NAG Library Routine Document

F08KRF (ZGESDD)

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

F08KRF (ZGESDD) computes the singular value decomposition (SVD) of a complex m by n matrix A, optionally computing the left and/or right singular vectors, by using a divide-and-conquer method.

2 Specification

```
SUBROUTINE F08KRF (JOBZ, M, N, A, LDA, S, U, LDU, VT, LDVT, WORK, LWORK,    &
                   RWORK, IWORK, INFO)

INTEGER                 M, N, LDA, LDU, LDVT, LWORK, IWORK(8*min(M,N)),     &
                        INFO
REAL (KIND=nag_wp)      S(min(M,N)), RWORK(*)
COMPLEX (KIND=nag_wp)   A(LDA,*), U(LDU,*), VT(LDVT,*),                     &
                        WORK(max(1,LWORK))
CHARACTER(1)            JOBZ
```

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

3 Description

The SVD is written as

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

where Σ is an m by n matrix which is zero except for its $\min(m,n)$ diagonal elements, U is an m by m unitary matrix, and V is an n by n unitary matrix. The diagonal elements of Σ are the singular values of A; they are real and non-negative, and are returned in descending order. The first $\min(m,n)$ columns of U and V are the left and right singular vectors of A.

Note that the routine returns V^{H}, not V.

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 (1996) *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

5 Parameters

1: JOBZ – CHARACTER(1) *Input*

On entry: specifies options for computing all or part of the matrix U.

JOBZ = 'A'
: All m columns of U and all n rows of V^{H} are returned in the arrays U and VT.

JOBZ = 'S'
: The first $\min(m,n)$ columns of U and the first $\min(m,n)$ rows of V^{H} are returned in the arrays U and VT.

JOBZ = 'O'
If M $\geq$ N, the first n columns of U are overwritten on the array A and all rows of V^{H} are returned in the array VT. Otherwise, all columns of U are returned in the array U and the first m rows of V^{H} are overwritten in the array VT.

JOBZ = 'N'
No columns of U or rows of V^{H} are computed.

Constraint: JOBZ = 'A', 'S', 'O' or 'N'.

2: M – INTEGER *Input*

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

Constraint: M ≥ 0.

3: N – INTEGER *Input*

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

Constraint: N ≥ 0.

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

Note: the second dimension of the array A must be at least max(1, N).

On entry: the m by n matrix A.

On exit: if JOBZ = 'O', A is overwritten with the first n columns of U (the left singular vectors, stored column-wise) if M $\geq$ N; A is overwritten with the first m rows of V^{H} (the right singular vectors, stored row-wise) otherwise.

If JOBZ $\neq$ 'O', the contents of A are destroyed.

5: LDA – INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F08KRF (ZGESDD) is called.

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

6: S(min(M, N)) – REAL (KIND=nag_wp) array *Output*

On exit: the singular values of A, sorted so that S(i) $\geq$ S($i+1$).

7: U(LDU, $*$) – COMPLEX (KIND=nag_wp) array *Output*

Note: the second dimension of the array U must be at least max(1, M) if JOBZ = 'A' or JOBZ = 'O' and M < N, max(1, min(M, N)) if JOBZ = 'S', and at least 1 otherwise.

On exit:

If JOBZ = 'A' or JOBZ = 'O' and M < N, U contains the m by m unitary matrix U.

If JOBZ = 'S', U contains the first min(m, n) columns of U (the left singular vectors, stored column-wise).

If JOBZ = 'O' and M $\geq$ N, or JOBZ = 'N', U is not referenced.

8: LDU – INTEGER *Input*

On entry: the first dimension of the array U as declared in the (sub)program from which F08KRF (ZGESDD) is called.

Constraints:

if JOBZ = 'S' or 'A' or JOBZ = 'O' and M < N, LDU $\geq$ max(1, M);
otherwise LDU ≥ 1.

9: VT(LDVT, $*$) – COMPLEX (KIND=nag_wp) array *Output*

Note: the second dimension of the array VT must be at least $\max(1, \mathrm{N})$ if JOBZ = 'A' or 'S' or JOBZ = 'O' and M $\geq$ N, and at least 1 otherwise.

