NAG Library Function Document

nag_zgelqf (f08avc)

1 Purpose

nag_zgelqf (f08avc) computes the LQ factorization of a complex m by n matrix.

2 Specification

```
#include <nag.h>
#include <nagf08.h>
void nag_zgelqf (Nag_OrderType order, Integer m, Integer n, Complex a[],
     Integer pda, Complex tau[], NagError *fail)
```

3 Description

nag_zgelqf (f08avc) 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). Functions 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: **order** – Nag_OrderType *Input*

On entry: the **order** argument specifies the two-dimensional storage scheme being used, i.e., row-major ordering or column-major ordering. C language defined storage is specified by **order** = Nag_RowMajor. See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

Constraint: **order** = Nag_RowMajor or Nag_ColMajor.

2: **m** – Integer *Input*

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

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

3: **n** – Integer *Input*

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

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

4: $\mathbf{a}[dim]$ – Complex *Input/Output*

Note: the dimension, *dim*, of the array **a** must be at least

$\max(1, \mathbf{pda} \times \mathbf{n})$ when **order** = Nag_ColMajor;
$\max(1, \mathbf{m} \times \mathbf{pda})$ when **order** = Nag_RowMajor.

The (i, j)th element of the matrix A is stored in

$\mathbf{a}[(j-1) \times \mathbf{pda} + i - 1]$ when **order** = Nag_ColMajor;
$\mathbf{a}[(i-1) \times \mathbf{pda} + j - 1]$ when **order** = Nag_RowMajor.

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.

5: **pda** – Integer *Input*

On entry: the stride separating row or column elements (depending on the value of **order**) in the array **a**.

Constraints:

if **order** = Nag_ColMajor, $\mathbf{pda} \geq \max(1, \mathbf{m})$;
if **order** = Nag_RowMajor, $\mathbf{pda} \geq \max(1, \mathbf{n})$.

6: $\mathbf{tau}[dim]$ – Complex *Output*

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

On exit: further details of the unitary matrix Q.

7: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

NE_BAD_PARAM

On entry, argument $\langle\textit{value}\rangle$ had an illegal value.

NE_INT

On entry, $\mathbf{m} = \langle\textit{value}\rangle$.
Constraint: $\mathbf{m} \geq 0$.

On entry, $\mathbf{n} = \langle\textit{value}\rangle$.
Constraint: $\mathbf{n} \geq 0$.

On entry, $\mathbf{pda} = \langle\textit{value}\rangle$.
Constraint: $\mathbf{pda} > 0$.

NE_INT_2

On entry, $\mathbf{pda} = \langle\textit{value}\rangle$ and $\mathbf{m} = \langle\textit{value}\rangle$.
Constraint: $\mathbf{pda} \geq \max(1, \mathbf{m})$.

On entry, $\mathbf{pda} = \langle\textit{value}\rangle$ and $\mathbf{n} = \langle\textit{value}\rangle$.
Constraint: $\mathbf{pda} \geq \max(1, \mathbf{n})$.

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 3.6.6 in the Essential Introduction for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in the Essential Introduction for further information.

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

nag_zgelqf (f08avc) is not threaded by NAG in any implementation.

nag_zgelqf (f08avc) 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 function. 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 nag_zgelqf (f08avc) may be followed by a call to nag_zunglq (f08awc):

```
nag_zunglq(order,n,n,MIN(m,n),&a,pda,tau,&fail)
```

but note that the first dimension of the array **a**, specified by the argument **pda**, must be at least **n**, which may be larger than was required by nag_zgelqf (f08avc).

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:

```
nag_zunglq(order,m,n,m,&a,pda,tau,&fail)
```

To apply Q to an arbitrary complex rectangular matrix C, nag_zgelqf (f08avc) may be followed by a call to nag_zunmlq (f08axc). For example,

```
nag_zunmlq(order,Nag_LeftSide,Nag_ConjTrans,m,p,MIN(m,n),&a,pda,
  tau,&c,pdc,&fail)
```

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

The real analogue of this function is nag_dgelqf (f08ahc).

