BY: S VISHAL
CH18B020

CH5115 PROJECT

Q1) a) Goal of the Paper:

The main goal of the paper is to detect online 'change-points' in the generative parameters of a data sequence adopting a Bayesian approach. They make the assumption that the parameters before and after a change point are independent of each other. They approach this problem by estimating the length of the current "run"

Contribution of the paper and Why is it useful

~~The authors state that~~

The authors state that most Bayesian approaches prior (pun not intended) to their approach use retrospective and offline approach ~~to~~ change pt detection. If that is so, then this work ~~work constitutes~~ is of immense value.

Many applications essentially require online detection, let's say a plant where certain vital states of system are being estimated or consider a medical application where ~~the~~ some signal indicating the health of the vital system of a man is presented, immediate detection of abnormalities might prove to be necessary.

The authors also cite an example of brightness change detection in vision systems which ~~extort~~ ~~ordeal~~ mandate the requirement of online detection.

Although frequentist approaches have been developed, Bayesian gives the additional advantages it always carries with it : i) use of prior information and a ii) probabilistic distribution of the required parameter. The latter advantage might prove ~~to~~ be useful in many scenarios.

The approach is versatile in the sense that ~~it~~ there is a clear cut separation between the implementation of the change-point algorithm and the implementation of the model. This means we can use this algorithms across processes with ~~to~~ different models. This course project itself is a nice example for this:

→ In q1) we apply the algorithm to predict parameters (mean) of a distribution.

→ In q2)

→ In q1) we use it to find change points in which the parameter of the distribution (mean) is changing

→ In q2) we use it to ~~d~~ find change points in an AR(1) model where the variance of the endogeneous white-noise driving force chngs. (variance)

So the algorithm works for 2 very different scenarios!

How the Algorithm Works.

To detect changepoints the authors try to compute the run length distribution (conditioned on the data)

Run length is defined as the number of datapoints which has been following a specific generative process. Alternatively, it is the time since the last change point occurred

By its very definition $r_{t+1} = \begin{cases} r_t + 1 & \text{probability } p \\ 0 & 1-p \\ \text{any other value} & 0 \end{cases}$

where p is a valid probability

[Example of run length values]

So by predicting r_t we will know whether changepoint has occured at the datapoint x_t, because at change point r_t will be 0

reqd. value : $P(r_t | x_{1:t})$ $\forall r_t$

(Note : $r_t \le n$ where n is the index of the current data pt)

$\Rightarrow$ we need $\dfrac{P(r_t, x_{1:t})}{P(x_{1:t})}$

~~To get~~ Note that $P(x_{1:t}) = \sum_{\forall r_t} P(r_t, x_{1:t})$

marginal.

So we need the joint distribution.

For that, after some neat simplifications (and assumption of Markov Property - for predicting r_t we need only r_{t-1}) the authors derive a recursive eqn of the form:

$$P(r_t, x_{1:t}) = \sum P(r_t | r_{t-1}) \, P(x_t | r_{t-1}, x_t^{(r)}) \, P(r_{t-1}, x_{1:t-1})$$

$P(r_t | r_{t-1})$ can be modelled using a hazard function (like geometric distribution)

Now we need the ~~s~~term: $P(x_t | r_{t-1}, x_t^{(r)})$

We can write it ~~as~~ as some

$$P(x_{t+1} | x_{1:t}) = \int_\Theta P(x_{t+1} | \theta) \, P(\theta | x_{1:t}) \, d\theta.$$

where $P(x_{t+1} | \theta)$ is the so called predictive -

and Θ is the parameter(s) involved in

the DGP of x. And $P(\theta | x_{1:t})$ is the posterior

Here the author's assume exponential family

likelihoods and conjugate exponential prior $P(\theta | X, \nu)$

which ensures an exponential posterior.

~~Bo~~ This allows modularity wherein we can update

the ~~param~~ hyperparameters X and ν independently

of run length predictions (thus, ensuring the delineation

of change point algorithms and model).

