NAG Library Function Document

nag_ztbrfs (f07vvc)

1 Purpose

nag_ztbrfs (f07vvc) returns error bounds for the solution of a complex triangular band system of linear equations with multiple right-hand sides, $AX = B$, $A^TX = B$ or $A^HX = B$.

2 Specification

```c
#include <nag.h>
#include <nagf07.h>

void nag_ztbrfs (Nag_OrderType order, Nag_UploType uplo,
                 Nag_TransType trans, Nag_DiagType diag, Integer n, Integer kd,
                 Integer nrhs, const Complex ab[], Integer pdab, const Complex b[],
                 Integer pdb, const Complex x[], Integer pdx, double ferr[],
                 double berr[], NagError *fail)
```

3 Description

nag_ztbrfs (f07vvc) returns the backward error $s$ and estimated bounds on the forward errors for the solution of a complex triangular band system of linear equations with multiple right-hand sides $AX = B$, $A^TX = B$ or $A^HX = B$. The function handles each right-hand side vector (stored as a column of the matrix $B$) independently, so we describe the function of nag_ztbrfs (f07vvc) 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

$$(A + \delta A)x = b + \delta b$$

$$|\delta a_{ij}| \leq \beta |a_{ij}| \quad \text{and} \quad |\delta b_i| \leq \beta |b_i|.$$ 

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 Arguments

1. **order** – Nag_OrderType

   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.
uplo – Nag_UploType

On entry: specifies whether $A$ is upper or lower triangular.

$\text{uplo} = \text{Nag}_\text{Upper}$

$A$ is upper triangular.

$\text{uplo} = \text{Nag}_\text{Lower}$

$A$ is lower triangular.

Constraint: $\text{uplo} = \text{Nag}_\text{Upper}$ or $\text{Nag}_\text{Lower}$.

trans – Nag_TransType

On entry: indicates the form of the equations.

$\text{trans} = \text{Nag}_\text{NoTrans}$

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

$\text{trans} = \text{Nag}_\text{Trans}$

The equations are of the form $A^TX = B$.

$\text{trans} = \text{Nag}_\text{ConjTrans}$

The equations are of the form $A^HX = B$.

Constraint: $\text{trans} = \text{Nag}_\text{NoTrans}$, $\text{Nag}_\text{Trans}$ or $\text{Nag}_\text{ConjTrans}$.

diag – Nag_DiagType

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

$\text{diag} = \text{Nag}_\text{NonUnitDiag}$

$A$ is a nonunit triangular matrix.

$\text{diag} = \text{Nag}_\text{UnitDiag}$

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

Constraint: $\text{diag} = \text{Nag}_\text{NonUnitDiag}$ or $\text{Nag}_\text{UnitDiag}$.

n – Integer

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

Constraint: $n \geq 0$.

kd – Integer

On entry: $kd$, the number of superdiagonals of the matrix $A$ if $\text{uplo} = \text{Nag}_\text{Upper}$, or the number of subdiagonals if $\text{uplo} = \text{Nag}_\text{Lower}$.

Constraint: $kd \geq 0$.

nrhs – Integer

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

Constraint: $nrhs \geq 0$.

$\text{ab}[\text{dim}]$ – const Complex

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

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 $\text{uplo}$ arguments as follows:
if order = Nag_ColMajor and uplo = Nag_Upper,
    \[ A_{ij} \text{ is stored in } AB[k_d + i - j + (j - 1) \times pdab], \]
    for \( j = 1, \ldots, n \) and \( i = \max(1, j - k_d), \ldots, j \);
if order = Nag_ColMajor and uplo = Nag_Lower,
    \[ A_{ij} \text{ is stored in } AB[i - j + (j - 1) \times pdab], \]
    for \( j = 1, \ldots, n \) and \( i = j, \ldots, \min(n, j + k_d) \);
if order = Nag_RowMajor and uplo = Nag_Upper,
    \[ A_{ij} \text{ is stored in } AB[j - i + (i - 1) \times pdab], \]
    for \( i = 1, \ldots, n \) and \( j = i, \ldots, \min(n, i + k_d) \);
if order = Nag_RowMajor and uplo = Nag_Lower,
    \[ A_{ij} \text{ is stored in } AB[k_d + j - i + (i - 1) \times pdab], \]
    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: \[ \text{pdab} \] \hspace{1cm} \text{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 ≥ kd + 1.

10: \[ b[dim] \] \hspace{1cm} \text{Input}

Note: the dimension, \( dim \), of the array \( b \) must be at least
\[ \max(1, \text{pdab} \times \text{nrhs}) \] when order = Nag_ColMajor;
\[ \max(1, n \times \text{pdab}) \] when order = Nag_RowMajor.

