NAG Library Function Document

nag_zungrq (f08cwc)

1 Purpose

nag_zungrq (f08cwc) generates all or part of the complex \( n \) by \( n \) unitary matrix \( Q \) from an \( RQ \) factorization computed by nag_zgerqf (f08cvc).

2 Specification

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

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

3 Description

nag_zungrq (f08cwc) is intended to be used following a call to nag_zgerqf (f08cvc), which performs an \( RQ \) factorization of a complex matrix \( A \) and represents the unitary matrix \( Q \) as a product of \( k \) elementary reflectors of order \( n \).

This function may be used to generate \( Q \) explicitly as a square matrix, or to form only its trailing rows. Usually \( Q \) is determined from the \( RQ \) factorization of a \( p \) by \( n \) matrix \( A \) with \( p \leq n \). The whole of \( Q \) may be computed by:

```
nag_zungrq(order,n,n,p,a,pda,tau,info)
```

(note that the matrix \( A \) must have at least \( n \) rows), or its trailing \( p \) rows as:

```
nag_zungrq(order,p,n,p,a,pda,tau,info)
```

The rows of \( Q \) returned by the last call form an orthonormal basis for the space spanned by the rows of \( A \); thus nag_zgerqf (f08cvc) followed by nag_zungrq (f08cwc) can be used to orthogonalize the rows of \( A \).

The information returned by nag_zgerqf (f08cvc) also yields the \( RQ \) factorization of the trailing \( k \) rows of \( A \), where \( k < p \). The unitary matrix arising from this factorization can be computed by:

```
nag_zungrq(order,n,n,k,a,pda,tau,info)
```
or its leading \( k \) columns by:

```
nag_zungrq(order,k,n,k,a,pda,tau,info)
```

4 References


5 Arguments

1: \textbf{order} – Nag_OrderType

\textit{Input}

\textit{On entry}: the \textbf{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

\textbf{Mark 25}
\texttt{order} = \texttt{Nag\_RowMajor}. See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

\textit{Constraint:} \texttt{order} = \texttt{Nag\_RowMajor} or \texttt{Nag\_ColMajor}.

2: \texttt{m} – Integer \hspace{1cm} \textit{Input}

\textit{On entry:} \(m\), the number of rows of the matrix \(Q\).

\textit{Constraint:} \(m \geq 0\).

3: \texttt{n} – Integer \hspace{1cm} \textit{Input}

\textit{On entry:} \(n\), the number of columns of the matrix \(Q\).

\textit{Constraint:} \(n \geq m\).

4: \texttt{k} – Integer \hspace{1cm} \textit{Input}

\textit{On entry:} \(k\), the number of elementary reflectors whose product defines the matrix \(Q\).

\textit{Constraint:} \(m \geq k \geq 0\).

5: \texttt{a[\text{dim}]} – Complex \hspace{1cm} \textit{Input/Output}

\textbf{Note:} the dimension, \texttt{dim}, of the array \texttt{a} must be at least

\[\max(1, \texttt{pda} \times \texttt{n})\] when \texttt{order} = \texttt{Nag\_ColMajor};

\[\max(1, \texttt{m} \times \texttt{pda})\] when \texttt{order} = \texttt{Nag\_RowMajor}.

\textit{On entry:} details of the vectors which define the elementary reflectors, as returned by \texttt{nag\_zgerqf} (\texttt{f08cvc}).

\textit{On exit:} the \(m\) by \(n\) matrix \(Q\).

If \texttt{order} = \texttt{Nag\_ColMajor}, the \((i,j)\)th element of the matrix is stored in \(\texttt{a}[(j-1) \times \texttt{pda} + i - 1]\).

If \texttt{order} = \texttt{Nag\_RowMajor}, the \((i,j)\)th element of the matrix is stored in \(\texttt{a}[(i-1) \times \texttt{pda} + j - 1]\).

6: \texttt{pda} – Integer \hspace{1cm} \textit{Input}

\textit{On entry:} the stride separating row or column elements (depending on the value of \texttt{order}) in the array \texttt{a}.

\textbf{Constraints:}

\[
\begin{align*}
\text{if order} &= \texttt{Nag\_ColMajor}, \; \texttt{pda} \geq \max(1, \texttt{m}); \\
\text{if order} &= \texttt{Nag\_RowMajor}, \; \texttt{pda} \geq \max(1, \texttt{n}).
\end{align*}
\]

7: \texttt{tau[\text{dim}]} – const Complex \hspace{1cm} \textit{Input}

\textbf{Note:} the dimension, \texttt{dim}, of the array \texttt{tau} must be at least \(\max(1, \texttt{k})\).

\textit{On entry:} \texttt{tau[i-1]} must contain the scalar factor of the elementary reflector \(H_i\), as returned by \texttt{nag\_zgerqf} (\texttt{f08cvc}).

8: \texttt{fail} – \texttt{Nag\_Error*} \hspace{1cm} \textit{Input/Output}

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

\section{Error Indicators and Warnings}

\textbf{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 \(value\) had an illegal value.

NE_INT
On entry, \(m = value\).
Constraint: \(m \geq 0\).
On entry, \(pda = value\).
Constraint: \(pda > 0\).

