NAG 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

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

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 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`

$A = U^TU$, 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  
    On entry: n, the order of the matrix A.  
    Constraint: n ≥ 0.

4:  ap[dim] – const double  
    Note: the dimension, dim, of the array ap must be at least max(1,n × (n + 1)/2).  
    On entry: the Cholesky factor of A stored in packed form, as returned by nag_dpptrf (f07gdc).

5:  anorm – double  
    On entry: the 1-norm of the original matrix A, which may be computed by calling nag_dsp_norm (f16rdc) with its argument norm = Nag_OneNorm.  
    anorm must be computed either before calling nag_dpptrf (f07gdc) or else from a copy of the original matrix A.  
    Constraint: anorm ≥ 0.0.

6:  rcond – double *  
    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.

7:  fail – NagError *  
    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.

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, anorm = ⟨value⟩.  
    Constraint: anorm ≥ 0.0.
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 Parallelism and Performance

nag_dppcon (f07ggc) is not threaded by NAG in any implementation.

nag_dppcon (f07ggc) 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_dppcon (f07ggc) 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_dptrfs (f07gec) 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 nag_zppcon (f07guc).

10 Example

This example estimates the condition number in the 1-norm (or $\infty$-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 nag_dpptrf (f07gdc). The true condition number in the 1-norm is 97.32.

10.1 Program Text

/* nag_dppcon (f07ggc) 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 */
    double anorm, rcond;
    Integer ap_len, i, j, n;
    Integer exit_status = 0;
    NagError fail;
    Nag_UploType uplo;
    Nag_OrderType order;
    /* Arrays */
    char nag_enum_arg[40];
double *ap = 0;

#define A_UPPER(I, J) ap[J*(J-1)/2 + I - 1]
#define A_LOWER(I, J) ap[(2*n-J)*(J-1)/2 + J - 1]
#define A_LOWER(I, J) ap[I*(I-1)/2 + J - 1]
#define A_UPPER(I, J) ap[(2*n-I)*(I-1)/2 + J - 1]

void INTialize_A(double *ap, int n); 

void INTialize_A(double *ap, int n) 
{
    for (i = 1; i <= n; ++i)
    {
        for (j = i; j <= n; ++j)
        {
            ap[i*(i-1)/2 + j - 1] = 1.0;
        }
    }
}

int main() 
{
    double *ap = 0;
    int n = 4;
    INTialize_A(ap, n);
    return 0;
}

int INTialize_A(double *ap, int n) 
{
    int i, j;
    for (i = 1; i <= n; ++i)
    {
        for (j = i; j <= n; ++j)
        {
            ap[i*(i-1)/2 + j - 1] = 1.0;
        }
    }
    return 0;
}

int INTialize_A(double *ap, int n) 
{
    int i, j;
    for (i = 1; i <= n; ++i)
    {
        for (j = i; j <= n; ++j)
        {
            ap[i*(i-1)/2 + j - 1] = 1.0;
        }
    }
    return 0;
}
```c
# else
    scanf("%lf", &A_LOWER(i, j));
# endif

#ifndef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif

/* Compute norm of A */
/* nag_dsp_norm (f16rdc).
 * 1-norm, infinity-norm, Frobenius norm, largest absolute
 * element, real symmetric matrix, packed storage
 */
if (fail.code != NE_NOERROR)
    { printf("Error from nag_dsp_norm (f16rdc).
%s
", fail.message);
      exit_status = 1;
      goto END;
    }

/* Factorize A */
/* nag_dpptrf (f07gdc).
 * Cholesky factorization of real symmetric
 * positive-definite matrix, packed storage
 */
if (fail.code != NE_NOERROR)
    { printf("Error from nag_dpptrf (f07gdc).
%s
", fail.message);
      exit_status = 1;
      goto END;
    }

/* Estimate condition number */
/* nag_dpcon (f07ggc).
 * Estimate condition number of real symmetric
 * positive-definite matrix, matrix already factorized by
 * nag_dpptrf (f07gdc), packed storage
 */
if (fail.code != NE_NOERROR)
    { printf("Error from nag_dpcon (f07ggc).
%s
", 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(ap);

return exit_status;
```
10.2 Program Data

nag_dppcon (f07gcc) Example Program Data

Nag_Lower : Value of uplo
Nag_Lower : 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

10.3 Program Results

nag_dppcon (f07gcc) Example Program Results

Estimate of condition number =  9.73e+01