NAG Library Function Document

nag_dtbtrs (f07vec)

1 Purpose

nag_dtbtrs (f07vec) solves a real triangular band system of linear equations with multiple right-hand sides, $AX = B$ or $A^{\mathrm{T}}X = B$.

2 Specification

```
#include <nag.h>
#include <nagf07.h>
void nag_dtbtrs (Nag_OrderType order, Nag_UploType uplo,
     Nag_TransType trans, Nag_DiagType diag, Integer n, Integer kd,
     Integer nrhs, const double ab[], Integer pdab, double b[], Integer pdb,
     NagError *fail)
```

3 Description

nag_dtbtrs (f07vec) solves a real triangular band system of linear equations $AX = B$ or $A^{\mathrm{T}}X = B$.

4 References

Golub G H and Van Loan C F (1996) *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

Higham N J (1989) The accuracy of solutions to triangular systems *SIAM J. Numer. Anal.* **26** 1252–1265

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 whether A is upper or lower triangular.

uplo = Nag_Upper
A is upper triangular.

uplo = Nag_Lower
A is lower triangular.

Constraint: **uplo** = Nag_Upper or Nag_Lower.

3: **trans** – Nag_TransType *Input*

On entry: indicates the form of the equations.

trans = Nag_NoTrans
The equations are of the form $AX = B$.

trans = Nag_Trans or Nag_ConjTrans
The equations are of the form $A^{\mathrm{T}}X = B$.

Constraint: **trans** = Nag_NoTrans, Nag_Trans or Nag_ConjTrans.

4: **diag** – Nag_DiagType *Input*

On entry: indicates whether A is a nonunit or unit triangular matrix.

diag = Nag_NonUnitDiag
A is a nonunit triangular matrix.

diag = Nag_UnitDiag
A is a unit triangular matrix; the diagonal elements are not referenced and are assumed to be 1.

Constraint: **diag** = Nag_NonUnitDiag or Nag_UnitDiag.

5: **n** – Integer *Input*

On entry: n, the order of the matrix A.

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

6: **kd** – Integer *Input*

On entry: k_d, the number of superdiagonals of the matrix A if **uplo** = Nag_Upper, or the number of subdiagonals if **uplo** = Nag_Lower.

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

7: **nrhs** – Integer *Input*

On entry: r, the number of right-hand sides.

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

8: **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 n by n triangular band matrix A.

This is stored as a notional two-dimensional array with row elements or column elements stored contiguously. The storage of elements of A_{ij}, depends on the **order** and **uplo** arguments as follows:

if **order** = Nag_ColMajor and **uplo** = Nag_Upper,
A_{ij} is stored in **ab**$[k_d + i - j + (j-1) \times \mathbf{pdab}]$, for $j = 1, \ldots, n$ and $i = \max(1, j - k_d), \ldots, j$;
if **order** = Nag_ColMajor and **uplo** = Nag_Lower,
A_{ij} is stored in **ab**$[i - j + (j-1) \times \mathbf{pdab}]$, for $j = 1, \ldots, n$ and $i = j, \ldots, \min(n, j + k_d)$;
if **order** = Nag_RowMajor and **uplo** = Nag_Upper,
A_{ij} is stored in **ab**$[j - i + (i-1) \times \mathbf{pdab}]$, for $i = 1, \ldots, n$ and $j = i, \ldots, \min(n, i + k_d)$;
if **order** = Nag_RowMajor and **uplo** = Nag_Lower,
A_{ij} is stored in **ab**$[k_d + j - i + (i-1) \times \mathbf{pdab}]$, for $i = 1, \ldots, n$ and $j = \max(1, i - k_d), \ldots, i$.

If **diag** = Nag_UnitDiag, the diagonal elements of AB are assumed to be 1, and are not referenced.

9: **pdab** – Integer *Input*

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

Constraint: **pdab** $\geq$ **kd** $+ 1$.

10: **b**$[dim]$ – double *Input/Output*

Note: the dimension, *dim*, of the array **b** must be at least

max$(1, \mathbf{pdb} \times \mathbf{nrhs})$ when **order** = Nag_ColMajor;
max$(1, \mathbf{n} \times \mathbf{pdb})$ when **order** = Nag_RowMajor.

The (i, j)th element of the matrix B is stored in

b$[(j-1) \times \mathbf{pdb} + i - 1]$ when **order** = Nag_ColMajor;
b$[(i-1) \times \mathbf{pdb} + j - 1]$ when **order** = Nag_RowMajor.

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

On exit: the n by r solution matrix X.

11: **pdb** – Integer *Input*

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** $\geq$ max$(1, \mathbf{n})$;
if **order** = Nag_RowMajor, **pdb** $\geq$ max$(1, \mathbf{nrhs})$.

12: **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 ⟨*value*⟩ had an illegal value.

