NAG C Library Function Document

nag_zpptrs (f07gsc)

1 Purpose

nag_zpptrs (f07gsc) solves a complex Hermitian positive-definite system of linear equations with multiple right-hand sides, $AX = B$, where $A$ has been factorized by nag_zpptrf (f07grc), using packed storage.

2 Specification

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

3 Description

To solve a complex Hermitian positive-definite system of linear equations $AX = B$, this function must be preceded by a call to nag_zpptrf (f07grc) which computes the Cholesky factorization of $A$ using packed storage. The solution $X$ is computed by forward and backward substitution.

If $\text{uplo} = \text{Nag\_Upper}$, $A = U^H U$, where $U$ is upper triangular; the solution $X$ is computed by solving $U^H Y = B$ and then $UX = Y$.

If $\text{uplo} = \text{Nag\_Lower}$, $A = LL^H$, where $L$ is lower triangular; the solution $X$ is computed by solving $LY = B$ and then $L^H X = Y$.

4 References


5 Parameters

1:  `order` – Nag_OrderType  
*Input*  
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  
*Input*  
On entry: indicates whether $A$ has been factorized as $U^H U$ or $LL^H$ as follows:  
- if `uplo = Nag\_Upper`, $A = U^H U$, where $U$ is upper triangular;  
- if `uplo = Nag\_Lower`, $A = LL^H$, where $L$ is lower triangular.  
*Constraint:* `uplo = Nag\_Upper` or `Nag\_Lower`.

3:  `n` – Integer  
*Input*  
On entry: $n$, the order of the matrix $A$.  
*Constraint:* $n \geq 0$.  

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

5:  \( \text{ap}[\dim] \) – const Complex  
\( \text{Input} \)  
\( \text{Note:} \) the dimension, \( \dim \), of the array \( \text{ap} \) must be at least \( \max(1, n \times (n + 1)/2) \).  
\( On\ entry: \) the Cholesky factor of \( A \) stored in packed form, as returned by \text{nag_zpptrf (f07grc)}.  

6:  \( \text{b}[\dim] \) – Complex  
\( \text{Input/Output} \)  
\( \text{Note:} \) the dimension, \( \dim \), 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 \( \max(1, \text{pdb} \times n) \) 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 \( n \) by \( r \) right-hand side matrix \( B \).  
\( On\ exit: \) the \( n \) by \( r \) solution matrix \( X \).  

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

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

6  \text{Error Indicators and Warnings}  

\text{NE\_INT}  
\( On\ entry, \text{n} = \langle \text{value} \rangle. \)  
\( Constraint: \text{n} \geq 0. \)  
\( On\ entry, \text{nrhs} = \langle \text{value} \rangle. \)  
\( Constraint: \text{nrhs} \geq 0. \)  
\( On\ entry, \text{pdb} = \langle \text{value} \rangle. \)  
\( Constraint: \text{pdb} > 0. \)

\text{NE\_INT\_2}  
\( On\ entry, \text{pdb} = \langle \text{value} \rangle, \text{n} = \langle \text{value} \rangle. \)  
\( Constraint: \text{pdb} \geq \max(1, n). \)  
\( On\ entry, \text{pdb} = \langle \text{value} \rangle, \text{nrhs} = \langle \text{value} \rangle. \)  
\( Constraint: \text{pdb} \geq \max(1, \text{nrhs}). \)

\text{NE\_ALLOC\_FAIL}  
Memory allocation failed.

\text{NE\_BAD\_PARAM}  
\( On\ entry, \) parameter \( \langle \text{value} \rangle \) had an illegal value.
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 $\text{ulplo} = \text{Nag}_\text{Upper}$, $|E| \leq c(n)e|U^H||U|$;
- if $\text{ulplo} = \text{Nag}_\text{Lower}$, $|E| \leq c(n)e|L||L^H|$,

$c(n)$ 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 $\text{cond}(A, x) = \|A^{-1}\|_{\infty} \|A\|_{\infty} / \|x\|_{\infty} \leq \text{cond}(A) = \|A^{-1}\|_{\infty} \leq \kappa_\infty(A)$. 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_zpprfs (f07gvc), and an estimate for $\kappa_\infty(A) (= \kappa_1(A))$ can be obtained by calling nag_zppcon (f07guc).

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_zpprfs (f07gvc) to refine the solution and return an error estimate.

The real analogue of this function is nag_dpptrs (f07gec).

9 Example

To solve the system of equations $AX = B$, where

$$ A = \begin{pmatrix} 3.23 + 0.00i & 1.51 - 1.92i & 1.90 + 0.84i & 0.42 + 2.50i \\ 1.51 + 1.92i & 3.58 + 0.00i & -0.23 + 1.11i & -1.18 + 1.37i \\ 1.90 - 0.84i & -0.23 - 1.11i & 4.09 + 0.00i & 2.33 - 0.14i \\ 0.42 - 2.50i & -1.18 - 1.37i & 2.33 + 0.14i & 4.29 + 0.00i \end{pmatrix} $$

and


Here $A$ is Hermitian positive-definite, stored in packed form, and must first be factorized by nag_zpptrf (f07gsc).

9.1 Program Text

/* nag_zpptrs (f07gsc) Example Program. */
/* Copyright 2001 Numerical Algorithms Group. */
/* Mark 7, 2001. */
*/
#include <stdio.h>

[NP3645/7]
```c
#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 */
    char uplo[2];
    Complex *ap=0, *b=0;

    /* Skipping */
    INIT_FAIL(fail);
    Vprintf("f07gsc 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
    /* Allocate memory */
    if ( !(ap = NAG_ALLOC(ap_len, 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
    {
        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("%*[^
  
  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 */
f07grc(order, uplo_enum, n, ap, &fail);
if (fail.code != NE_NOERROR)
{
  Vprintf("Error from f07grc.
%s
", fail.message);
  exit_status = 1;
goto END;
}
/* Compute solution */
f07gsc(order, uplo_enum, n, nrhs, ap, b, pdb, &fail);
if (fail.code != NE_NOERROR)
{
  Vprintf("Error from f07gsc.
%s
", 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.
%s
", fail.message);
  exit_status = 1;
goto END;
}
END:
if (ap) NAG_FREE(ap);
if (b) NAG_FREE(b);
return exit_status;
}

9.2 Program Data

f07gsc Example Program Data

<table>
<thead>
<tr>
<th>4</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(3.23, 0.00)</td>
<td>(3.58, 0.00)</td>
</tr>
<tr>
<td>(1.51, 1.92)</td>
<td>(1.48, 6.58)</td>
</tr>
<tr>
<td>(1.90, -0.84)</td>
<td>(6.17, -4.75)</td>
</tr>
<tr>
<td>(0.42, -2.50)</td>
<td>(-7.17, -21.83)</td>
</tr>
<tr>
<td>(6.17, -4.75)</td>
<td>(7.64, -10.79)</td>
</tr>
</tbody>
</table>

:Values of N and NRHS

‘L’ :Value of UPLO

End of matrix A

9.3 Program Results

f07gsc Example Program Results

Solution(s)

1

1 ( 1.0000, -1.0000) (-1.0000, 2.0000)
2 (-0.0000, 3.0000) (3.0000, -4.0000)
3 (-4.0000, -5.0000) (-2.0000, 3.0000)
4 (2.0000, 1.0000) (4.0000, -5.0000)