NAG Library Function Document

nag_dsycon (f07mgc)

1 Purpose

nag_dsycon (f07mgc) estimates the condition number of a real symmetric indefinite matrix \( A \), where \( A \) has been factorized by nag_dsytrf (f07mdc).

2 Specification

```c
#include <nag.h>
#include <nagf07.h>

void nag_dsycon (Nag_OrderType order, Nag_UploType uplo, Integer n,
const double a[], Integer pda, const Integer ipiv[], double anorm,
double *rcond, NagError *fail)
```

3 Description

nag_dsycon (f07mgc) estimates the condition number (in the 1-norm) of a real symmetric indefinite matrix \( A \):

\[
\kappa_1(A) = \| A \|_1 \| A^{-1} \|_1.
\]

Since \( A \) is symmetric, \( \kappa_1(A) = \kappa_\infty(A) = \| A \|_\infty \| A^{-1} \|_\infty. \)

Because \( \kappa_1(A) \) is infinite if \( A \) is singular, the function actually returns an estimate of the reciprocal of \( \kappa_1(A) \).

The function should be preceded by a call to nag_dsy_norm (f16rcc) to compute \( \| A \|_1 \) and a call to nag_dsytrf (f07mdc) to compute the Bunch–Kaufman factorization of \( A \). The function then uses Higham’s implementation of Hager’s method (see Higham (1988)) to estimate \( \| A^{-1} \|_1 \).

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

5 Arguments

1: \( \text{order} \) – Nag_OrderType

\( \text{Input} \)

\( \text{On entry:} \) the \text{order} argument specifies the two-dimensional storage scheme being used, i.e., row-major ordering or column-major ordering. C language defined storage is specified by \( \text{order} = \text{Nag_RowMajor}. \) See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

\( \text{Constraint:} \) \( \text{order} = \text{Nag_RowMajor} \) or \( \text{Nag.ColMajor}. \)

2: \( \text{uplo} \) – Nag_UploType

\( \text{Input} \)

\( \text{On entry:} \) specifies how \( A \) has been factorized.

\( \text{uplo} = \text{Nag.Upper} \)

\( A = PUDU^TPT, \) where \( U \) is upper triangular.

Mark 25
uplo = Nag_Lower
A = PLDL^TP, where L is lower triangular.

Constraint: uplo = Nag_Upper or Nag_Lower.

3: n – Integer  
   Input
   On entry: n, the order of the matrix A.
   Constraint: n ≥ 0.

4: a[dim] – const double  
   Input
   On entry: details of the factorization of A, as returned by nag_dsytrf (f07mdc).
   Note: the dimension, dim, of the array a must be at least max(1, pda × n).

5: pda – Integer  
   Input
   On entry: the stride separating row or column elements (depending on the value of order) of the matrix in the array a.
   Constraint: pda ≥ max(1, n).

6: ipiv[dim] – const Integer  
   Input
   On entry: details of the interchanges and the block structure of D, as returned by nag_dsytrf (f07mdc).
   Note: the dimension, dim, of the array ipiv must be at least max(1, n).

7: anorm – double  
   Input
   On entry: the 1-norm of the original matrix A, which may be computed by calling nag_dsy_norm (f16rcc) with its argument norm = Nag_OneNorm. anorm must be computed either before calling nag_dsytrf (f07mdc) or else from a copy of the original matrix A.
   Constraint: anorm ≥ 0.0.

8: rcond – double *  
   Output
   On exit: an estimate of the reciprocal of the condition number of A. rcond is set to zero if exact singularity is detected or the estimate underflows. If rcond is less than machine precision, A is singular to working precision.

9: fail – NagError *  
   Input/Output
   The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_ALLOC_FAIL
   Dynamic memory allocation failed.
   See Section 3.2.1.2 in the Essential Introduction for further information.

NE_BAD_PARAM
   On entry, argument (value) had an illegal value.

NE_INT
   On entry, n = (value).
   Constraint: n ≥ 0.
On entry, $pda = \langle value\rangle$.
Constraint: $pda > 0$.

**NE_INT_2**

On entry, $pda = \langle value\rangle$ and $n = \langle value\rangle$.
Constraint: $pda \geq \max(1, n)$.

**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.

An unexpected error has been triggered by this function. Please contact NAG. See Section 3.6.6 in the Essential Introduction for further information.

**NE_NO_LICENCE**

Your licence key may have expired or may not have been installed correctly. See Section 3.6.5 in the Essential Introduction for further information.

**NE_REAL**

On entry, $\text{anorm} = \langle value\rangle$.
Constraint: $\text{anorm} \geq 0.0$.

7 Accuracy

The computed estimate $\text{rcond}$ is never less than the true value $\rho$, and in practice is nearly always less than $10\rho$, although examples can be constructed where $\text{rcond}$ is much larger.

8 Parallelism and Performance

nag_dsycon ($f07mgc$) is not threaded by NAG in any implementation.

nag_dsycon ($f07mgc$) 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 function. Please also consult the Users’ Note for your implementation for any additional implementation-specific information.

