**NAG C Library Function Document**

**nag_dpocon (f07fgc)**

1 **Purpose**

nag_dpocon (f07fgc) estimates the condition number of a real symmetric positive-definite matrix $A$, where $A$ has been factorized by nag_dpotrf (f07fdc).

2 **Specification**

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

3 **Description**

nag_dpocon (f07fgc) 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_dsy_norm (f16rcc) to compute $\|A\|_1$ and a call to nag_dpotrf (f07fdc) 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  
*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 whether $A$ has been factorized as $U^TU$ or $LL^T$ as follows:

- if `uplo = Nag_Upper`, $A = U^TU$, where $U$ is upper triangular;
- if `uplo = Nag_Lower`, $A = LL^T$, 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:* $n \geq 0$. 

[NP3645/7]
4: \[a[dim]\] – const double \hspace{1cm} \textit{Input}\n
\textbf{Note:} the dimension, \(dim\), of the array \(a\) must be at least \(\max(1, pda \times n)\).

\textit{On entry:} the Cholesky factor of \(A\), as returned by \texttt{nag_dpotrf (f07fdc)}.

5: \[pda\] – Integer \hspace{1cm} \textit{Input}\n
\textit{On entry:} the stride separating row or column elements (depending on the value of \texttt{order}) of the matrix in the array \(a\).

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

6: \[anorm\] – double \hspace{1cm} \textit{Input}\n
\textit{On entry:} the 1-norm of the original matrix \(A\), which may be computed by calling \texttt{nag_dsy_norm (f16rcc)}. \(anorm\) must be computed either \textbf{before} calling \texttt{nag_dpotrf (f07fdc)} or else from a copy of the original matrix \(A\).

\textit{Constraint:} \(anorm \geq 0.0\).

7: \[rcond\] – double * \hspace{1cm} \textit{Output}\n
\textit{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 \textit{machine precision}, \(A\) is singular to working precision.

8: \[fail\] – NagError * \hspace{1cm} \textit{Output}\n
The NAG error parameter (see the Essential Introduction).

6 \ Error Indicators and Warnings

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

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

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

\textit{Constraint:} \(pda > 0\).

\textbf{NE_INT_2}\n
On entry, \(pda = \langle\text{value}\rangle\), \(n = \langle\text{value}\rangle\).

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

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

\textit{Constraint:} \(anorm \geq 0.0\).

\textbf{NE_ALLOC_FAIL}\n
Memory allocation failed.

\textbf{NE_BAD_PARAM}\n
On entry, parameter \(\langle\text{value}\rangle\) had an illegal value.

\textbf{NE_INTERNAL_ERROR}\n
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 $\rho$, and in practice is nearly always less than $10\rho$, although examples can be constructed where $rcond$ is much larger.

8 Further Comments

A call to nag_dpocon (f07fgc) 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_dpotrs (f07fec) 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 nag_zpocon (f07fuc).

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 and must first be factorized by nag_dpotrf (f07fdc). The true condition number in the 1-norm is 97.32.

9.1 Program Text

```c
/* nag_dpocon (f07fgc) Example Program.
 * Copyright 2001 Numerical Algorithms Group.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf16.h>
#include <nagf07.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];
    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("f07fgc Example Program Results\n\n");
```

[NP3645/7]
/* Skip heading in data file */
Vscanf("%*[\n ]");
Vscanf("%ld%*[\n ]", &n);
#ifdef NAG_COLUMN_MAJOR
pda = n;
#else
pda = n;
#endif

/* Allocate memory */
if ( !(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%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Factorize A */
f07fdc(order, uplo_enum, n, a, pda, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07fdc.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Estimate condition number */
f07fgc(order, uplo_enum, n, a, pda, anorm, &rcond, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07fgc.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
if (rcond >= X02AJC)
    Vprintf("Estimate of condition number = %10.2e\n\n", 1.0/rcond);
else
    Vprintf("A is singular to working precision\n");
END:
    if (a) NAG_FREE(a);
    return exit_status;
}

9.2 Program Data

f07fgc Example Program Data

4 : Value of N
'L' : Value of UPLO
4.16
-3.12  5.03
0.56 -0.83  0.76
-0.10  1.18  0.34  1.18 : End of matrix A

9.3 Program Results

f07fgc Example Program Results

Estimate of condition number = 9.73e+01