nag_kalman_sqrt_filt_cov_invar (g13ebc)

1. Purpose

nag_kalman_sqrt_filt_cov_invar (g13ebc) performs a combined measurement and time update of one iteration of the time-invariant Kalman filter. The method employed for this update is the square root covariance filter with the system matrices transformed into condensed observer Hessenberg form.

2. Specification

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

void g13ebc(Integer n, Integer m, Integer p, double s[], Integer tds,
            double a[], Integer tda, double b[], Integer tdb, double q[],
            Integer tdq, double c[], Integer tdc, double r[], Integer tdr,
            double k[], Integer tdk, double h[], Integer tdh, double tol,
            NagError *fail)
```

3. Description

For the state space system defined by

$$\begin{array}{ll} X_{i+1} = AX_i + BW_i & \mathrm{var}(W_i) = Q_i \\ Y_i \quad = CX_i + V_i & \mathrm{var}(V_i) = R_i \end{array}$$

the estimate of X_i given observations Y_1 to Y_{i-1} is denoted by $\hat{X}_{i|i-1}$, with $\mathrm{var}(\hat{X}_{i|i-1}) = P_{i|i-1} = S_i S_i^T$ (where A, B and C are time invariant).

The function performs one recursion of the square root covariance filter algorithm, summarized as follows:

$$\begin{pmatrix} R_i^{1/2} & 0 & CS_i \\ 0 & BQ_i^{1/2} & AS_i \end{pmatrix} U_1 = \begin{pmatrix} H_i^{1/2} & 0 & 0 \\ G_i & S_{i+1} & 0 \end{pmatrix}$$

(Pre-array) (Post-array)

where U_1 is an orthogonal transformation triangularizing the pre-array, and the matrix pair (A,C) is in lower observer Hessenberg form. The triangularization is carried out via Householder transformations exploiting the zero pattern of the pre-array.

An example of the pre-array is given below (where $n = 6$, $p = 2$ and $m = 3$):

$$\left(\begin{array}{cc|ccc|cccccc}
x & & & & & x & & & & & \\
x & x & & & & x & x & & & & \\ \hline
 & & x & x & x & x & x & x & & & \\
 & & x & x & x & x & x & x & x & & \\
 & & x & x & x & x & x & x & x & x & \\
 & & x & x & x & x & x & x & x & x & x \\
 & & x & x & x & x & x & x & x & x & x \\
 & & x & x & x & x & x & x & x & x & x
\end{array}\right)$$

The measurement-update for the estimated state vector X is

$$\hat{X}_{i|i} = \hat{X}_{i|i-1} - K_i[C\hat{X}_{i|i-1} - Y_i]$$

whilst the time-update for X is

$$\hat{X}_{i+1|i} = A\hat{X}_{i|i} + D_i U_i$$

where $D_i U_i$ represents any deterministic control used. The relationship between the Kalman gain matrix K_i and G_i is

$$AK_i = G_i \left(H_i^{1/2} \right)$$

The function returns the product of the matrices A and K_i, represented as AK_i, and the state covariance matrix $P_{i|i-1}$ factorised as $P_{i|i-1} = S_i S_i^T$ (see the Introduction to Chapter g13 for more information concerning the covariance filter).

4. Parameters

n

Input: The actual state dimension, n, i.e., the order of the matrices S_i and A.

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

m

Input: The actual input dimension, m, i.e., the order of the matrix $Q_i^{1/2}$.

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

p

Input: The actual output dimension, p, i.e., the order of the matrix $R_i^{1/2}$.

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

s[n][tds]

Input: The leading n by n lower triangular part of this array must contain S_i, the left Cholesky factor of the state covariance matrix $P_{i|i-1}$.

Output: The leading n by n lower triangular part of this array contains S_{i+1}, the left Cholesky factor of the state covariance matrix $P_{i+1|i}$.

tds

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

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

a[n][tda]

Input: The leading n by n part of this array must contain the lower observer Hessenberg matrix UAU^T. Where A is the state transition matrix of the discrete system and U is the unitary transformation generated by the function nag_trans_hessenberg_observer (g13ewc).

tda

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

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

b[n][tdb]

Input: If the array argument **q** (below) has been defined then the leading n by m part of this array must contain the matrix UB, otherwise (if **q** is the null pointer (double $*$)0) then the leading n by m part of the array must contain the matrix $UBQ_i^{1/2}$. B is the input weight matrix, Q_i is the noise covariance matrix and U is the same unitary transformation used for defining array arguments **a** and **c**.

