NAG Library Routine Document

F08KMF (DGESVDX)

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

F08KMF (DGESVDX) computes the singular value decomposition (SVD) of a real m by n matrix A, optionally computing the left and/or right singular vectors. All singular values or a selected set of singular values may be computed.

2 Specification

```
SUBROUTINE F08KMF (JOBU, JOBVT, RANGE, M, N, A, LDA, VL, VU, IL, IU, NS,    &
                   S, U, LDU, VT, LDVT, WORK, LWORK, IWORK, INFO)
INTEGER            M, N, LDA, IL, IU, NS, LDU, LDVT, LWORK,                &
                   IWORK(12*min(N,M)), INFO
REAL (KIND=nag_wp) A(LDA,*), VL, VU, S(min(M,N)), U(LDU,*),                &
                   VT(LDVT,*), WORK(max(1,LWORK))
CHARACTER(1)       JOBU, JOBVT, RANGE
```

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

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 $\min(m,n)$ 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; 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^{T}, not V.

Alternative to computing all singular values of A, a selected set can be computed. The set is either those singular values lying in a given interval, $\sigma \in (v_l, v_u]$, or those whose index (counting from largest to smallest in magnitude) lies in a given range $1 \leq i_l, \ldots, i_u \leq n$. In these cases, the corresponding left and right singular vectors can optionally be computed.

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 Arguments

1: JOBU – CHARACTER(1) *Input*

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

JOBU = 'V'
    The NS columns of U, as specified by RANGE, are returned in array U.

JOBU = 'N'
: No columns of U (no left singular vectors) are computed.

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

2: JOBVT – CHARACTER(1) *Input*

On entry: specifies options for computing all or part of the matrix V^{T}.

JOBVT = 'V'
: The NS rows of V^{T}, as specified by RANGE, are returned in the array VT.

JOBVT = 'N'
: No rows of V^{T} (no right singular vectors) are computed.

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

3: RANGE – CHARACTER(1) *Input*

On entry: indicates which singular values should be returned.

RANGE = 'A'
: All singular values will be found.

RANGE = 'V'
: All singular values in the half-open interval $(\mathrm{VL}, \mathrm{VU}]$ will be found.

RANGE = 'I'
: The ILth through IUth singular values will be found.

Constraint: RANGE = 'A', 'V' or 'I'.

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 matrix A.

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

6: $\mathrm{A}(\mathrm{LDA}, *)$ – REAL (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: if JOBU $\neq$ 'N' and JOBVT $\neq$ 'N', the contents of A are destroyed.

7: LDA – INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F08KMF (DGESVDX) is called.

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

8: VL – REAL (KIND=nag_wp) *Input*

On entry: if RANGE = 'V', the lower bounds of the interval to be searched for singular values.

If RANGE = 'A' or 'I', VL is not referenced.

Constraint: if RANGE = 'V', $0.0 \leq \mathrm{VL}$.

9: VU – REAL (KIND=nag_wp) *Input*

On entry: if RANGE = 'V', the upper bounds of the interval to be searched for singular values.

If RANGE = 'A' or 'I', VU is not referenced.

Constraint: if RANGE = 'V', VL < VU.

10: IL – INTEGER *Input*
11: IU – INTEGER *Input*

On entry: if RANGE = 'I', the indices (in ascending order) of the smallest and largest singular values to be returned.

If RANGE = 'A' or 'V', IL and IU are not referenced.

Constraints:

if RANGE = 'I' and $\min(M,N) = 0$, IL = 1 and IU = 0;
if RANGE = 'I' and $\min(M,N) > 0$, $1 \le IL \le IU \le \min(M,N)$.

12: NS – INTEGER *Output*

On exit: the total number of singular values found. $0 \le NS \le \min(M,N)$.

If RANGE = 'A', $NS = \min(M,N)$.

If RANGE = 'I', $NS = IU - IL + 1$.

If RANGE = 'V' then the value of NS is not known in advance and so an upper limit should be used when specifying array dimensions, e.g., $\min(M,N)$.

