NAG C Library Function Document

nag_zsptrs (f07qsc)

1 Purpose

nag_zsptrs (f07qsc) solves a complex symmetric system of linear equations with multiple right-hand sides, $AX = B$, where $A$ has been factorized by nag_zsptrf (f07qrc), using packed storage.

2 Specification

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

3 Description

To solve a complex symmetric system of linear equations $AX = B$, this function must be preceded by a call to nag_zsptrf (f07qrc) which computes the Bunch–Kaufman factorization of $A$ using packed storage.

If $\text{uplo} = \text{Nag\_Upper}$, $A = PUDU^T P^T$, 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 $\text{uplo} = \text{Nag\_Lower}$, $A = PLDL^T P^T$, 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: 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 how $A$ has been factorized as follows:
   - if uplo = Nag_Upper, $A = PUDU^T P^T$, where $U$ is upper triangular;
   - if uplo = Nag_Lower, $A = PLDL^T P^T$, 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: $n \geq 0$. 

[NP3645/7] f07qsc.1
4:  \( \text{nrhs} \) – Integer \hspace{2cm} \text{Input}  
    \( \text{On entry:} \ r, \text{the number of right-hand sides.} \)  
    \( \text{Constraint:} \ \text{nrhs} \geq 0. \)

5:  \( \text{ap}[\text{dim}] \) – const Complex \hspace{2cm} \text{Input}  
    \( \text{Note:} \ \text{the dimension,} \ \text{dim}, \ \text{of the array} \ \text{ap} \ \text{must be at least} \ \max(1, \text{n} \times (\text{n} + 1)/2). \)  
    \( \text{On entry:} \ \text{details of the factorization of} \ A \ \text{stored in packed form, as returned by} \ \text{nag_zsptrf} \ (\text{f07qrc}). \)

6:  \( \text{ipiv}[\text{dim}] \) – const Integer \hspace{2cm} \text{Input}  
    \( \text{Note:} \ \text{the dimension,} \ \text{dim}, \ \text{of the array} \ \text{ipiv} \ \text{must be at least} \ \max(1, \text{n}). \)  
    \( \text{On entry:} \ \text{details of the interchanges and the block structure of} \ D, \ \text{as returned by} \ \text{nag_zsptrf} \ (\text{f07qrc}). \)

7:  \( \text{b}[\text{dim}] \) – Complex \hspace{2cm} \text{Input/Output}  
    \( \text{Note:} \ \text{the dimension,} \ \text{dim}, \ \text{of the array} \ \text{b} \ \text{must be at least} \ \max(1, \text{pdb} \times \text{nrhs}) \) \( \text{when} \ \text{order} = \text{Nag_ColMajor} \) \( \text{and at least} \ \max(1, \text{pdb} \times \text{n}) \) \( \text{when} \ \text{order} = \text{Nag_RowMajor}. \)  
    \( \text{If} \ \text{order} = \text{Nag_ColMajor}, \ \text{the} \ (i, j)\text{th element of the matrix} \ B \ \text{is stored in} \ \text{b}[(\text{j} - 1) \times \text{pdb} + i - 1] \) \( \text{and} \ \text{if} \ \text{order} = \text{Nag_RowMajor}, \ \text{the} \ (i, j)\text{th element of the matrix} \ B \ \text{is stored in} \ \text{b}[(i - 1) \times \text{pdb} + j - 1]. \)  
    \( \text{On entry:} \ \text{the} \ \text{n by} \ r \ \text{right-hand side matrix} \ B. \)  
    \( \text{On exit:} \ \text{the} \ \text{n by} \ r \ \text{solution matrix} \ X. \)

8:  \( \text{pdb} \) – Integer \hspace{2cm} \text{Input}  
    \( \text{On entry:} \ \text{the stride separating matrix row or column elements (depending on the value of} \ \text{order}) \ \text{in the array} \ \text{b}. \)  
    \( \text{Constraints:} \)  
    \[ \begin{align*} 
    & \text{if} \ \text{order} = \text{Nag_ColMajor}, \ \text{pdb} \geq \max(1, \text{n}); \\
    & \text{if} \ \text{order} = \text{Nag_RowMajor}, \ \text{pdb} \geq \max(1, \text{nrhs}). 
    \end{align*} \]

9:  \( \text{fail} \) – NagError * \hspace{2cm} \text{Output}  
    \( \text{The NAG error parameter (see the Essential Introduction).} \)

### 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

\[
\begin{align*}
\text{if } \text{uplo} = \text{Nag Upper}, & \quad |E| \leq c(n)\epsilon P|U||D||U^T|P^T; \\
\text{if } \text{uplo} = \text{Nag Lower}, & \quad |E| \leq c(n)\epsilon P|L||D||L^T|P^T,
\end{align*}
\]

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

Forward and backward error bounds can be computed by calling \text{nag_zsprfs} (f07qvc), and an estimate for \( \kappa_\infty(A) \) (\( = \kappa_1(A) \)) can be obtained by calling \text{nag_zspcon} (f07quc).

