NAG C Library Function Document

\texttt{nag_zhptrs (f07psc)}

1 Purpose

\texttt{nag_zhptrs (f07psc)} solves a complex Hermitian indefinite system of linear equations with multiple right-hand sides, $AX = B$, where $A$ has been factorized by \texttt{nag_zhptrf (f07prc)}, using packed storage.

2 Specification

\begin{verbatim}
void nag_zhptrs (Nag_OrderType order, Nag_UploType uplo, Integer n, Integer nrhs,
               const Complex ap[], const Integer ipiv[], Complex b[], Integer pdb,
               NagError *fail)
\end{verbatim}

3 Description

To solve a complex Hermitian indefinite system of linear equations $AX = B$, this function must be preceded by a call to \texttt{nag_zhptrf (f07prc)} which computes the Bunch–Kaufman factorization of $A$ using packed storage.

If $\text{uplo} = \text{Nag Upper}$, $A = PUDU^HTP^T$, where $P$ is a permutation matrix, $U$ is an upper triangular matrix and $D$ is an Hermitian block diagonal matrix with 1 by 1 and 2 by 2 blocks; the solution $X$ is computed by solving $PUDY = B$ and then $U^HTP^TX = Y$.

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

4 References


5 Parameters

1: \texttt{order} – Nag_OrderType \hspace{1cm} \textit{Input}

\textit{On entry:} the \texttt{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 \texttt{order = Nag_RowMajor}. See Section 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.

\textit{Constraint:} \texttt{order = Nag_RowMajor} or \texttt{Nag_ColMajor}.

2: \texttt{uplo} – Nag_UploType \hspace{1cm} \textit{Input}

\textit{On entry:} indicates how $A$ has been factorized as follows:

- if $\text{uplo} = \text{Nag Upper}$, $A = PUDU^HTP^T$, where $U$ is upper triangular;
- if $\text{uplo} = \text{Nag Lower}$, $A = PLDL^HTP^T$, where $L$ is lower triangular.

\textit{Constraint:} \texttt{uplo = Nag_Upper} or \texttt{Nag_Lower}.

3: \texttt{n} – Integer \hspace{1cm} \textit{Input}

\textit{On entry:} $n$, the order of the matrix $A$.

\textit{Constraint:} \texttt{n} $\geq 0$.
4: \textbf{nrhs} – Integer \hspace{1cm} \textit{Input}

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

Constraint: \( \textbf{nrhs} \geq 0 \).

5: \( \textbf{ap}[\text{dim}] \) – const Complex \hspace{1cm} \textit{Input}

Note: the dimension, \( \text{dim} \), of the array \( \textbf{ap} \) must be at least \( \max(1, n \times (n + 1)/2) \).

On entry: details of the factorization of \( A \) stored in packed form, as returned by nag_zhptrf (f07prc).

6: \( \textbf{ipiv}[\text{dim}] \) – const Integer \hspace{1cm} \textit{Input}

Note: the dimension, \( \text{dim} \), of the array \( \textbf{ipiv} \) must be at least \( \max(1, n) \).

On entry: details of the interchanges and the block structure of \( D \), as returned by nag_zhptrf (f07prc).

7: \( \textbf{b}[\text{dim}] \) – Complex \hspace{1cm} \textit{Input/Output}

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

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

On entry: the \( n \) by \( r \) right-hand side matrix \( B \).

On exit: the \( n \) by \( r \) solution matrix \( X \).

8: \( \textbf{pdb} \) – Integer \hspace{1cm} \textit{Input}

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

Constraints:

\[
\begin{align*}
\text{if } \text{order} = \text{Nag.ColMajor}, & \quad \text{pdb} \geq \max(1, n) ; \\
\text{if } \text{order} = \text{Nag.RowMajor}, & \quad \text{pdb} \geq \max(1, \text{nrhs}) .
\end{align*}
\]

9: \( \textbf{fail} \) – NagError * \hspace{1cm} \textit{Output}

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, \( n = \langle \text{value} \rangle \).

Constraint: \( n \geq 0 \).

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, \( \text{pdb} = \langle \text{value} \rangle, n = \langle \text{value} \rangle \).

Constraint: \( \text{pdb} \geq \max(1, n) \).

