 1$.

6: $\mathbf{work}(\mathbf{m} \times \mathbf{t})$ – REAL (KIND=nag_wp) array

7: $\mathbf{iwork}(\mathbf{2} \times \mathbf{n} + \mathbf{5} \times \mathbf{t} + \mathbf{20})$ – INTEGER array

On initial entry: need not be set.

On intermediate re-entry: must not be changed.

5.2 Optional Input Parameters

1: **m** – INTEGER

Default: the first dimension of the arrays **y**, **work**. (An error is raised if these dimensions are not equal.)

The number of rows of the matrix A.

Constraint: $\mathbf{m} \geq 0$.

2: **n** – INTEGER

Default: the first dimension of the array **x** and the dimension of the array **iwork**. (An error is raised if these dimensions are not equal.)

n, the number of columns of the matrix A.

Constraint: $\mathbf{n} \geq 0$.

3: **t** – INTEGER

Suggested value: $\mathbf{t} = 2$.

Default: $\mathbf{t} = 2$

Default: the second dimension of the arrays **x**, **y**. (An error is raised if these dimensions are not equal.)

The number of columns t of the matrices X and Y. This is a argument that can be used to control the accuracy and reliability of the estimate and corresponds roughly to the number of columns of A that are visited during each iteration of the algorithm.

If $\mathbf{t} \geq 2$ then a partly random starting matrix is used in the algorithm.

Constraint: $1 \leq \mathbf{t} \leq \mathbf{m}$.

5.3 Output Parameters

1: **irevcm** – INTEGER

On intermediate exit: $\mathbf{irevcm} = 1$ or 2, and **x** contains the $n \times t$ matrix X and **y** contains the $m \times t$ matrix Y. The calling program must

(a) if $\mathbf{irevcm} = 1$, evaluate AX and store the result in **y**
or
if $\mathbf{irevcm} = 2$, evaluate $A^{\mathrm{T}}Y$ and store the result in **x**,

(b) call nag_linsys_real_gen_norm_rcomm (f04yd) once again, with all the other arguments unchanged.

On final exit: $\mathbf{irevcm} = 0$.

2: $\mathbf{x}(ldx, :)$ – REAL (KIND=nag_wp) array

The first dimension of the array **x** will be **n**.

The second dimension of the array **x** will be $\max(1, \mathbf{t})$.

On intermediate exit: if $\mathbf{irevcm} = 1$, contains the current matrix X.

On final exit: the array is undefined.

3: $\mathbf{y}(ldy, :)$ – REAL (KIND=nag_wp) array

The first dimension of the array **y** will be **m**.

The second dimension of the array **y** will be $\max(1, \mathbf{t})$.

On intermediate exit: if $\mathbf{irevcm} = 2$, contains the current matrix Y.

On final exit: the array is undefined.

4: **estnrm** – REAL (KIND=nag_wp)

On final exit: an estimate (a lower bound) for $\|A\|_1$.

5: **work**($\mathbf{m} \times \mathbf{t}$) – REAL (KIND=nag_wp) array
6: **iwork**($\mathbf{2} \times \mathbf{n} + \mathbf{5} \times \mathbf{t} + \mathbf{20}$) – INTEGER array

7: **ifail** – INTEGER

On final exit: **ifail** $= 0$ unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

Internal error; please contact NAG.

ifail $= -1$

Constraint: **irevcm** $= 0$, 1 or 2.

On initial entry, **irevcm** $= \langle value \rangle$.
Constraint: **irevcm** $= 0$.

ifail $= -2$

Constraint: $\mathbf{m} \geq 0$.

ifail $= -3$

Constraint: $\mathbf{n} \geq 0$.

ifail $= -5$

Constraint: $ldx \geq \mathbf{n}$.

ifail $= -7$

Constraint: $ldy \geq \mathbf{m}$.

ifail $= -9$

Constraint: $1 \leq \mathbf{t} \leq \mathbf{m}$.

ifail $= -10$

Constraint: if $\mathbf{t} > 1$, **seed** ≥ 1.

ifail $= -99$

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

ifail $= -399$

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

ifail $= -999$

Dynamic memory allocation failed.

7 Accuracy

In extensive tests on **random** matrices of size up to $m = n = 450$ the estimate **estnrm** has been found always to be within a factor two of $\|A\|_1$; often the estimate has many correct figures. However, matrices exist for which the estimate is smaller than $\|A\|_1$ by an arbitrary factor; such matrices are very unlikely to arise in practice. See Higham and Tisseur (2000) for further details.

8 Further Comments

8.1 Timing

For most problems the time taken during calls to nag_linsys_real_gen_norm_rcomm (f04yd) will be negligible compared with the time spent evaluating matrix products between calls to nag_linsys_real_gen_norm_rcomm (f04yd).

The number of matrix products required depends on the matrix A. At most six products of the form $Y = AX$ and five products of the form $X = A^{\mathrm{T}}Y$ will be required. The number of iterations is independent of the choice of t.

