NAG Library Function Document

nag_dppsv (f07gac)

1 Purpose

nag_dppsv (f07gac) computes the solution to a real system of linear equations

\[ AX = B, \]

where \( A \) is an \( n \) by \( n \) symmetric 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_dppsv (Nag_OrderType order, Nag_UploType uplo, Integer n, 
                Integer nrhs, double ap[], double b[], Integer pdb, NagError *fail)
```

3 Description

nag_dppsv (f07gac) uses the Cholesky decomposition to factor \( A \) as \( A = U^T U \) if \( \text{uplo} = \text{Nag}_\text{Upper} \) or \( A = LL^T \) if \( \text{uplo} = \text{Nag}_\text{Lower} \), 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

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:  \textbf{order} – Nag_OrderType 

   \textit{Input}

   \textit{On entry:} the \textit{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 \textit{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:} \textit{order} = Nag_RowMajor or Nag_ColMajor.

2:  \textbf{uplo} – Nag_UploType 

   \textit{Input}

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

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

3:  \textbf{n} – Integer 

   \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: 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: \( \text{ap}[\text{dim}] \) – double

Input/Output

Note: the dimension, \( \text{dim} \), of the array \( \text{ap} \) must be at least \( \max(1, n \times (n + 1)/2) \).

On entry: the \( n \) by \( n \) symmetric 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 and uplo} = \text{Nag_Upper}, \\
& \text{\( A_{ij} \) is stored in \( \text{ap}[(j - 1) \times j/2 + i - 1] \), for } i \leq j; \\
\text{if } \text{order} &= \text{Nag_ColMajor and uplo} = \text{Nag_Lower}, \\
& \text{\( A_{ij} \) is stored in \( \text{ap}[(2n - j) \times (j - 1)/2 + i - 1] \), for } i \geq j; \\
\text{if } \text{order} &= \text{Nag_RowMajor and uplo} = \text{Nag_Upper}, \\
& \text{\( A_{ij} \) is stored in \( \text{ap}[(2n - i) \times (i - 1)/2 + j - 1] \), for } i \leq j; \\
\text{if } \text{order} &= \text{Nag_RowMajor and uplo} = \text{Nag_Lower}, \\
& \text{\( A_{ij} \) is stored in \( \text{ap}[(i - 1) \times i/2 + j - 1] \), for } i \geq j.
\end{align*}
\]

On exit: if fail.code = NE_NOERROR, the factor \( U \) or \( L \) from the Cholesky factorization \( A = U^T U \) or \( A = LL^T \), in the same storage format as \( A \).

6: \( \text{b}[\text{dim}] \) – double

Input/Output

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

\[
\begin{align*}
\max(1, \text{pdb} \times \text{nrhs}) \text{ when } \text{order} = \text{Nag_ColMajor}; \\
\max(1, n \times \text{pdb}) \text{ when } \text{order} = \text{Nag_RowMajor}.
\end{align*}
\]

The \((i,j)\)th element of the matrix \( B \) is stored in

\[
\begin{align*}
\text{b}[(j - 1) \times \text{pdb} + i - 1] \text{ when } \text{order} = \text{Nag_ColMajor}; \\
\text{b}[(i - 1) \times \text{pdb} + j - 1] \text{ when } \text{order} = \text{Nag_RowMajor}.
\end{align*}
\]

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

On exit: if fail.code = 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 \( \text{b} \).

Constraints:

\[
\begin{align*}
\text{if } \text{order} &= \text{Nag_ColMajor, pdb} \geq \max(1, n); \\
\text{if } \text{order} &= \text{Nag_RowMajor, pdb} \geq \max(1, \text{nrhs}).
\end{align*}
\]

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 \langle value \rangle had an illegal value.
NE_INT
On entry, n = ⟨value⟩.
Constraint: n ≥ 0.

On entry, nrhs = ⟨value⟩.
Constraint: nrhs ≥ 0.

On entry, pdb = ⟨value⟩.
Constraint: pdb > 0.

NE_INT_2
On entry, pdb = ⟨value⟩ and n = ⟨value⟩.
Constraint: pdb ≥ max(1, n).

On entry, pdb = ⟨value⟩ and nrhs = ⟨value⟩.
Constraint: pdb ≥ 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 ⟨value⟩ 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, ̂x, satisfies an equation of the form

\((A + E) ̂x = b,\)

where

\[\|E\|_1 = O(\epsilon)\|A\|_1\]

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

\[\|\hat{x} - x\|_1 \leq \kappa(A)\|E\|_1\|A\|^{-1}_1\]

