

# NAG Library Function Document

## nag\_zunglq (f08awc)

### 1 Purpose

nag\_zunglq (f08awc) generates all or part of the complex unitary matrix  $Q$  from an  $LQ$  factorization computed by nag\_zgelqf (f08avc).

### 2 Specification

```
#include <nag.h>
#include <nagf08.h>

void nag_zunglq (Nag_OrderType order, Integer m, Integer n, Integer k,
                Complex a[], Integer pda, const Complex tau[], NagError *fail)
```

### 3 Description

nag\_zunglq (f08awc) is intended to be used after a call to nag\_zgelqf (f08avc), which performs an  $LQ$  factorization of a complex matrix  $A$ . The unitary matrix  $Q$  is represented as a product of elementary reflectors.

This function may be used to generate  $Q$  explicitly as a square matrix, or to form only its leading rows.

Usually  $Q$  is determined from the  $LQ$  factorization of a  $p$  by  $n$  matrix  $A$  with  $p \leq n$ . The whole of  $Q$  may be computed by:

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

(note that the array  $\mathbf{a}$  must have at least  $n$  rows) or its leading  $p$  rows by:

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

The rows of  $Q$  returned by the last call form an orthonormal basis for the space spanned by the rows of  $A$ ; thus nag\_zgelqf (f08avc) followed by nag\_zunglq (f08awc) can be used to orthogonalize the rows of  $A$ .

The information returned by the  $LQ$  factorization functions also yields the  $LQ$  factorization of the leading  $k$  rows of  $A$ , where  $k < p$ . The unitary matrix arising from this factorization can be computed by:

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

or its leading  $k$  rows by:

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

### 4 References

Golub G H and Van Loan C F (1996) *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

### 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 2.3.1.3 in How to Use the NAG Library and its Documentation 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  $Q$ .  
*Constraint:*  $m \geq 0$ .
- 3: **n** – Integer *Input*  
*On entry:*  $n$ , the number of columns of the matrix  $Q$ .  
*Constraint:*  $n \geq m$ .
- 4: **k** – Integer *Input*  
*On entry:*  $k$ , the number of elementary reflectors whose product defines the matrix  $Q$ .  
*Constraint:*  $m \geq k \geq 0$ .
- 5: **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.  
*On entry:* details of the vectors which define the elementary reflectors, as returned by nag\_zgelqf (f08avc).  
*On exit:* the  $m$  by  $n$  matrix  $Q$ .  
If **order** = Nag\_ColMajor, the  $(i, j)$ th element of the matrix is stored in  $\mathbf{a}[(j - 1) \times \mathbf{pda} + i - 1]$ .  
If **order** = Nag\_RowMajor, the  $(i, j)$ th element of the matrix is stored in  $\mathbf{a}[(i - 1) \times \mathbf{pda} + j - 1]$ .
- 6: **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,  $\mathbf{pda} \geq \max(1, \mathbf{n})$ .
- 7: **tau**[*dim*] – const Complex *Input*  
**Note:** the dimension, *dim*, of the array **tau** must be at least  $\max(1, \mathbf{k})$ .  
*On entry:* further details of the elementary reflectors, as returned by nag\_zgelqf (f08avc).
- 8: **fail** – NagError \* *Input/Output*  
The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

## 6 Error Indicators and Warnings

### NE\_ALLOC\_FAIL

Dynamic memory allocation failed.

See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

### NE\_BAD\_PARAM

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

**NE\_INT**

On entry, **m** =  $\langle value \rangle$ .  
 Constraint: **m**  $\geq$  0.

On entry, **pda** =  $\langle value \rangle$ .  
 Constraint: **pda**  $>$  0.

**NE\_INT\_2**

On entry, **m** =  $\langle value \rangle$  and **k** =  $\langle value \rangle$ .  
 Constraint: **m**  $\geq$  **k**  $\geq$  0.

On entry, **n** =  $\langle value \rangle$  and **m** =  $\langle value \rangle$ .  
 Constraint: **n**  $\geq$  **m**.

