NAG Library Routine Document

F08ZNF (ZGGLSE)

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

F08ZNF (ZGGLSE) solves a complex linear equality-constrained least squares problem.

2 Specification

```
SUBROUTINE F08ZNF (M, N, P, A, LDA, B, LDB, C, D, X, WORK, LWORK, INFO)
INTEGER                M, N, P, LDA, LDB, LWORK, INFO
COMPLEX (KIND=nag_wp) A(LDA,*), B(LDB,*), C(M), D(P), X(N),          &
                       WORK(max(1,LWORK))
```

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

3 Description

F08ZNF (ZGGLSE) solves the complex linear equality-constrained least squares (LSE) problem

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

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 $\text{rank}(\text{E}) = n$, 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.

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

1: M – INTEGER *Input*

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

Constraint: $\text{M} \geq 0$.

2: N – INTEGER *Input*

On entry: n, the number of columns of the matrices A and B.

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

3: P – INTEGER *Input*

On entry: p, the number of rows of the matrix B.

Constraint: $0 \leq P \leq N \leq M + P$.

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: A is overwritten.

5: LDA – INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F08ZNF (ZGGLSE) is called.

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

6: B(LDB, $*$) – COMPLEX (KIND=nag_wp) array *Input/Output*

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

On entry: the p by n matrix B.

On exit: B is overwritten.

7: LDB – INTEGER *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F08ZNF (ZGGLSE) is called.

Constraint: $LDB \geq \max(1, P)$.

8: C(M) – COMPLEX (KIND=nag_wp) array *Input/Output*

On entry: the right-hand side vector c for the least squares part of the LSE problem.

On exit: the residual sum of squares for the solution vector x is given by the sum of squares of elements $C(N - P + 1), C(N - P + 2), \ldots, C(M)$; the remaining elements are overwritten.

9: D(P) – COMPLEX (KIND=nag_wp) array *Input/Output*

On entry: the right-hand side vector d for the equality constraints.

On exit: D is overwritten.

10: X(N) – COMPLEX (KIND=nag_wp) array *Output*

On exit: the solution vector x of the LSE problem.

11: WORK(max(1, 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 F08ZNF (ZGGLSE) 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, $\mathrm{LWORK} \geq \mathrm{P} + \min(\mathrm{M}, \mathrm{N}) + \max(\mathrm{M}, \mathrm{N}) \times nb$, where nb is the optimal ***block size***.

Constraint: $\mathrm{LWORK} \geq \max(1, \mathrm{M} + \mathrm{N} + \mathrm{P})$ or $\mathrm{LWORK} = -1$.

13: INFO – INTEGER *Output*

On exit: $\mathrm{INFO} = 0$ unless the routine detects an error (see Section 6).

6 Error Indicators and Warnings

$\mathrm{INFO} < 0$

If $\mathrm{INFO} = -i$, argument i had an illegal value. An explanatory message is output, and execution of the program is terminated.

$\mathrm{INFO} = 1$

The upper triangular factor R associated with B in the generalized RQ factorization of the pair (B, A) is singular, so that $\mathrm{rank}(B) < p$; the least squares solution could not be computed.

$\mathrm{INFO} = 2$

The $(N - P)$ by $(N - P)$ part of the upper trapezoidal factor T associated with A in the generalized RQ factorization of the pair (B, A) is singular, so that the rank of the matrix (E) comprising the rows of A and B is less than n; the least squares solutions could not be computed.

7 Accuracy

For an error analysis, see Anderson *et al.* (1992) and Eldèn (1980). See also Section 4.6 of Anderson *et al.* (1999).

8 Parallelism and Performance

F08ZNF (ZGGLSE) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

F08ZNF (ZGGLSE) 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

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 $\frac{8}{3}n^2(3m - n)$.

