NAG C Library Function Document

nag_dorghr (f08nfc)

1 Purpose

nag_dorghr (f08nfc) generates the real orthogonal matrix $Q$ which was determined by nag_dgehrd (f08nec) when reducing a real general matrix $A$ to Hessenberg form.

2 Specification

```c
void nag_dorghr (Nag_OrderType order, Integer n, Integer ilo, Integer ihi, double a[], Integer pda, const double tau[], NagError *fail)
```

3 Description

nag_dorghr (f08nfc) is intended to be used following a call to nag_dgehrd (f08nec), which reduces a real general matrix $A$ to upper Hessenberg form $H$ by an orthogonal similarity transformation: $A = QHQ^T$. nag_dgehrd (f08nec) represents the matrix $Q$ as a product of $ihi/ilo$ elementary reflectors. Here $ilo$ and $ihi$ are values determined by nag_dgebal (f08nhc) when balancing the matrix; if the matrix has not been balanced, $ilo = 1$ and $ihi = n$.

This function may be used to generate $Q$ explicitly as a square matrix. $Q$ has the structure:

$$Q = \begin{pmatrix} I & 0 & 0 \\ 0 & Q_{22} & 0 \\ 0 & 0 & I \end{pmatrix}$$

where $Q_{22}$ occupies rows and columns $ilo$ to $ihi$.

4 References


5 Parameters

1: `order` – Nag_OrderType

   On entry: the order parameter 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 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.

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

2: `n` – Integer

   On entry: $n$, the order of the matrix $Q$.

   Constraint: $n \geq 0$.

3: `ilo` – Integer

4: `ihi` – Integer

   On entry: these must be the same parameters $ilo$ and $ihi$, respectively, as supplied to nag_dgehrd (f08nec).

   Constraints:

   - if $n > 0$, $1 \leq ilo \leq ihi \leq n$;
   - if $n = 0$, $ilo = 1$ and $ihi = 0$.

[NP3645/7]
5: \( a[dim] \) – double

**Input/Output**

**Note:** the dimension, \( dim \), of the array \( a \) must be at least \( \max(1, pda \times n) \).

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

**On entry:** details of the vectors which define the elementary reflectors, as returned by \text{nag\_dgehrd} (f08nec).

**On exit:** the \( n \) by \( n \) orthogonal matrix \( Q \).

6: \( pda \) – Integer

**Input**

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

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

7: \( \tau[dim] \) – const double

**Input**

**Note:** the dimension, \( dim \), of the array \( \tau \) must be at least \( \max(1, n - 1) \).

**On entry:** further details of the elementary reflectors, as returned by \text{nag\_dgehrd} (f08nec).

8: \text{fail} – \text{NagError} *

**Output**

The NAG error parameter (see the Essential Introduction).

### 6 Error Indicators and Warnings

**NE\_INT**

**On entry:** \( n = \langle\text{value}\rangle \).

**Constraint:** \( n \geq 0 \).

**On entry:** \( pda = \langle\text{value}\rangle \).

**Constraint:** \( pda > 0 \).

**NE\_INT\_2**

**On entry:** \( pda = \langle\text{value}\rangle, n = \langle\text{value}\rangle \).

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

**NE\_INT\_3**

**On entry:** \( n = \langle\text{value}\rangle, \text{ilo} = \langle\text{value}\rangle, \text{ihi} = \langle\text{value}\rangle \).

**Constraint:** if \( n > 0 \), \( 1 \leq \text{ilo} \leq \text{ihi} \leq n \);
if \( n = 0 \), \( \text{ilo} = 1 \) and \( \text{ihi} = 0 \).

**NE\_ALLOC\_FAIL**

Memory allocation failed.

**NE\_BAD\_PARAM**

**On entry:** parameter \( \langle\text{value}\rangle \) had an illegal value.

**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 consult NAG for assistance.
7 Accuracy

The computed matrix \( Q \) differs from an exactly orthogonal matrix by a matrix \( E \) such that
\[
\|E\|_2 = O(\epsilon),
\]
where \( \epsilon \) is the machine precision.

8 Further Comments

The total number of floating-point operations is approximately \( \frac{4}{3}q^3 \), where \( q = i_{hi} - i_{lo} \).

The complex analogue of this function is nag_zunghr (f08ntc).

9 Example

To compute the Schur factorization of the matrix \( A \), where
\[
A = \begin{pmatrix}
0.35 & 0.45 & -0.14 & -0.17 \\
0.09 & 0.07 & -0.54 & 0.35 \\
-0.44 & -0.33 & -0.03 & 0.17 \\
0.25 & -0.32 & -0.13 & 0.11
\end{pmatrix},
\]
Here \( A \) is general and must first be reduced to Hessenberg form by nag_dgehrd (f08nec). The program then calls nag_dorghr (f08nfc) to form \( Q \), and passes this matrix to nag_dhseqr (f08pec) which computes the Schur factorization of \( A \).

