NAG C Library Function Document
nag_dtrrfs (f07thc)

1 Purpose
nag_dtrrfs (f07thc) returns error bounds for the solution of a real triangular system of linear equations with multiple right-hand sides, \( AX = B \) or \( A^T X = B \).

2 Specification

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

3 Description
nag_dtrrfs (f07thc) returns the backward errors and estimated bounds on the forward errors for the solution of a real triangular system of linear equations with multiple right-hand sides \( AX = B \) or \( A^T X = B \). The function handles each right-hand side vector (stored as a column of the matrix \( B \)) independently, so we describe the function of nag_dtrrfs (f07thc) in terms of a single right-hand side \( b \) and solution \( x \).

Given a computed solution \( x \), the function computes the component-wise backward error \( \beta \). This is the size of the smallest relative perturbation in each element of \( A \) and \( b \) such that \( x \) is the exact solution of a perturbed system

\[
(A + \delta A)x = b + \delta b \quad |\delta a_{ij}| \leq \beta |a_{ij}| \quad \text{and} \quad |\delta b_i| \leq \beta |b_i|.
\]

Then the function estimates a bound for the component-wise forward error in the computed solution, defined by:

\[
\max_i |x_i - \hat{x}_i| / \max_i |x_i| \]

where \( \hat{x} \) is the true solution.

For details of the method, see the f07 Chapter Introduction.

4 References

5 Parameters

1: order – Nag_OrderType
   
   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
   
   On entry: indicates whether \( A \) is upper or lower triangular as follows:

   - Nag_Upper: \( A \) is upper triangular
   - Nag_Lower: \( A \) is lower triangular
if `uplo = Nag_Upper`, then $A$ is upper triangular;
if `uplo = Nag_Lower`, then $A$ is lower triangular.

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

3: `trans` – Nag_TransType
On entry: indicates the form of the equations as follows:
if `trans = Nag_NoTrans`, then the equations are of the form $AX = B$;
if `trans = Nag_Trans` or `Nag_ConjTrans`, then the equations are of the form $A^T X = B$.

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

4: `diag` – Nag_DiagType
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
On entry: $n$, the order of the matrix $A$.

**Constraint:** $n \geq 0$.

6: `nrhs` – Integer
On entry: $r$, the number of right-hand sides.

**Constraint:** $nrhs \geq 0$.

7: `a[dim]` – const double
On entry: the $n \times n$ triangular matrix $A$. If `uplo = Nag_Upper`, $A$ is upper triangular and the elements of the array below the diagonal are not referenced; if `uplo = Nag_Lower`, $A$ is lower triangular and the elements of the array above the diagonal are not referenced. If `diag = Nag_UnitDiag`, the diagonal elements of $A$ are not referenced, but are assumed to be 1.

8: `pda` – Integer
On entry: the stride separating row or column elements (depending on the value of `order`) of the matrix $A$ in the array $a$.

**Constraint:** $pda \geq \max(1, n)$.

9: `b[dim]` – const double
On entry: the $n \times r$ right-hand side matrix $B$.

**Note:** the dimension, `dim`, of the array $b$ must be at least $\max(1, pdb \times nrhs)$ when `order = Nag_ColMajor` and at least $\max(1, pdb \times n)$ when `order = Nag_RowMajor`.

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

On entry: the $n \times r$ right-hand side matrix $B$.

10: `pdb` – Integer
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 ≥ max(1, n);
if order = Nag_RowMajor, pdb ≥ max(1, nrhs).

11: \(x[dim] – \) const double

Note: the dimension, \(dim\), of the array \(x\) must be at least \(\max(1, pdx \times nrhs)\) when order = Nag_ColMajor and at least \(\max(1, pdx \times n)\) when order = Nag_RowMajor.

If order = Nag_ColMajor, the \((i, j)\)th element of the matrix \(X\) is stored in \(x[(j - 1) \times pdx + i - 1]\) and if order = Nag_RowMajor, the \((i, j)\)th element of the matrix \(X\) is stored in \(x[(i - 1) \times pdx + j - 1]\).

On entry: the \(n\) by \(r\) solution matrix \(X\), as returned by nag_dtrtrs (f07tec).

12: \(pdx – \) Integer

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

Constraints:

if order = Nag_ColMajor, pdx ≥ max(1, n);
if order = Nag_RowMajor, pdx ≥ max(1, nrhs).

13: \(ferr[dim] – \) double

Note: the dimension, \(dim\), of the array \(ferr\) must be at least \(\max(1, nrhs)\).

On exit: \(ferr[j - 1]\) contains an estimated error bound for the \(j\)th solution vector, that is, the \(j\)th column of \(X\), for \(j = 1, 2, \ldots, r\).

14: \(berr[dim] – \) double

Note: the dimension, \(dim\), of the array \(berr\) must be at least \(\max(1, nrhs)\).

