NAG Toolbox for MATLAB

f08ys

1 Purpose

f08ys computes the generalized singular value decomposition (GSVD) of two complex upper trapezoidal matrices A and B, where A is an m by n matrix and B is a p by n matrix.

A and B are assumed to be in the form returned by f08vs.

2 Syntax

```
[a, b, alpha, beta, u, v, q, ncycle, info] = f08ys(jobu, jobv, jobq, k,
l, a, b, tola, tolb, u, v, q, 'm', m, 'p', p, 'n', n)
```

3 Description

f08ys computes the GSVD of the matrices A and B which are assumed to have the form as returned by f08vs

$$A = \begin{cases} \begin{array}{r} \\ k \\ l \\ m-k-l \end{array} \begin{array}{c} \begin{array}{ccc} n-k-l & k & l \end{array} \\ \begin{pmatrix} 0 & A_{12} & A_{13} \\ 0 & 0 & A_{23} \\ 0 & 0 & 0 \end{pmatrix} \end{array}, & \text{if } m-k-l \geq 0; \\ \begin{array}{r} \\ k \\ m-k \end{array} \begin{array}{c} \begin{array}{ccc} n-k-l & k & l \end{array} \\ \begin{pmatrix} 0 & A_{12} & A_{13} \\ 0 & 0 & A_{23} \end{pmatrix} \end{array}, & \text{if } m-k-l < 0; \end{cases}$$

$$B = \begin{array}{r} \\ l \\ p-l \end{array} \begin{array}{c} \begin{array}{ccc} n-k-l & k & l \end{array} \\ \begin{pmatrix} 0 & 0 & B_{13} \\ 0 & 0 & 0 \end{pmatrix} \end{array},$$

where the k by k matrix A_{12} and the l by l matrix B_{13} are nonsingular upper triangular, A_{23} is l by l upper triangular if $m-k-l \geq 0$ and is $(m-k)$ by l upper trapezoidal otherwise.

f08ys computes unitary matrices Q, U and V, diagonal matrices D_1 and D_2, and an upper triangular matrix R such that

$$U^{\mathrm{H}} A Q = D_1 \begin{pmatrix} 0 & R \end{pmatrix}, \qquad V^{\mathrm{H}} B Q = D_2 \begin{pmatrix} 0 & R \end{pmatrix}.$$

Optionally Q, U and V may or may not be computed, or they may be premultiplied by matrices Q_1, U_1 and V_1 respectively.

If $(m-k-l) \geq 0$ then D_1, D_2 and R have the form

$$D_1 = \begin{array}{r} \\ k \\ l \\ m-k-l \end{array} \begin{array}{c} \begin{array}{cc} k & l \end{array} \\ \begin{pmatrix} I & 0 \\ 0 & C \\ 0 & 0 \end{pmatrix} \end{array},$$

$$D_2 = \begin{array}{r} \\ l \\ p-l \end{array} \begin{array}{c} \begin{array}{cc} k & l \end{array} \\ \begin{pmatrix} 0 & S \\ 0 & 0 \end{pmatrix} \end{array},$$

$$R = \begin{array}{c} \\ k \\ l \end{array} \begin{array}{c} \begin{array}{cc} k & l \end{array} \\ \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} \end{array},$$

where $C = \mathrm{diag}(\alpha_{k+1}, , , \ldots , , , \alpha_{k+l})$, $S = \mathrm{diag}(\beta_{k+1}, , , \ldots , , , \beta_{k+l})$.