~~Thus,~~ • For initial conditions we ~~draw~~ the

paper considers 2 cases:

① $P(r_0 = 0) = 1$ (change point occurred apriori before the first datum.) of data

② Some recent observations of a subset is used to get a prior over the initial run lngth as the normalized survival functn.

$$P(r_0 = \tau) = \frac{1}{Z} S(\tau) \; ; \; S(\tau) = \sum_{t=\tau+1}^{\infty} P_{gap}(g = t)$$

$Z \rightarrow$ normalising const

Thus, putting together all these pieces, the algorithm is complete! (we get a value for $P(r_t | x_{1:t})$)

PERFORMANCE & SCOPE FOR IMPROVEMENT

→ Three datasets are considered in the paper and the algorithm performs reasonably well in all 3 cases. It predicts the changepoints within a little error

→ It detects changes in both mean (NMR Well log data) and variance (DOW JONES RETURNS data)

→ The algorithm is computationally very friendly (as long as $P(x_{t+1} | r_t)$ is evaluated to be a simple function / exact solution for the integral exists)

So it can easily keep up pace with the arrival of new data.

→ However despite its very good performance in the project data as well as the original datasets the authors have used, there are some concerns.

(i) Underflow error: (<1)

As we keep on multiplying probabilities the values of the joint probabilities get closer and closer to zero i.e. eventually become so insignificant the computer can't evaluate resulting in the failure of the algorithm.

One alternative is to take logarithm of probabilities and work with those instead. But ~~other~~ ~~ie~~ ~~and~~ more robust solutions are needed.

A closely related issue is that the $P(r_t, x_{t})$ vector (along r_t) is likely to be sparse, so the ~~space~~ some approximation or modification is implementation needs to be made to ensure optimal ~~for~~ space usage (i.e. time as well since time is $O(N)$ complexity)

(ii) Initial conditions:

This is an issue which we face in Bayesian estimation, - choice of prior & hyperparameters.

One needs to test for the robustness of this algorithm in case there are errors in the prior or the model or the run length assumption $(P(r_0 = 0) = 1)$

Further Improvements could also be made for.

→ Further Improvements could also be made for approximations of the predictive in case the exact solution is unavailable

As a closing remark, I would like to state that the algorithm is immensely useful and as long as the applications are chosen carefully and is definitely a big contribution for the field of online Bayesian changepoint detection

BY S.VISHAL
CH18B020

CH5115 PROJECT REPORT

Q1) b) Implementing bocd on well-drilling NMR data

The given NMR data is plot and by viewing the plot we see that:

1. The mean of the DGP responsible for generation of the data is changing.
2. However, the ('local') variance seems to be more or less the same throughout

