NAG Library Function Document

nag_dspsv (f07pac)

1 Purpose

nag_dspsv (f07pac) computes the solution to a real system of linear equations

$$AX = B,$$

where A is an n by n symmetric matrix stored in packed format and X and B are n by r matrices.

2 Specification

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

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

3 Description

nag_dspsv (f07pac) uses the diagonal pivoting method to factor A as $A = UDU^{\mathrm{T}}$ if **uplo** $=$ Nag_Upper or $A = LDL^{\mathrm{T}}$ if **uplo** $=$ Nag_Lower, where U (or L) is a product of permutation and unit upper (lower) triangular matrices, 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

Anderson E, Bai Z, Bischof C, Blackford S, Demmel J, Dongarra J J, Du Croz J J, Greenbaum A, Hammarling S, McKenney A and Sorensen D (1999) *LAPACK Users' Guide* (3rd Edition) SIAM, Philadelphia http://www.netlib.org/lapack/lug

Golub G H and Van Loan C F (1996) *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

Higham N J (2002) *Accuracy and Stability of Numerical Algorithms* (2nd Edition) SIAM, Philadelphia

5 Arguments

1: **order** – Nag_OrderType *Input*

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.

Constraint: **order** $=$ Nag_RowMajor or Nag_ColMajor.

2: **uplo** – Nag_UploType *Input*

On entry: if **uplo** $=$ Nag_Upper, the upper triangle of A is stored.

If **uplo** $=$ Nag_Lower, the lower triangle of A is stored.

Constraint: **uplo** $=$ Nag_Upper or Nag_Lower.

3: **n** – Integer *Input*

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

Constraint: $\mathbf{n} \geq 0$.

4: **nrhs** – Integer *Input*

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

Constraint: $\mathbf{nrhs} \geq 0$.

5: **ap**[dim] – double *Input/Output*

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

On entry: the n by n symmetric matrix A, packed by rows or columns.

The storage of elements A_{ij} depends on the **order** and **uplo** arguments as follows:

if **order** = Nag_ColMajor and **uplo** = Nag_Upper,
A_{ij} is stored in $\mathbf{ap}[(j-1) \times j/2 + i - 1]$, for $i \leq j$;
if **order** = Nag_ColMajor and **uplo** = Nag_Lower,
A_{ij} is stored in $\mathbf{ap}[(2n-j) \times (j-1)/2 + i - 1]$, for $i \geq j$;
if **order** = Nag_RowMajor and **uplo** = Nag_Upper,
A_{ij} is stored in $\mathbf{ap}[(2n-i) \times (i-1)/2 + j - 1]$, for $i \leq j$;
if **order** = Nag_RowMajor and **uplo** = Nag_Lower,
A_{ij} is stored in $\mathbf{ap}[(i-1) \times i/2 + j - 1]$, for $i \geq j$.

On exit: the block diagonal matrix D and the multipliers used to obtain the factor U or L from the factorization $A = UDU^{\mathrm{T}}$ or $A = LDL^{\mathrm{T}}$ as computed by nag_dsptrf (f07pdc), stored as a packed triangular matrix in the same storage format as A.

6: **ipiv**[**n**] – Integer *Output*

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

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

if **uplo** = Nag_Upper and $\mathbf{ipiv}[i-2] = \mathbf{ipiv}[i-1] = -l < 0$, $\begin{pmatrix} d_{i-1,i-1} & \bar{d}_{i,i-1} \\ \bar{d}_{i,i-1} & d_{ii} \end{pmatrix}$ is a 2 by 2 pivot block and the $(i-1)$th row and column of A were interchanged with the lth row and column;

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

7: **b**[dim] – double *Input/Output*

Note: the dimension, *dim*, of the array **b** must be at least

$\max(1, \mathbf{pdb} \times \mathbf{nrhs})$ when **order** = Nag_ColMajor;
$\max(1, \mathbf{n} \times \mathbf{pdb})$ when **order** = Nag_RowMajor.

