NAG Library Function Document

nag_dpftrs (f07wec)

1 Purpose

nag_dpftrs (f07wec) solves a real symmetric positive definite system of linear equations with multiple right-hand sides,

$$AX = B,$$

using the Cholesky factorization computed by nag_dpftrf (f07wdc) stored in Rectangular Full Packed (RFP) format.

2 Specification

```
#include <nag.h>
#include <nagf07.h>
void nag_dpftrs (Nag_OrderType order, Nag_RFP_Store transr,
     Nag_UploType uplo, Integer n, Integer nrhs, const double ar[],
     double b[], Integer pdb, NagError *fail)
```

3 Description

nag_dpftrs (f07wec) is used to solve a real symmetric positive definite system of linear equations $AX = B$, the function must be preceded by a call to nag_dpftrf (f07wdc) which computes the Cholesky factorization of A, stored in RFP format. The RFP storage format is described in Section 3.3.3 in the f07 Chapter Introduction. The solution X is computed by forward and backward substitution.

If **uplo** = Nag_Upper, $A = U^{\mathrm{T}}U$, where U is upper triangular; the solution X is computed by solving $U^{\mathrm{T}}Y = B$ and then $UX = Y$.

If **uplo** = Nag_Lower, $A = LL^{\mathrm{T}}$, where L is lower triangular; the solution X is computed by solving $LY = B$ and then $L^{\mathrm{T}}X = Y$.

4 References

Gustavson F G, Waśniewski J, Dongarra J J and Langou J (2010) Rectangular full packed format for Cholesky's algorithm: factorization, solution, and inversion *ACM Trans. Math. Software* **37, 2**

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 2.3.1.3 in How to Use the NAG Library and its Documentation for a more detailed explanation of the use of this argument.

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

2: **transr** – Nag_RFP_Store *Input*

On entry: specifies whether the RFP representation of A is normal or transposed.

transr = Nag_RFP_Normal
The matrix A is stored in normal RFP format.

transr = Nag_RFP_Trans
The matrix A is stored in transposed RFP format.

Constraint: **transr** = Nag_RFP_Normal or Nag_RFP_Trans.

3: **uplo** – Nag_UploType *Input*

On entry: specifies how A has been factorized.

uplo = Nag_Upper
$A = U^{\mathrm{T}}U$, where U is upper triangular.

uplo = Nag_Lower
$A = LL^{\mathrm{T}}$, where L is lower triangular.

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

4: **n** – Integer *Input*

On entry: n, the order of the matrix A.

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

5: **nrhs** – Integer *Input*

On entry: r, the number of right-hand sides.

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

6: **ar**$[\mathbf{n} \times (\mathbf{n}+\mathbf{1})/\mathbf{2}]$ – const double *Input*

On entry: the Cholesky factorization of A stored in RFP format, as returned by nag_dpftrf (f07wdc).

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

9: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 3.2.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_INT

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

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

NE_INT_2

On entry, **pdb** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: **pdb** $\geq \max(1, \mathbf{n})$.

On entry, **pdb** = ⟨*value*⟩ and **nrhs** = ⟨*value*⟩.
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 How to Use the NAG Library and its Documentation 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 How to Use the NAG Library and its Documentation for further information.

7 Accuracy

For each right-hand side vector b, the computed solution x is the exact solution of a perturbed system of equations $(A + E)x = b$, where

if **uplo** = Nag_Upper, $|E| \leq c(n)\epsilon|U^{\mathrm{T}}||U|$;

if **uplo** = Nag_Lower, $|E| \leq c(n)\epsilon|L||L^{\mathrm{T}}|$,

$c(n)$ is a modest linear function of n, and ϵ is the ***machine precision***.

If $\hat{x}$ is the true solution, then the computed solution x satisfies a forward error bound of the form

$$\frac{\|x - \hat{x}\|_\infty}{\|x\|_\infty} \leq c(n)\,\mathrm{cond}(A, x)\epsilon$$

where $\mathrm{cond}(A, x) = \left\| |A^{-1}||A||x| \right\|_\infty / \|x\|_\infty \leq \mathrm{cond}(A) = \left\| |A^{-1}||A| \right\|_\infty \leq \kappa_\infty(A)$ and $\kappa_\infty(A)$ is the condition number when using the ∞-norm.

Note that $\mathrm{cond}(A, x)$ can be much smaller than $\mathrm{cond}(A)$.

8 Parallelism and Performance

nag_dpftrs (f07wec) 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' Notefor your implementation for any additional implementation-specific information.

9 Further Comments

The total number of floating-point operations is approximately $2n^2r$.

The complex analogue of this function is nag_zpftrs (f07wsc).

10 Example

This example solves the system of equations $AX = B$, where

$$A = \begin{pmatrix} 4.16 & -3.12 & 0.56 & -0.10 \\ -3.12 & 5.03 & -0.83 & 1.18 \\ 0.56 & -0.83 & 0.76 & 0.34 \\ -0.10 & 1.18 & 0.34 & 1.18 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 8.70 & 8.30 \\ -13.35 & 2.13 \\ 1.89 & 1.61 \\ -4.14 & 5.00 \end{pmatrix}.$$

Here A is symmetric positive definite, stored in RFP format, and must first be factorized by nag_dpftrf (f07wdc).