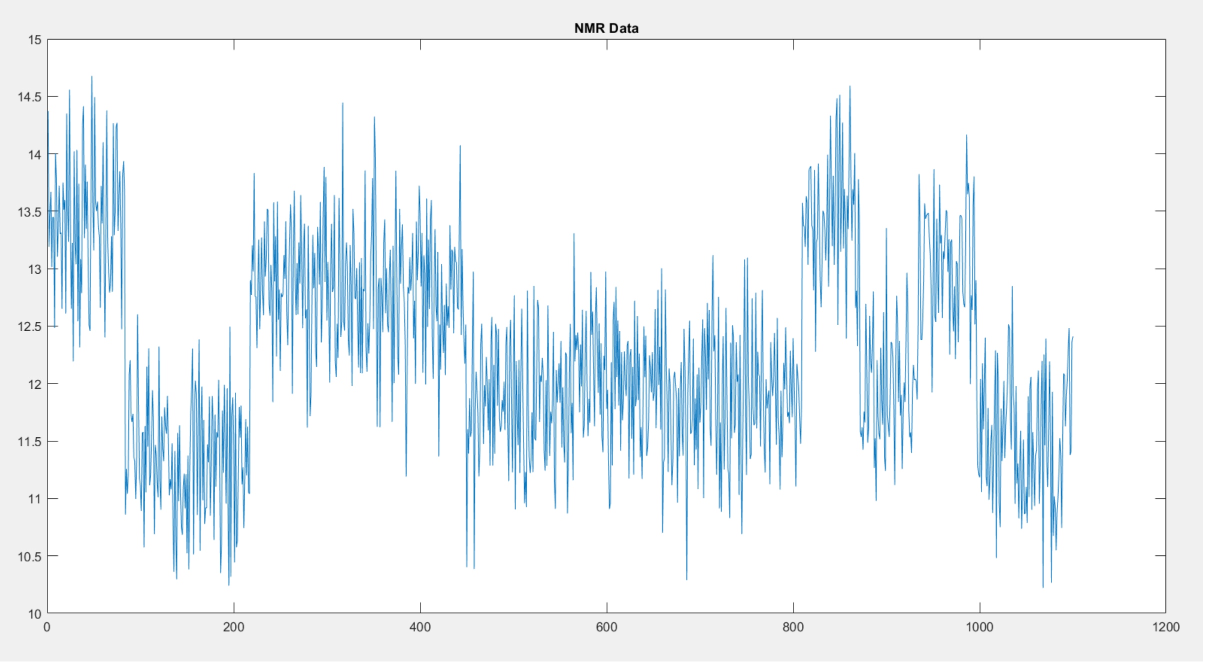

Figure 1: Plot of the given data

We assume that the DGP is piece-wise Gaussian, has multiple change-points in mean but has a constant standard deviation throughout.

A further simplifying assumption is that the changepoint occurrence is memoryless and is a geometric distribution with $\lambda_{CP} = 250$.

Note that we don't need to perform predictions here so it is sufficient to get run-length posterior to solve the problem. And to get the run length posterior using the bocd algorithm we need:

Joint probabilities of run length and data $P(r_t|x_{1:t})$**:**

These are obtained using the recursive equations set up in the paper.

UPM Predictive/Predictive Probability:

These are obtained from equation (11) in Mackay.[1] We have a Gaussian likelihood and we use a normal-gamma prior. Upon evaluating we get a t-distribution of the form:

$$p(x|D) = t_{2\alpha_i}(x|\mu_i, \frac{\beta_i(\kappa_i + 1)}{\alpha_i \kappa_i})$$

Where α, μ, β and κ are the hyperparameters.

Conditional prior on the change point $P(r_t|r_{t-1})$

We assume this to be a constant ('Markov Property'; as a result of the geometric distribution assumption) = 1/λ.

With all these set, we simply follow 'Algorithm 1' in [1]. Initial value of the joint probabilities are defined as $P(r_1=1|r_0=0)$ = predictive*(1-H), $P(x = x_1, r_1=0|r_0=0)$ = predictive*(H) and for all other r_t, it is set to be 0.

Some MATLAB implementation details

Computing pdf from t distribution:

MATLAB's tpdf computes the pdf of t distributions with mean zero and variance 1. So, I accordingly transformed the given data: y = (x- μ)/(standard-deviation). This has mean 0 and variance 1, therefore, I find f(y) and using f(y), we can get f(x) as f(y)/ (standard-deviation).

This can be proved quite easily by computing and comparing their CDFs.

Underflow errors:

After a certain iteration, the joint probabilities grow sufficiently small, that even the sum of the joint probabilities (evidence: $P(x_{1:t})$) goes to zero (mathematically it doesn't obviously, but it is too small to be evaluated on MATLAB, and it is taken as good as zero)

Now since this term is in the denominator of the runlength distribution, the runlength probabilities blow up to infinity (inf)

7. Determine Run Length Distribution

$$P(r_t \mid x_{1:t}) = P(r_t, x_{1:t})/P(x_{1:t})$$

To resolve this I tried transforming all probabilities to **log-scale** as shown in [3]. Unfortunately that didn't stop $P(x_{1:t})$ going to zero (log P went to –Inf). I believe this is because we anyway need to convert it to the original scale to sum it up to find the evidence/sum of joint distribution probabilities. (products can be done in log-scale: it will simply be addition) I also tried exponential transformations but they too didn't work.

