NAG Library Routine Document

F08AVF (ZGELQF)

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

F08AVF (ZGELQF) computes the LQ factorization of a complex m by n matrix.

2 Specification

```
SUBROUTINE F08AVF (M, N, A, LDA, TAU, WORK, LWORK, INFO)
INTEGER                 M, N, LDA, LWORK, INFO
COMPLEX (KIND=nag_wp) A(LDA,*), TAU(*), WORK(max(1,LWORK))
```

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

3 Description

F08AVF (ZGELQF) forms the LQ factorization of an arbitrary rectangular complex m by n matrix. No pivoting is performed.

If $m \leq n$, the factorization is given by:

$$A = \begin{pmatrix} L & 0 \end{pmatrix} Q$$

where L is an m by m lower triangular matrix (with real diagonal elements) and Q is an n by n unitary matrix. It is sometimes more convenient to write the factorization as

$$A = \begin{pmatrix} L & 0 \end{pmatrix} \begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix}$$

which reduces to

$$A = LQ_1,$$

where Q_1 consists of the first m rows of Q, and Q_2 the remaining $n - m$ rows.

If $m > n$, L is trapezoidal, and the factorization can be written

$$A = \begin{pmatrix} L_1 \\ L_2 \end{pmatrix} Q$$

where L_1 is lower triangular and L_2 is rectangular.

The LQ factorization of A is essentially the same as the QR factorization of A^{H}, since

$$A = \begin{pmatrix} L & 0 \end{pmatrix} Q \Leftrightarrow A^{\mathrm{H}} = Q^{\mathrm{H}} \begin{pmatrix} L^{\mathrm{H}} \\ 0 \end{pmatrix}.$$

The matrix Q is not formed explicitly but is represented as a product of $\min(m, n)$ elementary reflectors (see the F08 Chapter Introduction for details). Routines are provided to work with Q in this representation (see Section 9).

Note also that for any $k < m$, the information returned in the first k rows of the array A represents an LQ factorization of the first k rows of the original matrix A.

4 References

None.

5 Arguments

1: M – INTEGER *Input*

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

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

2: N – INTEGER *Input*

On entry: n, the number of columns of the matrix A.

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

3: A(LDA, $*$) – COMPLEX (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 $m \leq n$, the elements above the diagonal are overwritten by details of the unitary matrix Q and the lower triangle is overwritten by the corresponding elements of the m by m lower triangular matrix L.

If $m > n$, the strictly upper triangular part is overwritten by details of the unitary matrix Q and the remaining elements are overwritten by the corresponding elements of the m by n lower trapezoidal matrix L.

The diagonal elements of L are real.

4: LDA – INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F08AVF (ZGELQF) is called.

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

5: TAU($*$) – COMPLEX (KIND=nag_wp) array *Output*

Note: the dimension of the array TAU must be at least $\max(1, \min(\mathrm{M}, \mathrm{N}))$.

On exit: further details of the unitary matrix Q.

6: WORK(max(1, LWORK)) – COMPLEX (KIND=nag_wp) array *Workspace*

On exit: if $\mathrm{INFO} = 0$, the real part of $\mathrm{WORK}(1)$ contains the minimum value of LWORK required for optimal performance.

7: LWORK – INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08AVF (ZGELQF) is called.

If $\mathrm{LWORK} = -1$, 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, $\mathrm{LWORK} \geq \mathrm{M} \times nb$, where nb is the optimal ***block size***.

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

8: INFO – INTEGER *Output*

On exit: $\mathrm{INFO} = 0$ 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.

7 Accuracy

The computed factorization is the exact factorization of a nearby matrix $(A + E)$, where

$$\|E\|_2 = O(\epsilon)\|A\|_2,$$

and ϵ is the ***machine precision***.

8 Parallelism and Performance

F08AVF (ZGELQF) 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 real floating-point operations is approximately $\frac{8}{3}m^2(3n - m)$ if $m \leq n$ or $\frac{8}{3}n^2(3m - n)$ if $m > n$.

To form the unitary matrix Q F08AVF (ZGELQF) may be followed by a call to F08AWF (ZUNGLQ):

```
CALL ZUNGLQ(N,N,MIN(M,N),A,LDA,TAU,WORK,LWORK,INFO)
```

but note that the first dimension of the array A, specified by the argument LDA, must be at least N, which may be larger than was required by F08AVF (ZGELQF).

