# NAG Library Function Document

# nag_dtrsyl (f08qhc)

## 1 Purpose

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

## 2 Specification

```
#include <nag.h>
#include <nagf08.h>
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

$$\mathrm{op}(A)X \pm X\,\mathrm{op}(B) = \alpha C,$$

where $\mathrm{op}(A) = A$ or $A^{\mathrm{T}}$, 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

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 Arguments

1: **order** – Nag_OrderType *Input*

*On entry*: the **order** argument 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 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

*Constraint*: **order** = Nag_RowMajor or Nag_ColMajor.

2: **trana** – Nag_TransType *Input*

*On entry*: specifies the option $\mathrm{op}(A)$.

**trana** = Nag_NoTrans
$\mathrm{op}(A) = A$.

**trana** = Nag_Trans or Nag_ConjTrans
$\text{op}(A) = A^{\text{T}}$.

*Constraint*: **trana** = Nag_NoTrans, Nag_Trans or Nag_ConjTrans.

3: **tranb** – Nag_TransType *Input*

*On entry*: specifies the option $\text{op}(B)$.

**tranb** = Nag_NoTrans
$\text{op}(B) = B$.

**tranb** = Nag_Trans or Nag_ConjTrans
$\text{op}(B) = B^{\text{T}}$.

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

4: **sign** – Nag_SignType *Input*

*On entry*: indicates the form of the Sylvester equation.

**sign** = Nag_Plus
The equation is of the form $\text{op}(A)X + X\,\text{op}(B) = \alpha C$.

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

*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*: $\mathbf{m} \geq 0$.

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*: $\mathbf{n} \geq 0$.

7: **a**[*dim*] – const double *Input*

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

The $(i,j)$th element of the matrix $A$ is stored in

$\mathbf{a}[(j-1) \times \mathbf{pda} + i - 1]$ when **order** = Nag_ColMajor;
$\mathbf{a}[(i-1) \times \mathbf{pda} + j - 1]$ when **order** = Nag_RowMajor.

*On entry*: the $m$ by $m$ upper quasi-triangular matrix $A$ in canonical Schur form, as returned by nag_dhseqr (f08pec).

8: **pda** – Integer *Input*

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

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

9: **b**[*dim*] – const double *Input*

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

The $(i,j)$th element of the matrix $B$ is stored in

$\mathbf{b}[(j-1) \times \mathbf{pdb} + i - 1]$ when **order** = Nag_ColMajor;
$\mathbf{b}[(i-1) \times \mathbf{pdb} + j - 1]$ when **order** = Nag_RowMajor.

*On entry*: the $n$ by $n$ upper quasi-triangular matrix $B$ in canonical Schur form, as returned by nag_dhseqr (f08pec).

10: **pdb** – Integer *Input*

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

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

11: $\mathbf{c}[dim]$ – double *Input/Output*

**Note**: the dimension, *dim*, of the array **c** must be at least

$\max(1, \mathbf{pdc} \times \mathbf{n})$ when $\mathbf{order} = \text{Nag\_ColMajor}$;
$\max(1, \mathbf{m} \times \mathbf{pdc})$ when $\mathbf{order} = \text{Nag\_RowMajor}$.

The $(i, j)$th element of the matrix $C$ is stored in

$\mathbf{c}[(j-1) \times \mathbf{pdc} + i - 1]$ when $\mathbf{order} = \text{Nag\_ColMajor}$;
$\mathbf{c}[(i-1) \times \mathbf{pdc} + j - 1]$ when $\mathbf{order} = \text{Nag\_RowMajor}$.

*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 row or column elements (depending on the value of **order**) in the array **c**.

*Constraints*:

if $\mathbf{order} = \text{Nag\_ColMajor}$, $\mathbf{pdc} \geq \max(1, \mathbf{m})$;
if $\mathbf{order} = \text{Nag\_RowMajor}$, $\mathbf{pdc} \geq \max(1, \mathbf{n})$.

13: **scale** – double * *Output*

*On exit*: the value of the scale factor $\alpha$.

14: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 3.6 in the Essential Introduction).

# 6 Error Indicators and Warnings

**NE_ALLOC_FAIL**

Dynamic memory allocation failed.

**NE_BAD_PARAM**

On entry, argument ⟨*value*⟩ had an illegal value.

**NE_INT**

On entry, $\mathbf{m} = \langle value \rangle$.
Constraint: $\mathbf{m} \geq 0$.

On entry, $\mathbf{n} = \langle value \rangle$.
Constraint: $\mathbf{n} \geq 0$.

On entry, $\mathbf{pda} = \langle value \rangle$.
Constraint: $\mathbf{pda} > 0$.

On entry, $\mathbf{pdb} = \langle value \rangle$.
Constraint: $\mathbf{pdb} > 0$.

On entry, **pdc** $= \langle value \rangle$.
Constraint: **pdc** $> 0$.

**NE_INT_2**

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

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

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

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

**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 contact NAG for assistance.

**NE_PERTURBED**

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

## 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:

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

Then the residual is always small:

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

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:

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

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 9.

## 8 Parallelism and Performance

nag_dtrsyl (f08qhc) is not threaded by NAG in any implementation.

nag_dtrsyl (f08qhc) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the Users' Note for your implementation for any additional implementation-specific information.

## 9 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 :

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

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:

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

where $\tilde{X} = Q_1^{\mathrm{T}} X Q_2$ and $\tilde{C} = Q_1^{\mathrm{T}} C Q_2$. $\tilde{C}$ 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 $X = Q_1 \tilde{X} Q_2^{\mathrm{T}}$.

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

## 10 Example

This example solves the Sylvester equation $AX + XB = C$, where

$$A = \begin{pmatrix} 0.10 & 0.50 & 0.68 & -0.21 \\ -0.50 & 0.10 & -0.24 & 0.67 \\ 0.00 & 0.00 & 0.19 & -0.35 \\ 0.00 & 0.00 & 0.00 & -0.72 \end{pmatrix},$$

$$B = \begin{pmatrix} -0.99 & -0.17 & 0.39 & 0.58 \\ 0.00 & 0.48 & -0.84 & -0.15 \\ 0.00 & 0.00 & 0.75 & 0.25 \\ 0.00 & 0.00 & -0.25 & 0.75 \end{pmatrix}$$

and

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

### 10.1 Program Text

