Final Project

Report

On

Bayesian Online Change Point Detection

Course: Parameter & State Estimation CH5115

Department of Chemical Engineering

IIT Madras, Chennai

Submitted By:

Deepanshu (ED19D402)

1(a) Perform a critical review of the paper clearly highlighting its usefulness, limitations and shortcomings.

The paper **Bayesian online change point detection** by **Adams and MacKay** introduces a modular Bayesian framework for online estimation of changes in the generative parameters of sequential data. The paper examines the case where the model parameters before and after the change point are independent and derives an online algorithm for exact inference of the most recent change point.

Summary of the paper: The paper gives the formulation of change point detection on the basis of posterior probability of run length in terms of predictive posterior of new data as weight to the previous run length distribution, in recursive fashion. When a change point occurs the posterior prediction of that point decreases to reduce the weight of run length distribution and hence posterior probability of the current growth run length decreases and posterior probability of change point increases. In simple words as long as new data comes from same distribution; growth probability of current run length increases and change point probability decreases. Whenever new data comes from other than that of previous data distribution the predictive posterior of new data goes down, so growth probability of current run length decreases and change point probability increases. The previous run length probabilities acts as message passing to calculation of probability of next data. The algorithm has been applied on 3 real world datasets. The algorithm formulated in the paper is described in the steps below -

Algorithm: Bayesian Online Change point Detection

1. Initialize $P(r_0) = \tilde{S}$ or $P(r_0 = 0) = 1, \nu_1^{(0)} = \nu_{prior}$, $\chi_1^{(0)} = \chi_{prior}$
2. Observe New Datum y_t
3. Evaluate Predictive Probability:
$$\pi_t^{(r)} = P(y_t | \nu_t^{(r)}, \chi_t^{(r)})$$
4. Calculate Growth Probabilities:
$$P(r_t = r_{t-1} + 1, y_{1:t}) = P(r_{t-1}, y_{1:t}) \pi_t^{(r)} (1 - H(r_{t-1}))$$
5. Calculate Changepoint Probabilities :
$$P(r_t = 0, y_{1:t}) = \sum_{r_{t-1}} P(r_{t-1}, y_{1:t}) \pi_t^{(r)} H(r_{t-1})$$
6. Calculate Evidence $P(y_{1:t})$:
$$\sum_{r_t} P(r_t, y_{1:t})$$
7. Determine Run Length Distribution:
$$P(r_t | y_{1:t}) = P(r_t | y_{1:t}) / P(y_{1:t})$$
8. Update Sufficient Statistics:
$$u_{t+1}^{(0)} = u_{prior}, \quad \chi_{t+1}^0 = \chi_{prior}, u_{t+1}^{(r+1)} = u_t^{(r)} + 1, \chi_{t+1}^{(r+1)} = \chi_t^r + u(\chi_t)$$
9. Perform Prediction:
$$P(y_{t+1} | y_{1:t}) = \sum_{r_t} P\left(y_{t+1} \middle| y_t^{(r)}, r_t\right) P(r_t | y_{1:t})$$
10. Return to Step 2

Usefulness: The proposed method for online change point detection can be used in various domains for modeling and prediction point of view. Some of the applications are following:

1. Modeling and prediction of time series in application areas such as finance, biometrics, and robotics
2. Process control, EG analysis, DNA segmentation
3. Econometrics, Disease demographics etc.

Limitations: The limitations of the paper reside in the assumptions made during the formulation of the problem itself. Some of the limitations of the current algorithm are following:

1. This algorithm can be implemented for the non-overlapping sequences only. In many real world problems the data or sequences appears as overlapping sequences.
2. The proposed algorithm can't be used for non-parametric models. Again, the paper examines the case where the model parameters before and after the change point are independent.
3. The prior knowledge about the process and process parameter is one of the key aspects of the proposed algorithm. The choice of parameter and its distribution as prior play a major role in accurately detecting change points. In many cases we may not know the parameter and their distribution class beforehand.

4. The modeling assumption that the current length is conditionally depended on previous run length and independent of dataset in the past is limiting the usage of this algorithm. In short model is valid only if-

$$\Pr(\mathrm{r_t} \mid \mathrm{r_{t-1}}, \mathbf{y}_{1:t-1}) = \Pr(\mathrm{r_t} \mid \mathrm{r_{t-1}})$$

5. The model is valid if the data is conditionally dependent on current run length & recent data only, and independent of previous run length. In short model is valid if-

$$\Pr(\mathrm{y_t} \mid \mathrm{r_t}, \mathrm{r_{t-1}}, \mathbf{y}_{1:t-1}), = \Pr(\mathrm{y_t} \mid \mathrm{r_t}, \mathbf{y}_t^{(r)})$$

6. Although the modeling assumptions make the algorithm a bit strict, the proposed algorithm enjoys enormous amount of practical applications in real world problems.