On exit: if JOBZ = 'A' or JOBZ = 'O' and M $\geq$ N, VT contains the n by n unitary matrix V^{H}.

If JOBZ = 'S', VT contains the first $\min(m, n)$ rows of V^{H} (the right singular vectors, stored row-wise).

If JOBZ = 'O' and M $<$ N, or JOBZ = 'N', VT is not referenced.

10: LDVT – INTEGER *Input*

On entry: the first dimension of the array VT as declared in the (sub)program from which F08KRF (ZGESDD) is called.

Constraints:

if JOBZ = 'A' or JOBZ = 'O' and M $\geq$ N, LDVT $\geq \max(1, \mathrm{N})$;
if JOBZ = 'S', LDVT $\geq \max(1, \min(\mathrm{M}, \mathrm{N}))$;
otherwise LDVT ≥ 1.

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

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

12: LWORK – INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08KRF (ZGESDD) is called.

If 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 should generally be larger. Consider increasing LWORK by at least $nb \times \min(\mathrm{M}, \mathrm{N})$, where nb is the optimal ***block size***.

Constraints:

if JOBZ = 'N', LWORK $\geq 2 \times \min(\mathrm{M}, \mathrm{N}) + \max(1, \mathrm{M}, \mathrm{N})$;
if JOBZ = 'O', LWORK $\geq 2 \times \min(\mathrm{M}, \mathrm{N}) \times \min(\mathrm{M}, \mathrm{N}) + 2 \times \min(\mathrm{M}, \mathrm{N}) + \max(1, \mathrm{M}, \mathrm{N})$;
if JOBZ = 'S' or 'A',
LWORK $\geq \min(\mathrm{M}, \mathrm{N}) \times \min(\mathrm{M}, \mathrm{N}) + 2 \times \min(\mathrm{M}, \mathrm{N}) + \max(1, \mathrm{M}, \mathrm{N})$;
otherwise LWORK ≥ 1.

13: RWORK($*$) – REAL (KIND=nag_wp) array *Workspace*

Note: the dimension of the array RWORK must be at least $\max(1, 5 \times \min(\mathrm{M}, \mathrm{N}))$ if JOBZ = 'N', and at least $\max(1, \min(\mathrm{M}, \mathrm{N}) \times \max(5 \times \min(\mathrm{M}, \mathrm{N}) + 7, 2 \times \max(\mathrm{M}, \mathrm{N}) + 2 \times \min(\mathrm{M}, \mathrm{N}) + 1))$ otherwise.

14: IWORK($8 \times \min(\mathrm{M}, \mathrm{N})$) – INTEGER array *Workspace*

15: 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 > 0

F08KRF (ZGESDD) did not converge, the updating process failed.

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

F08KRF (ZGESDD) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

F08KRF (ZGESDD) 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

The total number of floating-point operations is approximately proportional to mn^2 when $m > n$ and m^2n otherwise.

The singular values are returned in descending order.

The real analogue of this routine is F08KDF (DGESDD).

10 Example

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

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

together with approximate error bounds for the computed singular values and vectors.

The example program for F08KPF (ZGESVD) illustrates finding a singular value decomposition for the case $m \geq n$.

