NAG C Library Function Document

nag_zgetrs (f07asc)

1 Purpose

nag_zgetrs (f07asc) solves a complex system of linear equations with multiple right-hand sides, $AX = B$, $A^TX = B$ or $A^H X = B$, where $A$ has been factorized by nag_zgetrf (f07arc).

2 Specification

```c
void nag_zgetrs (Nag_OrderType order, Nag_TransType trans, Integer n, Integer nrhs,
                   const Complex a[], Integer pda, const Integer ipiv[], Complex b[],
                   Integer pdb, NagError *fail)
```

3 Description

To solve a complex system of linear equations $AX = B$, $A^TX = B$ or $A^H X = B$, this function must be preceded by a call to nag_zgetrf (f07arc) which computes the $LU$ factorization of $A$ as $A = PLU$. The solution is computed by forward and backward substitution.

If $trans = \text{Nag\_NoTrans}$, the solution is computed by solving $PLY = B$ and then $UX = Y$.

If $trans = \text{Nag\_Trans}$, the solution is computed by solving $U^TY = B$ and then $LT^TPTX = Y$.

If $trans = \text{Nag\_ConjTrans}$, the solution is computed by solving $U^HY = B$ and then $L^HT^TPTX = Y$.

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:  
trans – Nag_TransType

On entry: indicates the form of the equations as follows:

if trans = Nag_NoTrans, $AX = B$ is solved for $X$;

if trans = Nag_Trans, $A^TX = B$ is solved for $X$;

if trans = Nag_ConjTrans, $A^H X = B$ is solved for $X$.

Constraint: trans = Nag_NoTrans, Nag_Trans or Nag_ConjTrans.

3:  
n – Integer

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

Constraint: $n \geq 0$. 

[NP3645/7]
4: \( \textbf{nrhs} \) – Integer  
\textit{Input}

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

\textit{Constraint:} \( \textbf{nrhs} \geq 0 \). 

5: \( \textbf{a}[\text{dim}] \) – const Complex  
\textit{Input}

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

If \( \text{order} = \text{Nag}\_\text{ColMajor} \), the \((i, j)\)th element of the matrix \( A \) is stored in \( \textbf{a}[(j - 1) \times \text{pda} + i - 1] \) and if \( \text{order} = \text{Nag}\_\text{RowMajor} \), the \((i, j)\)th element of the matrix \( A \) is stored in \( \textbf{a}[(i - 1) \times \text{pda} + j - 1] \).

\textit{On entry:} the \( LU \) factorization of \( A \), as returned by \texttt{nag_zgetrf} (f07arc).

6: \( \textbf{pda} \) – Integer  
\textit{Input}

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

\textit{Constraint:} \( \textbf{pda} \geq \max(1, \textbf{n}) \). 

7: \( \textbf{ipiv}[\text{dim}] \) – const Integer  
\textit{Input}

\textit{Note:} the dimension, \( \text{dim} \), of the array \( \textbf{ipiv} \) must be at least \( \max(1, \textbf{n}) \).

\textit{On entry:} the pivot indices, as returned by \texttt{nag_zgetrf} (f07arc).

8: \( \textbf{b}[\text{dim}] \) – Complex  
\textit{Input/Output}

\textit{Note:} the dimension, \( \text{dim} \), of the array \( \textbf{b} \) must be at least \( \max(1, \text{pdb} \times \textbf{nrhs}) \) when \( \text{order} = \text{Nag}\_\text{ColMajor} \) and at least \( \max(1, \text{pdb} \times \textbf{n}) \) when \( \text{order} = \text{Nag}\_\text{RowMajor} \).

If \( \text{order} = \text{Nag}\_\text{ColMajor} \), the \((i, j)\)th element of the matrix \( B \) is stored in \( \textbf{b}[(j - 1) \times \text{pdb} + i - 1] \) and if \( \text{order} = \text{Nag}\_\text{RowMajor} \), the \((i, j)\)th element of the matrix \( B \) is stored in \( \textbf{b}[(i - 1) \times \text{pdb} + j - 1] \).

