NAG Library Routine Document

G13EAF

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

G13EAF performs a combined measurement and time update of one iteration of the time-varying Kalman filter using a square root covariance filter.

2 Specification

```
 SUBROUTINE G13EAF(N, M, L, A, LDS, B, STQ, Q, LDQ, C, LDM, R, S, K, H,
1                  TOL, IWK, WK, IFAIL)
 INTEGER           N, M, L, LDS, LDQ, LDM, IWK(M), IFAIL
 double precision  A(LDS,N), B(LDS,L), Q(LDQ,*), C(LDM,N), R(LDM,M),
1                  S(LDS,N), K(LDS,M), H(LDM,M), TOL, WK((N+M)*(N+M+L))
 LOGICAL           STQ
```

3 Description

The Kalman filter arises from the state space model given by:

$$X_{i+1} = A_i X_i + B_i W_i, \quad \text{Var}(W_i) = Q_i$$

$$Y_i = C_i X_i + V_i, \qquad \text{Var}(V_i) = R_i$$

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

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

and

$$P_{i|i} = [I - K_i C_i]P_{i|i-1} \tag{2}$$

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

$$\hat{X}_{i+1|i} = A_i\hat{X}_{i|i} + D_i U_i \tag{3}$$

and

$$P_{i+1|i} = A_i P_{i|i} A_i^{\mathrm{T}} + B_i Q_i B_i^{\mathrm{T}} \tag{4}$$

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

$$\begin{pmatrix} R_i^{1/2} & C_i S_i & 0 \\ 0 & A_i S_i & B_i Q_i^{1/2} \end{pmatrix} U = \begin{pmatrix} H_i^{1/2} & 0 & 0 \\ G_i & S_{i+1} & 0 \end{pmatrix} \tag{5}$$

where U is an orthogonal transformation triangularizing the left-hand pre-array to produce the right-hand post-array. The relationship between the Kalman gain matrix, K_i, and G_i is given by

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

G13EAF requires the input of the lower triangular Cholesky factors of the noise covariance matrices $R_i^{1/2}$ and, optionally, $Q_i^{1/2}$ and the lower triangular Cholesky factor of the current state covariance matrix, S_i, and returns the product of the matrices A_i and K_i, A_iK_i, the Cholesky factor of the updated state covariance matrix S_{i+1} and the matrix $H_i^{1/2}$ used in the computation of the likelihood for the model.

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

1: N – INTEGER *Input*

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

Constraint: $N \geq 1$.

2: M – INTEGER *Input*

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

Constraint: $M \geq 1$.

3: L – INTEGER *Input*

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

Constraint: $L \geq 1$.

4: A(LDS,N) – ***double precision*** array *Input*

On entry: the state transition matrix, A_i.

5: LDS – INTEGER *Input*

On entry: the first dimension of the arrays A, B, S and K as declared in the (sub)program from which G13EAF is called.

Constraint: $LDS \geq N$.

6: B(LDS,L) – ***double precision*** array *Input*

On entry: the noise coefficient matrix B_i.

7: STQ – LOGICAL *Input*

On entry: 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.

8: Q(LDQ,$*$) – ***double precision*** array *Input*

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

9: LDQ – INTEGER *Input*

On entry: the first dimension of the array Q as declared in the (sub)program from which G13EAF is called.

Constraints:

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

10: C(LDM,N) – ***double precision*** array *Input*

On entry: the measurement coefficient matrix, C_i.

11: LDM – INTEGER *Input*

On entry: the first dimension of the arrays C, R and H as declared in the (sub)program from which G13EAF is called.

Constraint: LDM $\geq$ M.

12: R(LDM,M) – ***double precision*** array *Input*

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

13: S(LDS,N) – ***double precision*** array *Input/Output*

On entry: the lower triangular Cholesky factor of the state covariance matrix, S_i.

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

14: K(LDS,M) – ***double precision*** array *Output*

On exit: the Kalman gain matrix, K_i, premultiplied by the state transition matrix, A_i, A_iK_i.

15: H(LDM,M) – ***double precision*** array *Output*

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

16: TOL – ***double precision*** *Input*

On entry: the tolerance used to test for the singularity of $H_i^{1/2}$. If $0.0 \leq \text{TOL} < m^2 \times$ ***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 (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.

17: IWK(M) – INTEGER array *Workspace*
18: WK((N + M) $\times$ (N + M + L)) – ***double precision*** array *Workspace*

19: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction 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, if you are 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, 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-invariant A, B and C, G13EBF can be used.

The estimate of the state vector $\hat{X}_{i+1|i}$ can be computed from $\hat{X}_{i|i-1}$ by

$$\hat{X}_{i+1|i} = A_i\hat{X}_{i|i-1} + AK_ir_i$$

where

$$r_i = Y_i - C_i\hat{X}_{i|i-1}$$

are the independent one step prediction residuals. The required matrix-vector multiplications can be performed by F06PAF (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

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

where κ is a constant.

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

Note that the model

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

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

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 $\frac{7}{6}n^3 + n^2\left(\frac{5}{2}m + l\right) + n\left(\frac{1}{2}l^2 + m^2\right)$ operations and is backward stable (see Verhaegen and van Dooren (1986)).

9 Example

This example first inputs the number of updates to be computed and the problem sizes. The initial state vector and state covariance matrix are input followed by the model matrices A_i, B_i, C_i, R_i and optionally Q_i. The Cholesky factors of the covariance matrices can be computed if required. The model matrices can be input at each update or only once at the first step. At each update the observed values are input and the residuals are computed and printed and the estimate of the state vector, $\hat{X}_{i|i-1}$, and the deviance are updated. The deviance is $-2 \times$ log-likelihood ignoring the constant. After the final update the state covariance matrix is computed from S and printed along with final estimate of the state vector and 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 initial value of P, P_0, is the solution to

$$P_0 = A_1 P_0 A_1^{\mathrm{T}} + B_1 Q_1 B_1^{\mathrm{T}}.$$

For convenience, the mean of each series is input before the first update and subtracted from the observations before the measurement update is computed.

