NAG Library Function Document

nag_zsysv (f07nnc)

1 Purpose

nag_zsysv (f07nnc) computes the solution to a complex system of linear equations

\[ AX = B, \]

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

2 Specification

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

void nag_zsysv (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_zsysv (f07nnc) uses the diagonal pivoting method to factor \( A \) as

\[
\begin{align*}
\text{order} & \quad \text{uplo} & \quad A \\
\text{Nag_ColMajor} & \quad \text{Nag_Upper} & \quad U D U^T \\
\text{Nag_ColMajor} & \quad \text{Nag_Lower} & \quad L D L^T \\
\text{Nag_RowMajor} & \quad \text{Nag_Upper} & \quad U^T D U \\
\text{Nag_RowMajor} & \quad \text{Nag_Lower} & \quad L^T D L
\end{align*}
\]

where \( U \) (or \( L \)) is a product of permutation and unit upper (lower) triangular matrices, and \( D \) is symmetric 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


5 Arguments

1: \textbf{order} – Nag_OrderType

\textit{Input}

\textit{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.

\textit{Constraint:} order = Nag_RowMajor or Nag_ColMajor.
2: \texttt{uplo} – Nag_UploType \hspace{2cm} \textit{Input}

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

If \texttt{uplo} = Nag_Lower, the lower triangle of \( A \) is stored.

\textit{Constraint:} \texttt{uplo} = Nag_Upper or Nag_Lower.

3: \texttt{n} – Integer \hspace{2cm} \textit{Input}

\textit{On entry:} \( n \), the number of linear equations, i.e., the order of the matrix \( A \).

\textit{Constraint:} \( n \geq 0 \).

4: \texttt{nrhs} – Integer \hspace{2cm} \textit{Input}

\textit{On entry:} \( r \), the number of right-hand sides, i.e., the number of columns of the matrix \( B \).

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

5: \texttt{a[dim]} – Complex \hspace{2cm} \textit{Input/Output}

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

\textit{On entry:} the \( n \) by \( n \) symmetric matrix \( A \).

If \texttt{order} = Nag_ColMajor, \( A_{ij} \) is stored in \( \texttt{a}[(j-1) \times \texttt{pda} + i - 1] \).

If \texttt{order} = Nag_RowMajor, \( A_{ij} \) is stored in \( \texttt{a}[(i-1) \times \texttt{pda} + j - 1] \).

If \texttt{uplo} = Nag_Upper, the upper triangular part of \( A \) must be stored and the elements of the array below the diagonal are not referenced.

If \texttt{uplo} = Nag_Lower, the lower triangular part of \( A \) must be stored and the elements of the array above the diagonal are not referenced.

\textit{On exit:} if \texttt{fail.code} = NE_NOERROR, the block diagonal matrix \( D \) and the multipliers used to obtain the factor \( U \) or \( L \) from the factorization \( A = U D U^T \), \( A = L D L^T \), \( A = U^T D U \) or \( A = L^T D L \) as computed by \texttt{nag_zsytrf (f07nrc)}.

6: \texttt{pda} – Integer \hspace{2cm} \textit{Input}

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

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

7: \texttt{ipiv[dim]} – Integer \hspace{2cm} \textit{Output}

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

\textit{On exit:} details of the interchanges and the block structure of \( D \). More precisely,

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

- if \texttt{uplo} = Nag_Upper and \( \texttt{ipiv}[i-2] = \texttt{ipiv}[i-1] = -l < 0 \), \( \begin{pmatrix} d_{i-1,i-1} & d_{i,i-1} \\ d_{i,i-1} & d_{ii} \end{pmatrix} \) 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 \texttt{uplo} = Nag_Lower and \( \texttt{ipiv}[i-1] = \texttt{ipiv}[i] = -m < 0 \), \( \begin{pmatrix} d_{ii} & d_{i+1,i} \\ d_{i+1,i} & d_{i+1,i+1} \end{pmatrix} \) 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: \( b[dim] \) – Complex

*Note:* the dimension, \( dim \), of the array \( b \) must be at least
\[
\max(1, \text{pdb} \times \text{nrhs}) \quad \text{when} \quad \text{order} = \text{Nag\_ColMajor};
\]
\[
\max(1, n \times \text{pdb}) \quad \text{when} \quad \text{order} = \text{Nag\_RowMajor}.
\]

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

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

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

9: \( \text{pdb} \) – Integer

Input

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

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

10: \( \text{fail} \) – NagError*

Input/Output

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

Constraint: \( n \geq 0 \).

