

NAG Library Function Document

nag_linsys_real_gen_norm_rcomm (f04ydc)

1 Purpose

nag_linsys_real_gen_norm_rcomm (f04ydc) 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 Specification

```
#include <nag.h>
#include <nagf04.h>

void nag_linsys_real_gen_norm_rcomm (Integer *irevcm, Integer m, Integer n,
    double x[], Integer pdx, double y[], Integer pdy, double *estnm,
    Integer t, Integer seed, double work[], Integer iwork[], NagError *fail)
```

3 Description

nag_linsys_real_gen_norm_rcomm (f04ydc) computes an estimate (a lower bound) for the 1-norm

$$\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^m |a_{ij}| \quad (1)$$

of an m 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 $n \times t$ matrix X (with $t \ll n$) or an $m \times t$ matrix Y , can return AX or $A^T Y$. 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 $\|B^{-1}\|_1$ for a square matrix, B , 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 products $B^{-1}X$ and $B^{-T}Y$ required by nag_linsys_real_gen_norm_rcomm (f04ydc) 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 $\|A\|_\infty = \|A^T\|_1$, nag_linsys_real_gen_norm_rcomm (f04ydc) 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 Arguments

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 **irevcn**. Between intermediate exits and re-entries, **all arguments other than \mathbf{x} and \mathbf{y} must remain unchanged**.

- 1: **irevcn** – Integer * *Input/Output*
On initial entry: must be set to 0.
On intermediate exit: **irevcn** = 1 or 2, and \mathbf{x} contains the $n \times t$ matrix X and \mathbf{y} contains the $m \times t$ matrix Y . The calling program must
- (a) if **irevcn** = 1, evaluate AX and store the result in \mathbf{y}
or
if **irevcn** = 2, evaluate $A^T Y$ and store the result in \mathbf{x} ,
 - (b) call nag_linsys_real_gen_norm_rcomm (f04ydc) once again, with all the other arguments unchanged.
- On intermediate re-entry:* **irevcn** must be unchanged.
On final exit: **irevcn** = 0.
- 2: **m** – Integer *Input*
On entry: the number of rows of the matrix A .
Constraint: $\mathbf{m} \geq 0$.
- 3: **n** – Integer *Input*
On entry: n , the number of columns of the matrix A .
Constraint: $\mathbf{n} \geq 0$.
- 4: $\mathbf{x}[\mathit{dim}]$ – double *Input/Output*
Note: the dimension, dim , of the array \mathbf{x} must be at least $\mathbf{pdx} \times \mathbf{t}$.
The (i, j) th element of the matrix X is stored in $\mathbf{x}[(j - 1) \times \mathbf{pdx} + i - 1]$.
On initial entry: need not be set.
On intermediate exit: if **irevcn** = 1, contains the current matrix X .
On intermediate re-entry: if **irevcn** = 2, must contain $A^T Y$.
On final exit: the array is undefined.
- 5: **pdx** – Integer *Input*
On entry: the stride separating matrix row elements in the array \mathbf{x} .
Constraint: $\mathbf{pdx} \geq \mathbf{n}$.
- 6: $\mathbf{y}[\mathit{dim}]$ – double *Input/Output*
Note: the dimension, dim , of the array \mathbf{y} must be at least $\mathbf{pdy} \times \mathbf{t}$.
The (i, j) th element of the matrix Y is stored in $\mathbf{y}[(j - 1) \times \mathbf{pdy} + i - 1]$.
On initial entry: need not be set.
On intermediate exit: if **irevcn** = 2, contains the current matrix Y .
On intermediate re-entry: if **irevcn** = 1, must contain AX .
On final exit: the array is undefined.

- 7: **pdy** – Integer *Input*
On entry: the stride separating matrix row elements in the array **y**.
Constraint: **pdy** \geq **m**.
- 8: **estnrm** – double * *Input/Output*
On initial entry: need not be set.
On intermediate re-entry: must not be changed.
On final exit: an estimate (a lower bound) for $\|A\|_1$.
- 9: **t** – Integer *Input*
On entry: the number of columns t of the matrices X and Y . This is an 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 **t** \geq 2 then a partly random starting matrix is used in the algorithm.
Suggested value: **t** = 2.
Constraint: $1 \leq \mathbf{t} \leq \mathbf{m}$.
- 10: **seed** – Integer *Input*
On entry: the seed used for random number generation.
 If **t** = 1, **seed** is not used.
Constraint: if **t** > 1, **seed** \geq 1.
- 11: **work**[**m** \times **t**] – double *Communication Array*
 12: **iwork**[**2** \times **n** + **5** \times **t** + **20**] – Integer *Communication Array*
On initial entry: need not be set.
On intermediate re-entry: must not be changed.
- 13: **fail** – NagError * *Input/Output*
 The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_BAD_PARAM

On entry, argument $\langle value \rangle$ had an illegal value.

