

The State Space Approach to Modelling Dynamic Processes:

Applications in Neuroscience and Social Sciences

by Ho, Moon-ho; Shumway, Robert; and Ombao, Hernando

Supplementary Materials and a Matlab script accompanying chapter in Walls, T. A., & Schafer, J. L. (2006). Models for Intensive Longitudinal Data.

1. Introduction This supplementary note discusses the maximum likelihood estimation of state space models using Expectation-Maximization (EM) algorithm and bootstrap procedure for statistical inference. A Matlab program script implementing the Kalman filter, Kalman smoother and EM algorithm (used for Case Study 2) is available from the book's website ('MILDChp7.m').

2. Gaussian State Space Models A linear Gaussian state space model is characterized by an unobserved series of vectors $\theta_1, \dots, \theta_n$ (called *states*), that are associated with a series of observations y_1, \dots, y_n . The relation between the states and the observations is specified through the observation and the state equations. In the longitudinal settings with N subjects, the observation equation and state equations are defined as

$$y_{t,j} = \Gamma x_{t,j} + A\theta_{t,j} + e_{t,j}, \quad e_{t,j} \sim N_p(0, R), \quad (1)$$

$$\theta_{t,j} = \Phi\theta_{t-1,j} + w_{t,j}, \quad w_{t,j} \sim N_k(0, Q), \quad t = 1, \dots, n; j = 1, \dots, N \quad (2)$$

where $y_{t,j}$ is a $p \times 1$ vector of observations, $x_{t,j}$ is a $q \times 1$ input vector of fixed functions

which can be used to incorporate fixed trends or treatment effects. Deterministic (e.g., polynomial or sinusoidal) and stochastic (e.g., cubic spline) trends can be included in a straightforward manner under state space models (see case study 2 for an illustration) so to accommodate the nonstationarity in the time series data. Γ is a $p \times q$ regression coefficient matrix relating the input vector $x_{t,j}$ to the output time series $y_{t,j}$. The equation (1) is called the observation equation, which has the structure of a linear regression relating the state vector to the observed time series. Equation (2) is called the state equation, which describes the dynamics of the states, denoted by a $k \times 1$ vector $\theta_{t,j}$, in terms of a first-order Markov process. The relation between the state vector to the observed values is characterized by the matrix A of size $p \times k$. The dependence of the current state on the past is determined by the transition matrix, Φ .

3.1 Maximum Likelihood Estimation We now present a technical discussion on estimating the unknown parameters, collectively denoted as $\Theta = \{\Gamma, A, \Phi, R, Q\}$, and the unknown states, x_t , in the state space models. In this section, the following notation will be used. The conditional expectation of the state vector at time t given the data $Y_s = \{y_1, \dots, y_s\}$, from time 1 up to time s , is denoted as $x_t^s = E(x_t | y_1, \dots, y_s)$. The error between the actual value of the state vector at time t and its conditional expectation given Y_s is $(x_t - x_t^s)$. The covariance matrix between the errors at time t and u is expressed as $P_{t,u}^s = E\{(x_t - x_t^s)(x_u - x_u^s)'\}$. Moreover, given the observations, Y_1, \dots, Y_n , the problem of estimating x_t based on

observations *before* time t is called forecasting or prediction in time series literature. The problem of estimating x_t based on observations up to time t ($t \leq n$) is called filtering. The problem of estimating x_t based on all the T -observations is called smoothing. If Θ are known, we can estimate the unknown state vector by the Kalman filter procedure. The Kalman filter procedure consists of the two steps, namely, prediction and filtering:

1. Prediction: At the beginning of time t , based on all the available information up to time $(t - 1)$ (i.e., only observations y_1, \dots, y_{t-1} have been observed, but not y_t), we can obtain an optimal estimator (in the sense that it minimizes the mean square error) for the state vector, x_t^{t-1} , and the corresponding error variance-covariance matrix, P_t^{t-1} can be computed by (A.1) and (A.2) in the Appendix.
2. Filtering: Once y_t is realized at the end of time t , we can compute the prediction error (also known as innovation), $e_t = y_t - y_t^{t-1} = y_t - \Gamma z_t - A x_t^{t-1}$, with its covariance equal to $\Sigma_t = A P_t^{t-1} A' + R$. This innovation contains new information for x_t beyond that contained in x_t^{t-1} . In other words, we can make more accurate inference on x_t based on the information at time t by (A.3) and (A.4). The filtered value for the state, x_t^t , is a combination of predicted value, x_t^{t-1} and the prediction error, e_t , weighted by Kalman gain, K_t , defined in (A.5). The Kalman gain, K_t , determines the weight assigned to new information about x_t contained in the innovation (e_t). The estimator for the unknown states obtained from the Kalman filter is the best *linear* unbiased estimator for every distribution of the noises, e_t and w_t , and initial

distribution of x_0 . Best in the sense that the error covariance of any other linear estimator exceeds the error covariance of the Kalman filter estimator by a positive semi-definite matrix. It is the best among all linear and nonlinear estimators if the noises, e_t and w_t , and initial distribution of x_0 are assumed to be normal (Harvey, 1989). The values for a_0 and P_0 are usually unknown in empirical studies. It can be shown, however, that the Kalman filter estimates become independent of the estimates of a_0 and P_0 after sufficient time points. Moreover, the Kalman filter estimator is asymptotically unbiased in the limit (as $n \rightarrow \infty$, see Otter, 1985).

Given the Kalman filter estimates for the unknown states, we can compute the log-likelihood of the model based on the prediction error decomposition given by Schweppe (1965), and estimate the unknown parameters by the maximum likelihood method. The log likelihood for the state space model (excluding the constant) is given by:

$$-2 \log L(Y_n; \Theta) \propto \sum_{t=1}^n \log |\Sigma_t| + \sum_{t=1}^n (y_t - \Gamma z_t - Ax_t^{t-1})' \Sigma_t^{-1} (y_t - \Gamma z_t - Ax_t^{t-1}), \quad (3)$$

where $\Sigma_t = Cov(e_t) = AP_t^{t-1}A' + R$.

The unknown parameter estimates can be obtained by maximizing this likelihood function via the Newton-Raphson type algorithm. This type of algorithm has some disadvantages. First, the corrections in the successive iterations generally involve calculating the inverse of the matrix of second order partial derivatives which can be quite large if there are a large number of parameters. Moreover, the suc-

cessive steps involved in a Newton-Raphson may not necessarily increase the size of the likelihood or one may encounter extremely large steps which actually decrease the likelihood. In this chapter, however, we will focus on how to estimate the parameters in state space model by EM (Expectation-Maximization) algorithm instead (Shumway & Stoffer, 1982). The unattractive features in the Newton-Raphson algorithm can be circumvented using EM. The EM steps always increase the likelihood and guarantee convergence to a stationary point for an exponential family (Wu, 1981). The EM equations usually take on a simple heuristically appealing form in contrast to the highly nonlinear appearance of the Newton-Raphson or scoring corrections. Since the matrix of second partial derivatives is not computed in the EM algorithm, standard errors of the parameter estimates cannot be obtained directly. However, these partial derivatives can be approximated by perturbing the likelihood function in the neighborhood of the maximum. Alternatively, the standard errors may be obtained by resampling methods such as bootstrap. Since the EM algorithm may converge slowly in the latter stages of the iterative procedure, one may need to consider switching to another algorithm at this stage. The EM-algorithm for state space model requires the computation of the Kalman filter and the Kalman smoother for x_t . The Kalman smoother is another estimator for the state (x_t) which is based on *all* the observations at any time t and we denote it as $x_t^n = E(x_t|y_1, \dots, y_n)$ with the corresponding mean squared covariance estimator as $P_t^n = E\{(x_t - x_t^n)(x_t - x_t^n)'|y_1, \dots, y_n\}$. The recursion formulae to compute these

quantities are summarized in the Appendix from (A.6) to (A.10).