8 Further Comments

The total number of real floating-point operations is approximately \( 8n^2r \).

This function may be followed by a call to \text{nag_zsprfs} (f07qvc) to refine the solution and return an error estimate.

The real analogue of this function is \text{nag_dsptrs} (f07pec).

9 Example

To solve the system of equations \( AX = B \), where

\[
A = \begin{pmatrix}
-0.39 - 0.71i & 5.14 - 0.64i & -7.86 - 2.96i & 3.80 + 0.92i \\
5.14 + 0.64i & 8.86 + 1.81i & -3.52 + 0.58i & 5.32 - 1.59i \\
-7.86 - 2.96i & -3.52 + 0.58i & -2.83 - 0.03i & -1.54 - 2.86i \\
3.80 + 0.92i & 5.32 - 1.59i & -1.54 - 2.86i & -0.56 + 0.12i
\end{pmatrix}
\]

and

\[
B = \begin{pmatrix}
-55.64 + 41.22i & -19.09 - 35.97i \\
-48.18 + 66.00i & -12.08 - 27.02i \\
-0.49 - 1.47i & 6.95 + 20.49i \\
-6.43 + 19.24i & -4.59 - 35.53i
\end{pmatrix}
\]

Here \( A \) is symmetric, stored in packed form, and must first be factorized by \text{nag_zsptrf} (f07qrc).
9.1 Program Text

/* nag_zsptrs (f07qsc) 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];
    Complex *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("f07qsc Example Program Results\n\n");

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

    /* Allocate memory */
    if ( !(ipiv = NAG_ALLOC(n, Integer)) ||
        !(ap = NAG_ALLOC(n * (n + 1)/2, Complex)) ||
        !(b = NAG_ALLOC(n * nrhs, Complex)) )
    {
        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
    
    /* Call nag_zsptrs */
    /* Print results */

END:
exit_status;
}

f07qsc
{  
    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 , %lf )", &A_UPPER(i,j).re, &A_UPPER(i,j).im);  
        Vscanf("%*[\n] ");  
    }  
}  
else  
{  
    for (i = 1; i <= n; ++i)  
    {  
        for (j = i; j <= n; ++j)  
            Vscanf(" ( %lf , %lf )", &A_LOWER(i,j).re, &A_LOWER(i,j).im);  
        Vscanf("%*[\n] ");  
    }  
}  
for (i = 1; i <= n; ++i)  
{  
    for (j = 1; j <= nrhs; ++j)  
        Vscanf(" ( %lf , %lf )", &B(i,j).re, &B(i,j).im);  
}  
Vscanf("%*[\n] ");  
/* Factorize A */  
f07qrc(order, uplo_enum, n, ap, ipiv, &fail);  
if (fail.code != NE_NOERROR)  
{  
    Vprintf("Error from f07qrc.\n", fail.message);  
    exit_status = 1;  
    goto END;  
}  
/* Compute solution */  
f07qsc(order, uplo_enum, n, nrhs, ap, ipiv, b, pdb, &fail);  
if (fail.code != NE_NOERROR)  
{  
    Vprintf("Error from f07qsc.\n", fail.message);  
    exit_status = 1;  
    goto END;  
}  
/* Print solution */  
x04dbc(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, nrhs, b, pdb,  
        Nag_BracketForm, "%7.4f", "Solution(s)", Nag_IntegerLabels,  
        0, Nag_IntegerLabels, 0, b0, 0, 0, &fail);  
if (fail.code != NE_NOERROR)  
{  
    Vprintf("Error from x04dbc.\n", fail.message);  
    exit_status = 1;  
    goto END;  
}  
END:  
if (ipiv) NAG_FREE(ipiv);  
if (ap) NAG_FREE(ap);  
if (b) NAG_FREE(b);  
return exit_status;  
}

9.2 Program Data

f07qsc Example Program Data  
4 2  
'L'  
(-0.39,-0.71)  
( 5.14,-0.64)  
( 8.86, 1.81)  

Values of N and NRHS  
Value of UPLO
9.3 Program Results

f07qsc Example Program Results

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