NAG Library Function Document
nag_zhesv (f07mnc)

1 Purpose
nag_zhesv (f07mnc) computes the solution to a complex system of linear equations

\[ AX = B, \]

where \( A \) is an \( n \) by \( n \) Hermitian matrix and \( X \) and \( B \) are \( n \) by \( r \) matrices.

2 Specification

```c
#include <nag.h>
#include <nagf07.h>
void nag_zhesv (Nag_OrderType order, Nag_UploType uplo, Integer n,
                Integer nrhs, Complex a[], Integer pda, Integer ipiv[],
                Complex b[], Integer pdb, NagError *fail)
```

3 Description

nag_zhesv (f07mnc) uses the diagonal pivoting method to factor \( A \) as

- \( U (or \ L) \) is a product of permutation and unit upper (lower) triangular matrices, and \( D \) is Hermitian and block diagonal with 1 by 1 and 2 by 2 diagonal blocks. The factored form of \( A \) is then used to solve the system of equations \( AX = B \).

4 References

Anderson E, Bai Z, Bischof C, Blackford S, Demmel J, Dongarra J J, Du Croz J J, Greenbaum A,
Philadelphia http://www.netlib.org/lapack/lug

Press, Baltimore

5 Arguments

1: \texttt{order} – Nag_OrderType

\textit{Input}

\textit{On entry}: the \texttt{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 \texttt{order} = Nag_RowMajor. See Section 3.2.1.3 in the Essential Introduction for a more detailed
explanation of the use of this argument.

\textit{Constraint}: \texttt{order} = Nag_RowMajor or Nag_ColMajor.

2: \texttt{uplo} – Nag_UploType

\textit{Input}

\textit{On entry}: if \texttt{uplo} = Nag_Upper, the upper triangle of \( A \) is stored.

Mark 25
If $\text{uplo} = \text{Nag}_\text{Lower}$, the lower triangle of $A$ is stored.

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

3: $n$ – Integer

*Input*

*On entry:* $n$, the number of linear equations, i.e., the order of the matrix $A$.

*Constraint:* $n \geq 0$.

4: $\text{nrhs}$ – Integer

*Input*

*On entry:* $r$, the number of right-hand sides, i.e., the number of columns of the matrix $B$.

*Constraint:* $\text{nrhs} \geq 0$.

5: $a[\text{dim}]$ – Complex

*Input/Output*

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

*On entry:* the $n$ by $n$ Hermitian matrix $A$.

If $\text{order} = \text{Nag}_\text{ColMajor}$, $A_{ij}$ is stored in $a[(j - 1) \times \text{pda} + i - 1]$.

If $\text{order} = \text{Nag}_\text{RowMajor}$, $A_{ij}$ is stored in $a[(i - 1) \times \text{pda} + j - 1]$.

If $\text{uplo} = \text{Nag}_\text{Upper}$, the upper triangular part of $A$ must be stored and the elements of the array below the diagonal are not referenced.

If $\text{uplo} = \text{Nag}_\text{Lower}$, the lower triangular part of $A$ must be stored and the elements of the array above the diagonal are not referenced.

*On exit:* if $\text{fail.code} = \text{NE}_\text{NOERROR}$, the block diagonal matrix $D$ and the multipliers used to obtain the factor $U$ or $L$ from the factorization $A = UD_{U}$, $A = LDL_{U}$, $A = U^{H}DU$ or $A = L^{H}DL$ as computed by nag_zhetrf (f07mrc).

6: $\text{pda}$ – Integer

*Input*

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

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

7: $\text{ipiv}[\text{dim}]$ – Integer

*Output*

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

*On exit:* details of the interchanges and the block structure of $D$. More precisely,

- if $\text{ipiv}[i - 1] = k > 0$, $d_{ii}$ is a 1 by 1 pivot block and the $i$th row and column of $A$ were interchanged with the $k$th row and column;

- if $\text{uplo} = \text{Nag}_\text{Upper}$ and $\text{ipiv}[i - 2] = \text{ipiv}[i - 1] = -l < 0$, $\begin{pmatrix} \tilde{d}_{i+1,i-1} & \tilde{d}_{i,i-1} \\ \tilde{d}_{i,i-1} & \tilde{d}_{i,i} \end{pmatrix}$ is a 2 by 2 pivot block and the $(i - 1)$th row and column of $A$ were interchanged with the $l$th row and column;

- if $\text{uplo} = \text{Nag}_\text{Lower}$ and $\text{ipiv}[i - 1] = \text{ipiv}[i] = -m < 0$, $\begin{pmatrix} d_{i,i} & d_{i+1,i} \\ d_{i+1,i} & d_{i+1,i+1} \end{pmatrix}$ is a 2 by 2 pivot block and the $(i + 1)$th row and column of $A$ were interchanged with the $m$th row and column.

