CLICK REMOVAL IN DEGRADED AUDIO

Report for Module EEP55C22 Computational Methods

Ronald Melwin Laban, Student ID 22310290
Trinity College Dublin
labanr@tcd.ie

November 2022

This report is submitted in part fulfilment for the assessment required in EEP55C22 Computational Methods. I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year. These are found in Parts II and III at http://www.tcd.ie/calendar.

This report describes the algorithm designed for detection and removal of clicks in archived audio tracks. An example of a few clicks is shown in Figure 1. The sound manifests as a short sharp click or thump in the audio track. These degradation arise because of random points on the signal were converted to +1 or -1. Our problem is to identify the clicks(degradation) and restore the original clear signal back.

1 Background

Let us define the observed degraded signal as G_k and the original, clean signal as y_k. We can model the degradation as follows.

$$G_k = \begin{cases} r_k & \text{for random selected points} \\ y_k & \text{Otherwise} \end{cases} \tag{1}$$

Our problem is to detect and suppress the clicks(impulsive noise) using Auto- regressive model.

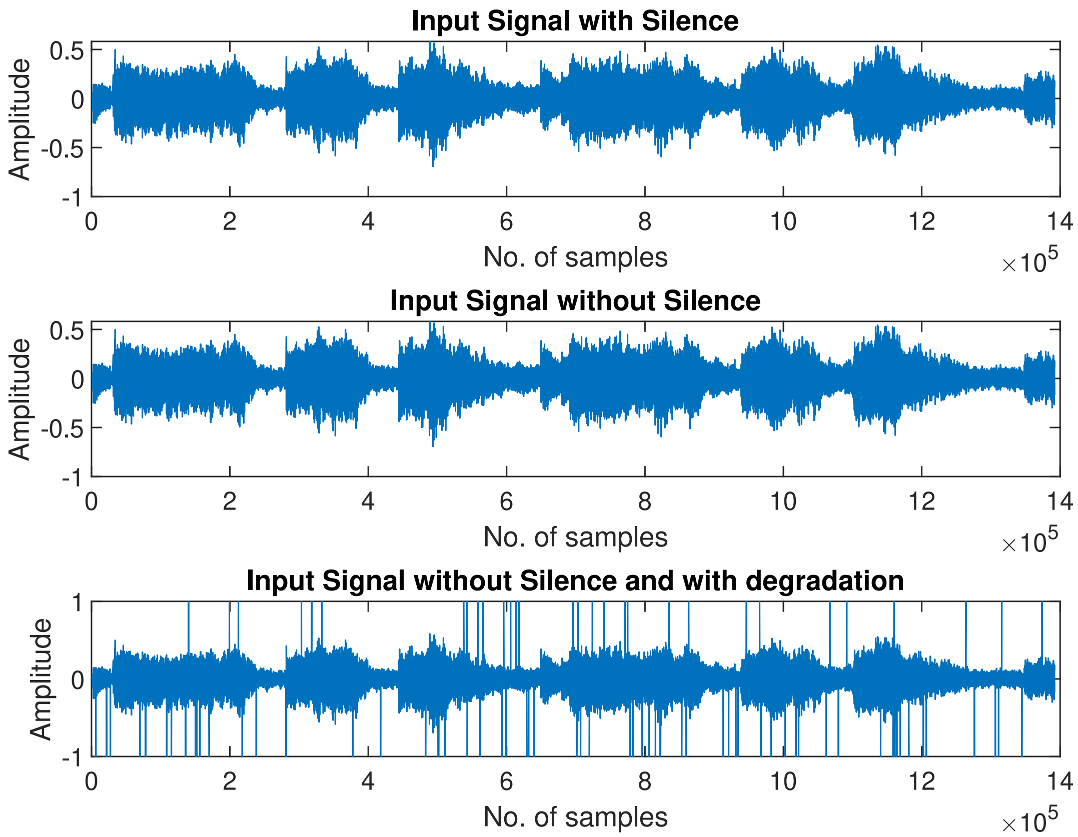

Figure 1: *Top : Original Input Signal. Middle : Input Signal without Silence. Bottom : Input signal without silence after passing through corruption mode*

