NAG Library Routine Document

f08khf (dgejsv)

1 Purpose

f08khf (dgejsv) computes the singular value decomposition (SVD) of a real m by n matrix A where $m \geq n$, and optionally computes the left and/or right singular vectors. **f08khf (dgejsv)** implements the preconditioned Jacobi SVD of Drmac and Veselic. This is the expert driver routine that calls **f08kjf (dgesvj)** after certain preconditioning. In most cases **f08kbf (dgesvd)** or **f08kdf (dgesdd)** is sufficient to obtain the SVD of a real matrix. These are much simpler to use and also handle the case $m < n$.

2 Specification

Fortran Interface

```
Subroutine f08khf (joba, jobu, jobv, jobr, jobt, jobp, m, n, a, lda,        &
                   sva, u, ldu, v, ldv, work, lwork, iwork, info)
Integer, Intent (In)                   :: m, n, lda, ldu, ldv, lwork
Integer, Intent (Out)                  :: iwork(m+3*n), info
Real (Kind=nag_wp), Intent (Inout) :: a(lda,*), u(ldu,*), v(ldv,*)
Real (Kind=nag_wp), Intent (Out)   :: sva(n), work(lwork)
Character (1), Intent (In)             :: joba, jobu, jobv, jobr, jobt,     &
                                          jobp
```

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

3 Description

The SVD is written as

$$A = U\Sigma V^{\mathrm{T}},$$

where Σ is an m by n matrix which is zero except for its n diagonal elements, U is an m by m orthogonal matrix, and V is an n by n orthogonal matrix. The diagonal elements of Σ are the singular values of A in descending order of magnitude. The columns of U and V are the left and the right singular vectors of A. The diagonal of Σ is computed and stored in the array **sva**.

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

Drmac Z and Veselic K (2008a) New fast and accurate Jacobi SVD algorithm I *SIAM J. Matrix Anal. Appl.* **29 4**

Drmac Z and Veselic K (2008b) New fast and accurate Jacobi SVD algorithm II *SIAM J. Matrix Anal. Appl.* **29 4**

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

5 Arguments

1: **joba** – Character(1) *Input*

On entry: specifies the form of pivoting for the QR factorization stage; whether an estimate of the condition number of the scaled matrix is required; and the form of rank reduction that is performed.

joba = 'C'
: The initial QR factorization of the input matrix is performed with column pivoting; no estimate of condition number is computed; and, the rank is reduced by only the underflowed part of the triangular factor R. This option works well (high relative accuracy) if $A = BD$, with well-conditioned B and arbitrary diagonal matrix D. The accuracy cannot be spoiled by column scaling. The accuracy of the computed output depends on the condition of B, and the procedure aims at the best theoretical accuracy.

joba = 'E'
: Computation as with **joba** = 'C' with an additional estimate of the condition number of B. It provides a realistic error bound.

joba = 'F'
: The initial QR factorization of the input matrix is performed with full row and column pivoting; no estimate of condition number is computed; and, the rank is reduced by only the underflowed part of the triangular factor R. If $A = D_1 \times C \times D_2$ with ill-conditioned diagonal scalings D_1, D_2, and well-conditioned matrix C, this option gives higher accuracy than the **joba** = 'C' option. If the structure of the input matrix is not known, and relative accuracy is desirable, then this option is advisable.

joba = 'G'
: Computation as with **joba** = 'F' with an additional estimate of the condition number of B, where $A = DB$ (i.e., $B = C \times D_2$). If A has heavily weighted rows, then using this condition number gives too pessimistic an error bound.

joba = 'A'
: Computation as with **joba** = 'C' except in the treatment of rank reduction. In this case, small singular values are to be considered as noise and, if found, the matrix is treated as numerically rank deficient. The computed SVD $A = U\Sigma V^{\mathrm{T}}$ restores A up to $f(m,n) \times \epsilon \times \|A\|$, where ϵ is ***machine precision***. This gives the procedure licence to discard (set to zero) all singular values below **n** $\times \epsilon \times \|A\|$.

joba = 'R'
: Similar to **joba** = 'A'. The rank revealing property of the initial QR factorization is used to reveal (using the upper triangular factor) a gap $\sigma_{r+1} < \epsilon\sigma_r$ in which case the numerical rank is declared to be r. The SVD is computed with absolute error bounds, but more accurately than with **joba** = 'A'.