9 Further Comments

A call to nag_dsycon ($f07mgc$) involves solving a number of systems of linear equations of the form $Ax = b$; the number is usually 4 or 5 and never more than 11. Each solution involves approximately $2n^2$ floating-point operations but takes considerably longer than a call to nag_dsysrs ($f07mec$) with one right-hand side, because extra care is taken to avoid overflow when $A$ is approximately singular.

The complex analogues of this function are nag_zhecon ($f07muc$) for Hermitian matrices and nag_zsycon ($f07nuc$) for symmetric matrices.
10 Example

This example estimates the condition number in the 1-norm (or $\infty$-norm) of the matrix $A$, where

$$
A = \begin{pmatrix}
2.07 & 3.87 & 4.20 & -1.15 \\
3.87 & -0.21 & 1.87 & 0.63 \\
4.20 & 1.87 & 1.15 & 2.06 \\
-1.15 & 0.63 & 2.06 & -1.81
\end{pmatrix}.
$$

Here $A$ is symmetric indefinite and must first be factorized by nag_dsytrf (f07mdc). The true condition number in the 1-norm is 75.68.

10.1 Program Text

/* nag_dsycon (f07mgc) Example Program. */
* Copyright 2014 Numerical Algorithms Group.
*/

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf07.h>
#include <nagf16.h>
#include <nagx02.h>

int main(void)
{
    /* Scalars */
    double anorm, rcond;
    Integer i, j, n, pda;
    Integer exit_status = 0;
    NagError fail;
    Nag_UploType uplo;
    Nag_OrderType order;
    /* Arrays */
    char nag_enum_arg[40];
    Integer *ipiv = 0;
    double *a = 0;

    INIT_FAIL(fail);

    printf("nag_dsycon (f07mgc) Example Program Results\n\n");

    /* Skip heading in data file */
    #ifdef _WIN32
        scanf_s("%*[\n] ");
    #else
        scanf("%*[\n] ");
    #endif
    #ifdef _WIN32
        scanf("%"NAG_IFMT"%*[\n] ", &n);
    #else
        scanf("%"NAG_IFMT"%*[\n] ", &n);
    #endif
    #ifdef NAG_COLUMN_MAJOR
        pda = n;
    #else
        pda = n;
    #endif

    INIT_FAIL(fail);

    printf("nag_dsycon (f07mgc) Example Program Results\n\n");
#endif
/* Allocate memory */
if (!(ipiv = NAG_ALLOC(n, Integer)) ||
    !(a = NAG_ALLOC(n * n, double)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

/* Read A from data file */
#ifdef _WIN32
    scanf_s(" %39s%\n\n", nag_enum_arg, _countof(nag_enum_arg));
#else
    scanf(" %39s%\n\n", nag_enum_arg);
#endif
/* nag_enum_name_to_value (x04nac).
  * Converts NAG enum member name to value */
uplo = (Nag_UploType) nag_enum_name_to_value(nag_enum_arg);
if (uplo == Nag_Upper)
{
    for (i = 1; i <= n; ++i)
        for (j = i; j <= n; ++j)
            #ifdef _WIN32
                scanf_s("%lf", &A(i, j));
            #else
                scanf("%lf", &A(i, j));
            #endif
    #ifdef _WIN32
        scanf_s("%\n\n");
    #else
        scanf("%\n\n");
    #endif
} else
{
    for (i = 1; i <= n; ++i)
        for (j = 1; j <= i; ++j)
            #ifdef _WIN32
                scanf_s("%lf", &A(i, j));
            #else
                scanf("%lf", &A(i, j));
            #endif
    #ifdef _WIN32
        scanf_s("%\n\n");
    #else
        scanf("%\n\n");
    #endif
}
/* Compute norm of A */
/* nag_dsy_norm (f16rcc).
  * l-norm, infinity-norm, Frobenius norm, largest absolute
  * element, real symmetric matrix */
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dsy_norm (f16rcc).\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Factorize A */
/* nag_dsytrf (f07mdc).
  * Bunch-Kaufman factorization of real symmetric indefinite
nag_dsytrf(order, uplo, n, a, pda, ipiv, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dsytrf (f07mdc).
            \n\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Estimate condition number */
/* nag_dsycon (f07mgc).
   * Estimate condition number of real symmetric indefinite
   * matrix, matrix already factorized by nag_dsytrf (f07mdc)
   */

nag_dsycon(order, uplo, n, a, pda, ipiv, anorm, &rcond,
             &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dsycon (f07mgc).
            \n\n", fail.message);
    exit_status = 1;
    goto END;
}

/* nag_machine_precision (x02ajc).
   * The machine precision
   */
if (rcond >= nag_machine_precision)
    printf("Estimate of condition number = %11.2e\n", 1.0/rcond);
else
    printf("A is singular to working precision\n");
END:
    NAG_FREE(ipiv);
    NAG_FREE(a);
    return exit_status;
}

10.2 Program Data

nag_dsycon (f07mgc) Example Program Data
4 :Value of n
   Nag_Lower :Value of uplo
   2.07
   3.87 -0.21
   4.20 1.87 1.15
-1.15 0.63 2.06 -1.81 :End of matrix A

10.3 Program Results

nag_dsycon (f07mgc) Example Program Results

Estimate of condition number = 7.57e+01