We will now describe the details of the use of EM algorithm for estimating unknown parameters in the state space models. If we can observe both the states, $X_n = \{x_1, \dots, x_n\}$ and the observations $Y_n = \{y_1, \dots, y_n\}$, we would consider $\{X_n, Y_n\}$ as the *complete data*. Under the Gaussian assumption, the complete data likelihood can be written as:

$$-2 \log L(Y_n, X_n; \Theta) \propto n \log |R| + \sum_{t=1}^n (y_t - \Gamma z_t - Ax_t)' R^{-1} (y_t - \Gamma z_t - Ax_t) + n \log |Q| + \sum_{t=1}^n (x_t - \Phi x_{t-1})' Q^{-1} (x_t - \Phi x_{t-1}) + \log |\Sigma_0| + (x_0 - \mu_0)' \Sigma_0^{-1} (x_0 - \mu_0). \quad (4)$$

To implement the EM algorithm, we write, at iteration j , $j = 1, 2, \dots$

$$H(\Theta | \Theta^{(j-1)}) = E[-2 \log L(Y_n, X_n; \Theta) | Y_n, \Theta^{(j-1)}] \quad (5)$$

where $\Theta^{(j-1)}$ refers to the value of the parameters at the preceding iteration. Calculation of (5) is the *expectation step*. Given the current value of the parameters, $\Theta^{(j-1)}$, the desired conditional expectation can be obtained using Kalman smoothers. This yields

$$\begin{aligned} H(\Theta | \Theta^{(j-1)}) &= n \log |R| + tr \{ R^{-1} \sum_{t=1}^n [(y_t - \Gamma z_t - Ax_t^n)(y_t - \Gamma z_t - Ax_t^n)' \\ &+ AP_t^n A'] \} + n \log |Q| + tr \{ Q^{-1} [S_{11} - S_{10} \Phi' - \Phi S_{10} + \Phi S_{00} \Phi'] \} \\ &+ \log |\Sigma_0| + tr \{ \Sigma_0^{-1} [(x_0^n - \mu_0)(x_0^n - \mu_0)' + P_0^n] \}, \text{ where} \end{aligned} \quad (6)$$

$$S_{00} = \sum_{t=1}^n \left(x_{t-1}^n (x_{t-1}^n)' + P_{t-1}^n \right), \quad (7)$$

$$S_{10} = \sum_{t=1}^n \left(x_t^n (x_{t-1}^n)' + P_{t,t-1}^n \right), \quad (8)$$

$$S_{11} = \sum_{t=1}^n \left(x_t^n (x_t^n)' + P_t^n \right), \quad (9)$$

and $tr(\cdot)$ is the trace operator. The components $x_{t-1}^n, x_t^n, P_{t-1}^n, P_t^n$ and $P_{t,t-1}^n$ of the above equations can be computed as in Properties (A2)-(A3) in the Appendix. Notice in (7) to (9), the smoothers are calculated under the present value of the parameters $\Theta^{(j-1)}$.

Minimizing (6) with respect to the unknown parameter set, $\Theta = \{\Gamma, A, \Phi, R, Q\}$, constitutes the *maximization step*. At iteration j , for the parameters Φ and Q , this yields the updated estimates, $\Phi^{(j)} = S_{10} S_{00}^{-1}$ (M-step Φ) and $Q^{(j)} = \frac{1}{n} (S_{11} - S_{10} S_{00}^{-1} S_{01})$ (M-Step Q). To develop estimators for Γ , A and R from the first term of (6), it will reduce to minimizing: $\sum_{t=1}^n E\{(y_t - \Gamma z_t - Ax_t)(y_t - \Gamma z_t - Ax_t)' | Y_n, \Theta^{(j-1)}\}$ over Γ and A and then evaluating R . Performing the first operation, we obtain

$$\begin{pmatrix} \Gamma^{(j)} & A^{(j)} \end{pmatrix} = \begin{pmatrix} \sum_{t=1}^n y_t z_t' & \sum_{t=1}^n y_t x_t^{n'} \end{pmatrix} \begin{pmatrix} \sum_t z_t z_t' & \sum_t z_t x_t^{n'} \\ \sum_t z_t x_t^n & S_{11} \end{pmatrix}^{-1} \quad (10)$$

with S_{11} given previously in (9). Finally, we can obtain estimator for R as: $R^{(j)} = \frac{1}{n} \sum_{t=1}^n \left\{ (y_t - \Gamma^{(j-1)} z_t - A^{(j-1)} x_t^n)(y_t - \Gamma^{(j-1)} z_t - A^{(j-1)} x_t^n)' + A^{(j-1)} P_t^n A^{(j-1)'} \right\}$ (M-step R). Although the parameters μ_0 and Σ_0 are omitted from the above argument, it is easy to see from the complete data log-likelihood (4) that one could take either