Constraint: **joba** = 'C', 'E', 'F', 'G', 'A' or 'R'.

2: **jobu** – Character(1) *Input*

On entry: specifies options for computing the left singular vectors U.

jobu = 'U'
: The first n left singular vectors (columns of U) are computed and returned in the array **u**.

jobu = 'F'
: All m left singular vectors are computed and returned in the array **u**.

jobu = 'W'
: No left singular vectors are computed, but the array **u** (with **ldu** $\geq$ **m** and second dimension at least **n**) is available as workspace for computing right singular values. See the description of **u**.

jobu = 'N'
: No left singular vectors are computed. **u** is not referenced when **jobv** = 'N' or 'W'.

Constraint: **jobu** = 'U', 'F', 'W' or 'N'.

3: **jobv** – Character(1) *Input*

On entry: specifies options for computing the right singular vectors V.

jobv = 'V'
: the n right singular vectors (columns of V) are computed and returned in the array **v**; Jacobi rotations are not explicitly accumulated.

jobv = 'J'
: the n right singular vectors (columns of V) are computed and returned in the array **v**, but they are computed as the product of Jacobi rotations. This option is allowed only if **jobu** = 'U' or 'F', i.e., in computing the full SVD.

This is equivalent to multiplying the input matrix, on the right, by the matrix V.

jobv = 'W'
: No right singular values are computed, but the array **v** (with **ldv** $\geq$ **n** and second dimension at least **n**) is available as workspace for computing left singular values. See the description of **v**.

jobv = 'N'
: No right singular vectors are computed. **v** is not referenced when **jobu** = 'N' or 'W' or **jobt** = 'N' or **m** $\neq$ **n**.

Constraints:

jobv = 'V', 'J', 'W' or 'N';
if **jobu** = 'W' or 'N', **jobv** $\neq$ 'J'.

4: **jobr** – Character(1) *Input*

On entry: specifies the conditions under which columns of A are to be set to zero. This effectively specifies a lower limit on the range of singular values; any singular values below this limit are (through column zeroing) set to zero. If $A \neq 0$ is scaled so that the largest column (in the Euclidean norm) of cA is equal to the square root of the overflow threshold, then **jobr** allows the routine to kill columns of A whose norm in cA is less than $\sqrt{sfmin}$ (for **jobr** = 'R'), or less than $sfmin/\epsilon$ (otherwise). $sfmin$ is the safe range argument, as returned by routine **x02amf**.

jobr = 'N'
: Only set to zero those columns of A for which the norm of corresponding column of $cA < sfmin/\epsilon$, that is, those columns that are effectively zero (to ***machine precision***) anyway. If the condition number of A is greater than the overflow threshold λ, where λ is the value returned by **x02alf**, you are recommended to use routine **f08kjf (dgesvj)**.

jobr = 'R'
: Set to zero those columns of A for which the norm of the corresponding column of $cA < \sqrt{sfmin}$. This approximately represents a restricted range for $\sigma(cA)$ of $\left[\sqrt{sfmin}, \sqrt{\lambda}\right]$.

For computing the singular values in the full range from the safe minimum up to the overflow threshold use **f08kjf (dgesvj)**.

Suggested value: **jobr** = 'R'.

Constraint: **jobr** = 'N' or 'R'.

5: **jobt** – Character(1) *Input*

On entry: specifies, in the case $n = m$, whether the routine is permitted to use the transpose of A for improved efficiency. If the matrix is square then the procedure may use transposed A if A^{T} seems to be better with respect to convergence. If the matrix is not square, **jobt** is ignored. The decision is based on two values of entropy over the adjoint orbit of $A^{\mathrm{T}}A$. See the descriptions of **work**(6) and **work**(7).

jobt = 'T'
: If $n = m$, perform an entropy test and then transpose if the test indicates possibly faster convergence of the Jacobi process if A^{T} is taken as input. If A is replaced with A^{T}, then the row pivoting is included automatically.

jobt = 'N'
: No entropy test and no transposition is performed.