Instead I normalized the joint probabilities such that their **sum is 1** in every iteration, i.e. divide every joint probability by its sum. This is correct, mathematically and is equivalent to substituting the joint probability by $P(r_t|x_{1:t})*P_{evidence}$ in the recursive equations ($P_{evidence}$ will cancel from both sides).

This method worked for me.

Update statements

$$\mu_n = \frac{\kappa_0\mu_0 + n\overline{x}}{\kappa_0 + n} \tag{86}$$

$$\kappa_n = \kappa_0 + n \tag{87}$$

$$\alpha_n = \alpha_0 + n/2 \tag{88}$$

$$\beta_n = \beta_0 + \frac{1}{2}\sum_{i=1}^{n}(x_i - \overline{x})^2 + \frac{\kappa_0 n(\overline{x} - \mu_0)^2}{2(\kappa_0 + n)} \tag{89}$$

I borrowed these expressions from Murphy.[2] For our case n=1, because we are updating the parameters using one data point at once.

Plotting (plot_rt_probs.m)

Mat2gray was used to form a grayscale image matrix from the matrix. I had to flip the axis and also do a 1-p to ensure higher probabilities correspond to black colour. This was inspired from [5]. To get a better gradient, I took a fractional power of all the elements. This results in an increase in value but obviously a<b => $a^x < b^x$ for all positive x, so it is conserved. Here too, I tried log normalization as described in MATLAB forums but it didn't yield the required results.

Results

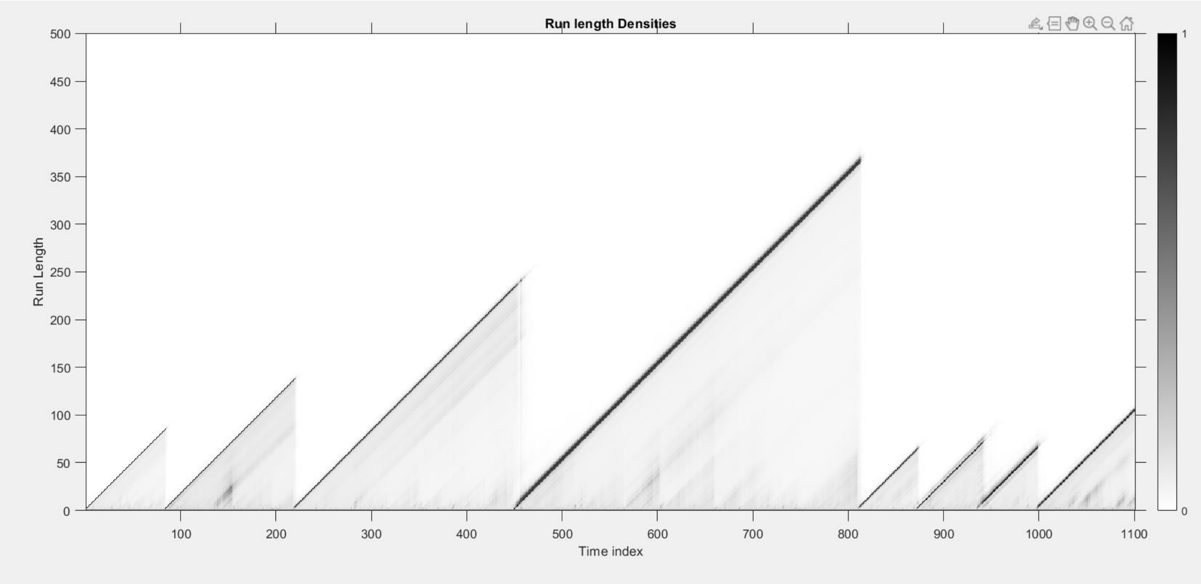

Figure 1.2: Run length posterior plotted (in a form similar to how it is done in the paper)

