NAG C Library Function Document

nag_ztrsyl (f08qvc)

1 Purpose
nag_ztrsyl (f08qvc) solves the complex triangular Sylvester matrix equation.

2 Specification

void nag_ztrsyl (Nag_OrderType order, Nag_TransType trana, Nag_TransType tranb,
                 Nag_SignType sign, Integer m, Integer n, const Complex a[], Integer pda,
                 const Complex b[], Integer pdb, Complex c[], Integer pdc, double *scal,
                 NagError *fail)

3 Description
nag_ztrsyl (f08qvc) solves the complex Sylvester matrix equation
\[ \text{op}(A)X \pm X\text{op}(B) = \alpha C, \]
where \( \text{op}(A) = A \) or \( A^H \), and the matrices \( A \) and \( B \) are upper triangular; \( \alpha \) is a scale factor (\( \leq 1 \)) determined by the function to avoid overflow in \( X \); \( A \) is \( m \) by \( m \) and \( B \) is \( n \) by \( n \) while the right-hand side matrix \( C \) and the solution matrix \( X \) are both \( m \) by \( n \). The matrix \( X \) is obtained by a straightforward process of back substitution (see Golub and Van Loan (1996)).

Note that the equation has a unique solution if and only if \( \alpha_i \pm \beta_j \neq 0 \), where \( \{\alpha_i\} \) and \( \{\beta_j\} \) are the eigenvalues of \( A \) and \( B \) respectively and the sign (+ or −) is the same as that used in the equation to be solved.

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: trana – Nag_TransType

On entry: specifies the option \( \text{op}(A) \) as follows:

- if trana = Nag_NoTrans, then \( \text{op}(A) = A \);
- if trana = Nag_ConjTrans, then \( \text{op}(A) = A^H \).

Constraint: trana = Nag_NoTrans or Nag_ConjTrans.
3: \( \text{tranb} \) – Nag_TransType

*Input*

On entry: specifies the option \( \text{op}(B) \) as follows:

- if \( \text{tranb} = \text{Nag_NoTrans} \), then \( \text{op}(B) = B \);
- if \( \text{tranb} = \text{Nag_ConjTrans} \), then \( \text{op}(B) = B^H \).

Constraint: \( \text{tranb} = \text{Nag_NoTrans} \) or \( \text{Nag_ConjTrans} \).

4: \( \text{sign} \) – Nag_SignType

*Input*

On entry: indicates the form of the Sylvester equation as follows:

- if \( \text{sign} = \text{Nag_Plus} \), then the equation is of the form \( \text{op}(A) X + X \text{op}(B) = \alpha C \);
- if \( \text{sign} = \text{Nag_Minus} \), then the equation is of the form \( \text{op}(A) X - X \text{op}(B) = \alpha C \).

Constraint: \( \text{sign} = \text{Nag_Plus} \) or \( \text{Nag_Minus} \).

5: \( m \) – Integer

*Input*

On entry: \( m \), the order of the matrix \( A \), and the number of rows in the matrices \( X \) and \( C \).

Constraint: \( m \geq 0 \).

6: \( n \) – Integer

*Input*

On entry: \( n \), the order of the matrix \( B \), and the number of columns in the matrices \( X \) and \( C \).

Constraint: \( n \geq 0 \).

7: \( a[\text{dim}] \) – const Complex

*Input*

Note: the dimension, \( \text{dim} \), of the array \( a \) must be at least \( \max(1, \text{pda} \times m) \).

If \( \text{order} = \text{Nag_ColMajor} \), the \( (i, j) \)th element of the matrix \( A \) is stored in \( a[(j - 1) \times \text{pda} + i - 1] \) and if \( \text{order} = \text{Nag_RowMajor} \), the \( (i, j) \)th element of the matrix \( A \) is stored in \( a[(i - 1) \times \text{pda} + j - 1] \).

On entry: the \( m \) by \( m \) upper triangular matrix \( A \).

8: \( \text{pda} \) – Integer

*Input*

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

Constraint: \( \text{pda} \geq \max(1, m) \).