10.1 Program Text

```
    Program f08krfe

!     F08KRF Example Program Text

!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: nag_wp, zgesdd
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter                    :: nb = 64, nin = 5, nout = 6,    &
                                               prerr = 0
!     .. Local Scalars ..
      Integer                               :: i, info, lda, ldu, ldvt,       &
                                               lwork, m, n
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable    :: a(:,:), a_copy(:,:), b(:),     &
                                               u(:,:), vt(:,:), work(:)
      Complex (Kind=nag_wp)                 :: dummy(1,1)
      Real (Kind=nag_wp), Allocatable       :: rwork(:), s(:)
      Integer, Allocatable                  :: iwork(:)
!     .. Intrinsic Procedures ..
      Intrinsic                             :: max, min, nint, real
!     .. Executable Statements ..
      Write (nout,*) 'F08KRF Example Program Results'
      Write (nout,*)
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) m, n
      lda = m
      ldu = m
      ldvt = n
      Allocate (a(lda,n),a_copy(m,n),s(m),u(ldu,m),vt(ldvt,n),b(m),rwork((5*m+ &
        7)*n),iwork(8*m))

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

!     Read the right hand side of the linear system
      Read (nin,*) b(1:m)

      a_copy(1:m,1:n) = a(1:m,1:n)

!     Use routine workspace query to get optimal workspace.
      lwork = -1
!     The NAG name equivalent of dgesdd is f08krf
      Call zgesdd('A',m,n,a,lda,s,u,ldu,vt,ldvt,dummy,lwork,rwork,iwork,info)

!     Make sure that there is enough workspace for blocksize nb.
      lwork = max((2*m+2)*m+2*n+nb*(m+n),nint(real(dummy(1,1))))
      Allocate (work(lwork))

!     Compute the singular values and left and right singular vectors
!     of A.

!     The NAG name equivalent of dgesdd is f08krf
      Call zgesdd('A',m,n,a,lda,s,u,ldu,vt,ldvt,work,lwork,rwork,iwork,info)

      If (info/=0) Then
        Write (nout,99999) 'Failure in F08KRF/ZGESDD. INFO =', info
99999   Format (1X,A,I4)
        Go To 100
      End If

!     Print the significant singular values of A

      Write (nout,*) 'Singular values of A:'
      Write (nout,99998) s(1:min(m,n))
```

```
99998 Format (1X,4(3X,F11.4))

      If (prerr>0) Then
        Call compute_error_bounds(m,n,s)
      End If

      If (m<n) Then
!       Compute V*Inv(S)*U^T * b to get minimum norm solution.
        Call compute_minimum_norm(m,n,a_copy,m,u,ldu,vt,ldvt,s,b)
      End If

100   Continue

    Contains
      Subroutine compute_minimum_norm(m,n,a,lda,u,ldu,vt,ldvt,s,b)

!       .. Use Statements ..
        Use nag_library, Only: dznrm2, zgemv
!       .. Implicit None Statement ..
        Implicit None
!       .. Scalar Arguments ..
        Integer, Intent (In)                    :: lda, ldu, ldvt, m, n
!       .. Array Arguments ..
        Complex (Kind=nag_wp), Intent (In)      :: a(lda,n), u(ldu,m), vt(ldvt,n)
        Complex (Kind=nag_wp), Intent (Inout)   :: b(m)
        Real (Kind=nag_wp), Intent (In)         :: s(m)
!       .. Local Scalars ..
        Complex (Kind=nag_wp)                   :: alpha, beta
        Real (Kind=nag_wp)                      :: norm
!       .. Local Arrays ..
        Complex (Kind=nag_wp), Allocatable      :: x(:), y(:)
!       .. Intrinsic Procedures ..
        Intrinsic                               :: allocated, cmplx
!       .. Executable Statements ..
        Allocate (x(n),y(m))

!       Compute V*Inv(S)*U^H * b to get least-squares solution.

!       y = U^H b
!       The NAG name equivelent of zgemv is f06saf
        alpha = cmplx(1.0_nag_wp,0.0_nag_wp,kind=nag_wp)
        beta = cmplx(0.0_nag_wp,0.0_nag_wp,kind=nag_wp)
        Call zgemv('C',m,m,alpha,u,ldu,b,1,beta,y,1)

        y(1:m) = y(1:m)/s(1:m)

!       x = V y
        Call zgemv('C',m,n,alpha,vt,ldvt,y,1,beta,x,1)

        Write (nout,*)
        Write (nout,*) 'Minimum norm solution:'
        Write (nout,99999) x(1:n)

        norm = dznrm2(n,x,1)

        Write (nout,*)
        Write (nout,*) 'Norm of Solution:'
        Write (nout,99998) norm

!       Find norm of residual ||b-Ax||, should be zero.
        alpha = cmplx(-1.0_nag_wp,0.0_nag_wp,kind=nag_wp)
        beta = cmplx(1._nag_wp,0.0_nag_wp,kind=nag_wp)
        Call zgemv('N',m,n,alpha,a,lda,x,1,beta,b,1)

        norm = dznrm2(m,b,1)

        Write (nout,*)
        Write (nout,*) 'Norm of Residual:'
        Write (nout,99998) norm

        If (allocated(x)) Deallocate (x)
```

```
        If (allocated(y)) Deallocate (y)

99999   Format (4X,'(',F8.4,',',F8.4,')')
99998   Format (4X,F11.4)

      End Subroutine compute_minimum_norm

      Subroutine compute_error_bounds(m,n,s)

!       Error estimates for singular values and vectors is computed
!       and printed here.

!       .. Use Statements ..
        Use nag_library, Only: ddisna, nag_wp, x02ajf
!       .. Implicit None Statement ..
        Implicit None
!       .. Scalar Arguments ..
        Integer, Intent (In)                    :: m, n
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (In)         :: s(m)
!       .. Local Scalars ..
        Real (Kind=nag_wp)                      :: eps, serrbd
        Integer                                 :: i, info
!       .. Local Arrays ..
        Real (Kind=nag_wp), Allocatable         :: rcondu(:), rcondv(:),        &
                                                   uerrbd(:), verrbd(:)
!       .. Executable Statements ..
        Allocate (rcondu(n),rcondv(n),uerrbd(n),verrbd(n))

!       Get the machine precision, EPS and compute the approximate
!       error bound for the computed singular values.  Note that for
!       the 2-norm, S(1) = norm(A)

        eps = x02ajf()
        serrbd = eps*s(1)

!       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)

!       Compute the error estimates for the singular vectors

        Do i = 1, n
          uerrbd(i) = serrbd/rcondu(i)
          verrbd(i) = serrbd/rcondv(i)
        End Do

!       Print the approximate error bounds for the singular values
!       and vectors

        Write (nout,*)
        Write (nout,*) 'Error estimate for the singular values'
        Write (nout,99999) serrbd
        Write (nout,*)
        Write (nout,*) 'Error estimates for the left singular vectors'
        Write (nout,99999) uerrbd(1:n)
        Write (nout,*)
        Write (nout,*) 'Error estimates for the right singular vectors'
        Write (nout,99999) verrbd(1:n)

99999   Format (4X,1P,6E11.1)

      End Subroutine compute_error_bounds

    End Program f08krfe
```

