

NAG C 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 $A^T X = B$.

2 Specification

```
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 $A^T X = B$.

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 Parameters

- 1: **order** – Nag_OrderType *Input*
On entry: the **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 **order** = **Nag_RowMajor**. See Section 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.
Constraint: **order** = **Nag_RowMajor** or **Nag_ColMajor**.
- 2: **uplo** – Nag_UploType *Input*
On entry: indicates whether A is upper or lower triangular as follows:
 if **uplo** = **Nag_Upper**, A is upper triangular;
 if **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 as follows:
 if **trans** = **Nag_NoTrans**, the equations are of the form $AX = B$;
 if **trans** = **Nag_Trans** or **Nag_ConjTrans**, the equations are of the form $A^T X = B$.
Constraint: **trans** = **Nag_NoTrans**, **Nag_Trans** or **Nag_ConjTrans**.
- 4: **diag** – Nag_DiagType *Input*
On entry: indicates whether A is a non-unit or unit triangular matrix as follows:

if **diag** = **Nag_NonUnitDiag**, A is a non-unit triangular matrix;

if **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: $n \geq 0$.

6: **kd** – Integer *Input*

On entry: k , the number of super-diagonals of the matrix A if **uplo** = **Nag_Upper** or the number of sub-diagonals if **uplo** = **Nag_Lower**.

Constraint: $kd \geq 0$.

7: **nrhs** – Integer *Input*

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

Constraint: **nrhs** ≥ 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 matrix A . This is stored as a notional two-dimensional array with row elements or column elements stored contiguously. The storage of elements a_{ij} depends on the **order** and **uplo** parameters as follows:

if **order** = **Nag_ColMajor** and **uplo** = **Nag_Upper**,
 a_{ij} is stored in **ab**[$k + i - j + (j - 1) \times \mathbf{pdab}$], for $i = 1, \dots, n$ and
 $j = i, \dots, \min(n, i + k)$;

if **order** = **Nag_ColMajor** and **uplo** = **Nag_Lower**,
 a_{ij} is stored in **ab**[$i - j + (j - 1) \times \mathbf{pdab}$], for $i = 1, \dots, n$ and
 $j = \max(1, i - k), \dots, i$;

if **order** = **Nag_RowMajor** and **uplo** = **Nag_Upper**,
 a_{ij} is stored in **ab**[$j - i + (i - 1) \times \mathbf{pdab}$], for $i = 1, \dots, n$ and
 $j = i, \dots, \min(n, i + k)$;

if **order** = **Nag_RowMajor** and **uplo** = **Nag_Lower**,
 a_{ij} is stored in **ab**[$k + j - i + (i - 1) \times \mathbf{pdab}$], for $i = 1, \dots, n$ and
 $j = \max(1, i - k), \dots, i$.

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** $\geq \mathbf{kd} + 1$.

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** and at least $\max(1, \mathbf{pdb} \times \mathbf{n})$ when **order** = **Nag_RowMajor**.

If **order** = **Nag_ColMajor**, the (i, j) th element of the matrix B is stored in **b**[$(j - 1) \times \mathbf{pdb} + i - 1$] and if **order** = **Nag_RowMajor**, the (i, j) th element of the matrix B is stored in **b**[$(i - 1) \times \mathbf{pdb} + j - 1$].

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 matrix 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 * *Output*
 The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, **n** = $\langle value \rangle$.
 Constraint: **n** ≥ 0 .
 On entry, **kd** = $\langle value \rangle$.
 Constraint: **kd** ≥ 0 .
 On entry, **nrhs** = $\langle value \rangle$.
 Constraint: **nrhs** ≥ 0 .
 On entry, **pdab** = $\langle value \rangle$.
 Constraint: **pdab** > 0 .
 On entry, **pdb** = $\langle value \rangle$.
 Constraint: **pdb** > 0 .

NE_INT_2

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

NE_SINGULAR

The matrix *A* is singular.

NE_ALLOC_FAIL

Memory allocation failed.

NE_BAD_PARAM

On entry, parameter $\langle value \rangle$ 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

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

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

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

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

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

8 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).

9 Example

To solve 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 1 sub-diagonal.

9.1 Program Text

```
/* nag_dtbtrs (f07vec) 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]
    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);
    Vprintf("f07vec Example Program Results\n\n");

    /* Skip heading in data file */
    Vscanf("%*[\n] ");
    Vscanf("%ld%ld%ld%*[\n] ", &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 '%*[\n] ", 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");
        exit_status = -1;
        goto END;
    }
    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] ");
    }

```

```

/* Compute solution */
f07vec(order, uplo_enum, Nag_NoTrans, Nag_NonUnitDiag, n,
      kd, nrhs, ab, pdab, b, pdb, &fail);
if (fail.code != NE_NOERROR)
{
  Vprintf("Error from f07vec.\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

f07vec Example Program Data

4	1	2				:Values of N, KD and NRHS
'L'						: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

9.3 Program Results

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
