NAG C Library Function Document

nag_dpotrs (f07fec)

1 Purpose
nag_dpotrs (f07fec) solves a real symmetric positive-definite system of linear equations with multiple right-hand sides, \( AX = B \), where \( A \) has been factorized by nag_dpotrf (f07fdc).

2 Specification
void nag_dpotrs (Nag_OrderType order, Nag_UploType uplo, Integer n, Integer nrhs, const double a[], Integer pda, double b[], Integer pdb, NagError *fail)

3 Description
To solve a real symmetric positive-definite system of linear equations \( AX = B \), this function must be preceded by a call to nag_dpotrf (f07fdc) which computes the Cholesky factorization of \( A \). The solution \( X \) is computed by forward and backward substitution.

If \( uplo = \text{Nag}_\text{Upper} \), \( A = U^T U \), where \( U \) is upper triangular; the solution \( X \) is computed by solving \( U^T Y = B \) and then \( UX = Y \).

If \( uplo = \text{Nag}_\text{Lower} \), \( A = LL^T \), where \( L \) is lower triangular; the solution \( X \) is computed by solving \( LY = B \) and then \( L^T X = Y \).

4 References

5 Parameters
1: order – Nag_OrderType
   
   On entry: the order parameter 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 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.
   
   Constraint: order = Nag_RowMajor or Nag_ColMajor.

2: uplo – Nag_UploType
   
   On entry: indicates whether \( A \) has been factorized as \( U^T U \) or \( LL^T \), as follows:
   
   if \( uplo = \text{Nag}_\text{Upper} \), then \( A = U^T U \), where \( U \) is upper triangular;
   
   if \( uplo = \text{Nag}_\text{Lower} \), then \( A = LL^T \), where \( L \) is lower triangular.
   
   Constraint: uplo = Nag_Upper or Nag_Lower.

3: n – Integer
   
   On entry: \( n \), the order of the matrix \( A \).
   
   Constraint: \( n \geq 0 \).

4: nrhs – Integer
   
   On entry: \( r \), the number of right-hand sides.
   
   Constraint: \( nrhs \geq 0 \).
5: \(a[\text{dim}]\) – const double

**Input**

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

**On entry:** the Cholesky factor of \(A\), as returned by nag_dpotrf (f07fdc).

6: \(pda\) – Integer

**Input**

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

**Constraint:** \(pda \geq \max(1, n)\).

7: \(b[\text{dim}]\) – double

**Input/Output**

**Note:** the dimension, \(\text{dim}\), of the array \(b\) must be at least \(\max(1, pdb \times nrhs)\) when \(\text{order} = \text{Nag\_ColMajor}\) and at least \(\max(1, pdb \times n)\) when \(\text{order} = \text{Nag\_RowMajor}\).

If \(\text{order} = \text{Nag\_ColMajor}\), the \((i, j)\)th element of the matrix \(B\) is stored in \(b[(j - 1) \times pdb + i - 1]\) and if \(\text{order} = \text{Nag\_RowMajor}\), the \((i, j)\)th element of the matrix \(B\) is stored in \(b[i - 1] \times pdb + j - 1]\).

**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 matrix row or column elements (depending on the value of \(\text{order}\)) in the array \(b\).

**Constraints:**

- if \(\text{order} = \text{Nag\_ColMajor}\), \(pdb \geq \max(1, n)\);
- if \(\text{order} = \text{Nag\_RowMajor}\), \(pdb \geq \max(1, nrhs)\).

9: \(\text{fail}\) – NagError *

**Output**

The NAG error parameter (see the Essential Introduction).

### 6 Error Indicators and Warnings

#### NE_INT

**On entry:** \(n = \langle value\rangle\).

**Constraint:** \(n \geq 0\).

**On entry:** \(nrhs = \langle value\rangle\).

**Constraint:** \(nrhs \geq 0\).

**On entry:** \(pda = \langle value\rangle\).

**Constraint:** \(pda > 0\).

**On entry:** \(pdb = \langle value\rangle\).

**Constraint:** \(pdb > 0\).

#### NE_INT_2

**On entry:** \(pda = \langle value\rangle\), \(n = \langle value\rangle\).

**Constraint:** \(pda \geq \max(1, n)\).

**On entry:** \(pdb = \langle value\rangle\), \(n = \langle value\rangle\).

**Constraint:** \(pdb \geq \max(1, n)\).

**On entry:** \(pdb = \langle value\rangle\), \(nrhs = \langle value\rangle\).

**Constraint:** \(pdb \geq \max(1, nrhs)\).
NE_ALLOC_FAIL
Memory allocation failed.

NE_BAD_PARAM
On entry, parameter (value) had an illegal value.

