NAG Library Function Document

nag_dsptrs (f07pec)

1 Purpose

nag_dsptrs (f07pec) solves a real symmetric indefinite system of linear equations with multiple right-hand sides,

\[ AX = B, \]

where \( A \) has been factorized by nag_dsptrf (f07pdc), using packed storage.

2 Specification

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

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

3 Description

nag_dsptrs (f07pec) is used to solve a real symmetric indefinite system of linear equations \( AX = B \), the function must be preceded by a call to nag_dsptrf (f07pdc) which computes the Bunch–Kaufman factorization of \( A \), using packed storage.

If \( \text{uplo} = \text{Nag_Upper} \), \( A = PUDU^TPT \), where \( P \) is a permutation matrix, \( U \) is an upper triangular matrix and \( D \) is a symmetric block diagonal matrix with 1 by 1 and 2 by 2 blocks; the solution \( X \) is computed by solving \( PUDY = B \) and then \( U^TPTX = Y \).

If \( \text{uplo} = \text{Nag_Lower} \), \( A = PLDL^TPT \), where \( L \) is a lower triangular matrix; the solution \( X \) is computed by solving \( PLDY = B \) and then \( L^TPTX = Y \).

4 References


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:* specifies how \( A \) has been factorized.

   uplo = Nag_Upper
   \( A = PUDU^TP \), where \( U \) is upper triangular.

Mark 25
\textbf{uplo} = Nag\_Lower  \\
\hspace{1cm} A = PDLD^T P^T, where \( L \) is lower triangular.  \\
\textit{Constraint: uplo} = Nag\_Upper or Nag\_Lower.

3: \hspace{1cm} \textbf{n} – Integer \hspace{1cm} \textit{Input}  \\
\hspace{1cm} \textit{On entry: } \( n \), the order of the matrix \( A \).  \\
\hspace{1cm} \textit{Constraint: } \( n \geq 0 \).

4: \hspace{1cm} \textbf{nrhs} – Integer \hspace{1cm} \textit{Input}  \\
\hspace{1cm} \textit{On entry: } \( r \), the number of right-hand sides.  \\
\hspace{1cm} \textit{Constraint: } \( nrhs \geq 0 \).

5: \hspace{1cm} \textbf{ap}[\text{dim}] – const double \hspace{1cm} \textit{Input}  \\
\hspace{1cm} \textit{Note: } the dimension, \( \text{dim} \), of the array \textbf{ap} must be at least \( \max(1, n \times (n + 1)/2) \).  \\
\hspace{1cm} \textit{On entry: } the factorization of \( A \) stored in packed form, as returned by nag\_dsptrf (f07pdc).

6: \hspace{1cm} \textbf{ipiv}[\text{dim}] – const Integer \hspace{1cm} \textit{Input}  \\
\hspace{1cm} \textit{Note: } the dimension, \( \text{dim} \), of the array \textbf{ipiv} must be at least \( \max(1, n) \).  \\
\hspace{1cm} \textit{On entry: } details of the interchanges and the block structure of \( D \), as returned by nag\_dsptrf (f07pdc).

7: \hspace{1cm} \textbf{b}[\text{dim}] – double \hspace{1cm} \textit{Input/Output}  \\
\hspace{1cm} \textit{Note: } the dimension, \( \text{dim} \), of the array \textbf{b} must be at least \( \max(1, pdb \times nrhs) \) when \( \text{order} = \text{Nag\_ColMajor} \);  \\
\hspace{1cm} \textit{max(1, n \times pdb) when } \text{order} = \text{Nag\_RowMajor} \).  \\
The \((i, j)\)th element of the matrix \( B \) is stored in  \\
\hspace{1cm} \textbf{b}[(j - 1) \times \text{pdb} + i - 1] \text{ when } \text{order} = \text{Nag\_ColMajor};  \\
\hspace{1cm} \textbf{b}[(i - 1) \times \text{pdb} + j - 1] \text{ when } \text{order} = \text{Nag\_RowMajor}.  \\
\hspace{1cm} \textit{On entry: } the \( n \) by \( r \) right-hand side matrix \( B \).  \\
\hspace{1cm} \textit{On exit: } the \( n \) by \( r \) solution matrix \( X \).