On exit: \(berr[j - 1]\) contains the component-wise backward error bound \(\beta\) for the \(j\)th solution vector, that is, the \(j\)th column of \(X\), for \(j = 1, 2, \ldots, r\).

15: \(fail \) – NagError *

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

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

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

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

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

On entry, \(pdx = \langle\text{value}\rangle\).
Constraint: \(pdx > 0\).

NE_INT_2

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

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

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

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

**NE_ALLOC_FAIL**
Memory allocation failed.

**NE_BAD_PARAM**
On entry, parameter $\langle\text{value}\rangle$ 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 bounds returned in $\text{ferr}$ are not rigorous, because they are estimated, not computed exactly; but in practice they almost always overestimate the actual error.

8 **Further Comments**
A call to nag_dtrrfs (f07thc) involves, for each right-hand side, solving a number of systems of linear equations of the form $Ax = b$ or $A^T x = b$; the number is usually 4 or 5 and never more than 11. Each solution involves approximately $n^2$ floating-point operations.
The complex analogue of this function is nag_ztrrfs (f07tvc).

9 **Example**
To solve the system of equations $AX = B$ and to compute forward and backward error bounds, where

\[
A = \begin{pmatrix}
4.30 & 0.00 & 0.00 & 0.00 \\
-3.96 & -4.87 & 0.00 & 0.00 \\
0.40 & 0.31 & -8.02 & 0.00 \\
-0.27 & 0.07 & -5.95 & 0.12 \\
\end{pmatrix}
\quad \text{and} \quad
B = \begin{pmatrix}
-12.90 & -21.50 \\
16.75 & 14.93 \\
-17.55 & 6.33 \\
-11.04 & 8.09 \\
\end{pmatrix}.
\]

9.1 **Program Text**
/* nag_dtrrfs (f07thc) 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, berr_len, ferr_len;
Integer pda, pdb, pdx;
Integer exit_status=0;
Nag_UploType uplo_enum;

NagError fail;
Nag_OrderType order;
/* Arrays */
char uplo[2];
double *a=0, *b=0, *berr=0, *ferr=0, *x=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]
#define X(I,J) x[(J-1)*pdx+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]
#define X(I,J) x[(I-1)*pdx+J-1]
order = Nag_RowMajor;
#endif

INIT_FAIL(fail);
Vprintf("f07thc Example Program Results\n\n");

/* Skip heading in data file */
Vscanf("%*\[\n\] ");
Vscanf("%ld%ld%*\[\n\] ", &n, &nrhs);
berr_len = nrhs;
ferr_len = nrhs;
#ifdef NAG_COLUMN_MAJOR
pda = n;
pdb = n;
pdx = n;
#else
pda = n;
pdb = nrhs;
pdx = nrhs;
#endif

/* Allocate memory */
if ( !(a = NAG_ALLOC(n * n, double)) ||
    !(b = NAG_ALLOC(n * nrhs, double)) ||
    !(berr = NAG_ALLOC(berr_len, double)) ||
    !(ferr = NAG_ALLOC(ferr_len, double)) ||
    !(x = NAG_ALLOC(n * nrhs, double)) )
{
    Vprintf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

/* Read A and B from data file, and copy B to X */
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;
}
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 ]");
for (i = 1; i <= n; ++i)
{
    for (j = 1; j <= nrhs; ++j)
        X(i,j) = B(i,j);
}
/* Compute solution in the array X */
f07tec(order, uplo_enum, Nag_NoTrans, Nag_NonUnitDiag, n, nrhs, a, pda, x, pdx, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07tec.\n\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Compute backward errors and estimated bounds on the */
/* forward errors */
f07thc(order, uplo_enum, Nag_NoTrans, Nag_NonUnitDiag, n, nrhs, a, pda, b, pdb, x, pdx, ferr, berr, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07thc.\n\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Print solution */
Vprintf("\n");
x04cac(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, nrhs, x, pdx, "Solution(s)", 0, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from x04cac.\n\n", fail.message);
    exit_status = 1;
    goto END;
}
Vprintf("\nBackward errors (machine-dependent)\n");
for (j = 1; j <= nrhs; ++j)
    Vprintf("%11.1e%" "");
Vprintf("\nEstimated forward error bounds "
"(machine-dependent)\n");
for (j = 1; j <= nrhs; ++j)
    Vprintf("%11.1e%" "");
Vprintf("\n");
END:
if (a) NAG_FREE(a);
if (b) NAG_FREE(b);
if (berr) NAG_FREE(berr);
if (ferr) NAG_FREE(ferr);
if (x) NAG_FREE(x);

return exit_status;
}

9.2 Program Data

f07thc 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

f07thc Example 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>

Backward errors (machine-dependent)
6.9e-17 0.0e+00

Estimated forward error bounds (machine-dependent)
8.3e-14 2.6e-14