Shortcomings: As far as presentation of the paper is concerned, it engages its reader from start to end. The motivation, problem formulation, scope of study, hypothesis and examples has been stated and explained very clearly. Some of the shortcoming which I found out is as follows:

1. The proposed algorithm demands the information about the process and its' parameters distribution beforehand.
2. The computational cost of posterior probability calculation for non-exponential conjugate models may be very high, especially in case of high dimensional data.
3. The performance of algorithm for non-exponential family likelihood hasn't been discussed in this paper; however paper clearly states that ***"Conjugate-exponential models are particularly convenient for integrating with the change-point detection scheme described here.***

1(b) Implement the algorithm described in the paper for the well-drilling nuclear magnetic response data.

Given: The process is piece-wise Gaussian and has multiple change points in mean. The change point occurrence in this process is memory less and can thus be modeled as a geometric distribution with the timescale, λ_{CP} = 250.and normal-gamma prior,

$$\mathrm{p}\,(\mu,\lambda;\ \mu_0,\mathrm{k},\alpha,\beta) = \mathrm{p}\,(\mu|\lambda;\ \mu_0,\mathrm{k})\mathrm{p}\,(\lambda;\ \alpha,\beta)$$

$$\text{where, } \mathrm{p}\,(\mu|\lambda;\ \mu_0,\mathrm{k}) = \mathrm{N}\left(\mu_0,\frac{1}{\mathrm{k}\lambda}\right)$$

$$\mathrm{p}\,(\lambda;\ \alpha,\beta) = \mathrm{Gamma}(\alpha,\beta)$$

$$\mu_0 = 1.15, \mathrm{k} = 0.01, \alpha = 20, \text{and } \beta = 2$$

Calculation of the posterior:

The posterior can be derived as follows:

$$\mathrm{p}(\mu,\lambda|\mathrm{D}) \propto \mathrm{NG}(\mu,\lambda|\mu 0,\kappa_0,\alpha_0,\beta_0)\mathrm{p}(\mathrm{D}|\mu,\lambda)$$

$$\propto \lambda^{\frac{1}{2}}\ \mathrm{e}^{-\frac{\kappa_0\lambda(\mu-\mu_0)^2}{2}}\lambda^{\alpha_0-1}\ \mathrm{e}^{-\beta_0\lambda}\ \times\ \lambda^{\frac{\mathrm{n}}{2}}\ \mathrm{e}^{-\frac{\lambda}{2}\sum_{\mathrm{i}=1}^{\mathrm{n}}(\mathrm{xi}-\mu)^2}$$

$$\propto \lambda^{\frac{1}{2}}\ \lambda^{\alpha_0+\frac{\mathrm{n}}{2}-1}\ \mathrm{e}^{-\beta_0\lambda}\ \mathrm{e}^{-\left(\frac{\lambda}{2}\right)\left[\kappa_0(\mu-\mu_0)^2+\sum_{\mathrm{i}=1}^{\mathrm{n}}(\mathrm{xi}-\mu)^2\right]}$$

Rearranging the terms,

$$\sum_{\mathrm{i}=1}^{\mathrm{n}}(\mathrm{x_i}-\mu)^2 = n(\mu-\bar{x})^2 + \sum_{\mathrm{i}=1}^{\mathrm{n}}(\mathrm{x_i}-\overline{x})^2$$

Again,

$$\kappa_0(\mu-\mu_0)^2 + n(\mu-\bar{x})^2 = (\kappa_0+n)(\mu-\mu_n)^2 + \frac{\kappa_0 n(x-\mu_0)^2}{\kappa 0+n}$$

$$\mu_n = \frac{\kappa_0\mu_0+nx}{\kappa_0+n}$$

Hence,

$$\kappa_0(\mu-\mu_0)^2 + \sum_{\mathrm{i}=1}^{\mathrm{n}}(\mathrm{x_i}-\mu)^2 = (\kappa_0+n)(\mu-\mu_n)^2 + \frac{\kappa_0 n(x-\mu_0)^2}{\kappa_0+n} + \sum_{\mathrm{i}=1}^{\mathrm{n}}(\mathrm{x_i}-\overline{x})^2$$

