NAG Toolbox for MATLAB

f08yv

1 Purpose

f08yv solves the generalized complex triangular Sylvester equations.

2 Syntax

```
[c, f, dif, scale, info] = f08yv(trans, ijob, a, b, c, d, e, f, 'm', m, 'n', n)
```

3 Description

f08yv solves either the generalized complex Sylvester equations

$$\begin{aligned} AR - LB &= \alpha C \\ DR - LE &= \alpha F\text{'} \end{aligned} \tag{1}$$

or the equations

$$\begin{aligned} A^{\mathrm{H}}R + D^{\mathrm{H}}L &= \alpha C \\ RB^{\mathrm{H}} + LE^{\mathrm{H}} &= -\alpha F\text{'} \end{aligned} \tag{2}$$

where the pair (A, D) are given m by m matrices in generalized Schur form, (B, E) are given n by n matrices in generalized Schur form and (C, F) are given m by n matrices. The pair (R, L) are the m by n solution matrices, and α is an output scaling factor determined by the function to avoid overflow in computing (R, L).

Equations (1) are equivalent to equations of the form

$$Zx = \alpha b,$$

where

$$Z = \begin{pmatrix} I \otimes A - B^{\mathrm{H}} \otimes I \\ I \otimes D - E^{\mathrm{H}} \otimes I \end{pmatrix}$$

and $\otimes$ is the Kronecker product. Equations (2) are then equivalent to

$$Z^{\mathrm{H}}y = \alpha b.$$

The pair (S, T) are in generalized Schur form if S and T are upper triangular as returned, for example, by f08xn, or f08xs with **job** = 'S'.

Optionally, the function estimates $\mathrm{Dif}[(A, D), (B, E)]$, the separation between the matrix pairs (A, D) and (B, E), which is the smallest singular value of Z. The estimate can be based on either the Frobenius norm, or the one norm. The one norm estimate can be three to ten times more expensive than the Frobenius norm estimate, but makes the condition estimation uniform with the nonsymmetric eigenproblem. The Frobenius norm estimate provides a low cost, but equally reliable estimate. For more information see Sections 2.4.8.3 and 4.11.1.3 of Anderson *et al.* 1999 and K*a*gström and Poromaa 1996.

4 References

Anderson E, Bai Z, Bischof C, Blackford S, Demmel J, Dongarra J J, Du Croz J J, Greenbaum A, Hammarling S, McKenney A and Sorensen D 1999 *LAPACK Users' Guide* (3rd Edition) SIAM, Philadelphia URL: http://www.netlib.org/lapack/lug

K*a*gström B 1994 A perturbation analysis of the generalized Sylvester equation $(AR - LB, DR - LE) = (c, F)$ *SIAM J. Matrix Anal. Appl.* **15** 1045–1060

Kågström B and Poromaa P 1996 LAPACK-style algorithms and software for solving the generalized Sylvester equation and estimating the separation between regular matrix pairs *ACM Trans. Math. Software* **22** 78–103

5 Parameters

5.1 Compulsory Input Parameters

1: **trans – string**

If **trans** = 'N', solve the generalized Sylvester equation (1).

If **trans** = 'C', solve the 'conjugate transposed' system (2).

Constraint: **trans** = 'N' or 'C'.

2: **ijob – int32 scalar**

Specifies what kind of functionality is to be performed when **trans** = 'N'.

ijob = 0

Solve (1) only.

ijob = 1

The functionality of **ijob** = 0 and 3.

ijob = 2

The functionality of **ijob** = 0 and 4.

ijob = 3

Only an estimate of $\mathrm{Dif}[(A,D),(B,E)]$ is computed based on the Frobenius norm.

ijob = 4

Only an estimate of $\mathrm{Dif}[(A,D),(B,E)]$ is computed based on the one norm.

If **trans** = 'C', **ijob** is not referenced.

3: **a**(**lda**,$*$) **– complex array**

The first dimension of the array **a** must be at least $\max(1,\mathbf{m})$

The second dimension of the array must be at least $\max(1,\mathbf{m})$

The upper triangular matrix A.

