NAG Toolbox for MATLAB

f08ye

1 Purpose

f08ye computes the generalized singular value decomposition (GSVD) of two real 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 f08ve.

2 Syntax

```
[a, b, alpha, beta, u, v, q, ncycle, info] = f08ye(jobu, jobv, jobq, k,
l, a, b, tola, tolb, u, v, q, 'm', m, 'p', p, 'n', n)
```

3 Description

f08ye computes the GSVD of the matrices A and B which are assumed to have the form as returned by f08ve

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

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

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.

f08ye computes orthogonal matrices Q, U and V, diagonal matrices D_1 and D_2, and an upper triangular matrix R such that

$$U^{\mathrm{T}}AQ = D_1 \begin{pmatrix} 0 & R \end{pmatrix}, \qquad V^{\mathrm{T}}BQ = D_2 \begin{pmatrix} 0 & R \end{pmatrix}.$$

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

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

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

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

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

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

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

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

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

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

In both cases the diagonal matrix C has nonnegative diagonal elements, the diagonal matrix S has 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 an orthogonal matrix U_1 on entry, and the product U_1U is returned.

If **jobu** = 'I', **u** is initialized to the unit matrix, and the orthogonal 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 an orthogonal matrix V_1 on entry, and the product V_1V is returned.

If **jobv** = 'I', **v** is initialized to the unit matrix, and the orthogonal 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 an orthogonal matrix Q_1 on entry, and the product Q_1Q is returned.

If **jobq** = 'I', **q** is initialized to the unit matrix, and the orthogonal 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 f08ye.

6: **a(lda,*) – double 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,*) – double 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,*) – double array**

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

if **jobu** = 'U', $\mathbf{ldu} \geq \max(1,\mathbf{m})$;
$\mathbf{ldu} \geq 1$ 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 orthogonal matrix returned by f08ve).

11: **v(ldv,*) – double array**

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

if **jobv** = 'V', $\mathbf{ldv} \geq \max(1,\mathbf{p})$;
$\mathbf{ldv} \geq 1$ 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 orthogonal matrix returned by f08ve).

12: **q(ldq,$*$) – double 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** = 'Q', **q** must contain an n by n matrix Q_1 (usually the orthogonal matrix returned by f08ve).

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

2: **p – int32 scalar**

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

p, the number of rows of the matrix B.

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

3: **n – int32 scalar**

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

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

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

5.3 Input Parameters Omitted from the MATLAB Interface

lda, ldb, ldu, ldv, ldq, work

5.4 Output Parameters

1: **a(lda,$*$) – double 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 $\max(1, \mathbf{n})$

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,$*$) – double 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 $\max(1, \mathbf{n})$

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 $\max(1, \mathbf{n})$.

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**,$*$) – **double 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 orthogonal matrix U.

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

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

6: **v**(**ldv**,$*$) – **double 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 orthogonal matrix V.

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

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

7: **q**(**ldq**,$*$) – **double 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 orthogonal 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 complex analogue of this function is f08ys.

9 Example

```
jobu = 'U';
jobv = 'V';
jobq = 'Q';
k = int32(1);
l = int32(2);
a = [-2.056883378018607, 10.77058932489743, -7.281358514603013;
     0, 7.194695015331533, -7.526215425991952;
     0, 0, 0.5812912724091043;
     0, 0, 0];
b = [0, 8.062257748298549, -3.130495168499705;
     0, 0, -4.919349550499538];
tola = 8.001412032943023e-15;
tolb = 3.000529512353634e-15;
u   =   [-0.1348399724926486,   0.5102518932875902,   -0.2435139114146218,
0.8137334712067352;
          0.6741998624632418,  -0.5466984570938458,  -0.3534879359244544,
0.3487429162314597;
         0.2696799449852967,  0.4829169704328976,  -0.6912652969189361,  -
0.4649905549752752;
          0.674199862463242,  0.4555820475782051,  0.5812912724091038,  -
1.816851790542584e-15];
v = [-0.4472135954999579, 0.8944271909999157;
     0.8944271909999157, 0.447213595499958];
q = [-0.8320502943378437, 0.554700196225229, 0;
     0.554700196225229, 0.8320502943378438, 0;
     1.847522788840025e-16, -1.231681859226683e-16, -0.9999999999999999];
[aOut, bOut, alpha, beta, uOut, vOut, qOut, ncycle, info] = ...
    f08ye(jobu, jobv, jobq, k, l, a, b, tola, tolb, u, v, q)
```

```
aOut =
```

```
   -2.0569   -9.0121   -9.3705
         0  -10.8822   -7.2688
         0         0   -6.0405
         0         0         0
bOut =
         0   -6.5869   -4.3998
         0         0   -6.0405
alpha =
    1.0000
    0.7960
    0.0799
beta =
         0
    0.6053
    0.9968
uOut =
   -0.1348    0.5252   -0.2092    0.8137
    0.6742   -0.5221   -0.3889    0.3487
    0.2697    0.5276   -0.6578   -0.4650
    0.6742    0.4161    0.6101   -0.0000
vOut =
    0.3554   -0.9347
    0.9347    0.3554
qOut =
   -0.8321   -0.0946   -0.5466
    0.5547   -0.1419   -0.8199
    0.0000   -0.9853    0.1706
ncycle =
           2
info =
           0
```