NE_INT

On entry, **irevcn** = $\langle value \rangle$.

Constraint: **irevcn** = 0, 1 or 2.

On entry, **m** = $\langle value \rangle$.

Constraint: **m** \geq 0.

On entry, **n** = $\langle value \rangle$.

Constraint: **n** \geq 0.

On initial entry, **irevcn** = $\langle value \rangle$.

Constraint: **irevcn** = 0.

NE_INT_2

On entry, **m** = $\langle value \rangle$ and **t** = $\langle value \rangle$.
 Constraint: $1 \leq \mathbf{t} \leq \mathbf{m}$.

On entry, **pdx** = $\langle value \rangle$ and **n** = $\langle value \rangle$.
 Constraint: **pdx** \geq **n**.

On entry, **pdy** = $\langle value \rangle$ and **m** = $\langle value \rangle$.
 Constraint: **pdy** \geq **m**.

On entry, **t** = $\langle value \rangle$ and **seed** = $\langle value \rangle$.
 Constraint: if **t** > 1, **seed** \geq 1.

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.

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 Parallelism and Performance

Not applicable.

9 Further Comments**9.1 Timing**

For most problems the time taken during calls to `nag_linsys_real_gen_norm_rcomm` (f04ydc) will be negligible compared with the time spent evaluating matrix products between calls to `nag_linsys_real_gen_norm_rcomm` (f04ydc).

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 $X = A^T Y$ will be required. The number of iterations is independent of the choice of t .

9.2 Overflow

It is your responsibility to guard against potential overflows during evaluation of the matrix products. In particular, when estimating $\|B^{-1}\|_1$ using a triangular factorization of B , `nag_linsys_real_gen_norm_rcomm` (f04ydc) should not be called if one of the factors is exactly singular – otherwise division by zero may occur in the substitutions.

9.3 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.

9.4 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:

```
do {
  f04ydc(&irevcm,m,n,x,pdx,y,pdy,&estnrm,t,seed,work,iwork,&fail);
  if (irevcm == 1){
    .. Code to compute  $y = A^{-1} x$  ..
  }
  else if (irevcm == 2){
    .. Code to compute  $x = A^{-T} y$  ..
  }
} (while irevcm != 0)
```

To compute $A^{-1}X$ or $A^{-T}Y$, solve the equation $AY = X$ or $A^T X = Y$, 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` (f04ydc) can be used in conjunction with NAG C Library functions for LU factorization of a real matrix `nag_dgetrf` (f07adc).

It is straightforward to use `nag_linsys_real_gen_norm_rcomm` (f04ydc) for the following other types of matrix, using the named functions for factorization and solution:

- nonsymmetric band (`nag_dgbtrf` (f07bdc) and `nag_dgbtrs` (f07bec));
- symmetric positive definite (`nag_dpotrf` (f07fdc) and `nag_dpotsr` (f07fec));
- symmetric positive definite band (`nag_dpbtrf` (f07hdc) and `nag_dpbtrs` (f07hec));
- symmetric positive definite tridiagonal (`nag_dptsv` (f07jac), `nag_dpstrf` (f07jdc) and `nag_dpstrs` (f07jec));
- symmetric positive definite variable bandwidth (`nag_real_cholesky_skyline` (f01mcc) and `nag_real_cholesky_skyline_solve` (f04mcc));
- symmetric positive definite sparse (`nag_sparse_sym_chol_fac` (f11jac) and `nag_sparse_sym_precon_ichol_solve` (f11jbc));
- symmetric indefinite (`nag_dsprtf` (f07pdc) and `nag_dsprts` (f07pec));
- nonsymmetric sparse (`nag_superlu_lu_factorize` (f11mec) and `nag_superlu_solve_lu` (f11mfc); note that `nag_superlu_condition_number_lu` (f11mgc) can also be used here).

For upper or lower triangular matrices, no factorization function is needed: $Y = A^{-1}X$ and $X = A^{-T}Y$ may be computed by calls to `nag_dtrsv` (f16pjc) (or `nag_dtbsv` (f16pkc) if the matrix is banded, or `nag_dtpsv` (f16plc) if the matrix is stored in packed form).