4: **b**(**ldb**,$*$) **– complex array**

The first dimension of the array **b** must be at least $\max(1,\mathbf{n})$

The second dimension of the array must be at least $\max(1,\mathbf{n})$

The upper triangular matrix B.

5: **c**(**ldc**,$*$) **– complex array**

The first dimension of the array **c** must be at least $\max(1,\mathbf{m})$

The second dimension of the array must be at least $\max(1,\mathbf{n})$

Contains the right-hand-side matrix C.

6: **d**(**ldd**,$*$) **– complex array**

The first dimension of the array **d** must be at least $\max(1,\mathbf{m})$

The second dimension of the array must be at least $\max(1,\mathbf{m})$

The upper triangular matrix D.

7: **e(lde,∗) – complex array**

The first dimension of the array **e** must be at least $\max(1, \mathbf{n})$

The second dimension of the array must be at least $\max(1, \mathbf{n})$

The upper triangular matrix E.

8: **f(ldf,∗) – complex array**

The first dimension of the array **f** must be at least $\max(1, \mathbf{m})$

The second dimension of the array must be at least $\max(1, \mathbf{n})$

Contains the right-hand side matrix F.

5.2 Optional Input Parameters

1: **m – int32 scalar**

Default: The second dimension of the array **a** The second dimension of the array **d**.

m, the order of the matrices A and D, and the row dimension of the matrices C, F, R and L.

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

2: **n – int32 scalar**

Default: The second dimension of the array **b** The second dimension of the array **c** The second dimension of the array **e** The second dimension of the array **f**.

n, the order of the matrices B and E, and the column dimension of the matrices C, F, R and L.

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

5.3 Input Parameters Omitted from the MATLAB Interface

lda, ldb, ldc, ldd, lde, ldf, work, lwork, iwork

5.4 Output Parameters

1: **c(ldc,∗) – complex array**

The first dimension of the array **c** must be at least $\max(1, \mathbf{m})$

The second dimension of the array must be at least $\max(1, \mathbf{n})$

If **trans** = 'C' or 'N' and **ijob** = 0, 1 or 2, **c** contains the solution matrix R.

If **trans** = 'N' and **ijob** = 3 or 4, **c** holds R, the solution achieved during the computation of the Dif estimate.

2: **f(ldf,∗) – complex array**

The first dimension of the array **f** must be at least $\max(1, \mathbf{m})$

The second dimension of the array must be at least $\max(1, \mathbf{n})$

If **trans** = 'C' or 'N' and **ijob** = 0, 1 or 2, **f** contains the solution matrix L.

If **trans** = 'N' and **ijob** = 3 or 4, **f** holds L, the solution achieved during the computation of the Dif estimate.

3: **dif – double scalar**

The estimate of Dif. If **trans** = 'C' or 'N' and **ijob** = 0, **dif** is not referenced.

4: **scale – double scalar**

α, the scaling factor in (1) or (2).

If $0 <$ **scale** < 1, **c** and **f** hold the solutions R and L, respectively, to a slightly perturbed system but the input arrays **a**, **b**, **d** and **e** have not been changed.

If **scale** $= 0$, **c** and **f** hold the solutions R and L, respectively, to the homogeneous system with $C = F = 0$. In this case **dif** is not referenced.

Normally, **scale** $= 1$.

5: **info – int32 scalar**

info $= 0$ unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

info $= -i$

If **info** $= -i$, parameter i had an illegal value on entry. The parameters are numbered as follows:

1: **trans**, 2: **ijob**, 3: **m**, 4: **n**, 5: **a**, 6: **lda**, 7: **b**, 8: **ldb**, 9: **c**, 10: **ldc**, 11: **d**, 12: **ldd**, 13: **e**, 14: **lde**, 15: **f**, 16: **ldf**, 17: **dif**, 18: **scale**, 19: **work**, 20: **lwork**, 21: **iwork**, 22: **info**.