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_dppsvx (f07gbc) is a comprehensive LAPACK driver that returns forward and backward error
bounds and an estimate of the condition number. Alternatively, nag_real_sym_posdef_packed_lin_solve
(f04bec) solves \(Ax = b\) and returns a forward error bound and condition estimate.
nag_real_sym_posdef_packed_lin_solve (f04bec) calls nag_dppsv (f07gac) to solve the equations.

8 Parallelism and Performance
nag_dppsv (f07gac) is threaded by NAG for parallel execution in multithreaded implementations of the
NAG Library.
nag_dppsv (f07gac) 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 $\frac{1}{3}n^3 + 2n^2r$, where $r$ is the number of right-hand sides.

The complex analogue of this function is nag_zppsv (f07gnc).

10 Example

This example solves the equations

\[ Ax = b, \]

where $A$ is the symmetric positive definite matrix

\[
A = \begin{pmatrix}
4.16 & -3.12 & 0.56 & -0.10 \\
-3.12 & 5.03 & -0.83 & 1.18 \\
0.56 & -0.83 & 0.76 & 0.34 \\
-0.10 & 1.18 & 0.34 & 1.18
\end{pmatrix}
\text{ and } \quad b = \begin{pmatrix}
8.70 \\
-13.35 \\
1.89 \\
-4.14
\end{pmatrix}.
\]

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

10.1 Program Text

/* nag_dppsv (f07gac) 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;
    Nag_OrderType order;

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

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

    #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]
    #endif

#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

INIT_FAIL(fail);

printf("nag_dppsv (f07gac) Example Program Results\n\n");

/* Skip heading in data file */
#endif _WIN32
scanf_s("%*[\n"]);
#else
scanf("%*[\n"]);
#endif

define _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;
}
define _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 (!(ap = NAG_ALLOC(n*(n+1)/2, double)) ||
  !(b = NAG_ALLOC(n*nrhs, double)))
{
  printf("Allocation failure\n");
  exit_status = -1;
  goto END;
}
define 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)
    #define _WIN32
      for (j = i; j <= n; ++j) scanf_s("%lf", &A_UPPER(i, j));
    #else
      for (j = i; j <= n; ++j) scanf("%lf", &A_UPPER(i, j));
    #endif
  else if (uplo == Nag_Lower)
    for (i = 1; i <= n; ++i)
      #define _WIN32
        for (j = i; j <= n; ++j) scanf_s("%lf", &A_LOWER(i, j));
      #else
        for (j = i; j <= n; ++j) scanf("%lf", &A_LOWER(i, j));
      #endif
#define _WIN32
scanf_s("%*[\n"]);
#else
scanf("%*[\n"]);
#endif

Mark 25
f07gac.5

f07 – Linear Equations (LAPACK)
/* Read b from data file */
for (i = 1; i <= n; ++i)
#endif _WIN32
    for (j = 1; j <= nrhs; ++j) scanf_s("%lf", &B(i, j));
#else
    for (j = 1; j <= nrhs; ++j) scanf("%lf", &B(i, j));
#endif
#ifndef _WIN32
    scanf_s("%*[\n]");
#else
    scanf("%*[\n]");
#endif
/* Solve the equations Ax = b for x using nag_dppsv (f07gac). */
nag_dppsv(order, uplo, n, nrhs, ap, b, pdb, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dppsv (f07gac).\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("%11.4f%s", B(i, j), j%7 == 0?"\n":" ");
    printf("\n");
}
/* Print details of factorization using nag_pack_real_mat_print (x04ccc). */
printf("\n");
fflush(stdout);
nag_pack_real_mat_print(order, uplo, Nag_NonUnitDiag, n, ap,
    "Cholesky factor", 0, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_pack_real_mat_print (x04ccc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
END:
NAG_FREE(ap);
NAG_FREE(b);
return exit_status;
#endif A_UPPER
#endif A_LOWER
#endif B

10.2 Program Data
nag_dppsv (f07gac) Example Program Data
4
4.16 -3.12 0.56 -0.10
   5.03 -0.83 1.18
   0.76 0.34
   1.18 : matrix A
   8.70 -13.35 1.89 -4.14 : vector b
10.3 Program Results

nag_dppsv (f07gac) Example Program Results

Solution

1.0000
-1.0000
2.0000
-3.0000

Cholesky factor

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.0396</td>
<td>-1.5297</td>
<td>0.2746</td>
<td>-0.0490</td>
</tr>
<tr>
<td>2</td>
<td>1.6401</td>
<td>-0.2500</td>
<td>0.6737</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>0.7887</td>
<td>0.6617</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>0.5347</td>
<td></td>
</tr>
</tbody>
</table>