Kalman Filter in Speech Enhancement

Orchisama Das
Roll No. - 001211102017
Reg No. - 119953 of 2012-2013
Dept. of Instrumentation and Electronics Engineering
Jadavpur University

April, 2016

Final year project thesis submitted for the partial fulfilment of Bachelor's degree in Engineering (B.E.).

Supervised by Dr. Bhaswati Goswami and Dr. Ratna Ghosh.

Contents

Chapter 1

Introduction

1.1 Abstract

In this thesis, two topics are integrated - the famous MMSE estimator, Kalman Filter and speech processing. In other words, the application of Kalman filter in speech enhancement is explored in detail. Speech enhancement is the removal of noise from corrupted speech and has applications in cellular and radio communication, voice controlled devices and as a preprocessing step in automatic speech/speaker recognition. The autoregressive model of speech is used to formulate the state-space equations, and subsequently the recursive Kalman filter equations. Filter tuning, or optimum estimation of filter parameters, i.e. the process noise covariance and the measurement noise covariance, is studied in detail. New algorithms for determination of filter parameters are proposed. Lastly, the effect of changing model order is observed, and a novel algorithm is proposed for optimum order determination. These modifications are tested on speech data from the NOIZEUS corpus, which have been corrupted with different types of noise (white, train and babble) of different signal to noise ratios.

The rest of the thesis is organised as follows:

- The rest of Chapter 1 reviews past work and gives an introduction to the autoregressive model of speech, Kalman filter and its application in speech enhancement.
- Chapter 2 dives into filter tuning, and algorithms for determination of optimum values of Kalman filter parameters.
- Chapter 3 explores the topic of AR model order determination, and proposes an algorithm for it.
- Chapter 4 tests the algorithms proposed in this thesis on data from the NOIZEUS speech corpus and compares both the qualitative and quantitative results.
- Chapter 5 culminates the thesis and delineates the scope for future work.

1.2 Past Work

R. Kalman in his famous paper [1] proposed the Kalman filter to predict the unknown states of a dynamic system. In essence, it is a set of recursive equations that estimate the state of a system by minimising the mean squared error. Since then, it has had various applications in Robotics, Statistics, Signal Processing and Power Systems. A very good introduction to the Kalman filter is given by Welch and Bishop in [2]. The simple Kalman filter works on linear systems, whereas the Extended Kalman Filter (EKF) is needed for non-linear systems. This work concentrates on the Simple Kalman Filter.

The Autoregressive model assumes that at any instant, a sample depends on its past p samples added with a stochastic component, where p is the order of the model. Linear Predictive coding (LPC) [3] ties the AR model to speech production by proposing that speech can be modelled as an all-pole, linear, time varying filter excited by either an impulse train of a particular pitch or noise.

Paliwal and Basu [4] were the first to apply the Kalman filter in speech enhancement. They came up with the mathematical formulation of the state-space model and Kalman filter equations, and compared the results to the Wiener filtering method [5]. Since then, various modifications of their algorithm have been proposed, such as [6] where So et al. analysed the Kalman gain trajectory as an indicator of filter performance, and the utility of long, tapered overlapping windows in smoothing residual noise in enhanced output. Similarly, iterative Kalman filtering was proposed by Gibson et al. [7].

Filter tuning, or optimum estimation of Kalman filter parameters and its application in speech enhancement have been focused on very recently in [8]. The filter parameters to be estimated are the measurement noise covariance R and the process noise covariance Q. The determination of R is relatively simpler than the determination of Q as it depends on the noise corrupted measurement and not on the system model. One method of estimating R is given in [9] where the noise variance is calculated from the noisy AR signal with the aid of the Yule-Walker equations [10]. In [8], another method was proposed where the speech signal was broken into frames and each frame was categorised as silent or voiced according to its spectral energy below 2kHz. The measurement noise variance,R was given as the mean of variances of all silent frames. In this thesis, yet another algorithm has been proposed which utilises the Power Spectral Density [11] to distinguish between voiced and silent frames. It has been seen to give a more accurate estimation of R than any of the previous two methods.

The process noise covariance, Q is an inherent property of the process model. A novel method of determining Q was proposed by Saha et al. in [12] where they utilised two performance metrics - the sensitivity metric and the robustness metric and ensured a balanced root mean squared performance between them to give a compromise value of Q. They tested the methodology on a 2D falling body with the Extended Kalman Filter, and concluded superior results. In [8], a similar method was used to determine the process noise variance, but the value of Q and Kalman gain were toggled between voiced and silent frames. This ensured Kalman gain adjustment and improved results.

AR, MA and ARIMA processes, their fit to time-series data and model order determination have been studied in detail by Box and Jenkins [13]. They utilised the Autocorrelation function (ACF) and the partial autocorrelation function (PACF) to determine model order for MA and AR processes respectively. An overview of their algorithm can be found in any Time Series Analysis textbook such as [14]. In this thesis, optimal model order is determined from the Cumulative Absolute Partial Autocorrelation function (CPACF).

The tuned Kalman filter with optimum order determination leads to a novel speech enhancement algorithm that is tested by standard evaluation metrics. Some pitfalls of the algorithm such as increased time complexity and a compromise in noise removal to

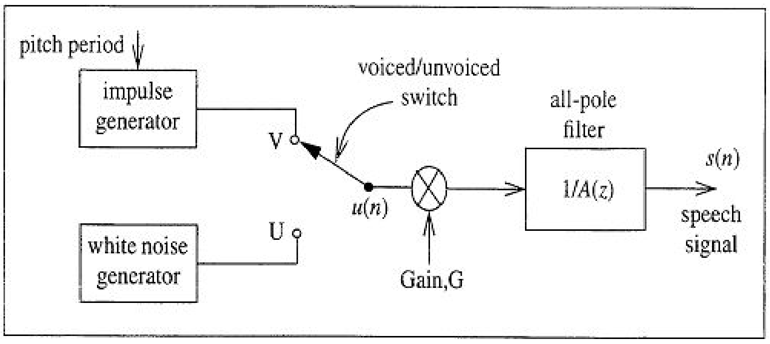


Figure 1.1: *Speech Production System*

preserve perceptual quality of speech are also discussed. In the next section, we introduce some of the concepts essential to this work.

1.3 Theory

In this section, the Autoregressive model of speech, Linear Prediction Coding, Yule-Walker equations and the Kalman Filter equations as applied to speech are discussed.

1.3.1 Auto-Regressive Model of Speech

Speech can be modelled as the output of a linear time-varying filter, excited by either quasi periodic pulses or noise. A schematic of the speech production model is given in figure 1.1.

A closer inspection of this system shows that speech can be modelled as a pth order autoregressive process, where the present sample, *x(k)* depends on the linear combination of past p samples added with a stochastic or random component that represents noise. In other words, it is an all-pole FIR filter with Gaussian noise as input.

$$x(k) = -\sum_{i=1}^{p} a_i x(k-i) + u(k) \tag{1.1}$$

where a_i are the linear prediction coefficients (LPCs) and $u(k)$, the process noise, is a zero-mean Gaussian noise with variance σ_u^2.

Linear Prediction Coding [3] is the prediction of LPCs. Linear Prediction Coding can be done by the autocorrelation method which makes use of the Yule-Walker equations. This process is explained in [10]. The Autocorrelation Function (ACF), R_{xx} at lag l is given by 1.2:

$$R_{xx}(l) = \mathbb{E}[x(k)x(k-l)] \tag{1.2}$$

1.1 can also be written as 1.3

$$\sum_{i=0}^{p} a_i x(k-i) = u(k); a_0 = 1 \tag{1.3}$$

Multiplying 1.3 with *x(k - l)* gives 1.4

$$\sum_{i=0}^{p} a_i \mathbb{E}[x(k-i)x(k-l)] = \mathbb{E}[u(k)x(k-l)] \tag{1.4}$$

The autocorrelation and cross-correlation terms can be identified and 1.4 can be rewritten as 1.5.

$$\sum_{i=0}^{p} a_i R_{xx}(l-i) = R_{ux}(l) \tag{1.5}$$

The cross-correlation term $R_{ux}(l)$ is 0 everywhere except at $l = 0$ where it equals σ_u^2. For $l > 0$, 1.5 can be rewritten as 1.6.

$$\sum_{i=1}^{p} a_i R_{xx}(l-i) = -R_{xx}(l) \tag{1.6}$$

In matrix form, it is expressed as 1.7

$$\begin{bmatrix} R_{xx}(0) & R_{xx}(-1) & \cdots & R_{xx}(1-p) \\ R_{xx}(1) & R_{xx}(0) & \cdots & R_{xx}(2-p) \\ \vdots & \vdots & \ddots & \vdots \\ R_{xx}(p-1) & R_{xx}(p-2) & \cdots & R_{xx}(0) \end{bmatrix} \times \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = - \begin{bmatrix} R_{xx}(1) \\ R_{xx}(2) \\ \vdots \\ R_{xx}(p) \end{bmatrix} \tag{1.7}$$

In vector form, the Linear Prediction Coefficients, **a** is given by 1.8:

$$\boldsymbol{a} = -\boldsymbol{R}^{-1}\boldsymbol{r} \tag{1.8}$$

1.3.2 Kalman Filter Equations

The Kalman Filter equations applied to the AR model of speech were first formulated by Paliwal and Basu in [4]. Before studying the Kalman filter equations, 1.1 is re-written in matrix form as 1.9.

$$\begin{bmatrix} x(k-p+1) \\ x(k-p+2) \\ \vdots \\ x(k) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -a_p & -a_{p-1} & -a_{p-2} & \cdots & -a_1 \end{bmatrix} \begin{bmatrix} x(k-p) \\ x(k-p+1) \\ \vdots \\ x(k-1) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} u(k) \tag{1.9}$$

or

$$\boldsymbol{X}(k) = \boldsymbol{\phi}\boldsymbol{X}(k-1) + \boldsymbol{G}u(k) \tag{1.10}$$

where $\boldsymbol{X}$(k) is the *(p×1)* state vector matrix, $\boldsymbol{\phi}$ is the *(p×p)* state transition matrix that uses LPCs calculated from noisy speech according to 1.8, $\boldsymbol{G}$ is the *(p×1)* input matrix and *u(k)* is the noise corrupted input signal at the kth instant.

When speech is noise corrupted, the output *y(k)* is given as:

$$y(k) = x(k) + w(k) \tag{1.11}$$

where $w(k)$ is the measurement noise, a zero-mean Gaussian noise with variance σ_w^2. In vector form, this equation may be written as

$$y(k) = \boldsymbol{H}\boldsymbol{X}(k) + w(k) \tag{1.12}$$

where $\boldsymbol{H}$ is the *(1×p)* observation matrix given by

$$H = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \tag{1.13}$$

The Kalman filter calculates $\hat{\boldsymbol{X}}$(k|k) which is the estimate of the state vector $\boldsymbol{X}$(k), given corrupted speech samples upto instant k, by using the following equations:

$$\hat{\boldsymbol{X}}(k|k-1) = \boldsymbol{\phi}\hat{\boldsymbol{X}}(k-1|k-1) \tag{1.14}$$

$$\boldsymbol{P}(k|k-1) = \boldsymbol{\phi}\boldsymbol{P}(k-1|k-1)\boldsymbol{\phi}^{\boldsymbol{T}} + \boldsymbol{G}Q\boldsymbol{G}^{\boldsymbol{T}} \tag{1.15}$$

$$\boldsymbol{K}(k) = \boldsymbol{P}(k|k-1)\boldsymbol{H}^{\boldsymbol{T}}(\boldsymbol{H}\boldsymbol{P}(k|k-1)\boldsymbol{H}^{\boldsymbol{T}} + R)^{-1} \tag{1.16}$$

$$\hat{\boldsymbol{X}}(k|k) = \hat{\boldsymbol{X}}(k|k-1) + \boldsymbol{K}(k)(y(k) - \boldsymbol{H}\hat{\boldsymbol{X}}(k|k-1)) \tag{1.17}$$

$$\boldsymbol{P}(k|k) = (\boldsymbol{I} - \boldsymbol{K}(k)\boldsymbol{H})\boldsymbol{P}(k|k-1) \tag{1.18}$$

where

- $\hat{\boldsymbol{X}}$(k|k-1) is the *a priori* estimate of the current state vector $\boldsymbol{X}$(k).
- $\boldsymbol{P}$(k|k-1) is the error covariance matrix of the *a priori* estimate, given by $\boldsymbol{E}[\boldsymbol{e}_k^- \boldsymbol{e}_k^{-T}]$ where $\boldsymbol{e}_{\boldsymbol{k}}^-$=$\boldsymbol{X}$(k)-$\hat{\boldsymbol{X}}$(k|k-1).
- Q is the process noise covariance matrix,which in this case is σ_u^2. Similarly, R is the measurement noise covariance matrix, which is σ_w^2.
- $\hat{\boldsymbol{X}}$(k|k) is the *a posteriori* estimate of the state vector. In our case, the last component of $\hat{\boldsymbol{X}}$(k|k) is $\hat{x}(k)$, which gives the final estimate of the processed speech signal.
- $\boldsymbol{P}$(k|k) is the error covariance matrix of the *a posteriori* estimate, given by $\boldsymbol{E}[\boldsymbol{e}_k \boldsymbol{e}_k^T]$ where $\boldsymbol{e}_{\boldsymbol{k}}$=$\boldsymbol{X}$(k)-$\hat{\boldsymbol{X}}$(k|k).
- Let $\hat{\boldsymbol{X}}$(0|0)=[*y(1)* $\cdots$ *y(p)*] and $\boldsymbol{P}(0|0) = \sigma_w^2\boldsymbol{I}$, where $\boldsymbol{I}$ is the *(p×p)* identity matrix.
- $\boldsymbol{K}$(k) is the Kalman gain for the kth instant. The term *y(k)* - $\boldsymbol{H}\hat{\boldsymbol{X}}$(k|k-1) is known as the *innovation.*