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

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

9: \( \textbf{pdb} \) – Integer  
\textit{Input}

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

\textit{Constraints:}

\begin{align*}
\text{if} \quad \text{order} = \text{Nag}\_\text{ColMajor}, \quad & \textbf{pdb} \geq \max(1, \textbf{n}) ; \\
\text{if} \quad \text{order} = \text{Nag}\_\text{RowMajor}, \quad & \textbf{pdb} \geq \max(1, \textbf{nrhs}) .
\end{align*}

10: \( \textbf{fail} \) – NagError *  
\textit{Output}

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

\textbf{NE\_INT}

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

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

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

\textit{On entry,} \( \textbf{pdb} = \langle \text{value} \rangle \).  
\textit{Constraint:} \( \textbf{pdb} \geq 0 \).
NE_INT_2
On entry, pda = ⟨value⟩, n = ⟨value⟩.
Constraint: pda ≥ max(1, n).
On entry, pdb = ⟨value⟩, n = ⟨value⟩.
Constraint: pdb ≥ max(1, n).
On entry, pdb = ⟨value⟩, nrhs = ⟨value⟩.
Constraint: pdb ≥ max(1, nrhs).

NE_ALLOC_FAIL
Memory allocation failed.

NE_BAD_PARAM
On entry, parameter ⟨value⟩ 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
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

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

c(n) is a modest linear function of n, and \( \epsilon \) is the machine precision.
If \( \hat{x} \) is the true solution, then the computed solution x satisfies a forward error bound of the form

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

where \text{cond}(A, x) = \|A^{-1}\| \|A\| \|x\|_\infty / \|x\|_\infty \leq \text{cond}(A) = \|A^{-1}\| \|A\| \|x\|_\infty \leq \kappa_\infty(A). \] Note that \text{cond}(A, x) can be much smaller than \text{cond}(A), and \text{cond}(A^{H}) (which is the same as \text{cond}(A^{T})) can be much larger (or smaller) than \text{cond}(A).
Forward and backward error bounds can be computed by calling nag_zgerfs (f07avc), and an estimate for \( \kappa_\infty(A) \) can be obtained by calling nag_zgecon (f07auc) with \text{norm} = \text{Nag InfNorm}.

8 Further Comments
The total number of real floating-point operations is approximately 8n^2r.
This function may be followed by a call to nag_zgerfs (f07avc) to refine the solution and return an error estimate.
The real analogue of this function is nag_dgetrs (f07aec).

9 Example
To solve the system of equations AX = B, where

\[ A = \begin{pmatrix}
-1.34 + 2.55i & 0.28 + 3.17i & -6.39 - 2.20i & 0.72 - 0.92i \\
-0.17 - 1.41i & 3.31 - 0.15i & -0.15 + 1.34i & 1.29 + 1.38i \\
-3.29 - 2.39i & 1.91 + 4.42i & -0.14 - 1.35i & 1.72 + 1.35i \\
2.41 + 0.39i & -0.56 + 1.47i & -0.83 - 0.69i & -1.96 + 0.67i
\end{pmatrix} \]

and
Here $A$ is nonsymmetric and must first be factorized by nag_zgetrf (f07arc).