10.2 Program Data

```
F08KRF Example Program Data

   4            6                          : m and n

 ( 0.96, 0.81) (-0.98,-1.98) ( 0.62, 0.46)
 (-0.37,-0.38) ( 0.83,-0.51) ( 1.08, 0.28)

 (-0.03,-0.96) (-1.20,-0.19) ( 1.01,-0.02)
 ( 0.19, 0.54) ( 0.20,-0.01) ( 0.20, 0.12)

 (-0.91,-2.06) (-0.66,-0.42) ( 0.63, 0.17)
 (-0.98, 0.36) (-0.17, 0.46) (-0.07,-1.23)

 (-0.05,-0.41) (-0.81,-0.56) (-1.11,-0.60)
 ( 0.22, 0.20) ( 1.47,-1.59) ( 0.26,-0.26) : Matrix A(1:m,1:n)

 ( 1.00, 0.00) ( 1.00, 0.00) ( 1.00, 0.00)
 ( 1.00, 0.00)                             : RHS b(1:n)
```

10.3 Program Results

```
 F08KRF Example Program Results

 Singular values of A:
         3.9994        3.0003        1.9944        0.9995

 Minimum norm solution:
    ( -0.4024,  0.3777)
    ( -0.2272,  0.3626)
    (  0.1704, -0.1532)
    (  0.2125,  0.0781)
    (  0.2041,  0.2236)
    (  0.2766, -0.1517)

 Norm of Solution:
         0.8846

 Norm of Residual:
         0.0000
```