$\widehat{\mu}_0 = x_t^n$ or $\widehat{\Sigma}_0 = (x_0^n - \mu_0)(x^n - \mu_0)' + P_0^n$, but not both, in the procedures given above. The overall procedure can be regarded as simply alternating between the Kalman filtering and smoothing recursion and the multivariate normal maximum likelihood estimators, as given by M-step Φ to M-step R . Convergence results for the EM-algorithm under general conditions can be found in Wu (1983). We summarize the iterative procedure as follows:

1. Initialize the procedure by selecting starting values for the parameters $\Theta^{(0)}$, and fix either μ_0 or Σ_0 or both. On iteration j , ($j = 1, 2, \dots$)
2. Compute the incomplete-data likelihood, $-2 \log L(Y_n; \Theta)$ in (3).
3. Perform the E-Step. Use Properties (A1)-(A3) to obtain the smoothed values x_t^n, P_t^n , and $P_{t,t-1}^n$, for $t = 1, \dots, n$, using the parameters $\Theta^{(j-1)}$. Use the smoothed values to calculate S_{11}, S_{10}, S_{00} given in (7)–(9).
4. Perform the M-Step. Update the estimates, Φ, Q, Γ, A and R using (M-step Φ) to (M-step R), to obtain $\Theta^{(j)}$.
5. Repeat Steps 2 to 4 to convergence.

3.2 Bootstrap Estimation of Standard Errors The EM algorithm may not provide a version of the information matrix that can be easily computed. This leads to a great challenge to find the variances and covariances of the estimated parameters. Computation of the information matrix via recursions is possible as in Harvey (1991) or Cavanaugh and Shumway (1986). Versions of the information

matrix, obtained from outputs arising naturally in the EM algorithm, such as in Meng and Rubin (1991) or Oakes (1999), are either hard to compute, as in the former, or will involve relatively untractable derivatives in the latter. A compromise that is easy to apply and will be robust towards distributional assumptions is the bootstrap, as derived in Stoffer and Wall (1991) and we focus on their methodology here.

Suppose we obtain maximum likelihood estimators for $\Theta = \{\Gamma A, \Phi, R, Q\}$ as $\hat{\Theta} = \{\hat{\Gamma}, \hat{A}, \hat{\Phi}, \hat{R}, \hat{Q}\}$. Define the residuals from these estimators as $\hat{v}_t = y_t - \hat{\Gamma}z_t - \hat{A}\hat{x}_t^{t-1}$ and construct scaled residuals of the form $\hat{\epsilon}_t = \hat{\Sigma}_t^{-1/2}\hat{v}_t$, where a hat over a quantity indicates that it has been evaluated at the maximum likelihood estimator $\hat{\Theta}$. Then, draw a random sample, say, $\hat{\epsilon}_t^*, t = 1, \dots, n$ with replacement from the scaled residuals. Rescale the residuals, i.e., $v_t^* = \hat{\Sigma}_t^{1/2}\hat{\epsilon}_t^*$ to obtain residuals with the correct time varying covariance matrix. To reconstruct the data, note that $y_t^* = \hat{\Gamma}z_t + \hat{A}\hat{x}_t^{t-1} + v_t^*$ and compute the values \hat{x}_t^{t-1} , using Property (A1) and the maximum likelihood estimator Θ with (A.3) in the Appendix replaced by $\hat{x}_t^t = \hat{x}_t^{t-1} + \hat{K}_t\hat{v}_t^*$. Using the reconstructed bootstrap sample to compute maximum likelihood estimators $\Theta^* = \{\Gamma^*, A^*, \Phi^*, R^*, Q^*\}$, using the procedure described in section 3.1. The above bootstrap steps are repeated a large number, B , times, obtaining $\{\hat{\Theta}_b^*, b = 1, \dots, B\}$. The finite sample distributions of $(\hat{\Theta} - \Theta)$ are approximated by the distributions of $(\hat{\Theta}_b^* - \hat{\Theta})$, $b = 1, \dots, B$, for example, the estimated variance of the estimated parameter $\hat{\theta}_i$, can be computed as $\hat{\sigma}_{\hat{\theta}}^2 = \frac{1}{B-1} \sum_{b=1}^B (\hat{\theta}_b^* - \bar{\theta}^*)^2$, where $\bar{\theta}^*$

denotes the mean of the bootstrap estimators.

