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
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

To solve a real symmetric positive-definite band system of linear equations \( AX = B \), this 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 \( LT X = Y \).

4 References


5 Parameters

1: \( \text{order} \) – Nag_OrderType

*Input*

On entry: the \( \text{order} \) 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 \( \text{order} = \text{Nag_RowMajor} \). See Section 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.

Constraint: \( \text{order} = \text{Nag_RowMajor} \) or \( \text{Nag_ColMajor} \).

2: \( \text{uplo} \) – Nag_UploType

*Input*

On entry: indicates whether \( A \) has been factorized as \( U^T U \) or \( LL^T \) as follows:

- if \( \text{uplo} = \text{Nag_Upper} \), \( A = U^T U \), where \( U \) is upper triangular;
- if \( \text{uplo} = \text{Nag_Lower} \), \( A = LL^T \), where \( L \) is lower triangular.

Constraint: \( \text{uplo} = \text{Nag_Upper} \) or \( \text{Nag_Lower} \).

3: \( n \) – Integer

*Input*

On entry: \( n \), the order of the matrix \( A \).

Constraint: \( n \geq 0 \).
4: \( \text{kd} \) – Integer \hspace{1em} \textit{Input}

\textit{On entry:} \( k \), the number of super-diagonals or sub-diagonals of the matrix \( A \).
\textit{Constraint:} \( \text{kd} \geq 0 \).

5: \( \text{nrhs} \) – Integer \hspace{1em} \textit{Input}

\textit{On entry:} \( r \), the number of right-hand sides.
\textit{Constraint:} \( \text{nrhs} \geq 0 \).

6: \( \text{ab}[\text{dim}] \) – const double \hspace{1em} \textit{Input}

\textit{Note:} the dimension, \( \text{dim} \), of the array \( \text{ab} \) must be at least \( \max(1, \text{pdab} \times \text{n}) \).
\textit{On entry:} the Cholesky factor of \( A \), as returned by \text{nag_dpbtrf} (f07hdc).

7: \( \text{pdab} \) – Integer \hspace{1em} \textit{Input}

\textit{On entry:} the stride separating row or column elements (depending on the value of \text{order}) of the matrix in the array \( \text{ab} \).
\textit{Constraint:} \( \text{pdab} \geq \text{kd} + 1 \).

8: \( \text{b}[\text{dim}] \) – double \hspace{1em} \textit{Input/Output}

\textit{Note:} the dimension, \( \text{dim} \), of the array \( \text{b} \) must be at least \( \max(1, \text{pdb} \times \text{nrhs}) \) when \( \text{order} = \text{Nag_ColMajor} \) and at least \( \max(1, \text{pdb} \times \text{n}) \) when \( \text{order} = \text{Nag_RowMajor} \).
If \( \text{order} = \text{Nag_ColMajor} \), the \((i,j)\)th element of the matrix \( B \) is stored in \( \text{b}[(j-1) \times \text{pdb} + i - 1] \) and if \( \text{order} = \text{Nag_RowMajor} \), the \((i,j)\)th element of the matrix \( B \) is stored in \( \text{b}[(i-1) \times \text{pdb} + j - 1] \).
\textit{On entry:} the \( n \) by \( r \) right-hand side matrix \( B \).
\textit{On exit:} the \( n \) by \( r \) solution matrix \( X \).

9: \( \text{pdb} \) – Integer \hspace{1em} \textit{Input}

\textit{On entry:} the stride separating matrix row or column elements (depending on the value of \text{order}) in the array \( \text{b} \).
\textit{Constraints:}
\begin{align*}
\text{if } \text{order} = \text{Nag_ColMajor}, & \quad \text{pdb} \geq \max(1, \text{n}) ; \\
\text{if } \text{order} = \text{Nag_RowMajor}, & \quad \text{pdb} \geq \max(1, \text{nrhs}) .
\end{align*}

10: \( \text{fail} \) – NagError * \hspace{1em} \textit{Output}

The NAG error parameter (see the Essential Introduction).

6 \quad \textbf{Error Indicators and Warnings}

\textbf{NE_INT}
\textit{On entry,} \( \text{n} = \langle \text{value} \rangle \).
\textit{Constraint:} \( \text{n} \geq 0 \).
\textit{On entry,} \( \text{kd} = \langle \text{value} \rangle \).
\textit{Constraint:} \( \text{kd} \geq 0 \).
\textit{On entry,} \( \text{nrhs} = \langle \text{value} \rangle \).
\textit{Constraint:} \( \text{nrhs} \geq 0 \).
\textit{On entry,} \( \text{pdab} = \langle \text{value} \rangle \).
\textit{Constraint:} \( \text{pdab} > 0 \).
\textit{On entry,} \( \text{pdb} = \langle \text{value} \rangle \).
\textit{Constraint:} \( \text{pdb} > 0 \).
On entry, `pdab = ⟨value⟩`, `kd = ⟨value⟩`.  
Constraint: `pdab >= kd + 1`.