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

On entry, **pda** =  $\langle value \rangle$  and **n** =  $\langle value \rangle$ .  
 Constraint: **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 2.7.6 in How to Use the NAG Library and its Documentation for further information.

**NE\_NO\_LICENCE**

Your licence key may have expired or may not have been installed correctly.  
 See Section 2.7.5 in How to Use the NAG Library and its Documentation for further information.

**7 Accuracy**

The computed matrix  $Q$  differs from an exactly unitary matrix by a matrix  $E$  such that

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

where  $\epsilon$  is the *machine precision*.

**8 Parallelism and Performance**

nag\_zunglq (f08awc) 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  $16mnk - 8(m+n)k^2 + \frac{16}{3}k^3$ ; when  $m = k$ , the number is approximately  $\frac{8}{3}m^2(3n - m)$ .

The real analogue of this function is nag\_dorglq (f08ajc).

## 10 Example

This example forms the leading 4 rows of the unitary matrix  $Q$  from the  $LQ$  factorization of the matrix  $A$ , where

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

The rows of  $Q$  form an orthonormal basis for the space spanned by the rows of  $A$ .

### 10.1 Program Text

```

/* nag_zunglq (f08awc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

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

int main(void)
{
    /* Scalars */
    Integer i, j, m, n, pda, tau_len;
    Integer exit_status = 0;
    NagError fail;
    Nag_OrderType order;
    /* Arrays */
    char *title = 0;
    Complex *a = 0, *tau = 0;
#ifdef NAG_LOAD_FP
    /* The following line is needed to force the Microsoft linker
       to load floating point support */
    float force_loading_of_ms_float_support = 0;
#endif /* NAG_LOAD_FP */

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

    INIT_FAIL(fail);

    printf("nag_zunglq (f08awc) 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 "%*[\n] ", &m, &n);
#else
    scanf("%" NAG_IFMT "%" NAG_IFMT "%*[\n] ", &m, &n);
#endif
#ifdef NAG_COLUMN_MAJOR
    pda = m;

```

```

#else
    pda = n;
#endif
    tau_len = m;

    /* Allocate memory */
    if (!(title = NAG_ALLOC(31, char)) ||
        !(a = NAG_ALLOC(m * n, Complex)) ||
        !(tau = NAG_ALLOC(tau_len, Complex)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    /* Read A 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

    /* 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;
    }

    /* Form the leading M rows of Q explicitly */
    /* nag_zunglq (f08awc).
     * Form all or part of unitary Q from LQ factorization
     * determined by nag_zgelqf (f08avc)
     */
    nag_zunglq(order, m, n, m, a, pda, tau, &fail);
    if (fail.code != NE_NOERROR) {
        printf("Error from nag_zunglq (f08awc).\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }

    /* Print the leading M rows of Q only */
#ifdef _WIN32
    sprintf_s(title, 31, "The leading %2" NAG_IFMT " rows of Q\n", m);
#else
    sprintf(title, "The leading %2" NAG_IFMT " rows of Q\n", m);
#endif
    /* 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, m,
                                  n, a, pda, Nag_BracketForm, "%7.4f", title,
                                  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(title);
  NAG_FREE(a);
  NAG_FREE(tau);

  return exit_status;
}

```

## 10.2 Program Data

```

nag_zunglq (f08awc) Example Program Data
  3 4                                     :Values of M and N
( 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

```

## 10.3 Program Results

nag\_zunglq (f08awc) Example Program Results

The leading 3 rows of Q

	1	2	3	4
1	(-0.1258, 0.1618)	(-0.2247, 0.3864)	( 0.3460, 0.2157)	(-0.7099,-0.2966)
2	(-0.1163,-0.6380)	(-0.3240, 0.4272)	(-0.1995,-0.5009)	(-0.0323,-0.0162)
3	(-0.4607, 0.1090)	( 0.2171,-0.4062)	( 0.2733,-0.6106)	(-0.0994,-0.3261)

---