The option **jobt** = 'T' can be used to compute only the singular values, or the full SVD (U, Σ and V). In the case where only one set of singular vectors (U or V) is required, the caller must still provide both **u** and **v**, as one of the matrices is used as workspace if the matrix A is transposed. See the descriptions of **u** and **v**.

Constraint: **jobt** = 'T' or 'N'.

6: **jobp** – Character(1) *Input*

On entry: specifies whether the routine should be allowed to introduce structured perturbations to drown denormalized numbers. For details see Drmac and Veselic (2008a) and Drmac and Veselic (2008b). For the sake of simplicity, these perturbations are included only when the full SVD or only the singular values are requested.

jobp = 'P'
: Introduce perturbation if A is found to be very badly scaled (introducing denormalized numbers).

jobp = 'N'
: Do not perturb.

Constraint: **jobp** = 'P' or 'N'.

7: **m** – Integer *Input*

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

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

8: **n** – Integer *Input*

On entry: n, the number of columns of the matrix A.

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

9: **a**(**lda**, $*$) – Real (Kind=nag_wp) array *Input/Output*

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

On entry: the m by n matrix A.

On exit: the contents of **a** are overwritten.

10: **lda** – Integer *Input*

On entry: the first dimension of the array **a** as declared in the (sub)program from which **f08khf (dgejsv)** is called.

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

11: **sva**(**n**) – Real (Kind=nag_wp) array *Output*

On exit: the, possibly scaled, singular values of A.

The singular values of A are $\sigma_i = \alpha \mathbf{sva}(i)$, for $i = 1, 2, \dots, n$, where $\alpha = \mathbf{work}(1)/\mathbf{work}(2)$. Normally $\alpha = 1$ and no scaling is required to obtain the singular values. However, if the largest singular value of A overflows or if small singular values have been saved from underflow by scaling the input matrix A, then $\alpha \neq 1$.

If **jobr** = 'R' then some of the singular values may be returned as exact zeros because they are below the numerical rank threshold or are denormalized numbers.

12: **u**(**ldu**, $*$) – Real (Kind=nag_wp) array *Output*

Note: the second dimension of the array **u** must be at least $\max(1, \mathbf{m})$ if **jobu** = 'F', $\max(1, \mathbf{n})$ if **jobu** = 'U' or 'W', and at least 1 otherwise.

On exit: if **jobu** = 'U', **u** contains the m by n matrix of the left singular vectors.

If **jobu** = 'F', **u** contains the m by m matrix of the left singular vectors, including an orthonormal basis of the orthogonal complement of Range(A).

u is not referenced when **jobu** = 'W' or 'N' and one of the following is satisfied:

jobv = 'W' or 'N', or

n $= 1$, or

A is the zero matrix.

13: **ldu** – Integer *Input*

On entry: the first dimension of the array **u** as declared in the (sub)program from which **f08khf (dgejsv)** is called.

Constraints:

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

14: **v**(**ldv**, $*$) – Real (Kind=nag_wp) array *Output*

Note: the second dimension of the array **v** must be at least $\max(1, \mathbf{n})$ if **jobv** = 'V', 'J' or 'W', and at least 1 otherwise.

On exit: if **jobv** = 'V' or 'J', **v** contains the n by n matrix of the right singular vectors.

v is not referenced when **jobv** = 'W' or 'N' and one of the following is satisfied:

jobu = 'U' or 'F' and **jobt** = 'T', or

n $= 1$, or

A is the zero matrix.

15: **ldv** – Integer *Input*

On entry: the first dimension of the array **v** as declared in the (sub)program from which **f08khf (dgejsv)** is called.