Equations 1.14 and 1.15 are known as the *time update* equations whereas 1.16, 1.17, 1.18 are known as the *measurement update* equations. Intuitively, the Kalman filter equations may be explained thus: The gain $\boldsymbol{K}$(k) is chosen such that it minimizes the *a posteriori* error covariance, $\boldsymbol{P}$(k|k). As $\boldsymbol{P}$(k|k-1) decreases, $\boldsymbol{K}$(k) reduces. An inspection of 1.17 shows that as $\boldsymbol{K}$(k) reduces, the *a priori* state estimate is trusted more and more and the noisy measurement is trusted less.

In this chapter the Autoregressive model of speech, Linear Predictive Coding and the Kalman Filter have been elucidated. In the next chapter, filter tuning or optimum parameter estimation will be discussed.

Chapter 2

Filter Tuning : Estimation of Optimum Filter Parameters

The two filter parameters that need to be tuned are the measurement noise covariance, R in 1.16 and the process noise covariance, Q in 1.15. Accurate estimation of these parameters can greatly enhance filter performance. This chapter will explain algorithms for optimum estimation of R and Q. It is to be noted that for the AR model of speech, R and Q are scalar quantities the values of which are the variances of process noise (σ_u^2), and measurement noise (σ_w^2) respectively.

2.1 Measurement Noise Covariance, *R*

The measurement noise covariance, R, is the variance of the noise corrupting the speech, σ_w^2. In [9], the autocorrelation function of noisy measurement was used to derive the following equation:

$$\sigma_w^2 = \frac{\sum_{i=1}^{p} a_i[R_{yy}(i) + \sum_{k=1}^{p} a_k R_{yy}(|i-k|)]}{\sum_{i=1}^{p} a_i^2} \tag{2.1}$$

In [8], we proposed an even simpler method where we divided the speech signal into 80ms frames with 10ms overlap, and classified each frame as silent or voiced depending on its spectral energy content based on the following criterion for silent frames:

$$E(i) < \frac{max(E)}{100} \tag{2.2}$$

where where *E(i)* is the energy of spectral components below 2kHz for the ith frame and *E*=[*E(1), E(2),* $\cdots$ *E(n)*] is the set of spectral energy components below 2kHz for all frames. In order to consider a single value of R for the total speech signal, the mean of variances of all silent frames was taken as R. This is because silent frames contain only the measurement noise, without any speech components.

It was observed that 2.1 gave a value of R which was too high, whereas 2.2 gave a value of R that was lesser than the actual value. As a result, the results of filtering with either value of R were not satisfactory. This led to the formulation of a new algorithm to classify silent and voiced regions in speech that is explained in the next section.

2.1.1 Power Spectral Density

It has been shown that the first step for determining R is the classification of voiced and silent regions in speech. A very common method of voiced/unvoiced classification relies on the Zero-Crossing Rate (ZCR) [15]. Generally, voiced regions have a much higher ZCR than unvoiced regions. This is true for clean speech signals. However, noise itself has a very high ZCR. In noisy speech, the silent regions have pure noise, and hence a high ZCR, which makes it impossible to distinguish between voiced and unvoiced regions using this method. As a result, a different method of frame classification is needed.
Before discussing the novel algorithm for measurement of R, it is important to discuss the power spectral density (PSD) [16] of a signal. PSD is the Fourier transform of the autocorrelation function (ACF) given by 2.3.

$$S(f) = \int_{-\infty}^{+\infty} R_{xx}(\tau) exp(-2\pi j f \tau) d\tau \tag{2.3}$$

White noise is an uncorrelated process, and hence, its autocorrelation function is zero everywhere except at $\tau = 0$ where it is equal to the variance of noise, i.e,

$$\begin{aligned} R_{ww}(\tau) &= 0,\ \tau \neq 0; \\ &= \sigma_w^2,\ \tau = 0 \end{aligned} \tag{2.4}$$

OR

$$R_{ww}(\tau) = \sigma_w^2 \delta(\tau) \tag{2.5}$$

where $\delta(\tau)$ is the Dirac-delta function which is 1 at τ=0, 0 otherwise. The Fourier transform of the ACF of white noise is its PSD, which is given by a uniform distribution over all frequencies.

$$S(f) = \sigma_w^2,\ -\infty < f < +\infty \tag{2.6}$$

Intuitively, this means that white noise contains all possible frequencies. This is analogous to white light, which is composed of all wavelengths. On the other hand, if we had a pure tone of 440Hz, the power spectrum, or PSD would contain a sharp spike at a frequency of 440Hz, just like its frequency spectrum.

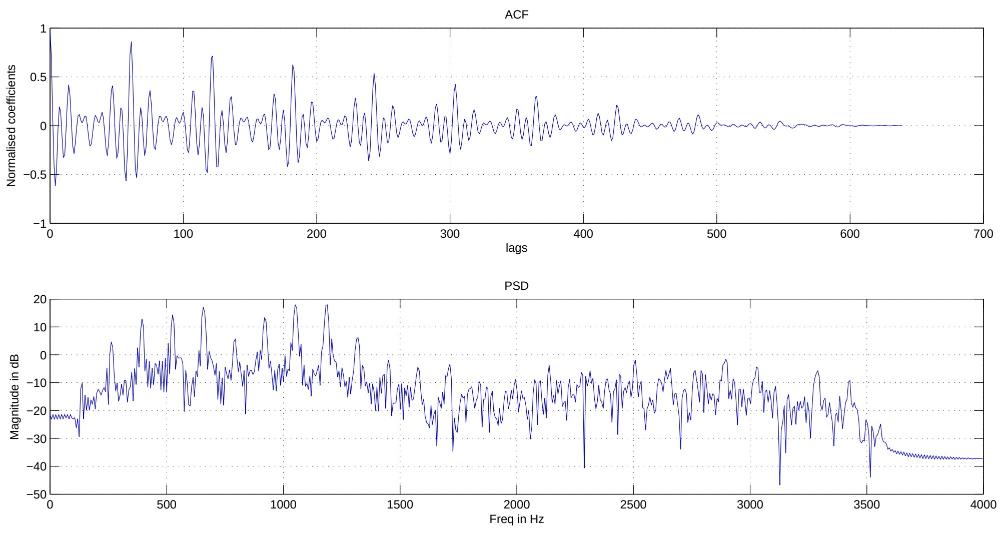
In general, the power spectrum of any kind of noise apart from white noise is fairly flat but band-limited. Therefore the power spectrum of silent regions in noisy speech will be flat but the power spectrum of voiced regions will contain peaks at the fundamental frequency and its and harmonics. Even in noise corrupted speech, the peaks in the power spectrum can still be easily distinguished.

To classify voiced and silent frames, the spectral flatness [17] is calculated as the ratio of the geometric mean to the arithmetic mean of the power spectrum.

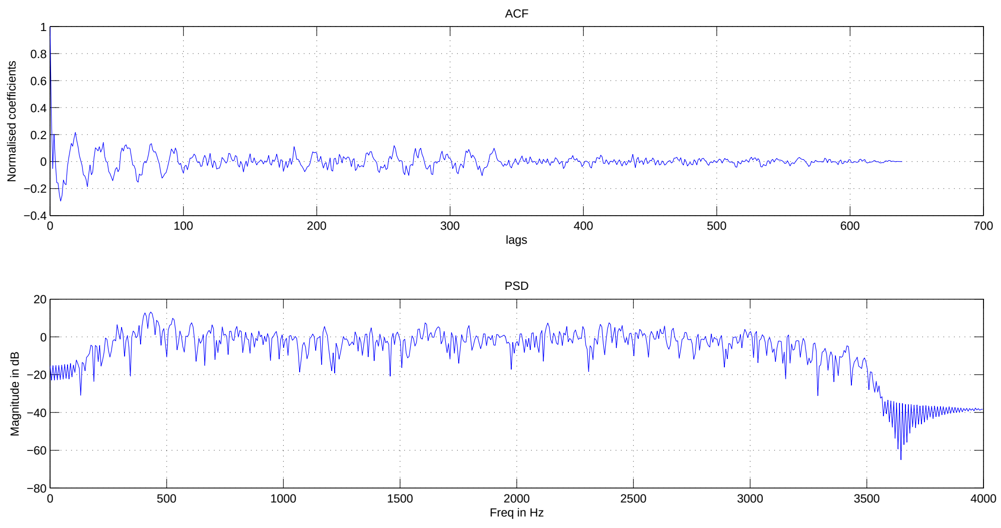
$$Flatness = \frac{\sqrt[N]{\prod_{n=0}^{N-1} x(n)}}{\frac{1}{N}\sum_{n=0}^{N-1} x(n)} \tag{2.7}$$

where *x(n)* represents the magnitude of the nth bin in the power spectrum.

It is observed that the spectral flatness of white noise is equal to 1, and for other noises it has a value close to 1. For a pure tone, spectral flatness is 0. Figure 2.1 shows the ACF and PSD plots for a voiced frame and a silent frame. The flat power spectrum of a silent frame gives a high value of spectral flatness close to 1, whereas the peaks in the power spectrum of a voiced frame give a low value of spectral flatness close to 0. The ACF of a silent frame has the highest value at lag 0, and is close to zero at all other lags. The ACF of a voiced frame is composed of additive sines.



(a) ACF and PSD of voiced frame



(b) ACF and PSD of silent frame

Figure 2.1: Autocorrelation function and Power Spectral Density

Using this observation, the algorithm for determination of R is summarised as follows:

i) The speech signal broken into frames of 80ms each with 10ms overlap.

ii) For each frame, the ACF and the PSD are calculated. Only the last $N/2$ samples are preserved as both these functions are even symmetric.

iii) The PSD is truncated and only values in the frequency range [100Hz,2000Hz] are kept. This limit is chosen because most of the spectral components of human speech lies in this frequency range.

iv) The spectral flatness is calculated according to 2.7 and normalised so that it lies between [0,1].

v) A threshold, $th = 0.707$ ($\frac{1}{\sqrt{2}}$) is chosen. Any frame with spectral flatness below th is classified as voiced and any frame with spectral flatness above th is classified as silent.

vi) Measurement noise variance, R is calculated as the maximum of the variances of all silent frames.

2.2 Process Noise Covariance, $\boldsymbol{Q}$

The process noise covariance, Q, is harder to determine accurately as it arises from the process model. In [12] the authors chose filter parameters that would provide a balanced RMSE performance between robustness and sensitivity. To do this, they defined the sensitivity and robustness metrics, J_1 and J_2 respectively, and from them determined the compromise value of $Q = Q_c$. Their algorithm was adopted in [8] where it was modified for the linear AR speech model. Additionally, two values of Q were used, Q_c for voiced frames and Q_2 (slightly less than Q_c) for silent frames. It was observed that a higher Kalman gain for voiced frames and a lower Kalman gain for silent frames was desirable, and toggling between two values of Q allowed Kalman gain adjustment.

