NAG Library Routine Document

F08VQF (ZGGSVD3)

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

F08VQF (ZGGSVD3) computes the generalized singular value decomposition (GSVD) of an m by n complex matrix A and a p by n complex matrix B.

2 Specification

```
SUBROUTINE F08VQF (JOBU, JOBV, JOBQ, M, N, P, K, L, A, LDA, B, LDB,        &
                   ALPHA, BETA, U, LDU, V, LDV, Q, LDQ, WORK, LWORK,       &
                   RWORK, IWORK, INFO)
INTEGER                 M, N, P, K, L, LDA, LDB, LDU, LDV, LDQ, LWORK,     &
                        IWORK(N), INFO
REAL (KIND=nag_wp)      ALPHA(N), BETA(N), RWORK(2*N)
COMPLEX (KIND=nag_wp) A(LDA,*), B(LDB,*), U(LDU,*), V(LDV,*),              &
                        Q(LDQ,*), WORK(max(1,LWORK))
CHARACTER(1)            JOBU, JOBV, JOBQ
```

The routine may be called by its LAPACK name ***zggsvd3***.

3 Description

Given an m by n complex matrix A and a p by n complex matrix B, the generalized singular value decomposition is given by

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

where U, V and Q are unitary matrices. Let l be the effective numerical rank of B and $(k+l)$ be the effective numerical rank of the matrix $\begin{pmatrix} A \\ B \end{pmatrix}$, then the first k generalized singular values are infinite and the remaining l are finite. R is a $(k+l)$ by $(k+l)$ nonsingular upper triangular matrix, D_1 and D_2 are m by $(k+l)$ and p by $(k+l)$ 'diagonal' matrices structured as follows:

if $m-k-l \geq 0$,

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

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

$$\begin{pmatrix} 0 & R \end{pmatrix} = \begin{array}{c} \\ k \\ l \end{array} \begin{array}{c} \begin{array}{ccc} n-k-l & k & l \end{array} \\ \begin{pmatrix} 0 & R_{11} & R_{12} \\ 0 & 0 & R_{22} \end{pmatrix} \end{array}$$

where

$$C = \mathrm{diag}(\alpha_{k+1}, \dots, \alpha_{k+l}),$$

$$S = \mathrm{diag}(\beta_{k+1}, \dots, \beta_{k+l}),$$

and

$$C^2 + S^2 = I.$$

R is stored as a submatrix of A with elements R_{ij} stored as $A_{i,n-k-l+j}$ on exit.

If $m - k - l < 0$,

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

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

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

where

$$C = \operatorname{diag}(\alpha_{k+1}, \dots, \alpha_m),$$
$$S = \operatorname{diag}(\beta_{k+1}, \dots, \beta_m),$$

and

$$C^2 + S^2 = I.$$

$\begin{pmatrix} R_{11} & R_{12} & R_{13} \\ 0 & R_{22} & R_{23} \end{pmatrix}$ is stored as a submatrix of A with R_{ij} stored as $A_{i,n-k-l+j}$, and R_{33} is stored as a submatrix of B with $(R_{33})_{ij}$ stored as $B_{m-k+i,n+m-k-l+j}$.

The routine computes C, S, R and, optionally, the unitary transformation matrices U, V and Q.

In particular, if B is an n by n nonsingular matrix, then the GSVD of A and B implicitly gives the SVD of $A \times B^{-1}$:

$$AB^{-1} = U\left(D_1 D_2^{-1}\right)V^{\mathrm{H}}.$$

If $\begin{pmatrix} A \\ B \end{pmatrix}$ has orthonormal columns, then the GSVD of A and B is also equal to the CS decomposition of A and B. Furthermore, the GSVD can be used to derive the solution of the eigenvalue problem:

$$A^{\mathrm{H}}Ax = \lambda B^{\mathrm{H}}Bx.$$

In some literature, the GSVD of A and B is presented in the form

$$U^{\mathrm{H}}AX = \begin{pmatrix} 0 & D_1 \end{pmatrix}, \quad V^{\mathrm{H}}BX = \begin{pmatrix} 0 & D_2 \end{pmatrix},$$

where U and V are orthogonal and X is nonsingular, and D_1 and D_2 are 'diagonal'. The former GSVD form can be converted to the latter form by setting

$$X = Q\begin{pmatrix} I & 0 \\ 0 & R^{-1} \end{pmatrix}.$$

A two stage process is used to compute the GSVD of the matrix pair (A, B). The pair is first reduced to upper triangular form by unitary transformations using F08VUF (ZGGSVP3). The GSVD of the resulting upper triangular matrix pair is then performed by F08YSF (ZTGSJA) which uses a variant of the Kogbetliantz algorithm (a cyclic Jacobi method).

