NAG Toolbox for MATLAB

g13eb

1 Purpose

g13eb performs a combined measurement and time update of one iteration of the time-invariant Kalman filter using a square root covariance filter.

2 Syntax

```
[a, b, c, s, k, h, u, ifail] = g13eb(transf, a, b, stq, q, c, r, s, 'n', n, 'm', m, 'l', l, 'tol', tol)
```

3 Description

The Kalman filter arises from the state space model given by

$$X_{i+1} = AX_i + BW_i, \quad \mathrm{Var}(W_i) = Q_i$$

$$Y_i = CX_i + V_i, \qquad \mathrm{Var}(V_i) = R_i$$

where X_i is the state vector of length n at time i, Y_i is the observation vector of length m at time i and W_i of length l and V_i of length m are the independent state noise and measurement noise respectively. The matrices A, B and C are time invariant.

The estimate of X_i given observations Y_1 to Y_{i-1} is denoted by $\hat{X}_{i|i-1}$ with state covariance matrix $\mathrm{Var}\left(\hat{X}_{i|i-1}\right) = P_{i|i-1} = S_i S_i^{\mathrm{T}}$ while the estimate of X_i given observations Y_1 to Y_i is denoted by $\hat{X}_{i|i}$ with covariance matrix $\mathrm{Var}\left(\hat{X}_{i|i}\right) = P_{i|i}$. The update of the estimate, $\hat{X}_{i|i-1}$, from time i to time $(i+1)$ is computed in two stages. First, the measurement-update is given by

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

where $K_i = P_{i|i}C^{\mathrm{T}}\left[CP_{i|i}C^{\mathrm{T}} + R_i\right]^{-1}$ is the Kalman gain matrix. The second stage is the time-update for X, which is given by

$$\hat{X}_{i+1|i} = A\hat{X}_{i|i} + D_i U_i \tag{2}$$

where $D_i U_i$ represents any deterministic control used.

The square root covariance filter algorithm provides a stable method for computing the Kalman gain matrix and the state covariance matrix. The algorithm can be summarized as

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

where U is an orthogonal transformation triangularizing the left-hand pre-array to produce the right-hand post-array. The triangularization is carried out via Householder transformations exploiting the zero pattern of the pre-array. The relationship between the Kalman gain matrix K_i and G_i is given by

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

In order to exploit the invariant parts of the model to simplify the computation of U the results for the transformed state space U^*X are computed where U^* is the transformation that reduces the matrix pair (A, C) to lower observer Hessenberg form. That is, the matrix U^* is computed such that the compound matrix

$$\begin{bmatrix} CU^{*T} \\ U^*AU^{*T} \end{bmatrix}$$

is a lower trapezoidal matrix. Further the matrix B is transformed to U^*B. These transformations need only be computed once at the start of a series, and g13eb will, optionally, compute them. g13eb returns transformed matrices U^*AU^{*T}, U^*B, CU^{*T} and U^*AK_i, the Cholesky factor of the updated transformed state covariance matrix S^*_{i+1} (where $U^*P_{i+1|i}U^{*T} = S^*_{i+1}S^{*T}_{i+1}$) and the matrix $H_i^{1/2}$, valid for both transformed and original models, which is used in the computation of the likelihood for the model. Note that the covariance matrices Q_i and R_i can be time-varying.

4 References

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

5 Parameters

5.1 Compulsory Input Parameters

1: **transf – string**

Indicates whether to transform the input matrix pair (A, C) to lower observer Hessenberg form. The transformation will only be required on the first call to g13eb.

transf = 'T'

The matrices in arrays **a** and **c** are transformed to lower observer Hessenberg form and the matrices in **b** and **s** are transformed as described in Section 3.

transf = 'H'

The matrices in arrays **a**, **c** and **b** should be as returned from a previous call to g13eb with **transf** = 'T'.

Constraint: **transf** = 'T' or 'H'.

2: **a(lds,n) – double array**

lds, the first dimension of the array, must be at least **n**.

If **transf** = 'T', the state transition matrix, A.

If **transf** = 'H', the transformed matrix as returned by a previous call to g13eb with **transf** = 'T'.

3: **b(lds,l) – double array**

lds, the first dimension of the array, must be at least **n**.

If **transf** = 'T', the noise coefficient matrix B.

If **transf** = 'H', the transformed matrix as returned by a previous call to g13eb with **transf** = 'T'.

