NAG 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,    &
                   K, H, U, TOL, IWK, WK, IFAIL)

INTEGER             N, M, L, LDS, LDQ, LDM, IWK(M), IFAIL
REAL (KIND=nag_wp) A(LDS,N), B(LDS,L), Q(LDQ,*), C(LDM,N), R(LDM,M),       &
                   S(LDS,N), K(LDS,M), H(LDM,M), U(LDS,*), TOL,            &
                   WK((N+M)*(N+M+L))
LOGICAL             STQ
CHARACTER(1)        TRANSF
```

3 Description

The Kalman filter arises from the state space model given by

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

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

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

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

where $K_i = P_{i|i}C^{\mathrm{T}}\big[CP_{i|i}C^{\mathrm{T}} + R_i\big]^{-1}$ 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 summarised 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 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 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 Arguments

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.

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 G13EBF with TRANSF = 'T'.

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

2: N – INTEGER *Input*

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

Constraint: N $\geq$ 1.

3: M – INTEGER *Input*

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

Constraint: M $\geq$ 1.

4: L – INTEGER *Input*

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

Constraint: L $\geq$ 1.

5: A(LDS, N) – REAL (KIND=nag_wp) 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 (KIND=nag_wp) 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., 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 (KIND=nag_wp) 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.

10: LDQ – INTEGER *Input*

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

Constraints:

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

11: C(LDM, N) – REAL (KIND=nag_wp) 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 (KIND=nag_wp) array *Input*

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

14: S(LDS, N) – REAL (KIND=nag_wp) 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 (KIND=nag_wp) 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 (KIND=nag_wp) array *Output*

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

17: U(LDS, $*$) – REAL (KIND=nag_wp) array *Output*

Note: the second dimension of the array U must be at least N if TRANSF = 'T', and at least 1 otherwise.

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

18: TOL – REAL (KIND=nag_wp) *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.

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

20: WK((N + M) × (N + M + L)) – REAL (KIND=nag_wp) array *Workspace*

21: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, −1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation for details.

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 argument, the recommended value is 0. **When the value −1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: IFAIL = 0 unless the routine detects an error or a warning has been flagged (see Section 6).

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.

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

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 Parallelism and Performance

G13EBF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

G13EBF makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 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

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

where

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

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 (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

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

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

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.

10 Example

This example 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 initial 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 additional states that do not change over time. The initial value of P, P_0, is the solution to

$$P_0 = AP_0A^{\mathrm{T}} + BQB^{\mathrm{T}}.$$

10.1 Program Text

```
    Program g13ebfe

!     G13EBF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: ddot, dgemv, dpotrf, dsyrk, dtrsv, g13ebf,       &
                             nag_wp, x04caf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Real (Kind=nag_wp), Parameter    :: one = 1.0_nag_wp
      Real (Kind=nag_wp), Parameter    :: zero = 0.0_nag_wp
      Integer, Parameter               :: inc1 = 1, nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: dev, tol
      Integer                          :: i, ifail, info, istep, l, ldm, ldq,  &
                                          lds, lwk, m, n, ncall, tdq
      Logical                          :: full, stq
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: a(:,:), ax(:), b(:,:), c(:,:),       &
                                          h(:,:), k(:,:), p(:,:), q(:,:),      &
                                          r(:,:), s(:,:), u(:,:), us(:,:),     &
                                          wk(:), x(:), y(:)
      Integer, Allocatable             :: iwk(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: log
!     .. Executable Statements ..
      Write (nout,*) 'G13EBF Example Program Results'
      Write (nout,*)

!     Skip heading in data file
      Read (nin,*)

!     Read in problem size
      Read (nin,*) n, m, l, stq

      lds = n
      If (.Not. stq) Then
        ldq = l
        tdq = l
      Else
        ldq = 1
        tdq = 1
      End If
      ldm = m
      lwk = (n+m)*(n+m+l)
      Allocate (a(lds,n),b(lds,l),q(ldq,tdq),c(ldm,n),r(ldm,m),s(lds,n),       &
        k(lds,m),h(ldm,m),u(lds,n),iwk(m),wk(lwk),ax(n),y(m),x(n),p(lds,n),    &
        us(lds,n))

!     Read in the state covariance matrix, S
      Read (nin,*)(s(i,1:n),i=1,n)
!     Read in flag indicating whether S is the full matrix, or its
!     Cholesky decomposition.
      Read (nin,*) full
!     If required, perform Cholesky decomposition on S
      If (full) Then
!       The NAG name equivalent of dpotrf is f07fdf
        Call dpotrf('L',n,s,lds,info)
        If (info>0) Then
          Write (nout,*) ' S not positive definite'
          Go To 100
        End If
      End If

!     Read in initial state vector
      Read (nin,*) ax(1:n)
```

```
!     Read in transition matrix, A
      Read (nin,*)(a(i,1:n),i=1,n)
!     Read in noise coefficient matrix, B
      Read (nin,*)(b(i,1:l),i=1,n)
!     Read in measurement coefficient matrix, C
      Read (nin,*)(c(i,1:n),i=1,m)