So,

$$p(\mu,\lambda|D) \propto \lambda^{\frac{1}{2}} e^{-\left(\frac{\lambda}{2}\right)(\kappa_0+n)(\mu-\mu_n)^2} \times \lambda^{\alpha_0+\frac{n}{2}-1} e^{-\beta_0\lambda} e^{-\left(\frac{\lambda}{2}\right)\sum_{i=1}^{n}(x_i-\bar{x})^2} e^{-\frac{\left(\frac{\lambda}{2}\right)\kappa_0 n(\bar{x}-\mu_0)^2}{\kappa_0+n}}$$

$$\propto N\left(\mu\middle|\mu n, \frac{1}{(\kappa+n)\lambda}\right) \times Ga\left(\lambda\middle|\alpha_0+\frac{n}{2}, \beta_n\right)$$

Where,

$$p(\mu,\lambda|D) = NG(\mu,\lambda|\mu_n,\kappa_n,\alpha_n,\beta_n)$$

$$\mu_n = \frac{\kappa_0\mu_0 + n\bar{x}}{\kappa_0 + n};\ \kappa_n = \kappa_0 + n;\ \alpha_n = \alpha_0 + \frac{n}{2};\ \beta_n = \beta_0 + \frac{1}{2}\sum_{i=1}^{n}(x_i-\bar{x})^2 + \frac{\kappa_0 n(\bar{x}-\mu_0)^2}{2(\kappa_0+n)}$$

In our case D = x:

$$\mu_{\mathbf{n}} = \frac{\boldsymbol{\kappa_0\mu_0 + x}}{\boldsymbol{\kappa_0 + 1}};\ \boldsymbol{\kappa_n = \kappa_0 + 1};\ \boldsymbol{\alpha_n = \alpha_0 + \frac{1}{2}};\ \boldsymbol{\beta_n = \beta_0 + \frac{\kappa_0(x-\mu_0)^2}{2(\kappa_0+1)}} \dots\dots (\mathbf{i})$$

We will use above result for prior parameter update.

Now,

$$p(x_{new}|x_i) = \frac{\Gamma(\alpha_n+1)}{\Gamma(\alpha_n)}\frac{\beta_n^{\alpha_n}}{\beta_{n+1}^{\alpha_n+1}}\left(\frac{\kappa_n}{\kappa_n+1}\right)^{\frac{1}{2}}(2\pi)^{-\frac{1}{2}}$$

Let,

$$\Lambda = \frac{\alpha_n\kappa_n}{\beta_n\kappa_{n+1}};$$

Finally,

$$\mathbf{p(x_{new}|x_i)} = \boldsymbol{(\pi)^{-\frac{1}{2}}}\frac{\boldsymbol{\Gamma\left(\frac{2\alpha_n+1}{2}\right)}}{\boldsymbol{\Gamma\left(\frac{2\alpha_n}{2}\right)}}\left(\frac{\boldsymbol{\Lambda}}{\boldsymbol{2\alpha_n}}\right)^{\frac{1}{2}}\left(\mathbf{1}+\frac{\boldsymbol{\Lambda(x-\mu_n)^2}}{\boldsymbol{2\alpha_n}}\right)^{-\frac{2\alpha_n+1}{2}} \dots\dots (\mathbf{ii})$$

We can see this is a T-distribution with center at μ_n, precision Λ, and degree of freedom $2\alpha_n$. We will use above result for posterior calculation. For implementation following MATLAB Code has been written:

- **pred_probability:** Calculating posterior probability of new data
- **Bayes_CP_Detect:** Detecting change point
- **plot_post:** Plotting the posterior run length distribution
- **project_Q2:** For initialization, reading data and executing functions

Now we are all set to use the above information and codes for detecting change points in "NMRlogWell" dataset.

MATLAB Code:

```
%% Import data
load('NMRlogWell')

% initialization
RL(1) = 0; i=0; len = length(y); post_prob = zeros(len+1,400);

% implementing bocd algorithm
while(sum(RL)<=len-1)
      i=i+1;
      y=y(RL(i)+1:end);
      lambda_cp = 250; Alpha = 20; Beta = 2; mu = 1.15; k = 0.01;
      [R,max_R,run_len] = Bayes_CP_Detect(Alpha,Beta,k,mu,lambda_cp,y);
      RL(i+1) = run_len;
      fprintf('\n Change point detected at t=%d',sum(RL));
      data = y(1:RL(i+1));
      t = 1+sum(RL(1:i)):sum(RL(1:i))+length(data);
      post_prob(1+sum(RL(1:i)):sum(RL(1:i))+length(data)+1,:) =
      R(1:run_len+1,1:400);
      figure(1);
      subplot(2,1,1)
      plot(t,data),hold on
      xline(sum(RL),'r-.','LineWidth',1.5,'Label','CP')
      ylabel('data(y)');xlabel('Time (T)')
      title('Bayesian Online Change Point Detection')
      subplot(2,1,2)
      Mat = 50+log10(R(1:run_len+1,1:400));
      Y = linspace(1,400,400); X = linspace(sum(RL(1:i)),sum(RL),RL(i+1)+1);
      contour(X,Y,Mat'); colorbar; hold on
      title('Posterior of Run Length on Log Scale')
      ylabel('Run Length');xlabel('Time (T)')
end

%% Shades
plot_post(post_prob)
```

