NAG Library Function Document

nag_dpbcon (f07hgc)

1 Purpose

nag_dpbcon (f07hgc) estimates the condition number of a real symmetric positive definite band matrix $A$, where $A$ has been factorized by nag_dpbtrf (f07hdc).

2 Specification

```c
#include <nag.h>
#include <nagf07.h>
void nag_dpbcon (Nag_OrderType order, Nag_UploType uplo, Integer n,
                 Integer kd, const double ab[], Integer pdab, double anorm,
                 double *rcond, NagError *fail)
```

3 Description

nag_dpbcon (f07hgc) estimates the condition number (in the 1-norm) of a real symmetric positive definite band 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_dsb_norm (f16rec) to compute $\|A\|_1$ and a call to nag_dpbtrf (f07hdc) 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 Arguments

1: order – Nag_OrderType

   *Input*

   On entry: the 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 order = Nag_RowMajor. See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

   Constraint: order = Nag_RowMajor or Nag_ColMajor.

2: uplo – Nag_UploType

   *Input*

   On entry: specifies how $A$ has been factorized.

   uplo = Nag_Upper
   $A = U^T U$, where $U$ is upper triangular.
**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:* \( n \geq 0 \).

4: \( kd \) – Integer

*Input*

*On entry:* \( kd \), the number of superdiagonals or subdiagonals of the matrix \( A \).

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

5: \( ab[dim] \) – const double

*Input*

*Note:* the dimension, \( dim \), of the array \( ab \) must be at least \( \max(1, pdab \times n) \).

*On entry:* the Cholesky factor of \( A \), as returned by nag_dpbtrf (f07hdc).

6: \( pdab \) – Integer

*Input*

*On entry:* the stride separating row or column elements (depending on the value of order) of the matrix in the array \( ab \).

*Constraint:* \( pdab \geq kd + 1 \).

7: \( anorm \) – double

*Input*

*On entry:* the 1-norm of the original matrix \( A \), which may be computed by calling nag_dsb_norm (f16rec) with its argument \( norm = \text{Nag_OneNorm} \). \( anorm \) must be computed either before calling nag_dpbtrf (f07hdc) or else from a copy of the original matrix \( A \).

*Constraint:* \( anorm \geq 0.0 \).

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 \( \langle \text{value} \rangle \) had an illegal value.

**NE_INT**

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

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

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

*Constraint:* \( n \geq 0 \).
On entry, \( \text{pdab} = \langle \text{value} \rangle \).
Constraint: \( \text{pdab} > 0 \).

**NE_INT_2**

On entry, \( \text{pdab} = \langle \text{value} \rangle \) and \( \text{kd} = \langle \text{value} \rangle \).
Constraint: \( \text{pdab} \geq \text{kd} + 1 \).

**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.

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

\( \text{nag_dpbcon (f07hgc)} \) is not threaded by NAG in any implementation.

\( \text{nag_dpbcon (f07hgc)} \) 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 \( \text{nag_dpbcon (f07hgc)} \) 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 \( 4nk \) floating-point operations (assuming \( n \gg k \)) but takes considerably longer than a call to \( \text{nag_dpbtrs (f07hec)} \) with one right-hand side, because extra care is taken to avoid overflow when \( A \) is approximately singular.

The complex analogue of this function is \( \text{nag_zpbcon (f07huc)} \).
10 Example

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

$$
A = \begin{pmatrix}
5.49 & 2.68 & 0.00 & 0.00 \\
2.68 & 5.63 & -2.39 & 0.00 \\
0.00 & -2.39 & 2.60 & -2.22 \\
0.00 & 0.00 & -2.22 & 5.17
\end{pmatrix}.
$$

Here $A$ is symmetric and positive definite, and is treated as a band matrix, which must first be factorized by nag_dpbtrf (f07hdc). The true condition number in the 1-norm is 74.15.