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 http://www.netlib.org/lapack/lug

Golub G H and Van Loan C F (2012) *Matrix Computations* (4th Edition) Johns Hopkins University Press, Baltimore

5 Arguments

1: JOBU – CHARACTER(1) *Input*

On entry: if JOBU = 'U', the unitary matrix U is computed.

If JOBU = 'N', U is not computed.

Constraint: JOBU = 'U' or 'N'.

2: JOBV – CHARACTER(1) *Input*

On entry: if JOBV = 'V', the unitary matrix V is computed.

If JOBV = 'N', V is not computed.

Constraint: JOBV = 'V' or 'N'.

3: JOBQ – CHARACTER(1) *Input*

On entry: if JOBQ = 'Q', the unitary matrix Q is computed.

If JOBQ = 'N', Q is not computed.

Constraint: JOBQ = 'Q' or 'N'.

4: M – INTEGER *Input*

On entry: m, the number of rows of the matrix A.

Constraint: $\mathrm{M} \geq 0$.

5: N – INTEGER *Input*

On entry: n, the number of columns of the matrices A and B.

Constraint: $\mathrm{N} \geq 0$.

6: P – INTEGER *Input*

On entry: p, the number of rows of the matrix B.

Constraint: $\mathrm{P} \geq 0$.

7: K – INTEGER *Output*
8: L – INTEGER *Output*

On exit: K and L specify the dimension of the subblocks k and l as described in Section 3; $(k+l)$ is the effective numerical rank of $\begin{pmatrix} A \\ B \end{pmatrix}$.

9: A(LDA, $*$) – COMPLEX (KIND=nag_wp) array *Input/Output*

Note: the second dimension of the array A must be at least $\max(1, \mathrm{N})$.

On entry: the m by n matrix A.

On exit: contains the triangular matrix R, or part of R. See Section 3 for details.

10: LDA – INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F08VQF (ZGGSVD3) is called.

Constraint: $\mathrm{LDA} \geq \max(1, \mathrm{M})$.

11: $\mathrm{B}(\mathrm{LDB}, *)$ – COMPLEX (KIND=nag_wp) array *Input/Output*

Note: the second dimension of the array B must be at least $\max(1, \mathrm{N})$.

On entry: the p by n matrix B.

On exit: contains the triangular matrix R if $m - k - l < 0$. See Section 3 for details.

12: LDB – INTEGER *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F08VQF (ZGGSVD3) is called.

Constraint: $\mathrm{LDB} \geq \max(1, \mathrm{P})$.

13: ALPHA(N) – REAL (KIND=nag_wp) array *Output*

On exit: see the description of BETA.

14: BETA(N) – REAL (KIND=nag_wp) array *Output*

On exit: ALPHA and BETA contain the generalized singular value pairs of A and B, α_i and β_i;

$\mathrm{ALPHA}(1:\mathrm{K}) = 1$,

$\mathrm{BETA}(1:\mathrm{K}) = 0$,

and if $m - k - l \geq 0$,

$\mathrm{ALPHA}(\mathrm{K}+1:\mathrm{K}+\mathrm{L}) = C$,

$\mathrm{BETA}(\mathrm{K}+1:\mathrm{K}+\mathrm{L}) = S$,

or if $m - k - l < 0$,

$\mathrm{ALPHA}(\mathrm{K}+1:\mathrm{M}) = C$,

$\mathrm{ALPHA}(\mathrm{M}+1:\mathrm{K}+\mathrm{L}) = 0$,

$\mathrm{BETA}(\mathrm{K}+1:\mathrm{M}) = S$,

$\mathrm{BETA}(\mathrm{M}+1:\mathrm{K}+\mathrm{L}) = 1$, and

$\mathrm{ALPHA}(\mathrm{K}+\mathrm{L}+1:\mathrm{N}) = 0$,

$\mathrm{BETA}(\mathrm{K}+\mathrm{L}+1:\mathrm{N}) = 0$.