1.1 Detection

In order to detect the click position, the first step is to AR coefficients so as to minimise the mean square error. It turns out that we can really solve this in closed form without using a numerical optimisation technique because the model is linear, i.e., the prediction is a weighted combination of data samples. By differentiating with respect to each coefficient, let's say a_j, we can solve for the least error by producing P (model order) simultaneous equations as shown in Equation 2.

$$\sum_{p=1}^{P} a_p \sum_{k=0}^{N-1} y_{k-p} y_{k-j} = \sum_{k=0}^{N-1} y_k y_{k-j} \tag{2}$$

This is usually summarised as **Ra = r**. We can then solve for a the coefficient vector using standard matrix solvers e.g. a = R^{-1}r

1.2 Interpolation

The key idea of interpolation is firstly to determine the residue for the given signal based on the AR coefficients obtained. These residue are used to obtain the position of the clicks based on the given threshold value and using interpolation these clicks are replaced by the value shown in Equation 3

$$y_u = -[A_u' A_u]^{-1} A_u' A_k y_k \tag{3}$$

where :
A_u = coefficient matrices corresponding to linear operations on the unknown data
A_k = coefficient matrices corresponding to linear operations on the known data
y_k = input matrices corresponding to linear operations on the known data

2 The algorithm

Our algorithm consist the following steps:

1. Input audio file is first processed to remove any silence in the signal.
2. The silence free audio is then passed through the degradation model, wherein random 100 locations are selected and their amplitude is made +1 or -1.
3. The processed degraded input is then broken into frame depending on the frame duration(in seconds).
4. Evaluate the AR coefficient

   (a) In order to calculate the AR coefficient, firstly create overlapping frames, wherein P(model order) number of elements will be overlapped from the previous frame to generate the next frame.
   (b) Once the overlap frames are being created, for processing we need to normalise the data for every frame.
   (c) Calculate R and r.
   (d) Calculate AR coefficient for the frame using Equation 2.

5. Evaluate the residue
   (a) For evaluation we will be using non overlapping data frames.
   (b) During evaluation every frame needs to be normalised.
   (c) Using the frame frame and AR coefficient calculate residue, the residue signals filters the noise impulses.
6. Evaluate the position of the click for every frame using the residue signals and threshold value set by the user.
7. Evaluate the new value from every frame in positions of the clicks using interpolation.
   (a) Evaluate the matrix A for every frame.
   (b) Extract A_u and A_y from A matrix.
   (c) Extract y_k from the data frame excluding the click position in the respective frame.
   (d) Evaluate y_u using Equation 3.
8. Evaluate the Mean Square Error.

3 Analysis and Results

The parameters for the above system are as follows:
Length of the audio signal = 1392000 samples
Length of the audio signal after silence removal = 1392000 samples
Model Order = 03
Frame Duration = 0.5 sec
Frame Size = 24000 samples
Threshold point = 0.3

To test our algorithm, first the clean audio signal was processed for silence removal and then passed through the degradation model to obtain degraded input signal as seen in Figure 1. For

