NAG Library Function Document

nag_det_real_band_sym (f03bhc)

1 Purpose

nag_det_real_band_sym (f03bhc) computes the determinant of a n by n symmetric positive definite banded matrix A that has been stored in band-symmetric storage. nag_dpbtrf (f07hdc) must be called first to supply the Cholesky factorized form. The storage (upper or lower triangular) used by nag_dpbtrf (f07hdc) is relevant as this determines which elements of the stored factorized form are referenced.

2 Specification

```
#include <nag.h>
#include <nagf03.h>
void nag_det_real_band_sym (Nag_OrderType order, Nag_UploType uplo,
     Integer n, Integer kd, const double ab[], Integer pdab, double *d,
     Integer *id, NagError *fail)
```

3 Description

The determinant of A is calculated using the Cholesky factorization $A = U^{\mathrm{T}}U$, where U is an upper triangular band matrix, or $A = LL^{\mathrm{T}}$, where L is a lower triangular band matrix. The determinant of A is the product of the squares of the diagonal elements of U or L.

4 References

Wilkinson J H and Reinsch C (1971) *Handbook for Automatic Computation II, Linear Algebra* Springer–Verlag

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 2.3.1.3 in How to Use the NAG Library and its Documentation 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: indicates whether the upper or lower triangular part of A was stored and how it was factorized. This should not be altered following a call to nag_dpbtrf (f07hdc).

uplo = Nag_Upper
The upper triangular part of A was originally stored and A was factorized as $U^{\mathrm{T}}U$ where U is upper triangular.

uplo = Nag_Lower
The lower triangular part of A was originally stored and A was factorized as 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: $\mathbf{n} > 0$.

4: **kd** – Integer *Input*

On entry: k_d, the number of superdiagonals or subdiagonals of the matrix A.

Constraint: $\mathbf{kd} \geq 0$.

5: **ab**[dim] – const double *Input*

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

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: $\mathbf{pdab} \geq \mathbf{kd} + 1$.

7: **d** – double * *Output*

8: **id** – Integer * *Output*

On exit: the determinant of A is given by $\mathbf{d} \times 2.0^{\mathbf{id}}$. It is given in this form to avoid overflow or underflow.

9: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 3.2.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_INT

On entry, $\mathbf{kd} = \langle value \rangle$.
Constraint: $\mathbf{kd} \geq 0$.

On entry, $\mathbf{n} = \langle value \rangle$.
Constraint: $\mathbf{n} > 0$.

NE_INT_2

On entry, $\mathbf{pdab} = \langle value \rangle$ and $\mathbf{kd} = \langle value \rangle$.
Constraint: $\mathbf{pdab} \geq \mathbf{kd} + 1$.

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 How to Use the NAG Library and its Documentation for further information.

NE_MAT_NOT_POS_DEF

The matrix A is not positive definite.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

The accuracy of the determinant depends on the conditioning of the original matrix. For a detailed error analysis see page 54 of Wilkinson and Reinsch (1971).

8 Parallelism and Performance

nag_det_real_band_sym (f03bhc) is not threaded in any implementation.

9 Further Comments

The time taken by nag_det_real_band_sym (f03bhc) is approximately proportional to n.

This function should only be used when $m \ll n$ since as m approaches n, it becomes less efficient to take advantage of the band form.

10 Example

This example calculates the determinant of the real symmetric positive definite band matrix

$$\begin{pmatrix} 5 & -4 & 1 & & & & \\ -4 & 6 & -4 & 1 & & & \\ 1 & -4 & 6 & -4 & 1 & & \\ & 1 & -4 & 6 & -4 & 1 & \\ & & 1 & -4 & 6 & -4 & 1 \\ & & & 1 & -4 & 6 & -4 \\ & & & & 1 & -4 & 5 \end{pmatrix}.$$

