NAG Toolbox for MATLAB

f08me

1 Purpose

f08me computes the singular value decomposition of a real upper or lower bidiagonal matrix, or of a real general matrix which has been reduced to bidiagonal form.

2 Syntax

```
[d, e, vt, u, c, info] = f08me(uplo, d, e, vt, u, c, 'n', n, 'ncvt',
ncvt, 'nru', nru, 'ncc', ncc)
```

3 Description

f08me computes the singular values and, optionally, the left or right singular vectors of a real upper or lower bidiagonal matrix B. In other words, it can compute the singular value decomposition (SVD) of B as

$$B = U\Sigma V^{\mathrm{T}}.$$

Here Σ is a diagonal matrix with real diagonal elements σ_i (the singular values of B), such that

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0;$$

U is an orthogonal matrix whose columns are the left singular vectors u_i; V is an orthogonal matrix whose rows are the right singular vectors v_i. Thus

$$Bu_i = \sigma_i v_i \qquad \text{and} \qquad B^{\mathrm{T}} v_i = \sigma_i u_i, \qquad i = 1, 2, \ldots, n.$$

To compute U and/or V^{T}, the arrays **u** and/or **vt** must be initialized to the unit matrix before f08me is called.

The function may also be used to compute the SVD of a real general matrix A which has been reduced to bidiagonal form by an orthogonal transformation: $A = QBP^{\mathrm{T}}$. If A is m by n with $m \geq n$, then Q is m by n and P^{T} is n by n; if A is n by p with $n < p$, then Q is n by n and P^{T} is n by p. In this case, the matrices Q and/or P^{T} must be formed explicitly by f08kf and passed to f08me in the arrays **u** and/or **vt** respectively.

f08me also has the capability of forming $U^{\mathrm{T}}C$, where C is an arbitrary real matrix; this is needed when using the SVD to solve linear least-squares problems.

f08me uses two different algorithms. If any singular vectors are required (i.e., if **ncvt** > 0 or **nru** > 0 or **ncc** > 0), the bidiagonal QR algorithm is used, switching between zero-shift and implicitly shifted forms to preserve the accuracy of small singular values, and switching between QR and QL variants in order to handle graded matrices effectively (see Demmel and Kahan 1990). If only singular values are required (that is, if **ncvt** $=$ **nru** $=$ **ncc** $= 0$), they are computed by the differential qd algorithm (see Fernando and Parlett 1994), which is faster and can achieve even greater accuracy.

The singular vectors are normalized so that $\|u_i\| = \|v_i\| = 1$, but are determined only to within a factor ± 1.

4 References

Demmel J W and Kahan W 1990 Accurate singular values of bidiagonal matrices *SIAM J. Sci. Statist. Comput.* **11** 873–912

Fernando K V and Parlett B N 1994 Accurate singular values and differential qd algorithms *Numer. Math.* **67** 191–229

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: **uplo – string**

Indicates whether B is an upper or lower bidiagonal matrix.

uplo = 'U'

B is an upper bidiagonal matrix.

uplo = 'L'

B is a lower bidiagonal matrix.

Constraint: **uplo** = 'U' or 'L'.

2: **d**$(*)$ **– double array**

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

The diagonal elements of the bidiagonal matrix B.

3: **e**$(*)$ **– double array**

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

The off-diagonal elements of the bidiagonal matrix B.

4: **vt(ldvt,**$*$**) – double array**

The first dimension, **ldvt**, of the array **vt** must satisfy

if **ncvt** > 0, **ldvt** $\geq \max(1,\mathbf{n})$;
ldvt ≥ 1 otherwise.

The second dimension of the array must be at least $\max(1,\mathbf{ncvt})$

If **ncvt** > 0, **vt** must contain an n by $ncvt$ matrix. If the right singular vectors of B are required, $ncvt = n$ and **vt** must contain the unit matrix; if the right singular vectors of A are required, **vt** must contain the orthogonal matrix P^{T} returned by f08kf with **vect** = 'P'.

5: **u(ldu,**$*$**) – double array**

The first dimension of the array **u** must be at least $\max(1,\mathbf{nru})$

The second dimension of the array must be at least $\max(1,\mathbf{n})$

If **nru** > 0, **u** must contain an nru by n matrix. If the left singular vectors of B are required, $nru = n$ and **u** must contain the unit matrix; if the left singular vectors of A are required, **u** must contain the orthogonal matrix Q returned by f08kf with **vect** = 'Q'.

6: **c(ldc,**$*$**) – double array**

The first dimension, **ldc**, of the array **c** must satisfy

if **ncc** > 0, **ldc** $\geq \max(1,\mathbf{n})$;
ldc ≥ 1 otherwise.

The second dimension of the array must be at least $\max(1,\mathbf{ncc})$

The n by ncc matrix C if **ncc** > 0.

5.2 Optional Input Parameters

1: **n – int32 scalar**

Default: The dimension of the array **d** The second dimension of the array **u**.

n, the order of the matrix *B*.

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

2: **ncvt – int32 scalar**

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

$ncvt$, the number of columns of the matrix V^{T} of right singular vectors. Set $\mathbf{ncvt} = 0$ if no right singular vectors are required.

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

3: **nru – int32 scalar**

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

nru, the number of rows of the matrix U of left singular vectors. Set $\mathbf{nru} = 0$ if no left singular vectors are required.

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

4: **ncc – int32 scalar**

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

ncc, the number of columns of the matrix C. Set $\mathbf{ncc} = 0$ if no matrix C is supplied.

