NAG Toolbox for MATLAB

f08kc

1 Purpose

f08kc computes the minimum norm solution to a real linear least-squares problem

$$\min_x \|b - Ax\|_2.$$

2 Syntax

```
[a, b, s, rank, info] = f08kc(a, b, rcond, 'm', m, 'n', n, 'nrhs_p',
nrhs_p, 'lwork', lwork)
```

3 Description

f08kc uses the singular value decomposition (SVD) of A, where A is an m by n matrix which may be rank-deficient.

Several right-hand side vectors b and solution vectors x can be handled in a single call; they are stored as the columns of the m by r right-hand side matrix B and the n by r solution matrix X.

The problem is solved in three steps:

1. reduce the coefficient matrix A to bidiagonal form with Householder transformations, reducing the original problem into a 'bidiagonal least-squares problem' (BLS);
2. solve the BLS using a divide-and-conquer approach;
3. apply back all the Householder transformations to solve the original least-squares problem.

The effective rank of A is determined by treating as zero those singular values which are less than **rcond** times the largest singular value.

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 URL: 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

5.1 Compulsory Input Parameters

1: **a**(**lda**,$*$) – **double array**

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

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

The m by n matrix A.

2: **b**(**ldb**,$*$) – **double array**

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

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

The m by r right-hand side matrix B.

3: **rcond – double scalar**

Used to determine the effective rank of A. Singular values $\mathbf{s}(i) \leq \mathbf{rcond} \times \mathbf{s}(1)$ are treated as zero. If $\mathbf{rcond} < 0$, ***machine precision*** is used instead.

5.2 Optional Input Parameters

1: **m – int32 scalar**

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

m, the number of rows of the matrix A.

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

2: **n – int32 scalar**

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

n, the number of columns of the matrix A.

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

3: **nrhs_p – int32 scalar**

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

r, the number of right-hand sides, i.e., the number of columns of the matrices B and X.

Constraint: $\mathbf{nrhs\_p} \geq 0$.

4: **lwork – int32 scalar**

Default: The dimension of the array .

The exact minimum amount of workspace needed depends on **m**, **n** and **nrhs_p**. As long as **lwork** is at least

$$12r + 2r \times smlsiz + 8r \times nlvl + r \times \mathbf{nrhs\_p} + (smlsiz + 1)^2,$$

where $smlsiz$ is equal to the maximum size of the subproblems at the bottom of the computation tree (usually about 25), $nlvl = \max(0, \mathrm{int}(\log_2(\min(\mathbf{m}, \mathbf{n})/(smlsiz + 1))) + 1)$ and $r = \min(\mathbf{m}, \mathbf{n})$, the code will execute correctly.

If $\mathbf{lwork} = -1$, a workspace query is assumed; the function only calculates the optimal size of the **work** array and the minimum size of the **iwork** array, and returns these values as the first entries of the **work** and **iwork** arrays, and no error message related to **lwork** is issued.

Suggested value: for optimal performance, **lwork** should generally be larger than the minimum required as set out above. Consider increasing **lwork** by at least $nb \times \min(\mathbf{m}, \mathbf{n})$, where nb is the optimal ***block size***.

Default: $64 \min(\mathbf{m}, \mathbf{n}) + \left(12r + 2r \times smlsiz + 8r \times nlvl + r \times \mathbf{nrhs\_p} + (smlsiz + 1)^2\right)$

Constraint:
$\mathbf{lwork} \geq 12r + 2r \times smlsiz + 8r \times nlvl + r \times \mathbf{nrhs\_p} + (smlsiz + 1)^2$ or $\mathbf{lwork} = -1$.

5.3 Input Parameters Omitted from the MATLAB Interface

lda, ldb, work, iwork

5.4 Output Parameters

1: **a(lda,$*$) – double array**

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

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

The contents of **a** are destroyed.

2: **b(ldb,∗) – double array**

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

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

b contains the n by r solution matrix X. If $m \geq n$ and $\mathbf{rank} = n$, the residual sum of squares for the solution in the ith column is given by the sum of squares of elements $n + 1, \ldots, m$ in that column.

3: **s(∗) – double array**

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

The singular values of A in decreasing order.

4: **rank – int32 scalar**

The effective rank of A, i.e., the number of singular values which are greater than $\mathbf{rcond} \times \mathbf{s}(1)$.

5: **info – int32 scalar**

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

6 Error Indicators and Warnings

Errors or warnings detected by the function:

info $= -i$

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

1: **m**, 2: **n**, 3: **nrhs_p**, 4: **a**, 5: **lda**, 6: **b**, 7: **ldb**, 8: **s**, 9: **rcond**, 10: **rank**, 11: **work**, 12: **lwork**, 13: **iwork**, 14: **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.

info > 0

The algorithm for computing the SVD failed to converge; if **info** $= i$, i off-diagonal elements of an intermediate bidiagonal form did not converge to zero.

7 Accuracy

See Section 4.5 of Anderson *et al.* 1999 for details.

8 Further Comments

The complex analogue of this function is f08kq.

9 Example

```
a = [-0.09, -1.56, -1.48, -1.09, 0.08, -1.59;
     0.14, 0.2, -0.43, 0.84, 0.55, -0.72;
     -0.46, 0.29, 0.89, 0.77, -1.13, 1.06;
     0.68, 1.09, -0.71, 2.11, 0.14, 1.24;
     1.29, 0.51, -0.96, -1.27, 1.74, 0.34];
```

```
b = [7.4;
     4.3;
     -8.1;
     1.8;
     8.7;
     0];
rcond = 0.01;
[aOut, bOut, s, rank, info] = f08kc(a, b, rcond)
```

```
aOut =
    2.8904    0.5234    0.4966    0.3657   -0.0268    0.5335
   -2.3501    1.9769    0.3575   -0.6148    0.4290   -0.1595
   -0.5887    1.2464   -1.5225    0.3678   -0.1209   -0.2363
   -0.6738   -0.3535   -1.4945    2.2693    0.3032   -0.3490
    0.2022    0.8123    0.5101    0.9654    0.0030   -0.6641
bOut =
    1.5938
   -0.1180
   -3.1501
    0.1554
    2.5529
   -1.6730
s =
    3.9997
    2.9962
    2.0001
    0.9988
    0.0025
rank =
           4
info =
           0
```