We applied the EM algorithm to estimate the models reported in two case studies. The recursive formulae for estimating the parameters in case study 1 are reported here for readers' reference. We repeat the state space representation for the model we reported in the book chapter here.

Observation Equations:

$$\begin{pmatrix} y_{1t} \\ y_{2t} \\ y_{3t} \end{pmatrix} = \begin{pmatrix} \gamma_1 & 0 & 0 \\ 0 & \gamma_2 & 0 \\ 0 & 0 & \gamma_3 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + \begin{pmatrix} a_{1t} & 0 & 0 \\ 0 & a_{2t} & 0 \\ 0 & 0 & a_{3t} \end{pmatrix} \begin{pmatrix} \beta_{1t} \\ \beta_{2t} \\ \beta_{3t} \end{pmatrix} + \begin{pmatrix} e_{1t} \\ e_{2t} \\ e_{3t} \end{pmatrix}.$$

or compactly as: $Y_t = \Gamma + A_t B_t + E_t$.

State Equations:

$$\begin{pmatrix} \beta_{1t} \\ \beta_{2t} \\ \beta_{3t} \end{pmatrix} = \begin{pmatrix} \lambda_{1.1}a_{1,t-1} & \lambda_{1.2}a_{2,t-1} & \lambda_{1.3}a_{3,t-1} \\ \lambda_{2.1}a_{1,t-1} & \lambda_{2.2}a_{2,t-1} & \lambda_{2.3}a_{3,t-1} \\ \lambda_{3.1}a_{1,t-1} & \lambda_{3.2}a_{2,t-1} & \lambda_{3.3}a_{3,t-1} \end{pmatrix} \begin{pmatrix} \beta_{1,t-1} \\ \beta_{2,t-1} \\ \beta_{3,t-1} \end{pmatrix} + \begin{pmatrix} w_{1t} \\ w_{2t} \\ w_{3t} \end{pmatrix}.$$

or compactly as: $B_t = (\Lambda A_{t-1})B_{t-1} + W_t$, where Λ is a 3×3 matrix containing the 9 $\lambda_{i,j}$ coefficients.

Notice that in this example, the coefficient matrix Γ in (1) contains the three baseline intensity values, $\gamma_1, \gamma_2, \gamma_3$ on its diagonal, and the coefficient matrix A contains the convoluted hemodynamic response function which is time-varying but

is assumed to be known. For convenience, we write $\Gamma_v = (\gamma_1, \gamma_2, \gamma_3)'$. The coefficient matrix Φ in (2) is also time-varying and can be written as, $\Phi_t = \Lambda A_{t-1}$. Assuming the both the measurement errors ($R = cov(e_{1t}, e_{2t}, e_{3t})'$) and process errors ($Q = cov(w_{1t}, w_{2t}, w_{3t})'$) to be normal, we apply the EM-algorithm to obtain the unknown parameters: Γ_v, Λ, R and Q and the updated estimates are given by:

$$\begin{aligned}\Gamma_v^{(j)} &= \frac{1}{n} \sum_{t=1}^n (Y_t - A_t B_t^n) \\ \Lambda^{(j)} &= \left(\sum_{t=1}^n S_{10,t} A'_{t-1} \right) \left(\sum_{t=1}^n A_{t-1} S_{00,t} A'_{t-1} \right)^{-1} \\ R^{(j)} &= \frac{1}{n} \sum_{t=1}^n [(Y_t - \Gamma_v^{(j-1)} - A_t B_t^n)(Y_t - \Gamma_v^{(j-1)} - A_t B_t^n)' + A_t P_t^n A_t'] \\ Q^{(j)} &= \frac{1}{n} \sum_{t=1}^n (S_{11,t} - \Lambda^{(j-1)} A_{t-1} S'_{10,t} - S_{10,t} A'_{t-1} \Lambda_t^{(j-1)'} + \Lambda^{(j-1)} A_{t-1} S_{00,t} A'_{t-1} \Lambda^{(j-1)'})\end{aligned}$$