9.1 Program Text

```
*     G13EAF Example Program Text
*     Mark 17 Release. NAG Copyright 1995.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, MMAX, LMAX
      PARAMETER        (NMAX=4,MMAX=2,LMAX=2)
*     .. Local Scalars ..
      DOUBLE PRECISION DEV, TOL
      INTEGER          I, IFAIL, INFO, ISTEP, J, L, LDM, LDQ, LDS, M, N,
     +                 NCALL
      LOGICAL          CONST, FULL, STQ
*     .. Local Arrays ..
      DOUBLE PRECISION 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), WK((NMAX+MMAX)*(NMAX+MMAX+LMAX)),
     +                 X(NMAX), Y(MMAX), YMEAN(MMAX)
      INTEGER          IWK(MMAX)
*     .. External Functions ..
      DOUBLE PRECISION DDOT
      EXTERNAL         DDOT
*     .. External Subroutines ..
      EXTERNAL         DAXPY, DCOPY, DGEMV, DPOTRF, DTRMV, DTRSV, G13EAF
*     .. Intrinsic Functions ..
      INTRINSIC        LOG
*     .. Executable Statements ..
      WRITE (NOUT,*) 'G13EAF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) NCALL, N, M, L, STQ, FULL, CONST
      IF (N.LE.NMAX .AND. M.LE.MMAX .AND. L.LE.LMAX) THEN
```

```
         LDS = NMAX
         LDM = MMAX
         LDQ = LMAX
         READ (NIN,*) ((S(I,J),J=1,N),I=1,N)
         IF (FULL) THEN
            CALL DPOTRF('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,*) (X(I),I=1,N)
         READ (NIN,*) (YMEAN(I),I=1,M)
         TOL = 0.0D0
         DEV = 0.0D0
         WRITE (NOUT,*)
         WRITE (NOUT,*) '         Residuals'
         WRITE (NOUT,*)
*
*         Loop through data
*
         DO 40 ISTEP = 1, NCALL
            IF ( .NOT. CONST .OR. ISTEP.EQ.1) THEN
               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)
               READ (NIN,*) ((R(I,J),J=1,M),I=1,M)
               IF (FULL .AND. R(1,1).NE.0.0D0) THEN
                  CALL DPOTRF('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 DPOTRF('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
            END IF
            IFAIL = 0
*
            CALL G13EAF(N,M,L,A,LDS,B,STQ,Q,LDQ,C,LDM,R,S,K,H,TOL,IWK,
     +                  WK,IFAIL)
*
            READ (NIN,*) (Y(I),I=1,M)
            CALL DAXPY(M,-1.0D0,YMEAN,1,Y,1)
*
*           Perform time and measurement update
*
            CALL DGEMV('N',M,N,-1.0D0,C,LDM,X,1,1.0D0,Y,1)
            WRITE (NOUT,99999) (Y(I),I=1,M)
            CALL DGEMV('N',N,N,1.0D0,A,LDS,X,1,0.0D0,AX,1)
            CALL DGEMV('N',N,M,1.0D0,K,LDS,Y,1,1.0D0,AX,1)
            CALL DCOPY(N,AX,1,X,1)
*
*           Update loglikelihood
*
            CALL DTRSV('L','N','N',M,H,LDM,Y,1)
            DEV = DEV + DDOT(M,Y,1,Y,1)
            DO 20 I = 1, M
               DEV = DEV + 2.0D0*LOG(H(I,I))
   20       CONTINUE
   40    CONTINUE
*
*        Compute P from S
```

```
*
         DO 60 I = 1, N
            CALL DCOPY(I,S(I,1),LDS,P(1,I),1)
            CALL DTRMV('L','N','N',I,S,LDS,P(1,I),1)
            CALL DCOPY(I-1,P(1,I),1,P(I,1),LDS)
   60    CONTINUE
         WRITE (NOUT,*)
         WRITE (NOUT,*) ' Final X(I+1:I) '
         WRITE (NOUT,*)
         WRITE (NOUT,99999) (X(J),J=1,N)
         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
*
99999 FORMAT (6F12.4)
99998 FORMAT (A,E13.4)
      END
```

9.2 Program Data

```
G13EAF Example Program Data
48 4 2 2 F T T                  : NCALL, N, M, L, STQ, FULL, CONST
 8.2068  2.0599  1.4807  0.3627
 2.0599  7.9645  0.9703  0.2136
 1.4807  0.9703  0.9253  0.2236
 0.3627  0.2136  0.2236  0.0542 : End of S
 0.000  0.000  0.000  0.000     : X
 4.404  7.991                   : YMEAN
 0.607 -0.033  1.000  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     : End of A
 1.000  0.000
 0.000  1.000
 0.543  0.125
 0.134  0.026                   : End of B
 1.000  0.000  0.000  0.000
 0.000  1.000  0.000  0.000     : End of C
 0.000  0.000
 0.000  0.000                   : End of R
 2.598  0.560
 0.560  5.330                   : End of Q
-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                    : End of Y
```

9.3 Program Results

```
 G13EAF Example Program Results

          Residuals

     -5.8940     -0.6510
     -1.4710     -1.0407
      5.1658      0.0447
     -1.3280      0.4580
      1.3652     -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

 Final Value of P

     2.5980
     0.5600      5.3300
     1.4807      0.9703      0.9253
     0.3627      0.2136      0.2236      0.0542

 Deviance =    0.2229E+03
```