10.1 Program Text

```
/* nag_det_real_band_sym (f03bhc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf03.h>
#include <nagf07.h>
#include <nagx04.h>

int main(void)
{
  /* Scalars */
  Integer exit_status = 0;
  Integer i, id, j, kd, kl, ku, k, n, pdab;
  double d;
  /* Arrays */
```

```
  char nag_enum_arg[40];
  double *ab = 0;
  /* NAG types */
  NagError fail;
  Nag_UploType uplo;
  Nag_OrderType order;

  printf("nag_det_real_band_sym (f03bhc) 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 "%*[^\n]", &n, &kd);
#else
  scanf("%" NAG_IFMT "%" NAG_IFMT "%*[^\n]", &n, &kd);
#endif
  k = kd + 1;
  pdab = k;

  if (!(ab = NAG_ALLOC(k * n, double)))
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }

#ifdef _WIN32
  scanf_s("%39s %*[^\n] ", nag_enum_arg, (unsigned)_countof(nag_enum_arg));
#else
  scanf("%39s %*[^\n] ", nag_enum_arg);
#endif
  uplo = (Nag_UploType) nag_enum_name_to_value(nag_enum_arg);

  /* Define matrix element A_ij in terms of elements of array ab[] */
#ifdef NAG_COLUMN_MAJOR
#define AB_UPPER(I, J) ab[(J-1)*pdab + k + I - J - 1]
#define AB_LOWER(I, J) ab[(J-1)*pdab + I - J]
  order = Nag_ColMajor;
#else
#define AB_UPPER(I, J) ab[(I-1)*pdab + J - I]
#define AB_LOWER(I, J) ab[(I-1)*pdab + k + J - I - 1]
  order = Nag_RowMajor;
#endif
  if (uplo == Nag_Upper) {
    /* Read in upper triangular banded matrix  */
    ku = kd;
    kl = 0;
    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 if (uplo == Nag_Lower) {
    /* Read in lower triangular banded matrix  */
    ku = 0;
    kl = kd;
    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
  }
  else {
    printf("Illegal value read for uplo\n");
    exit_status = -4;
    goto END;
  }

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

  /* nag_band_real_mat_print (x04cec)
   * Print real packed banded matrix (easy-to-use)
   */
  fflush(stdout);
  nag_band_real_mat_print(order, n, n, kl, ku, ab, pdab,
                          "Array ab after factorization", NULL, &fail);
  if (fail.code != NE_NOERROR) {
    printf("%s\n", fail.message);
    exit_status = 2;
    goto END;
  }

  /* nag_det_real_band_sym (f03bhc)
   * Determinant of real symmetric positive definite banded matrix
   */
  nag_det_real_band_sym(order, uplo, n, kd, ab, pdab, &d, &id, &fail);
  if (fail.code != NE_NOERROR) {
    printf("%s\n", fail.message);
    exit_status = 3;
    goto END;
  }

  printf("\nd = %12.5f  id = %10" NAG_IFMT "\n", d, id);
  printf("Value of determinant = %13.5e\n", d * pow(2.0, id));

END:
  NAG_FREE(ab);

  return exit_status;
}
```

10.2 Program Data

```
nag_det_real_band_sym (f03bhc) Example Program Data
 7  2                          : n, kd
 Nag_Lower                     : uplo
  5
 -4   6
  1  -4   6
      1  -4   6
          1  -4   6
              1  -4   6
                  1  -4   5   : ab
```

10.3 Program Results

```
nag_det_real_band_sym (f03bhc) Example Program Results

 Array ab after factorization
            1          2          3          4          5          6          7
 1      2.2361
 2     -1.7889     1.6733
 3      0.4472    -1.9124     1.4639
 4                 0.5976    -1.9518     1.3540
 5                           0.6831    -1.9695     1.2863
 6                                     0.7385    -1.9789     1.2403
 7                                               0.7774    -1.9846     0.6761

d =      0.25000  id =          8
Value of determinant =   6.40000e+01
```