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

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

14: $U(LDU,*)$ – REAL (KIND=nag_wp) array *Output*

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

Where $nsmax$ is a value larger than the output value NS.

On exit: m by NS.

If JOBU = 'V', U contains the first NS columns of U (the left singular vectors, stored columnwise).

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

15: LDU – INTEGER *Input*

On entry: the first dimension of the array U as declared in the (sub)program from which F08KMF (DGESVDX) is called.

Constraints:

if JOBU = 'V', $LDU \ge \max(1,M)$;
otherwise $LDU \ge 1$.

16: $VT(LDVT,*)$ – REAL (KIND=nag_wp) array *Output*

Note: the second dimension of the array VT must be at least $\max(1,N)$ if JOBVT = 'V', and at least 1 otherwise.

On exit: if JOBVT = 'V', VT contains the first NS rows of V^T (the right singular vectors, stored row-wise).

If JOBVT = 'N', VT is not referenced.

17: LDVT – INTEGER *Input*

Note: If JOBVT = 'V' and RANGE = 'V' then the value of NS is not known in advance and so an upper limit should be used, e.g., min(M, N).

On entry: the first dimension of the array VT as declared in the (sub)program from which F08KMF (DGESVDX) is called.

Constraints:

if JOBVT = 'V', LDVT $\geq$ max(1, NS);
otherwise LDVT $\geq$ 1.

18: WORK(max(1, LWORK)) – REAL (KIND=nag_wp) array *Workspace*

On exit: if INFO = 0, WORK(1) returns the optimal LWORK.

If INFO > 0, WORK(2 : min(M, N)) contains the unconverged superdiagonal elements of an upper bidiagonal matrix B whose diagonal is in S (not necessarily sorted). B satisfies $A = UBV^{\mathrm{T}}$, so it has the same singular values as A, and singular vectors related by U and V^{T}.

19: LWORK – INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08KMF (DGESVDX) 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(M, N), where nb is the optimal ***block size***.

Constraint: LWORK $\geq$ max(1, 3 $\times$ min(M, N) + max(M, N), 5 $\times$ min(M, N)).

20: IWORK(12 $\times$ min(N, M)) – INTEGER array *Workspace*

On exit:

if INFO = 0, the first NS elements of IWORK are zero;

if INFO > 0, IWORK contains the indices of the eigenvectors that failed to converge in F08JBF (DSTEVX) and F08MBF (DBDSVDX).

21: 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

If F08KMF (DGESVDX) did not converge, INFO specifies how many superdiagonals of an intermediate bidiagonal form did not converge to zero.

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

F08KMF (DGESVDX) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

F08KMF (DGESVDX) 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 complex analogue of this routine is F08KZF (ZGESVDX).

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 approximate error bounds for the computed singular values and vectors.