Command Window:

```
Change point detected at t=87

Change point detected at t=222

Change point detected at t=452

Change point detected at t=818

Change point detected at t=877

Change point detected at t=954

Change point detected at t=1000
```

Plots: It can be seen from the plot that change points (abrupt change in mean of the data) occurred at time instances:

T = 87, 222, 452, 818, 877, 954, 1000

The same can be verified from the posterior of run length probability plots as well.

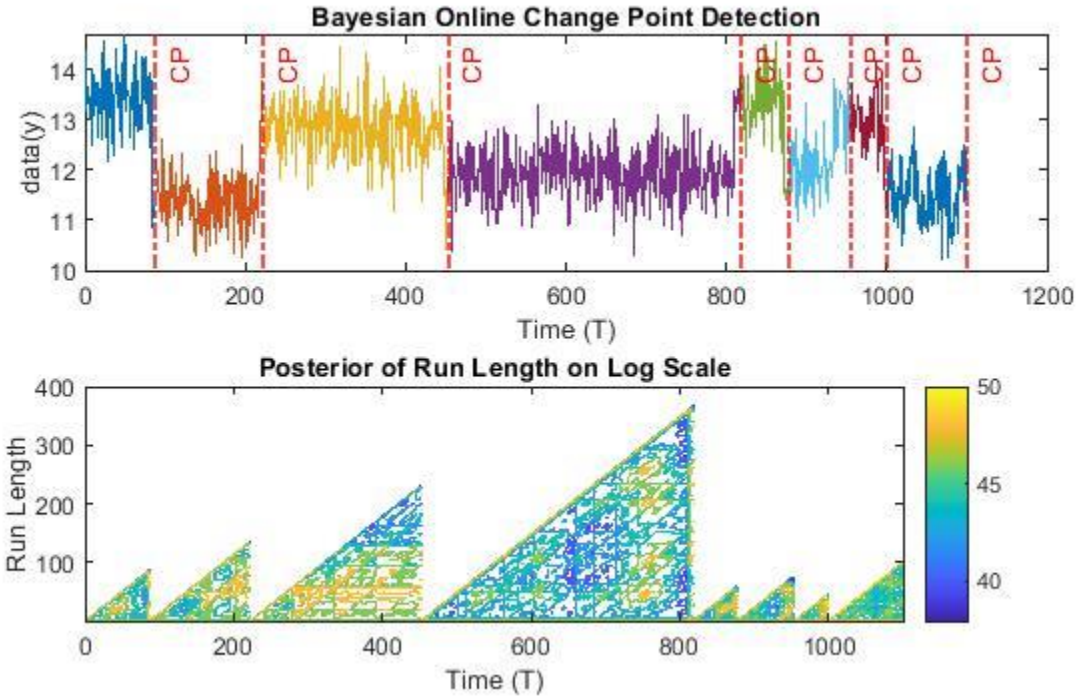

Figure 1(a) Bayesian online change point detection (b) contour plot of posterior of run length