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

$\mathbf{b}[(j-1) \times \mathbf{pdb} + i - 1]$ when **order** = Nag_ColMajor;
$\mathbf{b}[(i-1) \times \mathbf{pdb} + j - 1]$ when **order** = Nag_RowMajor.

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.

8: **pdb** – Integer *Input*

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

Constraints:

if **order** = Nag_ColMajor, **pdb** $\geq \max(1, \mathbf{n})$;
if **order** = Nag_RowMajor, **pdb** $\geq \max(1, \mathbf{nrhs})$.

9: **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** = $\langle value \rangle$.
Constraint: **n** ≥ 0.

On entry, **nrhs** = $\langle value \rangle$.
Constraint: **nrhs** ≥ 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** $\geq \max(1, \mathbf{n})$.

On entry, **pdb** = $\langle value \rangle$ and **nrhs** = $\langle value \rangle$.
Constraint: **pdb** $\geq \max(1, \mathbf{nrhs})$.

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 $\langle value \rangle$ 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 ϵ 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 $\kappa(A) = \left\|A^{-1}\right\|_1\|A\|_1$, 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_dspsvx (f07pbc) is a comprehensive LAPACK driver that returns forward and backward error bounds and an estimate of the condition number. Alternatively, nag_real_sym_packed_lin_solve (f04bjc) solves $AX = B$ and returns a forward error bound and condition estimate. nag_real_sym_packed_lin_solve (f04bjc) calls nag_dspsv (f07pac) to solve the equations.

8 Parallelism and Performance

nag_dspsv (f07pac) is not threaded by NAG in any implementation.

nag_dspsv (f07pac) 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 analogues of nag_dspsv (f07pac) are nag_zhpsv (f07pnc) for Hermitian matrices, and nag_zspsv (f07qnc) for symmetric matrices.

10 Example

This example solves the equations

$$Ax = b,$$

where A is the symmetric matrix

$$A = \begin{pmatrix} -1.81 & 2.06 & 0.63 & -1.15 \\ 2.06 & 1.15 & 1.87 & 4.20 \\ 0.63 & 1.87 & -0.21 & 3.87 \\ -1.15 & 4.20 & 3.87 & 2.07 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} 0.96 \\ 6.07 \\ 8.38 \\ 9.50 \end{pmatrix}.$$

Details of the factorization of A are also output.

10.1 Program Text

```
/* nag_dspsv (f07pac) 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, pdb;

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

  /* Nag Types */
  NagError      fail;
  Nag_OrderType order;
  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]
#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_dspsv (f07pac) 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
  /* 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)) ||
      !(ipiv = NAG_ALLOC(n, Integer)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

#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)
#ifdef _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
    for (i = 1; i <= n; ++i)
#ifdef _WIN32
      for (j = 1; j <= i; ++j) scanf_s("%lf", &A_LOWER(i, j));
#else
      for (j = 1; j <= i; ++j) scanf("%lf", &A_LOWER(i, j));
#endif
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif

  /* Read b from data file */
  for (i = 1; i <= n; ++i)
#ifdef _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
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif

  /* Solve the equations Ax = b for x using nag_dspsv (f07pac). */
  nag_dspsv(order, uplo, n, nrhs, ap, ipiv, b, pdb, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_dspsv (f07pac).\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(" %10.4f%s", B(i, j), j%7 == 0?"\n":"");
      printf("\n");
    }
```

```
 END:
  NAG_FREE(ap);
  NAG_FREE(b);
  NAG_FREE(ipiv);

  return exit_status;
}
#undef A_UPPER
#undef A_LOWER
#undef B
```

10.2 Program Data

```
nag_dspsv (f07pac) Example Program Data
  4      1                   : n, nrhs
  Nag_Lower                  : uplo
 -1.81
  2.06   1.15
  0.63   1.87  -0.21
 -1.15   4.20   3.87   2.07 : matrix A
  0.96   6.07   8.38   9.50 : vector b
```

10.3 Program Results

```
nag_dspsv (f07pac) Example Program Results

   Solution
    -5.0000
    -2.0000
     1.0000
     4.0000
```