NAG Library Function Document

nag_zppsv (f07gnc)

1 Purpose

nag_zppsv (f07gnc) computes the solution to a complex system of linear equations

\[ AX = B, \]

where \( A \) is an \( n \) by \( n \) Hermitian positive definite matrix stored in packed format and \( X \) and \( B \) are \( n \) by \( r \) matrices.

2 Specification

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

void nag_zppsv (Nag_OrderType order, Nag_UploType uplo, Integer n,
               Integer nrhs, Complex ap[], Complex b[], Integer pdb, NagError *fail)
```

3 Description

nag_zppsv (f07gnc) uses the Cholesky decomposition to factor \( A \) as \( A = U^H U \) if \( \text{uplo} = \text{Nag_Upper} \) or \( A = LL^H \) if \( \text{uplo} = \text{Nag_Lower} \), where \( U \) is an upper triangular matrix and \( L \) is a lower triangular matrix. The factored form of \( A \) is then used to solve the system of equations \( AX = B \).

4 References


5 Arguments

1: \( \text{order} \) – Nag_OrderType

\( \text{Input} \)

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

Constraint: \text{order} = \text{Nag_RowMajor} or \text{Nag_ColMajor}.

2: \( \text{uplo} \) – Nag_UploType

\( \text{Input} \)

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

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

Constraint: \text{uplo} = \text{Nag_Upper} or \text{Nag_Lower}.

3: \( n \) – Integer

\( \text{Input} \)

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

Constraint: \( n \geq 0 \).
4: nrhs – Integer
Input
On entry: \(r\), the number of right-hand sides, i.e., the number of columns of the matrix \(B\).
Constraint: \(nrhs \geq 0\).

5: ap[dim] – Complex
Input/Output
Note: the dimension, \(dim\), of the array \(ap\) must be at least \(\max(1, n \times (n + 1)/2)\).
On entry: the \(n\) by \(n\) Hermitian matrix \(A\), packed by rows or columns.
The storage of elements \(A_{ij}\) depends on the \(\text{order}\) and \(\text{uplo}\) arguments as follows:
\[
\begin{align*}
&\text{if } \text{order} = \text{Nag\_ColMajor} \text{ and } \text{uplo} = \text{Nag\_Upper}, \\
&A_{ij} \text{ is stored in } ap[(j - 1) \times j/2 + i - 1], \text{ for } i \leq j;
&\text{if } \text{order} = \text{Nag\_ColMajor} \text{ and } \text{uplo} = \text{Nag\_Lower}, \\
&A_{ij} \text{ is stored in } ap[(2n - j) \times (j - 1)/2 + i - 1], \text{ for } i \geq j;
&\text{if } \text{order} = \text{Nag\_RowMajor} \text{ and } \text{uplo} = \text{Nag\_Upper}, \\
&A_{ij} \text{ is stored in } ap[(2n - i) \times (i - 1)/2 + j - 1], \text{ for } i \leq j;
&\text{if } \text{order} = \text{Nag\_RowMajor} \text{ and } \text{uplo} = \text{Nag\_Lower}, \\
&A_{ij} \text{ is stored in } ap[(i - 1) \times i/2 + j - 1], \text{ for } i \geq j.
\end{align*}
\]
On exit: if \(\text{fail.code} = \text{NE\_NOERROR}\), the factor \(U\) or \(L\) from the Cholesky factorization \(A = U^H U\) or \(A = LL^H\), in the same storage format as \(A\).

6: b[dim] – Complex
Input/Output
Note: the dimension, \(dim\), of the array \(b\) must be at least
\[
\max(1, \text{pdb} \times \text{nrhs}) \text{ when } \text{order} = \text{Nag\_ColMajor};
\max(1, n \times \text{pdb}) \text{ when } \text{order} = \text{Nag\_RowMajor}.
\]
The \((i, j)\)th element of the matrix \(B\) is stored in
\[
\begin{align*}
&b[(j - 1) \times \text{pdb} + i - 1] \text{ when } \text{order} = \text{Nag\_ColMajor};
&b[(i - 1) \times \text{pdb} + j - 1] \text{ when } \text{order} = \text{Nag\_RowMajor}.
\end{align*}
\]
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 \(X\).

7: pdb – Integer
Input
On entry: the stride separating row or column elements (depending on the value of \(\text{order}\)) in the array \(b\).
Constraints:
\[
\begin{align*}
&\text{if } \text{order} = \text{Nag\_ColMajor}, \text{ pdb } \geq \max(1, n); \\
&\text{if } \text{order} = \text{Nag\_RowMajor}, \text{ pdb } \geq \max(1, \text{nrhs}).
\end{align*}
\]

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

On entry, n = \langle value\rangle.
Constraint: n \geq 0.

On entry, nrhs = \langle value\rangle.
Constraint: nrhs \geq 0.

On entry, pdb = \langle value\rangle.
Constraint: pdb > 0.

NE_INT_2

On entry, pdb = \langle value\rangle and n = \langle value\rangle.
Constraint: pdb \geq \max(1, n).

On entry, pdb = \langle value\rangle and nrhs = \langle value\rangle.
Constraint: pdb \geq \max(1, 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_MAT_NOT_POS_DEF

The leading minor of order \langle value\rangle of A is not positive definite, so the factorization could not be completed, and the solution has not been computed.