If $(m-k-l) < 0$ then D_1, D_2 and R have the form

$$D_1 = \begin{array}{c} \\ k \\ m-k \end{array} \begin{array}{c} \begin{array}{ccc} k & m-k & k+l-m \end{array} \\ \begin{pmatrix} I & 0 & 0 \\ 0 & C & 0 \end{pmatrix} \end{array},$$

$$D_2 = \begin{array}{r} \\ m-k \\ k+l-m \\ p-l \end{array} \begin{array}{c} \begin{array}{ccc} k & m-k & k+l-m \end{array} \\ \begin{pmatrix} 0 & S & 0 \\ 0 & 0 & I \\ 0 & 0 & 0 \end{pmatrix} \end{array},$$

$$R = \begin{array}{r} \\ k \\ m-k \\ k+l-m \end{array} \begin{array}{c} \begin{array}{ccc} k & m-k & k+l-m \end{array} \\ \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ 0 & R_{22} & R_{23} \\ 0 & 0 & R_{33} \end{pmatrix} \end{array},$$

where $C = \mathrm{diag}(\alpha_{k+1}, , , \ldots , , , \alpha_m)$, $S = \mathrm{diag}(\beta_{k+1}, , , \ldots , , , \beta_m)$.

In both cases the diagonal matrix C has real nonnegative diagonal elements, the diagonal matrix S has real positive diagonal elements, so that S is nonsingular, and $C^2 + S^2 = 1$. See Section 2.3.5.3 of Anderson *et al.* 1999 for further information.

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

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

5 Parameters

5.1 Compulsory Input Parameters

1: **jobu – string**

If **jobu** = 'U', **u** must contain a unitary matrix U_1 on entry, and the product $U_1 U$ is returned.

If **jobu** = 'I', **u** is initialized to the unit matrix, and the unitary matrix U is returned.

If **jobu** = 'N', U is not computed.

Constraint: **jobu** = 'I', 'U' or 'N'.

2: **jobv – string**

If **jobv** = 'V', **v** must contain a unitary matrix V_1 on entry, and the product $V_1 V$ is returned.

If **jobv** = 'I', **v** is initialized to the unit matrix, and the unitary matrix V is returned.

If **jobv** = 'N', V is not computed.

Constraint: **jobv** = 'V' or 'N'.

3: **jobq – string**

If **jobq** = 'Q', **q** must contain a unitary matrix Q_1 on entry, and the product Q_1Q is returned.

If **jobq** = 'I', **q** is initialized to the unit matrix, and the unitary matrix Q is returned.

If **jobq** = 'N', Q is not computed.

Constraint: **jobq** = 'Q' or 'N'.

4: **k – int32 scalar**

5: **l – int32 scalar**

k and **l** specify the sizes, k and l, of the subblocks of A and B, whose GSVD is to be computed by f08ys.

6: **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 $\max(1, \mathbf{n})$

The m by n matrix A.

7: **b(ldb,∗) – complex array**

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

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

The p by n matrix B.

8: **tola – double scalar**

9: **tolb – double scalar**

tola and **tolb** are the convergence criteria for the Jacobi-Kogbetliantz iteration procedure. Generally, they should be the same as used in the preprocessing step performed by f08vs, say

$$\mathbf{tola} = \max(\mathbf{m}, \mathbf{n})\|A\|\epsilon,$$
$$\mathbf{tolb} = \max(\mathbf{p}, \mathbf{n})\|B\|\epsilon,$$

where ϵ is the ***machine precision***.

10: **u(ldu,∗) – complex array**

The first dimension, **ldu**, of the array **u** must satisfy

if **jobu** = 'U', **ldu** $\geq \max(1, \mathbf{m})$;
ldu ≥ 1 otherwise.

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

If **jobu** = 'U', **u** must contain an m by m matrix U_1 (usually the unitary matrix returned by f08vs).

11: **v(ldv,∗) – complex array**

The first dimension, **ldv**, of the array **v** must satisfy

if **jobv** = 'V', **ldv** $\geq \max(1, \mathbf{p})$;
ldv ≥ 1 otherwise.

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

If **jobv** = 'V', **v** must contain an p by p matrix V_1 (usually the unitary matrix returned by f08vs).

12: **q**(**ldq**,∗) – **complex array**

The first dimension, **ldq**, of the array **q** must satisfy

if **jobq** = 'Q', **ldq** ≥ max(1, **n**);
ldq ≥ 1 otherwise.

The second dimension of the array must be at least max(1, **n**)

If **jobq** = 'Q', **q** must contain an n by n matrix Q_1 (usually the unitary matrix returned by f08vs).