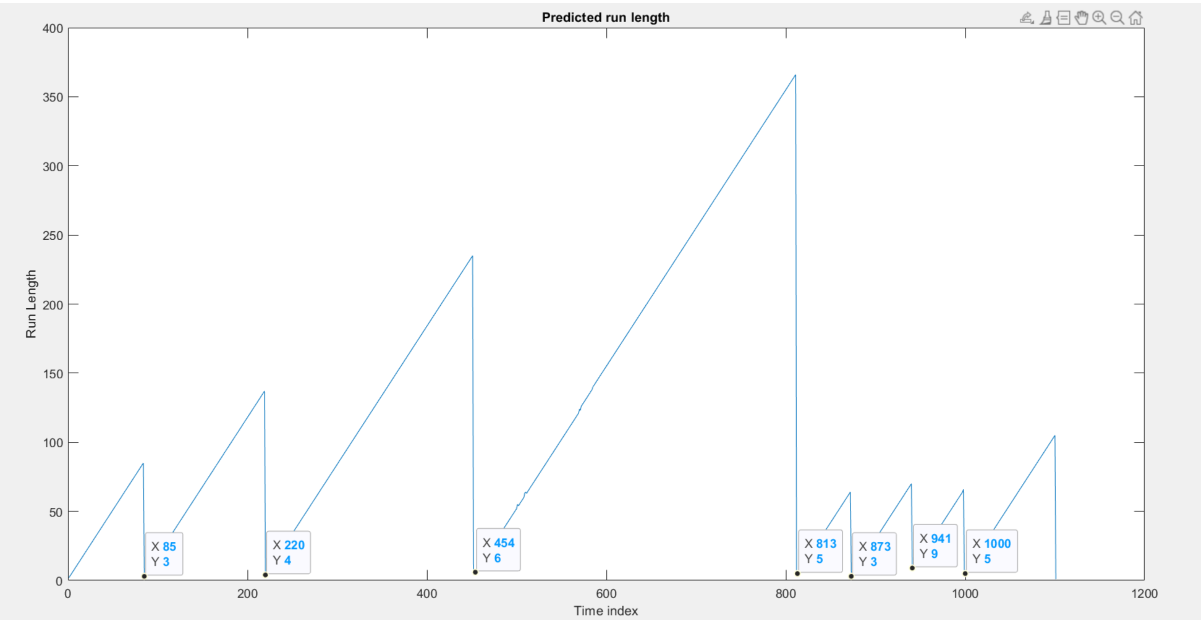

Figure 1.3: Predicted run length (run length with the highest probability at that instant)

Changepoints were found to be: 85, 220, 454, 813, 873, 941, 1000

Q2) a) Fitting AR model and finding Change points

AR(1) Model fitting

Following the plot of ACF and PACF of historic data

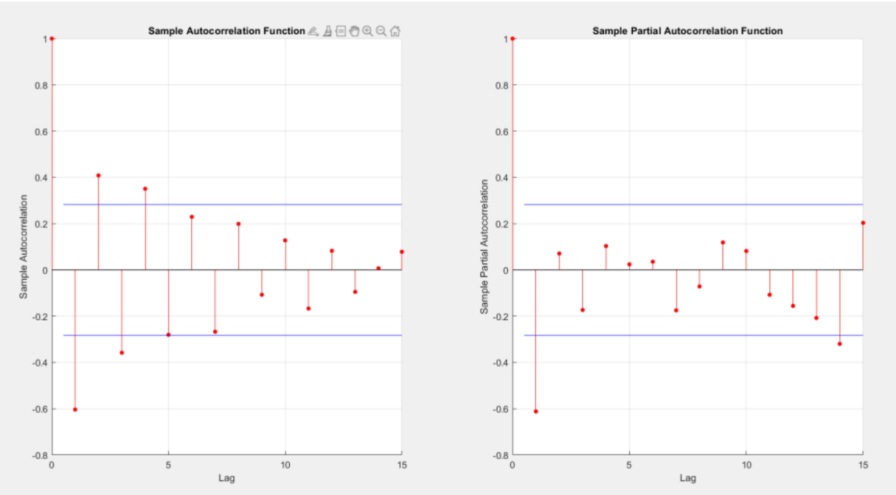

Figure 2.1: Historic data correlations

ACF exponentially dies, PACF goes to zero after step 1. This means that the model is AR(1).

