NAG C 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
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

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

If \(uplo = \text{Nag}_\text{Upper}\), \(A = PUDU^T P^T\), 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 \(U^T P^T X = Y\).

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

4 References


5 Parameters

1: \(\text{order} – \text{Nag}_\text{OrderType}\)

   \text{Input}

   On entry: the \text{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 \text{order} = \text{Nag}_\text{RowMajor}. See Section 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.

   Constraint: \text{order} = \text{Nag}_\text{RowMajor} or \text{Nag}_\text{ColMajor}.

2: \(\text{uplo} – \text{Nag}_\text{UploType}\)

   \text{Input}

   On entry: indicates how \(A\) has been factorized as follows:

   if \(uplo = \text{Nag}_\text{Upper}\), \(A = PUDU^T P^T\), where \(U\) is upper triangular;

   if \(uplo = \text{Nag}_\text{Lower}\), \(A = PLDL^T P^T\), where \(L\) is lower triangular.

   Constraint: \text{uplo} = \text{Nag}_\text{Upper} or \text{Nag}_\text{Lower}.

3: \(\text{n} – \text{Integer}\)

   \text{Input}

   On entry: \(n\), the order of the matrix \(A\).

   Constraint: \(n \geq 0\).
4: nrhs – Integer
   Input
   On entry: $r$, the number of right-hand sides.
   Constraint: $\text{nrhs} \geq 0$.

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

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

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

9: fail – NagError *
   Output
   The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT
   On entry, $\text{n} = \langle\text{value}\rangle$.
   Constraint: $\text{n} \geq 0$.

   On entry, $\text{nrhs} = \langle\text{value}\rangle$.
   Constraint: $\text{nrhs} \geq 0$.

   On entry, $\text{pdb} = \langle\text{value}\rangle$.
   Constraint: $\text{pdb} > 0$.

NE_INT_2
   On entry, $\text{pdb} = \langle\text{value}\rangle$, $\text{n} = \langle\text{value}\rangle$.
   Constraint: $\text{pdb} \geq \max(1, \text{n})$.

   On entry, $\text{pdb} = \langle\text{value}\rangle$, $\text{nrhs} = \langle\text{value}\rangle$.
   Constraint: $\text{pdb} \geq \max(1, \text{nrhs})$. 
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 `uplo` = `Nag_Upper`, $|E| \leq c(n)\epsilon P|U| |D| |U^T| P^T$;
- if `uplo` = `Nag_Lower`, $|E| \leq c(n)\epsilon P|L| |D| |L^T| P^T$,

$c(n)$ 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

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

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

Forward and backward error bounds can be computed by calling `nag_dsprfs (f07phc)`, and an estimate for
$\kappa_\infty(A)$ ($= \kappa_1(A)$) can be obtained by calling `nag_dspcon (f07pcg)`.

8 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.

9 Example
To solve 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_dspptrf (f07pdc)`.

9.1 Program Text

```c
/* nag_dsprfs (f07pec) Example Program. */
* * Copyright 2001 Numerical Algorithms Group. *
* * Mark 7, 2001. */
```
#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_enum;
    Nag_OrderType order;
    /* Arrays */
    Integer *ipiv=0;
    char uplo[2];
    double *ap=0, *b=0;

    #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);
    Vprintf("f07pec Example Program Results

");
    /* Skip heading in data file */
    Vscanf("%*[^
] ");
    Vscanf("%ld%ld%*[^\n] ", &n, &nrhs);
    ap_len = n*(n+1)/2;
    #ifdef NAG_COLUMN_MAJOR
    pdb = n;
    #else
    pdb = nrhs;
    #endif

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

    /* Read A and B from data file */
    Vscanf(" %ls '%[^\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)
```c
Vscanf("%lf", &A_UPPER(i,j));
} else {
    for (i = 1; i <= n; ++i) {
        for (j = 1; j <= i; ++j)
            Vscanf("%lf", &A_LOWER(i,j));
    }
    Vscanf("%*[\n ] ");
}
for (i = 1; i <= nrhs; ++i) {
    for (j = 1; j <= n; ++j)
        Vscanf("%lf", &B(i,j));
} Vscanf("%*[\n ] ");

/* Factorize A */
f07pdc(order, uplo_enum, n, ap, ipiv, &fail);
if (fail.code != NE_NOERROR) {
    Vprintf("Error from f07pdc.\n\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Compute solution */
f07pec(order, uplo_enum, n, nrhs, ap, ipiv, b, pdb, &fail);
if (fail.code != NE_NOERROR) {
    Vprintf("Error from f07pec.\n\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\n", fail.message);
    exit_status = 1;
    goto END;
}
END:
if (ap) NAG_FREE(ap);
if (ipiv) NAG_FREE(ipiv);
if (b) NAG_FREE(b);
return exit_status;
```

9.2 Program Data

f07pec Example Program Data

<table>
<thead>
<tr>
<th>4</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(4 \times 2) : Values of (N) and (NRHS)</td>
<td></td>
</tr>
<tr>
<td>(L) : Value of UPLO</td>
<td></td>
</tr>
<tr>
<td>2.07</td>
<td></td>
</tr>
<tr>
<td>3.87</td>
<td>-0.21</td>
</tr>
<tr>
<td>4.20</td>
<td>1.87</td>
</tr>
<tr>
<td>-1.15</td>
<td>0.63</td>
</tr>
<tr>
<td>2.06</td>
<td>-1.81</td>
</tr>
<tr>
<td>-9.50</td>
<td>27.85</td>
</tr>
<tr>
<td>-8.38</td>
<td>9.90</td>
</tr>
<tr>
<td>-6.07</td>
<td>19.25</td>
</tr>
<tr>
<td>-0.96</td>
<td>3.93</td>
</tr>
</tbody>
</table>

:End of matrix \(A\) and \(B\)
### Program Results

**f07pec Example Program Results**

<table>
<thead>
<tr>
<th>Solution(s)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-4.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>2</td>
<td>-1.0000</td>
<td>4.0000</td>
</tr>
<tr>
<td>3</td>
<td>2.0000</td>
<td>3.0000</td>
</tr>
<tr>
<td>4</td>
<td>5.0000</td>
<td>2.0000</td>
</tr>
</tbody>
</table>