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 \(\text{uplo} = \text{Nag}_\text{Upper}\), \(|E| \leq c(n)\epsilon P|U||D|P^H|P^T|\);
- if \(\text{uplo} = \text{Nag}_\text{Lower}\), \(|E| \leq c(n)\epsilon P|L||D|L^H|P^T|,
where \(c(n)\) 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 \(\text{cond}(A, x) = |||A^{-1}|||A|||x||_{\infty}/||x||_{\infty} \leq \text{cond}(A) = |||A^{-1}|||A||_{\infty} \leq \kappa_\infty(A)\). 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_zhprfs (f07pvc), and an estimate for \(\kappa_\infty(A) \ (= \kappa_1(A))\) can be obtained by calling nag_zhpcon (f07puc).

8 Further Comments
The total number of real floating-point operations is approximately \(8n^2r\).
This function may be followed by a call to nag_zhprfs (f07pvc) to refine the solution and return an error estimate.
The real analogue of this function is nag_dsprts (f07pec).

9 Example
To solve the system of equations \(AX = B\), where
\[
A = \begin{pmatrix}
-1.36 + 0.00i & 1.58 + 0.90i & 2.21 - 0.21i & 3.91 + 1.50i \\
1.58 - 0.90i & -8.87 + 0.00i & -1.84 - 0.03i & -1.78 + 1.18i \\
2.21 + 0.21i & -1.84 + 0.03i & -4.63 + 0.00i & 0.11 + 0.11i \\
3.91 - 1.50i & -1.78 - 1.18i & 0.11 - 0.11i & -1.84 + 0.00i
\end{pmatrix}
\]
and
\[
B = \begin{pmatrix}
-7.79 + 5.48i & -35.39 + 18.01i \\
-0.77 - 16.05i & 4.23 - 70.02i \\
-9.58 + 3.88i & -24.79 - 8.40i \\
2.98 - 10.18i & 28.68 - 39.89i
\end{pmatrix}
\]
Here \(A\) is Hermitian indefinite, stored in packed form, and must first be factorized by nag_zhptrf (f07pdc).
9.1 Program Text

/* nag_zhptrs (f07psc) Example Program.
 * Copyright 2001 Numerical Algorithms Group.
 */

#include <stdio.h>
#include <nag.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;

    INIT_FAIL(fail);
    Vprintf("f07psc Example Program Results\n\n");
    /* Skip heading in data file */
    Vscanf("%*[^
] ");
    Vscanf("%ld%ld%*[^
] ", &n, &nrhs);
    ap_len = n * (n + 1)/2;
    //if 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("f07psc Example Program Results\n\n");
    /* Skip heading in data file */
    Vscanf("%*[^
] ");
    Vscanf("%ld%ld%*[^
] ", &n, &nrhs);
    ap_len = n * (n + 1)/2;
    #ifdef NAG_COLUMN_MAJOR
    pdb = n;
    #else
    pdb = nrhs;
    #endif
    #ifdef NAG_COLUMN_MAJOR
    pdb = n;
    #else
    pdb = nrhs;
    #endif
    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("%*[\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 , %lf )", &A_UPPER(i,j).re, &A_UPPER(i,j).im);
    }
    Vscanf("%*[\n ]");
}
else
{
    for (i = 1; i <= n; ++i)
    {
        for (j = 1; j <= i; ++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 */
f07prc(order, uplo_enum, n, ap, ipiv, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07prc.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Compute solution */
f07psc(order, uplo_enum, n, nrhs, ap, ipiv, b, pdb, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07psc.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Print solution */
x04dbc(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, nrhs, b, pdb,
   NagBracketForm, "%7.4f", "Solution(s)", Nag_IntegerLabels,
   0, Nag_IntegerLabels, 0, 80, 0, 0, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from x04dbc.\n%s\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
f07psc Example Program Data
4 2
'L'
(-1.36, 0.00)
( 1.58,-0.90) (-8.87, 0.00)
( 2.21, 0.21) (-1.84, 0.03) (-4.63, 0.00)
9.3 Program Results

f07psc Example Program Results

Solution(s)

<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>(3.0000, -4.0000)</td>
</tr>
<tr>
<td>2</td>
<td>(-1.0000, 2.0000)</td>
<td>(-1.0000, 5.0000)</td>
</tr>
<tr>
<td>3</td>
<td>(3.0000, -2.0000)</td>
<td>(7.0000, -2.0000)</td>
</tr>
<tr>
<td>4</td>
<td>(2.0000, 1.0000)</td>
<td>(-8.0000, 6.0000)</td>
</tr>
</tbody>
</table>