4. Software Package Used in this Chapter The analyses for both case studies in this chapter were performed by Matlab (Mathworks, 2004). The implementation of Kalman filter, Kalman smoother, and E-M algorithm for obtaining maximum likelihood estimate of the parameters in the state space model can be found in the Matlab script ‘MILDChp7.m’, which is listed in Appendix 2. It can also be downloaded from the book’s website. The script was used for the analysis reported in Case Study 2.

Appendix 1

For the properties below, we use the notations: $x_t^s = E\{x_t | Y_s\}$, where $Y_s =$

$\{y_1, \dots, y_s\}$ denotes the vectors up to time s . For $s = t - 1$, the expectations is a forecast whereas for $s = t$, the expectation is the Kalman filtered value. For $s = n$, the expectation is conditional on the entire data and is the Kalman smoother. The conditional covariances $P_t^s = E\{(x_t - x_t^s)(x_t - x_t^s)' | Y_s\}$, and $P_{tu}^s = E\{(x_t - x_t^s)(x_u - x_u^s)' | Y_s\}$ are interpreted in the same way. We summarize the equations for the Kalman filters, smoothers and their covariances in the three properties below.

Property A1: The Kalman Filter

For the state space model specified in (1) and (2) with initial conditions $x_0^0 = \mu_0$ with $P_0^0 = \Sigma_0$, for $t = 1, \dots, n$

$$x_t^{t-1} = \Phi_t x_{t-1}^{t-1} \tag{A.1}$$

$$P_t^{(t-1)} = \Phi_t P_{t-1}^{t-1} \Phi_t' + Q_t, \text{ with} \tag{A.2}$$

$$x_t^t = x_t^{t-1} + K_t(y_t - \Gamma z_t - Ax_{t-1}^{t-1}) \text{ and} \tag{A.3}$$

$$P_t^t = (I - K_t A) P_{t-1}^{t-1} \tag{A.4}$$

where the Kalman gain is

$$K_t = P_t^{t-1} A' [A P_t^{t-1} A' + R_t]^{-1}. \tag{A.5}$$

Property A2: The Kalman Smoother

For the state space model specified in (1) and (2) with initial conditions $x_n^n = \mu_0$

with P_n^n via Property A1, for $t = n, n-1, \dots, 1$,

$$x_{t-1}^n = x_{t-1}^{t-1} + J_{t-1}(x_t^n - x_t^{t-1}) \quad (\text{A.6})$$

$$P_{t-1}^n = P_{t-1}^{t-1} + J_{t-1}(P_t^n - T_t^{t-1})J'_{t-1}, \text{ where} \quad (\text{A.7})$$

$$J_{t-1} = P_{t-1}^{t-1}\Phi'_t[P_t^{t-1}]^{-1}. \quad (\text{A.8})$$

Property A3: The Lag-One Covariance Smoother

For the state space model specified in (1) and (2) with $K_t, J_t, t = 1, \dots, n$ obtained from Properties A1 and A2, with initial condition

$$P_{n,n-1}^n = (I - K_n A_n)\Phi_n P_{n-1}^{n-1}, \quad (\text{A.9})$$

for $t = n, n-1, \dots, 2$,

$$P_{t-1,t-2}^n = P_{t-1}^{t-1}J'_{t-2} + J_{t-1}(P_{t,t-1} - \Phi_t P_{t-1}^{t-1})J'_{t-2} \quad (\text{A.10})$$