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 consult NAG for assistance.

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 $\text{uplo} = \text{Nag\_Upper}$, $|E| \leq c(n)\epsilon|U|\|U\|_1$;

if $\text{uplo} = \text{Nag\_Lower}$, $|E| \leq c(n)\epsilon|L|\|L^T\|_1$,

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

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

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

where $\text{cond}(A, x) = \|A^{-1}\|_1 |A| \|x\|_\infty / \|x\|_\infty \leq \text{cond}(A) = \|A^{-1}\|_1 |A|_\infty \leq \kappa_\infty(A)$. Note that $\text{cond}(A, x)$ can be much smaller than $\text{cond}(A)$.

Forward and backward error bounds can be computed by calling nag_dporfs (f07fhc), and an estimate for $\kappa_\infty(A)$ can be obtained by calling nag_dpocon (f07fgc).

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

This function may be followed by a call to nag_dporfs (f07fhc) to refine the solution and return an error estimate.

The complex analogue of this function is nag_zpotrs (f07fsc).

9 Example
To solve 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 and must first be factorized by nag_dpotrf (f07fdc).

9.1 Program Text

/* nag_dpotrs (f07fec) Example Program. */
/* Copyright 2001 Numerical Algorithms Group. */
/* Mark 7, 2001. */
#include <stdio.h>
```c
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf07.h>
#include <nagx04.h>

int main(void)
{
    /* Scalars */
    Integer i, j, n, nrhs, pda, pdb;
    Integer exit_status=0;
    NagError fail;
    Nag_OrderType order;
    /* Arrays */
    char uplo[2];
    double *a=0, *b=0;
    Nag_UploType uplo_enum;

    #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);
    Vprintf("f07fec Example Program Results\n\n");
    /* Skip heading in data file */
    Vscanf("%*[^
]");
    Vscanf("%ld%ld%*[^
] ", &n, &nrhs);
    #ifdef NAG_COLUMN_MAJOR
    pda = n;
    pdb = n;
    #else
    pda = n;
    pdb = nrhs;
    #endif

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

    /* Read A and B from data file */
    Vscanf(" '%s '%*[\n] ", uplo);
    if (*(unsigned char *)uplo == 'L')
        uplo_enum = Nag_Lower;
    else if (*(unsigned char *)uplo == 'U')
        uplo_enum = Nag_Upper;
    else
    {
        Vprintf("Unrecognised character for Nag_UploType type\n");
        exit_status = -1;
        goto END;
    }

    if (uplo_enum == Nag_Upper)
    {
        for (i = 1; i <= n; ++i)
        {
            for (j = i; j <= n; ++j)
                Vscanf("%lf", &A(i,j));
            Vscanf("%*[\n] ");
        }
    }
    else
        ...
```
for (i = 1; i <= n; ++i)
{
    for (j = 1; j <= i; ++j)
        Vscanf("%lf", &A(i,j));
    Vscanf("%*[\n ] ");
}
for (i = 1; i <= n; ++i)
{
    for (j = 1; j <= nrhs; ++j)
        Vscanf("%lf", &B(i,j));
} Vscanf("%*[\n ] ");

/* Factorize A */
 f07fdc(order, uplo_enum, n, a, pda, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07fdc.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Compute solution */
 f07fec(order, uplo_enum, n, nrhs, a, pda, b, pdb, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07fec.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Print solution */
x04cac(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, nrhs, b, pdb,
"Solution(s)", 0, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from x04cac.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
END:
if (a) NAG_FREE(a);
if (b) NAG_FREE(b);
return exit_status;

9.2 Program Data
f07fec Example Program Data
4 2 :Values of N and NRHS
    'L' :Value of UPLO
4.16
-3.12  5.03
0.56  -0.83  0.76
-0.10  1.18  0.34  1.18 :End of matrix A
8.70  8.30
-13.35  2.13
1.89  1.61
-4.14  5.00 :End of matrix B
## 9.3 Program Results

f07fec Example Program Results

<table>
<thead>
<tr>
<th>Solution(s)</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0000</td>
<td>4.0000</td>
</tr>
<tr>
<td>2</td>
<td>-1.0000</td>
<td>3.0000</td>
</tr>
<tr>
<td>3</td>
<td>2.0000</td>
<td>2.0000</td>
</tr>
<tr>
<td>4</td>
<td>-3.0000</td>
<td>1.0000</td>
</tr>
</tbody>
</table>