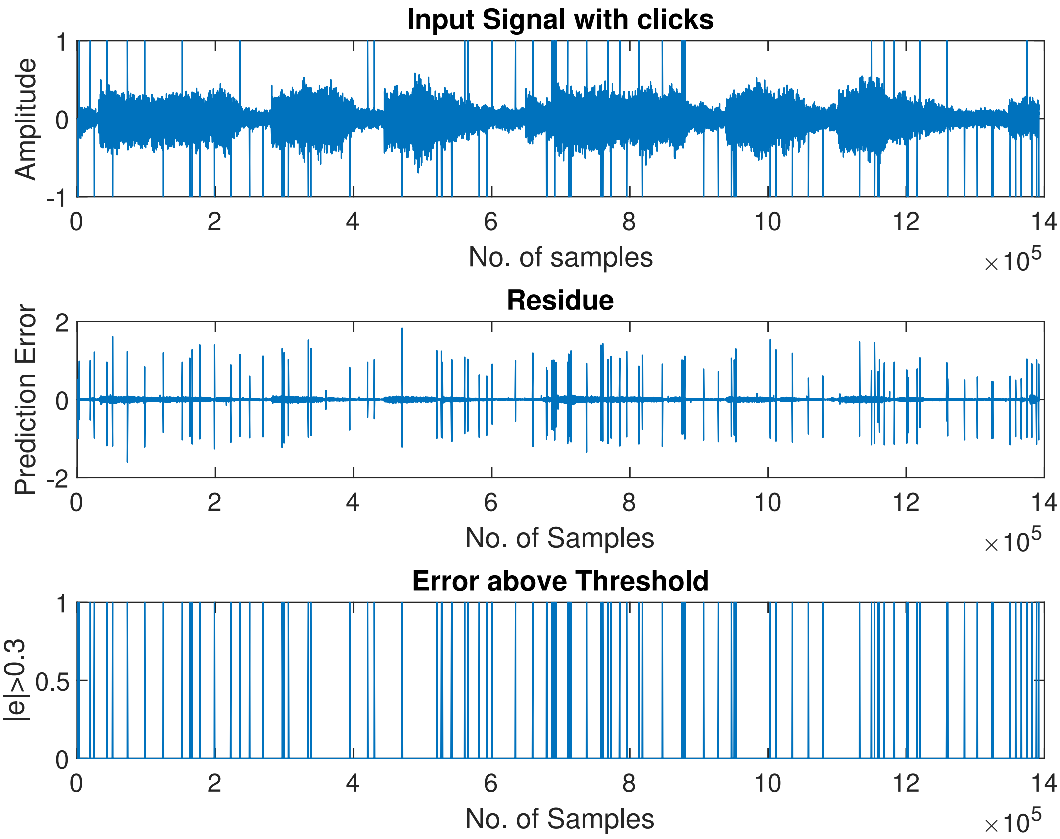

Figure 2: *Top : Degraded Signal. Middle : Residue for degraded signal. Bottom : Residue value above threshold*

the degraded signal the residue was calculated and the error above threshold was estimated based on the residue signal as seen in Figure 2 Following, that, position for clicks in every frame was estimated and replaced by the new value obtain from the interpolation function to obtain suppressed click audio signal as seen in Figure 3 For the above parameters, the MSE obtained was 1.7954e-08 and the total execution time was around 110 seconds.

In, Figure 4 , the graph on the left exhibits the relationship between threshold and mean squared error (MSE) wherein it is observed that initially the MSE didn't increase with the increase in the threshold value. However, between a threshold value of 0.75 and 1.25, an exponential increase was observed, as evident in the graph. Moreover, after a threshold value of 1.25, the value of MSE was found to be static. The graph on the right depicts the relationship between threshold and execution time in seconds. It is evident from the graph that as the execution time increases,