10.1 Program Text

```
/* nag_dpftrs (f07wec) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

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

int main(void)
{
  /*  Scalars */
  Integer exit_status = 0;
  Integer i, j, k, lar1, lar2, lenar, n, nrhs, pdar, pdb, q;
  /*  Arrays */
  double *ar = 0, *b = 0;
  char nag_enum_arg[40];
  /* NAG types */
  Nag_RFP_Store transr;
  Nag_UploType uplo;
  Nag_OrderType order;
  NagError fail;

#ifdef NAG_COLUMN_MAJOR
  order = Nag_ColMajor;
#define AR(I,J) ar[J*pdar + I]
#define B(I, J) b[J*pdb + I]
#else
  order = Nag_RowMajor;
#define AR(I,J) ar[I*pdar + J]
#define B(I, J) b[I*pdb + J]
```

```
#endif

  INIT_FAIL(fail);
  printf("nag_dpftrs (f07wec) Example Program Results\n");
  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif
#ifdef _WIN32
  scanf_s("%" NAG_IFMT "%" NAG_IFMT "", &n, &nrhs);
#else
  scanf("%" NAG_IFMT "%" NAG_IFMT "", &n, &nrhs);
#endif
#ifdef _WIN32
  scanf_s("%39s", nag_enum_arg, (unsigned)_countof(nag_enum_arg));
#else
  scanf("%39s", nag_enum_arg);
#endif
  uplo = (Nag_UploType) nag_enum_name_to_value(nag_enum_arg);
#ifdef _WIN32
  scanf_s("%39s%*[^\n] ", nag_enum_arg, (unsigned)_countof(nag_enum_arg));
#else
  scanf("%39s%*[^\n] ", nag_enum_arg);
#endif
  transr = (Nag_RFP_Store) nag_enum_name_to_value(nag_enum_arg);

  lenar = (n * (n + 1)) / 2;
  if (!(ar = NAG_ALLOC(lenar, double)) || !(b = NAG_ALLOC(n * nrhs, double)))
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }

  /* Setup dimensions for RFP array ar and for b. */
  k = n / 2;
  q = n - k;
  if (transr == Nag_RFP_Normal) {
    lar1 = 2 * k + 1;
    lar2 = q;
  }
  else {
    lar1 = q;
    lar2 = 2 * k + 1;
  }
  if (order == Nag_RowMajor) {
    pdar = lar2;
    pdb = nrhs;
  }
  else {
    pdar = lar1;
    pdb = n;
  }
  /* Read matrix into RFP array ar. */
  for (i = 0; i < lar1; i++) {
    for (j = 0; j < lar2; j++) {
#ifdef _WIN32
      scanf_s("%lf ", &AR(i, j));
#else
      scanf("%lf ", &AR(i, j));
#endif
    }
  }
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif
  /* Read B */
```

```
  for (i = 0; i < n; i++)
    for (j = 0; j < nrhs; j++)
#ifdef _WIN32
      scanf_s("%lf", &B(i, j));
#else
      scanf("%lf", &B(i, j));
#endif

  /* Factorize A using nag_dpftrf (f07wdc) which peforms a Cholesky
   * factorization of a real symmetric positive definite matrix in
   * Rectangular Full Packed format
   */
  nag_dpftrf(order, transr, uplo, n, ar, &fail);
  printf("\n");
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_dpftrf (f07wdc)\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }

  /* Compute solution of Ax = B using nag_dpftrs (f07wec) which
   * Solves a real symmetric positive definite system of linear equations,
   * for a factorized matrix in Rectangular Full Packed format
   */
  nag_dpftrs(order, transr, uplo, n, nrhs, ar, b, pdb, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_dpftrs (f07wec)\n%s\n", fail.message);
    exit_status = 2;
    goto END;
  }

  /* nag_gen_real_mat_print (x04cac).
   * Print real general matrix (easy-to-use)
   */
  fflush(stdout);
  nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, nrhs,
                         b, pdb, "Solution(s)", 0, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_gen_real_mat_print (x04cac)\n%s\n", fail.message);
    exit_status = 3;
  }

END:
  NAG_FREE(ar);
  NAG_FREE(b);
  return exit_status;
}
```

10.2 Program Data

```
nag_dpftrs (f07wec) Example Program Data
  4     2   Nag_Lower  Nag_RFP_Normal    : n, nrhs, uplo, transr

  0.76    0.34
  4.16    1.18
 -3.12    5.03
  0.56   -0.83
 -0.10    1.18                          : ar[]

  8.70    8.30
-13.35    2.13
  1.89    1.61
 -4.14    5.00                          : b
```

10.3 Program Results

```
nag_dpftrs (f07wec) Example Program Results

 Solution(s)
              1          2
 1       1.0000     4.0000
 2      -1.0000     3.0000
 3       2.0000     2.0000
 4      -3.0000     1.0000
```