NAG Toolbox for MATLAB

f04km

1 Purpose

f04km solves a complex linear equality-constrained least-squares problem.

2 Syntax

```
[a, b, c, d, x, ifail] = f04km(a, b, c, d, 'm', m, 'n', n, 'p', p)
```

3 Description

f04km solves the complex linear equality-constrained least-squares (LSE) problem

$$\underset{x}{\text{minimize}} \|c - Ax\|_2 \qquad \text{subject to} \qquad Bx = d$$

where A is an m by n matrix, B is a p by n matrix, c is an m element vector and d is a p element vector. It is assumed that $p \leq n \leq m + p$, $\text{rank}(B) = p$ and $\text{rank}(E) = n$, where $E = \begin{pmatrix} A \\ B \end{pmatrix}$. These conditions ensure that the LSE problem has a unique solution, which is obtained using a generalized RQ factorization of the matrices B and A.

f04km is based on the LAPACK routine CGGLSE/ZGGLSE, see Anderson *et al.* 1999.

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

Anderson E, Bai Z and Dongarra J 1992 Generalized *QR* factorization and its applications *Linear Algebra Appl. (Volume 162–164)* 243–271

Eldèn L 1980 Perturbation theory for the least-squares problem with linear equality constraints *SIAM J. Numer. Anal.* **17** 338–350

5 Parameters

5.1 Compulsory Input Parameters

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

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

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

The p by n matrix B.

3: **c**$(*)$ – **complex array**

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

The right-hand side vector c for the least-squares part of the LSE problem.

4: **d**$(*)$ – **complex array**

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

The right-hand side vector d for the equality constraints.

5.2 Optional Input Parameters

1: **m** – **int32 scalar**

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

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

n, the number of columns of the matrices A and B.

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

3: **p** – **int32 scalar**

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

p, the number of rows of the matrix B.

Constraint: $0 \leq \mathbf{p} \leq \mathbf{n} \leq \mathbf{m} + \mathbf{p}$.

5.3 Input Parameters Omitted from the MATLAB Interface

lda, ldb, work, lwork

5.4 Output Parameters

1: **a**(**lda,***$*$*) – **complex 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 array is overwritten.

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

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

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

The array is overwritten.

3: **c**$(*)$ – **complex array**

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

The residual sum of squares for the solution vector x is given by the sum of squares of elements $\mathbf{c}(\mathbf{n} - \mathbf{p} + 1), \mathbf{c}(\mathbf{n} - \mathbf{p} + 2), \ldots, \mathbf{c}(\mathbf{m})$, provided $m + p > n$; the remaining elements are overwritten.

4: **d**$(*)$ **– complex array**

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

The array is overwritten.

5: **x**$(*)$ **– complex array**

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

The solution vector x of the LSE problem.

6: **ifail – int32 scalar**

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

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

On entry, $\mathbf{m} < 0$,
or $\mathbf{n} < 0$,
or $\mathbf{p} < 0$,
or $\mathbf{p} > \mathbf{n}$,
or $\mathbf{p} < \mathbf{n} - \mathbf{m}$,
or $\mathbf{lda} < \max(1, \mathbf{m})$,
or $\mathbf{ldb} < \max(1, \mathbf{p})$,
or $\mathbf{lwork} < \max(1, \mathbf{m} + \mathbf{n} + \mathbf{p})$ and $\mathbf{lwork} \neq -1$.

7 Accuracy

For an error analysis, see Anderson *et al.* 1992 and Eldèn 1980.

8 Further Comments

When $m \geq n = p$, the total number of real floating-point operations is approximately $\frac{8}{3}n^2(6m + n)$; if $p \ll n$, the number reduces to approximately $\frac{8}{3}n^2(3m - n)$.

9 Example

```
a = [complex(0.96, -0.8100000000000001), complex(-0.03, +0.96), complex(-
0.91, +2.06), complex(-0.05, +0.41);
      complex(-0.98, +1.98), complex(-1.2, +0.19), complex(-0.66, +0.42),
...
     complex(-0.8100000000000001, +0.5600000000000001);
       complex(0.62, -0.46), complex(1.01, +0.02), complex(0.63, -0.17),
complex(-1.11, +0.6);
      complex(0.37, +0.38), complex(0.19, -0.54), complex(-0.98, -0.36),
complex(0.22, -0.2);
       complex(0.83, +0.51), complex(0.2, +0.01), complex(-0.17, -0.46),
complex(1.47, +1.59);
                complex(1.08,   -0.28),   complex(0.2,   -0.12),   complex(-
0.07000000000000001, +1.23), complex(0.26, +0.26)];
b = [complex(1, +0), complex(0, +0), complex(-1, +0), complex(0, +0);
     complex(0, +0), complex(1, +0), complex(0, +0), complex(-1, +0)];
c = [complex(-1.54, +0.76);
     complex(0.12, -1.92);
     complex(-9.08, -4.31);
     complex(7.49, +3.65);
```

```
     complex(-5.63, -2.12);
     complex(2.37, +8.029999999999999)];
d = [complex(0, +0);
     complex(0, +0)];
[aOut, bOut, cOut, dOut, x, ifail] = f04km(a, b, c, d)
```

```
aOut =
   -2.7118              -1.4390 - 1.0315i   -0.1054 + 1.3176i   -0.3924 -
0.1950i
   -0.2024 + 0.6829i   -1.8583              -0.9453 + 0.1928i    1.4355 +
0.2631i
    0.2443 - 0.2408i   -0.4279 + 0.1339i    2.9079              -0.2395 +
0.1886i
  -0.1408 + 0.0504i    0.2817 - 0.1483i   -0.4394 - 0.1145i   -1.5759
    0.1577 - 0.0379i    0.4242 + 0.4187i   -0.0867 - 0.7048i   -0.0589 +
0.0550i
    0.3069 + 0.1458i    0.0940 - 0.3436i   -0.1277 + 0.1658i    0.1632 -
0.4069i
bOut =
   -0.4142         0    1.4142         0
         0   -0.4142         0    1.4142
cOut =
   1.5259 - 2.7609i
  -1.0721 + 5.2614i
  -1.7460 + 0.7977i
   4.1351 - 1.8334i
   1.0997 - 9.8334i
  -1.0307 + 7.1861i
dOut =
     0
     0
x =
   1.0789 - 1.9523i
  -0.7581 + 3.7203i
   1.0789 - 1.9523i
  -0.7581 + 3.7203i
ifail =
           0
```