NE_INT_2
On entry, \(m = value\) and \(k = value\).
Constraint: \(m \geq k \geq 0\).
On entry, \(n = value\) and \(m = value\).
Constraint: \(n \geq m\).
On entry, \(pda = value\) and \(m = value\).
Constraint: \(pda \geq \max(1,m)\).
On entry, \(pda = value\) and \(n = value\).
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 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 matrix \(Q\) differs from an exactly unitary matrix by a matrix \(E\) such that
\[
\|E\|_2 = O\epsilon
\]
and \(\epsilon\) is the machine precision.

8 Parallelism and Performance
nag_zungrq (f08cwc) is not threaded by NAG in any implementation.
nag_zungrq (f08cwc) 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 floating-point operations is approximately \(16mnk - 8(m + n)k^2 + \frac{16}{3}k^3\); when \(m = k\) this becomes \(\frac{2}{3}m^2(3n - m)\).
The real analogue of this function is nag_dorgrq (f08cjc).
10 Example

This example generates the first four rows of the matrix \( Q \) of the \( RQ \) factorization of \( A \) as returned by \texttt{nag_zgerqf (f08cvc)}, where

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

10.1 Program Text

```c
/* nag_zungrq (f08cwc) Example Program. *
* Copyright 2014 Numerical Algorithms Group.
* Mark 23, 2011. */

#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_zungrq (f08cwc) Example Program Results\n\n");

   /* Skip heading in data file */
   #ifdef _WIN32
   scanf_s("%*[\n]");
   #else
   scanf("%*[\n]");
   #endif
   #ifdef _WIN32
   scanf("%"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(27, char)) ||
```

f08cwc.4

---

**10 Example**

This example generates the first four rows of the matrix \( Q \) of the \( RQ \) factorization of \( A \) as returned by \texttt{nag_zgerqf (f08cvc)}, where

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

**10.1 Program Text**

```c
/* nag_zungrq (f08cwc) Example Program. *
* Copyright 2014 Numerical Algorithms Group.
* Mark 23, 2011. */

#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_zungrq (f08cwc) Example Program Results\n\n");

   /* Skip heading in data file */
   #ifdef _WIN32
   scanf_s("%*[\n]");
   #else
   scanf("%*[\n]");
   #endif
   #ifdef _WIN32
   scanf("%"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(27, 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_zgerqf(order, m, n, a, pda, tau, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_zgerqf (f08cvc).
%s
", fail.message);
    exit_status = 1;
    goto END;
}

nag_zungrq(order, m, n, m, a, pda, tau, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_zungrq (f08cwc).
%s
", fail.message);
    exit_status = 1;
    goto END;
}

#ifdef _WIN32
    sprintf_s(title, 27, "The leading %4"NAG_IFMT" rows of Q", m);
#else
    sprintf(title, "The leading %4"NAG_IFMT" rows of Q", m);
#endif

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).
%s
", 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_zungrq (f08cwc) Example Program Data

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>Values of M and N</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0.96, -0.81)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-0.98, 1.98)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0.62, -0.46)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-0.37, 0.38)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0.83, 0.51)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(1.08, -0.28)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-0.03, 0.96)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-1.20, 0.19)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(1.01, 0.02)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0.19, -0.54)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0.20, 0.01)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-0.91, 2.06)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-0.66, 0.42)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0.63, -0.17)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-0.98, -0.36)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-0.17, -0.46)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-0.07, 1.23)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-0.05, 0.41)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-0.81, 0.56)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-1.11, 0.60)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0.22, -0.20)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(1.47, 1.59)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0.26, 0.26)</td>
</tr>
</tbody>
</table>

End of matrix A

10.3 Program Results

nag_zungrq (f08cwc) Example Program Results

The leading 4 rows of Q

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(0.2810, 0.5020)</td>
<td>(0.2707, -0.3296)</td>
<td>(-0.2864, -0.0094)</td>
<td>(0.2262, -0.3854)</td>
</tr>
<tr>
<td>2</td>
<td>(-0.2051, -0.1092)</td>
<td>(0.5711, 0.0432)</td>
<td>(-0.5416, 0.0454)</td>
<td>(-0.3387, 0.2228)</td>
</tr>
<tr>
<td>3</td>
<td>(0.3083, -0.6874)</td>
<td>(0.2251, -0.1313)</td>
<td>(-0.2062, 0.0691)</td>
<td>(0.3259, 0.1178)</td>
</tr>
<tr>
<td>4</td>
<td>(0.0181, -0.1483)</td>
<td>(0.2930, -0.2025)</td>
<td>(0.4015, -0.2170)</td>
<td>(-0.0796, 0.0723)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(0.0341, -0.0760)</td>
<td>(-0.3936, -0.2083)</td>
</tr>
<tr>
<td>2</td>
<td>(0.0098, -0.0712)</td>
<td>(-0.1296, 0.3691)</td>
</tr>
<tr>
<td>3</td>
<td>(0.0753, 0.1412)</td>
<td>(0.0264, -0.4134)</td>
</tr>
<tr>
<td>4</td>
<td>(-0.5317, -0.5751)</td>
<td>(-0.0940, -0.0940)</td>
</tr>
</tbody>
</table>