Fitlm is used to fit an AR model. The constant term is deemed insignificant.

```
Linear regression model:
    y ~ x1

Estimated Coefficients:
           Estimate      SE        tStat       pValue
           ________    _______    _______    __________

    x1     -0.60857    0.11558    -5.2654    3.2462e-06


Number of observations: 49, Error degrees of freedom: 48
Root Mean Squared Error: 0.294
```

Residual Analysis

The ACF and PACF are calculated. Both become zero at lags > 0, indicative of white noise. This is confirmed using the **Ljung-Box test,** where the null hypothesis that the residuals are white is not rejected.

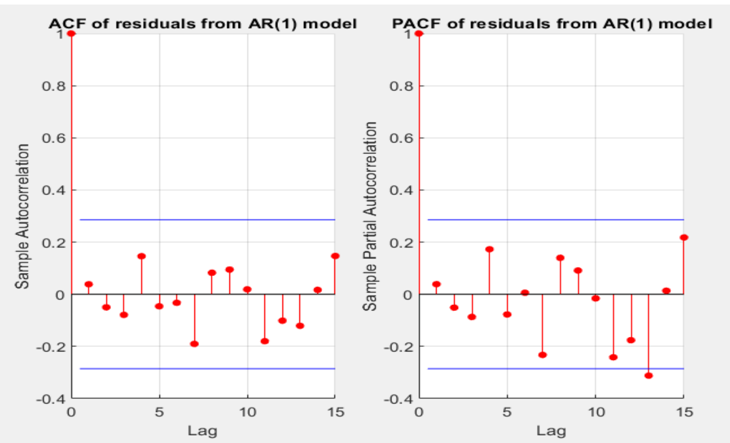

Fig 2.2: Residual correlations

RLS + BOCD on new data

First RLS is performed on the new data with the initial parameters same as that of the AR model. This is done upto the first 200 data points.

