NAG Library Function Document

nag_dposv (f07fac)

1 Purpose

nag_dposv (f07fac) computes the solution to a real system of linear equations

$$AX = B,$$

where A is an n by n symmetric positive definite matrix and X and B are n by r matrices.

2 Specification

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

3 Description

nag_dposv (f07fac) uses the Cholesky decomposition to factor A as $A = U^{\mathrm{T}}U$ if **uplo** $=$ Nag_Upper or $A = LL^{\mathrm{T}}$ if **uplo** $=$ 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

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

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: **n** ≥ 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** ≥ 0.

5: **a**$[dim]$ – double *Input/Output*

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

On entry: the n by n symmetric matrix A.

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

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

If **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 **uplo** = 'Nag_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 **fail.code** = NE_NOERROR, the factor U or L from the Cholesky factorization $A = U^{\mathrm{T}}U$ or $A = LL^{\mathrm{T}}$.

6: **pda** – Integer *Input*

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

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

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

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

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, **pda** $= \langle value \rangle$.
Constraint: **pda** > 0.

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

NE_INT_2

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

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.

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.

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) for further details.

nag_dposvx (f07fbc) is a comprehensive LAPACK driver that returns forward and backward error bounds and an estimate of the condition number. Alternatively, nag_real_sym_posdef_lin_solve (f04bdc) solves $Ax = b$ and returns a forward error bound and condition estimate. nag_real_sym_posdef_lin_solve (f04bdc) calls nag_dposv (f07fac) to solve the equations.

8 Parallelism and Performance

nag_dposv (f07fac) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_dposv (f07fac) 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 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_zposv (f07fnc).

10 Example

This example solves the equations

$$Ax = b,$$

where A is the symmetric positive definite matrix

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

Details of the Cholesky factorization of A are also output.

10.1 Program Text

```
/* nag_dposv (f07fac) Example Program.
 *
 * Copyright 2008 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, pda, pdb;
  /* Arrays */
  double        *a = 0, *b = 0;

  /* Nag Types */
  NagError      fail;
  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_dposv (f07fac) Example Program Results\n\n");

  /* Skip heading in data file */
  scanf("%*[^\n]");
  scanf("%ld%ld%*[^\n]", &n, &nrhs);
  if (n < 0 || nrhs < 0)
    {
      printf("Invalid n or nrhs\n");
      exit_status = 1;
      goto END;
    }

  /* Allocate memory */
  if (!(a = NAG_ALLOC(n * n, double)) ||
      !(b = NAG_ALLOC(n *nrhs, double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

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

  /* Read the upper triangular part of A from data file */
  for (i = 1; i <= n; ++i)
    for (j = i; j <= n; ++j) scanf("%lf", &A(i, j));
  scanf("%*[^\n]");

  /* Read b from data file */
  for (i = 1; i <= n; ++i)
    for (j = 1; j <= nrhs; ++j) scanf("%lf", &B(i, j));
  scanf("%*[^\n]");

  /* Solve the equations Ax = b for x using nag_dposv (f07fac). */
  nag_dposv(order, Nag_Upper, n, nrhs, a, pda, b, pdb, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_dposv (f07fac).\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_gen_real_mat_print (x04cac). */
  printf("\n");
  fflush(stdout);
  nag_gen_real_mat_print(order, Nag_UpperMatrix, Nag_NonUnitDiag, n, n, a, pda,
                         "Cholesky factor U", 0, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_gen_real_mat_print (x04cac).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

 END:
  NAG_FREE(a);
```

```
  NAG_FREE(b);

  return exit_status;
}

#undef A
#undef B
```

10.2 Program Data

```
nag_dposv (f07fac) Example Program Data
  4      1                   : n, nrhs
  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_dposv (f07fac) Example Program Results

Solution
     1.0000
    -1.0000
     2.0000
    -3.0000

 Cholesky factor U
           1          2          3          4
 1      2.0396    -1.5297     0.2746    -0.0490
 2                 1.6401    -0.2500     0.6737
 3                            0.7887     0.6617
 4                                       0.5347
```