10.1 Program Text

```
    Program f08kmfe

!     F08KMF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: dgesvdx, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: vl, vu
      Integer                          :: i, il, info, iu, lda, ldu, ldvt,     &
                                          lwork, m, n, ns
      Character (1)                    :: range
!     .. Local Arrays ..
```

```
      Real (Kind=nag_wp), Allocatable  :: a(:,:), a_copy(:,:), b(:), s(:),     &
                                          u(:,:), vt(:,:), work(:)
      Real (Kind=nag_wp)               :: dummy(1,1)
      Integer, Allocatable             :: iwork(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: nint
!     .. Executable Statements ..
      Write (nout,*) 'F08KMF 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(n),vt(ldvt,n),u(ldu,m),b(m),            &
        iwork(12*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)

!     Read range for selected singular values
      Read (nin,*) range

      If (range=='I' .Or. range=='i') Then
        Read (nin,*) il, iu
      Else If (range=='V' .Or. range=='v') Then
        Read (nin,*) vl, vu
      End If

      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 dgesvdx is f08kmf
      Call dgesvdx('V','V',range,m,n,a,lda,vl,vu,il,iu,ns,s,u,ldu,vt,ldvt,     &
        dummy,lwork,iwork,info)

!     Make sure that there is enough workspace for block size nb.
      lwork = nint(dummy(1,1))
      Allocate (work(lwork))

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

!     The NAG name equivalent of dgesvd is f08kmf
      Call dgesvdx('V','V',range,m,n,a,lda,vl,vu,il,iu,ns,s,u,ldu,vt,ldvt,     &
        work,lwork,iwork,info)

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

!     Print the selected singular values of A

      Write (nout,*) 'Singular values of A:'
      Write (nout,99998) s(1:ns)
99998 Format (1X,4(3X,F11.4))

      Call compute_error_bounds(m,ns,s)

      If (m>n .And. ns==n) Then
!       Compute V*Inv(S)*U^T * b to get least squares solution.
        Call compute_least_squares(m,n,a_copy,m,u,ldu,vt,ldvt,s,b)
      End If
```

```
100   Continue

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

!       .. Use Statements ..
        Use nag_library, Only: dgemv, dnrm2
!       .. Implicit None Statement ..
        Implicit None
!       .. Scalar Arguments ..
        Integer, Intent (In)           :: lda, ldu, ldvt, m, n
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (In) :: a(lda,n), s(n), u(ldu,m),           &
                                          vt(ldvt,n)
        Real (Kind=nag_wp), Intent (Inout) :: b(m)
!       .. Local Scalars ..
        Real (Kind=nag_wp)             :: alpha, beta, norm
!       .. Local Arrays ..
        Real (Kind=nag_wp), Allocatable :: x(:), y(:)
!       .. Intrinsic Procedures ..
        Intrinsic                      :: allocated
!       .. Executable Statements ..
        Allocate (x(n),y(n))

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

!       y = U^T b
!       The NAG name equivalent of dgemv is f06paf
        alpha = 1._nag_wp
        beta = 0._nag_wp
        Call dgemv('T',m,n,alpha,u,ldu,b,1,beta,y,1)

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

!       x = V y
        Call dgemv('T',n,n,alpha,vt,ldvt,y,1,beta,x,1)

        Write (nout,*)
        Write (nout,*) 'Least squares solution:'
        Write (nout,99999) x(1:n)

!       Find norm of residual ||b-Ax||.
        alpha = -1._nag_wp
        beta = 1._nag_wp
        Call dgemv('N',m,n,alpha,a,lda,x,1,beta,b,1)

        norm = dnrm2(m,b,1)

        Write (nout,*)
        Write (nout,*) 'Norm of Residual:'
        Write (nout,99999) norm

        If (allocated(x)) Then
          Deallocate (x)
        End If
        If (allocated(y)) Then
          Deallocate (y)
        End If

99999   Format (1X,4(3X,F11.4))

      End Subroutine compute_least_squares

      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(n)
!       .. 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,*) 'Estimates given as multiples of machine precision'
        Write (nout,*) 'Error estimate for the singular values'
        Write (nout,99999) nint(serrbd/x02ajf())
        Write (nout,*)
        Write (nout,*) 'Error estimates for the left singular vectors'
        Write (nout,99999) nint(uerrbd(1:n)/x02ajf())
        Write (nout,*)
        Write (nout,*) 'Error estimates for the right singular vectors'
        Write (nout,99999) nint(verrbd(1:n)/x02ajf())

99999   Format (4X,6I11)

      End Subroutine compute_error_bounds

    End Program f08kmfe
```

10.2 Program Data

```
F08KMF 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
```

```
  1.0    1.0    1.0    1.0
  1.0    1.0                 : RHS b(1:m)

  'V'                        : range
  1.00   4.00                : (vl,vu) or (il,iu)
```

10.3 Program Results

```
F08KMF Example Program Results

Singular values of A:
        3.6831        1.3569

Estimates given as multiples of machine precision
Error estimate for the singular values
             4

Error estimates for the left singular vectors
             2          3

Error estimates for the right singular vectors
             2          2
```