```
/* nag_dtrsyl (f08qhc) Example Program.
 *
 * Copyright 2001 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 */

#include <stdio.h>
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf08.h>
#include <nagf16.h>
#include <nagx04.h>
#include <nagx02.h>

int main(void)
{
  /* Scalars */
  Integer       i, j, m, n, pda, pdb, pdc, pdd, pde, pdf;
```

```
  Integer       exit_status = 0;
  double        alpha, beta, norm, scale;
  Nag_SignType  sign = Nag_Minus;
  NagError      fail;
  Nag_OrderType order;
  /* Arrays */
  double        *a = 0, *b = 0, *c = 0, *d = 0, *e = 0, *f = 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]
#define D(I, J) d[(J-1)*pdd + I - 1]
#define E(I, J) e[(J-1)*pde + I - 1]
#define F(I, J) f[(J-1)*pdf + 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]
#define D(I, J) d[(I-1)*pdd + J - 1]
#define E(I, J) e[(I-1)*pde + J - 1]
#define F(I, J) f[(I-1)*pdf + J - 1]
  order = Nag_RowMajor;
#endif

  INIT_FAIL(fail);

  printf("nag_dtrsyl (f08qhc) Example Program Results\n\n");

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

  /* Allocate memory */
  if (!(a = NAG_ALLOC(m * m, double)) ||
      !(b = NAG_ALLOC(n * m, double)) ||
      !(c = NAG_ALLOC(m * n, double)) ||
      !(d = NAG_ALLOC(m * n, double)) ||
      !(e = NAG_ALLOC(m * n, double)) ||
      !(f = NAG_ALLOC(m * n, double))
      )
    {
      printf("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)
        scanf("%lf", &A(i, j));
    }
  scanf("%*[^\n] ");
  for (i = 1; i <= n; ++i)
```