It is possible that **info** refers to a parameter that is omitted from the MATLAB interface. This usually indicates that an error in one of the other input parameters has caused an incorrect value to be inferred.

info > 0

(A, D) and (B, E) have common or close eigenvalues and so no solution could be computed.

7 Accuracy

See K*a*gström 1994 for a perturbation analysis of the generalized Sylvester equation.

8 Further Comments

The total number of floating point operations needed to solve the generalized Sylvester equations is approximately $8mn(n + m)$. The Frobenius norm estimate of dif does not require additional significant computation, but the one norm estimate is typically five times more expensive.

The real analogue of this function is f08yh.

9 Example

```
trans = 'No transpose';
ijob = int32(0);
a = [complex(4, +4), complex(1, +1), complex(1, +1), complex(2, -1);
     complex(0, +0), complex(2, +1), complex(1, +1), complex(1, +1);
     complex(0, +0), complex(0, +0), complex(2, -1), complex(1, +1);
     complex(0, +0), complex(0, +0), complex(0, +0), complex(6, -2)];
b = [complex(2, +0), complex(1, +1), complex(1, +1), complex(3, -1);
     complex(0, +0), complex(1, +0), complex(2, +1), complex(1, +1);
     complex(0, +0), complex(0, +0), complex(1, +0), complex(1, +1);
     complex(0, +0), complex(0, +0), complex(0, +0), complex(2, +0)];
c = [complex(-13, +9), complex(2, +8), complex(-2, +8), complex(-2, +5);
     complex(-9, -1), complex(0, +5), complex(-7, -3), complex(-6, +0);
     complex(-1, +1), complex(4, +2), complex(4, -5), complex(9, -5);
     complex(-6, +6), complex(9, +1), complex(-2, +4), complex(22, -8)];
```

```
d = [complex(1, +1), complex(1, -1), complex(1, +1), complex(1, -1);
     complex(0, +0), complex(6, -4), complex(1, -1), complex(1, +1);
     complex(0, +0), complex(0, +0), complex(2, +4), complex(1, -1);
     complex(0, +0), complex(0, +0), complex(0, +0), complex(2, +3)];
e = [complex(1, +0), complex(1, +1), complex(1, -1), complex(1, +1);
     complex(0, +0), complex(2, +0), complex(1, +1), complex(1, -1);
     complex(0, +0), complex(0, +0), complex(2, +0), complex(1, +1);
     complex(0, +0), complex(0, +0), complex(0, +0), complex(1, +0)];
f = [complex(-6, +5), complex(4, -4), complex(-3, +11), complex(3, -7);
     complex(-5, +11), complex(12, -4), complex(-2, +2), complex(0, +14);
     complex(-5, -1), complex(0, +4), complex(-2, +10), complex(3, -1);
     complex(-6, -2), complex(1, +1), complex(-7, -3), complex(4, +7)];
[cOut, fOut, dif, scale, info] = f08yv(trans, ijob, a, b, c, d, e, f)
```

```
cOut =
    1.0000 + 1.0000i    1.0000 + 1.0000i    1.0000 + 1.0000i    1.0000 +
1.0000i
   -1.0000 + 1.0000i    2.0000 + 1.0000i   -1.0000 + 1.0000i   -1.0000 +
1.0000i
   -1.0000 + 1.0000i    1.0000 + 1.0000i    3.0000 + 1.0000i    1.0000 +
1.0000i
   -1.0000 + 1.0000i    1.0000 + 1.0000i   -1.0000 + 1.0000i    4.0000 +
1.0000i
fOut =
    4.0000 + 1.0000i   -1.0000 + 1.0000i    1.0000 + 1.0000i   -1.0000 +
1.0000i
    1.0000 + 1.0000i    3.0000 + 1.0000i   -1.0000 + 1.0000i    1.0000 +
1.0000i
   -1.0000 + 1.0000i    1.0000 + 1.0000i    2.0000 + 1.0000i   -1.0000 +
1.0000i
    1.0000 + 1.0000i   -1.0000 + 1.0000i    1.0000 + 1.0000i    1.0000 +
1.0000i
dif =
     1
scale =
     0
info =
           0
```