Appendix 2. Matlab Script (MILDChp7.m)

```
%%%%% Maximum Likelihood Estimation for State-Space Model %%%%%  
  
% Equation Numbers are from Technical Appendix  
  
% The matlab program implements maximum likelihood estimation of state-space model  
% for Case Study 2 in the book chapter.  
  
% Read in data file (in excel format)  
  
% mytimeseries is a n x nts matrix where  
  
% n=length of time series, and  
  
% nts=number of time series  
  
mytimeseries=xlsread(' [Insert file name] ');  
  
% Find the dimension of the data  
  
[n,nts]=size(mytimeseries);  
  
% Set input parameters  
  
niter=50; % Number of iterations  
  
np=4; % dimension of state vector  
  
nq=2; % dimension of observation vector  
  
r=18 % no. of replicated time series  
  
np2=np*np;  
  
% initial variance for the state vector  
  
S0=.02*eye(np,np);  
  
% Matrix Phi in the state eqt. for Case Study 2
```

```

Phi=[2 0 -1 0; 0 2 0 -1;1 0 0 0; 0 1 0 0];

% initial error variances in the state eqt.
Qf=.005*eye(np,np);

% Matrix A in the obs. eqt. for Case Study 2
Af=[1 0 0 0;0 1 0 0];

% Initial error variance in the obs. eqt.
Rf=[.05 0; 0 .05];

% Initialization of matrices to save intermediate results
B=zeros(nq,nq); C11=zeros(nq,nq); R22_1=zeros(nq,nq);
x0=zeros(np,n+1); P0=zeros(np,np); P00=zeros(np,np);
Pdiff=zeros(np,np); p0=zeros(np2,n+1); x1=zeros(np,n+1);
P1=zeros(np,np); p1=zeros(np2,n+1); P10=zeros(np,np);
p10=zeros(np2,n+1); sd=zeros(n,r); d=zeros(n,r); Yf=zeros(n,2);
P0_b=zeros(np,np);

%%% Initialize variance for the state vector
p0(:,1)=S0(:);

%%% Begin Iterations (E-M Algorithm)
for it=1:niter

    Rsum=zeros(nq,nq);

        Qsum=zeros(np,np);

            x0_b=zeros(np,n+1);

```

```

    p0_b=zeros(np2,n+1);

for ir=1:r

    Yf=D(:,2*ir-1:2*ir);

    % ----- E-step starts here ----- %

    % Initial Mean for the State Vector

        x0(1:2,1)=Yf(1,:)';

        x0(3:4,1)=Yf(1,:)';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    la=0; % initalize a variable for calculating loglikelihood

        %%%% Forward Recursions (Kalman Filter) %%%%

    for ik=2:n+1

        % Initialize Observation Parameters

            A=Af;

            Q=Qf;

            y=Yf(ik-1,:)';

        % Adjust for missing data

            i0=find(y>0);

            i1=find(y==0);

            n0=length(i0);

            n1=length(i1);

            if n0>0

```

```

        R(i0,i0)=Rf(i0,i0);

    end

    if n1>0

        y(i1)=zeros(n1,1);

        A(i1,:)=zeros(n1,np);

    end

% Begin recursions

% Equation (A.1)

    x1(:,ik)=Phi*x0(:,ik-1);

    P0(:)=p0(:,ik-1);

% Equation (A.2)

    P1=Phi*P0*Phi'+Q;

    p1(:,ik)=P1(:);

% Equation (A.5)

    S=A*P1*A'+R;

    K=P1*A'/S; % Kalman Gain

% Equation (A.3)

    x0(:,ik)=x1(:,ik)+K*e;

% Equation (A.4)

    P0=(eye(np,np)-K*A)*P1;

    p0(:,ik)=P0(:);

```

```

% Compute filter residual

e=y-A*x1(:,ik);

% Increment -2 log L Equation (4)

la=la+log(det(S))+e'/S*e;

end % End of Forward Recursion (Kalman Filter)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% Backward Recursions (Kalman Smoother) %%%%

% Initialize Lag-One Smoother Covariance Equation (A.9)

P0(:)=p0(:,n);

P10=(eye(np,np)-K*A)*Phi*P0;

p10(:,n+1)=P10(:);

% Backward Recursion starts

for ik=n+1:-1:2

    P0(:)=p0(:,ik-1);

    P1(:)=p1(:,ik);

    % Equation (A.8)

    J=P0*Phi'/P1;

    % Equation (A.10):  $P_{\{t-1,t-2\}}^n$ 

    if ik<n+1 & ik>1

        P10=C10*J';

        p10(:,ik)=P10(:);
    end
end

