NAG Fortran Library Routine Document

G13EBF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

G13EBF performs a combined measurement and time update of one iteration of the time-invariant Kalman filter using a square root covariance filter.

2 Specification

```
 SUBROUTINE G13EBF(TRANSF, N, M, L, A, LDS, B, STQ, Q, LDQ, C, LDM, R, S,
1                  K, H, U, TOL, IWK, WK, IFAIL)
 INTEGER           N, M, L, LDS, LDQ, LDM, IWK(M), IFAIL
 real              A(LDS,N), B(LDS,L), Q(LDQ,*), C(LDM,N), R(LDM,M),
1                  S(LDS,N), K(LDS,M), H(LDM,M), U(LDS,N), TOL,
2                  WK((N+M)*(N+M+L))
 LOGICAL           STQ
 CHARACTER*1       TRANSF
```

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

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

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

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

where D_iU_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 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 G13EBF will, optionally, compute them. G13EBF 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 Hesenberg 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

1: TRANSF – CHARACTER*1 *Input*

On entry: 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 G13EBF. If 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. If TRANSF = 'H' the matrices in arrays A, C and B should be as returned from a previous call to G13EBF with TRANSF='T'.

Constraint: TRANSF = 'T' or 'H'.

2: N – INTEGER *Input*

On entry: the size of the state vector, n.

Constraint: N $\geq$ 1.

3: M – INTEGER *Input*

On entry: the size of the observation vector, m.

Constraint: M $\geq$ 1.

4: L – INTEGER *Input*

On entry: the dimension of the state noise, l.

Constraint: L $\geq$ 1.

5: A(LDS,N) – ***real*** array *Input/Output*

On entry: if TRANSF = 'T', the state transition matrix, A. If TRANSF = 'H', the transformed matrix as returned by a previous call to G13EBF with TRANSF='T'.

On exit: if TRANSF = 'T', the transformed matrix, U^*AU^{*T}, otherwise A is unchanged.

6: LDS – INTEGER *Input*

On entry: the first dimension of the arrays A, B, S, K and U as declared in the (sub)program from which G13EBF is called.

Constraint: LDS $\geq$ N.

7: B(LDS,L) – ***real*** array *Input/Output*

On entry: if TRANSF = 'T', the noise coefficient matrix B. If TRANSF = 'H', the transformed matrix as returned by a previous call to G13EBF with TRANSF = 'T'.

On exit: if TRANSF = 'T', the transformed matrix, U^*B, otherwise B is unchanged.

8: STQ – LOGICAL *Input*

On entry: if STQ = .TRUE., then 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.

9: Q(LDQ,*) – ***real*** array *Input*

Note: the second dimension of the array Q must be at least at least L if STQ = .FALSE. and 1 if STQ = .TRUE..

On entry: 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.

10: LDQ – INTEGER *Input*

On entry: the first dimension of the array Q as declared in the (sub)program from which G13EBF is called.

Constraint: if STQ = .FALSE., LDQ $\geq$ L otherwise LDQ $\geq$ 1.

11: C(LDM,N) – ***real*** array *Input/Output*

On entry: if TRANSF = 'T', the measurement coefficient matrix, C. If TRANSF = 'H', the transformed matrix as returned by a previous call to G13EBF with TRANSF='T'.

On exit: if TRANSF = 'T', the transformed matrix, CU^{*T}, otherwise C is unchanged.

12: LDM – INTEGER *Input*

On entry: the first dimension of the arrays C, R and H as declared in the (sub)program from which G13EBF is called.

Constraint: LDM $\geq$ M.

13: R(LDM,M) – ***real*** array *Input*

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

14: S(LDS,N) – ***real*** array *Input/Output*

On entry: 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 G13EBF with TRANSF='T'.

On exit: the lower triangular Cholesky factor of the transformed state covariance matrix, S_{i+1}^*.

15: K(LDS,M) – ***real*** array *Output*

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

16: H(LDM,M) – ***real*** array *Output*

On exit: the lower triangular matrix $H_i^{1/2}$.

17: U(LDS,N) – ***real*** array *Output*

On exit: if TRANSF = 'T' the n by n transformation matrix U^*, otherwise U is not referenced.

18: TOL – ***real*** *Input*

On entry: 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 F07TGF (STRCON/DTRCON). If this estimate is less than TOL then $H^{1/2}$ is assumed to be singular.

Suggested value: TOL=0.0.

Constraint: TOL $\geq$ 0.0.