The notation $\mathrm{ALPHA}(\mathrm{K}:\mathrm{N})$ above refers to consecutive elements $\mathrm{ALPHA}(i)$, for $i = \mathrm{K}, \ldots, \mathrm{N}$.

15: $\mathrm{U}(\mathrm{LDU}, *)$ – COMPLEX (KIND=nag_wp) array *Output*

Note: the second dimension of the array U must be at least $\max(1, \mathrm{M})$ if JOBU = 'U', and at least 1 otherwise.

On exit: if JOBU = 'U', U contains the m by m unitary matrix U.

If JOBU = 'N', U is not referenced.

16: LDU – INTEGER *Input*

On entry: the first dimension of the array U as declared in the (sub)program from which F08VQF (ZGGSVD3) is called.

Constraints:

if JOBU = 'U', $\text{LDU} \geq \max(1, \text{M})$;
otherwise $\text{LDU} \geq 1$.

17: $\text{V}(\text{LDV}, *)$ – COMPLEX (KIND=nag_wp) array *Output*

Note: the second dimension of the array V must be at least $\max(1, \text{P})$ if JOBV = 'V', and at least 1 otherwise.

On exit: if JOBV = 'V', V contains the p by p unitary matrix V.

If JOBV = 'N', V is not referenced.

18: LDV – INTEGER *Input*

On entry: the first dimension of the array V as declared in the (sub)program from which F08VQF (ZGGSVD3) is called.

Constraints:

if JOBV = 'V', $\text{LDV} \geq \max(1, \text{P})$;
otherwise $\text{LDV} \geq 1$.

19: $\text{Q}(\text{LDQ}, *)$ – COMPLEX (KIND=nag_wp) array *Output*

Note: the second dimension of the array Q must be at least $\max(1, \text{N})$ if JOBQ = 'Q', and at least 1 otherwise.

On exit: if JOBQ = 'Q', Q contains the n by n unitary matrix Q.

If JOBQ = 'N', Q is not referenced.

20: LDQ – INTEGER *Input*

On entry: the first dimension of the array Q as declared in the (sub)program from which F08VQF (ZGGSVD3) is called.

Constraints:

if JOBQ = 'Q', $\text{LDQ} \geq \max(1, \text{N})$;
otherwise $\text{LDQ} \geq 1$.

21: $\text{WORK}(\max(1, \text{LWORK}))$ – COMPLEX (KIND=nag_wp) array *Workspace*

On exit: if $\text{INFO} = 0$, the real part of $\text{WORK}(1)$ contains the minimum value of LWORK required for optimal performance.

22: LWORK – INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)routine from which F08VQF (ZGGSVD3) is called.

If $\text{LWORK} = -1$, a workspace query is assumed; the routine only calculates the optimal size of the WORK array, returns this value as the first entry of the WORK array, and no error message related to LWORK is issued.

Suggested value: for optimal performance, LWORK must generally be larger than the minimum; increase workspace by, say, $nb \times (\text{N} + 1)$, where nb is the optimal ***block size***.

Constraints:

if JOBV = 'V', $\text{LWORK} \geq \max(\text{N} + 1, \text{M}, \text{P})$;
if JOBV = 'N', $\text{LWORK} \geq \max(\text{N} + 1, \text{M})$.

23: $\text{RWORK}(2 \times \text{N})$ – REAL (KIND=nag_wp) array *Workspace*

24: IWORK(N) – INTEGER array *Output*

On exit: stores the sorting information. More precisely, if I is the ordered set of indices of ALPHA containing C (denote as ALPHA(I), see BETA), then the corresponding elements IWORK(I) contain the swap pivots, J, that sorts I such that ALPHA(I) is in descending numerical order.