2.2.1 Sensitivity and Robustness Metrics

The method described in this sub-section is exactly similar to that in [8]. Let two terms A_k and B be defined for a particular frame as

$$\begin{aligned} A_k &= \boldsymbol{H}(\boldsymbol{\phi}\boldsymbol{P}(k-1|k-1)\boldsymbol{\phi}^{\boldsymbol{T}})\boldsymbol{H}^{\boldsymbol{T}} \\ B &= \boldsymbol{H}(\boldsymbol{G}Q\boldsymbol{G}^{\boldsymbol{T}})\boldsymbol{H}^{\boldsymbol{T}} = \sigma_u^2 = Q_f \end{aligned} \tag{2.8}$$

In case of the speech model, the term A_k denotes the kth instant of the *a priori* state estimation error covariance while B represents the kth instant estimate of the process noise covariance in the measured output. Furthermore, in our case A_k, B and R are all scalars. R is constant for all frames because it is the variance of the noise corrupting the speech signal. However, B, though constant for a particular frame, is varied from frame to frame in order to capture the process dynamics. This choice of the *framewise constant* B is done using the performance metrics as discussed hereafter.

The two performance metrics J_1, J_2 and a controlling parameter, n_q as given in [12], are defined in this case as:

$$\begin{aligned} J_1 &= [(A_k + B + R)^{-1}R] = \frac{\sigma_w^2}{A_k + \sigma_u^2 + \sigma_w^2} \\ J_2 &= [(A_k + B)^{-1}B] = \frac{B}{A_k + B} = \frac{\sigma_u^2}{A_k + \sigma_u^2} \\ n_q &= log_{10}(B) = log_{10}(\sigma_u^2) \end{aligned} \tag{2.9}$$

Any mismatch between the assumed process noise covariance σ_u^2 and the actual process noise covariance is due to error in modelling, hence J_2, which is dependent on σ_u^2 is termed as the robustness metric. Similarly, any mismatch between actual R of the measurement and assumed R adversely affects the *a posteriori* estimate. Since it is reflected in J_1, it is termed as the sensitivity metric.

Let the process noise variance, σ_u^2 for a frame be denoted as Q_f. For each frame of speech, a nominal value of $Q_f = Q_{f-nom}$ is taken for initial calculation. This Q_f is then varied as $Q_{f-nom} \times 10^n$ where $n \in \boldsymbol{Z}$. Hence, $n_q = n \times \log_{10} Q_f$ and so, in this case, the metrics are obtained in terms of changing n instead of n_q. For each value of n, corresponding Q_f, J_1 *and* J_2 values are determined.

The typical plot of the metrics J_1 and J_2 for one voiced frame and one silent frame is shown in Fig 2.2.

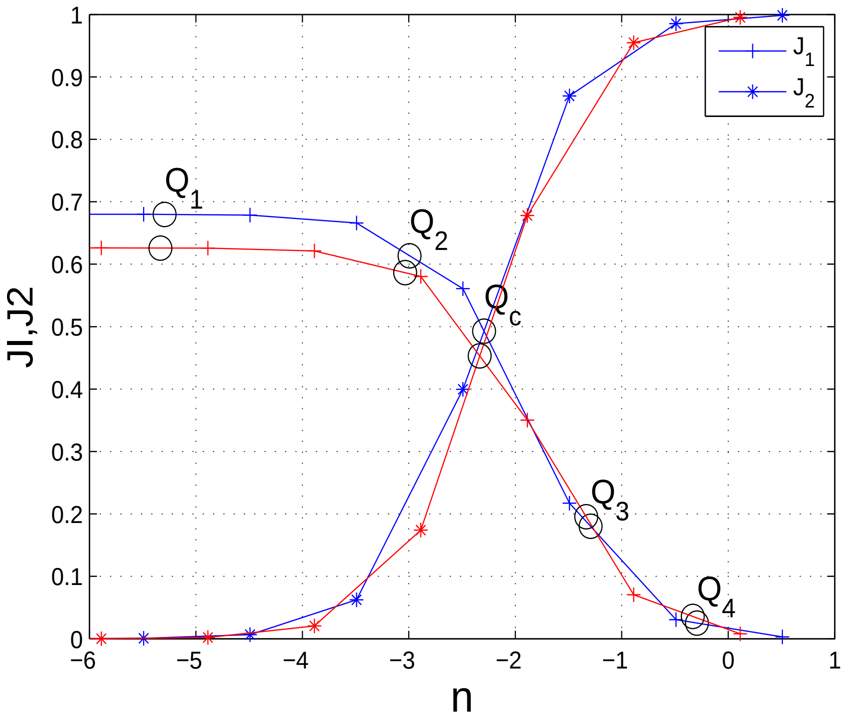


Figure 2.2: J_1, J_2 *v/s* n plot for a i) voiced frame (blue) ii) silent frame (red)

If the value of Q_f is increased such that it exceeds R substantially, then from 2.9, we can say that J_1 reduces to zero while J_2 is high. On the other hand if Q_f is decreased to a small value, then J_2 reduces to zero and J_1 is high, as evident in the graph.

Thus, robust filter performance may be expected for large values of Q_f, whereas small values of Q_f give sensitive filter performance. A trade-off between the two can be achieved by taking the working value of Q_f as the intersection point of J_1 and J_2. In Fig. 2.2, five values of Q_f have been marked in increasing order, with Q_1 being the lowest and Q_4 being the highest. Q_c is the value of Q_f at intersection of J_1 and J_2.

2.2.2 Kalman Gain

In [8], the Kalman gain's dependence on Q and its effect on filter performance was studied, and the Kalman gain trajectory was manipulated to give superior performance. Equation

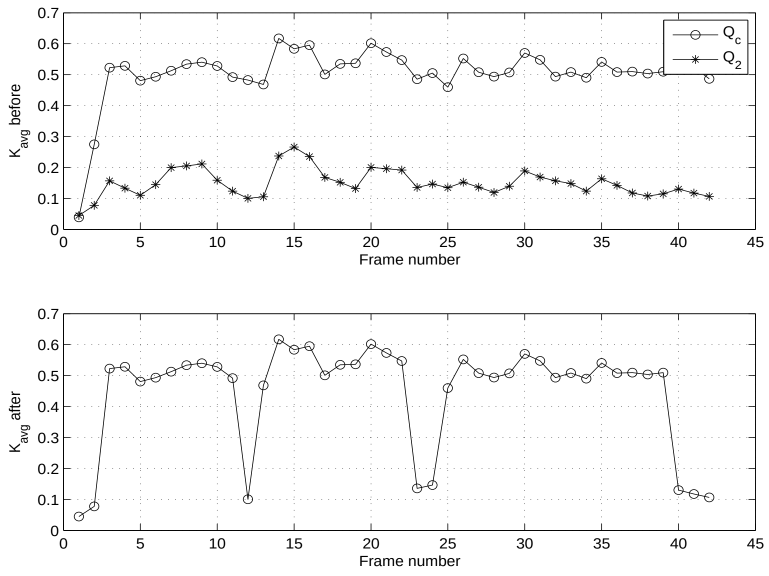


Figure 2.3: Kalman gain curve i) before adjustment ii) after adjustment

1.18 can be simplified in scalar form as:

$$\hat{x}(k|k) = K_k y(k) + (1 - K_k)\hat{x}(k|k-1) \tag{2.10}$$

A high value of Kalman gain indicates that the *aposteriori* estimate borrows heavily from the noisy input. A low value of gain indicates that the *aposteriori* estimate relies more on the *apriori* estimate. This information, along with the fact that K varies directly with Q can be used for Kalman gain adjustment. In voiced frames, we would ideally like to retain as much information as possible from the original noisy speech, hence a high value of Kalman gain is desirable. On the other hand, silent frames, which are composed purely of noise, should have a low value of Kalman gain. This is because the output should borrow as little as possible from the noise, and more from the *apriori* estimate.

The gain adjustment is done by selecting $Q = Q_c$ for voiced frames and $Q = Q_2$ $(< Q_c)$ for silent frames. This ensures that voiced frames have a high Kalman gain whereas silent frames have low Kalman gain as depicted in figure 2.3.

In this chapter, Kalman filter parameter tuning has been explained in detail, and algorithms for optimum determination of R and Q have been suggested, and the role of Kalman gain has been explained. In the next chapter, we will explore the topic of AR model order determination and its effect on filter performance.

Chapter 3

Model Order Determination

For most applications of speech processing, AR model order is fixed to be in the range of **10-15**. However, in [18], Rabiner says, *"The simplified all pole model is a natural representation of non-nasal voiced sounds, but for nasal and fricative sounds the detailed acoustic theory calls for both poles and zeros in the vocal tract transfer function. We shall see, however, that if order p is high enough, the all-pole model provides a good representation for almost all sounds of speech."* The same issue is elaborated in [19] where the authors propose a *reflection coefficient cutoff* (RCC) heuristic that can be used to determine quickly the best filter order for either a corpus of vowels or for a single vowel. Moreover, they discuss the effects of choosing incorrect filter order thus: *"If the filter order is too low, the formant*[1] *peaks are smeared or averaged; if it is too high, the estimated formant locations are biased towards the F0 harmonics. In the worst case, an inappropriate filter order can lead to spurious formant peaks or to formants being missed altogether."*

The need for model order determination is obvious. In this thesis, standard time-series analysis techniques [13] are used for AR model order determination with the help of the Partial Autocorrelation Function (PACF) that is explained in the next section.

3.1 Partial Autocorrelation Function

As the name suggests, the Partial Autocorrelation Function is derived from the Autocorrelation Function. Autocorrelation is the correlation or dependence of a variable with itself at two points in time that depends on the lag between them. Let there be a variable y whose value is y_t at time instant t. The autocorrelation between y_t and y_{t-h} at lag h would depend linearly on $y_1, y_2 \cdots y_{t-h+1}$. However, the partial autocorrelation between y_t and y_{t-h} is the autocorrelation between them with the linear dependence on $y_1, y_2 \cdots y_{t-h+1}$ removed.

The autocorrelation of y_t at lag h is given by:

$$\begin{aligned}\sigma_h &= \frac{\mathbb{E}[(y_t-\mu)(y_{t-h}-\mu)]}{\sigma^2} \\ &= \frac{\gamma(h)}{\gamma(0)}\end{aligned} \tag{3.1}$$

where μ is the mean, σ is the standard deviation and γ*(h)* is the autocovariance at lag *h*.

[1]In speech, formants are the vocal tract resonances that appear as peaks in the frequency spectrum

The partial autocorrelation at lag h is denoted by ϕ_h which is the last component of:

$$\boldsymbol{\phi_h} = \Gamma_h^{-1}\boldsymbol{\gamma_h} \tag{3.2}$$

OR

$$\begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_h \end{bmatrix} = \begin{bmatrix} \gamma(0) & \gamma(-1) & \cdots & \gamma(1-h) \\ \gamma(1) & \gamma(0) & \cdots & \gamma(2-h) \\ \vdots & \vdots & \ddots & \vdots \\ \gamma(h-1) & \gamma(h-2) & \cdots & \gamma(0) \end{bmatrix}^{-1} \times \begin{bmatrix} \gamma(1) \\ \gamma(2) \\ \vdots \\ \gamma(h) \end{bmatrix} \tag{3.3}$$

Not surprisingly, these equations resemble the Yule-Walker equations in Section 1.3.1. In fact, the same set of equations are used to estimate LPCs and PACF. It is to be noted that only the last element of $\boldsymbol{\phi}_h$ is the partial autocorrelation coefficient at lag h.

3.1.1 PACF of an AR(p) Process

We know, a causal AR(p) process can be defined as:

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots \phi_p y_{t-p} + z_t;\ z_t \sim WN(0, \sigma^2) \tag{3.4}$$

According to [14], for $h \geqslant p$, the best linear predictor of $\hat{y}_{h+1}$ in terms of $y_1,\ y_2,\ \cdots\ y_h$ is given by:

$$\hat{y}_{h+1} = \phi_1 y_h + \phi_2 y_{h-1} + \cdots + \phi_p y_{h+1-p} \tag{3.5}$$

The coefficient of y_1 is ϕ_p if $h = p$ and 0 for $h > p$. This indicates that the PACF for lag $h > p$ is zero. Intuitively, we can explain it thus : y_t and y_{t+h} are uncorrelated if they are independent. In an AR(p) process, for $h > p$, y_{t+h} does not depend on y_t (it only depends on the past p samples). Hence, PACF for lag $h > p$ is zero.