19: IWK(M) – INTEGER array *Workspace*

20: WK((N+M)*(N+M+L)) – ***real*** array *Workspace*

21: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. Users who are unfamiliar with this parameter should refer to Chapter P01 for details.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, for users not familiar with this parameter the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

6 Error Indicators and Warnings

If on entry IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

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, G13EAF can be used.

The initial estimate of the transformed state vector can be computed from the estimate of the original state vector $\hat{X}_{1|0}$, say, by premultiplying it by U^* as returned by G13EBF with TRANSF = 'T'; that is, $\hat{X}^*_{1|0} = U^*\hat{X}_{1|0}$. The estimate of the transformed state vector $\hat{X}^*_{i+1|i}$ can be computed from the previous value $\hat{X}^*_{i|i-1}$ by

$$\hat{X}^*_{i+1|i} = (U^*AU^{*T})\hat{X}^*_{i|i-1} + (U^*AK_i)r_i$$

where

$$r_i = Y_i - (CU^{*T})\hat{X}^*_{i|i-1}$$

are the independent one-step prediction residuals for both the transformed and original model. The estimate of the original state vector can be computed from the transformed state vector as $U^{*T}\hat{X}^*_{1+1|i}$. The required matrix-vector multiplications can be performed by F06PAF (SGEMV/DGEMV).

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

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

where κ is a constant.

The Cholesky factors of the covariance matrices can be computed using F07FDF (SPOTRF/DPOTRF).

Note that the model

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

$$Y_i = CX_i + V_i, \qquad \text{var}(V_i) = R_i$$

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

9 Example

The example program first inputs the number of updates to be computed and the problem sizes. The initial state vector and the Cholesky factor of the state covariance matrix are input followed by the model matrices $A, B, C, R^{1/2}$ and optionally $Q^{1/2}$ (the Cholesky factors of the covariance matrices being input). At the first update the matrices are transformed using the TRANSF = 'T' option and the inital value of the state vector is transformed. At each update the observed values are input and the residuals are computed and printed and the estimate of the transformed state vector, $\hat{U}^*X_{i|i-1}$, and the deviance are updated. The deviance is $-2\times$log-likelihood ignoring the constant. After the final update the estimate of the state vector is computed from the transformed state vector and the state covariance matrix is computed from S and these are printed along with the value of the deviance.

The data is for a two-dimensional time series to which a VARMA(1,1) has been fitted. For the specification of a VARMA model as a state space model see the G13 Chapter Introduction. The means of the two series are included as addtional states that do not change over time. The initial value of P, P_0, is the solution to

$$P_0 = AP_0A^T + BQB^T.$$

9.1 Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     G13EBF Example Program Text
*     Mark 17 Release. NAG Copyright 1995.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, MMAX, LMAX
      PARAMETER        (NMAX=6,MMAX=2,LMAX=2)
*     .. Local Scalars ..
      real             DEV, TOL
      INTEGER          I, IFAIL, INFO, ISTEP, J, L, LDM, LDQ, LDS, M, N,
     +                 NCALL
      LOGICAL          FULL, STQ
      CHARACTER        TRANSF
*     .. Local Arrays ..
      real             A(NMAX,NMAX), AX(NMAX), B(NMAX,LMAX),
     +                 C(MMAX,NMAX), H(MMAX,MMAX), K(NMAX,MMAX),
     +                 P(NMAX,NMAX), Q(LMAX,LMAX), R(MMAX,MMAX),
     +                 S(NMAX,NMAX), U(NMAX,NMAX), US(NMAX,NMAX),
     +                 WK((NMAX+MMAX)*(NMAX+MMAX+LMAX)), X(NMAX),
     +                 Y(MMAX)
      INTEGER          IWK(MMAX)
*     .. External Functions ..
      real             sdot
      EXTERNAL         sdot
*     .. External Subroutines ..
      EXTERNAL         scopy, sgemv, spotrf, ssyrk, strsv, F06QHF,
     +                 G13EBF
*     .. Intrinsic Functions ..
      INTRINSIC        LOG
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G13EBF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) NCALL, N, M, L, STQ, FULL
      IF (N.LE.NMAX .AND. M.LE.MMAX .AND. L.LE.LMAX) THEN
         LDS = NMAX
         LDM = MMAX
         LDQ = LMAX
         CALL F06QHF('G',N,N,0.0e0,0.0e0,S,LDS)
         READ (NIN,*) ((S(I,J),J=1,N),I=1,N)
         IF (FULL) THEN
            CALL spotrf('L',N,S,LDS,INFO)
            IF (INFO.GT.0) THEN
               WRITE (NOUT,*) ' S not positive definite'
               GO TO 100
            END IF
         END IF
         READ (NIN,*) (AX(I),I=1,N)
         READ (NIN,*) ((A(I,J),J=1,N),I=1,N)
         READ (NIN,*) ((B(I,J),J=1,L),I=1,N)
         READ (NIN,*) ((C(I,J),J=1,N),I=1,M)
         CALL F06QHF('G',M,M,0.0e0,0.0e0,R,LDM)
         READ (NIN,*) ((R(I,J),J=1,M),I=1,M)
         IF (FULL) THEN
            CALL spotrf('L',M,R,LDM,INFO)
            IF (INFO.GT.0) THEN
               WRITE (NOUT,*) ' R not positive definite'
               GO TO 100
            END IF
         END IF
         IF ( .NOT. STQ) THEN
            READ (NIN,*) ((Q(I,J),J=1,L),I=1,L)
            IF (FULL) THEN
               CALL spotrf('L',L,Q,LDQ,INFO)
               IF (INFO.GT.0) THEN
```