The following pseudocode sorts the set I:

```
for $i \in I$
    $j = J_i$
    swap $I_i$ and $I_j$
end
```

25: INFO – INTEGER *Output*

On exit: INFO $= 0$ unless the routine detects an error (see Section 6).

6 Error Indicators and Warnings

INFO < 0

If INFO $= -i$, argument i had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO $= 1$

The Jacobi-type procedure failed to converge.

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

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

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

8 Parallelism and Performance

F08VQF (ZGGSVD3) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

F08VQF (ZGGSVD3) 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 X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

This routine replaces the deprecated routine F08VNF (ZGGSVD) which used an unblocked algorithm and therefore did not make best use of level 3 BLAS routines.

The diagonal elements of the matrix R are real.

The real analogue of this routine is F08VCF (DGGSVD3).

10 Example

This example finds the generalized singular value decomposition

$$A = U\Sigma_1\begin{pmatrix}0 & R\end{pmatrix}Q^{\mathrm{H}}, \quad B = V\Sigma_2\begin{pmatrix}0 & R\end{pmatrix}Q^{\mathrm{H}},$$

where

$$A = \begin{pmatrix} 0.96-0.81i & -0.03+0.96i & -0.91+2.06i & -0.05+0.41i \\ -0.98+1.98i & -1.20+0.19i & -0.66+0.42i & -0.81+0.56i \\ 0.62-0.46i & 1.01+0.02i & 0.63-0.17i & -1.11+0.60i \\ 0.37+0.38i & 0.19-0.54i & -0.98-0.36i & 0.22-0.20i \\ 0.83+0.51i & 0.20+0.01i & -0.17-0.46i & 1.47+1.59i \\ 1.08-0.28i & 0.20-0.12i & -0.07+1.23i & 0.26+0.26i \end{pmatrix}$$

and

$$B = \begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix},$$

together with estimates for the condition number of R and the error bound for the computed generalized singular values.

The example program assumes that $m \geq n$, and would need slight modification if this is not the case.