5.2 Optional Input Parameters

1: **m** – **int32 scalar**

Default: The first dimension of the array **a**.

m, the number of rows of the matrix A.

Constraint: **m** ≥ 0.

2: **p** – **int32 scalar**

Default: The first dimension of the array **b**.

p, the number of rows of the matrix B.

Constraint: **p** ≥ 0.

3: **n** – **int32 scalar**

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

n, the number of columns of the matrices A and B.

Constraint: **n** ≥ 0.

5.3 Input Parameters Omitted from the MATLAB Interface

lda, ldb, ldu, ldv, ldq, work

5.4 Output Parameters

1: **a**(**lda**,∗) – **complex array**

The first dimension of the array **a** must be at least max(1, **m**)

The second dimension of the array must be at least max(1, **n**)

If $m-k-l \geq 0$, $\mathbf{a}(1:, k+l, n-k-l+1:, n)$ contains the $(k+l)$ by $(k+l)$ upper triangular matrix R.

If $m-k-l < 0$, $\mathbf{a}(1:, m, n-k-l+1:, n)$ contains the first m rows of the $(k+l)$ by $(k+l)$ upper triangular matrix R, and the submatrix R_{33} is returned in $\mathbf{b}(m-k+1:, l, n+m-k-l+1:, n)$.

2: **b**(**ldb**,∗) – **complex array**

The first dimension of the array **b** must be at least max(1, **p**)

The second dimension of the array must be at least max(1, **n**)

If $m-k-l < 0$, $\mathbf{b}(m-k+1:l, n+m-k-l+1:n)$ contains the submatrix R_{33} of R.

3: **alpha**(∗) – **double array**

Note: the dimension of the array **alpha** must be at least max(1, **n**).

See the description of **beta**.

4: **beta**$(*)$ **– double array**

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

alpha and **beta** contain the generalized singular value pairs of A and B:

alpha$(i) = 1$, **beta**$(i) = 0$, for $i = 1, 2, \ldots, k$, and
if $m - k - l \geq 0$, **alpha**$(i) = \alpha_i$, **beta**$(i) = \beta_i$, for $i = k + 1, k + 2, \ldots, k + l$, or
if $m - k - l < 0$, **alpha**$(i) = \alpha_i$, **beta**$(i) = \beta_i$, for $i = k + 1, k + 2, \ldots, m$ and **alpha**$(i) = 0$, **beta**$(i) = 1$, for $i = m + 1, m + 2, \ldots, k + l$.

Furthermore, if $k + l < n$, **alpha**$(i) =$ **beta**$(i) = 0$, for $i = k + l + 1, k + l + 2, \ldots, n$.

5: **u**(**ldu**,$*$) **– complex array**

The first dimension, **ldu**, of the array **u** must satisfy

if **jobu** = 'U', **ldu** $\geq \max(1, \mathbf{m})$;
ldu ≥ 1 otherwise.

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

If **jobu** = 'I', **u** contains the unitary matrix U.

If **jobu** = 'U', **u** contains the product $U_1 U$.

If **jobu** = 'N', **u** is not referenced.

6: **v**(**ldv**,$*$) **– complex array**

The first dimension, **ldv**, of the array **v** must satisfy

if **jobv** = 'V', **ldv** $\geq \max(1, \mathbf{p})$;
ldv ≥ 1 otherwise.

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

If **jobv** = 'I', **v** contains the unitary matrix V.

If **jobv** = 'V', **v** contains the product $V_1 V$.

If **jobv** = 'N', **v** is not referenced.

7: **q**(**ldq**,$*$) **– complex array**

The first dimension, **ldq**, of the array **q** must satisfy

if **jobq** = 'Q', **ldq** $\geq \max(1, \mathbf{n})$;
ldq ≥ 1 otherwise.

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

If **jobq** = 'I', **q** contains the unitary matrix Q.

If **jobq** = 'Q', **q** contains the product $Q_1 Q$.

If **jobq** = 'N', **q** is not referenced.

8: **ncycle – int32 scalar**

The number of cycles required for convergence.