For determining model order from PACF, a boundary of $\pm 1.96/\sqrt{N}$ is imposed on the PACF plot, where N stands for the number of samples. The last lag, p, beyond which the PACF lies within the limits of $\pm 1.96/\sqrt{N}$ is chosen as the optimum model order.

3.1.2 Example : PACF of Random Walk

To understand this better let's take the help of a random walk signal, which is an AR(1) process whose probability distribution is given by:

$$\begin{aligned} f(x) &= \frac{1}{2};\ x = \pm 1 \\ &= 0\ otherwise \end{aligned} \tag{3.6}$$

This means that a person walking in a straight line can randomly go left or right from his current position in his next step. This can be generated computationally very easily by taking the cumulative sum of a random distribution of -1 and +1 only. The random walk signal of length = 100 samples, its PACF and ACF are plotted in figure 3.1.

It is observed that the PACF plot falls within the bounds $\pm 1.96/\sqrt{N}$ after lag 1 indicating that random walk is an AR(1) process. However, the ACF plot does not satisfy the same conditions, asserting that it is the PACF, not the ACF that should be used to determine model order of an AR process.[2]

[2]For MA processes, the ACF is used to determine model order, not the PACF.

3.2 Cumulative Absolute Partial Autocorrelation Function

So far in this chapter, we have established that the PACF is needed for accurate model order determination of an AR process. However, for noise corrupted speech, the boundary condition that was described earlier to determine order from PACF cannot be used because the PACF plot has some outliers at very high lags. Obviously, these are spurious values that should be eliminated.

To overcome this problem, instead of relying on the PACF plot, we calculate the Cumulative Absolute Partial Autocorrelation Function (CPACF), which is given by the equation:

$$CPACF(l) = \sum_{i=1}^{l} |PACF(i)| \tag{3.7}$$

In figures 3.2, 3.3 and 3.4, PACF and CPACF of speech corrupted with three different types of noise are plotted: *white, train* and *babble.* The plots for each kind of noise are discussed in the following subsections.

3.2.1 White noise

The PACF and CPACF plots for speech corrupted with white noise are given in figure 3.2. For voiced frames, as shown in plot 3.2a, the CPACF function grows rapidly before saturating. The lag at which saturation begins to set in should be the optimum model order. Beyond this lag, the PACF can be imagined to lie within certain bounds, and therefore has converged. The lag at which PACF converges (or CPACF saturates) is quite high (∼50), yielding a substantially high model order. The CPACF plot of the silent frame, plot 3.2a tells a different story. From the plot 3.2b, we can conclude that white noise is an AR(0) process, which makes sense because the samples in a random distribution are uncorrelated. As a result, the CPACF plot of a silent frame does not saturate but keeps on increasing as a linear function of lags.

3.2.2 Train

Figure 3.3 shows the PACF and CPACF plots for speech corrupted with noise from a moving train. CPACF of both silent and voiced frames saturate, unlike the case of white noise where CPACF of silent frames did not saturate. As seen in plot 3.3a, voiced frames saturate more quickly at a relatively lower lag, yielding an order ~ 30. Silent frames which have pure noise, are slower to saturate, giving an order ~ 40. Both plots seem to resemble the logarithm curve as a function of the number of lags, indicating that the PACF function definitely converges for higher lags, at a rate much faster than that of white noise.

3.2.3 Babble

PACF and CPACF plots of speech corrupted with babble[3] noise are shown in figure 3.4. The nature of the CPACF plots of both voiced and silent frames strongly resembles those of figure 3.3. However, the difference between CPACF plots of silent frame of babble noise

[3]a crowd of people talking in the background

in 3.4b and train noise in 3.3b is distinct. Babble is a complex, band-limited, coloured noise with characteristics very different from statistical white noise. Its CPACF converges quickly, yielding a lower model order. Train noise resembles white noise somewhat more, and the difference can be inferred audibly. Hence, its CPACF saturates at a higher lag.

3.2.4 Proposed Algorithm for Order Determination

Regardless of the nature of the frame (voiced/silent), optimum model order of each frame is determined in the following way:

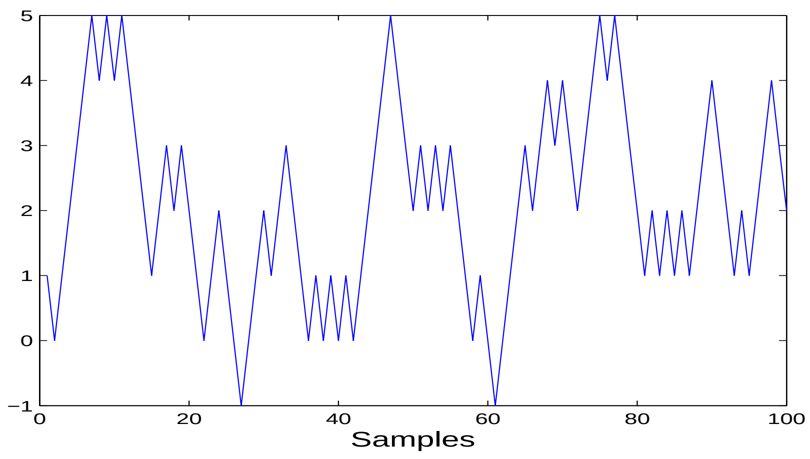
- The PACF is calculated for 100 lags (we assume that the maximum possible order cannot exceed 100).
- The CPACF is calculated according to 3.7.
- The saturation value of CPACF is taken as $CPACF_{sat} = 0.7 \times range(CPACF)$. The lag corresponding to $CPACF_{sat}$ is determined to be the model order for that particular frame.

Notes on Implementation

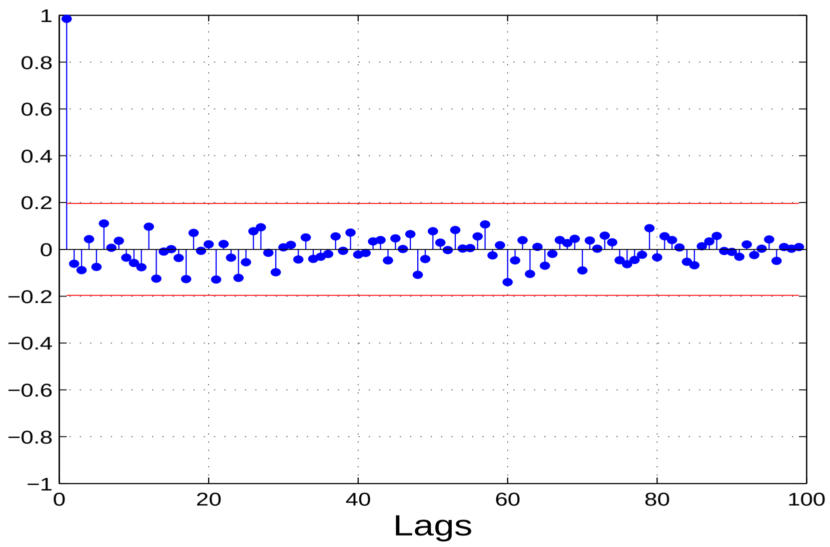
The following points are to be noted:

- **0.7** is an arbitrary value that should be experimented with.
- The order determined by this method is quite high. Increased model order means a significant increase in computational complexity and less accurate LPC estimation from noisy speech. Hence, filter performance may be affected adversely.
- Each frame of speech has a unique order. During frame-wise Kalman filtering, the *a posteriori* error covariance matrix, $\boldsymbol{P(k|k)}$, is carried forward from the previous frame to the next frame. If order of the current and last frames are different, then changes in dimensions of the *a posteriori* error covariance matrix need to be accounted for, either by truncating or zero-padding.
- In case of speech corrupted with AGWN (Additive Gaussian White Noise), higher model order led to a significant improvement in the audible quality of the enhanced speech. However, the same cannot be concluded for other types of band-limited noise (train or babble). Increasing the model order for these types of noise did not enhance the speech output. As a flip-side, increased time complexity of the algorithm made the program run very slowly. These results are discussed in the next chapter.

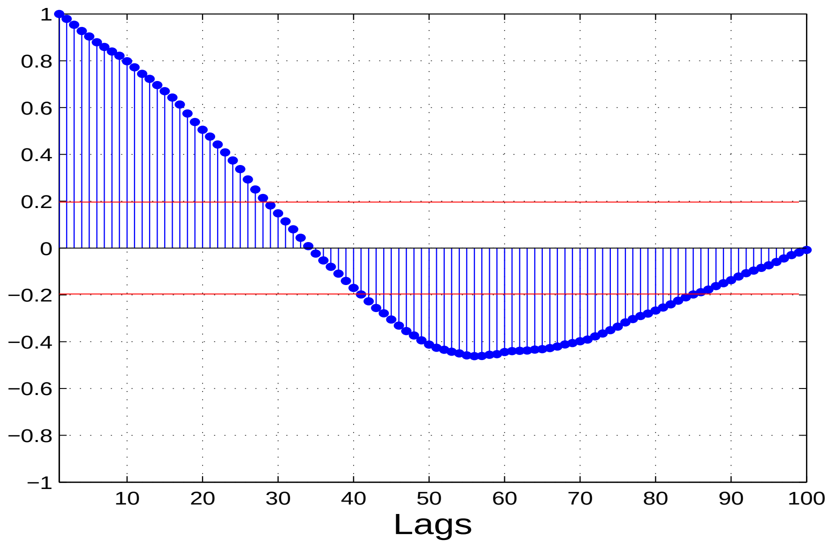
In this chapter, we have discussed the possible methods of model order determination of an AR process, and proposed a new methodology for the same, which utilises the Cumulative Absolute Partial Autocorrelation Function (CPACF). Some shortcomings of increasing model order have also been deliberated. In the next chapter, we will study the results of all the algorithms discussed so far, including filter tuning and optimum order determination, as applied to a noise corrupted speech signal available in the NOIZEUS [20] speech corpus.