The \( (i,j) \)th element of the matrix \( B \) is stored in
\[ b[(j - 1) \times \text{pdab} + i - 1] \] when order = Nag_ColMajor;
\[ b[(i - 1) \times \text{pdab} + j - 1] \] when order = Nag_RowMajor.

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

11: \[ \text{pdb} \] \hspace{1cm} \text{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, \( \text{pdb} \geq \max(1, n) \;
if order = Nag_RowMajor, \( \text{pdb} \geq \max(1, \text{nrhs}) \).

12: \[ x[dim] \] \hspace{1cm} \text{Input}

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

The \( (i,j) \)th element of the matrix \( X \) is stored in
\[ x[(j - 1) \times \text{pdx} + i - 1] \] when order = Nag_ColMajor;
\[ x[(i - 1) \times \text{pdx} + j - 1] \] when order = Nag_RowMajor.

On entry: the \( n \) by \( r \) solution matrix \( X \), as returned by nag_ztbtrs (f07vsc).

13: \[ \text{pdx} \] \hspace{1cm} \text{Input}

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

if order = Nag_ColMajor, \( \text{pdx} \geq \max(1, n) \);
if order = Nag_RowMajor, \( \text{pdx} \geq \max(1, \text{nrhs}) \).

14: ferr[\text{nrhs}] – double

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 \( j = 1, 2, \ldots, r \).

15: berr[\text{nrhs}] – double

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 \( j = 1, 2, \ldots, r \).

16: fail – NagError*

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 \( \langle \text{value} \rangle \) had an illegal value.

NE_INT

On entry, \( \text{kd} = \langle \text{value} \rangle \).
Constraint: \( \text{kd} \geq 0 \).
On entry, \( \text{n} = \langle \text{value} \rangle \).
Constraint: \( \text{n} \geq 0 \).
On entry, \( \text{nrhs} = \langle \text{value} \rangle \).
Constraint: \( \text{nrhs} \geq 0 \).
On entry, \( \text{pdab} = \langle \text{value} \rangle \).
Constraint: \( \text{pdab} > 0 \).
On entry, \( \text{pdb} = \langle \text{value} \rangle \).
Constraint: \( \text{pdb} > 0 \).
On entry, \( \text{pdx} = \langle \text{value} \rangle \).
Constraint: \( \text{pdx} > 0 \).

NE_INT_2

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

7 Accuracy

The bounds returned in `ferr` are not rigorous, because they are estimated, not computed exactly; but in practice they almost always overestimate the actual error.

8 Parallelism and Performance

`nag_ztbrfs (f07vvc)` is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

`nag_ztbrfs (f07vvc)` 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

A call to `nag_ztbrfs (f07vvc)`, for each right-hand side, involves solving a number of systems of linear equations of the form $Ax = b$ or $A^Hx = b$; the number is usually 5 and never more than 11. Each solution involves approximately $8nk$ real floating-point operations (assuming $n \gg k$).

The real analogue of this function is `nag_dtbrfs (f07vhc)`.

10 Example

This example solves the system of equations $AX = B$ and to compute forward and backward error bounds, where

$$A = \begin{pmatrix}
-1.94 + 4.43i & 0.00 + 0.00i & 0.00 + 0.00i & 0.00 + 0.00i \\
-3.39 + 3.44i & 4.12 - 4.27i & 0.00 + 0.00i & 0.00 + 0.00i \\
1.62 + 3.68i & -1.84 + 5.53i & 0.43 - 2.66i & 0.00 + 0.00i \\
0.00 + 0.00i & -2.77 - 1.93i & 1.74 - 0.04i & 0.44 + 0.10i
\end{pmatrix}$$

and

$$B = \begin{pmatrix}
-8.86 - 3.88i & -24.09 - 5.27i \\
-15.77 - 23.41i & -57.97 + 8.14i \\
-7.63 + 22.78i & 19.09 - 29.51i \\
14.74 - 2.40i & 19.17 + 21.33i
\end{pmatrix}.$$
10.1 Program Text

/* nag_ztbrfs (f07vvc) 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, pdx;
    Integer ferr_len, berr_len;
    Integer exit_status = 0;
    Nag_UploType uplo;
    NagError fail;
    Nag_OrderType order;
    /* Arrays */
    char nag_enum_arg[40];
    Complex *ab = 0, *b = 0, *x = 0;
    double *berr = 0, *ferr = 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]
    #define X(I, J) x[(J-1)*pdx + 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]
    #define X(I, J) x[(I-1)*pdx + J - 1]
    order = Nag_RowMajor;
    #endif

    INIT_FAIL(fail);
    printf("nag_ztbrfs (f07vvc) 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;
    pdx = n;
    #else
    pdb = nrhs;
    pdx = nrhs;
    #endif

    ferr_len = nrhs;
    berr_len = nrhs;