9: **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: **jobu**, 2: **jobv**, 3: **jobq**, 4: **m**, 5: **p**, 6: **n**, 7: **k**, 8: **l**, 9: **a**, 10: **lda**, 11: **b**, 12: **ldb**, 13: **tola**, 14: **tolb**, 15: **alpha**, 16: **beta**, 17: **u**, 18: **ldu**, 19: **v**, 20: **ldv**, 21: **q**, 22: **ldq**, 23: **work**, 24: **ncycle**, 25: **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 $= 1$

The procedure does not converge after 40 cycles.

7 Accuracy

The computed generalized singular value decomposition is nearly the exact generalized singular value decomposition for nearby matrices $(A + E)$ and $(B + F)$, where

$$\|E\|_2 = O\epsilon\|A\|_2 \qquad \text{and} \qquad \|F\|_2 = O\epsilon\|B\|_2,$$

and ϵ is the ***machine precision***. See Section 4.12 of Anderson *et al.* 1999 for further details.

8 Further Comments

The real analogue of this function is f08ye.

9 Example

```
jobu = 'U';
jobv = 'V';
jobq = 'Q';
k = int32(2);
l = int32(2);
a  =  [complex(-2.711752201068527,  +0),  complex(-1.438958913156753,  -
1.031547056142432), ...
          complex(-0.1054299872559687,  +1.317598266756125),  complex(-
0.3924031110146076, -0.1950399449446725);
          complex(0,  +0),  complex(-1.858321263186379,  +0),  complex(-
0.9452914456581194, ...
                    +0.1927945636773471),    complex(1.43554017705668,
+0.2631257591989612);
       complex(0, +0), complex(0, +0), complex(2.907898496201423, +0),
complex(-0.239455444941618, +0.1885561529206693);
          complex(0,  +0),  complex(0,  +0),  complex(0,  +0),  complex(-
1.575857265213969, +0);
     complex(0, +0), complex(0, +0), complex(0, +0), complex(0, +0);
     complex(0, +0), complex(0, +0), complex(0, +0), complex(0, +0)];
b  =  [complex(0,  +0),  complex(0,  +0),  complex(1.414213562373095,  +0),
complex(0, +0);
               complex(0,   +0),   complex(0,   +0),   complex(0,   +0),
complex(1.414213562373095, +0)];
tola = 4.572525929912148e-15;
tolb = 4.445228907190568e-16;
u   =   [complex(-0.01303782072911974,   -0.325945518227994),   complex(-
0.140394835983486, ...
                -0.2616689404621504),    complex(0.2435736556563269,    -
0.7795555232315882), ...
          complex(-0.07400675774617811,  -0.27822662427074),  complex(-
```

```
0.04594698328726612, ...
            +0.0001405197391311852),   complex(-0.05277254157452753,   -
0.2249224525591382);
                     complex(0.4276405199151281,    -0.6258153949977484),
complex(0.08629802995020347, ...
                    -0.0381735742243086),     complex(-0.320349621724286,
+0.1447524795146413), ...
           complex(0.1074014812379654,   +0.188238055274002),   complex(-
0.08031059739194546, ...
                -0.4360450898200673),    complex(-0.0381165318571037,    -
0.2190669278274417);
                     complex(-0.325945518227994,    +0.164276541186909),
complex(0.3816303904390148, ...
                -0.1821887659659533),    complex(0.1721662316582284,    -
0.001400939408589397), ...
                    complex(-0.4976983263285567,     +0.1782570459454953),
complex(0.0597139539115568, ...
                -0.5897441349396421),    complex(-0.1384983195551851,    -
0.09094051842148129);
          complex(0.1590614128952611,  -0.005215128291647905),  complex(-
0.2820697791776328, ...
                     +0.1973200481868614),     complex(0.253068199443448,
+0.1905281836917221), ...
          complex(-0.3779444174129674,   +0.2681556290123255),   complex(-