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

5.3 Input Parameters Omitted from the MATLAB Interface

ldvt, ldu, ldc, work

5.4 Output Parameters

1: **d(∗) – double array**

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

The singular values in decreasing order of magnitude, unless $\mathbf{info} > 0$ (in which case see Section 6).

2: **e(∗) – double array**

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

e is overwritten, but if $\mathbf{info} > 0$ see Section 6.

3: **vt(ldvt,∗) – double array**

The first dimension, **ldvt**, of the array **vt** must satisfy

if $\mathbf{ncvt} > 0$, $\mathbf{ldvt} \geq \max(1, \mathbf{n})$;
$\mathbf{ldvt} \geq 1$ otherwise.

The second dimension of the array must be at least $\max(1, \mathbf{ncvt})$

The n by $ncvt$ matrix V^{T} or $V^{\mathrm{T}}P^{\mathrm{T}}$ of right singular vectors, stored by rows.

If $\mathbf{ncvt} = 0$, **vt** is not referenced.

4: **u(ldu,$*$) – double array**

The first dimension of the array **u** must be at least $\max(1, \mathbf{nru})$

The second dimension of the array must be at least $\max(1, \mathbf{n})$

The nru by n matrix U or QU of left singular vectors, stored as columns of the matrix.

If $\mathbf{nru} = 0$, **u** is not referenced.

5: **c(ldc,$*$) – double array**

The first dimension, **ldc**, of the array **c** must satisfy

if $\mathbf{ncc} > 0$, $\mathbf{ldc} \geq \max(1, \mathbf{n})$;
$\mathbf{ldc} \geq 1$ otherwise.

The second dimension of the array must be at least $\max(1, \mathbf{ncc})$

c contains the matrix $U^{\mathrm{T}}C$. If $\mathbf{ncc} = 0$, **c** is not referenced.

6: **info – int32 scalar**

$\mathbf{info} = 0$ unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

$\mathbf{info} = -i$

If $\mathbf{info} = -i$, parameter i had an illegal value on entry. The parameters are numbered as follows:

1: **uplo**, 2: **n**, 3: **ncvt**, 4: **nru**, 5: **ncc**, 6: **d**, 7: **e**, 8: **vt**, 9: **ldvt**, 10: **u**, 11: **ldu**, 12: **c**, 13: **ldc**, 14: **work**, 15: **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.

$\mathbf{info} > 0$

The algorithm failed to converge and **info** specifies how many off-diagonals did not converge. In this case, **d** and **e** contain on exit the diagonal and off-diagonal elements, respectively, of a bidiagonal matrix orthogonally equivalent to B.

7 Accuracy

Each singular value and singular vector is computed to high relative accuracy. However, the reduction to bidiagonal form (prior to calling the function) may exclude the possibility of obtaining high relative accuracy in the small singular values of the original matrix if its singular values vary widely in magnitude.

If σ_i is an exact singular value of B and $\tilde{\sigma}_i$ is the corresponding computed value, then

$$|\tilde{\sigma}_i - \sigma_i| \leq p(m,n)\epsilon\sigma_i$$

where $p(m,n)$ is a modestly increasing function of m and n, and ϵ is the ***machine precision***. If only singular values are computed, they are computed more accurately (i.e., the function $p(m,n)$ is smaller), than when some singular vectors are also computed.

If u_i is the corresponding exact left singular vector of B, and $\tilde{u}_i$ is the corresponding computed left singular vector, then the angle $\theta(\tilde{u}_i, u_i)$ between them is bounded as follows:

$$\theta(\tilde{u}_i, u_i) \leq \frac{p(m,n)\epsilon}{relgap_i}$$

where $relgap_i$ is the relative gap between σ_i and the other singular values, defined by

$$relgap_i = \min_{i \neq j} \frac{|\sigma_i - \sigma_j|}{(\sigma_i + \sigma_j)}.$$

A similar error bound holds for the right singular vectors.

8 Further Comments

The total number of floating-point operations is roughly proportional to n^2 if only the singular values are computed. About $6n^2 \times nru$ additional operations are required to compute the left singular vectors and about $6n^2 \times ncvt$ to compute the right singular vectors. The operations to compute the singular values must all be performed in scalar mode; the additional operations to compute the singular vectors can be vectorized and on some machines may be performed much faster.

The complex analogue of this function is f08ms.

9 Example

```
uplo = 'U';
d = [3.62;
     -2.41;
     1.92;
     -1.43];
e = [1.26;
     -1.53;
     1.19];
vt = [1, 0, 0, 0;
     0, 1, 0, 0;
     0, 0, 1, 0;
     0, 0, 0, 1];
u = [1, 0, 0, 0;
     0, 1, 0, 0;
     0, 0, 1, 0;
     0, 0, 0, 1];
c = [];
[dOut, eOut, vtOut, uOut, cOut, info] = f08me(uplo, d, e, vt, u, c)
```

```
dOut =
    4.0001
    3.0006
    1.9960
    0.9998
eOut =
     0
     0
     0
vtOut =
    0.8261    0.5246    0.2024    0.0369
    0.4512   -0.4056   -0.7350   -0.3030
    0.2823   -0.5644    0.1731    0.7561
    0.1852   -0.4916    0.6236   -0.5789
uOut =
    0.9129    0.3740    0.1556    0.0512
   -0.3935    0.7005    0.5489    0.2307
    0.1081   -0.5904    0.6173    0.5086
   -0.0132    0.1444   -0.5417    0.8280
cOut =
info =
           0
```