nag_kalman_sqrt_filt_info_var (g13ecc)

1. Purpose

nag_kalman_sqrt_filt_info_var (g13ecc) performs a combined measurement and time update of one iteration of the time-varying Kalman filter. The method employed for this update is the square root information filter with the system matrices in their original form.

2. Specification

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

void g13ecc(Integer n, Integer m, Integer p, Nag_ab_input inp_ab,
            double t[], Integer tdt, double ainv[], Integer tda,
            double b[], Integer tdb, double rinv[], Integer tdr,
            double c[], Integer tdc, double qinv[], Integer tdq,
            double x[], double rinvy[], double z[], double tol,
            NagError *fail)
```

3. Description

For the state space system defined by

$$\begin{aligned} X_{i+1} &= A_i X_i + B_i W_i \qquad \mathrm{var}(W_i) = Q_i \\ Y_i &= C_i X_i + V_i \qquad \mathrm{var}(V_i) = R_i \end{aligned}$$

the estimate of X_i given observations Y_1 to Y_{i-1} is denoted by $\hat{X}_{i|i-1}$ with $\mathrm{var}(\hat{X}_{i|i-1}) = P_{i|i-1} = S_i S_i^T$.

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

$$U_1 \begin{pmatrix} Q_i^{-1/2} & 0 & Q_i^{-1/2}\bar{w}_i \\ S_i^{-1}A_i^{-1}B_i & S_i^{-1}A_i^{-1} & S_i^{-1}\hat{X}_{i|i} \\ 0 & R_{i+1}^{-1/2}C_{i+1} & R_{i+1}^{-1/2}Y_{i+1} \end{pmatrix} = \begin{pmatrix} F_{i+1}^{-1/2} & * & * \\ 0 & S_{i+1}^{-1} & \xi_{i+1|i+1} \\ 0 & 0 & E_{i+1} \end{pmatrix}$$

(Pre-array) (Post-array)

where U_1 is an orthogonal transformation triangularizing the pre-array. The triangularization is done entirely via Householder transformations exploiting the zero pattern of the pre-array. The term $\bar{w}_i$ is the mean process noise and E_{i+1} is the estimated error at instant $i+1$. The inverse of the state covariance matrix $P_{i|i}$ is factored as follows

$$P_{i|i}^{-1} = \left(S_i^{-1}\right)^T S_i^{-1}$$

where $P_{i|i} = S_i S_i^T$ (S_i is lower triangular).

The new state filtered state estimate is computed via

$$hatX_{i+1|i+1} = S_{i+1}\xi_{i+1|i+1}$$

The function returns S_{i+1}^{-1} and $\hat{X}_{i+1|i+1}$ (see the Introduction to Chapter g13 for more information concerning the information filter).

4. Parameters

n

Input: The actual state dimension, n, i.e., the order of the matrices S_i and A_i^{-1}.

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 $R_{i+1}^{-1/2}$.

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

inp_ab

Input: Indicates how the matrix B_i is to be passed to the function.
If **inp_ab = Nag_ab_prod**, then array **b** must contain the product $A_i^{-1}B_i$.
If **inp_ab = Nag_ab_sep**, then array **b** must contain B_i.

t[n][tdt]

Input: The leading n by n upper triangular part of this array must contain S_i^{-1} the square root of the inverse of the state covariance matrix $P_{i|i}$.

Output: The leading n by n upper triangular part of this array contains S_{i+1}^{-1}, the square root of the inverse of the of the state covariance matrix $P_{i+1|i+1}$.

tdt

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

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

ainv[n][tda]

Input: The leading n by n part of this array must contain A_i^{-1} the inverse of the state transition matrix.

tda

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

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

b[n][tdb]

Input: The leading n by m part of this array must contain B_i (if **inp_ab** = **Nag_ab_sep**) or its product with A_i^{-1} (if **inp_ab** = **Nag_ab_prod**).

tdb

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

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

rinv[p][tdr]

Input: If the measurement noise covariance matrix is to be supplied separately from the output weight matrix, then the leading p by p upper triangular part of this array must contain $R_{i+1}^{-1/2}$, the right Cholesky factor of the inverse of the measurement noise covariance matrix. If this information is not to be input separately from the output weight matrix (see below) then the array **rinv** must be set to the null pointer, i.e., (double $*$)0.

tdr

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

Constraint: **tdr** $\geq$ **p** if **rinv** is defined.

c[p][tdc]

Input: The leading p by n part of this array must contain C_{i+1}, the output weight matrix (or its product with $R_{i+1}^{-1/2}$ if the array **rinv** has been set to the null pointer (double $*$)0) of the discrete system at instant $i+1$.

tdc

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

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

qinv[m][tdq]

Input: The leading m by m upper triangular part of this array must contain $Q_i^{-1/2}$ the right Cholesky factor of the inverse of the process noise covariance matrix.

tdq

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

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

x[n]

Input: This array must contain the estimated state $\hat{X}_{i|i}$

Output: The estimated state $\hat{X}_{i+1|i+1}$.

rinvy[p]

Input: This array must contain $R_{i+1}^{-1/2}Y_{i+1}$, the product of the upper triangular matrix $R_{i+1}^{-1/2}$ and the measured output Y_{i+1}.

z[m]

Input: This array must contain $\bar{w}_i$, the mean value of the state process noise.

tol

Input: **tol** is used to test for near singularity of the matrix S_{i+1}^{-1}. If the user sets **tol** to be less than $n^2 \times \epsilon$ then the tolerance is taken as $n^2 \times \epsilon$, where ϵ is the ***machine precision***.

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 **inp_ab** had an illegal value.

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 **tdt** = ⟨*value*⟩ while **n** = ⟨*value*⟩.
These parameters must satisfy **tdt** ≥ **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** ≥ **m**.

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

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

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

NE_MAT_SINGULAR

The matrix inverse(S) is singular.

NE_ALLOC_FAIL

Memory allocation failed.

6. Further Comments

The algorithm requires approximately $\frac{7}{6}n^3 + n^2(\frac{7}{2}m + p) + n(\frac{1}{2}p^2 + m^2)$ operations and is backward stable (see Verhaegen and Van Dooren 1986).

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.

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_info_invar (g13edc)