(a) Random Walk of length 100

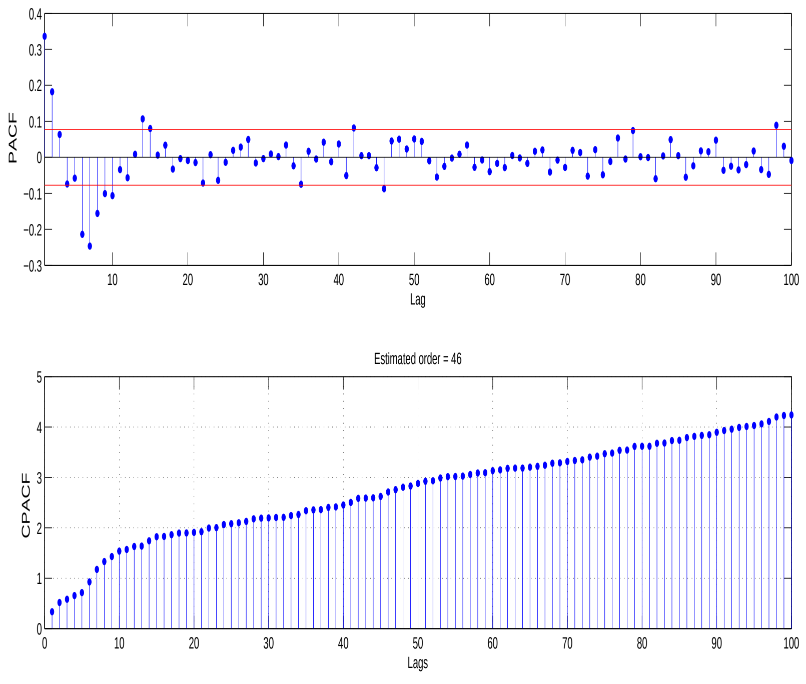


(b) PACF of Random Walk

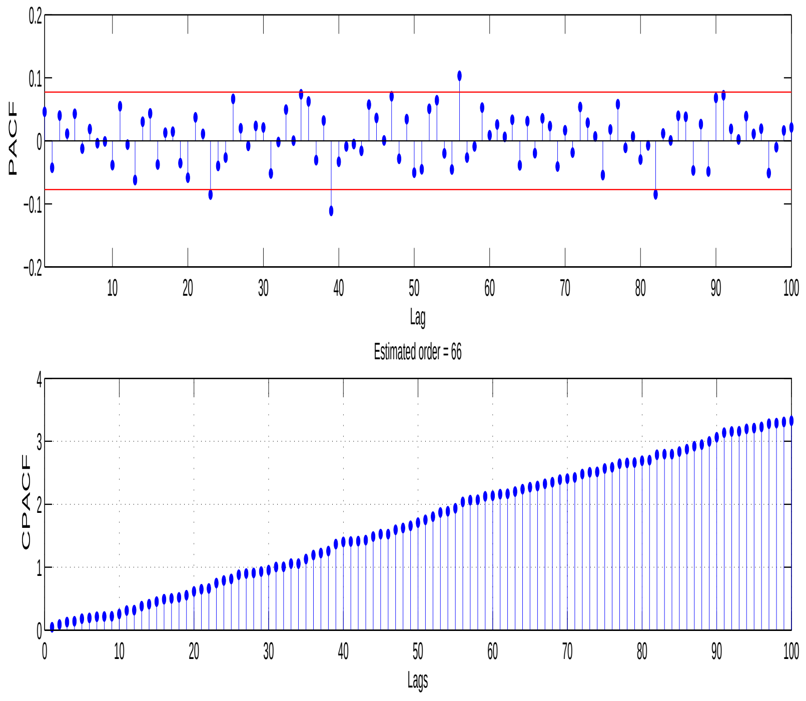


(c) ACF of Random Walk

Figure 3.1: Random Walk Signal : PACF and ACF

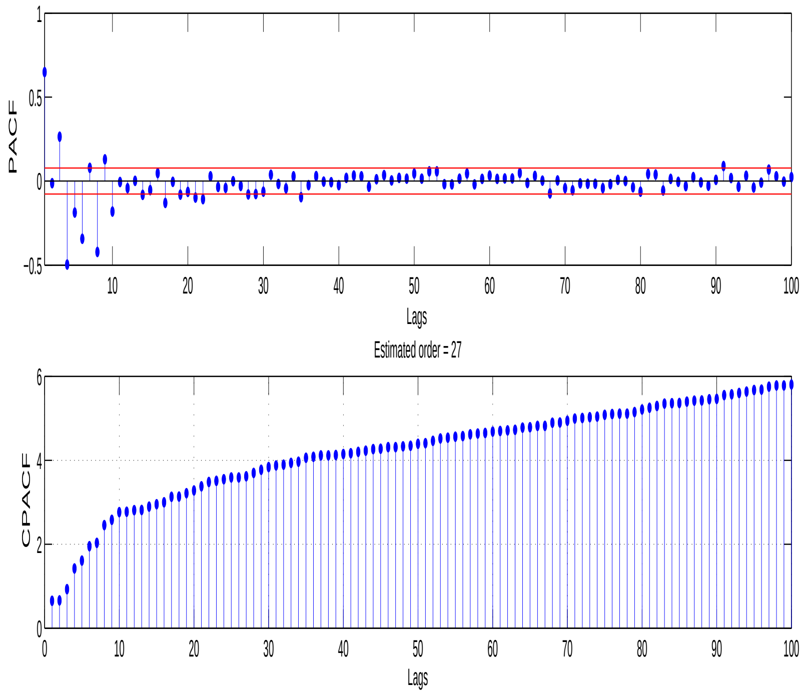


(a) Voiced frame of speech corrupted with white noise of 5dB SNR

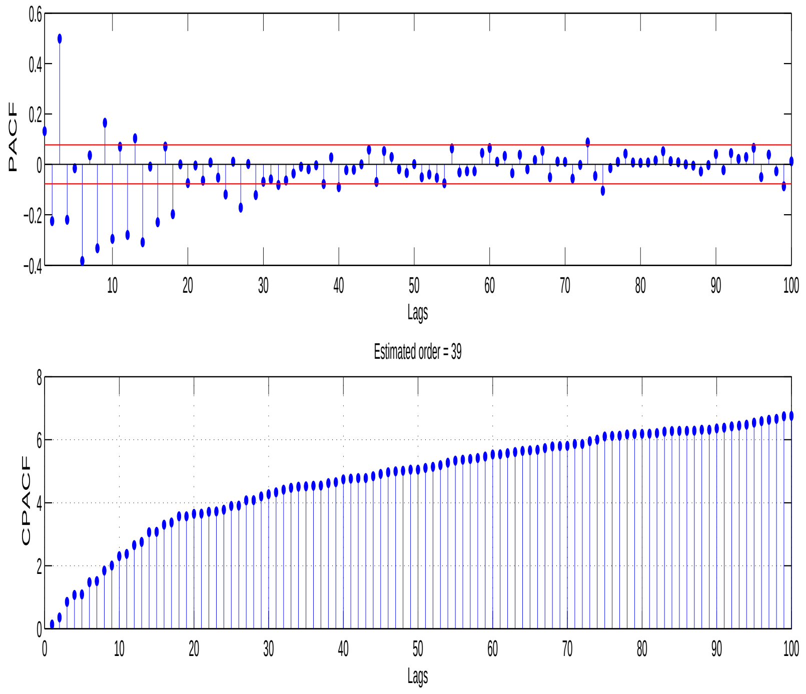


(b) Silent frame of speech corrupted with white noise of 5dB SNR

Figure 3.2: PACF and CPACF of speech corrupted with white noise

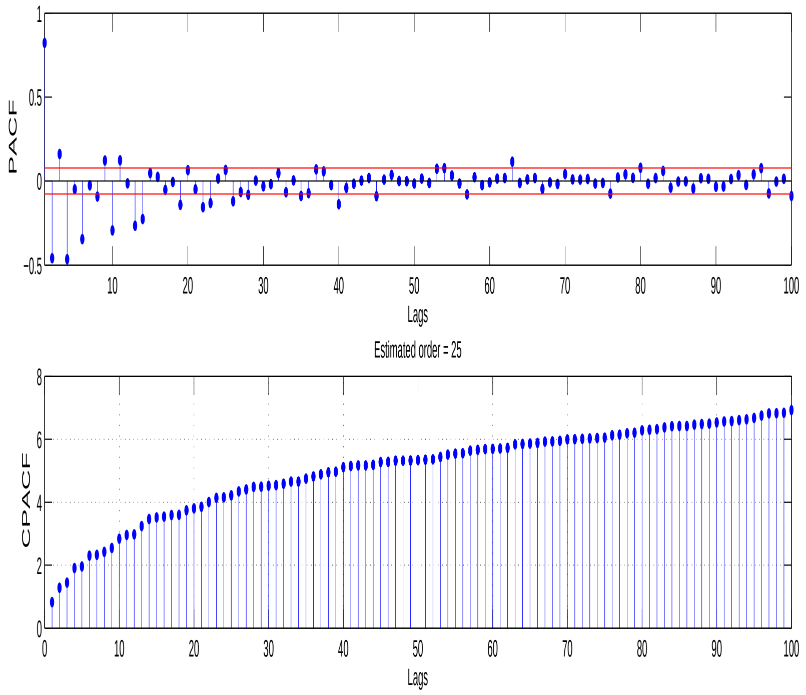


(a) Voiced frame of speech corrupted with train noise of 5dB SNR

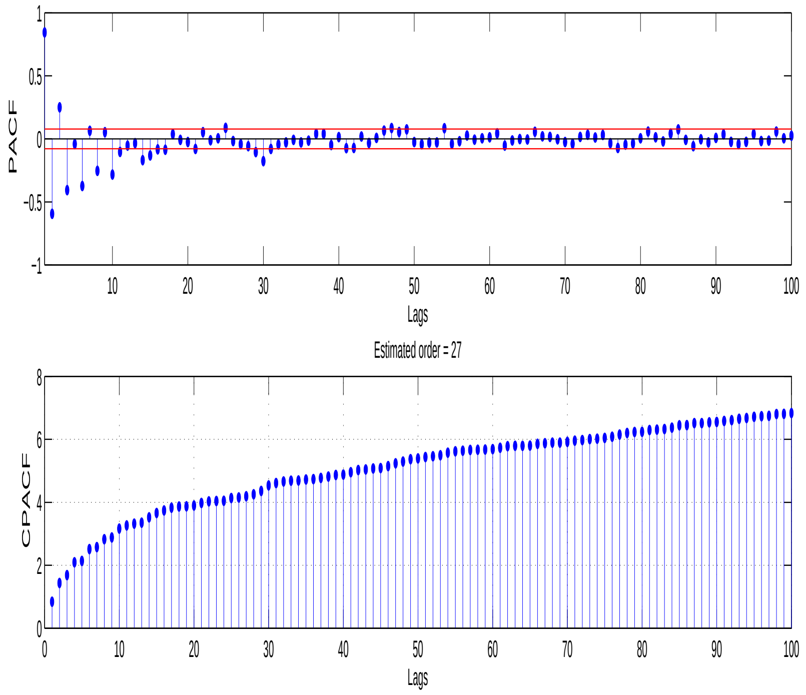


(b) Silent frame of speech corrupted with train noise of 5dB SNR

Figure 3.3: PACF and CPACF of speech corrupted with train noise



(a) Voiced frame of speech corrupted with babble noise of 5dB SNR



(b) Silent frame of speech corrupted with babble noise of 5dB SNR

Figure 3.4: PACF and CPACF of speech corrupted with babble noise

Chapter 4

Experimental Results

In this chapter, we will discuss the results of the Kalman filter algorithm described in Section 1.3.2, along with filter tuning and automatic order estimation, when applied to enhance a noise corrupted speech from the NOIZEUS [20] database [1]. Before looking at the results, it is important to review the methodology that has been applied to clean the noise corrupted speech sample - a female speaker uttering the sentence - *"The clothes dried on a thin wooden rack"*.

4.1 Overview of Tuned Kalman Filter Algorithm with Order Estimation

i) The noisy speech signal is divided into 80ms frames with 10ms overlap.

ii) The frames are classified as silent/voiced according to the method proposed in Section 2.1.1. Measurement noise variance R is calculated as the maximum of variances of all silent frames.

iii) Model order is either fixed at $p = 15$ or calculated according to Section 3.2.4.

iv) For each frame, the pth order LPC coefficients are calculated from noisy speech. The state transition matrix $\boldsymbol{\phi}$ is determined from these coefficients. The prediction error covariance from LPC estimation is taken to be the nominal process noise covariance Q_{f-nom}.

v) Process noise variance Q_f is varied as $10^n Q_{f-nom}$ as mentioned before. The last *a posteriori* error covariance matrix of the previous frame is taken as $\boldsymbol{P}$(k-1|k-1) for the calculation of A_k. J_1 and J_2 are calculated according to 2.9. Ideally, for most balanced performance, $Q_f = Q_c$ should be selected at the point of intersection of J_1 and J_2 curves. However, in this case, a range of values around Q_c are selected by moving along the J_2 curve, according to the equation:

$$\begin{aligned} J_{2i} &= J_{2c} + \frac{1}{4}(i+1) \times (J_{2max} - J_{2c}) \; for \; 0 \leqslant i < 3 \\ &= J_{2min} + \frac{1}{4}(i-3) \times (J_{2c} - J_{2min}) \; for \; 3 \leqslant i \leqslant 6 \end{aligned} \tag{4.1}$$

[1]`http://ecs.utdallas.edu/loizou/speech/noizeus/`

where J_{2c} is the value of J_2 at its point of intersection with J_1. Q_i corresponding to J_{2i} is selected for $0 \leqslant i \leqslant 6$. There is no toggling between two values of Q for voiced and silent frames, and hence no gain adjustment is done either.

vi) Kalman filter equations 1.14 to 1.18 are executed for each frame. If the order of the last frame and the current frame are different, the dimensions of $\boldsymbol{P(k|k)}$ are adjusted.

vii) Iterative Kalman filtering is done, without any filter tuning and with LPCs calculated from *a posteriori* state estimates, $\boldsymbol{X(k|k)}$.

viii) Overlap adding of *a posteriori* state estimates obtained after iterative filtering to yield the final enhanced speech output.

