nag_trans_hessenberg_observer (g13ewc)

1. Purpose

nag_trans_hessenberg_observer (g13ewc) reduces the matrix pair (A,C) to lower or upper observer Hessenberg form using (and optionally accumulating) the unitary state-space transformations.

2. Specification

```
#include <nag.h>
#include <nagg13.h>

void g13ewc(Integer n, Integer p, Nag_ObserverForm reduceto, double a[],
            Integer tda, double c[], Integer tdc, double u[], Integer tdu,
            NagError *fail)
```

3. Description

The function computes a unitary state-space transformation U, which reduces the matrix pair (A,C) to give a compound matrix in one of the following observer Hessenberg forms:

$$\begin{pmatrix} U\,A\,U^T \\ \hline C\,U^T \end{pmatrix} = \overbrace{\left(\begin{array}{cccccccc} * & . & . & . & . & . & . & * \\ . & & & & & & & . \\ . & & & & & & & . \\ . & & & & & & & . \\ * & & & & & & & . \\ & . & & & & & & . \\ & & . & & & & & . \\ & & & * & . & . & . & * \\ \hline & & & & * & . & . & * \\ & & & & & . & & . \\ & & & & & & . & . \\ & & & & & & & * \end{array}\right)}^{n} \begin{array}{l} n \\ \\ p \end{array}$$

if **reduceto** = **Nag_UH_Observer**, or

$$\begin{pmatrix} C\,U^T \\ \hline U\,A\,U^T \end{pmatrix} = \overbrace{\left(\begin{array}{cccccccc} * & & & & & & & \\ . & . & & & & & & \\ . & & . & & & & & \\ * & . & . & * & & & & \\ \hline * & . & . & . & * & & & \\ . & & & & & . & & \\ . & & & & & & . & \\ . & & & & & & & * \\ . & & & & & & & . \\ . & & & & & & & . \\ . & & & & & & & . \\ * & . & . & . & . & . & . & * \end{array}\right)}^{n} \begin{array}{l} p \\ \\ n \end{array}$$

if **reduceto** = **Nag_LH_Observer**.

If $p > n$, then the matrix CU^T is trapezoidal and if $p + 1 \geq n$, then the matrix UAU^T is full.

4. Parameters

n

Input: The actual state dimension, n, i.e., the order of the matrix A.

Constraint: $\mathbf{n} \geq 1$.

p

Input: The actual output dimension, p.

Constraint: $\mathbf{p} \geq 1$.

reduceto

Input: Indicates whether the matrix pair (A,C) is to be reduced to upper or lower observer Hessenberg form as follows:
reduceto = **Nag_UH_Observer**, (Upper observer Hessenberg form);
reduceto = **Nag_LH_Observer**, (Lower observer Hessenberg form).

a[n][tda]

Input: The leading n by n part of this array must contain the state transition matrix A to be transformed.

Output: The leading n by n part of this array contains the transformed state transition matrix UAU^T.

tda

Input: The trailing dimension of array **a** as declared in the calling program.

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

c[p][tdc]

Input: The leading p by n part of this array must contain the output matrix C to be transformed.

Output: The leading p by n part of this array contains the transformed output matrix CU^T.

tdc

The trailing dimension of array **c** as declared in the calling program.

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

u[n][tdu]

Input: If **u** is defined, then the leading n by n part of this array must contain either a transformation matrix (e.g. from a previous call to this function) or be initialised as the identity matrix. If this information is not to be input then **u** must be set to the null pointer, i.e., (double $*$)0.

Output: If **u** is defined, then the leading n by n part of this array contains the product of the input matrix U and the state-space transformation matrix which reduces the given pair to observer Hessenberg form.

tdu

Input: The trailing dimension of array **u** as declared in the calling program.

Constraint: $\mathbf{tdu} \geq \mathbf{n}$ if **u** is defined.

fail

The NAG error parameter, see the Essential Introduction to the NAG C Library.

5. Error Indications and Warnings

NE_BAD_PARAM

On entry parameter **reduceto** had an illegal value.

NE_INT_ARG_LT

On entry, **n** must not be less than 1: **n** = ⟨*value*⟩.
On entry, **p** must not be less than 1: **p** = ⟨*value*⟩.

NE_2_INT_ARG_LT

On entry **tda** = ⟨*value*⟩ while **n** = ⟨*value*⟩.
These parameters must satisfy **tda** ≥ **n**.

On entry **tdc** = ⟨*value*⟩ while **n** = ⟨*value*⟩.
These parameters must satisfy **tdc** ≥ **n**.

On entry **tdu** = ⟨*value*⟩ while **n** = ⟨*value*⟩.
These parameters must satisfy **tdu** ≥ **n**.

6. Further Comments

The algorithm requires $O((n+m)n^2)$ operations (see Van Dooren and Verhaegen 1985).

6.1. Accuracy

The algorithm is backward stable.