8: \hspace{1cm} \textbf{pdb} – Integer \hspace{1cm} \textit{Input}  \\
\hspace{1cm} \textit{On entry: } the stride separating row or column elements (depending on the value of \textbf{order}) in the array \textbf{b}.  \\
\hspace{1cm} \textit{Constraints:}  \\
\hspace{1.5cm} \textit{if } \text{order} = \text{Nag\_ColMajor} \text{, } \text{pdb} \geq \max(1, n);  \\
\hspace{1.5cm} \textit{if } \text{order} = \text{Nag\_RowMajor} \text{, } \text{pdb} \geq \max(1, nrhs).$

9: \hspace{1cm} \textbf{fail} – Nag\_Error\* \hspace{1cm} \textit{Input/Output}  \\
\hspace{1cm} \textit{The NAG error argument (see Section 3.6 in the Essential Introduction).}$

\section{Error Indicators and Warnings}

\textbf{NE\_ALLOC\_FAIL}  \\
\hspace{1cm} Dynamic memory allocation failed.  \\
\hspace{1cm} See Section 3.2.1.2 in the Essential Introduction for further information.
NE_BAD_PARAM
On entry, argument \texttt{\textit{value}} had an illegal value.

NE_INT
On entry, \texttt{n} = \texttt{\textit{value}}.
Constraint: \texttt{n} \geq 0.

On entry, \texttt{nrhs} = \texttt{\textit{value}}.
Constraint: \texttt{nrhs} \geq 0.

On entry, \texttt{pdb} = \texttt{\textit{value}}.
Constraint: \texttt{pdb} > 0.

NE_INT_2
On entry, \texttt{pdb} = \texttt{\textit{value}} and \texttt{n} = \texttt{\textit{value}}.
Constraint: \texttt{pdb} \leq \max(1, \texttt{n}).

On entry, \texttt{pdb} = \texttt{\textit{value}} and \texttt{nrhs} = \texttt{\textit{value}}.
Constraint: \texttt{pdb} \leq \max(1, \texttt{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.

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

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

where \(\text{cond}(A, x) = \|\|A^{-1}\|A\|\|x\|_\infty / \|x\|_\infty \leq \text{cond}(A) = \|\|A^{-1}\|A\|\| \leq \kappa_\infty(A)\).

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

Forward and backward error bounds can be computed by calling \texttt{nag_dsptrs} (\texttt{f07pec}), and an estimate for \(\kappa_\infty(A)\) (\(= \kappa_1(A)\)) can be obtained by calling \texttt{nag_dspcon} (\texttt{f07pgc}).

8 Parallelism and Performance
\texttt{nag_dsptrs} (\texttt{f07pec}) is not threaded by NAG in any implementation.

\texttt{nag_dsptrs} (\texttt{f07pec}) 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 $2n^2r$.

This function may be followed by a call to nag_dsprfs (f07phc) to refine the solution and return an error estimate.

The complex analogues of this function are nag_zhptrs (f07psc) for Hermitian matrices and nag_zsptrs (f07qsc) for symmetric matrices.

10 Example

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

$$A = \begin{pmatrix} 2.07 & 3.87 & 4.20 & -1.15 \\ 3.87 & -0.21 & 1.87 & 0.63 \\ 4.20 & 1.87 & 1.15 & 2.06 \\ -1.15 & 0.63 & 2.06 & -1.81 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} -9.50 & 27.85 \\ -8.38 & 9.90 \\ -6.07 & 19.25 \\ -0.96 & 3.93 \end{pmatrix}.$$

Here $A$ is symmetric indefinite, stored in packed form, and must first be factorized by nag_dsptrf (f07pdc).