Constraints:

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

16: **work**(**lwork**) – Real (Kind=nag_wp) array *Workspace*

On exit: contains information about the completed job.

work(1)
$\alpha = \mathbf{work}(1)/\mathbf{work}(2)$ is the scaling factor such that $\sigma_i = \alpha \mathbf{sva}(i)$, for $i = 1, 2, \ldots, n$ are the computed singular values of A. (See the description of **sva**.)

work(2)
See the description of **work**(1).

work(3)

sconda, an estimate for the condition number of column equilibrated A (if **joba** = 'E' or 'G'). *sconda* is an estimate of $\sqrt{\left(\left\|(R^{\mathrm{T}}R)^{-1}\right\|_1\right)}$. It is computed using **f07fgf (dpocon)**. It satisfies $n^{-\frac{1}{4}} \times sconda \le \left\|R^{-1}\right\|_2 \le n^{\frac{1}{4}} \times sconda$ where R is the triangular factor from the QR factorization of A. However, if R is truncated and the numerical rank is determined to be strictly smaller than n, *sconda* is returned as -1, thus indicating that the smallest singular values might be lost.

If full SVD is needed, and you are familiar with the details of the method, the following two condition numbers are useful for the analysis of the algorithm.

work(4)

An estimate of the scaled condition number of the triangular factor in the first QR factorization.

work(5)

An estimate of the scaled condition number of the triangular factor in the second QR factorization.

The following two parameters are computed if **jobt** = 'T'.

work(6)

The entropy of $A^{\mathrm{T}}A$: this is the Shannon entropy of diag $A^{\mathrm{T}}A/$ trace $A^{\mathrm{T}}A$ taken as a point in the probability simplex.

work(7)

The entropy of AA^{T}.

17: **lwork** – Integer *Input*

On entry: the dimension of the array **work** as declared in the (sub)program from which **f08khf (dgejsv)** is called.

If **jobu** = 'N' and **jobv** = 'N'

if **joba** ≠ 'E' or 'G'

the minimal requirement is **lwork** $\ge \max(2\mathbf{m}+\mathbf{n}, 4\mathbf{n}+1, 7)$.

For optimal performance the requirement is **lwork** $\ge \max(2\mathbf{m}+\mathbf{n}, 3\mathbf{n}+(\mathbf{n}+1)\times nb, 7)$, where nb is the block size used by **f08aef (dgeqrf)** and **f08bff (dgeqp3)**. Assuming a value of $nb = 256$ is wise, but choosing a smaller value (e.g., $nb = 128$) should still lead to acceptable performance.

If **joba** = 'E' or 'G'

the minimal requirement is **lwork** $\ge \max(2\mathbf{m}+\mathbf{n}, \mathbf{n}(\mathbf{n}+4), 7)$.

For optimal performance the requirement is **lwork** $\ge \max(2\mathbf{m}+\mathbf{n}, 3\mathbf{n}+(\mathbf{n}+1)\times nb, \mathbf{n}(\mathbf{n}+4), 7)$.

If **jobu** ≠ 'U' or 'F' and **jobv** = 'V' or 'J'

the minimal requirement is **lwork** $\ge \max(2\mathbf{m}+\mathbf{n}, 4\mathbf{n}+1, 7)$.

For optimal performance the requirement is **lwork** $\ge \max(2\mathbf{m}+\mathbf{n}, \mathbf{n}(3+nb), 7)$, where nb is described above.

If **jobu** = 'U' or 'F' and **jobv** ≠ 'V' or 'J'

the minimal requirement is **lwork** $\ge \max(2\mathbf{m}+\mathbf{n}, 4\mathbf{n}+1, 7)$.

For optimal performance the requirement is **lwork** $\ge \max(2\mathbf{m}+\mathbf{n}, \mathbf{n}(3+nb), 7)$, where nb is described above.

If **jobu** = 'U' or 'F' and **jobv** = 'V'

lwork $\ge \max(2\mathbf{n}(\mathbf{n}+3), 2\mathbf{m}+\mathbf{n}, 7)$.

If **jobu** = 'U' or 'F' and **jobv** = 'J'

the minimal requirement is **lwork** $\ge \max(\mathbf{n}(\mathbf{n}+4), 2\mathbf{m}+\mathbf{n}, \mathbf{n}(\mathbf{n}+2)+6, 7)$.

For better performance $\mathbf{lwork} \geq \max(2\mathbf{m} + \mathbf{n}, \mathbf{n}(\mathbf{n} + 3 + nb), 7)$, where nb is described above.