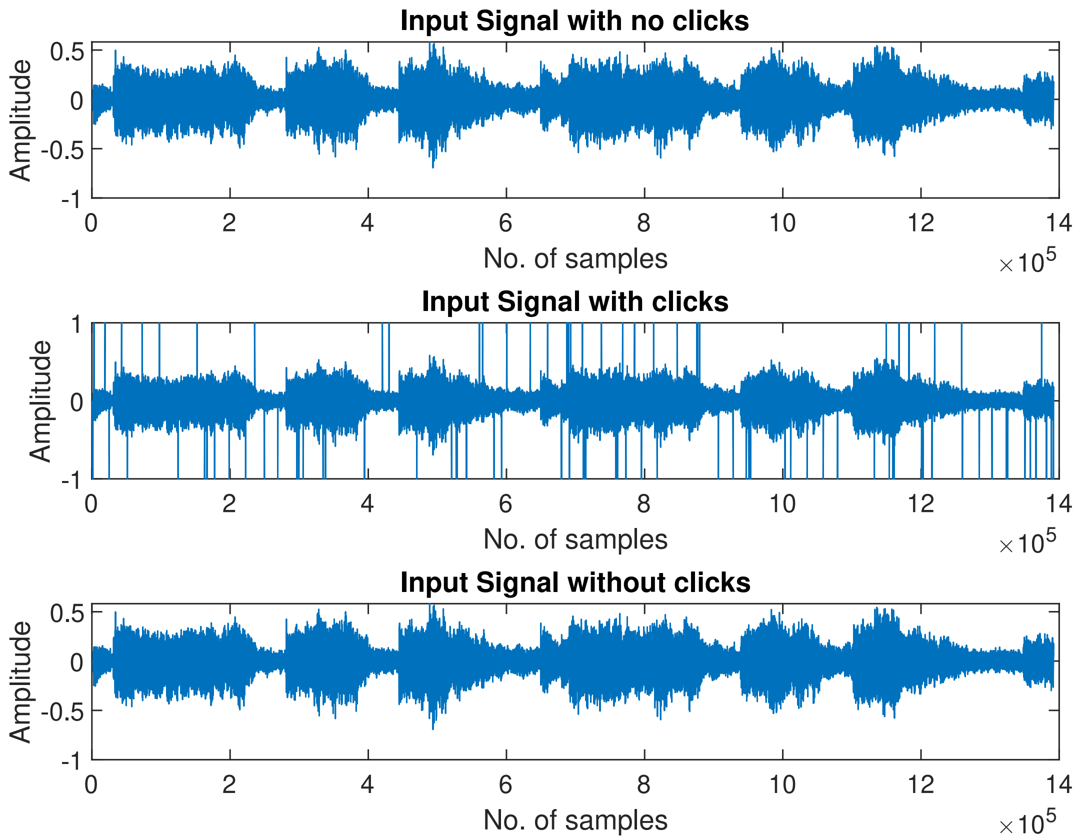

Figure 3: *Top : Original audio signal middle : Degraded audio signal with clicks. Bottom : Restored audio signal*

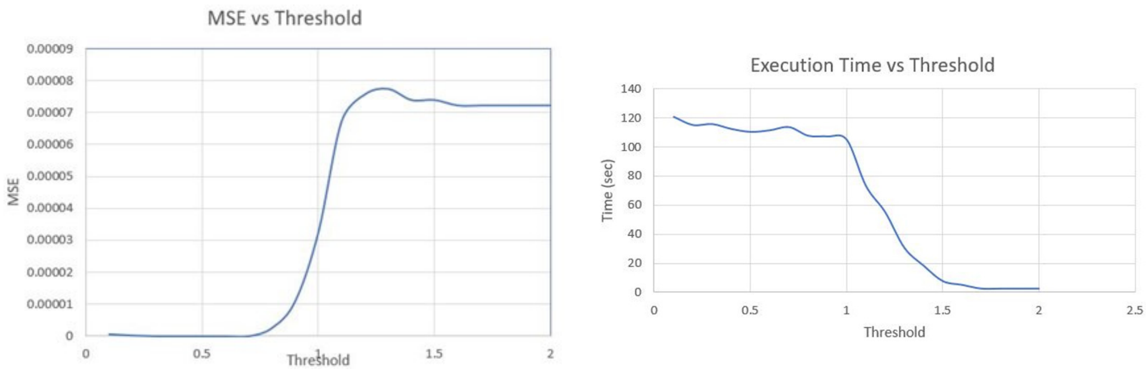

Figure 4: *Left : MSE vs Threshold Right : Execution Time vs Threshold (Model Order = 3, Block Size = 24000)*