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

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

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

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

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

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

**NE\_INT\_2**

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

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

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

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

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

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

NE_SINGULAR
Element (value) of the diagonal is exactly zero. The factorization has been completed, but the
block diagonal matrix D is exactly singular, so the solution could not be computed.

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

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

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

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

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) and Chapter 11 of Higham (2002) for further
details.

nag_zsysvx (f07nnc) is a comprehensive LAPACK driver that returns forward and backward error
bounds and an estimate of the condition number. Alternatively, nag_complex_sym_lin_solve (f04dhc)
solves $Ax = b$ and returns a forward error bound and condition estimate. nag_complex_sym_lin_solve
(f04dhc) calls nag_zsysv (f07nnc) to solve the equations.

8 Parallelism and Performance
nag_zsysv (f07nnc) is not threaded by NAG in any implementation.
nag_zsysv (f07nnc) 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
The total number of floating-point operations is approximately $\frac{4}{3}n^3 + 8n^2r$, where $r$ is the number of
right-hand sides.

The real analogue of this function is nag_dsylv (f07mac). The complex Hermitian analogue of this
function is nag_zhesv (f07nnc).
10 Example

This example solves the equations

\[ Ax = b, \]

where \( A \) is the complex symmetric matrix

\[
A = \begin{pmatrix}
-0.56 + 0.12i & -1.54 - 2.86i & 5.32 - 1.59i & 3.80 + 0.92i \\
-1.54 - 2.86i & -2.83 - 0.03i & -3.52 + 0.58i & -7.86 - 2.96i \\
5.32 - 1.59i & -3.52 + 0.58i & 8.86 + 1.81i & 5.14 - 0.64i \\
3.80 + 0.92i & -7.86 - 2.96i & 5.14 - 0.64i & -0.39 - 0.71i
\end{pmatrix}
\]

and

\[
b = \begin{pmatrix}
-6.43 + 19.24i \\
-0.49 - 1.47i \\
-48.18 + 66.00i \\
-55.64 + 41.22i
\end{pmatrix}.
\]

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

10.1 Program Text

/* nag_zsysv (f07nnc) Example Program. 
* Copyright 2014 Numerical Algorithms Group. 
* Mark 23, 2011. 
*/
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf07.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 ]
#else
#define A(I, J) a[(I-1)*pda +J-1 ]
#define B(I, J) b[(I-1)*pdb +J-1 ]
#endif

INIT_FAIL(fail);
printf("nag_zsysv (f07nnc) Example Program Results\n\n");

/* Skip heading in data file */
#endif_WIN32
scanf_s("%*[\n");

Mark 25
else
  scanf("%*[\n]");
#endif

#ifdef _WIN32
  scanf_s("%"NAG_IFMT"%"NAG_IFMT"%*[\n]", &n, &nrhs);
#else
  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;
}
#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;
#endif 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
#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
else
  scanf("%*[\n]");
```c
#define _WIN32
#define %*
#define %[^
]

/* Solve the equations Ax = b for x using nag_zsysv (f07nnc). */
int status = 0;
Call nag_zsysv(order, uplo, n, nrhs, a, pda, ipiv, b, pdb, &status);
if (status != NE_NOERROR)
{
    printf("Error from nag_zsysv (f07nnc).\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_zsysv (f07nnc) Example Program Data

nag_zsysv (f07nnc) Example Program Results

10.3 Program Results
```