tdb

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

Constraint: **tdb** $\geq$ **m**.

q[m][tdq]

Input: If the noise covariance matrix is to be supplied separately from the input weight matrix then the leading m by m lower triangular part of this array must contain $Q_i^{1/2}$, the left Cholesky factor process noise covariance matrix. If the noise covariance matrix is to be input with the weight matrix as $BQ_i^{1/2}$ then the array **q** must be set to the null pointer, i.e., (double $*$)0.

tdq

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

Constraint: **tdq** $\geq$ **m** if **q** is defined.

c[p][tdc]

Input: The leading p by n part of this array must contain the lower observer Hessenberg matrix CU^T. Where C is the the output weight matrix of the discrete system and U is the unitary transformation matrix generated by the function nag_trans_hessenberg_observer (g13ewc).

tdc

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

Constraint: **tdc** $\geq$ **n**.

r[p][tdr]

Input: The leading p by p lower triangular part of this array must contain $R_i^{1/2}$, the left Cholesky factor of the measurement noise covariance matrix.

tdr

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

Constraint: **tdr** $\geq$ **p**.

k[n][tdk]

Output: If **k** is defined, then the leading n by p part of this array contains the AK_i, the product of the Kalman filter gain matrix K_i with the state transition matrix A. If this is not required then the array **k** must be set to the null pointer, i.e., (double $*$)0.

tdk

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

Constraint: **tdk** $\geq$ **p** if **k** is defined.

h[p][tdh]

Output: If **k** is defined, then the leading p by p lower triangular part of this array contains $H_i^{1/2}$. If **k** has not been defined then array **h** is not referenced and may be set to the null pointer i.e., (double $*$)0.

tdh

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

Constraint: **tdh** $\geq$ **p** if **k** and **h** are defined.

tol

Input: If **k** is defined, then tol is used to test for near singularity of the matrix $H_i^{1/2}$. If the user sets **tol** to be less than $p^2\epsilon$ then the tolerance is taken as $p^2\epsilon$, where ϵ is the ***machine precision***. Otherwise, **tol** need not be set by the user.

fail

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

5. Error Indications and Warnings

NE_INT_ARG_LT

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

NE_2_INT_ARG_LT

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

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

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

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

On entry **tdr** = ⟨*value*⟩ while **p** = ⟨*value*⟩.
These parameters must satisfy **tdr** ≥ **p**.

On entry **tdq** = ⟨*value*⟩ while **m** = ⟨*value*⟩.
These parameters must satisfy **tdq** ≥ **m**.

On entry **tdk** = ⟨*value*⟩ while **p** = ⟨*value*⟩.
These parameters must satisfy **tdk** ≥ **p**.

On entry **tdh** = ⟨*value*⟩ while **p** = ⟨*value*⟩.
These parameters must satisfy **tdh** ≥ **p**.

NE_MAT_SINGULAR

The matrix sqrt(H) is singular.

NE_NULL_ARRAY

Array **h** has null address.

NE_ALLOC_FAIL

Memory allocation failed.

6. Further Comments

The algorithm requires $\frac{1}{6}n^3 + n^2(\frac{3}{2}p + m) + 2np^2 + \frac{2}{3}p^3$ operations and is backward stable (see Verhaegen *et al*).

6.1. Accuracy

The use of the square root algorithm improves the stability of the computations.

6.2. References

Anderson B D O and Moore J B (1979) *Optimal Filtering* Prentice Hall, Englewood Cliffs, New Jersey.

Van Dooren P and Verhaegen M H G (1988) Condensed Forms for Efficient Time-Invariant Kalman Filtering *SIAM J. Sci. Stat. Comput.* **9** 516–530.

Vanbegin M, Van Dooren P and Verhaegen M H G (1989) Algorithm 675: FORTRAN Subroutines for Computing the Square Root Covariance Filter and Square Root Information Filter in Dense or Hessenberg Forms *ACM Trans. Math. Software* **15** 243–256.

Verhaegen M H G and Van Dooren P (1986) Numerical Aspects of Different Kalman Filter Implementations *IEEE Trans. Auto. Contr.* **AC-31** 907–917.

7. See Also

nag_kalman_sqrt_filt_cov_var (g13eac)
nag_trans_hessenberg_observer (g13ewc)