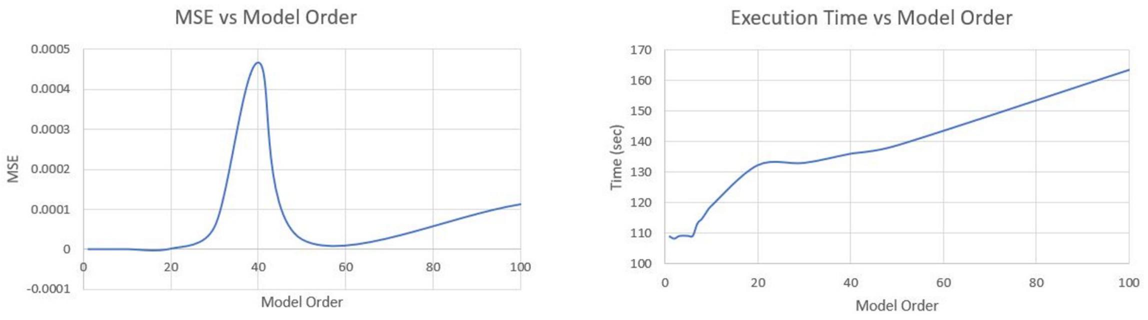

Figure 5: *Left : MSE vs Model Order : Execution Time vs Model Order (Threshold = 0.3, lock Size = 24000)*

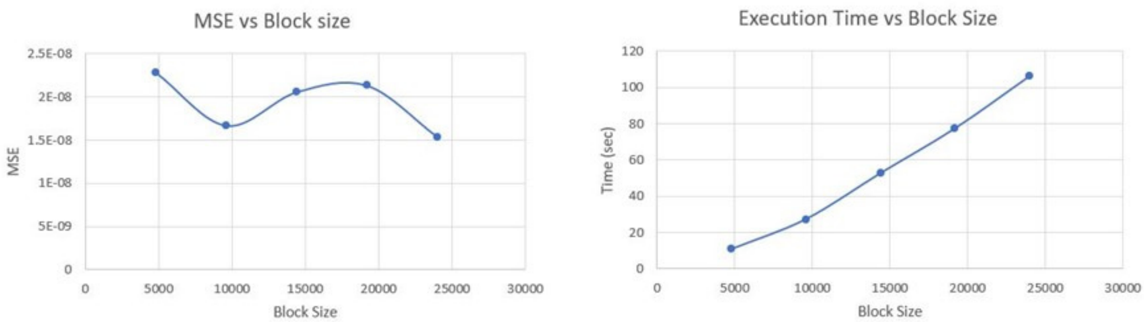

Figure 6: *Left : MSE vs Block Size : Execution Time vs Block Size (Model Order = 3, Threshold = 0.3)*

the threshold value decreases steadily, which exhibits an inverse relation.

In Figure 5, the graph of model order vs. MSE, a bell-shaped-kind of a curve is observed wherein MSE is maximum at model order 40, and then there is a steady decrease in MSE. An exponential graph is observed when a graph between model order and time is plotted. The graph continues to grow linearly where for a model order of 100 the time taken is >160 seconds. In Figure 6, the graph between block size and MSE depicts a sinusoidal-type curve with sufficient number of peaks and troughs in an alternative manner. For instance, at block size 5000, the MSE value lies between 2 × 10-8 and 2.5 × 10-8 whereas when the block size is 10000, the MSE value lies mainly just about 1.5 × 10-8. A linear graph is observed when block size is plotted w.r.t. time in seconds. The minimum time taken is 0 to 20 seconds for block size 5000, and block size 25000, the time taken is 100–120 seconds.

4 Conclusions

The project aimed to present a solution for click removal in case of noisy speech and musical signals. The algorithm used was on the basis of detection–interpolation scheme. From the analysis, it was evident that linear prediction systems were capable of modeling clean speech signals.Analysis were done on the basis of threshold point, model order and block size versus MSE and Execution time.

Bibliography