NAG C Library Function Document

nag_dtrsyl (f08qhc)

1 Purpose

nag_dtrsyl (f08qhc) solves the real quasi-triangular Sylvester matrix equation.

2 Specification

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

3 Description

nag_dtrsyl (f08qhc) solves the real Sylvester matrix equation

\[ \text{op}(A)X \pm X\text{op}(B) = \alpha C, \]

where \( \text{op}(A) = A \) or \( A^T \), and the matrices \( A \) and \( B \) are upper quasi-triangular matrices in canonical Schur form (as returned by nag_dhseqr (f08pec)); \( \alpha \) 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  
   \text{Input}  
   \text{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.  
   \text{Constraint:} \ order = Nag_RowMajor \ or \ Nag_ColMajor.  

2: \ trana – Nag_TransType  
   \text{Input}  
   \text{On entry:} specifies the option \text{op}(A) \ as follows:  
   \text{if} \ trana = Nag_NoTrans, \text{ then } \text{op}(A) = A;  
   \text{if} \ trana = Nag_Trans \text{ or } Nag_ConjTrans, \text{ then } \text{op}(A) = A^T.  
   \text{Constraint:} \ trana = Nag_NoTrans, Nag_Trans \text{ or } Nag_ConjTrans.
3: \( \text{tranb} \) – Nag_TransType

*Input*

*On entry:* specifies the option \( \op(B) \) as follows:

- if \( \text{tranb} = \text{Nag\_NoTrans} \), then \( \op(B) = B \);
- if \( \text{tranb} = \text{Nag\_Trans} \) or \( \text{Nag\_ConjTrans} \), then \( \op(B) = B^T \).

*Constraint:* \( \text{tranb} = \text{Nag\_NoTrans}, \text{Nag\_Trans} \) 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 \( \op(A)X + X\op(B) = \alpha C \);
- if \( \text{sign} = \text{Nag\_Minus} \), then the equation is of the form \( \op(A)X - X\op(B) = \alpha C \).

*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: \( a[dim] \) – const double

*Input*

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

If \( order = \text{Nag\_ColMajor} \), the \((i,j)\)th element of the matrix \( A \) is stored in \( a[(j-1) \times pda + i - 1] \) and if \( order = \text{Nag\_RowMajor} \), 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 quasi-triangular matrix \( A \) in canonical Schur form, as returned by \( \text{nag\_dhseqr (f08pec)} \).

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 double

*Input*

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

If \( order = \text{Nag\_ColMajor} \), the \((i,j)\)th element of the matrix \( B \) is stored in \( b[(j-1) \times pdb + i - 1] \) and if \( order = \text{Nag\_RowMajor} \), 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 quasi-triangular matrix \( B \) in canonical Schur form, as returned by \( \text{nag\_dhseqr (f08pec)} \).

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: \( c[dim] \) – double

*Input/Output*

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

\(\text{Input}\)

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, \(pdc \geq \max(1, m)\);
- if order = Nag_RowMajor, \(pdc \geq \max(1, n)\).

13: scale – double *

\(\text{Output}\)

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

14: fail – NagError *

\(\text{Output}\)

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

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

Constraint: \(m \geq 0\).

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

Constraint: \(n \geq 0\).

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

Constraint: \(pda > 0\).

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

Constraint: \(pdb > 0\).

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

Constraint: \(pdc > 0\).

NE_INT_2

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

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

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

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

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

Constraint: \(pdc \geq \max(1, m)\).

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

Constraint: \(pdc \geq \max(1, n)\).

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

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

Then the residual is always small:

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

However, $\hat{X}$ 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:

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

but this may be a considerable overestimate. 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 floating-point operations is approximately $mn(m+n)$.

To solve the general real Sylvester equation

$$AX \pm XB = C,$$

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

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

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

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

where $\hat{X} = Q_1^T X Q_2$ and $\tilde{C} = Q_1^T C Q_2$. $\tilde{C}$ may be computed by matrix multiplication; nag_dtrsyl (f08qhc) may be used to solve the transformed equation; and the solution to the original equation can be obtained as $X = Q_1 \hat{X} Q_2^T$.