10 Example

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

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

10.1 Program Text

```

/* nag_linsys_real_gen_norm_rcomm (f04ydc) Example Program.
 *
 * Copyright 2013, Numerical Algorithms Group.
 *
 * Mark 23, 2013.
 */

#include <nag.h>
#include <math.h>
#include <nag_stdlib.h>
#include <nagf04.h>
#include <nagf07.h>
#include <nagf16.h>

int main(void)
{
    /* Scalars */
    Integer      exit_status = 0, irevcm = 0, seed = 354;
    Integer      i, j, m, n, pda, pdx, pdy, t;
    double       cond = 0.0, nrma = 0.0, nrminv = 0.0;

    /* Local Arrays */
    Integer      *icomm = 0, *ipiv = 0;
    double       *a = 0, *work = 0, *x = 0, *y = 0;

    /* Nag Types */
    Nag_OrderType order;
    Nag_Error      fail;
    Nag_TransType trans;

    INIT_FAIL(fail);

#define A(I, J) a[(J-1)*pda + I-1]

    order = Nag_ColMajor;

    /* Output preamble */
    printf("nag_linsys_real_gen_norm_rcomm (f04ydc) ");
    printf("Example Program Results\n\n");
    fflush(stdout);

    /* Skip heading in data file */
    scanf("%*[\n] ");

    /* Read in the problem size and the value of the parameter t */
    scanf("%ld %ld %ld %*[\n] ", &m, &n, &t);

    pda = n;
    pdx = n;
    pdy = m;

    if (!(a = NAG_ALLOC(m*n, double)) ||
        !(x = NAG_ALLOC(n*t, double)) ||
        !(y = NAG_ALLOC(m*t, double)) ||
        !(work = NAG_ALLOC(m*t, double)) ||
        !(ipiv = NAG_ALLOC(n, Integer)) ||
        !(icomm = NAG_ALLOC(2*n+5*t+20, Integer))) {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    /* Read in the matrix a from data file */
    for (i = 1; i <= m; i++)
        for (j = 1; j <= n; j++) scanf(" %lf ", &A(i, j));
    scanf("%*[\n]");

    /* Compute the 1-norm of A */

```

```

nag_dge_norm(order, Nag_OneNorm, m, n, a, pda, &nrma, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dge_norm\n%s\n",fail.message);
    exit_status = 1;
    goto END;
}
printf("Estimated norm of A is: %7.2f\n\n",nrma);

/*
 * Estimate the norm of A(-1) without explicitly forming A(-1)
 */

/* Compute an LU factorization of A using nag_dgetrf (f07adc) */
nag_dgetrf(order, m, n, a, pda, ipiv, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dgetrf\n%s\n",fail.message);
    exit_status = 2;
    goto END;
}

/* Estimate the norm of A(-1) using the LU factors of A
 * nag_linsys_real_gen_norm_rcomm (f04ydc)
 * Estimate of the 1-norm of a real matrix
 */
do {
    nag_linsys_real_gen_norm_rcomm(&irevcm, m, n, x, pdx, y, pdy,
                                  &nrminv, t, seed, work, icomm, &fail);
    if (irevcm == 1)
    {
        /* Compute y = inv(A)*x by solving Ay = x */
        trans = Nag_NoTrans;
        nag_dgetrs(order, trans, n, t, a, pda, ipiv, x, pdx, &fail);
        if (fail.code != NE_NOERROR)
        {
            printf("Error from nag_dgetrs\n%s\n",fail.message);
            exit_status = 3;
            goto END;
        }
        for (i = 0; i < n*t; i++) y[i] = x[i];
    }

    else if (irevcm == 2)
    {
        /* Compute x = inv(A)T y by solving AT x = y */
        trans = Nag_Trans;
        nag_dgetrs(order, trans, n, t, a, pda, ipiv, y, pdy, &fail);
        if (fail.code != NE_NOERROR)
        {
            printf("Error from nag_dgetrs\n%s\n",fail.message);
            exit_status = 4;
            goto END;
        }
        for (i = 0; i < n*t; i++) x[i] = y[i];
    }
} while (irevcm != 0);

if (fail.code != NE_NOERROR)
{
    printf("Error from nag_linsys_real_gen_norm_rcomm (f04ydc)\n%s\n",
          fail.message);
    exit_status = 5;
    goto END;
}

printf("Estimated norm of inverse of A is: %7.2f\n\n",nrminv);

/* Compute and print the estimated condition number */
cond = nrma*nrminv;

```

```

printf("Estimated condition number of A is: %7.2f\n",cond);

END:
NAG_FREE(a);
NAG_FREE(x);
NAG_FREE(y);
NAG_FREE(work);
NAG_FREE(icomm);
NAG_FREE(ipiv);
return exit_status;
}

```

10.2 Program Data

nag_linsys_real_gen_norm_rcomm (f04ydc) Example Program Data

```

6      6      2                               :Values of m, n and t
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 of matrix a

```

10.3 Program Results

nag_linsys_real_gen_norm_rcomm (f04ydc) Example Program Results

```

Estimated norm of A is:    18.20

Estimated norm of inverse of A is:    2.97

Estimated condition number of A is:    54.14

```