0.04644286251388844, ...
                +0.3086446465209617),    complex(-0.3735446784515351,    -
0.5514759261689023);
          complex(-0.1720992336243808,  -0.01303782072911973),  complex(-
0.5094241468123243, ...
                    -0.5031861402367152),     complex(0.03205679660303476,
+0.1835804465941344), ...
                     complex(0.2042228100553802,     +0.1660110081059865),
complex(0.5784271028571478, ...
               -0.1243936858522373),    complex(-0.01881481701367883,    -
0.05568560546891608);
           complex(-0.2633639787282192,  -0.2477185938532755),  complex(-
0.1086101552997524, ...
                +0.2847377773945727),    complex(0.1414218525561267,    -
0.1570689808307628), ...
                    complex(-0.0873351716364392,     +0.5468287081721547),
complex(0.01576327345977116, ...
                    +0.04712962008740441),     complex(0.6500687733214622,
+0.004917308707417217)];
v = [complex(1, +0), complex(0, +0);
     complex(0, +0), complex(1, +0)];
q     =     [complex(0.7071067811865475,     +0),     complex(0,     +0),
complex(0.7071067811865476, +0), complex(0, +0);
       complex(0, +0), complex(0.7071067811865475, +0), complex(0, +0),
complex(0.7071067811865476, +0);
          complex(0.7071067811865476,  +0),  complex(0,  +0),  complex(-
0.7071067811865475, +0), complex(0, +0);
       complex(0, +0), complex(0.7071067811865476, +0), complex(0, +0),
complex(-0.7071067811865475, +0)];
[aOut, bOut, alpha, beta, uOut, vOut, qOut, ncycle, info] = ...
    f08ys(jobu, jobv, jobq, k, l, a, b, tola, tolb, u, v, q)
```

```
aOut =
   -2.7118              -1.4390 - 1.0315i   -0.0769 + 1.3613i   -0.2814 -
0.0324i
         0              -1.8583              -1.0760 + 0.0310i    1.3292 +
0.3677i
         0                    0            3.2537                    0
         0                    0                 0              -2.1084
         0                    0                 0                    0
         0                    0                 0                    0
bOut =
         0         0    1.4142         0
         0         0         0   -1.4142
alpha =
    1.0000
```

```
    1.0000
    0.9006
    0.7417
beta =
         0
         0
    0.4346
    0.6707
uOut =
  Columns 1 through 4
   -0.0130 - 0.3259i   -0.1404 - 0.2617i    0.2518 - 0.7979i   -0.0510 -
0.2175i
    0.4276 - 0.6258i    0.0863 - 0.0382i   -0.3219 + 0.1611i    0.1198 +
0.1632i
   -0.3259 + 0.1643i    0.3816 - 0.1822i    0.1323 - 0.0146i   -0.5067 +
0.1862i
    0.1591 - 0.0052i   -0.2821 + 0.1973i    0.2160 + 0.1881i   -0.4016 +
0.2679i
   -0.1721 - 0.0130i   -0.5094 - 0.5032i    0.0365 + 0.2032i    0.1927 +
0.1557i
   -0.2634 - 0.2477i   -0.1086 + 0.2847i    0.1091 - 0.1271i   -0.0882 +
0.5617i
  Columns 5 through 6
  -0.0459 + 0.0001i  -0.0528 - 0.2249i
  -0.0803 - 0.4360i  -0.0381 - 0.2191i
   0.0597 - 0.5897i  -0.1385 - 0.0909i
  -0.0464 + 0.3086i  -0.3735 - 0.5515i
   0.5784 - 0.1244i  -0.0188 - 0.0557i
   0.0158 + 0.0471i   0.6501 + 0.0049i
vOut =
   0.9893 - 0.0000i  -0.1146 + 0.0903i
  -0.1146 - 0.0903i  -0.9893 - 0.0000i
qOut =
    0.7071                   0               0.6995 - 0.0000i    0.0810 -
0.0638i
         0              0.7071              -0.0810 - 0.0638i    0.6995 +
0.0000i
    0.7071                   0              -0.6995 + 0.0000i   -0.0810 +
0.0638i
         0              0.7071               0.0810 + 0.0638i   -0.6995 -
0.0000i
ncycle =
           2
info =
           0
```