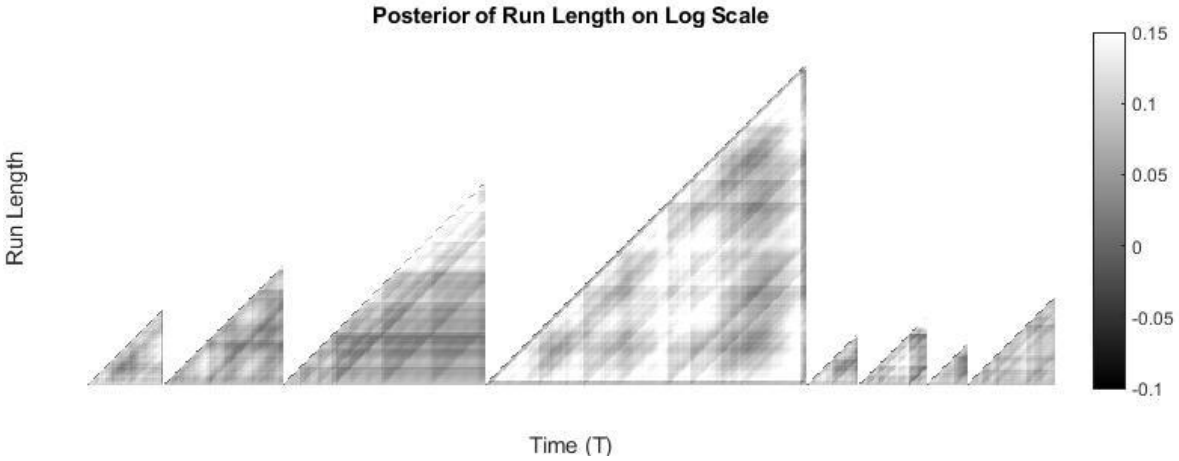

Figure 1(c) Posterior of run length on Log scale

Darker pixels show the higher posterior run length probability. The plot has been scaled on logarithmic scale and relevant some transformations have been made to make look it more convenient for visualization perspective.

2(a) Extending the Bayesian online change point detection algorithm by integrating it with the Recursive Least Squares (RLS) algorithm.

PACF of the 'historic' data (figure 2(a)) suggests that model is AR (1). The initial model has been fitted and tested for whiteness of residuals (figure 2(b)).

Initial Model: $y[k] + 0.6051^{(\pm 0.0131)} y[k-1] = e[k];\ \sigma_e^2 = 0.0849$

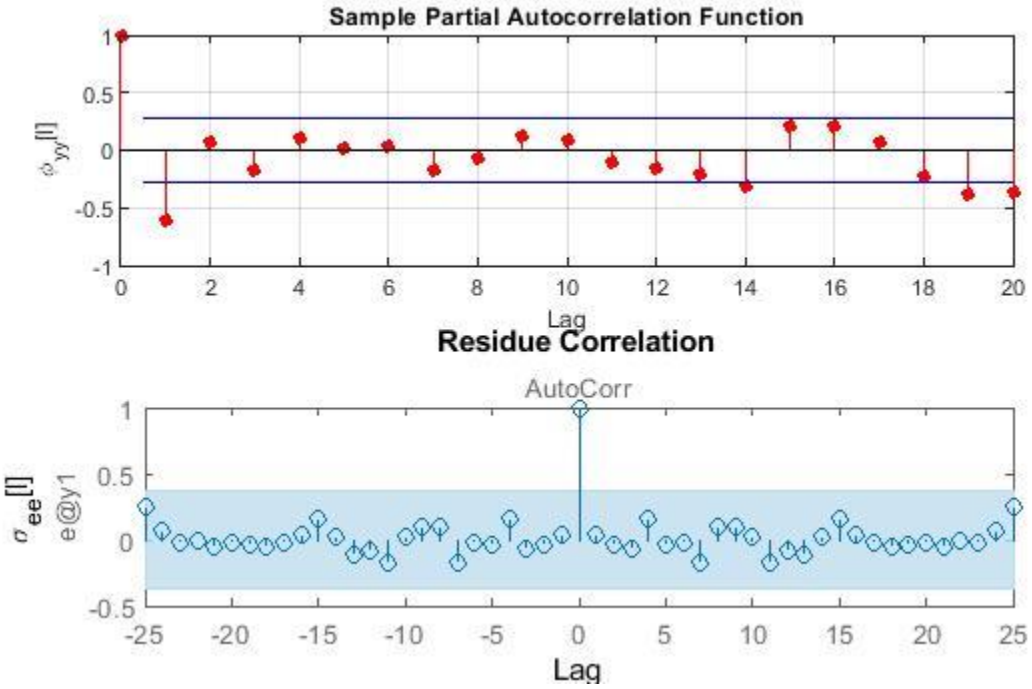

Figure 2(a) PACF of 'historic' data (b) ACF of residuals

Initial model's parameters have been used for initializing the RLS algorithm on 'new' dataset. Refining and tracking the parameters is being done and after runtime t = 200, Bayesian online change point algorithm has been executed. It has been found out at that, at some time around instant t = 290, there was an abrupt change in estimated parameter. The same can be seen in figure 3(a). Bayesian online change point algorithm has been stopped at t = 293 and noted as change point.