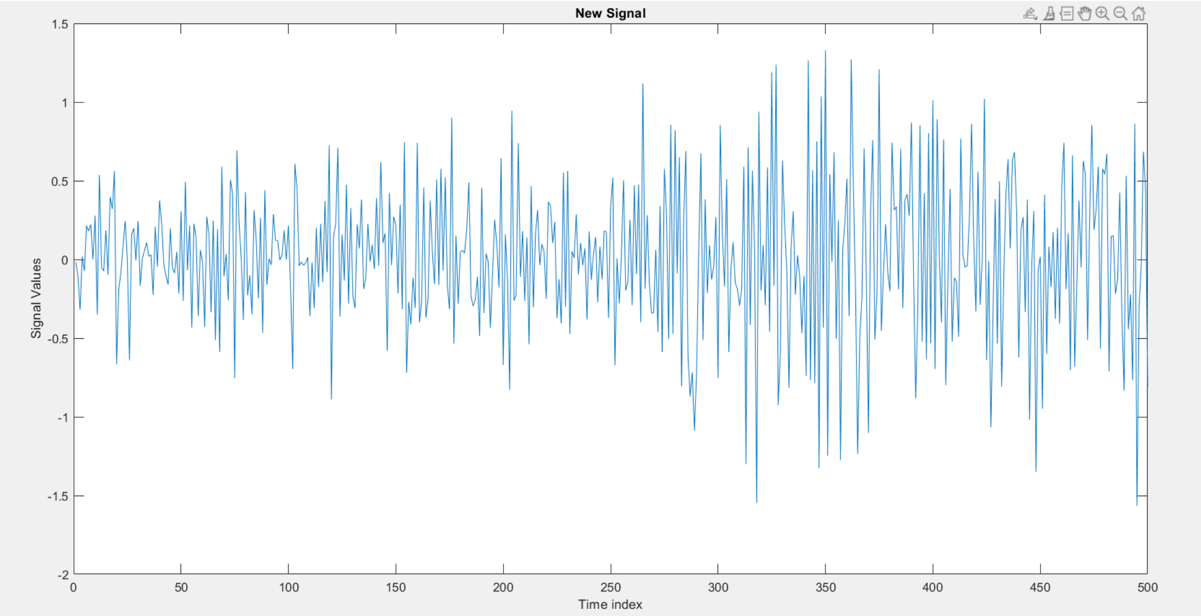

Figure 2.3: The new signal values. We can see that the mean is constant but there seems to be a change in variance at some point near 300.

Then we perform bocd, the code is same as the one used in the previous part with just different **initial hyper-parameter values**. As each new data point arrives, the **residual is passed to the bocd algorithm** and the RLS model is continually updated. The iteration is stopped once there is a steep reduction in the predicted run length (close to 0 obviously, since, r_{t+1} can be $r_t + 1$ or 0).

By observing the probability plot we see a drastic change in run length at 88 where r=3 is the maximum probable scenario. So we can consider that indices [86,87,88] belong to the new DGP. Also, note that plot is made after 200, so the **change point is estimated to be at 200+86 = 286**.

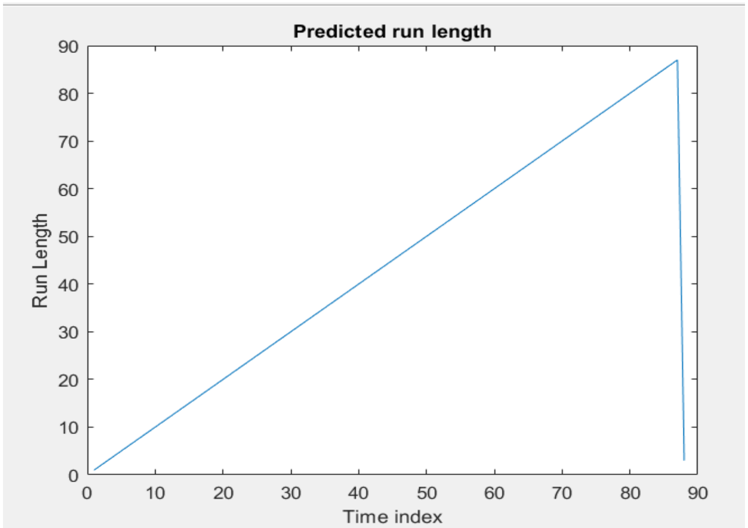

Figure 2.4: Predicted run length for data-points after 200

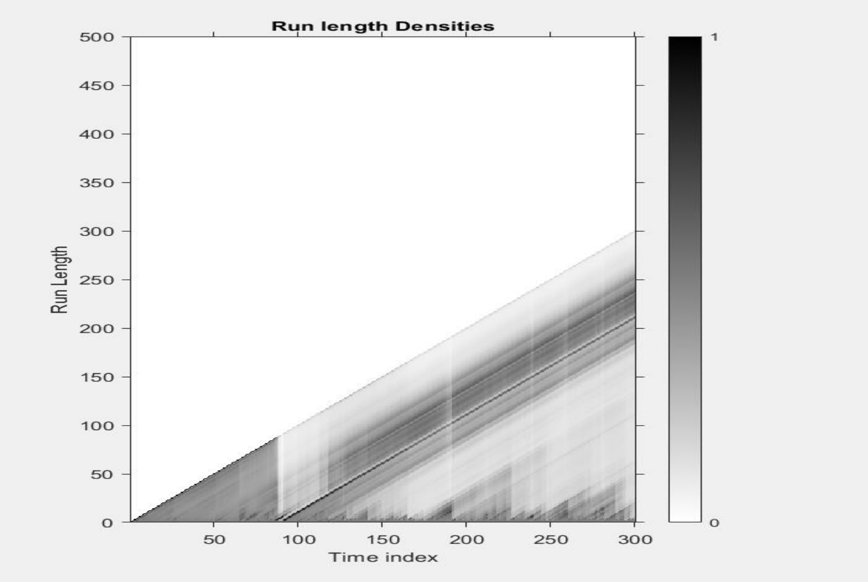

Figure 2.5: Run length posterior. Note that run length posterior after the first change-point near 100 are irrelevant for our analysis.

