

# NAG Library Function Document

## nag\_zungql (f08ctc)

### 1 Purpose

nag\_zungql (f08ctc) generates all or part of the complex  $m$  by  $m$  unitary matrix  $Q$  from a  $QL$  factorization computed by nag\_zgeqlf (f08csc).

### 2 Specification

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

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

### 3 Description

nag\_zungql (f08ctc) is intended to be used after a call to nag\_zgeqlf (f08csc), which performs a  $QL$  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 trailing columns.

Usually  $Q$  is determined from the  $QL$  factorization of an  $m$  by  $p$  matrix  $A$  with  $m \geq p$ . The whole of  $Q$  may be computed by:

```
nag_zungql(order,m,m,p,a,pda,tau,&fail)
```

(note that the array **a** must have at least  $m$  columns) or its trailing  $p$  columns by:

```
nag_zungql(order,m,p,p,a,pda,tau,&fail)
```

The columns of  $Q$  returned by the last call form an orthonormal basis for the space spanned by the columns of  $A$ ; thus nag\_zgeqlf (f08csc) followed by nag\_zungql (f08ctc) can be used to orthogonalize the columns of  $A$ .

The information returned by nag\_zgeqlf (f08csc) also yields the  $QL$  factorization of the trailing  $k$  columns of  $A$ , where  $k < p$ . The unitary matrix arising from this factorization can be computed by:

```
nag_zungql(order,m,m,k,a,pda,tau,&fail)
```

or its trailing  $k$  columns by:

```
nag_zungql(order,m,k,k,a,pda,tau,&fail)
```

### 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 <http://www.netlib.org/lapack/lug>

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:*  $m \geq n \geq 0$ .
  
- 4:    **k** – Integer *Input*  
*On entry:*  $k$ , the number of elementary reflectors whose product defines the matrix  $Q$ .  
*Constraint:*  $n \geq k \geq 0$ .
  
- 5:    **a**[ $dim$ ] – Complex *Input/Output*  
**Note:** the dimension,  $dim$ , of the array **a** must be at least  
 $\max(1, pda \times n)$  when **order** = Nag\_ColMajor;  
 $\max(1, m \times pda)$  when **order** = Nag\_RowMajor.  
*On entry:* details of the vectors which define the elementary reflectors, as returned by nag\_zgeqlf (f08csc).  
*On exit:* the  $m$  by  $n$  matrix  $Q$ .  
If **order** = Nag\_ColMajor, the  $(i, j)$ th element of the matrix is stored in  $a[(j-1) \times pda + i - 1]$ .  
If **order** = Nag\_RowMajor, the  $(i, j)$ th element of the matrix is stored in  $a[(i-1) \times pda + j - 1]$ .
  
- 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,  $pda \geq \max(1, m)$ ;  
if **order** = Nag\_RowMajor,  $pda \geq \max(1, n)$ .
  
- 7:    **tau**[ $dim$ ] – const Complex *Input*  
**Note:** the dimension,  $dim$ , of the array **tau** must be at least  $\max(1, k)$ .  
*On entry:* further details of the elementary reflectors, as returned by nag\_zgeqlf (f08csc).
  
- 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  $n = \langle value \rangle$ .

Constraint:  $m \geq n \geq 0$ .

On entry,  $n = \langle value \rangle$  and  $k = \langle value \rangle$ .

Constraint:  $n \geq k \geq 0$ .

On entry,  $pda = \langle value \rangle$  and  $m = \langle value \rangle$ .

Constraint:  $pda \geq \max(1, m)$ .

On entry,  $pda = \langle value \rangle$  and  $n = \langle value \rangle$ .

Constraint:  $pda \geq \max(1, n)$ .

### 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\_zungql (f08ctc) 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  $n = k$ , the number is approximately  $\frac{8}{3}n^2(3m - n)$ .

The real analogue of this function is nag\_dorgql (f08cfc).

## 10 Example

This example generates the first four columns of the matrix  $Q$  of the  $QL$  factorization of  $A$  as returned by nag\_zgeqlf (f08csc), where

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

### 10.1 Program Text

```
/* nag_zungql (f08ctc) 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;
    Integer exit_status = 0;
    /* Arrays */
    char *title = 0;
    Complex *a = 0, *tau = 0;
    /* Nag Types */
    Nag_OrderType order;
    NagError fail;

#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_zungql (f08ctc) 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

/* Allocate memory */
if (!(title = NAG_ALLOC(31, char)) ||
    !(a = NAG_ALLOC(m * n, Complex)) || !(tau = NAG_ALLOC(n, 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

/* nag_zgeqlf (f08csc).
 * Compute the QL factorization of A.
 */
nag_zgeqlf(order, m, n, a, pda, tau, &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_zgeqlf (f08csc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* nag_zungql (f08ctc).
 * Form the leading n columns of Q explicitly.
 */
nag_zungql(order, m, n, n, a, pda, tau, &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_zungql (f08ctc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

#ifdef _WIN32
    sprintf_s(title, 31, "The leading %4" NAG_IFMT " columns of Q", n);
#else
    sprintf(title, "The leading %4" NAG_IFMT " columns of Q", n);
#endif

/* nag_gen_complx_mat_print_comp (x04dbc).
 * Print the leading n columns of Q.
 */
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;
    }

END:
    NAG_FREE(title);
    NAG_FREE(a);
    NAG_FREE(tau);

    return exit_status;
}

#undef A

```

## 10.2 Program Data

nag\_zungql (f08ctc) Example Program Data

```

        6              4                      :Values of M and N

( 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

```

## 10.3 Program Results

nag\_zungql (f08ctc) Example Program Results

```

The leading      4 columns of Q
               1              2              3              4
1 ( 0.2810, 0.5020) (-0.2051,-0.1092) ( 0.3083,-0.6874) ( 0.0181,-0.1483)
2 ( 0.2707,-0.3296) ( 0.5711, 0.0432) ( 0.2251,-0.1313) ( 0.2930,-0.2025)
3 (-0.2864,-0.0094) (-0.5416, 0.0454) (-0.2062, 0.0691) ( 0.4015,-0.2170)
4 ( 0.2262,-0.3854) (-0.3387, 0.2228) ( 0.3259, 0.1178) (-0.0796, 0.0723)
5 ( 0.0341,-0.0760) ( 0.0098,-0.0712) ( 0.0753, 0.1412) (-0.5317,-0.5751)
6 (-0.3936,-0.2083) (-0.1296, 0.3691) ( 0.0264,-0.4134) (-0.0940,-0.0940)

```

---