10 Example

This example solves the least squares problem

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

where

$$c = \begin{pmatrix} -2.54 + 0.09i \\ 1.65 - 2.26i \\ -2.11 - 3.96i \\ 1.82 + 3.30i \\ -6.41 + 3.77i \\ 2.07 + 0.66i \end{pmatrix},$$

and

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

$$B = \begin{pmatrix} 1.0 + 0.0i & 0 & -1.0 + 0.0i & 0 \\ 0 & 1.0 + 0.0i & 0 & -1.0 + 0.0i \end{pmatrix}$$

and

$$d = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

The constraints $Bx = d$ correspond to $x_1 = x_3$ and $x_2 = x_4$.

Note that the block size (NB) of 64 assumed in this example is not realistic for such a small problem, but should be suitable for large problems.

10.1 Program Text

```
    Program f08znfe

!     F08ZNF Example Program Text

!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: dznrm2, nag_wp, zgglse
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nb = 64, nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: rnorm
      Integer                          :: i, info, lda, ldb, lwork, m, n, p
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: a(:,:), b(:,:), c(:), d(:),        &
                                          work(:), x(:)
!     .. Executable Statements ..
      Write (nout,*) 'F08ZNF Example Program Results'
      Write (nout,*)
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) m, n, p
      lda = m
      ldb = p
      lwork = p + n + nb*(m+n)
      Allocate (a(lda,n),b(ldb,n),c(m),d(p),work(lwork),x(n))

!     Read A, B, C and D from data file

      Read (nin,*)(a(i,1:n),i=1,m)
      Read (nin,*)(b(i,1:n),i=1,p)
```

```
      Read (nin,*) c(1:m)
      Read (nin,*) d(1:p)

!     Solve the equality-constrained least-squares problem

!     minimize ||c - A*x|| (in the 2-norm) subject to B*x = D

!     The NAG name equivalent of zgglse is f08znf
      Call zgglse(m,n,p,a,lda,b,ldb,c,d,x,work,lwork,info)

!     Print least-squares solution

      Write (nout,*) 'Constrained least-squares solution'
      Write (nout,99999) x(1:n)

!     Compute the square root of the residual sum of squares

!     The NAG name equivalent of dznrm2 is f06jjf
      rnorm = dznrm2(m-n+p,c(n-p+1),1)
      Write (nout,*)
      Write (nout,*) 'Square root of the residual sum of squares'
      Write (nout,99998) rnorm

99999 Format (4(' (',F7.4,',',F7.4,')':))
99998 Format (1X,1P,E10.2)
    End Program f08znfe
```

10.2 Program Data

```
F08ZNF Example Program Data

  6             4             2                         :Values of M, N and P

( 0.96,-0.81) (-0.03, 0.96) (-0.91, 2.06) (-0.05, 0.41)
(-0.98, 1.98) (-1.20, 0.19) (-0.66, 0.42) (-0.81, 0.56)
( 0.62,-0.46) ( 1.01, 0.02) ( 0.63,-0.17) (-1.11, 0.60)
( 0.37, 0.38) ( 0.19,-0.54) (-0.98,-0.36) ( 0.22,-0.20)
( 0.83, 0.51) ( 0.20, 0.01) (-0.17,-0.46) ( 1.47, 1.59)
( 1.08,-0.28) ( 0.20,-0.12) (-0.07, 1.23) ( 0.26, 0.26) :End of matrix A

( 1.00, 0.00) ( 0.00, 0.00) (-1.00, 0.00) ( 0.00, 0.00)
( 0.00, 0.00) ( 1.00, 0.00) ( 0.00, 0.00) (-1.00, 0.00) :End of matrix B

(-2.54, 0.09)
( 1.65,-2.26)
(-2.11,-3.96)
( 1.82, 3.30)
(-6.41, 3.77)
( 2.07, 0.66)                                           :End of vector c

( 0.00, 0.00)
( 0.00, 0.00)                                           :End of vector d
```

10.3 Program Results

```
 F08ZNF Example Program Results

 Constrained least-squares solution
 ( 1.0874,-1.9621) (-0.7409, 3.7297) ( 1.0874,-1.9621) (-0.7409, 3.7297)

 Square root of the residual sum of squares
   1.59E-01
```