8: $\text{b}[\text{dim}]$ – Complex

*Input/Output*

**Note:** the dimension, $\text{dim}$, of the array $\text{b}$ must be at least

- max$(1, \text{pdb} \times \text{nrhs})$ when $\text{order} = \text{Nag}_\text{ColMajor}$;
- max$(1, n \times \text{pdb})$ when $\text{order} = \text{Nag}_\text{RowMajor}$.
The \((i, j)\)th element of the matrix \(B\) is stored in
\[
\begin{align*}
\text{b}[(j-1) \times \text{pdb} + i - 1] & \text{ when order = Nag\_ColMajor;} \\
\text{b}[(i-1) \times \text{pdb} + j - 1] & \text{ when order = Nag\_RowMajor.}
\end{align*}
\]

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

On exit: if \text{fail\_code} = NE\_NOERROR, the \(n\) by \(r\) solution matrix \(X\).

9: \quad \text{pdb} \quad \text{– Integer} \quad \text{Input}

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

Constraints:
\[
\begin{align*}
\text{if order = Nag\_ColMajor, } \text{pdb} & \geq \max(1, n); \\
\text{if order = Nag\_RowMajor, } \text{pdb} & \geq \max(1, \text{nrhs}).
\end{align*}
\]

10: \quad \text{fail} \quad \text{– NagError } \ast \quad \text{Input/Output}

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 \quad \text{Error Indicators and Warnings}

\textbf{NE\_ALLOC\_FAIL}

Dynamic memory allocation failed.

See Section 3.2.1.2 in the Essential Introduction for further information.

\textbf{NE\_BAD\_PARAM}

On entry, argument \langle value \rangle had an illegal value.

\textbf{NE\_INT}

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

Constraint: \(n \geq 0\).

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

Constraint: \(\text{nrhs} \geq 0\).

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

Constraint: \(\text{pda} > 0\).

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

Constraint: \(\text{pdb} > 0\).

\textbf{NE\_INT\_2}

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

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

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

Constraint: \(\text{pdb} \geq \max(1, n)\).

On entry, \(\text{pdb} = \langle value \rangle\) and \(\text{nrhs} = \langle value \rangle\).

Constraint: \(\text{pdb} \geq \max(1, \text{nrhs})\).

\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 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.
7 Accuracy

The computed solution for a single right-hand side, $\hat{x}$, satisfies an equation of the form

$$(A + E)\hat{x} = b,$$

where

$$\|E\|_1 = O(\varepsilon)\|A\|_1$$

and $\varepsilon$ is the machine precision. An approximate error bound for the computed solution is given by

$$\frac{\|\hat{x} - x\|_1}{\|x\|_1} \leq \kappa(A)\frac{\|E\|_1}{\|A\|_1},$$

where $\kappa(A) = \|A^{-1}\|_1\|A\|_1$, the condition number of $A$ with respect to the solution of the linear equations. See Section 4.4 of Anderson et al. (1999) for further details.

8 Parallelism and Performance

nag_zhesvx (f07mpc) is a comprehensive LAPACK driver that returns forward and backward error bounds and an estimate of the condition number. Alternatively, nag_herm_lin_solve (f04chc) solves $Ax = b$ and returns a forward error bound and condition estimate. nag_herm_lin_solve (f04chc) calls nag_zhesv (f07mnc) to solve the equations.

9 Further Comments

The total number of floating-point operations is approximately $\frac{3}{4}n^3 + 8n^2r$, where $r$ is the number of right-hand sides.

The real analogue of this function is nag_dsysv (f07mac). The complex symmetric analogue of this function is nag_zsysv (f07nnc).