4: **stq – logical scalar**

If **stq** = **true**, the state noise covariance matrix Q_i is assumed to be the identity matrix. Otherwise the lower triangular Cholesky factor, $Q_i^{1/2}$, must be provided in **q**.

5: **q(ldq,∗) – double array**

The first dimension, **ldq**, of the array **q** must satisfy

if **stq** = **false**, **ldq** $\geq$ **l**;
ldq ≥ 1 otherwise.

The second dimension of the array must be at least **l** if **stq** = **false** and at least 1 if **stq** = **true**

If **stq** = **false**, **q** must contain the lower triangular Cholesky factor of the state noise covariance matrix, $Q_i^{1/2}$. Otherwise **q** is not referenced.

6: **c(ldm,n) – double array**

ldm, the first dimension of the array, must be at least **m**.

If **transf** = 'T', the measurement coefficient matrix, C.

If **transf** = 'H', the transformed matrix as returned by a previous call to g13eb with **transf** = 'T'.

7: **r(ldm,m) – double array**

ldm, the first dimension of the array, must be at least **m**.

The lower triangular Cholesky factor of the measurement noise covariance matrix, $R_i^{1/2}$.

8: **s(lds,n) – double array**

lds, the first dimension of the array, must be at least **n**.

If **transf** = 'T' the lower triangular Cholesky factor of the state covariance matrix, S_i.

If **transf** = 'H' the lower triangular Cholesky factor of the covariance matrix of the transformed state vector S_i^* as returned from a previous call to g13eb with **transf** = 'T'.

5.2 Optional Input Parameters

1: **n – int32 scalar**

Default: The dimension of the arrays **a**, **c**, **s**. (An error is raised if these dimensions are not equal.)

n, the size of the state vector.

Constraint: **n** ≥ 1.

2: **m – int32 scalar**

Default: The dimension of the arrays **r**, **k**, **h**. (An error is raised if these dimensions are not equal.)

m, the size of the observation vector.

Constraint: **m** ≥ 1.

3: **l – int32 scalar**

Default: The dimension of the array **b**.

l, the dimension of the state noise.

Constraint: **l** ≥ 1.

4: **tol – double scalar**

The tolerance used to test for the singularity of $H_i^{1/2}$. If $0.0 \leq$ **tol** $< m^2 \times$ ***machine precision***, then $m^2 \times$ ***machine precision*** is used instead. The inverse of the condition number of $H^{1/2}$ is estimated by a call to f07tg. If this estimate is less than **tol** then $H^{1/2}$ is assumed to be singular.

Suggested value: **tol** = 0.0.

Default: 0.0

Constraint: **tol** ≥ 0.0.

5.3 Input Parameters Omitted from the MATLAB Interface

lds, ldq, ldm, iwk, wk

5.4 Output Parameters

1: **a**(**lds**,**n**) – **double array**

If **transf** = 'T', the transformed matrix, U^*AU^{*T}, otherwise **a** is unchanged.

2: **b**(**lds**,**l**) – **double array**

If **transf** = 'T', the transformed matrix, U^*B, otherwise **b** is unchanged.

3: **c**(**ldm**,**n**) – **double array**

If **transf** = 'T', the transformed matrix, CU^{*T}, otherwise **c** is unchanged.

4: **s**(**lds**,**n**) – **double array**

The lower triangular Cholesky factor of the transformed state covariance matrix, S^*_{i+1}.

5: **k**(**lds**,**m**) – **double array**

The Kalman gain matrix for the transformed state vector premultiplied by the state transformed transition matrix, U^*AK_i.

6: **h**(**ldm**,**m**) – **double array**

The lower triangular matrix $H_i^{1/2}$.

7: **u**(**lds**,$*$) – **double array**

The first dimension of the array **u** must be at least **n**

The second dimension of the array must be at least **n** if **transf** = 'T', and at least 1 otherwise

If **transf** = 'T' the n by n transformation matrix U^*, otherwise **u** is not referenced.

8: **ifail** – **int32 scalar**

0 unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail = 1

On entry, **transf** $\neq$ 'T' or 'H',
or **n** < 1,
or **m** < 1,
or **l** < 1,
or **lds** $<$ **n**,
or **ldm** $<$ **m**,
or **stq** = **true** and **ldq** < 1,
or **stq** = **false** and **ldq** $<$ **l**,
or **tol** < 0.0.

ifail = 2

The matrix $H_i^{1/2}$ is singular.

7 Accuracy

The use of the square root algorithm improves the stability of the computations as compared with the direct coding of the Kalman filter. The accuracy will depend on the model.