18: $\mathbf{iwork}(\mathbf{m} + 3 \times \mathbf{n})$ – Integer array *Workspace*

On exit: contains information about the completed job.

iwork(1)
The numerical rank of A determined after the initial QR factorization with pivoting. See the descriptions of **joba** and **jobr**.

iwork(2)
The number of computed nonzero singular values.

iwork(3)
If nonzero, a warning message: If $\mathbf{iwork}(3) = 1$ then some of the column norms of A were denormalized (tiny) numbers. The requested high accuracy is not warranted by the data.

19: **info** – Integer *Output*

On exit: $\mathbf{info} = 0$ unless the routine detects an error (see Section 6).

6 Error Indicators and Warnings

$\mathbf{info} < 0$

If $\mathbf{info} = -i$, argument i had an illegal value. An explanatory message is output, and execution of the program is terminated.

$\mathbf{info} > 0$

f08khf (dgejsv) did not converge in the allowed number of iterations (30). The computed values might be inaccurate.

7 Accuracy

The computed singular value decomposition is nearly the exact singular value decomposition for a nearby matrix $(A + E)$, where

$$\|E\|_2 = O(\epsilon)\|A\|_2,$$

and ϵ is the ***machine precision***. In addition, the computed singular vectors are nearly orthogonal to working precision. See Section 4.9 of Anderson *et al.* (1999) for further details.

8 Parallelism and Performance

f08khf (dgejsv) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

f08khf (dgejsv) 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

f08khf (dgejsv) implements a preconditioned Jacobi SVD algorithm. It uses **f08aef (dgeqrf)**, **f08ahf (dgelqf)** and **f08bff (dgeqp3)** as preprocessors and preconditioners. Optionally, an additional row pivoting can be used as a preprocessor, which in some cases results in much higher accuracy. An example is matrix A with the structure $A = D_1 C D_2$, where D_1, D_2 are arbitrarily ill-conditioned diagonal matrices and C is a well-conditioned matrix. In that case, complete pivoting in the first QR factorizations provides accuracy dependent on the condition number of C, and independent of D_1, D_2. Such higher accuracy is not completely understood theoretically, but it works well in practice. Further, if A can be written as $A = BD$, with well-conditioned B and some diagonal D, then the high accuracy is guaranteed, both theoretically and in software, independent of D.

10 Example

This example finds the singular values and left and right singular vectors of the 6 by 4 matrix

$$A = \begin{pmatrix} 2.27 & -1.54 & 1.15 & -1.94 \\ 0.28 & -1.67 & 0.94 & -0.78 \\ -0.48 & -3.09 & 0.99 & -0.21 \\ 1.07 & 1.22 & 0.79 & 0.63 \\ -2.35 & 2.93 & -1.45 & 2.30 \\ 0.62 & -7.39 & 1.03 & -2.57 \end{pmatrix},$$

together with the condition number of A and approximate error bounds for the computed singular values and vectors.