```
    {
      for (j = 1; j <= n; ++j)
        scanf("%lf", &B(i, j));
    }
  scanf("%*[^\n] ");
  for (i = 1; i <= m; ++i)
    {
      for (j = 1; j <= n; ++j)
        scanf("%lf", &C(i, j));
    }
  scanf("%*[^\n] ");

  /* Copy C into F */
  for (i = 1; i <= m; ++i)
    {
      for (j = 1; j <= m; ++j)
        F(i, j) = C(i, j);
    }

  /* nag_gen_real_mat_print (x04cac): Print Matrix C. */
  fflush(stdout);
  nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, m, n,
                         c, pdc, "Matrix C", 0, &fail);
  printf("\n");
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_gen_real_mat_print (x04cac).\n%s\n",
             fail.message);
      exit_status = 1;
      goto END;
    }

  /* Reorder the Schur factorization T */
  /* nag_dtrsyl (f08qhc).
   * Solve real Sylvester matrix equation AX + XB = C, A and B
   * are upper quasi-triangular or transposes
   */
  nag_dtrsyl(order, Nag_NoTrans, Nag_NoTrans, sign, m, n, a, pda,
             b, pdb, c, pdc, &scale, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_dtrsyl (f08qhc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

  /* nag_dgemm (f16yac): Compute aC - (A*X + X*B*sign) from solution*/
  /* and store in matrix E*/
  alpha = 1.0;
  beta = 0.0;
  nag_dgemm(order, Nag_NoTrans, Nag_NoTrans, m, n, m, alpha, a, pda,
            c, pdc, beta, d, pdd, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_dgemm (f16yac).\n%s\n",
             fail.message);
      exit_status = 1;
      goto END;
    }
  if(sign == Nag_Minus)
    alpha = -1.0;
  else
    alpha = 1.0;
  beta = 1.0;
  nag_dgemm(order, Nag_NoTrans, Nag_NoTrans, m, n, n, alpha, c, pdc, b,
            pdb, beta, d, pdd, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_dgemm (f16yac).\n%s\n",
              fail.message);
      exit_status = 1;
```

```
        goto END;
      }
    for(i=1; i<=m; i++)
      {
        for (j=1; j<=n; j++)
          E(i, j) = scale * F(i, j) - D(i, j);
      }

    /* nag_dge_norm (f16rac): Find norm of matrix E and print warning if */
    /* it is too large */
    nag_dge_norm(order, Nag_OneNorm, n, n, e, pde, &norm, &fail);
    if (fail.code != NE_NOERROR)
      {
        printf("Error from nag_dge_norm (f16rac).\n%s\n",
                fail.message);
        exit_status = 1;
        goto END;
      }
    if (norm > pow(x02ajc(),0.8))
      {
        printf("%s\n%s\n","Norm of aC - (A*X + X*B*sign) is much greater than 0.",
               "nag_dtrsyl (f08qhc) has failed.");
      }
    else
      {
        printf(" SCALE = %11.2e\n", scale);
      }
   END:
    NAG_FREE(a);
    NAG_FREE(b);
    NAG_FREE(c);
    NAG_FREE(d);
    NAG_FREE(e);
    NAG_FREE(f);

    return exit_status;
}
```

## 10.2 Program Data

```
nag_dtrsyl (f08qhc) Example Program Data
  4  4                              :Values of M and N
  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 of matrix A
 -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 of matrix B
  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 of matrix C
```

## 10.3 Program Results

```
nag_dtrsyl (f08qhc) Example Program Results

 Matrix C
             1          2          3          4
 1      0.6300    -0.5600     0.0800    -0.2300
 2     -0.4500    -0.3100     0.2700     1.2100
 3      0.2000    -0.3500     0.4100     0.8400
 4      0.4900    -0.0500    -0.5200    -0.0800

 SCALE =    1.00e+00
```