NAG Library Function Document

nag_trans_hessenberg_controller (g13exc)

1 Purpose

nag_trans_hessenberg_controller (g13exc) reduces the matrix pair (B, A) to lower or upper controller Hessenberg form using (and optionally accumulating) the unitary state-space transformations.

2 Specification

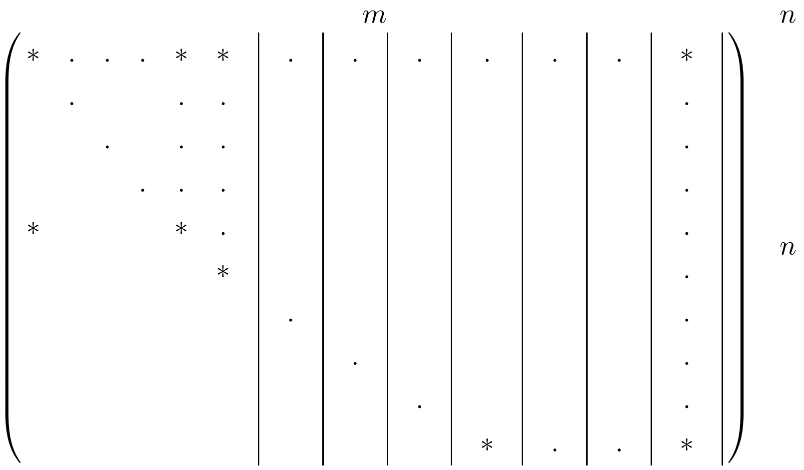
```
#include <nag.h>
#include <nagg13.h>
```

```
void nag_trans_hessenberg_controller (Integer n, Integer m,
     Nag_ControllerForm reduceto, double a[], Integer tda, double b[],
     Integer tdb, double u[], Integer tdu, NagError *fail)
```

3 Description

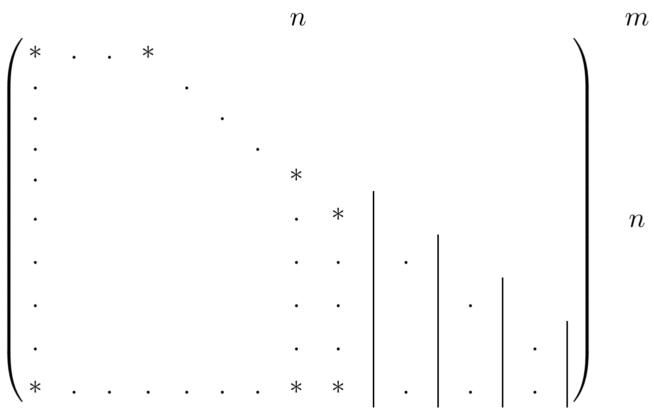
nag_trans_hessenberg_controller (g13exc) computes a unitary state-space transformation U, which reduces the matrix pair (B, A) to give a compound matrix in one of the following controller Hessenberg forms:

$$(UB \mid UAU^{\mathrm{T}}) =$$



if **reduceto** $=$ Nag_UH_Controller, or

$$(UAU^{\mathrm{T}} \mid UB) =$$



if **reduceto** $=$ Nag_LH_Controller. If $m > n$, then the matrix UB is trapezoidal and if $m + 1 \geq n$ then the matrix UAU^{T} is full.

4 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

5 Arguments

1: **n** – Integer *Input*

On entry: the actual state dimension, n, i.e., the order of the matrix A.

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

2: **m** – Integer *Input*

On entry: the actual input dimension, m.

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

3: **reduceto** – Nag_ControllerForm *Input*

On entry: indicates whether the matrix pair (B, A) is to be reduced to upper or lower controller Hessenberg form as follows:

reduceto = Nag_UH_Controller
    Upper controller Hessenberg form).

reduceto = Nag_LH_Controller
    Lower controller Hessenberg form).

Constraint: **reduceto** = Nag_UH_Controller or Nag_LH_Controller.