9: \( b[\text{dim}] \) – const Complex

*Input*

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

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 \( n \) upper triangular matrix \( B \).

10: \( \text{pdb} \) – Integer

*Input*

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

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

11: \( c[\text{dim}] \) – Complex

*Input/Output*

Note: the dimension, \( \text{dim} \), of the array \( c \) must be at least \( \max(1, \text{pdc} \times n) \) when \( \text{order} = \text{Nag_ColMajor} \) and at least \( \max(1, \text{pdc} \times m) \) when \( \text{order} = \text{Nag_RowMajor} \).

If \( \text{order} = \text{Nag_ColMajor} \), the \( (i, j) \)th element of the matrix \( C \) is stored in \( c[(j - 1) \times \text{pdc} + i - 1] \) and if \( \text{order} = \text{Nag_RowMajor} \), the \( (i, j) \)th element of the matrix \( C \) is stored in \( c[(i - 1) \times \text{pdc} + j - 1] \).
On entry: the \( m \) by \( n \) right-hand side matrix \( C \).
On exit: \( c \) is overwritten by the solution matrix \( X \).

12: \texttt{pdc} – Integer \hspace{0.5cm} \textit{Input}

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

Constraints:
- if \( \texttt{order} = \text{Nag\_ColMajor} \), \( \texttt{pdc} \geq \max(1, \texttt{m}) \);
- if \( \texttt{order} = \text{Nag\_RowMajor} \), \( \texttt{pdc} \geq \max(1, \texttt{n}) \).

13: \texttt{scal} – double * \hspace{0.5cm} \textit{Output}

On exit: the value of the scale factor \( \alpha \).

14: \texttt{fail} – NagError * \hspace{0.5cm} \textit{Output}

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

\textbf{NE\_INT}

On entry, \( \texttt{m} = \langle \text{value} \rangle \).
Constraint: \( \texttt{m} \geq 0 \).

On entry, \( \texttt{n} = \langle \text{value} \rangle \).
Constraint: \( \texttt{n} \geq 0 \).

On entry, \( \texttt{pda} = \langle \text{value} \rangle \).
Constraint: \( \texttt{pda} > 0 \).

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

On entry, \( \texttt{pdc} = \langle \text{value} \rangle \).
Constraint: \( \texttt{pdc} > 0 \).

\textbf{NE\_INT\_2}

On entry, \( \texttt{pda} = \langle \text{value} \rangle, \texttt{m} = \langle \text{value} \rangle \).
Constraint: \( \texttt{pda} \geq \max(1, \texttt{m}) \).

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

On entry, \( \texttt{pdc} = \langle \text{value} \rangle, \texttt{m} = \langle \text{value} \rangle \).
Constraint: \( \texttt{pdc} \geq \max(1, \texttt{m}) \).

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

\textbf{NE\_PERTURBED}

\( A \) and \( B \) have common or close eigenvalues, perturbed values of which were used to solve the equation.

\textbf{NE\_ALLOC\_FAIL}

Memory allocation failed.

\textbf{NE\_BAD\_PARAM}

On entry, parameter \( \langle \text{value} \rangle \) had an illegal value.
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
Consider the equation $AX - XB = C$. (To apply the remarks to the equation $AX + XB = C$, simply replace $B$ by $-B$.)

Let $\tilde{X}$ be the computed solution and $R$ the residual matrix:

$$ R = C - (A\tilde{X} - \tilde{XB}). $$

Then the residual is always small:

$$ \|R\|_F = O(\epsilon) \left( \|A\|_F + \|B\|_F \right) \|X\|_F. $$

However, $\tilde{X}$ is not necessarily the exact solution of a slightly perturbed equation; in other words, the solution is not backwards stable.

For the forward error, the following bound holds:

$$ \|\tilde{X} - X\|_F \leq \frac{\|R\|_F}{\text{sep}(A, B)} $$

but this may be a considerable over estimate. See Golub and Van Loan (1996) for a definition of $\text{sep}(A, B)$, and Higham (1992) for further details.

These remarks also apply to the solution of a general Sylvester equation, as described in Section 8.