10.1 Program Text

```
    Program f08vqfe

!     F08VQF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: nag_wp, x02ajf, x04dbf, zggsvd3, ztrcon
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter              :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)              :: eps, rcond, serrbd
      Integer                         :: i, ifail, info, irank, j, k, l, lda, &
                                         ldb, ldq, ldu, ldv, lwork, m, n, p
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: a(:,:), b(:,:), q(:,:), u(:,:),    &
                                         v(:,:), work(:)
      Complex (Kind=nag_wp)           :: wdum(1)
      Real (Kind=nag_wp), Allocatable  :: alpha(:), beta(:), rwork(:)
      Integer, Allocatable            :: iwork(:)
      Character (1)                   :: clabs(1), rlabs(1)
!     .. Intrinsic Procedures ..
      Intrinsic                       :: nint, real
!     .. Executable Statements ..
      Write (nout,*) 'F08VQF Example Program Results'
      Write (nout,*)
      Flush (nout)
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) m, n, p
      lda = m
      ldb = p
      ldq = n
      ldu = m
      ldv = p
      Allocate (a(lda,n),b(ldb,n),q(ldq,n),u(ldu,m),v(ldv,p),alpha(n),beta(n), &
        rwork(2*n),iwork(n))

!     Perform workspace query to get optimal size of work
!     The NAG name equivalent of zggsvd3 is f08vqf
```

```
      lwork = -1
      Call zggsvd3('U','V','Q',m,n,p,k,l,a,lda,b,ldb,alpha,beta,u,ldu,v,ldv,q, &
        ldq,wdum,lwork,rwork,iwork,info)
      lwork = nint(real(wdum(1)))
      Allocate (work(lwork))

!     Read the m by n matrix A and p by n matrix B from data file

      Read (nin,*)(a(i,1:n),i=1,m)
      Read (nin,*)(b(i,1:n),i=1,p)

!     Compute the generalized singular value decomposition of (A, B)
!     (A = U*D1*(0 R)*(Q**H), B = V*D2*(0 R)*(Q**H), m.ge.n)
!     The NAG name equivalent of zggsvd3 is f08vqf
      Call zggsvd3('U','V','Q',m,n,p,k,l,a,lda,b,ldb,alpha,beta,u,ldu,v,ldv,q, &
        ldq,work,lwork,rwork,iwork,info)

      If (info==0) Then

!       Print solution

        irank = k + l
        Write (nout,*) 'Number of infinite generalized singular values (K)'
        Write (nout,99999) k
        Write (nout,*) 'Number of finite generalized singular values (L)'
        Write (nout,99999) l
        Write (nout,*) 'Numerical rank of (A**T B**T)**T (K+L)'
        Write (nout,99999) irank
        Write (nout,*)
        Write (nout,*) 'Finite generalized singular values'
        Write (nout,99998)(alpha(j)/beta(j),j=k+1,irank)

        Write (nout,*)
        Flush (nout)

!       ifail: behaviour on error exit
!              =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
        ifail = 0
        Call x04dbf('General',' ',m,m,u,ldu,'Bracketed','1P,E12.4',          &
          'Unitary matrix U','Integer',rlabs,'Integer',clabs,80,0,ifail)

        Write (nout,*)
        Flush (nout)

        Call x04dbf('General',' ',p,p,v,ldv,'Bracketed','1P,E12.4',          &
          'Unitary matrix V','Integer',rlabs,'Integer',clabs,80,0,ifail)

        Write (nout,*)
        Flush (nout)

        Call x04dbf('General',' ',n,n,q,ldq,'Bracketed','1P,E12.4',          &
          'Unitary matrix Q','Integer',rlabs,'Integer',clabs,80,0,ifail)

        Write (nout,*)
        Flush (nout)

        Call x04dbf('Upper triangular','Non-unit',irank,irank,a(1,n-irank+1),  &
          lda,'Bracketed','1P,E12.4','Nonsingular upper triangular matrix R',  &
          'Integer',rlabs,'Integer',clabs,80,0,ifail)

!       Call ZTRCON (F07TUF) to estimate the reciprocal condition
!       number of R

        Call ztrcon('Infinity-norm','Upper','Non-unit',irank,a(1,n-irank+1),   &
          lda,rcond,work,rwork,info)

        Write (nout,*)
        Write (nout,*) 'Estimate of reciprocal condition number for R'
        Write (nout,99997) rcond
        Write (nout,*)
```

```
!       So long as irank = n, get the machine precision, eps, and
!       compute the approximate error bound for the computed
!       generalized singular values

        If (irank==n) Then
          eps = x02ajf()
          serrbd = eps/rcond
          Write (nout,*) 'Error estimate for the generalized singular values'
          Write (nout,99997) serrbd
        Else
          Write (nout,*) '(A**T B**T)**T is not of full rank'
        End If
      Else
        Write (nout,99996) 'Failure in ZGGSVD3. INFO =', info
      End If

99999 Format (1X,I5)
99998 Format (4X,8(1P,E13.4))
99997 Format (3X,1P,E11.1)
99996 Format (1X,A,I4)
    End Program f08vqfe
```

10.2 Program Data

```
F08VQF Example Program Data

  6            4            2                         :Values of M, N and P

( 0.96,-0.81) (-0.03, 0.96) (-0.91, 2.06) (-0.05, 0.41)
(-0.98, 1.98) (-1.20, 0.19) (-0.66, 0.42) (-0.81, 0.56)
( 0.62,-0.46) ( 1.01, 0.02) ( 0.63,-0.17) (-1.11, 0.60)
( 0.37, 0.38) ( 0.19,-0.54) (-0.98,-0.36) ( 0.22,-0.20)
( 0.83, 0.51) ( 0.20, 0.01) (-0.17,-0.46) ( 1.47, 1.59)
( 1.08,-0.28) ( 0.20,-0.12) (-0.07, 1.23) ( 0.26, 0.26) :End of matrix A

( 1.00, 0.00) ( 0.00, 0.00) (-1.00, 0.00) ( 0.00, 0.00)
( 0.00, 0.00) ( 1.00, 0.00) ( 0.00, 0.00) (-1.00, 0.00) :End of matrix B
```

