NAG C 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
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 Parameters

1:  order – Nag_OrderType  
   
   *Input*
   
   On entry: the order parameter specifies the two-dimensional storage scheme being used, i.e., row-major ordering or column-major ordering. C language defined storage is specified by order = Nag_RowMajor. See Section 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.
   
   Constraint: order = Nag_RowMajor or Nag_ColMajor.

2:  uplo – Nag_UploType  
   
   *Input*
   
   On entry: indicates how $A$ has been factorized as follows:
   
   if uplo = Nag_Upper, $A = PUDU^T P^T$, where $U$ is upper triangular;
   
   if uplo = Nag_Lower, $A = PLDL^T P^T$, where $L$ is lower triangular.
   
   Constraint: uplo = Nag_Upper or Nag_Lower.
3: \textbf{n} – Integer \hspace{1cm} \textit{Input}

\textit{On entry:} \textit{n}, the order of the matrix \textit{A}.

\textit{Constraint:} \textit{n} \geq 0.

4: \textbf{a}[	extit{dim}] – const double \hspace{1cm} \textit{Input}

\textbf{Note:} the dimension, \textit{dim}, of the array \textit{a} must be at least \textit{max}(1, \textit{pda} \times \textit{n}).

\textit{On entry:} details of the factorization of \textit{A}, as returned by \texttt{nag_dsytrf} (f07mdc).

5: \textbf{pda} – Integer \hspace{1cm} \textit{Input}

\textit{On entry:} the stride separating row or column elements (depending on the value of \texttt{order}) of the matrix in the array \textit{a}.

\textit{Constraint:} \textit{pda} \geq \textit{max}(1, \textit{n}).

6: \textbf{ipiv}[	extit{dim}] – const Integer \hspace{1cm} \textit{Input}

\textbf{Note:} the dimension, \textit{dim}, of the array \textit{ipiv} must be at least \textit{max}(1, \textit{n}).

\textit{On entry:} details of the interchanges and the block structure of \textit{D}, as returned by \texttt{nag_dsytrf} (f07mdc).

7: \textbf{anorm} – double \hspace{1cm} \textit{Input}

\textit{On entry:} the 1-norm of the \textbf{original} matrix \textit{A}, which may be computed by calling \texttt{nag_dsy_norm} (f16rcc). \textbf{anorm} must be computed either \textbf{before} calling \texttt{nag_dsytrf} (f07mdc) or else from a copy of the original matrix \textit{A}.

\textit{Constraint:} \textbf{anorm} \geq 0.0.

8: \textbf{rcond} – double * \hspace{1cm} \textit{Output}

\textit{On exit:} an estimate of the reciprocal of the condition number of \textit{A}. \textbf{rcond} is set to zero if exact singularity is detected or the estimate underflows. If \textbf{rcond} is less than \textit{machine precision}, \textit{A} is singular to working precision.

9: \textbf{fail} – NagError * \hspace{1cm} \textit{Output}

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

\textbf{NE_INT}

On entry, \textbf{n} = \langle \text{value} \rangle.

\textit{Constraint:} \textbf{n} \geq 0.

On entry, \textbf{pda} = \langle \text{value} \rangle.

\textit{Constraint:} \textbf{pda} > 0.

\textbf{NE_INT_2}

On entry, \textbf{pda} = \langle \text{value} \rangle, \textbf{n} = \langle \text{value} \rangle.

\textit{Constraint:} \textbf{pda} \geq \textit{max}(1, \textbf{n}).

\textbf{NE_REAL}

On entry, \textbf{anorm} = \langle \text{value} \rangle.

\textit{Constraint:} \textbf{anorm} \geq 0.0.
NE_ALLOC_FAIL

Memory allocation failed.

NE_BAD_PARAM

On entry, parameter (value) had an illegal value.

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 consult NAG for assistance.

7 Accuracy

The computed estimate rcond is never less than the true value ρ, and in practice is nearly always less than 10ρ, although examples can be constructed where rcond is much larger.

8 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 2n² floating-point operations but takes considerably longer than a call to nag_dsytr (f07mec) with 1 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.

9 Example

To estimate the condition number in the 1-norm (or infinity-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_dsytr (f07mdc). The true condition number in the 1-norm is 75.68.

9.1 Program Text

```c
/* nag_dsycon (f07mgc) Example Program. 
 * Copyright 2001 Numerical Algorithms Group. 
 * Mark 7, 2001. */

#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_OrderType order;
```
/* Arrays */
char uplo[2];
Integer *ipiv=0;
double *a=0;
Nag_UploType uplo_enum;

#ifdef NAG_COLUMN_MAJOR
#define A(I,J) a[(J-1)*pda + I - 1]
order = Nag_ColMajor;
#else
#define A(I,J) a[(I-1)*pda + J - 1]
order = Nag_RowMajor;
#endif
INIT_FAIL(fail);
Vprintf("f07mgc Example Program Results\n\n");

/* Skip heading in data file */
Vscanf("%*[^\n] ");
Vscanf("%ld%*[^\n] ", &n);
#ifdef NAG_COLUMN_MAJOR
pda = n;
#else
pda = n;
#endif

/* Allocate memory */
if ( !(ipiv = NAG_ALLOC(n, Integer)) ||
    !(a = NAG_ALLOC(n * n, double)) )
{
    Vprintf("Allocation failure\n");
    exit_status = -1;
    goto END;
}
/* Read A from data file */
Vscanf(" %1s '%\n" ", uplo);
if (*(unsigned char *)uplo == 'L')
    uplo_enum = Nag_Lower;
else if (*(unsigned char *)uplo == 'U')
    uplo_enum = Nag_Upper;
else
{
    Vprintf("Unrecognised character for Nag_UploType type\n");
    exit_status = -1;
    goto END;
}
if (uplo_enum == Nag_Upper)
{
    for (i = 1; i <= n; ++i)
    {
        for (j = i; j <= n; ++j)
            Vscanf("%lf", &A(i,j));
        Vscanf("%*[^\n] ");
    }
else
{
    for (i = 1; i <= n; ++i)
    {
        for (j = 1; j <= i; ++j)
            Vscanf("%lf", &A(i,j));
        Vscanf("%*[^\n] ");
    }
}
/* Compute norm of A */
f16rcc(order, Nag_OneNorm, uplo_enum, n, a, pda, &anorm, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f16rcc.\n\", fail.message);
}
exit_status = 1;
goto END;
} /* Factorize A */
f07mdc(order, uplo_enum, n, a, pda, ipiv, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07mdc.\n%s\n", fail.message);
    exit_status = 1;
goto END;
}
/* Estimate condition number */
f07mgc(order, uplo_enum, n, a, pda, ipiv, anorm, &rcond, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07mgc.\n%s\n", fail.message);
    exit_status = 1;
goto END;
}
if (rcond >= X02AJC)
    Vprintf("Estimate of condition number = %10.2e\n", 1.0/rcond);
else
    Vprintf("A is singular to working precision\n");
END:
if (ipiv) NAG_FREE(ipiv);
if (a) NAG_FREE(a);
return exit_status;

9.2 Program Data
f07mgc Example Program Data

4 :Value of N
'L' :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

9.3 Program Results
f07mgc Example Program Results

Estimate of condition number = 7.57e+01