On executing Bayesian online change point algorithm after 200 time steps the estimated parameter started converging to a fixed value till t = 85 (overall time step 285). During this time interval the posterior of run length also increased. Somewhere around t = 90 (overall time step 290) the estimated parameter shown an abrupt change and posterior of run length probability dropped to zero. The estimated run length before change point has been reported at RL= 93 and change point has been detected at t=293 on 'new' dataset. This can also be observed in figure 3(a), (b) &(c). Model before change point detection using RLS:

Model before CP detection: $y[k] + 0.5567^{(\pm 0.0088)} y[k-1] = e[k];$

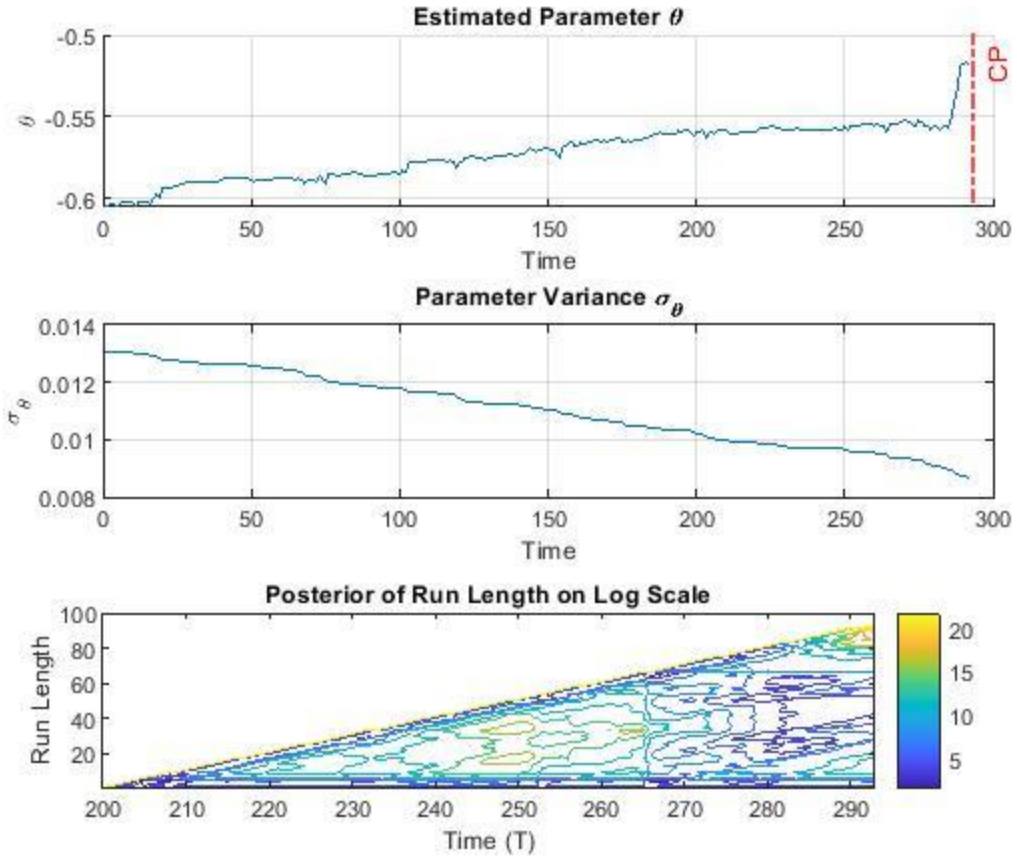

Figure 3(a) Estimated parameter (b) Parameter variance (c) Posterior of run length

2(b) Updating the model after change point detection.

PACF of the 'new' dataset after CP detection has been observed and has been found out that model's order remains same viz. AR(1). PACF can be seen LS and RLS both methods have been used for updating the model's parameter and error variance. The whiteness of residuals has been successfully tested. Refer figure 4 (a) and (b).

Updated model using RLS: $\mathrm{y[k]} + 0.3913^{(\pm 0.0067)}\mathrm{y[k-1]} = \mathrm{e[k]};$

Updated model using LS: $\mathrm{y[k]} + 0.3813^{(\pm 0.0042)}\mathrm{y[k-1]} = \mathrm{e[k]};\ \sigma^2_{\mathrm{e_{new}}} = 0.3046$

One can see that the variance of driving white noise has been increased from $\boldsymbol{\sigma^2_{\mathrm{e}} = 0.0849}$ **to** $\boldsymbol{\sigma^2_{\mathrm{e_{new}}} = 0.3046}$ causing the abrupt change in AR (1) model at around t = 290. That has been reported at t = 293 by Bayesian online change point algorithm.

It can also be observed that the estimated parameters using RLS and LS are approximately same.