```

```

end

if ik>2

    P10(:)=p10(:,ik);

    P0(:)=p0(:,ik-1);

    C10=P0+J*(P10-Phi*P0);

end

% Equation (A.6)

x0(:,ik-1)=x0(:,ik-1)+J*(x0(:,ik)-x1(:,ik));

% Equation (A.7)

P0(:)=p0(:,ik-1);

P00(:)=p0(:,ik);

P1(:)=p1(:,ik);

P0=P0+J*(P00-P1)*J';

p0(:,ik-1)=P0(:);

end % Backward recursion ends here

% -----End of E-Step ----- %

% Accumulate sums of products for update

S00=zeros(np,np);

S10=zeros(np,np);

S11=zeros(np,np);

R=zeros(nq,nq);

```

```

yp=zeros(nq,1);

y2=zeros(nq,1);

id=0;

% M-Step starts here (Updating Parameters)

for ik=2:n+1

    % For Phi and Q update

    P00(:)=p0(:,ik-1);

    P10(:)=p10(:,ik);

    P0(:)=p0(:,ik);

    % Equation (7)

    S00=S00+x0(:,ik-1)*x0(:,ik-1)'+P00;

    % Equation (8)

    S10=S10+x0(:,ik)*x0(:,ik-1)'+P10;

    % Equation (9)

    S11=S11+x0(:,ik)*x0(:,ik)'+P0;

    y=Yf(ik-1,:)' ;

    i0=find(y>0);

    i1=find(y==0);

    n0=length(i0);

    n1=length(i1);

    % Fully observed case

```

```

if n0==nq

    e=y-Af*x0(:,ik);

    % Equation (3.31)

    R=R+(e*e'+Af*P0*Af')/n;

end

% See Shumway and Stoffer (2000, Chp 4) for formulae need
% to handle missing data via E-M algorithm in State-Space Models
% (Fully unobserved and partially unobserved cases below)
% Ref: Shumway, R.H., & Stoffer, D.S. (2000).
% Time series analysis and its applications. New York: Springer.
% Fully unobserved case

if n1==nq

    yp=Af*x0(:,ik);

    R=R+Rf/n;

end

% Partially unobserved

if n1>0 & n0>0

    B(i1,i0)=Rf(i1,i0)/Rf(i0,i0);

    % Predicted for unobserved part

    yp(i1)=Af(i1,:)*x0(:,ik);

    % Observed part

```

```

e(i0)=y(i0)-Af(i0,:)*x0(:,ik);

y2(i1)=Af(i1,:)*x0(:,ik)+B(i1,i0)*e(i0);

trm=(Af(i1,:)-B(i1,i0)*Af(i0,:))*P0;

C11(i0,i0)=e(i0)*e(i0)'+Af(i0,:)*P0*Af(i0,:);

% Increment estimator for covariance R

R(i0,i0)=R(i0,i0)+C11(i0,i0)/n;

R(i0,i1)=R(i0,i1)+C11(i0,i0)*B(i1,i0)'/n;

R(i1,i0)=R(i0,i1)';

R22_1(i1,i1)=Rf(i1,i1)-B(i1,i0)*Rf(i0,i1);

R(i1,i1)=R(i1,i1)+(R22_1(i1,i1)+B(i1,i0)*C11(i0,i0)*B(i1,i0)')/n;

end

Q_temp=(S11-Phi*S10'-S10*Phi'+Phi*S00*Phi')/n;

Q(1,1)=Q_temp(1,1);, Q(2,2)=Q_temp(2,2);

end % End of M-step

Rsum=Rsum+R/r;

Qsum=Qsum+Q/r;

x0_b=x0_b+x0/r;

p0_b=p0_b+p0/r^2;

end % end of "ir" loop

Rf=Rsum;

Qf=Qsum;

```

```
% Calculation of  $-2 \log L$  %  
  
if it>(niter-3)  
  
    iteration=it;  
  
    m2_log_L=la;  
  
    Obs_cov=Rf;  
  
    Mod_cov=Qf;  
  
end  
  
end % end of "it" loop
```