9.1 Program Text

/* nag_zgetrs (f07asc) Example Program.
 * * Copyright 2001 Numerical Algorithms Group.
 * * Mark 7, 2001.
 * /

#include <stdio.h>
#include <nag.h>
#include <nagf07.h>
#include <nagx04.h>

int main(void)
{
   /* Scalars */
   Integer i, ipiv_len, j, n, nrhs, pda, pdb;
   Integer exit_status=0;
   NagError fail;
   Nag_OrderType order;
   /* Arrays */
   Complex *a=0, *b=0;
   Integer *ipiv=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("f07asc Example Program Results\n\n");
   /* Skip heading in data file */
   Vscanf("%*[^
"]");
   Vscanf("%ld%ld%*[^
"]", &n, &nrhs);
   #ifdef NAG_COLUMN_MAJOR
   pda = n;
   pdb = n;
   #else
   pda = n;
   pdb = nrhs;
   #endif

   ipiv_len = n;

   /* Allocate memory */
   if ( !(a = NAG_ALLOC(n * n, Complex)) ||
       !(b = NAG_ALLOC(n * nrhs, Complex)) ||
       !(ipiv = NAG_ALLOC(ipiv_len, Integer))
   )
   {
      Vprintf("Allocation failure\n");
      exit_status = -1;
      goto END;
   }

   /* Read A and B from data file */
for (i = 1; i <= n; ++i)
{
    for (j = 1; j <= n; ++j)
    {
        Vscanf(" ( %lf , %lf )", &A(i,j).re, &A(i,j).im);
    }
}
Vscanf("%*[\n ] ");
for (i = 1; i <= n; ++i)
{
    for (j = 1; j <= nrhs; ++j)
    {
        Vscanf(" ( %lf , %lf )", &B(i,j).re, &B(i,j).im);
    }
}
Vscanf("%*[\n ] ");

    /* Factorize A */
    f07arc(order, n, n, a, pda, ipiv, &fail);
    if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from f07arc.
%s
", fail.message);
        exit_status = 1;
        goto END;
    }

    /* Compute solution */
    f07asc(order, Nag_NoTrans, n, nrhs, a, pda, ipiv, b, pdb, &fail);
    if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from f07asc.
%s
", fail.message);
        exit_status = 1;
        goto END;
    }

    /* Print solution */
    x04dbc(order, Nag_GeneralMatrix, Nag_NoUnitDiag, n, nrhs, b, pdb,
           Nag_BracketForm, "%7.4f", "Solution(s)", Nag_IntegerLabels, 0,
           Nag_IntegerLabels, 0, 80, 0, 0, &fail);
    if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from x04dbc.
%s
", fail.message);
        exit_status = 1;
        goto END;
    }
END:
if (a) NAG_FREE(a);
if (b) NAG_FREE(b);
if (ipiv) NAG_FREE(ipiv);
return exit_status;

9.2 Program Data

f07asc Example Program Data

4 2 :Values of N and NRHS
(-1.34, 2.55) ( 0.28, 3.17) (-6.39,-2.20) ( 0.72,-0.92)
(-0.17,-1.41) ( 3.31,-0.15) (-0.15, 1.34) ( 1.29, 1.38)
(-3.29,-2.39) (-1.91, 4.42) (-0.14,-1.35) ( 1.72, 1.35)
 (2.41, 0.39) (-0.56, 1.47) (-0.83,-0.69) (-1.96, 0.67) :End of matrix A
(26.26, 51.78) (31.32, -6.70)
 (6.43, -8.68) (15.86, -1.42)
(-5.75, 25.31) (-2.15, 30.19)
 (1.16, 2.57) (-2.56, 7.55) :End of matrix B

[NP3645/7] f07asc.5
### 9.3 Program Results

**f07asc Example Program Results**

<table>
<thead>
<tr>
<th>Solution(s)</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( 1.0000, 1.0000)</td>
<td>(-1.0000, -2.0000)</td>
</tr>
<tr>
<td>2</td>
<td>( 2.0000, -3.0000)</td>
<td>( 5.0000,  1.0000)</td>
</tr>
<tr>
<td>3</td>
<td>(-4.0000, -5.0000)</td>
<td>(-3.0000,  4.0000)</td>
</tr>
<tr>
<td>4</td>
<td>( 0.0000,  6.0000)</td>
<td>( 2.0000, -3.0000)</td>
</tr>
</tbody>
</table>