8.2 Overflow

It is your responsibility to guard against potential overflows during evaluation of the matrix products. In particular, when estimating $\left\|B^{-1}\right\|_1$ using a triangular factorization of B, nag_linsys_real_gen_norm_rcomm (f04yd) should not be called if one of the factors is exactly singular – otherwise division by zero may occur in the substitutions.

8.3 Choice of *t*

The argument t controls the accuracy and reliability of the estimate. For $t = 1$, the algorithm behaves similarly to the LAPACK estimator xLACON. Increasing t typically improves the estimate, without increasing the number of iterations required.

For $t \geq 2$, random matrices are used in the algorithm, so for repeatable results the same value of **seed** should be used each time.

A value of $t = 2$ is recommended for new users.

8.4 Use in Conjunction with NAG Library Routines

To estimate the 1-norm of the inverse of a matrix A, the following skeleton code can normally be used:

```
...  code to factorize A ...
 if (A is not singular)
   icase = 0;
   [icase, x, estnrm, work, iwork, ifail] = f04yd(icase, x, estnrm, work,
iwork);
   while (icase ~= 0)
     if (icase == 1)
        ...  code to compute inv(A)*x ...
     else
        ...  code to compute inv(transpose(A))*x ...
     end
     [icase, x, estnrm, work, iwork, ifail] = f04yd(icase, x, estnrm, work,
iwork);
   end
 end
```

To compute $A^{-1}X$ or $A^{-\mathrm{T}}Y$, solve the equation $AY = X$ or $A^{\mathrm{T}}X = Y$, storing the result in **y** or **x** respectively. The code will vary, depending on the type of the matrix A, and the NAG function used to factorize A.

The factorization will normally have been performed by a suitable function from Chapters F01, F03 or F07. Note also that many of the 'Black Box' functions in Chapter F04 for solving systems of equations also return a factorization of the matrix. The example program in Section 10 illustrates how nag_linsys_real_gen_norm_rcomm (f04yd) can be used in conjunction with NAG Toolbox functions for LU factorization of a real matrix nag_lapack_dgetrf (f07ad).

It is straightforward to use nag_linsys_real_gen_norm_rcomm (f04yd) for the following other types of matrix, using the named functions for factorization and solution:

nonsymmetric tridiagonal (nag_matop_real_gen_tridiag_lu (f01le) and nag_linsys_real_tridiag_fac_solve (f04le));

nonsymmetric almost block-diagonal (nag_matop_real_gen_blkdiag_lu (f01lh) and nag_linsys_real_blkdiag_fac_solve (f04lh));

nonsymmetric band (nag_lapack_dgbtrf (f07bd) and nag_lapack_dgbtrs (f07be));

symmetric positive definite (nag_lapack_dpotrf (f07fd) and nag_lapack_dpotrs (f07fe));

symmetric positive definite band (nag_lapack_dpbtrf (f07hd) and nag_lapack_dpbtrs (f07he));

symmetric positive definite tridiagonal (nag_lapack_dptsv (f07ja), nag_lapack_dpttrf (f07jd) and nag_lapack_dpttrs (f07je));

symmetric positive definite variable bandwidth (nag_matop_real_vband_posdef_fac (f01mc) and nag_linsys_real_posdef_vband_solve (f04mc));

symmetric positive definite sparse (nag_sparse_real_symm_precon_ichol (f11ja) and nag_sparse_real_symm_precon_ichol_solve (f11jb));

symmetric indefinite (nag_lapack_dsptrf (f07pd) and nag_lapack_dsptrs (f07pe));

nonsymmetric sparse (nag_sparse_direct_real_gen_lu (f11me) and nag_sparse_direct_real_gen_solve (f11mf); note that nag_sparse_direct_real_gen_cond (f11mg) can also be used here).

9 Example

For this function two examples are provided. There is a single example program for nag_linsys_real_gen_norm_rcomm (f04yd), with a main program and the code to solve the two example problems is given in Example 1 (EX1) and Example 2 (EX2).

Example 1 (EX1)

This example estimates the condition number $\|A\|_1\left\|A^{-1}\right\|_1$ of the matrix A given by

$$A = \begin{pmatrix} 0.7 & -0.2 & 1.0 & 0.0 & 2.0 & 0.1 \\ 0.3 & 0.7 & 0.0 & 1.0 & 0.9 & 0.2 \\ 0.0 & 0.0 & 0.2 & 0.7 & 0.0 & -1.1 \\ 0.0 & 3.4 & -0.7 & 0.2 & 0.1 & 0.1 \\ 0.0 & -4.0 & 0.0 & 1.0 & 9.0 & 0.0 \\ 0.4 & 1.2 & 4.3 & 0.0 & 6.2 & 5.9 \end{pmatrix}.$$

Example 2 (EX2)

