NAG C Library Function Document

nag_dtrtrs (f07tec)

1 Purpose

nag_dtrtrs (f07tec) solves a real triangular system of linear equations with multiple right-hand sides, \( AX = B \) or \( A^T X = B \).

2 Specification

```c
void nag_dtrtrs (Nag_OrderType order, Nag_UploType uplo, Nag_TransType trans,
                Nag_DiagType diag, Integer n, Integer nrhs,
                const double a[], Integer pda,
                double b[], Integer pdb, NagError *fail)
```

3 Description

nag_dtrtrs (f07tec) solves a real triangular system of linear equations \( AX = B \) or \( A^T X = B \).

4 References


5 Parameters

1: `order` – Nag_OrderType  
   \( \text{Input} \)  
   \( \text{On entry: the order parameter specifies the two-dimensional storage scheme being used, i.e., row-major ordering or column-major ordering. C language defined storage is specified by order = Nag_RowMajor. See Section 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.} \)  
   \( \text{Constraint: order = Nag_RowMajor or Nag_ColMajor.} \)

2: `uplo` – Nag_UploType
   \( \text{Input} \)  
   \( \text{On entry: indicates whether A is upper or lower triangular as follows:} \)
   
   \[ \begin{align*}
   & \text{if uplo = Nag_Upper, } A \text{ is upper triangular;} \\
   & \text{if uplo = Nag_Lower, } A \text{ is lower triangular.}
   \end{align*} \]
   \( \text{Constraint: uplo = Nag_Upper or Nag_Lower.} \)

3: `trans` – Nag_TransType
   \( \text{Input} \)  
   \( \text{On entry: indicates the form of the equations as follows:} \)
   
   \[ \begin{align*}
   & \text{if trans = Nag_NoTrans, then the equations are of the form } AX = B; \\
   & \text{if trans = Nag_Trans or Nag_ConjTrans, then the equations are of the form } A^T X = B.
   \end{align*} \]
   \( \text{Constraint: trans = Nag_NoTrans, Nag_Trans or Nag_ConjTrans.} \)

4: `diag` – Nag_DiagType
   \( \text{Input} \)  
   \( \text{On entry: indicates whether A is a non-unit or unit triangular matrix as follows:} \)
if \( \text{diag} = \text{Nag\_NonUnitDiag} \), \( A \) is a non-unit triangular matrix;

if \( \text{diag} = \text{Nag\_UnitDiag} \), \( A \) is a unit triangular matrix; the diagonal elements are not referenced and are assumed to be 1.

\[ \text{Constraint: diag} = \text{Nag\_NonUnitDiag} \text{ or } \text{Nag\_UnitDiag}. \]

5: \( n \) – Integer \( \quad \text{Input} \)

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

\[ \text{Constraint: } n \geq 0. \]

6: \( \text{nrhs} \) – Integer \( \quad \text{Input} \)

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

\[ \text{Constraint: nrhs} \geq 0. \]

7: \( \text{a}[\text{dim}] \) – const double \( \quad \text{Input} \)

Note: the dimension, \( \text{dim} \), of the array \( \text{a} \) must be at least max(1, \( \text{pda} \times n \)).

On entry: the \( n \) by \( n \) triangular matrix \( A \). If \( \text{uplo} = \text{Nag\_Upper} \), \( A \) is upper triangular and the elements of the array below the diagonal are not referenced; if \( \text{uplo} = \text{Nag\_Lower} \), \( A \) is lower triangular and the elements of the array above the diagonal are not referenced. If \( \text{diag} = \text{Nag\_UnitDiag} \), the diagonal elements of \( A \) are not referenced, but are assumed to be 1.

8: \( \text{pda} \) – Integer \( \quad \text{Input} \)

On entry: the stride separating row or column elements (depending on the value of \( \text{order} \)) of the matrix \( A \) in the array \( \text{a} \).

\[ \text{Constraint: pda} \geq \max(1, n). \]

9: \( \text{b}[\text{dim}] \) – double \( \quad \text{Input/Output} \)

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, \( \text{pdb} \times n \)) 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 \( \text{b}[ (i - 1) \times \text{pdb} + j - 1 ] \).

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

On exit: the \( n \) by \( r \) solution matrix \( X \).

10: \( \text{pdb} \) – Integer \( \quad \text{Input} \)

On entry: the stride separating matrix row or column elements (depending on the value of \( \text{order} \)) in the array \( \text{b} \).

Constraints:

\[ \text{if order} = \text{Nag\_ColMajor, pdb} \geq \max(1, n); \]

\[ \text{if order} = \text{Nag\_RowMajor, pdb} \geq \max(1, \text{nrhs}). \]

11: \( \text{fail} \) – NagError * \( \quad \text{Output} \)

The NAG error parameter (see the Essential Introduction).

6 \quad \text{Error Indicators and Warnings}

\text{NE\_INT}

On entry, \( n = \langle \text{value} \rangle \).

Constraint: \( n \geq 0 \).
On entry, \texttt{nrhs} = \langle value \rangle.
Constraint: \texttt{nrhs} \geq 0.

On entry, \texttt{pda} = \langle value \rangle.
Constraint: \texttt{pda} > 0.

On entry, \texttt{pdb} = \langle value \rangle.
Constraint: \texttt{pdb} > 0.

\textbf{NE_INT_2}
On entry, \texttt{pda} = \langle value \rangle, \texttt{n} = \langle value \rangle.
Constraint: \texttt{pda} \geq \text{max}(1, \texttt{n}).

On entry, \texttt{pdb} = \langle value \rangle, \texttt{n} = \langle value \rangle.
Constraint: \texttt{pdb} \geq \text{max}(1, \texttt{n}).