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 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) for further details.

nag_zppsvx (f07gpc) is a comprehensive LAPACK driver that returns forward and backward error bounds and an estimate of the condition number. Alternatively, nag_herm_posdef_packed_lin_solve (f04cec) solves \( Ax = b \) and returns a forward error bound and condition estimate. nag_herm_posdef_packed_lin_solve (f04cec) calls nag_zppsv (f07gnc) to solve the equations.

8 Parallelism and Performance

nag_zppsv (f07gnc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.
nag_zppsv (f07gnc) 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 $\frac{3}{2}n^3 + 8n^2r$, where $r$ is the number of right-hand sides.

The real analogue of this function is nag_dppsv (f07gac).

10 Example

This example solves the equations

$$Ax = b,$$

where $A$ is the Hermitian positive definite matrix

$$A = \begin{pmatrix} 3.23 & 1.51 - 1.92i & 1.90 + 0.84i & 0.42 + 2.50i \\ 1.51 + 1.92i & 3.58 & -0.23 + 1.11i & -1.18 + 1.37i \\ 1.90 - 0.84i & -0.23 - 1.11i & 4.09 & 2.33 - 0.14i \\ 0.42 - 2.50i & -1.18 - 1.37i & 2.33 + 0.14i & 4.29 \end{pmatrix},$$

and

$$b = \begin{pmatrix} 3.93 - 6.14i \\ 6.17 + 9.42i \\ -7.17 - 21.83i \\ 1.99 - 14.38i \end{pmatrix}.$$ 

Details of the Cholesky factorization of $A$ are also output.

10.1 Program Text

/* nag_zppsv (f07gnc) Example Program. 
 * Copyright 2014 Numerical Algorithms Group. 
 * Mark 23, 2011. */

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

int main(void)
{
    /* Scalars */
    Integer exit_status = 0, i, j, n, nrhs, pdb;

    /* Arrays */
    Complex *ap = 0, *b = 0;
    char nag_enum_arg[40];

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

INIT_FAIL(fail);

#ifdef NAG_COLUMN_MAJOR
#define A_UPPER(I, J) ap[(J*(J-1)/2 + I - 1)
#define A_LOWER(I, J) ap[(2*n-J)*(J-1)/2 + I - 1)
#define B(I, J) b[(J-1)*pdb + I - 1]

order = Nag_ColMajor;
#else
#define A_LOWER(I, J) ap[(I*(I-1)/2 + J - 1)
#define A_UPPER(I, J) ap[(2*n-I)*(I-1)/2 + J - 1)
#define B(I, J) b[(I-1)*pdb + J - 1]

order = Nag_RowMajor;
#endif

printf("nag_zppsv (f07gnc) 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
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
uplo = (Nag_UploType) nag_enum_name_to_value(nag_enum_arg);

#ifdef NAG_COLUMN_MAJOR
pdb = n;
#else
pdb = nrhs;
#endif

/* Read the upper or lower 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)
#else
    for (i = 1; i <= n; ++i)
        for (j = i; j <= n; ++j)
#endif
        #ifdef _WIN32
        scanf_s(" ( %lf , %lf )", &A_UPPER(i, j).re, &A_UPPER(i, j).im);
#else
        scanf(" ( %lf , %lf )", &A_UPPER(i, j).re, &A_UPPER(i, j).im);
#endif
    else if (uplo == Nag_Lower)
        for (i = 1; i <= n; ++i)
            for (j = 1; j <= i; ++j)
/* 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_zppsv (f07gnc). */
if (fail.code != NE_NOERROR)
  { printf("Error from nag_zppsv (f07gnc).\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");
  }

/* Print details of factorization using * nag_pack_complx_mat_print_comp (x04ddc). */
printf("\n");
flush(stdout);
nag_pack_complx_mat_print_comp(order, uplo, Nag_NonUnitDiag, n, ap,
                               Nag_BracketForm, "%7.4f", "Cholesky factor",
                               Nag_IntegerLabels, 0, Nag_IntegerLabels, 0,
                               80, 0, 0, &fail);
if (fail.code != NE_NOERROR)
  { printf("Error from nag_pack_complx_mat_print_comp (x04ddc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }
END:
NAG_FREE(ap);
NAG_FREE(b);
return exit_status;
}
#define A_UPPER
#define A_LOWER
#define B

f07gnc
NAG Library Manual

f07gnc.6 Mark 25
10.2 Program Data

nag_zppsv (f07gnc) Example Program Data

\[
\begin{array}{c}
4 \quad 1 : n, \text{nrhs} \\
\text{Nag_Upper} : \text{uplo} \\
(3.23, \ 0.00) \ (1.51, \ -1.92) \ (1.90, \ 0.84) \ (0.42, \ 2.50) \\
(3.58, \ 0.00) \ (-0.23, \ 1.11) \ (-1.18, \ 1.37) \\
(4.09, \ 0.00) \ (2.33, \ -0.14) \\
(4.29, \ 0.00) \ : \text{matrix } A \\
(3.93, \ -6.14) \ (6.17, \ 9.42) \ (-7.17, \ -21.83) \ (1.99, \ -14.38) \ : \text{vector } b
\end{array}
\]

10.3 Program Results

nag_zppsv (f07gnc) Example Program Results

Solution
\[
\begin{array}{c}
(1.0000, \ -1.0000) \\
(-0.0000, \ 3.0000) \\
(-4.0000, \ -5.0000) \\
(2.0000, \ 1.0000)
\end{array}
\]

Cholesky factor
\[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
1 \ (1.7972, \ 0.0000) & (0.8402, -1.0683) & (1.0572, \ 0.4674) & (0.2337, \ 1.3910) \\
2 & 4 & (1.3164, \ 0.0000) & (-0.4702, -0.3131) & (0.0834, -0.0368) \\
3 & & 3 & (1.5604, \ 0.0000) & (0.9360, -0.9900) \\
4 & & & 4 & (0.6603, \ 0.0000)
\end{array}
\]