10.1 Program Text

```c
/* nag_dsptrs (f07pec) Example Program. *
 * Copyright 2014 Numerical Algorithms Group.
 */
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf07.h>
#include <nagx04.h>

int main(void)
{
    /* Scalars */
    Integer ap_len, i, j, n, nrhs, pdb;
    Integer exit_status = 0;
    NagError fail;
    Nag_UploType uplo;
    Nag_OrderType order;
    /* Arrays */
    Integer *ipiv = 0;
    char nag_enum_arg[40];
    double *ap = 0, *b = 0;

    #ifdef NAG_LOAD_FP
    /* The following line is needed to force the Microsoft linker to load floating point support */
    float force_loading_of_ms_float_support = 0;
    #endif /* NAG_LOAD_FP */

    #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_UPPER(I, J) ap[J*(J-1)/2 + I - 1]
    #define A_LOWER(I, J) ap[I*(I-1)/2 + J - 1]
    order = Nag_RowMajor;
    #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_dsptrs (f07pec) Example Program Results\n\n");

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

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

/* Read A and B from data file */
#endif _WIN32
scanf_s("%39s%*[\n ", nag_enum_arg, _countof(nag_enum_arg));
#else
scanf("%39s%*[\n ", nag_enum_arg);
#endif _WIN32

uplo = (Nag_UploType) nag_enum_name_to_value(nag_enum_arg);
if (uplo == Nag_Upper)
{
    for (i = 1; i <= n; ++i)
    {
        for (j = i; j <= n; ++j)
        {
            #ifdef _WIN32
            scanf_s("%lf", &A_UPPER(i, j));
            #else
            scanf("%lf", &A_UPPER(i, j));
            #endif
        }
        #ifdef _WIN32
        scanf_s("%*[\n ");
        #else
        scanf("%*[\n ");
        #endif
    }
    else
    {
        for (i = 1; i <= n; ++i)
        {
            for (j = 1; j <= i; ++j)
            {
                #ifdef _WIN32
                scanf_s("%lf", &A_UPPER(i, j));
                #else
                scanf("%lf", &A_UPPER(i, j));
                #endif
            }
            #ifdef _WIN32
            scanf_s("%*[\n ");
            #else
            scanf("%*[\n ");
            #endif
        }
    }
}
```c
    scanf_s("%lf", &A_LOWER(i, j));
#else
    scanf("%lf", &A_LOWER(i, j));
#endif
}
#ifdef _WIN32
    scanf_s("%*['\n] ");
#else
    scanf("%*['\n] ");
#endif
}
for (i = 1; i <= n; ++i)
{
    for (j = 1; j <= nrhs; ++j)
#else
    scanf("%lf", &B(i, j));
#endif
    }
#ifdef _WIN32
    scanf_s("%*['\n] ");
#else
    scanf("%*['\n] ");
#endif
/* Factorize A */
/* nag_dsptrf (f07pdc).
   * Bunch-Kaufman factorization of real symmetric indefinite matrix, packed storage */
nag_dsptrf(order, uplo, n, ap, ipiv, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dsptrf (f07pdc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Compute solution */
/* nag_dsptrs (f07pec).
   * Solution of real symmetric indefinite system of linear equations, multiple right-hand sides, matrix already factorized by nag_dsptrf (f07pdc), packed storage */
nag_dsptrs(order, uplo, n, nrhs, ap, ipiv, b, pdb, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dsptrs (f07pec).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Print solution */
/* 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 = 1;
    goto END;
}
END:
NAG_FREE(ap);
NAG_FREE(ipiv);
NAG_FREE(b);
return exit_status;
}
10.2 Program Data

nag_dsptrs (f07pec) Example Program Data

Values of n and nrhs
Nag_Lower  :Value of uplo
2:

2.07
3.87 -0.21
4.20  1.87  1.15
-1.15  0.63  2.06 -1.81 :End of matrix A
-9.50  27.85
-8.38  9.90
-6.07  19.25
-0.96  3.93 :End of matrix B

10.3 Program Results

nag_dsptrs (f07pec) Example Program Results

Solution(s)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-4.0000 1.0000</td>
</tr>
<tr>
<td>2</td>
<td>-1.0000  4.0000</td>
</tr>
<tr>
<td>3</td>
<td>2.0000  3.0000</td>
</tr>
<tr>
<td>4</td>
<td>5.0000  2.0000</td>
</tr>
</tbody>
</table>