4.2 Quantitative results

To quantitatively measure the quality of the enhanced speech, and to compare it to the original clean speech, we need some evaluation metrics. Common objective measures described in [21] are SNR, Segmental SNR and Frequency Weighted Segmental SNR. Out of these, according to [22], segmental SNR is more consistent with subjective preference scoring than several other methods. Hence, we rely on the difference between the segmental SNR of noisy and enhanced speech to evaluate the performance of our algorithm. The segmental SNR is given by:

$$SegSNR = \frac{1}{N}\sum_{i=1}^{N} 10\ log_{10}\left[\frac{\sum_{n\in frame_k}|s(n)|^2}{\sum_{n\in frame_k}|\hat{s}(n)-s(n)|^2}\right] \tag{4.2}$$

where $s(n)$ is the noise-free signal and $\hat{s}(n)$ is the enhanced speech signal. N is the number of frames and $n \in frame_k$ denotes the samples n in the kth frame. Segmental SNR is expressed in decibels (dB) and a higher value of segmental SNR usually indicates more noise removal from enhanced speech.

Another more commonly used subjective evaluator of speech is the PESQ (Perceptual Evaluation of Speech Quality) test which is discussed by Hu and Loizou in [23][2]. It is a family of standards comprising a test methodology for automated assessment of the speech quality as experienced by a user of a telephony system. It is standardised as ITU-T recommendation P.862 (02/01). A high value of PESQ indicates superior performance of the speech enhancement algorithm. The block diagram of PESQ evaluation is given in figure 4.1.

Segmental SNR gives an indication of the amount of noise reduction, whereas PESQ gives an idea about the perceptual quality of enhanced speech. A very high segmental SNR can be rarely misleading when caused by a significant removal of spectral components of speech along with noise. In that case, the enhanced speech will have a low PESQ indicating that the high segmental SNR was due to loss of intelligibility. Hence, both parameters compliment each other, and are used together to evaluate speech enhancement algorithms.

Segmental SNR and PESQ tests were carried out on a sample of speech corrupted with three different types of noise (white, train and babble), cleaned according to the

[2]The MATLAB code can be downloaded from `http://ecs.utdallas.edu/loizou/speech/software.htm`

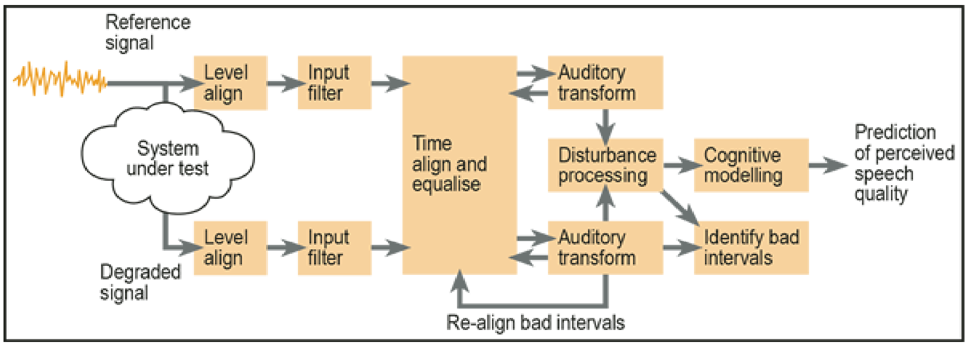


Figure 4.1: *PESQ Block Diagram*

algorithm described in Section 4.1, tested with multiple values of Q around Q_c for both fixed and estimated order. The Segmental SNR plots are given in figure 4.2 and the PESQ plots are given in figure 4.3.

It is seen that Segmental SNR is greater for lower order systems than for higher order systems, indicating that the fixed order of **15** performs better as far as noise removal is concerned. However, the PESQ of higher order systems is more, which implies that significant improvement in the intelligibility of enhanced speech is achieved by increasing model order. These results are discussed further in the next section. The following table summarises the quantitative results:

Table 4.1: *Segmental SNR and PESQ Performance for Different Types of Noise*

Noise Type	*Order*	*SNR (dB)*	*Best* Q_i	*Seg SNR Noisy(dB)*	*Seg SNR Processed(dB)*	*PESQ*
White	15	0	Q_4=0.00012732	-8.819208	1.655220	1.807339
White	53	0	Q_4=0.00016374	-8.819208	0.740564	1.896065
White	15	5	Q_4=9.8624e-005	-6.319208	3.002766	1.692509
White	50	5	Q_4=9.7156e-005	-6.319208	1.828233	2.095206
White	15	10	Q_5=0.00012086	3.819208	4.349849	2.018423
White	48	10	Q_5=0.00010170	-3.819208	3.488442	2.223566
Train	15	0	Q_1=0.00017351	-7.905742	-0.443979	1.897596
Train	31	0	Q_1=0.00029121	-7.905742	-0.856497	1.961122
Train	15	5	Q_1=0.00010111	-3.557488	1.198366	2.026621
Train	32	5	Q_3=0.00045214	-3.557488	1.114640	2.138670
Train	15	10	Q_6=0.0190400	1.481678	2.635199	2.379670
Train	29	10	Q_6=0.0180990	1.481678	2.455842	2.553597
Babble	15	0	Q_1=0.00061337	-8.295660	-2.491006	1.734588
Babble	30	0	Q_1=0.00060997	-8.295660	-3.416224	1.808091
Babble	15	5	Q_3=0.0014059	-3.685627	-0.583802	2.088812
Babble	31	5	Q_3=0.0014178	-3.685627	-0.884745	2.215525
Babble	15	10	Q_2=0.001072	1.270681	1.625915	2.411438
Babble	29	10	Q_5=0.0076442	1.270681	1.564816	2.594426

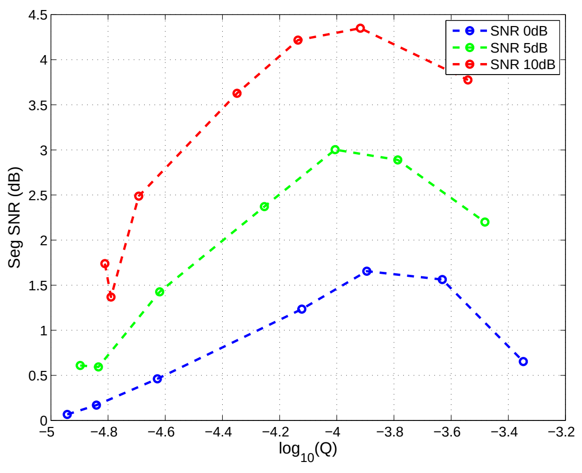
It is observed that for white noise $Q > Q_c$ gives better results. For train and babble noise, the value of Q that gives best performance depends on the SNR of noise corrupted speech. For low SNR speech (high ratio of noise), $Q < Q_c$ gives better performance. For intermediate SNR, $Q = Q_c$ gives best performance and for low SNR (low ratio of noise), $Q > Q_c$ results in best performance. This is because, for low SNR speech (very noisy), the measurement is to be trusted less and the *a priori* state estimate should be trusted more. In other words, a more sensitive performance is required, which is satisfied by a lower value of Q. For high SNR speech (least noisy), the measurement is to be trusted more, and hence robustness is given priority. As a result, a higher value of Q gives superior results. For intermediate level of noise, a compromise between sensitivity and robustness gives best performance, which is given by $Q = Q_c$.

4.3 Qualitative results

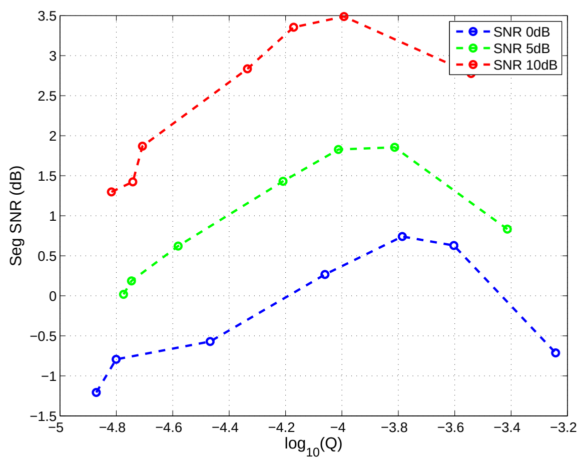
While quantitative results are useful in evaluating speech enhancement algorithms, the ultimate judge is the listening test. However, listening test results are highly subjective and may vary from listener to listener. In our case, the listening tests comply with the quantitative results. A few decibels of difference in segmental SNRs are hard to distinguish by ear. What is observable though, is the improvement in the subjective quality of speech on increasing model order, especially in case of speech corrupted with white noise where intelligibility improves significantly. However, it comes with the introduction of a background hum.

Another method of evaluating qualitative results are by studying the spectrograms of the original, noisy and enhanced speech. The spectrogram is a 3D plot which represents the Short Time Fourier Transform (STFT) of a non-stationary signal, with time and frequency on the x and y axes and amplitude in dBs represented by depth of colour. The original spectrogram of uncorrupted speech, spectrograms of speech corrupted with different types of noise of SNR 5dB along with their enhanced versions are given in figures 4.4, 4.5 and 4.6.

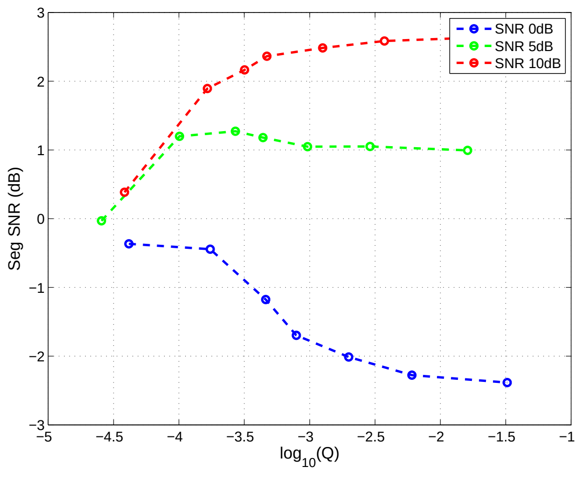
It is evident from the spectrograms that a lower order model performs better noise removal than a higher order model. However, because the higher order models preserve more of the spectral components in the enhanced output, they improve intelligibility.