6.2. References

Van Dooren P and Verhaegen M (1985) On the use of unitary state-space transformations. *In: Contemporary Mathematics on Linear Algebra and its Role in Systems Theory* **47** AMS, Providence.

7. See Also

nag_kalman_sqrt_filt_info_invar (g13edc)

8. Example

To reduce the matrix pair (A,C) to upper observer Hessenberg form.

8.1. Program Text

```
/* nag_trans_hessenberg_observer(g13ewc)  Example Program
 *
 * Copyright 1994 Numerical Algorithms Group
 *
 * Mark 3, 1994.
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagg13.h>


#define NMAX 20
#define PMAX 20
```

```
#define TDA NMAX
#define TDC NMAX
#define TDU NMAX

main()
{
  double a[NMAX][TDA];
  double c[PMAX][TDC];
  double u[NMAX][TDU];
  Integer i, j, n, p;
  Nag_ObserverForm reduceto;
  double zero = 0.0, one = 1.0;
  Integer nmax, pmax;

  Vprintf("g13ewc Example Program Results\n");

  /* Skip the heading in the data file and read the data. */
  Vscanf("%*[^\n]");

  nmax = NMAX;
  pmax = PMAX;

  Vscanf("%ld%ld",&n,&p);
  if (n<=0 || p<=0 ||
      n>nmax || p>pmax)
    {
      Vfprintf(stderr,"One of n or p is out of range n = %ld, p = %ld\n", n, p);
      exit(EXIT_FAILURE);
    }

  reduceto = Nag_UH_Observer;

  for (j = 0; j < n; ++j)
    for (i = 0; i < n; ++i)
      Vscanf("%lf", &a[i][j]);
  for (i = 0; i < p; ++i)
    for (j = 0; j < n; ++j)
      Vscanf("%lf", &c[i][j]);

  if (u) /* Initialise U as the identity matrix. */
    for (i = 0; i < n; ++i)
      {
        for (j = 0; j < n; ++j)
          u[i][j] = zero;
        u[i][i] = one;
      }

  /* Reduce the pair (A,C) to reduceto observer Hessenberg form. */
  g13ewc(n, p, reduceto, (double *)a, (Integer)TDA, (double *)c, (Integer)TDC,
         (double *)u, (Integer)TDU,  NAGERR_DEFAULT);

  Vprintf("\nThe transformed state transition matrix is \n\n");
  for (i = 0; i < n; ++i)
    {
      for (j = 0; j < n; ++j)
        Vprintf("%8.4f ",a[i][j]);
      Vprintf("\n");
    }
  Vprintf("\nThe transformed input matrix is \n\n");
  for (i = 0; i < p; ++i)
    {
      for (j = 0; j < n; ++j)
        Vprintf("%8.4f ",c[i][j]);
      Vprintf("\n");
    }
  if (u)
    {
      Vprintf("\nThe transformation matrix that reduces (A,C) \
to observer Hessenberg form is \n\n");
      for (i = 0; i < n; ++i)
```

```
        {
          for (j = 0; j < n; ++j)
            Vprintf("%8.4f ",u[i][j]);
          Vprintf("\n");
        }
    }
  exit(EXIT_SUCCESS);
}
```

8.2. Program Data

```
g13ewc Example Program Data
   5     3
  15.0  21.0  -3.0   3.0   9.0
  20.0   1.0   2.0   8.0   9.0
   4.0   1.0   7.0  13.0  14.0
   5.0   6.0  12.0  13.0  -6.0
   5.0  11.0  17.0  -7.0  -1.0
   7.0  -1.0   3.0  -6.0  -3.0
   4.0   5.0   6.0  -2.0  -3.0
   9.0   8.0   5.0   2.0   1.0
```

8.3. Program Results

```
g13ewc Example Program Results

The transformed state transition matrix is

  7.1637  -0.9691 -16.5046   0.2869   0.9205
 -2.3285  11.5431  -8.7471   3.4122  -3.7118
-10.5440  -7.6032  -0.3215   3.6571  -0.4335
 -3.6845   5.6449   0.5906 -15.6996  17.4267
  0.0000  -6.4260   1.5591  14.4317  32.3143

The transformed input matrix is

  0.0000   0.0000   7.6585   5.2973  -4.1576
  0.0000   0.0000   0.0000   5.8305  -7.4837
  0.0000   0.0000   0.0000   0.0000 -13.2288

The transformation matrix that reduces (A,C) to observer Hessenberg form is

  0.1863  -0.4823   0.2645   0.6648  -0.4698
 -0.1137  -0.3601   0.6748  -0.0512   0.6320
  0.6742  -0.5151  -0.1897  -0.4940  -0.0097
 -0.1872   0.0813   0.5439  -0.5371  -0.6116
 -0.6803  -0.6047  -0.3780  -0.1512  -0.0756
```