10 Example

This example solves the equations

$$Ax = b,$$

where $A$ is the Hermitian matrix.
\[
A = \begin{pmatrix}
-1.84 & 0.11 - 0.11i & -1.78 - 1.18i & 3.91 - 1.50i \\
0.11 + 0.11i & -4.63 & -1.84 + 0.03i & 2.21 + 0.21i \\
-1.78 + 1.18i & -1.84 - 0.03i & -8.87 & 1.58 - 0.90i \\
3.91 + 1.50i & 2.21 - 0.21i & 1.58 + 0.90i & -1.36
\end{pmatrix}
\]

and

\[
b = \begin{pmatrix}
2.98 - 10.18i \\
-9.58 + 3.88i \\
-0.77 - 16.05i \\
7.79 + 5.48i
\end{pmatrix}.
\]

Details of the factorization of \(A\) are also output.

### 10.1 Program Text

/* nag_zhesv (f07mnc) Example Program.  
 * Copyright 2014 Numerical Algorithms Group.  
 * Mark 23, 2011.  
 */

#include <stdio.h>
#include <nag.h>
#include <naq_stdlib.h>
#include <naqf07.h>

int main(void)
{

    /* Scalars */
    Integer exit_status = 0, i, j, n, nrhs, pda, pdb;

    /* Arrays */
    Complex *a = 0, *b = 0;
    Integer *ipiv = 0;
    char nag_enum_arg[40];

    /* Nag Types */
    NagError fail;
    Nag_UploType uplo;
    Nag_OrderType order;

    #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);
    printf("nag_zhesv (f07mnc) 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"%*[\n]", &n, &nrhs);
    #else

Mark 25

f07mnc.5
scanf("%"NAG_IFMT"%"NAG_IFMT"%*[\n]", &n, &nrhs);
#endif
if (n < 0 || nrhs < 0)
{
    printf("Invalid n or nrhs\n");
    exit_status = 1;
    goto END;
}
#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);
/* Allocate memory */
if (!(a = NAG_ALLOC(n * n, Complex)) ||
    !(b = NAG_ALLOC(n*nrhs, Complex)) ||
    !(ipiv = NAG_ALLOC(n, Integer)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}
    pda = n;
#ifdef NAG_COLUMN_MAJOR
    pdb = n;
#else
    pdb = nrhs;
#endif
/* Read the triangular part of the matrix A from data file */
if (uplo == Nag_Upper)
    for (i = 1; i <= n; ++i)
        for (j = i; j <= n; ++j)
            #ifdef _WIN32
                scanf_s(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
            #else
                scanf(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
            #endif
    else
        for (i = 1; i <= n; ++i)
            for (j = 1; j <= i; ++j)
                #ifdef _WIN32
                    scanf_s(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
                #else
                    scanf(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
                #endif
        #ifdef _WIN32
            scanf_s("%*[\n]");
        #else
            scanf("%*[\n]");
        #endif
/* Read b from data file */
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
        #ifdef _WIN32
            scanf_s("%*[\n]");
        #else
            scanf("%*[\n]");
        #endif
/* Solve the equations Ax = b for x using nag_zhesv (f07mnc). */
nag_zhesv(order, uplo, n, nrhs, a, pda, ipiv, b, pdb, &fail);
if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_zhesv (f07mnc).\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }

/* Print solution */
printf(" Solution\n");
for (i = 1; i <= n; ++i)
    {
        for (j = 1; j <= nrhs; ++j)
            printf(" (%7.4f, %7.4f)%s", B(i, j).re, B(i, j).im, j%4 == 0?"\n":"");
        printf("\n");
    }

END:
NAG_FREE(a);
NAG_FREE(b);
NAG_FREE(ipiv);

return exit_status;
}
#undef A
#undef B

10.2 Program Data

nag_zhesv (f07mnc) Example Program Data

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Nag_Lower</td>
<td>n, nrhs</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( -1.84, 0.00)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 0.11, 0.11)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( -1.78, 1.18)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( -1.84, -0.03)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( -1.87, 0.00)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 2.21, -0.21)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 1.58, 0.90)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( -1.36 , 0.00)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 7.79, 5.48)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

: matrix A

10.3 Program Results

nag_zhesv (f07mnc) Example Program Results

Solution

( 2.0000, 1.0000)
( 3.0000, -2.0000)
(-1.0000, 2.0000)
( 1.0000, -1.0000)