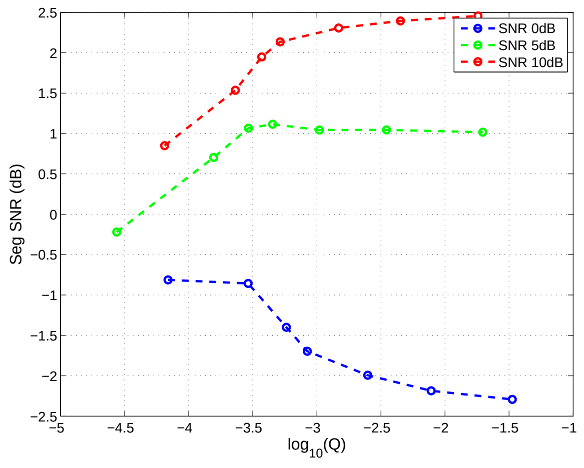
(a) White noise - Fixed order



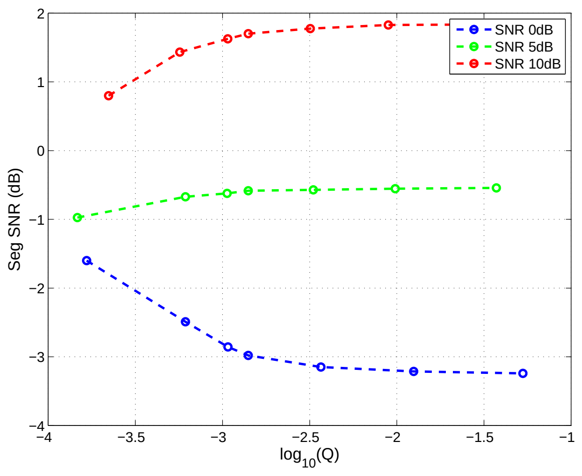
(b) White noise - Estimated order



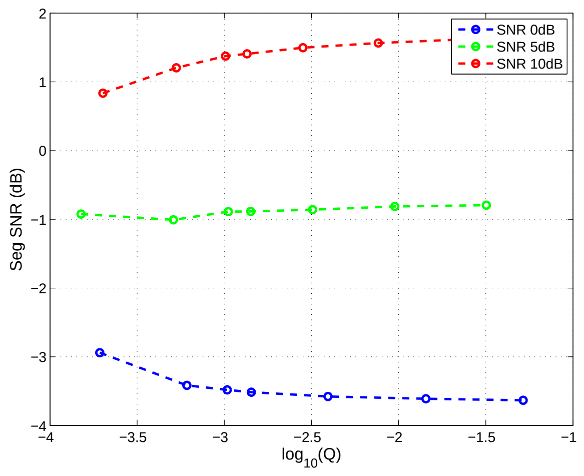
(c) Train - Fixed order



(d) Train - Estimated order

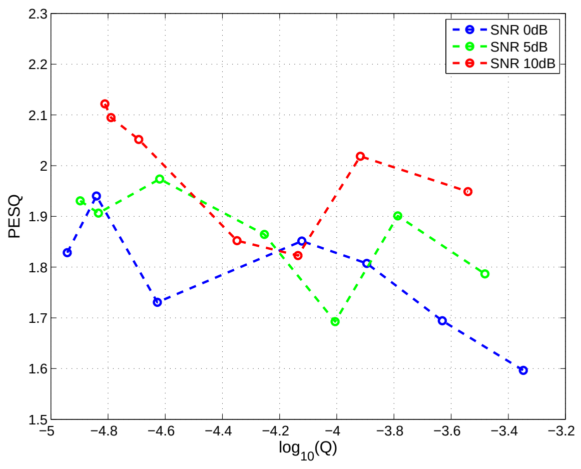


(e) Babble - Fixed order

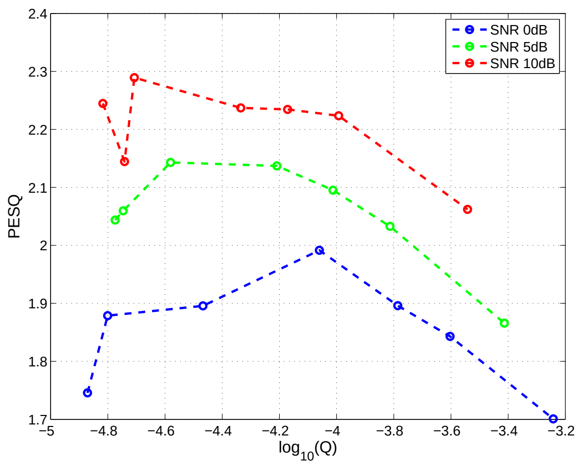


(f) Babble - Estimated order

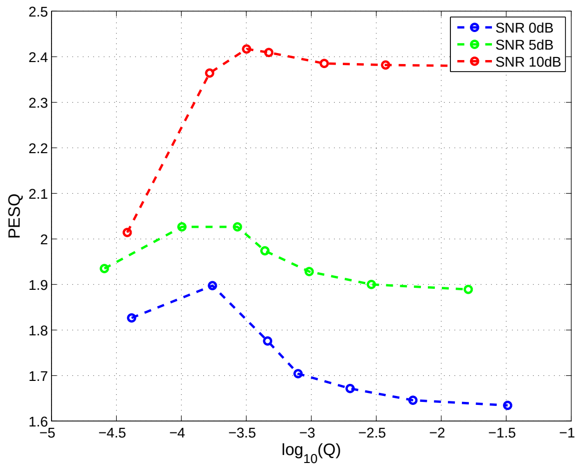
Figure 4.2: Segmental SNR (dB) v/s $\log_{10}$ Q



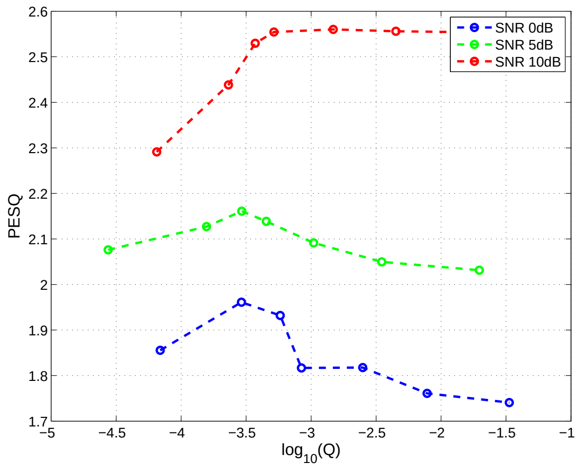
(a) White noise - Fixed order



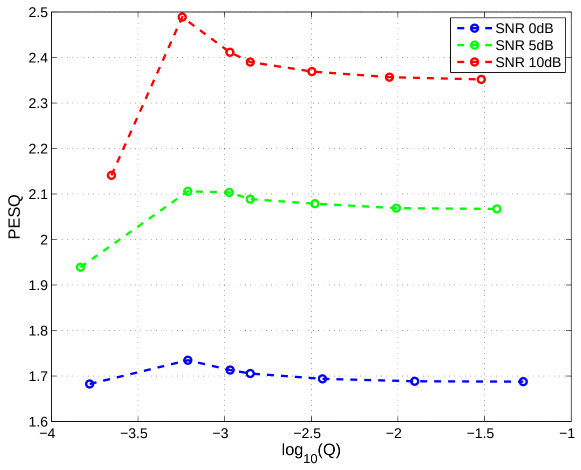
(b) White noise - Estimated order



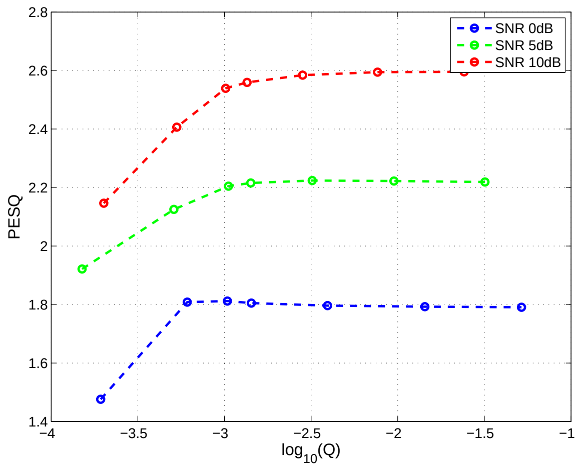
(c) Train - Fixed order



(d) Train - Estimated order

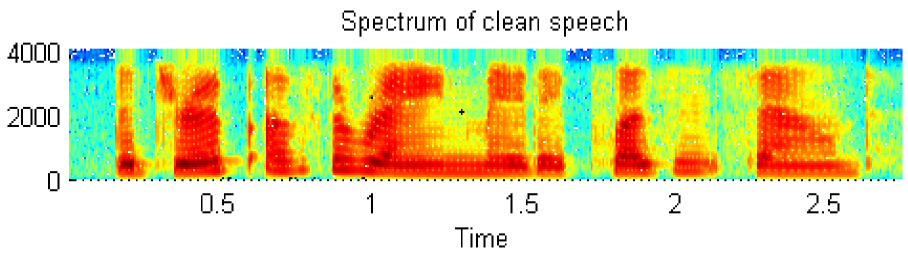


(e) Babble - Fixed order

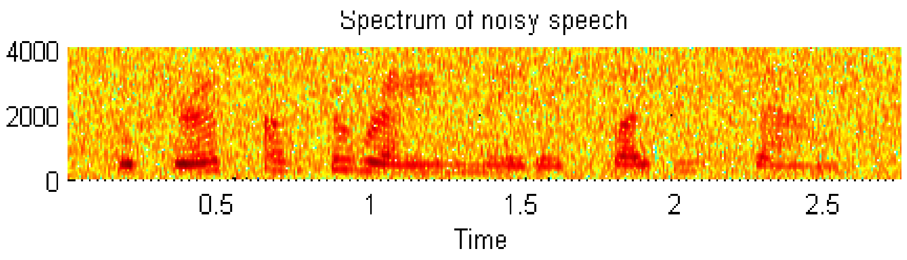


(f) Babble - Estimated order

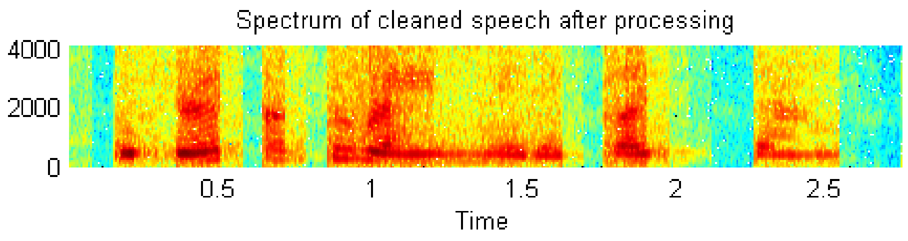
Figure 4.3: PESQ v/s $\log_{10}$ Q



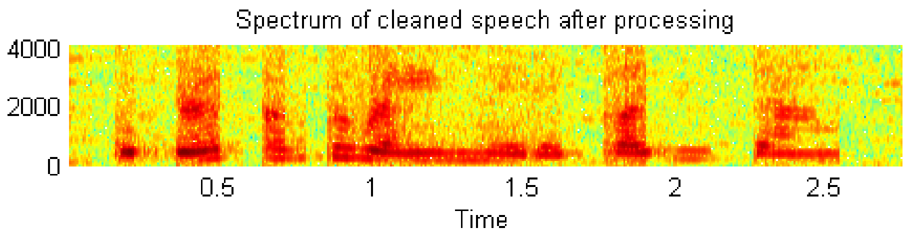
(a) Clean Speech



(b) Corrupted with White Noise of 5dB SNR

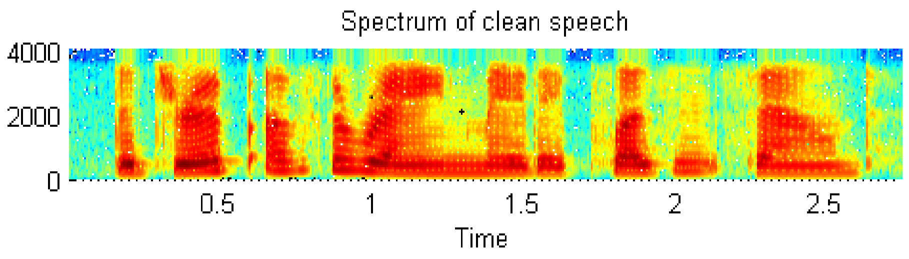


(c) Enhanced Speech - Fixed Order = 15

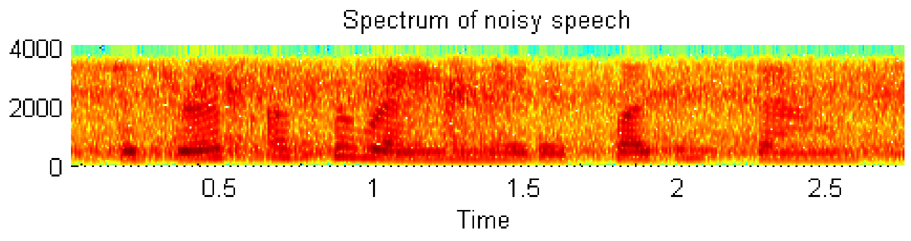


(d) Enhanced Speech - Estimated Order = 50

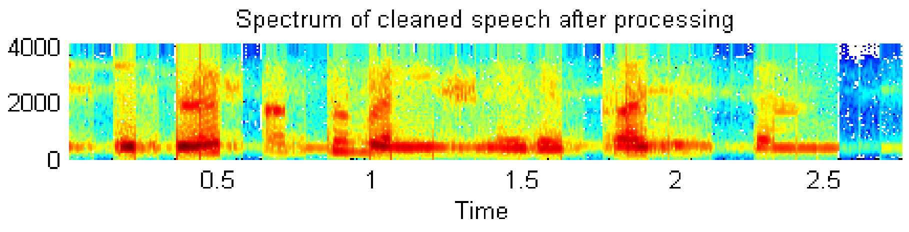
Figure 4.4: Spectrograms of speech corrupted with white noise and enhanced speech