4: **a**[**n** × **tda**] – double *Input/Output*

Note: the (i, j)th element of the matrix A is stored in $\mathbf{a}[(i-1) \times \mathbf{tda} + j - 1]$.

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

On exit: the leading n by n part of this array contains the transformed state transition matrix UAU^{T}.

5: **tda** – Integer *Input*

On entry: the stride separating matrix column elements in the array **a**.

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

6: **b**[**n** × **tdb**] – double *Input/Output*

Note: the (i, j)th element of the matrix B is stored in $\mathbf{b}[(i-1) \times \mathbf{tdb} + j - 1]$.

On entry: the leading n by m part of this array must contain the input matrix B to be transformed.

On exit: the leading n by m part of this array contains the transformed input matrix UB.

7: **tdb** – Integer *Input*

On entry: the stride separating matrix column elements in the array **b**.

Constraint: $\mathbf{tdb} \geq \mathbf{m}$.

8: **u**[**n** × **tdu**] – double *Input/Output*

Note: the (i, j)th element of the matrix U is stored in $\mathbf{u}[(i-1) \times \mathbf{tdu} + j - 1]$.

On entry: if **u** is not **NULL**, 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 initialized as the identity matrix. If this information is not to be input then **u** must be set to **NULL**.

On exit: if **u** is not **NULL**, 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.

9: **tdu** – Integer *Input*

On entry: the stride separating matrix column elements in the array **u**.

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

10: **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_2_INT_ARG_LT

On entry, **tda** $= \langle$*value*$\rangle$ while **n** $= \langle$*value*$\rangle$. These arguments must satisfy **tda** $\geq$ **n**. On entry **tdb** $= \langle$*value*$\rangle$ while **m** $= \langle$*value*$\rangle$. These arguments must satisfy **tdb** $\geq$ **m**. On entry **tdu** $= \langle$*value*$\rangle$ while **n** $= \langle$*value*$\rangle$. These arguments must satisfy **tdu** $\geq$ **n**.

NE_BAD_PARAM

On entry, argument **reduceto** had an illegal value.

NE_INT_ARG_LT

On entry, **m** $= \langle$*value*$\rangle$.
Constraint: **m** ≥ 1.

On entry, **n** $= \langle$*value*$\rangle$.
Constraint: **n** ≥ 1.

7 Accuracy

The algorithm is backward stable.

8 Parallelism and Performance

nag_trans_hessenberg_controller (g13exc) is not threaded in any implementation.

9 Further Comments

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

10 Example

To reduce the matrix pair (B, A) to upper controller Hessenberg form, and return the unitary state-space transformation matrix U.