8 Further Comments
The total number of real floating-point operations is approximately $4mn(m + n)$.

To solve the general complex Sylvester equation

$$ AX \pm XB = C $$

where $A$ and $B$ are general matrices, $A$ and $B$ must first be reduced to Schur form :

$$ A = Q_1 \tilde{A} Q_1^H \quad \text{and} \quad B = Q_2 \tilde{B} Q_2^H $$

where $\tilde{A}$ and $\tilde{B}$ are upper triangular and $Q_1$ and $Q_2$ are unitary. The original equation may then be transformed to:

$$ \tilde{A}X \pm \tilde{X}\tilde{B} = \tilde{C} $$

where $\tilde{X} = Q_1^H X Q_2$ and $\tilde{C} = Q_1^H C Q_2$. $\tilde{C}$ may be computed by matrix multiplication; nag_ztrsyl (f08qvc) may be used to solve the transformed equation; and the solution to the original equation can be obtained as $X = Q_1 \tilde{X} Q_2^H$.

The real analogue of this function is nag_dtrsyl (f08qhc).

9 Example
To solve the Sylvester equation $AX + XB = C$, where

$$ A = \begin{pmatrix} -6.00 - 7.00i & 0.36 - 0.36i & -0.19 + 0.48i & 0.88 - 0.25i \\ 0.00 + 0.00i & -5.00 + 2.00i & -0.03 - 0.72i & -0.23 + 0.13i \\ 0.00 + 0.00i & 0.00 + 0.00i & 8.00 - 1.00i & 0.94 + 0.53i \\ 0.00 + 0.00i & 0.00 + 0.00i & 0.00 + 0.00i & 3.00 - 4.00i \end{pmatrix}. $$
$B = \begin{pmatrix} 0.50 - 0.20i & -0.29 - 0.16i & -0.37 + 0.84i & -0.55 + 0.73i \\ 0.00 + 0.00i & -0.40 + 0.90i & 0.06 + 0.22i & -0.43 + 0.17i \\ 0.00 + 0.00i & 0.00 + 0.00i & -0.90 - 0.10i & -0.89 - 0.42i \\ 0.00 + 0.00i & 0.00 + 0.00i & 0.00 + 0.00i & 0.30 - 0.70i \end{pmatrix}$

and

$C = \begin{pmatrix} 0.63 + 0.35i & 0.45 - 0.56i & 0.08 - 0.14i & -0.17 - 0.23i \\ -0.17 + 0.09i & -0.07 - 0.31i & 0.27 - 0.54i & 0.35 + 1.21i \\ -0.93 - 0.44i & -0.33 - 0.35i & 0.41 - 0.03i & 0.57 + 0.84i \\ 0.54 + 0.25i & -0.62 - 0.05i & -0.52 - 0.13i & 0.11 - 0.08i \end{pmatrix}$

9.1 Program Text

`/* nag_ztrsyl (f08qvc) Example Program. */
* Copyright 2001 Numerical Algorithms Group.
* * Mark 7, 2001.
*/
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf08.h>
#include <nagx04.h>

int main(void)
{
    /* Scalars */
    Integer i, j, m, n, pda, pdb, pdc;
    Integer exit_status=0;
    double scale;
    NagError fail;
    Nag_OrderType order;
    /* Arrays */
    #ifdef NAG_COLUMN_MAJOR
    #define A(I,J) a[(J-1)*pda+I-1]
    #define B(I,J) b[(J-1)*pdb+I-1]
    #define C(I,J) c[(J-1)*pdc+I-1]
    order = Nag_ColMajor;
    #else
    #define A(I,J) a[(I-1)*pda+J-1]
    #define B(I,J) b[(I-1)*pdb+J-1]
    #define C(I,J) c[(I-1)*pdc+J-1]
    order = Nag_RowMajor;
    #endif
    INIT_FAIL(fail);
   _VFprintf("f08qvc Example Program Results\n\n");

