NAG Toolbox for MATLAB

f04yc

1 Purpose

f04yc estimates the 1-norm of a real matrix without accessing the matrix explicitly. It uses reverse communication for evaluating matrix-vector products. The function may be used for estimating matrix condition numbers.

2 Syntax

```
[icase, x, estnrm, work, iwork, ifail] = f04yc(icase, x, estnrm, work, iwork, 'n', n)
```

3 Description

f04yc computes an estimate (a lower bound) for the 1-norm

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

of an n by n real matrix $A = (a_{ij})$. The function regards the matrix A as being defined by a user-supplied 'Black Box' which, given an input vector x, can return either of the matrix-vector products Ax or $A^{\mathrm{T}}x$. A reverse communication interface is used; thus control is returned to the calling program whenever a matrix-vector 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 $\|B^{-1}\|_1$, and hence the **condition number** $\kappa_1(B) = \|B\|_1 \|B^{-1}\|_1$, without forming B^{-1} explicitly ($A = B^{-1}$ above).

If, for example, an LU factorization of B is available, the matrix-vector products $B^{-1}x$ and $B^{-\mathrm{T}}x$ required by f04yc 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 $\|A\|_\infty = \|A^{\mathrm{T}}\|_1$, f04yc can be used to estimate the ∞-norm of A by working with A^{T} instead of A.

The algorithm used is based on a method given by Hager 1984 and is described by Higham 1988. A comparison of several techniques for condition number estimation is given by Higham 1987.

4 References

Hager W W 1984 Condition estimates *SIAM J. Sci. Statist. Comput.* **5** 311–316

Higham N J 1987 A survey of condition number estimation for triangular matrices *SIAM Rev.* **29** 575–596

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

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 **parameter ICASE**. Between intermediate exits and re-entries, **all parameters other than x must remain unchanged**.

5.1 Compulsory Input Parameters

1: **icase – int32 scalar**

On initial entry: must be set to 0.

2: **x(n) – double array**

On initial entry: need not be set.

On intermediate re-entry: must contain Ax (if **icase** $= 1$) or $A^{\mathrm{T}}x$ (if **icase** $= 2$).

3: **estnrm – double scalar**

On initial entry: need not be set.

4: **work(n) – double array**

On initial entry: need not be set.

5: **iwork(n) – int32 array**

5.2 Optional Input Parameters

1: **n – int32 scalar**

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

On initial entry: n, the order of the matrix A.

Constraint: $\mathbf{n} \geq 1$.

5.3 Input Parameters Omitted from the MATLAB Interface

None.

5.4 Output Parameters

1: **icase – int32 scalar**

On intermediate exit: **icase** $= 1$ or 2, and $\mathbf{x}(i)$, for $i = 1, 2, \ldots, n$, contain the elements of a vector x. The calling program must

(a) evaluate Ax (if **icase** $= 1$) or $A^{\mathrm{T}}x$ (if **icase** $= 2$),

(b) place the result in **x**, and

(c) call f04yc once again, with all the other parameters unchanged.

On final exit: **icase** $= 0$.

2: **x(n) – double array**

On intermediate exit: contains the current vector x.

On final exit: the array is undefined.

3: **estnrm – double scalar**

On intermediate exit: should not be changed.

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

4: **work(n) – double array**

On final exit: contains a vector v such that $v = Aw$ where **estnrm** $= \|v\|_1 / \|w\|_1$ (w is not returned). If $A = B^{-1}$ and **estnrm** is large, then v is an approximate null vector for B.

5: **iwork(n) – int32 array**

6: **ifail – int32 scalar**

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

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

On entry, **n** < 1.

7 Accuracy

In extensive tests on **random** matrices of size up to $n = 100$ the estimate **estnrm** has been found always to be within a factor eleven 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 1988 for further details.

8 Further Comments

8.1 Timing

The total time taken within f04yc is proportional to n. For most problems the time taken during calls to f04yc will be negligible compared with the time spent evaluating matrix-vector products between calls to f04yc.

The number of matrix-vector products required varies from 4 to 11 (or is 1 if $n = 1$). In most cases 4 or 5 products are required; it is rare for more than 7 to be needed.

8.2 Overflow

It is your responsibility to guard against potential overflows during evaluation of the matrix-vector products. In particular, when estimating $\left\|B^{-1}\right\|_1$ using a triangular factorization of B, f04yc should not be called if one of the factors is exactly singular – otherwise division by zero may occur in the substitutions.

8.3 Use in Conjunction with NAG Fortran 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] = f04yc(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] = f04yc(icase, x, estnrm, work,
iwork);
  end
end
```

To compute $A^{-1}x$ or $A^{-\mathrm{T}}x$, solve the equation $Ay = x$ or $A^{\mathrm{T}}y = x$ for y, overwriting y on x. The code will vary, depending on the type of the matrix A, and the NAG function used to factorize A.

Note that if A is any type of **symmetric** matrix, then $A = A^{\mathrm{T}}$, and the `if`statement after the `while` can be reduced to:

```
...  code to compute inv(A)*x ...
```

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 9 illustrates how f04yc can be used in conjunction with NAG Library functions for two important types of matrix: full nonsymmetric matrices (factorized by f03af) and sparse nonsymmetric matrices (factorized by f01br).

It is straightforward to use f04yc for the following other types of matrix, using the named functions for factorization and solution:

nonsymmetric tridiagonal (f01le and f04le);
nonsymmetric almost block-diagonal (f01lh and f04lh);
nonsymmetric band (f07bd and f07be);
symmetric positive-definite (f03ae and f04ag, or f07fd and f07fe);
symmetric positive-definite band (f07hd and f07he);
symmetric positive-definite tridiagonal (f07ja, f07jd and f07je);
symmetric positive-definite variable bandwidth (f01mc and f04mc);
symmetric positive-definite sparse (f11ja and f11jb);
symmetric indefinite (f07pd and f07pe).

9 Example

```
a = [ 1.5,  2.0,  3.0, -2.1,  0.3;
      2.5,  3.0, -4.0,  2.3, -1.1;
      3.5,  4.0,  0.5, -3.1, -1.4;
     -0.4, -3.2, -2.1,  3.1,  2.1;
      1.7,  3.7,  1.9, -2.2, -3.3];
icase = int32(0);
x = zeros(5, 1);
estnrm = 0;
work = zeros(5, 1);
iwork = zeros(5, 1, 'int32');
[icase, x, estnrm, work, iwork, ifail] = f04yc(icase, x, estnrm, work,
iwork);
while (icase > 0)
  if (icase ==1)
    x = a*x;
  elseif (icase == 2)
    x = transpose(a)*x;
  end
  [icase, x, estnrm, work, iwork, ifail] = f04yc(icase, x, estnrm, work,
iwork);
end
 estnrm
```

```
estnrm =
   15.9000
```