On entry, `pdb = ⟨value⟩`, `n = ⟨value⟩`.  
Constraint: `pdb >= max(1, n)`.

On entry, `pdb = ⟨value⟩`, `nrhs = ⟨value⟩`.  
Constraint: `pdb >= max(1, nrhs)`.

**NE_ALLOC_FAIL**
Memory allocation failed.

**NE_BAD_PARAM**
On entry, parameter ⟨value⟩ had an illegal value.

**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 consult NAG for assistance.

### 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 `uplo = Nag_Upper`, `|E| <= c(k+1)epsilon|U^T||U|`;
- if `uplo = Nag_Lower`, `|E| <= c(k+1)epsilon|L||L^T|`,

where `c(k+1)` is a modest linear function of `k+1`, and `epsilon` is the *machine precision*.

If `xtilde` is the true solution, then the computed solution `x` satisfies a forward error bound of the form

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

where `\text{cond}(A, x) = ||A^{-1}|| |A||x||_\infty / ||x||_\infty \leq \text{cond}(A) = ||A^{-1}|| |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 `nag_dpbrfs (f07hhc)`, and an estimate for `\kappa_\infty(A)` (\(= \kappa_1(A)\)) can be obtained by calling `nag_dpbccon (f07hgc)`.

### 8 Further Comments

The total number of floating-point operations is approximately `4nk` assuming `n >> k`.

This function may be followed by a call to `nag_dpbrfs (f07hhc)` to refine the solution and return an error estimate.

The complex analogue of this function is `nag_zpbtrs (f07hsc)`.

### 9 Example

To solve 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 `nag_dpbftrf (f07hdc)`.
9.1 Program Text
/* nag_dpbtrs (f07hec) Example Program. */
/* Copyright 2001 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_enum;
    NagError fail;
    Nag_OrderType order;
    /* Arrays */
    char uplo[2];
    double *ab=0, *b=0;
    #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]
    #define B(I,J) b[(J-1)*pdb + I - 1]
    #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]
    #define B(I,J) b[(I-1)*pdb + J - 1]
    #endif
    INIT_FAIL(fail);
    Vprintf("f07hec Example Program Results\n\n");
    /* Skip heading in data file */
    Vscanf("%*[^
] ");
    Vscanf("%ld%ld%ld%*[^
] ", &n, &kd, &nrhs);
    pdab = kd + 1;
    #ifdef NAG_COLUMN_MAJOR
    pdb = n;
    #else
    pdb = nrhs;
    #endif
    /* Allocate memory */
    if ( !(ab = NAG_ALLOC((kd+1) * n, double)) ||
        !(b = NAG_ALLOC(n * nrhs, double)) )
    {
        Vprintf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
    /* Read A from data file */
    Vscanf(" ' %1s '%*[^
] " , 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");
k = kd + 1;
if (uplo_enum == Nag_Upper)
{
  for (i = 1; i <= n; ++i)
  {
    for (j = i; j <= MIN(i+kd,n); ++j)
    {
      Vscanf("%lf", &AB_UPPER(i,j));
    }
    Vscanf("%*['\n]");
  }
}
else
{
  for (i = 1; i <= n; ++i)
  {
    for (j = MAX(1,i-kd); j <= i; ++j)
    {
      Vscanf("%lf", &AB_LOWER(i,j));
    }
    Vscanf("%*['\n]");
  }
}
/* Read B from data file */
for (i = 1; i <= n; ++i)
{
  for (j = 1; j <= nrhs; ++j)
  {
    Vscanf("%lf", &B(i,j));
  }
  Vscanf("%*['\n]");
}
/* Factorize A */
f07hdc(order, uplo_enum, n, kd, ab, pdab, &fail);
if (fail.code != NE_NOERROR)
{
  Vprintf("Error from f07hdc.\n%s\n", fail.message);
  exit_status = 1;
  goto END;
}
/* Compute solution */
f07hec(order, uplo_enum, n, kd, nrhs, ab, pdab, b, pdb, &fail);
if (fail.code != NE_NOERROR)
{
  Vprintf("Error from f07hec.\n%s\n", fail.message);
  exit_status = 1;
  goto END;
}
/* Print solution */
x04cac(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, nrhs, b, pdb, "Solution(s)", 0, &fail);
if (fail.code != NE_NOERROR)
{
  Vprintf("Error from x04cac.\n%s\n", fail.message);
  exit_status = 1;
  goto END;
}
END:
if (ab) NAG_FREE(ab);
if (b) NAG_FREE(b);
return exit_status;

9.2 Program Data

f07hec Example Program Data

4 1 2 :Values of N, KD and NRHS
'L' :Value of UPLO
5.49
2.68 5.63
-2.22 5.17 :End of matrix A

[NP3645/7]
22.09  5.10
  9.31  30.81
 -5.24 -25.82
11.83  22.90  :End of matrix B

9.3  Program Results

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>