    /* Skip heading in data file */
    Vscanf("%*[^\n] ");
    Vscanf("\d%*[^\n] ", &m, &n);
    #ifdef NAG_COLUMN_MAJOR
    pda = m;
    pdb = n;
    pdc = m;
    #else
    pda = m;
    pdb = n;
    pdc = n;
    #endif

    /* Allocate memory */
if (!(a = NAG_ALLOC(m * m, Complex)) ||
!(b = NAG_ALLOC(n * m, Complex)) ||
!(c = NAG_ALLOC(m * n, Complex)))
{
    Vprintf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

/* Read A, B and C from data file */
for (i = 1; i <= m; ++i)
{
    for (j = 1; j <= m; ++j)
        Vscanf("( %lf , %lf ) ", &A(i,j).re, &A(i,j).im);
}
Vscanf("%*[\n ]");
for (i = 1; i <= n; ++i)
{
    for (j = 1; j <= n; ++j)
        Vscanf("( %lf , %lf ) ", &B(i,j).re, &B(i,j).im);
}
Vscanf("%*[\n ]");
for (i = 1; i <= m; ++i)
{
    for (j = 1; j <= n; ++j)
        Vscanf("( %lf , %lf ) ", &C(i,j).re, &C(i,j).im);
}
Vscanf("%*[\n ]");

/* Reorder the Schur factorization T */
f08qvc(order, Nag_NoTrans, Nag_NoTrans, Nag_Plus, m, n, a, pda,
b, pdb, c, pdc, &scale, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f08qvc.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Print the solution matrix X stored in C */
x04dbc(order, Nag_GeneralMatrix, Nag_NoUnitDiag, m, n,
c, pdc, Nag_BracketForm, "%7.4f", "Solution matrix X",
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;
}
Vprintf("\n SCALE = %10.2e\n", scale);

END:
if (a) NAG_FREE(a);
if (b) NAG_FREE(b);
if (c) NAG_FREE(c);
return exit_status;
}

9.2 Program Data

f08qvc Example Program Data
4 4
(-0.00,-7.00) ( 0.36,-0.36) (-0.19, 0.48) ( 0.88,-0.25)
( 0.00, 0.00) (-5.00, 2.00) (-0.03,-0.72) (-0.23, 0.13)
( 0.00, 0.00) ( 0.00, 0.00) ( 8.00,-1.00) ( 0.94, 0.53)
( 0.00, 0.00) ( 0.00, 0.00) ( 0.00, 0.00) ( 3.00,-4.00)
:End of matrix A
( 0.50,-0.20) (-0.29,-0.16) (-0.37, 0.84) (-0.55, 0.73)
( 0.00, 0.00) (-0.40, 0.90) ( 0.06, 0.22) (-0.43, 0.17)
( 0.00, 0.00) ( 0.00, 0.00) (-0.90,-0.10) (-0.89,-0.42)
( 0.00, 0.00) ( 0.00, 0.00) ( 0.00, 0.00) ( 0.30,-0.70)
:End of matrix B
( 0.63, 0.35) ( 0.45,-0.56) ( 0.08,-0.14) (-0.17,-0.23)
### 9.3 Program Results

**f08qvce** Example Program Results

Solution matrix $X$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(-0.0611, 0.0249)</td>
<td>(-0.0031, 0.0798)</td>
<td>(-0.0062, 0.0165)</td>
<td>(0.0054,-0.0063)</td>
</tr>
<tr>
<td>2</td>
<td>(0.0215,-0.0003)</td>
<td>(-0.0155, 0.0570)</td>
<td>(-0.0665, 0.0718)</td>
<td>(0.0290,-0.2636)</td>
</tr>
<tr>
<td>3</td>
<td>(-0.0949,-0.0785)</td>
<td>(-0.0415,-0.0298)</td>
<td>(0.0357, 0.0244)</td>
<td>(0.0284, 0.1108)</td>
</tr>
<tr>
<td>4</td>
<td>(0.0281, 0.1052)</td>
<td>(-0.0970,-0.1214)</td>
<td>(-0.0271,-0.0940)</td>
<td>(0.0402, 0.0048)</td>
</tr>
</tbody>
</table>

SCALE = 1.00e+00