8 Further Comments

For models with time-varying A, B and C, g13ea can be used.

If W_i and V_i are independent multivariate Normal variates then the log-likelihood for observations $i = 1, 2, \ldots, t$ is given by

$$l(\theta) = \kappa - \tfrac{1}{2}\sum_{i=1}^{t} ln(\det(H_i)) - \tfrac{1}{2}\sum_{i=1}^{t} \left(Y_i - C_i X_{i|i-1}\right)^{\mathrm{T}} H_i^{-1} \left(Y_i - C_i X_{i|i-1}\right)$$

where κ is a constant.

The Cholesky factors of the covariance matrices can be computed using f07fd.

Note that the model

$$X_{i+1} = AX_i + W_i, \quad \mathrm{Var}(W_i) = Q_i$$

$$Y_i = CX_i + V_i, \qquad \mathrm{Var}(V_i) = R_i$$

can be specified either with **b** set to the identity matrix and **stq** = **false** and the matrix $Q^{1/2}$ input in **q** or with **stq** = **true** and **b** set to $Q^{1/2}$.

The algorithm requires $\frac{1}{6}n^3 + n^2\left(\frac{3}{2}m + l\right) + 2nm^2 + \frac{2}{3}p^3$ operations and is backward stable (see Verhaegen and van Dooren 1986). The transformation to lower observer Hessenberg form requires $O\left((n+m)n^2\right)$ operations.

9 Example

```
transf = 'T';
a = [0.607, -0.033, 1, 0, 0, 0;
     0, 0.543, 0, 1, 0, 0;
     0, 0, 0, 0, 0, 0;
     0, 0, 0, 0, 0, 0;
     0, 0, 0, 0, 1, 0;
     0, 0, 0, 0, 0, 1];
b = [1, 0;
     0, 1;
     0.543, 0.125;
     0.134, 0.026;
     0, 0;
     0, 0];
stq = false;
q = [1.612, 0;
     0.347, 2.282];
c = [1, 0, 0, 0, 1, 0;
     0, 1, 0, 0, 0, 1];
r = [0, 0;
     0, 0];
s = [2.8648, 0, 0, 0, 0, 0;
     0.7191, 2.729, 0, 0, 0, 0;
     0.5169, 0.2194, 0.781, 0, 0, 0;
     0.1266, 0.0449, 0.1899, 0.0098, 0, 0;
     0, 0, 0, 0, 0, 0;
     0, 0, 0, 0, 0, 0];
[aOut, bOut, cOut, sOut, k, h, u, ifail] = g13eb(transf, a, b, stq, q, c,
r, s)
```

```
aOut =
    0.8035   -0.0165    0.7341         0         0         0
         0    0.7715    0.0051    0.7431         0         0
    0.0526    0.0096   -0.1245   -0.0103    0.2587         0
   -0.0004    0.0702    0.0062   -0.1339    0.0207   -0.2917
    0.1887    0.0325   -0.4466   -0.0336    0.9273    0.0072
    0.0157   -0.2154   -0.0536    0.4132    0.0072    0.9060
```

```
bOut =
   -0.7071         0
         0   -0.7071
   -0.3338   -0.1045
   -0.1252    0.1934
    0.8279    0.0858
    0.0152   -0.6787
cOut =
   -1.4142         0         0         0         0         0
         0   -1.4142         0         0         0         0
sOut =
    1.2666         0         0         0         0         0
    0.2794    1.6137         0         0         0         0
    0.4511    0.2352   -0.3882         0         0         0
    0.1037   -0.4422   -0.0912   -0.0000         0         0
   -1.4634   -0.1949    0.1105    0.0000   -0.0000         0
    0.2262    1.5486   -0.0302   -0.0000   -0.0000    0.0000
k =
   -0.5425   -0.0335
   -0.0283   -0.3956
    0.1459    0.0179
    0.0077    0.1215
    0.5230    0.0611
    0.0163   -0.3726
h =
    2.8648         0
    0.7191    2.7290
u =
   -0.7071         0         0         0   -0.7071         0
         0   -0.7071         0         0         0   -0.7071
    0.1893    0.0159   -0.9632         0   -0.1893   -0.0159
   -0.0013    0.2173    0.0067   -0.9516    0.0013   -0.2173
    0.6790    0.0516    0.2685    0.0236   -0.6790   -0.0516
    0.0563   -0.6707    0.0000   -0.3065   -0.0563    0.6707
ifail =
           0
```