10.3 Program Results

```
 F08VQF Example Program Results

 Number of infinite generalized singular values (K)
     2
 Number of finite generalized singular values (L)
     2
 Numerical rank of (A**T B**T)**T (K+L)
     4

 Finite generalized singular values
       2.0720E+00   1.1058E+00

 Unitary matrix U
                                1                             2
 1  ( -1.3038E-02, -3.2595E-01) ( -1.4039E-01, -2.6167E-01)
 2  (  4.2764E-01, -6.2582E-01) (  8.6298E-02, -3.8174E-02)
 3  ( -3.2595E-01,  1.6428E-01) (  3.8163E-01, -1.8219E-01)
 4  (  1.5906E-01, -5.2151E-03) ( -2.8207E-01,  1.9732E-01)
 5  ( -1.7210E-01, -1.3038E-02) ( -5.0942E-01, -5.0319E-01)
 6  ( -2.6336E-01, -2.4772E-01) ( -1.0861E-01,  2.8474E-01)

                                3                             4
 1  (  2.5177E-01, -7.9789E-01) ( -5.0956E-02, -2.1750E-01)
 2  ( -3.2188E-01,  1.6112E-01) (  1.1979E-01,  1.6319E-01)
 3  (  1.3231E-01, -1.4565E-02) ( -5.0671E-01,  1.8615E-01)
 4  (  2.1598E-01,  1.8813E-01) ( -4.0163E-01,  2.6787E-01)
 5  (  3.6488E-02,  2.0316E-01) (  1.9271E-01,  1.5574E-01)
 6  (  1.0906E-01, -1.2712E-01) ( -8.8159E-02,  5.6169E-01)

                                5                             6
```

```
1  ( -4.5947E-02,  1.4052E-04) ( -5.2773E-02, -2.2492E-01)
2  ( -8.0311E-02, -4.3605E-01) ( -3.8117E-02, -2.1907E-01)
3  (  5.9714E-02, -5.8974E-01) ( -1.3850E-01, -9.0941E-02)
4  ( -4.6443E-02,  3.0864E-01) ( -3.7354E-01, -5.5148E-01)
5  (  5.7843E-01, -1.2439E-01) ( -1.8815E-02, -5.5686E-02)
6  (  1.5763E-02,  4.7130E-02) (  6.5007E-01,  4.9173E-03)

Unitary matrix V
                            1                           2
1  (  9.8930E-01,  1.0471E-19) ( -1.1461E-01,  9.0250E-02)
2  ( -1.1461E-01, -9.0250E-02) ( -9.8930E-01,  1.0471E-19)

Unitary matrix Q
                            1                           2
1  (  7.0711E-01,  0.0000E+00) (  0.0000E+00,  0.0000E+00)
2  (  0.0000E+00,  0.0000E+00) (  7.0711E-01,  0.0000E+00)
3  (  7.0711E-01,  0.0000E+00) (  0.0000E+00,  0.0000E+00)
4  (  0.0000E+00,  0.0000E+00) (  7.0711E-01,  0.0000E+00)

                            3                           4
1  (  6.9954E-01, -1.1784E-18) (  8.1044E-02, -6.3817E-02)
2  ( -8.1044E-02, -6.3817E-02) (  6.9954E-01,  1.1784E-18)
3  ( -6.9954E-01,  1.1784E-18) ( -8.1044E-02,  6.3817E-02)
4  (  8.1044E-02,  6.3817E-02) ( -6.9954E-01, -1.1784E-18)

Nonsingular upper triangular matrix R
                            1                           2
1  ( -2.7118E+00,  0.0000E+00) ( -1.4390E+00, -1.0315E+00)
2                              ( -1.8583E+00,  0.0000E+00)
3
4

                            3                           4
1  ( -7.6930E-02,  1.3613E+00) ( -2.8137E-01, -3.2425E-02)
2  ( -1.0760E+00,  3.1016E-02) (  1.3292E+00,  3.6772E-01)
3  (  3.2537E+00,  0.0000E+00) ( -6.3858E-17,  3.4216E-33)
4                              ( -2.1084E+00,  0.0000E+00)

Estimate of reciprocal condition number for R
      1.3E-01

Error estimate for the generalized singular values
      8.3E-16
```