10.1 Program Text

```
/* nag_trans_hessenberg_controller (g13exc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group
 *
 * Mark 26, 2016.
 */

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

#define A(I, J) a[(I) *tda + J]
#define B(I, J) b[(I) *tdb + J]
#define U(I, J) u[(I) *tdu + J]
int main(void)
{
  Integer exit_status = 0, i, j, m, n, tda, tdb, tdu;
  double *a = 0, *b = 0, one = 1.0, *u = 0, zero = 0.0;
  Nag_ControllerForm reduceto;
  NagError fail;

  INIT_FAIL(fail);

  printf("nag_trans_hessenberg_controller (g13exc) Example Program Results\n");

  /* Skip the heading in the data file and read the data. */
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif

#ifdef _WIN32
  scanf_s("%" NAG_IFMT "%" NAG_IFMT "", &n, &m);
#else
  scanf("%" NAG_IFMT "%" NAG_IFMT "", &n, &m);
#endif
  if (n >= 1 || m >= 1) {
    if (!(a = NAG_ALLOC(n * n, double)) ||
        !(b = NAG_ALLOC(n * m, double)) || !(u = NAG_ALLOC(n * n, double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
    tda = n;
    tdb = m;
    tdu = n;
  }
  else {
    printf("Invalid n or m.\n");
    exit_status = 1;
    return exit_status;
  }
  reduceto = Nag_UH_Controller;

  for (j = 0; j < n; ++j)
    for (i = 0; i < n; ++i)
#ifdef _WIN32
      scanf_s("%lf", &A(i, j));
#else
      scanf("%lf", &A(i, j));
#endif
  for (j = 0; j < m; ++j)
    for (i = 0; i < n; ++i)
#ifdef _WIN32
```

```
      scanf_s("%lf", &B(i, j));
#else
      scanf("%lf", &B(i, j));
#endif

  if (u) /* Initialize 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 (B,A) to reduceto controller Hessenberg form. */
  /* nag_trans_hessenberg_controller (g13exc).
   * Unitary state-space transformation to reduce (BA) to
   * lower or upper controller Hessenberg form
   */
  nag_trans_hessenberg_controller(n, m, reduceto, a, tda, b, tdb, u, tdu,
                                  &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_trans_hessenberg_controller (g13exc).\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }

  printf("\nThe transformed state transition matrix is\n\n");
  for (i = 0; i < n; ++i) {
    for (j = 0; j < n; ++j)
      printf("%8.4f ", A(i, j));
    printf("\n");
  }

  printf("\nThe transformed input matrix is\n\n");
  for (i = 0; i < n; ++i) {
    for (j = 0; j < m; ++j)
      printf("%8.4f ", B(i, j));
    printf("\n");
  }
  if (u) {
    printf("\nThe matrix that reduces (B,A) to ");
    printf("controller Hessenberg form is\n\n");
    for (i = 0; i < n; ++i) {
      for (j = 0; j < n; ++j)
        printf("%8.4f ", U(i, j));
      printf("\n");
    }
  }
END:
  NAG_FREE(a);
  NAG_FREE(b);
  NAG_FREE(u);
  return exit_status;
}
```

10.2 Program Data

```
nag_trans_hessenberg_controller (g13exc) Example Program Data
  6     3
  35.0   1.0   6.0  26.0  19.0  24.0
   3.0  32.0   7.0  21.0  23.0  25.0
  31.0   9.0   2.0  22.0  27.0  20.0
   8.0  28.0  33.0  17.0  10.0  15.0
  30.0   5.0  34.0  12.0  14.0  16.0
   4.0  36.0  29.0  13.0  18.0  11.0
   1.0   5.0  11.0
```

```
  -1.0   4.0  11.0
  -5.0   1.0   9.0
 -11.0  -4.0   5.0
 -19.0 -11.0  -1.0
 -29.0 -20.0  -9.0
```

10.3 Program Results

```
nag_trans_hessenberg_controller (g13exc) Example Program Results

The transformed state transition matrix is

 60.3649  58.8853   5.0480  -5.4406   2.1382  -7.3870
 54.5832  33.1865  36.5234   6.3272  -3.1377   8.8154
 17.6406  21.4501 -13.5942   0.5417   1.6926   0.0786
 -9.0567  10.7202   0.3531   1.5444  -1.2846  24.6407
  0.0000   6.8796 -20.1372  -2.6440   2.4983 -21.8071
  0.0000   0.0000   0.0000  -0.0000   0.0000  27.0000

The transformed input matrix is

-16.8819  -8.8260  13.9202
  0.0000  13.8240  39.9205
  0.0000   0.0000   4.1928
  0.0000   0.0000   0.0000
  0.0000   0.0000   0.0000
  0.0000   0.0000   0.0000

The matrix that reduces (B,A) to controller Hessenberg form is

 -0.0592  -0.2962  -0.6516   0.0592  -0.2369  -0.6516
 -0.3995  -0.1168   0.2350  -0.7579  -0.4406  -0.0543
 -0.5311  -0.5286  -0.3131   0.1029   0.2119   0.5339
 -0.2594   0.5309  -0.3641  -0.3950   0.5927  -0.1051
  0.6357  -0.0637  -0.4542  -0.4149  -0.1423   0.4394
 -0.2887   0.5774  -0.2887   0.2887  -0.5774   0.2887
```