The complex analogue of this function is nag_ztrsyl (f08qvc).

9 Example

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

$$A = \begin{pmatrix}
0.10 & 0.50 & 0.68 & -0.21 \\
-0.50 & 0.10 & -0.24 & 0.67 \\
0.00 & 0.00 & 0.19 & -0.35 \\
0.00 & 0.00 & 0.00 & -0.72 \\
\end{pmatrix}, \quad B = \begin{pmatrix}
-0.99 & -0.17 & 0.39 & 0.58 \\
0.00 & 0.48 & -0.84 & -0.15 \\
0.00 & 0.00 & 0.75 & 0.25 \\
0.00 & 0.00 & -0.25 & 0.75 \\
\end{pmatrix}$$

f08qhc
and

\[
C = \begin{pmatrix}
0.63 & -0.56 & 0.08 & -0.23 \\
-0.45 & -0.31 & 0.27 & 1.21 \\
0.20 & -0.35 & 0.41 & 0.84 \\
0.49 & -0.05 & -0.52 & -0.08 \\
\end{pmatrix}
\]

9.1 Program Text

/* nag_dtrsyl (f08qhc) 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 */
    double *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("f08qhc 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, double)) ||
        !(b = NAG_ALLOC(n * m, double)) ||
        !(c = NAG_ALLOC(m * n, double)) )
    {
        Vprintf("Allocation failure\n\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", &A(i,j));
}
Vscanf("%*[\n ] ");
for (i = 1; i <= n; ++i)
{
  for (j = 1; j <= n; ++j)
    Vscanf("%lf", &B(i,j));
}
Vscanf("%*[\n ] ");
for (i = 1; i <= m; ++i)
{
  for (j = 1; j <= n; ++j)
    Vscanf("%lf", &C(i,j));
}
Vscanf("%*[\n ] ");

Reorder the Schur factorization T
f08qhc(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 f08qhc.\n%s\n", fail.message);
  exit_status = 1;
  goto END;
}
Print the solution matrix X stored in C
x04cac(order, Nag_GeneralMatrix, Nag_NonUnitDiag, m, n,
  c, pdc, "Solution matrix X", 0, &fail);
if (fail.code != NE_NOERROR)
{
  Vprintf("Error from x04cac.\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

f08qhc Example Program Data

<table>
<thead>
<tr>
<th>Values of M and N</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 4</td>
</tr>
<tr>
<td>0.10 0.50 0.68 -0.21</td>
</tr>
<tr>
<td>-0.50 0.10 -0.24 0.67</td>
</tr>
<tr>
<td>0.00 0.00 0.19 -0.35</td>
</tr>
<tr>
<td>0.00 0.00 0.00 -0.72</td>
</tr>
<tr>
<td>-0.99 -0.17 0.39 0.58</td>
</tr>
<tr>
<td>0.00 0.48 -0.84 -0.15</td>
</tr>
<tr>
<td>0.00 0.00 0.75 0.25</td>
</tr>
<tr>
<td>0.00 0.00 -0.25 0.75</td>
</tr>
<tr>
<td>0.63 -0.56 0.08 -0.23</td>
</tr>
<tr>
<td>-0.45 -0.31 0.27 1.21</td>
</tr>
<tr>
<td>0.20 -0.35 0.41 0.84</td>
</tr>
<tr>
<td>0.49 -0.05 -0.52 -0.08</td>
</tr>
</tbody>
</table>
### Program Results

**f08qhc 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>-0.4209</td>
<td>0.1764</td>
<td>0.2438</td>
<td>-0.9577</td>
</tr>
<tr>
<td>2</td>
<td>0.5600</td>
<td>-0.8337</td>
<td>-0.7221</td>
<td>0.5386</td>
</tr>
<tr>
<td>3</td>
<td>-0.1246</td>
<td>-0.3392</td>
<td>0.6221</td>
<td>0.8691</td>
</tr>
<tr>
<td>4</td>
<td>-0.2865</td>
<td>0.4113</td>
<td>0.5535</td>
<td>0.3174</td>
</tr>
</tbody>
</table>

SCALE = 1.00e+00