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

/* Read A and B from data file, and copy B to X */
#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 , %lf )", &AB_UPPER(i, j).re,
                        &AB_UPPER(i, j).im);
            #else
                scanf(" ( %lf , %lf )", &AB_UPPER(i, j).re,
                        &AB_LOWER(i, j).im);
            #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 , %lf )", &AB_LOWER(i, j).re,
                        &AB_LOWER(i, j).im);
            #else
                scanf(" ( %lf , %lf )", &AB_LOWER(i, j).re,
                        &AB_LOWER(i, j).im);
            #endif
        }
    }
    #ifdef _WIN32
        scanf_s("%*[\n] ");
    #else
        scanf("%*[\n] ");
    #endif
}
for (i = 1; i <= n; ++i)
{
    for (j = 1; j <= nrhs; ++j)
        #ifdef _WIN32
            scanf_s(" ( %lf , %lf )", &B(i, j).re, &B(i, j).im);
        #else
            scanf(" ( %lf , %lf )", &B(i, j).re, &B(i, j).im);
        #endif
}
```c
/* Copy B to X */
for (i = 1; i <= n; ++i)
{
    for (j = 1; j <= nrhs; ++j)
    {
        X(i, j).re = B(i, j).re;
        X(i, j).im = B(i, j).im;
    }
}

/* Improve solution, and compute backward errors and */
/* estimated bounds on the forward errors */
/* nag_ztbrfs (f07vvc).  
* Error bounds for solution of complex band triangular  
* system of linear equations, multiple right-hand sides */
/* nag_ztbrfs(order, uplo, Nag_NoTrans, Nag_NonUnitDiag, n,  
  kd, nrhs, ab, pdab, x, pdx, ferr, berr,  
  &fail); */
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_ztbrfs (f07vvc).\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Print solution */
/* nag_gen_complx_mat_print_comp (x04dbc).  
* Print complex general matrix (comprehensive) */
fflush(stdout);
/* nag_gen_complx_mat_print_comp(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n,  
  nrhs, x, pdx, Nag_BracketForm, "%7.4f",  
  "Solution(s)", Nag_IntegerLabels, 0,  
  Nag_IntegerLabels, 0, 80, 0, 0, &fail); */
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_gen_complx_mat_print_comp (x04dbc).\n", fail.message);
    exit_status = 1;
    goto END;
}

printf("\nBackward errors (machine-dependent)\n");
for (j = 1; j <= nrhs; ++j)
    printf("%11.1e%s", berr[j-1], j%7 == 0?"\n":" ");
printf("\nEstimated forward error bounds (machine-dependent)\n");
for (j = 1; j <= nrhs; ++j)
    printf("%11.1e%s", ferr[j-1], j%7 == 0?"\n":" ");
END:
NAG_FREE(berr);
```
NAG_FREE(ferr);
NAG_FREE(ab);
NAG_FREE(b);
NAG_FREE(x);
return exit_status;
}

## 10.2 Program Data

nag_ztbrfs (f07vvc) Example Program Data

<table>
<thead>
<tr>
<th>n</th>
<th>k</th>
<th>nrhs</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Nag_Lower :Value of uplo
(-1.94, 4.43)
(-3.39, 3.44) ( 4.12, -4.27)
( 1.62, 3.68) (-1.84, 5.53) ( 0.43, -2.66)
(-2.77, -1.93) ( 1.74, -0.04) ( 0.44, 0.10) :End of matrix A
(-8.86, -3.88) (-24.09, -5.27)
(-15.57, -23.41) (-57.97, 8.14)
(-7.63, 22.78) ( 19.09, -29.51)
(-14.74, -2.40) ( 19.17, 21.33) :End of matrix B

## 10.3 Program Results

nag_ztbrfs (f07vvc) Example Program Results

Solution(s)

<table>
<thead>
<tr>
<th>i</th>
<th>x1</th>
<th>x2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( 0.0000, 2.0000)</td>
<td>( 1.0000, 5.0000)</td>
</tr>
<tr>
<td>2</td>
<td>( 1.0000, -3.0000)</td>
<td>( -7.0000, -2.0000)</td>
</tr>
<tr>
<td>3</td>
<td>(-4.0000, -5.0000)</td>
<td>( 3.0000, 4.0000)</td>
</tr>
<tr>
<td>4</td>
<td>( 2.0000, -1.0000)</td>
<td>( -6.0000, -9.0000)</td>
</tr>
</tbody>
</table>

Backward errors (machine-dependent)
4.1e-17 4.2e-17

Estimated forward error bounds (machine-dependent)
1.8e-14 2.2e-14