On entry, \texttt{pdb} = \langle value \rangle, \texttt{nrhs} = \langle value \rangle.
Constraint: \texttt{pdb} \geq \text{max}(1, \texttt{nrhs}).

\textbf{NE_SINGULAR}
a(\langle value \rangle, \langle value \rangle) is zero, and the matrix \texttt{A} is singular.

\textbf{NE_ALLOC_FAIL}
Memory allocation failed.

\textbf{NE_BAD_PARAM}
On entry, parameter \langle value \rangle had an illegal value.

\textbf{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.

\section{Accuracy}
The solutions of triangular systems of equations are usually computed to high accuracy. See Higham (1989).

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

\[ |E| \leq c(n)\epsilon|A|, \]

\(c(n)\) is a modest linear function of \texttt{n}, and \(\epsilon\) is the \textit{machine precision}.

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

\[ \frac{\|x - \hat{x}\|\infty}{\|x\|\infty} \leq c(n)\text{cond}(A, x)\epsilon, \] provided \(c(n)\text{cond}(A, x)\epsilon < 1,\)

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

Note that \(\text{cond}(A, x) \leq \text{cond}(A) = \|A^{-1}\|\|A\|\|x\|\infty \leq \kappa_\infty(A)\); \(\text{cond}(A, x)\) 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 \texttt{nag_dtrrfs (f07thc)}, and an estimate for \(\kappa_\infty(A)\) can be obtained by calling \texttt{nag_dtrcon (f07tgc)} with \texttt{norm} = \texttt{Nag InfNorm}.

\section{Further Comments}
The total number of floating-point operations is approximately \(n^2r\).
The complex analogue of this function is \texttt{nag_ztrtrs} (\texttt{f07tsc}).

9 Example

To solve the system of equations \( AX = B \), where
\[
A = \begin{pmatrix}
4.30 & 0.00 & 0.00 & 0.00 \\
-3.96 & -4.87 & 0.00 & 0.00 \\
0.40 & 0.31 & -8.02 & 0.00 \\
-0.27 & 0.07 & -5.95 & 0.12
\end{pmatrix}
\text{ and } B = \begin{pmatrix}
-12.90 & -21.50 \\
16.75 & 14.93 \\
-17.55 & 6.33 \\
-11.04 & 8.09
\end{pmatrix}.
\]

9.1 Program Text

/* nag_dtrtrs (f07tec) 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, n, nrhs, pda, pdb;
    Integer exit_status=0;
    Nag_UploType uplo_enum;
    NagError fail;
    Nag_OrderType order;
    /* Arrays */
    char uplo[2];
    double *a=0, *b=0;

    #ifdef NAG_COLUMN_MAJOR
    #define A(I,J) a[(J-1)*pda+I-1]
    #define B(I,J) b[(J-1)*pdb+I-1]
    order = Nag_ColMajor;
    #else
    #define A(I,J) a[(I-1)*pda+J-1]
    #define B(I,J) b[(I-1)*pdb+J-1]
    order = Nag_RowMajor;
    #endif

    INIT_FAIL(fail);
    Vprintf("f07tec Example Program Results\n\n");
    /* Skip heading in data file */
    Vscanf("%*[\n]");
    Vscanf("%ld%ld%*[\n]", &n, &nrhs);
    #ifdef NAG_COLUMN_MAJOR
    pda = n;
    pdb = n;
    #else
    pda = n;
    pdb = nrhs;
    #endif

    /* Allocate memory */
    if (!((a = NAG_ALLOC(n * n, double)) ||
          (b = NAG_ALLOC(n * nrhs, double)) )
        
    Vprintf("Allocation failure\n");
    
    /* Solve the system of equations AX = B */
    NAG_CALL(nag_dtrtrs)
    (order, n, n, a, n, &uplo[0], pda, b, nrhs, &uplo[0], pdb, &fail);
    
    /* Print the solution */
    Vprintf("Solution:
    ");
    for (i=0; i<n; i++)
    {
        for (j=0; j<nrhs; j++)
        {
            Vfprintf("%.15e", b[i*nrhs+j]);
            if (j < nrhs-1)
                Vfprintf(",");
        }
        Vfprintf("\n");
    }
    
    Vprintf("\n\nNumber of failures = %d\n", fail.nfail);
    return exit_status;
}
exit_status = -1;
goto END;
}
/* Read A and B from data file */
Vscanf(" %ls %*[\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;
}
if (uplo_enum == Nag_Upper)
{
    for (i = 1; i <= n; ++i)
    {
        for (j = i; j <= n; ++j)
            Vscanf("%lf", &A(i,j));
    }
    Vscanf("%*[\n"] ;
}
else
{
    for (i = 1; i <= n; ++i)
    {
        for (j = 1; j <= i; ++j)
            Vscanf("%lf", &A(i,j));
    }
    Vscanf("%*[\n"] ;
}
for (i = 1; i <= n; ++i)
{
    for (j = 1; j <= nrhs; ++j)
        Vscanf("%lf", &B(i,j));
    Vscanf("%*[\n"] ;
}
/* Compute solution */
f07tec(order, uplo_enum, Nag_NoTrans, Nag_NonUnitDiag, n, nrhs, a, pda, b, pdb, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07tec.\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", fail.message);
    exit_status = 1;
goto END;
}
END:
if (a) NAG_FREE(a);
if (b) NAG_FREE(b);
return exit_status;
9.2 Program Data

f07tec Example Program Data

4 2 :Values of N and NRHS
'L' :Value of UPLO

4.30

-3.96 -4.87
0.40 0.31 -8.02
-0.27 0.07 -5.95 0.12 :End of matrix A

-12.90 -21.50
16.75 14.93
-17.55 6.33
-11.04 8.09 :End of matrix B

9.3 Program Results

Solution(s)

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