10.1 Program Text

```
    Program f08khfe

!     F08KHF Example Program Text

!     Mark 26.1 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: ddisna, dgejsv, nag_wp, x02ajf, x04caf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nb = 64, nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: eps, serrbd
      Integer                          :: i, ifail, info, j, lda, ldu, ldv,    &
                                          lwork, m, n
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: a(:,:), rcondu(:), rcondv(:), s(:),  &
                                          u(:,:), v(:,:), work(:)
      Integer, Allocatable             :: iwork(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: abs, max
!     .. Executable Statements ..
      Write (nout,*) 'F08KHF Example Program Results'
      Write (nout,*)
      Flush (nout)
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) m, n
      lda = m
      ldu = m
      ldv = n
      lwork = max(3*n+n*n+m,3*n+n*n+n*nb,7)
      Allocate (a(lda,n),rcondu(m),rcondv(m),s(n),u(ldu,n),v(ldv,n),           &
        work(lwork),iwork(m+3*n))

!     Read the m by n matrix A from data file
      Read (nin,*)((a(i,j),j=1,n),i=1,m)

!     Compute the singular values and left and right singular vectors
!     of A (A = U*S*V^T, m.ge.n)
!     The NAG name equivalent of dgejsv is f08khf
      Call dgejsv('E','U','V','R','N','N',m,n,a,lda,s,u,ldu,v,ldv,work,lwork,  &
        iwork,info)
      If (info==0) Then

!       Compute the approximate error bound for the computed singular values
!       using the 2-norm, s(1) = norm(A), and machine precision, eps.
        eps = x02ajf()
        serrbd = eps*s(1)

!       Print solution
        If (abs(work(1)-work(2))<2.0_nag_wp*eps) Then
!         No scaling required
          Write (nout,'(1X,A)') 'Singular values'
          Write (nout,99999)(s(j),j=1,n)
        Else
          Write (nout,'(/1X,A)') 'Scaled singular values'
          Write (nout,99999)(s(j),j=1,n)
          Write (nout,'(/1X,A)') 'For true singular values, multiply by a/b,'
          Write (nout,99996) ' where a = ', work(1), ' and b = ', work(2)
        End If

!       ifail: behaviour on error exit
!              =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
        Write (nout,*)
        Flush (nout)
        ifail = 0
        Call x04caf('General',' ',m,n,u,ldu,'Left singular vectors',ifail)

        Write (nout,*)
        Flush (nout)
        ifail = 0
        Call x04caf('General',' ',n,n,v,ldv,'Right singular vectors',ifail)

!       Call DDISNA (F08FLF) to estimate reciprocal condition numbers for
!       the singular vectors.

        Call ddisna('Left',m,n,s,rcondu,info)
        Call ddisna('Right',m,n,s,rcondv,info)

!       Print the approximate error bounds for the singular values
!       and vectors.
        Write (nout,*)
        Write (nout,'(/1X,A)')                                               &
          'Estimate of the condition number of column equilibrated A'
        Write (nout,99998) work(3)
        Write (nout,'(/1X,A)') 'Error estimate for the singular values'
        Write (nout,99998) serrbd
        Write (nout,'(/1X,A)') 'Error estimates for left singular vectors'
        Write (nout,99998)(serrbd/rcondu(i),i=1,n)
        Write (nout,'(/1X,A)') 'Error estimates for right singular vectors'
        Write (nout,99998)(serrbd/rcondv(i),i=1,n)
      Else
        Write (nout,99997) 'Failure in DGEJSV. INFO =', info
      End If

99999 Format (3X,8F8.4)
99998 Format (4X,1P,6E11.1)
99997 Format (1X,A,I4)
99996 Format (1X,2(A,1P,E13.5))
    End Program f08khfe
```

10.2 Program Data

```
F08KHF Example Program Data

   6      4                    :Values of M and N

   2.27  -1.54   1.15  -1.94
   0.28  -1.67   0.94  -0.78
  -0.48  -3.09   0.99  -0.21
   1.07   1.22   0.79   0.63
  -2.35   2.93  -1.45   2.30
   0.62  -7.39   1.03  -2.57  :End of matrix A
```

10.3 Program Results

```
 F08KHF Example Program Results

 Singular values
     9.9966  3.6831  1.3569  0.5000

 Left singular vectors
          1       2       3       4
 1   0.2774 -0.6003 -0.1277  0.1323
 2   0.2020 -0.0301  0.2805  0.7034
 3   0.2918  0.3348  0.6453  0.1906
 4  -0.0938 -0.3699  0.6781 -0.5399
 5  -0.4213  0.5266  0.0413 -0.0575
 6   0.7816  0.3353 -0.1645 -0.3957

 Right singular vectors
          1       2       3       4
 1   0.1921 -0.8030  0.0041 -0.5642
 2  -0.8794 -0.3926 -0.0752  0.2587
 3   0.2140 -0.2980  0.7827  0.5027
 4  -0.3795  0.3351  0.6178 -0.6017


 Estimate of the condition number of column equilibrated A
        9.0E+00

 Error estimate for the singular values
        1.1E-15

 Error estimates for left singular vectors
        1.8E-16    4.8E-16    1.3E-15    2.2E-15

 Error estimates for right singular vectors
        1.8E-16    4.8E-16    1.3E-15    1.3E-15
```