When $m \leq n$, it is often only the first m rows of Q that are required, and they may be formed by the call:

```
CALL ZUNGLQ(M,N,M,A,LDA,TAU,WORK,LWORK,INFO)
```

To apply Q to an arbitrary complex rectangular matrix C, F08AVF (ZGELQF) may be followed by a call to F08AXF (ZUNMLQ). For example,

```
CALL ZUNMLQ('Left','Conjugate Transpose',M,P,MIN(M,N),A,LDA,TAU, &
            C,LDC,WORK,LWORK,INFO)
```

forms the matrix product $C = Q^{\mathrm{H}}C$, where C is m by p.

The real analogue of this routine is F08AHF (DGELQF).

10 Example

This example finds the minimum norm solutions of the under-determined systems of linear equations

$$Ax_1 = b_1 \quad \text{and} \quad Ax_2 = b_2$$

where b_1 and b_2 are the columns of the matrix B,

$$A = \begin{pmatrix} 0.28 - 0.36i & 0.50 - 0.86i & -0.77 - 0.48i & 1.58 + 0.66i \\ -0.50 - 1.10i & -1.21 + 0.76i & -0.32 - 0.24i & -0.27 - 1.15i \\ 0.36 - 0.51i & -0.07 + 1.33i & -0.75 + 0.47i & -0.08 + 1.01i \end{pmatrix}$$

and

$$B = \begin{pmatrix} -1.35 + 0.19i & 4.83 - 2.67i \\ 9.41 - 3.56i & -7.28 + 3.34i \\ -7.57 + 6.93i & 0.62 + 4.53i \end{pmatrix}.$$

10.1 Program Text

```
    Program f08avfe

!     F08AVF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: nag_wp, x04dbf, zgelqf, ztrsm, zunmlq
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Complex (Kind=nag_wp), Parameter :: one = (1.0_nag_wp,0.0_nag_wp)
      Complex (Kind=nag_wp), Parameter :: zero = (0.0_nag_wp,0.0_nag_wp)
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: i, ifail, info, lda, ldb, lwork, m,  &
                                          n, nrhs
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: a(:,:), b(:,:), tau(:), work(:)
      Character (1)                    :: clabs(1), rlabs(1)
!     .. Executable Statements ..
      Write (nout,*) 'F08AVF Example Program Results'
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) m, n, nrhs
      lda = m
      ldb = n
      lwork = 64*n
      Allocate (a(lda,n),b(ldb,nrhs),tau(n),work(lwork))

!     Read A and B from data file

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

!     Compute the LQ factorization of A
!     The NAG name equivalent of zgelqf is f08avf
      Call zgelqf(m,n,a,lda,tau,work,lwork,info)

!     Solve L*Y = B, storing the result in B
!     The NAG name equivalent of ztrsm is f06zjf
      Call ztrsm('Left','Lower','No transpose','Non-Unit',m,nrhs,one,a,lda,b,  &
        ldb)

!     Set rows (M+1) to N of B to zero

      If (m<n) Then
        b(m+1:n,1:nrhs) = zero
      End If

!     Compute minimum-norm solution X = (Q**H)*B in B
!     The NAG name equivalent of zunmlq is f08axf
      Call zunmlq('Left','Conjugate transpose',n,nrhs,m,a,lda,tau,b,ldb,work,  &
        lwork,info)

!     Print minimum-norm solution(s)

      Write (nout,*)
      Flush (nout)

!     ifail: behaviour on error exit
!             =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
```

```
      ifail = 0
      Call x04dbf('General',' ',n,nrhs,b,ldb,'Bracketed','F7.4',              &
        'Minimum-norm solution(s)','Integer',rlabs,'Integer',clabs,80,0,ifail)

    End Program f08avfe
```

10.2 Program Data

```
F08AVF Example Program Data
  3  4  2                                          :Values of M, N and NRHS
 ( 0.28,-0.36) ( 0.50,-0.86) (-0.77,-0.48) ( 1.58, 0.66)
 (-0.50,-1.10) (-1.21, 0.76) (-0.32,-0.24) (-0.27,-1.15)
 ( 0.36,-0.51) (-0.07, 1.33) (-0.75, 0.47) (-0.08, 1.01)   :End of matrix A
 (-1.35, 0.19) ( 4.83,-2.67)
 ( 9.41,-3.56) (-7.28, 3.34)
 (-7.57, 6.93) ( 0.62, 4.53)                               :End of matrix B
```

10.3 Program Results

```
 F08AVF Example Program Results

 Minimum-norm solution(s)
                     1                  2
 1  (-2.8501, 6.4683) (-1.1682,-1.8886)
 2  ( 1.6264,-0.7799) ( 2.8377, 0.7654)
 3  ( 6.9290, 4.6481) (-1.7610,-0.7041)
 4  ( 1.4048, 3.2400) ( 1.0518,-1.6365)
```