Model parameters (till 200 datapoints + historic data): -0.4019

Q2) b) Update the model, Build a new model & find var(driving force)

Updating the old model

We already performed rls for all data points during bocd computation. So we choose the theta estimate obtained at the data point with index (**change-point – 1)** and use it to compute the residuals.

The **AR(1) coefficient before corruption** of the channel is estimated as = **-0.4453**

Variance of the residuals is used as an estimator of the **Variance of the driving force.**

The residuals are found to be white, using the **Ljung-Box test,** confirming that the model is not an underfit.

Estimate of the variance of the endogenous white noise driving force of the system = 0.0984

Building a new model after the change point

We expect the AR(1) model to be followed again because it is said that the variance of the white noise innovations change and not the form of the model itself.

This is confirmed by the PACFs which go to zero for all lags > 1

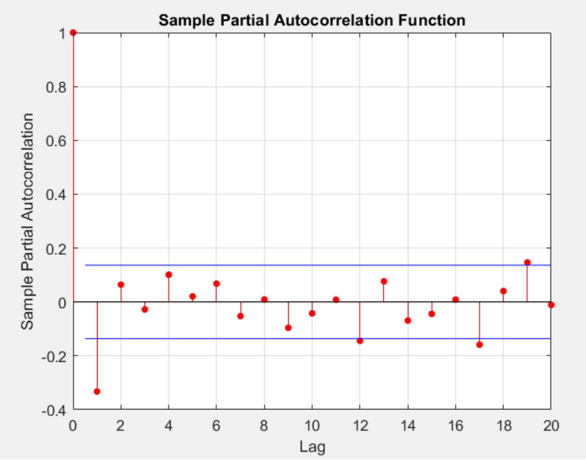

Figure 2.6: PACF of the data points after the change point

An AR(1) model is quickly built with again the constant term deemed insignificant.

```
Linear regression model:
    y ~ x1

Estimated Coefficients:
          Estimate       SE         tStat       pValue
          ________    ________    _______    __________

    x1    -0.33269    0.064747    -5.1384    6.2534e-07


Number of observations: 214, Error degrees of freedom: 213
Root Mean Squared Error: 0.569
```

The residuals are verified to be **white** using the **Ljung-Box test.**

The **AR(1) coefficient after corruption** of the channel is estimated as = **-0.3327**

Variance of the residuals is used as an estimator of the **Variance of the driving force.**

Estimate of the variance of the endogenous white noise driving force of the system = 0.3236

Summary:

Before Corruption:

Model: $y[k] = -0.4453*y[k-1] + \xi[k]$; $\xi[k] \sim GWN(0,\sigma^2)$; estimate of σ^2 =0.0984

After corruption:

Model: $y[k] = -0.3327*y[k-1] + \xi[k]$; $\xi[k] \sim GWN(0,\sigma^2)$; estimate of σ^2 =0.3236

References:

1. R. P. Adams and D. J. C. MacKay, “Bayesian Online Changepoint Detection”, arXiv:0710.3742 [stat], Oct. 2007.
2. K. P. Murphy, “Conjugate bayesian analysis of the gaussian distribution,” tech. rep., 2007.
3. http://gregorygundersen.com/blog/2019/08/13/bocd/
4. http://gregorygundersen.com/blog/2020/10/20/implementing-bocd/
5. https://github.com/gwgundersen/bocd/blob/master/bocd.py