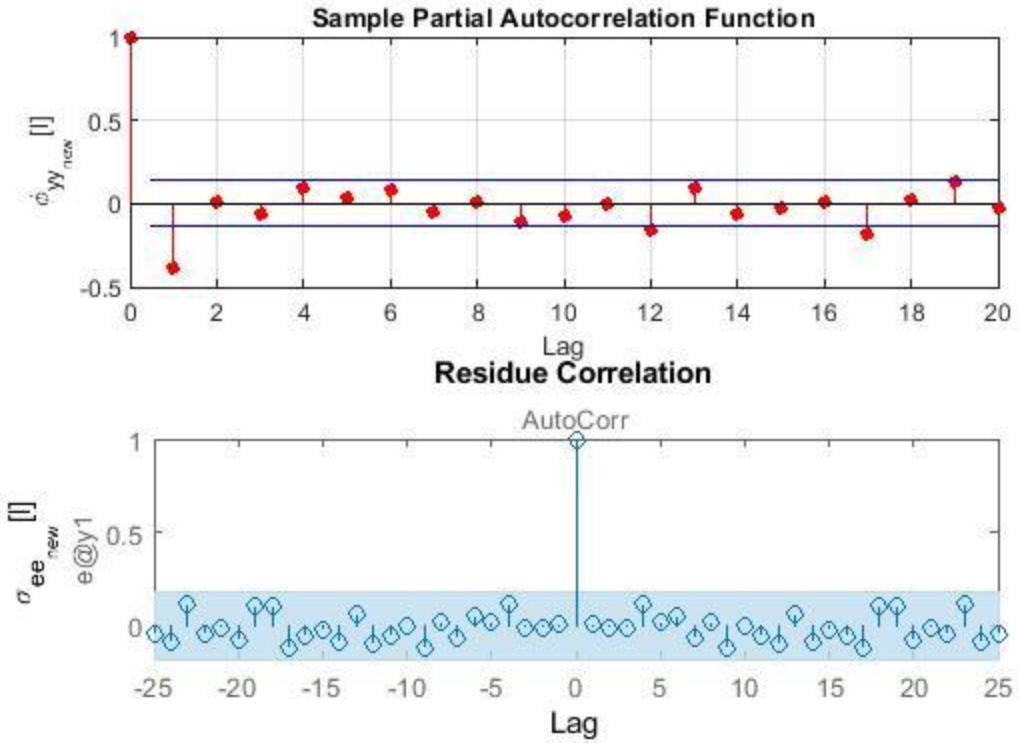

Figure 4(a) PACF of rest of 'new' data (b) ACF of residuals

The updated parameter is shown in figure 5. It can be seen that parameter setteles down in between t = 150 to t = 200 and parameter variance is decreasing as well, showing that parameter will converge asymptotically.