!     Read in measurement noise covariance matrix, R
      Read (nin,*)(r(i,1:m),i=1,m)
!     Read in flag indicating whether R is the full matrix, or its Cholesky
!     decomposition
      Read (nin,*) full
!     If required, perform Cholesky decomposition on R
      If (full) Then
!       The NAG name equivalent of dpotrf is f07fdf
        Call dpotrf('L',m,r,ldm,info)
        If (info>0) Then
          Write (nout,*) ' R not positive definite'
          Go To 100
        End If
      End If

!     Read in state noise matrix Q, if not assume to be identity matrix
      If (.Not. stq) Then
        Read (nin,*)(q(i,1:l),i=1,l)
!       Read in flag indicating whether Q is the full matrix, or
!       its Cholesky decomposition
        Read (nin,*) full
!       Perform Cholesky factorization on Q, if full matrix is supplied
        If (full) Then
!         The NAG name equivalent of dpotrf is f07fdf
          Call dpotrf('L',l,q,ldq,info)
          If (info>0) Then
            Write (nout,*) ' Q not positive definite'
            Go To 100
          End If
        End If
      End If

!     Read in control parameters
      Read (nin,*) ncall, tol

!     Display titles
      Write (nout,*) '         Residuals'
      Write (nout,*)

!     Loop through data
      dev = 0.0E0_nag_wp
      Do istep = 1, ncall

!       Read in observed values
        Read (nin,*) y(1:m)

        If (istep==1) Then
!         Make first call to G13EBF
          ifail = 0
          Call g13ebf('T',n,m,l,a,lds,b,stq,q,ldq,c,ldm,r,s,k,h,u,tol,iwk,wk,  &
            ifail)

!         The NAG name equivalent of dgemv is f06paf
          Call dgemv('N',n,n,one,u,lds,ax,inc1,zero,x,inc1)

        Else
!         Make remaining calls to G13EBF
          ifail = 0
          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

!       Perform time and measurement update x <= Ax + K(y-Cx)
!       The NAG name equivalent of dgemv is f06paf
```

```
        Call dgemv('N',m,n,-one,c,ldm,x,inc1,one,y,inc1)
        Call dgemv('N',n,n,one,a,lds,x,inc1,zero,ax,inc1)
        Call dgemv('N',n,m,one,k,lds,y,inc1,one,ax,inc1)
        x(1:n) = ax(1:n)

!       Display the residuals
        Write (nout,99999) y(1:m)

!       Update log-likelihood
!       The NAG name equivalent of dtrsv is f06pjf
        Call dtrsv('L','N','N',m,h,ldm,y,inc1)
!       The NAG name equivalent of ddot is f06eaf
        dev = dev + ddot(m,y,1,y,1)
        Do i = 1, m
          dev = dev + 2.0_nag_wp*log(h(i,i))
        End Do
      End Do

!     Calculate back-transformed x <- U^T x
!     The NAG name equivalent of dgemv is f06paf
      Call dgemv('T',n,n,one,u,lds,ax,inc1,zero,x,inc1)

!     Compute back-transformed P from S
      Do i = 1, n
        Call dgemv('T',n-i+1,n,one,u(i,1),lds,s(i,i),inc1,zero,us(1,i),inc1)
      End Do
!     The NAG name equivalent of dsyrk is f06ypf
      Call dsyrk('L','N',n,n,one,us,lds,zero,p,lds)

!     Display final results
      Write (nout,*)
      Write (nout,*) ' Final X(I+1:I) '
      Write (nout,99999) x(1:n)
      Write (nout,*)
      Flush (nout)
      ifail = 0
      Call x04caf('Lower','N',n,n,p,lds,'Final Value of P',ifail)
      Write (nout,99998) ' Deviance = ', dev

100   Continue

99999 Format (6F12.4)
99998 Format (A,E13.4)
    End Program g13ebfe
```

10.2 Program Data

```
G13EBF Example Program Data
 6 2 2 F                                                :: N,M,L,STQ
 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     :: End of S
 F                                                      :: FULL flag for S
 0.000  0.000  0.000  0.000  4.404  7.991           :: AX
 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           :: End of A
 1.000  0.000
 0.000  1.000
 0.543  0.125
 0.134  0.026
 0.000  0.000
 0.000  0.000                                          :: End of B
 1.000  0.000  0.000  0.000  1.000  0.000
```

```
 0.000  1.000  0.000  0.000  0.000  1.000          :: End of C
 0.000  0.000
 0.000  0.000                                      :: End of R
 F                                                 :: FULL flag for R
 1.612  0.000
 0.347  2.282                                      :: End of Q
 F                                                 :: FULL flag for Q
 48 0.0                                            :: NCALL,TOL
-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
```

10.3 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
              1           2           3           4           5
1    2.5985E+00
2    5.5936E-01   5.3279E+00
3    1.4809E+00   9.6973E-01   9.2536E-01
4    3.6275E-01   2.1348E-01   2.2366E-01   5.4159E-02
5   -4.0547E-16  -8.7283E-17  -2.3108E-16  -5.6603E-17   9.6581E-32
6    4.4742E-17   9.6312E-18   2.5499E-17   6.2458E-18  -9.3231E-33

              6
1
2
3
4
5
6    1.3378E-32
Deviance =    0.2229E+03
```