NAG Library Function Document
nag_dpbtrs (f07hec)

1 Purpose
nag_dpbtrs (f07hec) solves a real symmetric positive definite band system of linear equations with multiple right-hand sides,

\[ AX = B, \]

where \( A \) has been factorized by nag_dpbtrf (f07hdc).

2 Specification

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

void nag_dpbtrs (Nag_OrderType order, Nag_UploType uplo, Integer n, 
    Integer kd, Integer nrhs, const double ab[], Integer pdab, double b[], 
    Integer pdb, NagError *fail)
```

3 Description

nag_dpbtrs (f07hec) is used to solve a real symmetric positive definite band system of linear equations \( AX = B \), the function must be preceded by a call to nag_dpbtrf (f07hdc) which computes the Cholesky factorization of \( A \). The solution \( X \) is computed by forward and backward substitution.

If \( \text{uplo} = \text{Nag_Upper} \), \( A = U^T U \), where \( U \) is upper triangular; the solution \( X \) is computed by solving \( U^T Y = B \) and then \( UX = Y \).

If \( \text{uplo} = \text{Nag_Lower} \), \( A = LL^T \), where \( L \) is lower triangular; the solution \( X \) is computed by solving \( LY = B \) and then \( L^T X = Y \).

4 References


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 = LLᵀ, 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:  kd – Integer  
    On entry: kd, the number of superdiagonals or subdiagonals of the matrix A.
    Constraint: kd ≥ 0.

5:  nrhs – Integer  
    On entry: nrhs, the number of right-hand sides.
    Constraint: nrhs ≥ 0.

6:  ab[dim] – const double  
    Note: the dimension, dim, of the array ab must be at least max(1, pdab × n).
    On entry: the Cholesky factor of A, as returned by nag_dpbtrf (f07hdc).

7:  pdab – Integer  
    On entry: the stride separating row or column elements (depending on the value of order) of the
    matrix in the array ab.
    Constraint: pdab ≥ kd + 1.

8:  b[dim] – double  
    Note: the dimension, dim, of the array b must be at least
    max(1, pdb × nrhs) when order = Nag_ColMajor;
    max(1, n × pdb) when order = Nag_RowMajor.

The (i, j)th element of the matrix B is stored in
    b[(j - 1) × pdb + i - 1] when order = Nag_ColMajor;
    b[(i - 1) × pdb + j - 1] when order = Nag_RowMajor.

    On entry: the n by r right-hand side matrix B.
    On exit: the n by r solution matrix X.

9:  pdb – Integer  
    On entry: the stride separating row or column elements (depending on the value of order) in the
    array b.
    Constraints:
    if order = Nag_ColMajor, pdb ≥ max(1, n);
    if order = Nag_RowMajor, pdb ≥ max(1, nrhs).

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

**NE_INT_2**
On entry, \( pdab = \langle \text{value} \rangle \) and \( kd = \langle \text{value} \rangle \).
Constraint: \( pdab \geq kd + 1 \).
On entry, \( pdb = \langle \text{value} \rangle \) and \( n = \langle \text{value} \rangle \).
Constraint: \( pdb \geq \max(1,n) \).
On entry, \( pdb = \langle \text{value} \rangle \) and \( nrhs = \langle \text{value} \rangle \).
Constraint: \( pdb \geq \max(1,nrhs) \).

**NE_INTERNAL_ERROR**
An internal error has occurred in this function. Check the function call and any array sizes. If the
 caller 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.

7 Accuracy

For each right-hand side vector \( b \), the computed solution \( x \) is the exact solution of a perturbed system of
 equations \( (A + E)x = b \), where

- if \( \text{uplo} = \text{Nag_Upper} \), \( |E| \leq c(k + 1)\epsilon U^T ||U|; \)
- if \( \text{uplo} = \text{Nag_Lower} \), \( |E| \leq c(k + 1)\epsilon L^T ||L|^T \),

\( c(k + 1) \) is a modest linear function of \( k + 1 \), and \( \epsilon \) is the **machine precision**.
If \( \hat{x} \) is the true solution, then the computed solution \( x \) satisfies a forward error bound of the form

\[
\frac{\|x - \hat{x}\|_\infty}{\|x\|_\infty} \leq c(k + 1) \text{cond}(A, x) \epsilon
\]

where \( \text{cond}(A, x) = \|A^{-1}\|_\infty \|A\|_\infty / \|x\|_\infty \leq \text{cond}(A) = \|A^{-1}\|_\infty \|A\|_\infty \leq \kappa_\infty(A) \). Note that \( \text{cond}(A, x) \) can be much smaller than \( \text{cond}(A) \).

Forward and backward error bounds can be computed by calling \text{nag_dpbtrs} (f07hec), and an estimate for \( \kappa_\infty(A) \) (\( = \kappa_1(A) \)) can be obtained by calling \text{nag_dpbcon} (f07hgc).

8 Parallelism and Performance

\text{nag_dpbtrs} (f07hec) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