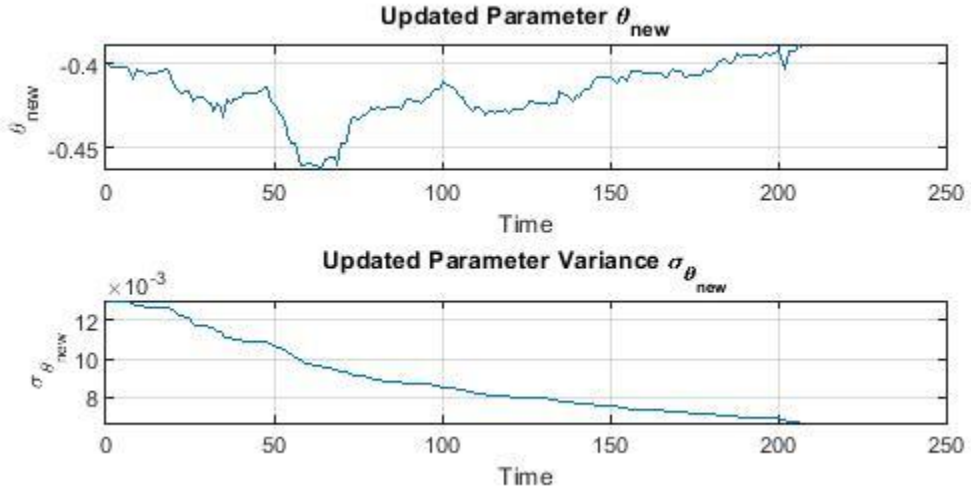

Figure 5(a) estimated parameter (b) Parameter variance

MATLAB Code:

```
%% Load data
load('historic')
figure(1);
subplot(2,1,1); parcorr(y);ylabel('\phi_{yy}[l]')

%% LS estimates for initial model
sys = ar(y',1); % ar routine uses LS method
subplot(2,1,2)
resid(y',sys);ylabel('\sigma_{ee}[l]');
theta1 = -sys.Report.Parameters.ParVector  ;
var_v = sys.Report.Parameters.FreeParCovariance;
sigma_e = sys.NoiseVariance;


%% Recursive LS for new dataset
load('new')
% Fit Model 1
thetainit = theta1; Pinit = var_v; stop = 200;
obj_p1 =
recursiveLS(1,'InitialParameters',thetainit,'InitialParameterCovariance',Pini
t);
thetaest_vec = []; Ptheta = [];K=[]; i=0; RL(1) = 0;
for n = 2:numel(y)
    if n>stop+sum(RL)
        break
    end
    H = y(n-1);
    [theta,~] = obj_p1(y(n),H);
    Ptheta(n-1,1:length(thetainit)) = obj_p1.ParameterCovariance;
    thetaest_vec(n-1,1:length(thetainit)) = theta;
    K(n-1,1:length(thetainit)) = obj_p1.ParameterCovariance*H';
    if n>=stop
         y= y(n:end);
        while(i<1)
            i=i+1;
            y=y(RL(i)+1:end);
            lambda_cp = 50; Alpha = 10; Beta = 1; mu = 0; k = 1;
            [R,max_R,run_len] = Bayes_CP_Detect(Alpha,Beta,k,mu,lambda_cp,y);
            RL(i+1) = run_len;
            fprintf('\n Change point detected at time %d \n',n+sum(RL));
            data = y(1:RL(i+1));
            t = 1+sum(RL(1:i)):sum(RL(1:i))+length(data);
            figure(2);
            subplot(3,1,1)
            xline(n+sum(RL),'r-.','LineWidth',1.5,'Label','CP'); hold on
            ylabel('data(y)');xlabel('Time (T)')
            title('Bayesian Online Change Point Detection');hold on
            subplot(3,1,3)
            Mat = -log(R(1:run_len+1,1:100));
            Y1 = linspace(1,100,100); X =
n+linspace(sum(RL(1:i)),sum(RL),RL(i+1)+1);
            contour(X,Y1,Mat'); colorbar; hold on
            title('Posterior of Run Length on Log Scale')
            ylabel('Run Length');xlabel('Time (T)')
```

```
        end
    end
    load('new')
end
figure(2)
subplot(3,1,1); plot(thetaest_vec); title('Estimated Parameter \theta');
ylabel('\theta');xlabel('Time');grid on; hold on;

%% Model Update
y_new = y(stop+run_len:end);
% Fit Model using RLS
thetainit1 = -0.4; Pinit1 = var_v;
obj_p2 =
recursiveLS(1,'InitialParameters',thetainit1,'InitialParameterCovariance',Pin
it1);
thetaest_vec1 = []; Ptheta1 = []; Kn = [];
for n = 2:numel(y_new)
    H1 = y_new(n-1);
    [theta,~] = obj_p2(y_new(n),H1);
    thetaest_vec1(n-1,1:length(thetainit1)) = theta;
    Ptheta1 (n-1,1:length(thetainit1)) = obj_p2.ParameterCovariance;
    Kn(n-1,1:length(thetainit1)) = obj_p2.ParameterCovariance*H1';
end

%% Plots
figure(2)
subplot(3,1,2)
plot(Ptheta); title('Parameter Variance \sigma_{\theta}');
ylabel('\sigma_\theta');
xlabel('Time');grid on; hold on;

figure(3)
subplot(2,1,1)
plot(thetaest_vec1); title('Updated Parameter \theta_{new}');
ylabel('\theta_{new}');xlabel('Time');grid on; hold on;
subplot(2,1,2)
plot(Ptheta1);title('Updated Parameter Variance \sigma_{\theta_{new}}');
ylabel('\sigma_{\theta_{new}}');xlabel('Time');grid on; hold on;

%% Fit model using LS estimates
figure(4);
subplot(2,1,1); parcorr(y_new);ylabel('\phi_{yy_{new}}[l]');
sys1 = ar(y_new',1); % ar routine uses LS method
subplot(2,1,2)
resid(y_new',sys1);ylabel('\sigma_{ee_{new}}[l]');
theta_new = -sys1.Report.Parameters.ParVector  ;
var_v_new = sys1.Report.Parameters.FreeParCovariance;
sigma_e_new = sys1.NoiseVariance;

%% Error Calculation
fprintf('Error Variance in base model= %d \n',sigma_e);
fprintf('Error Variance in updated model= %d \n',sigma_e_new);
```