```
               WRITE (NOUT,*) ' Q not positive definite'
               GO TO 100
            END IF
         END IF
      END IF

      TOL = 0.0e0
      DEV = 0.0e0
      TRANSF = 'T'
      WRITE (NOUT,*)
      WRITE (NOUT,*) '         Residuals'
      WRITE (NOUT,*)
*
*      Loop through data
*
      DO 40 ISTEP = 1, NCALL
         IFAIL = 0

         IF (ISTEP.EQ.1) THEN
*
*      Make first call to G13EBF
*
            CALL G13EBF('T',N,M,L,A,LDS,B,STQ,Q,LDQ,C,LDM,R,S,K,H,U,
     +                  TOL,IWK,WK,IFAIL)
*
            CALL sgemv('N',N,N,1.0e0,U,LDS,AX,1,0.0e0,X,1)
         ELSE
*
            CALL G13EBF('H',N,M,L,A,LDS,B,STQ,Q,LDQ,C,LDM,R,S,K,H,U,
     +                  TOL,IWK,WK,IFAIL)
         END IF
*
         READ (NIN,*) (Y(I),I=1,M)
*
*     Perform time and measurement update
*
         CALL sgemv('N',M,N,-1.0e0,C,LDM,X,1,1.0e0,Y,1)
         WRITE (NOUT,99999) (Y(I),I=1,M)
         CALL sgemv('N',N,N,1.0e0,A,LDS,X,1,0.0e0,AX,1)
         CALL sgemv('N',N,M,1.0e0,K,LDS,Y,1,1.0e0,AX,1)
         CALL scopy(N,AX,1,X,1)
*
*     Update loglikelihood
*
         CALL strsv('L','N','N',M,H,LDM,Y,1)
         DEV = DEV + sdot(M,Y,1,Y,1)
         DO 20 I = 1, M
            DEV = DEV + 2.0e0*LOG(H(I,I))
   20    CONTINUE
         TRANSF = 'H'
   40 CONTINUE
*
*     Calculate back-transformed X
*
      CALL sgemv('T',N,N,1.0e0,U,LDS,AX,1,0.0e0,X,1)
      WRITE (NOUT,*)
      WRITE (NOUT,*) ' Final X(I+1:I) '
      WRITE (NOUT,*)
      WRITE (NOUT,99999) (X(J),J=1,N)
*
*     Compute back-transformed P from S
*
      DO 60 I = 1, N
         CALL sgemv('T',N-I+1,N,1.0e0,U(I,1),LDS,S(I,I),1,0.0e0,
     +              US(1,I),1)
   60 CONTINUE
      CALL ssyrk('L','N',N,N,1.0e0,US,LDS,0.0e0,P,LDS)
      WRITE (NOUT,*)
      WRITE (NOUT,*) ' Final Value of P'
      WRITE (NOUT,*)
      DO 80 I = 1, N
```

```
            WRITE (NOUT,99999) (P(I,J),J=1,I)
   80    CONTINUE
         WRITE (NOUT,*)
         WRITE (NOUT,99998) ' Deviance = ', DEV
      END IF
  100 CONTINUE
      STOP
*
99999 FORMAT (6F12.4)
99998 FORMAT (A,e13.4)
      END
```

9.2 Program Data