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

```
/* nag_zgelqf (f08avc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <naga02.h>
#include <nagf07.h>
#include <nagf08.h>
#include <nagx04.h>

int main(void)
{
  /* Scalars */
  Integer       i, j, m, n, nrhs, pda, pdb, tau_len;
  Integer       exit_status = 0;
  NagError      fail;
  Nag_OrderType order;
  /* Arrays */
  Complex       *a = 0, *b = 0, *tau = 0;

#ifdef NAG_COLUMN_MAJOR
#define A(I, J) a[(J - 1) * pda + I - 1]
#define B(I, J) b[(J - 1) * pdb + I - 1]
  order = Nag_ColMajor;
#else
#define A(I, J) a[(I - 1) * pda + J - 1]
#define B(I, J) b[(I - 1) * pdb + J - 1]
  order = Nag_RowMajor;
#endif

  INIT_FAIL(fail);

  printf("nag_zgelqf (f08avc) Example Program Results\n\n");

  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif
#ifdef _WIN32
  scanf_s("%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%*[^\n] ", &m, &n, &nrhs);
#else
  scanf("%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%*[^\n] ", &m, &n, &nrhs);
#endif

#ifdef NAG_COLUMN_MAJOR
  pda = m;
  pdb = n;
#else
  pda = n;
  pdb = nrhs;
#endif

  tau_len = MIN(m, n);

  /* Allocate memory */
  if (!(a = NAG_ALLOC(m * n, Complex)) ||
      !(b = NAG_ALLOC(n * nrhs, Complex)) ||
      !(tau = NAG_ALLOC(tau_len, Complex)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* Read A and B from data file */
  for (i = 1; i <= m; ++i)
    {
      for (j = 1; j <= n; ++j)
#ifdef _WIN32
        scanf_s(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
#else
        scanf(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
#endif
    }
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif
  for (i = 1; i <= m; ++i)
    {
      for (j = 1; j <= nrhs; ++j)
#ifdef _WIN32
        scanf_s(" ( %lf , %lf )", &B(i, j).re, &B(i, j).im);
#else
        scanf(" ( %lf , %lf )", &B(i, j).re, &B(i, j).im);
```

```
#endif
    }
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

  /* Compute the LQ factorization of A */
  /* nag_zgelqf (f08avc).
   * LQ factorization of complex general rectangular matrix
   */
  nag_zgelqf(order, m, n, a, pda, tau, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_zgelqf (f08avc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  /* Solve L*Y = B, storing the result in B */
  /* nag_ztrtrs (f07tsc).
   * Solution of complex triangular system of linear
   * equations, multiple right-hand sides
   */
  nag_ztrtrs(order, Nag_Lower, Nag_NoTrans, Nag_NonUnitDiag, m,
             nrhs, a, pda, b, pdb, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_ztrtrs (f07tsc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  /* Set rows (M+1) to N of B to zero */
  if (m < n)
    {
      for (i = m + 1; i <= n; ++i)
        {
          for (j = 1; j <= nrhs; ++j)
            {
              B(i, j).re = 0.0;
              B(i, j).im = 0.0;
            }
        }
    }

  /* Compute minimum-norm solution X = (Q**H)*B in B */
  /* nag_zunmlq (f08axc).
   * Apply unitary transformation determined by nag_zgelqf (f08avc)
   */
  nag_zunmlq(order, Nag_LeftSide, Nag_ConjTrans, n, nrhs, m, a, pda,
             tau, b, pdb, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_zunmlq (f08axc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

  /* Print minimum-norm solution(s) */
  /* nag_gen_complx_mat_print_comp (x04dbc).
   * Print complex general matrix (comprehensive)
   */
  fflush(stdout);
  nag_gen_complx_mat_print_comp(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n,
                                nrhs, b, pdb, Nag_BracketForm, "%7.4f",
                                "Minimum-norm solution(s)",
                                Nag_IntegerLabels, 0, Nag_IntegerLabels, 0, 80,
                                0, 0, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf(
             "Error from nag_gen_complx_mat_print_comp (x04dbc).\n%s\n",
             fail.message);
      exit_status = 1;
      goto END;
    }
 END:
  NAG_FREE(a);
  NAG_FREE(b);
  NAG_FREE(tau);
  return exit_status;
}
```

10.2 Program Data

```
nag_zgelqf (f08avc) 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

```
nag_zgelqf (f08avc) 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)
```