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; α is a scale factor (≤ 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

Golub G H and Van Loan C F (1996) *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

Higham N J (1992) Perturbation theory and backward error for $AX - XB = C$ *Numerical Analysis Report* University of Manchester

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: **trana** – Nag_TransType *Input*

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: **tranb** – Nag_TransType *Input*

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

if **tranb** = **Nag_NoTrans**, then $\mathrm{op}(B) = B$;

if **tranb** = **Nag_ConjTrans**, then $\mathrm{op}(B) = B^H$.

Constraint: **tranb** = **Nag_NoTrans** or **Nag_ConjTrans**.

4: **sign** – Nag_SignType *Input*

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

if **sign** = **Nag_Plus**, then the equation is of the form $\mathrm{op}(A)X + X\ \mathrm{op}(B) = \alpha C$;

if **sign** = **Nag_Minus**, then the equation is of the form $\mathrm{op}(A)X - X\ \mathrm{op}(B) = \alpha C$.

Constraint: **sign** = **Nag_Plus** or **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** ≥ 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** ≥ 0.

7: **a**[dim] – const Complex *Input*

Note: the dimension, dim, of the array **a** must be at least max(1, **pda** × **m**).

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

On entry: the m by m upper triangular matrix A.

8: **pda** – Integer *Input*

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

Constraint: **pda** $\geq$ max(1, **m**).

9: **b**[dim] – const Complex *Input*

Note: the dimension, dim, of the array **b** must be at least max(1, **pdb** × **n**).

If **order** = **Nag_ColMajor**, the (i, j)th element of the matrix B is stored in **b**[$(j-1) \times$ **pdb** $+ i - 1$] and if **order** = **Nag_RowMajor**, the (i, j)th element of the matrix B is stored in **b**[$(i-1) \times$ **pdb** $+ j - 1$].

On entry: the n by n upper triangular matrix B.

10: **pdb** – Integer *Input*

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

Constraint: **pdb** $\geq$ max(1, **n**).

11: **c**[dim] – Complex *Input/Output*

Note: the dimension, dim, of the array **c** must be at least max(1, **pdc** × **n**) when **order** = **Nag_ColMajor** and at least max(1, **pdc** × **m**) when **order** = **Nag_RowMajor**.

If **order** = **Nag_ColMajor**, the (i, j)th element of the matrix C is stored in **c**[$(j-1) \times$ **pdc** $+ i - 1$] and if **order** = **Nag_RowMajor**, the (i, j)th element of the matrix C is stored in **c**[$(i-1) \times$ **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: **pdc** – Integer *Input*

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

Constraints:

if **order** = **Nag_ColMajor**, **pdc** $\geq \max(1, \mathbf{m})$;
if **order** = **Nag_RowMajor**, **pdc** $\geq \max(1, \mathbf{n})$.

13: **scal** – double * *Output*

On exit: the value of the scale factor α.

14: **fail** – NagError * *Output*

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, **m** = ⟨*value*⟩.
Constraint: **m** ≥ 0.

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 0.

On entry, **pda** = ⟨*value*⟩.
Constraint: **pda** > 0.

On entry, **pdb** = ⟨*value*⟩.
Constraint: **pdb** > 0.

On entry, **pdc** = ⟨*value*⟩.
Constraint: **pdc** > 0.

NE_INT_2

On entry, **pda** = ⟨*value*⟩, **m** = ⟨*value*⟩.
Constraint: **pda** $\geq \max(1, \mathbf{m})$.

On entry, **pdb** = ⟨*value*⟩, **n** = ⟨*value*⟩.
Constraint: **pdb** $\geq \max(1, \mathbf{n})$.

On entry, **pdc** = ⟨*value*⟩, **m** = ⟨*value*⟩.
Constraint: **pdc** $\geq \max(1, \mathbf{m})$.

On entry, **pdc** = ⟨*value*⟩, **n** = ⟨*value*⟩.
Constraint: **pdc** $\geq \max(1, \mathbf{n})$.

NE_PERTURBED

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

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

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:

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

Then the residual is always small:

$$\|R\|_F = O(\epsilon)\ (\|A\|_F + \|B\|_F)\|\tilde{X}\|_F.$$

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:

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

but this may be a considerable over estimate. See Golub and Van Loan (1996) for a definition of $sep(A,B)$, 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:

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

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 */
  Complex *a=0, *b=0, *c=0;

#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);
  Vprintf("f08qvc Example Program Results\n\n");

  /* Skip heading in data file */
  Vscanf("%*[^\n] ");
  Vscanf("%ld%ld%*[^\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_NonUnitDiag, 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                                                         :Values of M and N
 (-6.00,-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)
```

```
(-0.17, 0.09) (-0.07,-0.31) ( 0.27,-0.54) ( 0.35, 1.21)
(-0.93,-0.44) (-0.33,-0.35) ( 0.41,-0.03) ( 0.57, 0.84)
( 0.54, 0.25) (-0.62,-0.05) (-0.52,-0.13) ( 0.11,-0.08)   :End of matrix C
```

9.3 Program Results

```
f08qvc Example Program Results

 Solution matrix X
                     1                   2                   3                   4
 1  (-0.0611, 0.0249)  (-0.0031, 0.0798)  (-0.0062, 0.0165)  ( 0.0054,-0.0063)
 2  ( 0.0215,-0.0003)  (-0.0155, 0.0570)  (-0.0665, 0.0718)  ( 0.0290,-0.2636)
 3  (-0.0949,-0.0785)  (-0.0415,-0.0298)  ( 0.0357, 0.0244)  ( 0.0284, 0.1108)
 4  ( 0.0281, 0.1052)  (-0.0970,-0.1214)  (-0.0271,-0.0940)  ( 0.0402, 0.0048)

 SCALE =   1.00e+00
```