```
G13EBF Example Program Data

48 6 2 2 F F

 2.8648  0.0000  0.0000  0.0000  0.0000  0.0000
 0.7191  2.7290  0.0000  0.0000  0.0000  0.0000
 0.5169  0.2194  0.7810  0.0000  0.0000  0.0000
 0.1266  0.0449  0.1899  0.0098  0.0000  0.0000
 0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
 0.0000  0.0000  0.0000  0.0000  0.0000  0.0000

 0.000  0.000  0.000  0.000  4.404  7.991

 0.607 -0.033  1.000  0.000  0.000  0.000
 0.000  0.543  0.000  1.000  0.000  0.000
 0.000  0.000  0.000  0.000  0.000  0.000
 0.000  0.000  0.000  0.000  0.000  0.000
 0.000  0.000  0.000  0.000  1.000  0.000
 0.000  0.000  0.000  0.000  0.000  1.000


 1.000  0.000
 0.000  1.000
 0.543  0.125
 0.134  0.026
 0.000  0.000
 0.000  0.000

 1.000  0.000  0.000  0.000  1.000  0.000
 0.000  1.000  0.000  0.000  0.000  1.000

 0.000  0.000
 0.000  0.000

 1.612  0.000
 0.347  2.282

-1.490  7.340
-1.620  6.350
 5.200  6.960
 6.230  8.540
 6.210  6.620
 5.860  4.970
 4.090  4.550
 3.180  4.810
 2.620  4.750
 1.490  4.760
 1.170 10.880
 0.850 10.010
-0.350 11.620
 0.240 10.360
 2.440  6.400
 2.580  6.240
 2.040  7.930
 0.400  4.040
 2.260  3.730
```

```
 3.340  5.600
 5.090  5.350
 5.000  6.810
 4.780  8.270
 4.110  7.680
 3.450  6.650
 1.650  6.080
 1.290 10.250
 4.090  9.140
 6.320 17.750
 7.500 13.300
 3.890  9.630
 1.580  6.800
 5.210  4.080
 5.250  5.060
 4.930  4.940
 7.380  6.650
 5.870  7.940
 5.810 10.760
 9.680 11.890
 9.070  5.850
 7.290  9.010
 7.840  7.500
 7.550 10.020
 7.320 10.380
 7.970  8.150
 7.760  8.370
 7.000 10.730
 8.350 12.140
```

9.3 Program Results

```
 G13EBF Example Program Results

          Residuals

     -5.8940     -0.6510
     -1.4710     -1.0407
      5.1658      0.0447
     -1.3281      0.4580
      1.3653     -1.5066
     -0.2337     -2.4192
     -0.8685     -1.7065
     -0.4624     -1.1519
     -0.7510     -1.4218
     -1.3526     -1.3335
     -0.6707      4.8593
     -1.7389      0.4138
     -1.6376      2.7549
     -0.6137      0.5463
      0.9067     -2.8093
     -0.8255     -0.9355
     -0.7494      1.0247
     -2.2922     -3.8441
      1.8812     -1.7085
     -0.7112     -0.2849
      1.6747     -1.2400
     -0.6619      0.0609
      0.3271      1.0074
     -0.8165     -0.5325
     -0.2759     -1.0489
     -1.9383     -1.1186
     -0.3131      3.5855
      1.3726     -0.1289
      1.4153      8.9545
      0.3672     -0.4126
     -2.3659     -1.2823
     -1.0130     -1.7306
      3.2472     -3.0836
     -1.1501     -1.1623
```

```
     0.6855     -1.2751
     2.3432      0.2570
    -1.6892      0.3565
     1.3871      3.0138
     3.3840      2.1312
    -0.5118     -4.7670
     0.8569      2.3741
     0.9558     -1.2209
     0.6778      2.1993
     0.4304      1.1393
     1.4987     -1.2255
     0.5361      0.1237
     0.2649      2.4582
     2.0095      2.5623

 Final X(I+1:I)

     3.6698      2.5888     -0.0000      0.0000      4.4040      7.9910

 Final Value of P

     2.5985
     0.5594      5.3279
     1.4809      0.9697      0.9254
     0.3627      0.2135      0.2237      0.0542
    -0.0000     -0.0000     -0.0000     -0.0000      0.0000
    -0.0000     -0.0000     -0.0000     -0.0000      0.0000      0.0000

Deviance =    0.2229E+03
```