9.1 Program Text

```c
/* nag_dorghr (f08nfc) Example Program.  
 * Copyright 2001 Numerical Algorithms Group.  
*/
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf08.h>
#include <nagx04.h>

int main(void)
{
    /* Scalars */
    Integer i, j, n, pda, pdz, tau_len, wr_len, wi_len;
    Integer exit_status=0;
    NagError fail;
    Nag_OrderType order;
    /* Arrays */
    double *a=0, *tau=0, *wi=0, *wr=0, *z=0;
    #ifdef NAG_COLUMN_MAJOR
    #define A(I,J) a[(J-1)*pda+I-1]
    #define Z(I,J) z[(J-1)*pdz+I-1]
    order = Nag_ColMajor;
    #else
    #define A(I,J) a[(I-1)*pda+J-1]
    #define Z(I,J) z[(I-1)*pdz+J-1]
    order = Nag_RowMajor;
    #endif
    INIT_FAIL(fail);
    Vprintf("f08nfc Example Program Results\n\n");
    /* Skip heading in data file */
    Vscanf("%*[^\n] ");
}
```

[NP3645/7]
Vscanf("%ld%*[\n ] ", &n);
#ifdef NAG_COLUMN_MAJOR
    pda = n;
    pdz = n;
#else
    pda = n;
    pdz = n;
#endif
    tau_len = n - 1;
    wr_len = n;
    wi_len = n;

    /* Allocate memory */
    if ( !(a = NAG_ALLOC(n * n, double)) ||
         !(tau = NAG_ALLOC(tau_len, double)) ||
         !(wi = NAG_ALLOC(wi_len, double)) ||
         !(wr = NAG_ALLOC(wi_len, double)) ||
         !(z = NAG_ALLOC(n * n, double)) )
    {
        Vprintf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    /* Read A from data file */
    for (i = 1; i <= n; ++i)
    {
        for (j = 1; j <= n; ++j)
            Vscanf("%lf", &A(i,j));
    }
    Vscanf("%*[\n ] ");

    /* Reduce A to upper Hessenberg form H = (Q**T)*A*Q */
    f08nec(order, n, 1, n, a, pda, tau, &fail);
    if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from f08nec.\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }

    /* Copy A into Z */
    for (i = 1; i <= n; ++i)
    {
        for (j = 1; j <= n; ++j)
            Z(i,j) = A(i,j);
    }

    /* Form Q explicitly, storing the result in Z */
    f08nfc(order, n, 1, n, z, pdz, tau, &fail);
    if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from f08nfc.\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }

    /* Calculate the Schur factorization of H = Y*T*(Y**T) and form */
    /* Q*Y explicitly, storing the result in Z */
    /* Note that A = Z*T*(Z**T), where Z = Q*Y */
    f08pec(order, Nag_Schur, Nag_UpdateZ, n, 1, n, a, pda,
            wr, wi, z, pdz, &fail);
    if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from f08pec.\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }

    /* Print Schur form */
x04cac(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, n,
a, pda, "Schur form", 0, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from x04cac.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Print Schur vectors */
Vprintf("\n");
x04cac(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, n,
z, pdz, "Schur vectors of A", 0, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from x04cac.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

END:
if (a) NAG_FREE(a);
if (tau) NAG_FREE(tau);
if (wi) NAG_FREE(wi);
if (wr) NAG_FREE(wr);
if (z) NAG_FREE(z);
return exit_status;

9.2 Program Data
f08nfc Example Program Data
4 :Value of N
  0.35  0.45 -0.14 -0.17
  0.09  0.07 -0.54  0.35
-0.44 -0.33 -0.03  0.17
  0.25 -0.32 -0.13  0.11 :End of matrix A

9.3 Program Results
f08nfc Example Program Results

Schur form

<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>0.7995</td>
<td>0.0060</td>
<td>-0.1144</td>
<td>-0.0336</td>
</tr>
<tr>
<td>2</td>
<td>0.0000</td>
<td>-0.0994</td>
<td>-0.6483</td>
<td>-0.2026</td>
</tr>
<tr>
<td>3</td>
<td>0.0000</td>
<td>0.2478</td>
<td>-0.0994</td>
<td>-0.3474</td>
</tr>
<tr>
<td>4</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>-0.1007</td>
</tr>
</tbody>
</table>

Schur vectors of A

<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>-0.6551</td>
<td>-0.3450</td>
<td>-0.1037</td>
<td>0.6641</td>
</tr>
<tr>
<td>2</td>
<td>-0.5236</td>
<td>0.6141</td>
<td>0.5807</td>
<td>-0.1068</td>
</tr>
<tr>
<td>3</td>
<td>0.5362</td>
<td>0.2935</td>
<td>0.3073</td>
<td>0.7293</td>
</tr>
<tr>
<td>4</td>
<td>-0.0956</td>
<td>0.6463</td>
<td>-0.7467</td>
<td>0.1249</td>
</tr>
</tbody>
</table>