This example estimates the condition number of the sparse matrix A (stored in symmetric coordinate storage format) given by

$$A = \begin{pmatrix} 0.0 & 0.0 & 0.0 & 1.0 & 0.0 \\ 3.0 & 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 2.0 & 0.0 & 2.0 & 0.0 \\ 2.0 & 0.0 & 4.0 & 0.0 & 5.0 \\ 0.0 & 1.0 & 2.0 & 0.0 & 0.0 \end{pmatrix}.$$

9.1 Program Text

```
function f04yd_example

fprintf('f04yd example results\n\n');

% Example 1: Compute the condition number of a dense matrix
fprintf('\nExample 1\n');

a = [0.7,  -0.2,   1.0,   0.0,   2.0,   0.1;
     0.3,   0.7,   0.0,   1.0,   0.9,   0.2;
     0.0,   0.0,   0.2,   0.7,   0.0,  -1.1;
     0.0,   3.4,  -0.7,   0.2,   0.1,   0.1;
     0.0,  -4.0,   0.0,   1.0,   9.0,   0.0;
```

```
     0.4,   1.2,   4.3,   0.0,   6.2,   5.9];
t = nag_int(2);
m = 6;
n = 6;
x = zeros(n, t);
y = zeros(m, t);
estnrm = 0;
seed = nag_int(354);
irevcm = nag_int(0);
work = zeros(n*t, 1);
iwork = zeros(2*n+5*t+20, 1, nag_int_name);

nrma =  norm(a, 1);
fprintf('\nThe norm of a is %6.2f\n', nrma);

% Estimate the norm of a^(-1) without explicitly forming a^(-1)

% Perform an LU factorization so that A=LU where L and U are lower and upper
% triangular.
[a, ipiv, info] = f07ad(a);

first = true;

while first || (irevcm ~= 0)
  first = false;

  [irevcm, x, y, estnrm, work, iwork, ifail] = ...
      f04yd( ...
             irevcm, x, y, estnrm, seed, work, iwork);

  switch irevcm
    case 1
      % Compute y = inv(a)*x
      [y, info] = f07ae('n', a, ipiv, x);
    case 2
      % Compute x = transpose(inv(a))*y
      [x, info] = f07ae('t', a, ipiv, y);
    otherwise
  end
end

fprintf('The estimated norm of inverse(a) is: %6.2f\n', estnrm);
fprintf('\nEstimated condition number of a: %6.2f\n', estnrm*nrma);

% Example 2: Compute the condition number of a sparse matrix
% (stored in symmetric coordinate storage format)
fprintf('\nExample 2\n');

t = nag_int(2);
n = nag_int(5);
nz = nag_int(10);
a = zeros(4*nz, 1);
icn = zeros(4*nz, 1, nag_int_name);
irn = zeros(4*nz, 1, nag_int_name);
a(1:nz)   = [3; 2; 1; 2; 1; 4; 2; 1; 2; 5];
irn(1:nz) = [2; 4; 2; 3; 5; 4; 5; 1; 3; 4];
icn(1:nz) = [1; 1; 2; 2; 2; 3; 3; 4; 4; 5];

x = zeros(n, t);
y = zeros(n, t);
estnrm = 0;
seed = nag_int(412);
irevcm = nag_int(0);
work = zeros(n*t, 1);
iwork = zeros(2*n+5*t+20, 1, nag_int_name);

% Compute 1-norm of a
nrma =  0;
for i = 1:n
  asum = 0;
  for j = 1:nz
```

```
    if (icn(j)==i)
      asum = asum + abs(a(j));
    end
  end
  nrma = max(nrma,asum);
end

fprintf('\nThe norm of a is %6.2f\n', nrma);

% Perform an LU factorization so that A=LU where L and U are lower and upper
% triangular.
abort = [true; true; false; false];
[a, irn, icn, ikeep, w, idisp, ifail] = ...
  f01br(n, nz, a, irn, icn, abort);

% Compute an estimate of the 1-norm of inv(a)

first = true;

while first || (irevcm ~= 0)
  first = false;

  [irevcm, x, y, estnrm, work, iwork, ifail] = ...
    f04yd( ...
           irevcm, x, y, estnrm, seed, work, iwork);

  switch irevcm
    case 1
      % Compute y = inv(a)*x
      for i=1:2
        [y(:, i), resid] = f04ax( ...
                                  a, icn, ikeep, x(:, i), irevcm, idisp);
      end
    case 2
      % Compute x = transpose(inv(a))*y
      for i=1:2
        [x(:, i), resid] = f04ax( ...
                                  a, icn, ikeep, y(:, i), irevcm, idisp);
      end
    otherwise
  end
end

fprintf('The estimated norm of inverse(a) is: %6.2f\n', estnrm);
fprintf('\nEstimated condition number of a: %6.2f\n', estnrm*nrma);
```

9.2 Program Results

```
     f04yd example results

Example 1

The norm of a is  18.20
The estimated norm of inverse(a) is:   2.97

Estimated condition number of a:  54.14

Example 2

The norm of a is   6.00
The estimated norm of inverse(a) is:   3.37

Estimated condition number of a:  20.20
```