10.1 Program Text

```c
/* nag_dpbcon (f07hgc) Example Program. */
* Copyright 2014 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 */
    Integer i, j, k, kd, n, pdab;
    Integer exit_status = 0;
    double anorm, rcond;
    NagError fail;
    Nag_UploType uplo;
    Nag_OrderType order;
    /* Arrays */
    char nag_enum_arg[40];
    double *ab = 0;

    #ifdef NAG_COLUMN_MAJOR
        #define AB_UPPER(I, J) ab[(J-1)*pdab + k + I - J - 1]
        #define AB_LOWER(I, J) ab[(I-1)*pdab + J - I]
    #else
        #define AB_UPPER(I, J) ab[(I-1)*pdab + J - I]
        #define AB_LOWER(I, J) ab[(J-1)*pdab + k + J - I - 1]
    #endif

    INIT_FAIL(fail);
    printf("nag_dpbcon (f07hgc) Example Program Results\n\n");

    /* Skip heading in data file */
    #ifdef _WIN32
        scanf_s("%*[\n] ");
    #else
        scanf("%*[\n] ");
    #endif
    #ifdef _WIN32
        scanf_s("%NAG_IFMT"%NAG_IFMT"%NAG_IFMT"%*[\n] ", &n, &kd);
    #else
        scanf("%NAG_IFMT"%NAG_IFMT"%NAG_IFMT"%*[\n] ", &n, &kd);
    #endif
    pdab = kd + 1;
    /* Allocate memory */
    ```
if (!(ab = NAG_ALLOC((kd+1) * n, double)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

/* Read A from data file */
#ifdef _WIN32
    scanf_s(" %39s%[\n] ", nag_enum_arg, _countof(nag_enum_arg));
#else
    scanf(" %39s%[\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);

k = kd + 1;
if (uplo == Nag_Upper)
{
    for (i = 1; i <= n; ++i)
    {
        for (j = i; j <= MIN(i+kd, n); ++j)
        {
            #ifdef _WIN32
                scanf_s("%lf", &AB_UPPER(i, j));
            #else
                scanf("%lf", &AB_UPPER(i, j));
            #endif
        }
    }
    #ifdef _WIN32
        scanf_s("%*[\n] ");
    #else
        scanf("%*[\n] ");
    #endif
    else
    {
        for (i = 1; i <= n; ++i)
        {
            for (j = MAX(1, i-kd); j <= i; ++j)
            {
                #ifdef _WIN32
                    scanf_s("%lf", &AB_LOWER(i, j));
                #else
                    scanf("%lf", &AB_LOWER(i, j));
                #endif
            }
        }
        #ifdef _WIN32
            scanf_s("%*[\n] ");
        #else
            scanf("%*[\n] ");
        #endif
    }
}

/* Compute norm of A */
/* nag_dsb_norm (f16rec).
* l1-norm, infinity-norm, Frobenius norm, largest absolute
* element, real symmetric band matrix */
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dsb_norm (f16rec).\n%s
", fail.message);
    exit_status = 1;
    goto END;
}

/* Factorize A */
/* nag_dpbttrf (f07hdc).
* Cholesky factorization of real symmetric
* positive-definite band matrix */
nag_dpbttrf(order, uplo, n, kd, ab, pdab, &anorm, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dpbtrf (f07hdc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Estimate condition number */
*/

nag_dpbcon(order, uplo, n, kd, ab, pdab, anorm, &rcond, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dpbcon (f07hgc).\n%s\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\n", 1.0/rcond);
else
    printf("A is singular to working precision\n");

END:
NAG_FREE(ab);
return exit_status;
}

10.2 Program Data
nag_dpbcon (f07hgc) Example Program Data
4 1 :Values of n and kd
Nag_Lower :Value of uplo
5.49 5.63
2.68 -2.39 2.60
-2.22 5.17 :End of matrix A

10.3 Program Results
nag_dpbcon (f07hgc) Example Program Results

Estimate of condition number = 7.42e+01