NAG C Library Function Document

nag_dppcon (f07ggc)

1 Purpose

nag_dppcon (f07ggc) estimates the condition number of a real symmetric positive-definite matrix $A$, where $A$ has been factorized by nag_dpptrf (f07gdc), using packed storage.

2 Specification

void nag_dppcon (Nag_OrderType order, Nag_UploType uplo, Integer n,
    const double ap[], double anorm, double *rcond, NagError *fail)

3 Description

nag_dppcon (f07ggc) estimates the condition number (in the 1-norm) of a real symmetric positive-definite 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_dsp_norm (f16rdc) to compute $\|A\|_1$ and a call to nag_dpptrf (f07gdc) to compute the Cholesky 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

   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

   On entry: indicates whether $A$ has been factorized as $U^T U$ or $L L^T$ as follows:

   if uplo = Nag_Upper, $A = U^T U$, where $U$ is upper triangular;

   if uplo = Nag_Lower, $A = L L^T$, where $L$ is lower triangular.

   Constraint: uplo = Nag_Upper or Nag_Lower.

3:  n – Integer

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

   Constraint: $n \geq 0$. 
4: \( \text{ap} \left[ \text{dim} \right] \) – const double

**Input**

**Note:** the dimension, \( \text{dim} \), of the array \( \text{ap} \) must be at least \( \max(1, n \times (n + 1)/2) \).

**On entry:** the Cholesky factor of \( A \) stored in packed form, as returned by \text{nag_dpptrf} (f07gdc).

5: \( \text{anorm} \) – double

**Input**

**On entry:** the 1-norm of the original matrix \( A \), which may be computed by calling \text{nag_dsp_norm} (f16rdc). \( \text{anorm} \) must be computed either before calling \text{nag_dpptrf} (f07gdc) or else from a copy of the original matrix \( A \).

**Constraint:** \( \text{anorm} \geq 0.0 \).

6: \( \text{rcond} \) – double *

**Output**

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

7: \( \text{fail} \) – NagError *

**Output**

The NAG error parameter (see the Essential Introduction).

## 6 Error Indicators and Warnings

**NE_INT**

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

**Constraint:** \( n \geq 0 \).

**NE_REAL**

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

**Constraint:** \( \text{anorm} \geq 0.0 \).

**NE_ALLOC_FAIL**

Memory allocation failed.

**NE_BAD_PARAM**

On entry, parameter \( \langle \text{value} \rangle \) 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 \( \text{rcond} \) 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 Further Comments

A call to \text{nag_dppcon} (f07gge) 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 \text{nag_dpptrs} (f07gec) with 1 right-hand side, because extra care is taken to avoid overflow when \( A \) is approximately singular.

The complex analogue of this function is \text{nag_zppcon} (f07guc).
9 Example

To estimate the condition number in the 1-norm (or infinity-norm) of the matrix \( A \), where
\[
A = \begin{pmatrix}
4.16 & -3.12 & 0.56 & -0.10 \\
-3.12 & 5.03 & -0.83 & 1.18 \\
0.56 & -0.83 & 0.76 & 0.34 \\
-0.10 & 1.18 & 0.34 & 1.18 \\
\end{pmatrix}.
\]

Here \( A \) is symmetric positive-definite, stored in packed form, and must first be factorized by \texttt{nag_dpptrf} (f07gdc). The true condition number in the 1-norm is 97.32.

9.1 Program Text

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

int main(void)
{
    double anorm, rcond;
    Integer ap_len, i, j, n;
    Integer exit_status=0;
    NagError fail;
    Nag_UploType uplo_enum;
    Nag_OrderType order;
    char uplo[2];
    double *ap=0;

    INIT_FAIL(fail);
    Vprintf("f07ggc Example Program Results\n\n");
    /* Skip heading in data file */
    Vscanf(" %d\n ", &n);
    ap_len = n * (n + 1)/2;
    /* Allocate memory */
    if ( !(ap = NAG_ALLOC(ap_len, double)) )
    {
        Vprintf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
    /* Read A from data file */
    Vscanf(" %s %s[\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_UPPER(i,j));
        }  
        Vscanf("%*[\n "]);
    }
else
    {  
        for (i = 1; i <= n; ++i)
        {  
            for (j = 1; j <= i; ++j)
                Vscanf("%lf", &A_LOWER(i,j));
        }  
        Vscanf("%*[\n "]);
    }
/* Compute norm of A */
f16rdc(order, Nag_OneNorm, uplo_enum, n, ap, &anorm, &fail);
if (fail.code != NE_NOERROR)
    {  
        Vprintf("Error from f16rdc.\n\s\n", fail.message);
        exit_status = 1;
        goto END;  
    }
/* Factorize A */
f07gdc(order, uplo_enum, n, ap, &fail);
if (fail.code != NE_NOERROR)
    {  
        Vprintf("Error from f07gdc.\n\s\n", fail.message);
        exit_status = 1;
        goto END;  
    }
/* Estimate condition number */
f07ggc(order, uplo_enum, n, ap, anorm, &rcond, &fail);
if (fail.code != NE_NOERROR)
    {  
        Vprintf("Error from f07ggc.\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 (ap) NAG_FREE(ap);
    return exit_status;
9.2 Program Data

f07ggc Example Program Data

\[
\begin{array}{ccc}
4 & \text{:Value of } N \\
'L' & \text{:Value of UPLO} \\
4.16 \\
-3.12 & 5.03 \\
0.56 & -0.83 & 0.76 \\
-0.10 & 1.18 & 0.34 & 1.18 & \text{:End of matrix } A
\end{array}
\]

9.3 Program Results

f07ggc Example Program Results

Estimate of condition number = 9.73e+01