\text{nag_dpbtrs} (f07hec) 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

The total number of floating-point operations is approximately \( 4nk \), assuming \( n \gg k \).

This function may be followed by a call to \text{nag_dpbtrs} (f07hec) to refine the solution and return an error estimate.

The complex analogue of this function is \text{nag_zpbtrs} (f07hsc).

10 Example

This example solves the system of equations \( AX = B \), where

\[
A = \begin{pmatrix}
5.49 & 2.68 & 0.00 & 0.00 \\
2.68 & 5.63 & -2.39 & 0.00 \\
0.00 & -2.39 & 2.60 & -2.22 \\
0.00 & 0.00 & -2.22 & 5.17
\end{pmatrix}
\quad \text{and} \quad
B = \begin{pmatrix}
22.09 & 5.10 \\
9.31 & 30.81 \\
-5.24 & -25.82 \\
11.83 & 22.90
\end{pmatrix}.
\]

Here \( A \) is symmetric and positive definite, and is treated as a band matrix, which must first be factorized by \text{nag_dpbtrf} (f07hdc).

10.1 Program Text

/* nag_dpbtrs (f07hec) Example Program.
 * * Copyright 2014 Numerical Algorithms Group.
 * * Mark 7, 2001.
 */
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf07.h>
#include <nagx04.h>

int main(void)
{
    /* Scalars */
    Integer i, j, k, kd, n, nrhs, pdab, pdb;
    Integer exit_status = 0;
Nag_UploType uplo;
NagError fail;
Nag_OrderType order;
/* Arrays */
char nag_enum_arg[40];
double *ab = 0, *b = 0;

#ifndef NAG_LOAD_FP
/* The following line is needed to force the Microsoft linker to load floating point support */
float force_loading_of_ms_float_support = 0;
#endif /* NAG_LOAD_FP */

#ifndef 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 + k + J - I - 1]
#define B(I, J) b[(I-1)*pdb + J - 1]
#else
#define AB_UPPER(I, J) ab[(J-1)*pdab + I - J]
#define AB_LOWER(I, J) ab[(I-1)*pdab + k + J - I]
#define B(I, J) b[(I-1)*pdb + J]
#endif

INIT_FAIL(fail);

printf("nag_dpbtrs (f07hec) Example Program Results\n\n");

/* Skip heading in data file */
#ifndef _WIN32
scanf_s("%[\n] ");
#else
scanf("%[\n] ");
#endif
#ifndef _WIN32
scanf("%NAG_IFMT%NAG_IFMT%NAG_IFMT%[\n] ", &n, &kd, &nrhs);
#else
scanf("%NAG_IFMT%NAG_IFMT%NAG_IFMT%[\n] ", &n, &kd, &nrhs);
#endif
pdab = kd + 1;
#endif

/* Allocate memory */
if (!(ab = NAG_ALLOC((kd+1) * n, double)) ||
!(b = NAG_ALLOC(n * nrhs, double)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

/* Read A from data file */
#ifndef _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++)
    {
```c
{ 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
    /* Read B from data file */
    for (i = 1; i <= n; ++i)
    { for (j = 1; j <= nrhs; ++j)
        #ifdef _WIN32
            scanf_s("%lf", &B(i, j));
        #else
            scanf("%lf", &B(i, j));
        #endif
        /* Factorize A */
        /* nag_dpbtrf (f07hd). *
        * Cholesky factorization of real symmetric *
        * positive-definite band matrix *
        */
        nag_dpbtrf(order, uplo, n, kd, ab, pdab, &fail);
        if (fail.code != NE_NOERROR)
        { printf("Error from nag_dpbtrf (f07hd).\n%s\n", fail.message);
            exit_status = 1;
            goto END;
        }
        /* Compute solution */
        /* nag_dpbtrs (f07hec). *
        * Solution of real symmetric positive-definite band system *
        * of linear equations, multiple right-hand sides, matrix *
        * already factorized by nag_dpbtrf (f07hd). *
        */
        nag_dpbtrs(order, uplo, n, kd, nrhs, ab, pdab, b, pdb, &fail);
        if (fail.code != NE_NOERROR)
        { printf("Error from nag_dpbtrs (f07hec).\n%s\n", fail.message);
            exit_status = 1;
            goto END;
```
/* Print solution */
/* nag_gen_real_mat_print (x04cac).
 * Print real general matrix (easy-to-use)
 */
fflush(stdout);
nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, nrhs, 
b, pdb, "Solution(s)", 0, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_gen_real_mat_print (x04cac).\n%s\n", 
fail.message);
    exit_status = 1;
    goto END;
}
END:
NAG_FREE(ab);
NAG_FREE(b);
return exit_status;

10.2 Program Data
nag_dpbtrs (f07hec) Example Program Data

4 1 2 :Values of n, kd and nrhs
Nag_Lower :Value of uplo
5.49
2.68 5.63 -2.39 2.60 -2.22 5.17 :End of matrix A
22.09 5.10
9.31 30.81
-5.24 -25.82
11.83 22.90 :End of matrix B

10.3 Program Results
nag_dpbtrs (f07hec) Example Program Results

Solution(s)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.0000</td>
<td>-2.0000</td>
</tr>
<tr>
<td>2</td>
<td>-2.0000</td>
<td>6.0000</td>
</tr>
<tr>
<td>3</td>
<td>-3.0000</td>
<td>-1.0000</td>
</tr>
<tr>
<td>4</td>
<td>1.0000</td>
<td>4.0000</td>
</tr>
</tbody>
</table>