NE_INT

On entry, **kd** = ⟨*value*⟩.
Constraint: **kd** ≥ 0.

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 0.

On entry, **nrhs** = ⟨*value*⟩.
Constraint: **nrhs** ≥ 0.

On entry, **pdab** = ⟨*value*⟩.
Constraint: **pdab** > 0.

On entry, **pdb** = ⟨*value*⟩.
Constraint: **pdb** > 0.

NE_INT_2

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

On entry, **pdb** $= \langle value \rangle$ and **n** $= \langle value \rangle$.
Constraint: **pdb** $\geq \max(1, \mathbf{n})$.

On entry, **pdb** $= \langle value \rangle$ and **nrhs** $= \langle value \rangle$.
Constraint: **pdb** $\geq \max(1, \mathbf{nrhs})$.

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_SINGULAR

Element $\langle value \rangle$ of the diagonal is exactly zero. A is singular and the solution has not been computed.

7 Accuracy

The solutions of triangular systems of equations are usually computed to high accuracy. See Higham (1989).

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

$$|E| \leq c(k)\epsilon|A|,$$

$c(k)$ is a modest linear function of k, and ϵ is the ***machine precision***.

If $\hat{x}$ is the true solution, then the computed solution x satisfies a forward error bound of the form

$$\frac{\|x - \hat{x}\|_\infty}{\|x\|_\infty} \leq c(k)\,\mathrm{cond}(A, x)\epsilon, \qquad \text{provided} \qquad c(k)\,\mathrm{cond}(A, x)\epsilon < 1,$$

where $\mathrm{cond}(A, x) = \left\| |A^{-1}||A||x| \right\|_\infty / \|x\|_\infty$.

Note that $\mathrm{cond}(A, x) \leq \mathrm{cond}(A) = \left\| |A^{-1}||A| \right\|_\infty \leq \kappa_\infty(A)$; $\mathrm{cond}(A, x)$ can be much smaller than $\mathrm{cond}(A)$ and it is also possible for $\mathrm{cond}(A^{\mathrm{T}})$ to be much larger (or smaller) than $\mathrm{cond}(A)$.

Forward and backward error bounds can be computed by calling nag_dtbrfs (f07vhc), and an estimate for $\kappa_\infty(A)$ can be obtained by calling nag_dtbcon (f07vgc) with **norm** = Nag_InfNorm.

8 Parallelism and Performance

nag_dtbtrs (f07vec) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_dtbtrs (f07vec) 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 $2nkr$ if $k \ll n$.

The complex analogue of this function is nag_ztbtrs (f07vsc).

10 Example

This example solves the system of equations $AX = B$, where

$$A = \begin{pmatrix} -4.16 & 0.00 & 0.00 & 0.00 \\ -2.25 & 4.78 & 0.00 & 0.00 \\ 0.00 & 5.86 & 6.32 & 0.00 \\ 0.00 & 0.00 & -4.82 & 0.16 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} -16.64 & -4.16 \\ -13.78 & -16.59 \\ 13.10 & -4.94 \\ -14.14 & -9.96 \end{pmatrix}.$$

Here A is treated as a lower triangular band matrix with one subdiagonal.

10.1 Program Text

```
/* nag_dtbtrs (f07vec) 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;

#ifdef 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 */

#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]
  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]
#define B(I, J)        b[(I-1)*pdb + J - 1]
  order = Nag_RowMajor;
#endif

  INIT_FAIL(fail);

  printf("nag_dtbtrs (f07vec) 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, &nrhs);
#else
  scanf("%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%*[^\n] ", &n, &kd, &nrhs);
#endif
  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)))
    {
      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
    }
```

```
  /* 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
#ifdef _WIN32
      scanf_s("%*[^\n] ");
#else
      scanf("%*[^\n] ");
#endif
    }

  /* Compute solution */
  /* nag_dtbtrs (f07vec).
   * Solution of real band triangular system of linear
   * equations, multiple right-hand sides
   */
  nag_dtbtrs(order, uplo, Nag_NoTrans, Nag_NonUnitDiag, n,
             kd, nrhs, ab, pdab, b, pdb, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_dtbtrs (f07vec).\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_dtbtrs (f07vec) Example Program Data
  4  1  2                     :Values of n, kd and nrhs
  Nag_Lower                   :Value of uplo
 -4.16
 -2.25   4.78
         5.86   6.32
               -4.82   0.16   :End of matrix A
-16.64  -4.16
-13.78 -16.59
 13.10  -4.94
-14.14  -9.96                 :End of matrix B
```

10.3 Program Results

```
nag_dtbtrs (f07vec) Example Program Results

 Solution(s)
              1          2
 1       4.0000     1.0000
 2      -1.0000    -3.0000
 3       3.0000     2.0000
 4       2.0000    -2.0000
```