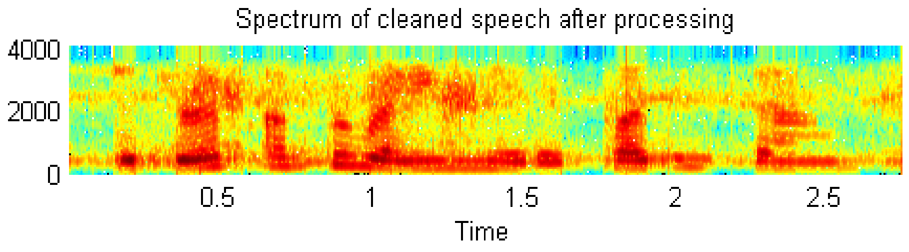
(a) Clean Speech



(b) Corrupted with Train Noise of 5dB SNR

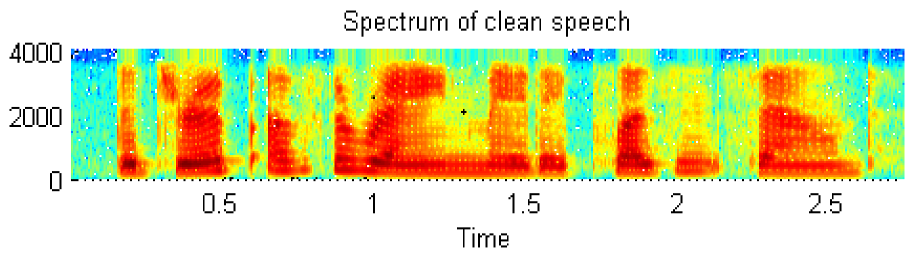


(c) Enhanced Speech - Fixed Order = 15

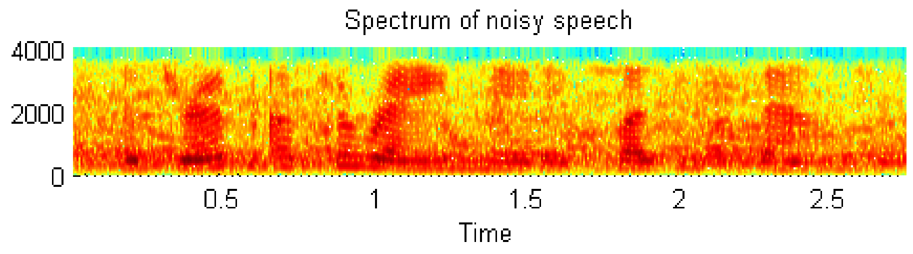


(d) Enhanced Speech - Estimated Order = 32

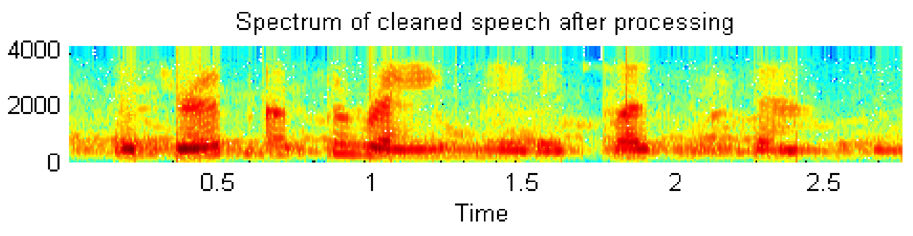
Figure 4.5: Spectrograms of speech corrupted with train noise and enhanced speech



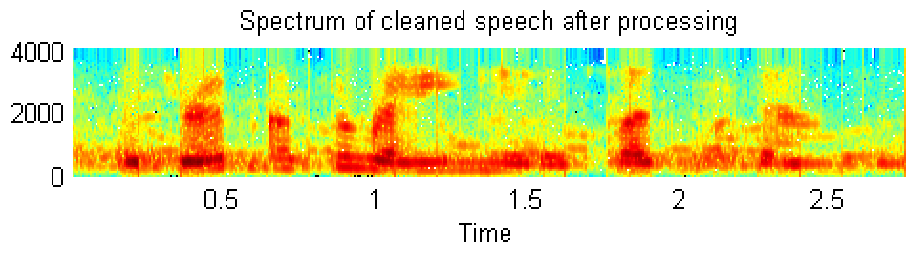
(a) Clean Speech



(b) Corrupted with Babble Noise of 5dB SNR



(c) Enhanced Speech - Fixed Order = 15



(d) Enhanced Speech - Estimated Order = 31

Figure 4.6: Spectrograms of speech corrupted with babble noise and enhanced speech

Chapter 5

Conclusion

This thesis has dealt with application of the Kalman Filter in speech enhancement. Even though the algorithm proposed by Paliwal and Basu in [4] lies at the heart of this work, it has been enhanced and modified in numerous ways. It has culminated in a thesis that revolves around advanced topics in Digital Signal Processing, Speech Processing and Time Series Analysis. In the concluding chapter of this thesis, we discuss in brief, all the chapters and propose extensions and scope for future study.

In Chapter 1, we did a literature survey, introduced the Kalman Filter and the Autoregressive Model of speech. We also studied the autocorrelation function and discussed Linear Prediction Coefficient estimation by the autocorrelation method.

In Chapter 2, we devised methods for filter tuning. We discussed the Power Spectral Density function in detail and derived an algorithm for determination of measurement noise variance, R, based on the spectral flatness of the PSD function. In section 2.2, we discussed the algorithm in [12] to determine an optimum value of process noise covariance, Q, by making use of the robustness and sensitivity metrics.

In Chapter 3, the motivation behind studying AR model order was discussed. We studied the Partial Autocorrelation Function (PACF) proposed by Box and Jenkins in [13] to determine the order of an AR process. From PACF, we derived the Cumulative Absolute Partial Autocorrelation Function (CPACF), which was utilised in determining optimum model order for each frame of noise corrupted speech. We also looked at PACF and CPACF plots of speech corrupted by different types of noise.

In Chapter 4, we first gave an overview of the speech enhancement algorithm. Following that, we discussed the qualitative and quantitative results of applying our algorithm to clean a corrupted speech from the NOIZEUS corpus. We looked at the Segmental SNR and PESQ plots for different values of Q and different types of noises of different SNRs. Finally we studied the spectrograms of the original, corrupted and enhanced signals and discussed the implication of our results.

5.1 Future Work

The tuned Kalman filter proposed in [8] was used to clean a noise corrupted archival piece of vocal singing clip (sung by Rabindranath Tagore) with the aim of applying the algorithm for music enhancement. However, the algorithm failed to perform as desired. The reasons for that were discussed in [24]. It was observed that the value of sensitivity metrics, J_1, was very low, whereas that of the robustness metrics, J_2, was high. (robustness prone system). As a result, the estimated value of process noise variance, Q, was quite high leading to a very high value of Kalman gain. That means the output borrowed heavily from the noisy input and very little noise enhancement was achieved.

Since the algorithm in [8] has been modified considerably in this thesis, it is expected to work better on music enhancement. The effect of increasing model order could be the key in case of music. According to So in [6], for a fixed, low order of the AR (Autoregressive) model, the harmonic structure of music is often lost. It was concluded in [24] that a proper selection of the system order needed to be evolved for modelling the complex harmonic structure in signals like music. That has been done in this thesis and the next step is to test the algorithm with automatic order determination on music signals.

Appendix A

MATLAB scripts and functions

All the MATLAB functions that implement the speech enhancement algorithm are included in this appendix.[1]

A.1 Function to implement Kalman Filter based speech enhancement algorithm

```matlab
function [] = KF_speech(filename, noiseType, orderType)
%Applies tuned Kalman filter with order estimation on noisy speech.
%filename - name of .wav speech file from NOIZEUS
%noiseType - white, train or babble
%orderType = estimated or fixed

parentpath = fileparts(pwd);
SNR = [0,5,10];

%this folder contains MATLAB files needed to calculate PESQ
%download it from http://ecs.utdallas.edu/loizou/speech/software.htm
%and extract it in the parent directory
addpath(strcat(parentpath,'\composite\'));

%this folder contains clean and corrupted .wav speech files downloaded from
%NOIZEUS database - http://ecs.utdallas.edu/loizou/speech/noizeus/
soundpath = strcat(parentpath,'\Noisy speech samples\');

%folder where results are saved - create if does not exist
saveToPath = ['Results\Rnew all noise ',orderType, ' order\',noiseType,...
    '\',filename,'\'];
if exist(saveToPath, 'dir') == 0
    mkdir(saveToPath);
end

%writing results to txt file
[fileID] = fopen([saveToPath,filename,'_',noiseType,'_results.txt'],'w+');
fprintf(fileID,'%s %s %s %s %s %s %s %s\r\n','R(new)','Q_chosen',...
    'log(Q)','SNR','SegSNR_before','SegSNR_after','PESQ','Average_order');

for snri = 1:length(SNR)
```

[1]Programs can be downloaded as .zip file or cloned as repository from `https://github.com/orchidas/KF-speech-thesis`

```

    %read a clean audio signal
    [z,fs] = wavread(strcat(soundpath,'clean\',filename,'.wav'));
    %read a corrupted audio signal
    if(strcmp(noiseType, 'white') == 1)
        wn = wavread([soundpath,'white_noise.wav']);
        [noise,snr] = makeSNR(z, wn, SNR(snri));
        y = noise + z;
    else
        [y,fs] = wavread(strcat(soundpath, noiseType,'\',...
            num2str(SNR(snri)),'dB\',filename,'_',type,'_sn',....
            num2str(SNR(snri)),'.wav'));
    end
    y=y';
    z=z';

    %dividing into 80ms frames with 10ms overlap
    start=1;
    l=0.08*fs;
    overlap=0.01*fs;
    totseg=ceil(length(y)/(l-overlap));
    segment=zeros(totseg,l);
    zseg=zeros(totseg,l);
    for i=1:totseg-1
        segment(i,1:l)=y(1,start:start+l-1);
        zseg(i,1:l)=z(1,start:start+l-1);
        start=(l-overlap)*i+1;
    end
    segment(totseg,1:length(y)-start+1)=y(start:length(y));
    zseg(totseg,1:length(z)-start+1)=z(start:length(z));
    cleanspeech=zeros(totseg,l);
    cleanSpeech=zeros(1,length(y));

    %determine order
    if strcmp(orderType,'fixed') == 1
        order = ones(totseg,1).*15;
    else
        order = findOrder(segment,SNR(snri),type,saveToPath);
    end

    %calculate measurement noise variance R
    R = measurementNoiseNew(segment, fs);

    J1=zeros(1,10);
    J2=zeros(1,10);
    nq=zeros(1,10);
    u=1;

    for m = 0:6

        segsnr_before=0;
        segsnr_after=0;
        Q_arr=zeros(1,totseg);

        for i=1:totseg
            %initializing
            X=y(1:order(i))';
            P=zeros(l,order(i),order(i));
            t1=zeros(order(i),order(i));
            H=[zeros(1,order(i)-1),1];
            G=H';

            %first iteration of Kalman filter
            [A,Q1]=lpc(segment(i,:),order(i));
            temp=eye(order(i));
            PHI=[temp(2:order(i),:);-fliplr(A(2:end))];

            if(i == 1)
                P(1,:,:)=R*eye(order(i));
            else
                P(1,:,:) = Y(:,:);
            end

            %calculating optimum value of process noise variance, Q
            q=1;
            for n=-5:4
                Q0=(10^n)*Q1;
                t1(:,:)=P(1,:,:);
                Ak=H*(PHI*t1*PHI')*H';
                Bk=H*Q0*H';
                J1(q)=R/(Ak+Bk+R);
                J2(q)=Bk/(Ak+Bk);
                nq(q)=log10(Bk);
                q=q+1;
            end

            %interpolate nq, J1 and J2 to increase resolution, and
            %to get more accurate approximation of Q
            nqi = -5:0.25:4;
            J2i = interp1(nq,J2,nqi);
            J1i = interp1(nq,J1,nqi);
            [nq_nom,Jc]=intersections(nqi,J1i,nqi,J2i);

            if m < 3
                J2_desired = (0.25*(m+1)*(Jc-min(J2i))) + min(J2i);
            else
                J2_desired = (0.25*(m-3)*(max(J2i)-Jc))+ Jc;
            end

            [difference, index] = min(abs(J2i - J2_desired));
            Q = 10^(nqi(index));

            %plot J1,J2 for voiced frame
            if(i == 4)
                figure(1);
                plot(nqi,J1i,'-b+');hold on;grid on;
                plot(nqi,J2i,'-b*');hold on;grid on;
                scatter(nqi(index),J2i(index),'k');
            end

            %plot J1,J2 for silent frame
            if(i == totseg - 1)
                figure(1);
                plot(nqi,J1i,'-r+');hold on;grid on;
                plot(nqi,J2i,'-r*');hold on;grid on;
                scatter(nqi(index),J2i(index),'k');
                xlabel('nq','FontSize',18);
                ylabel('JI,J2','FontSize',16);
                axis([min(nqi)-2, max(nqi), 0, 1]);
                hold off;
                legend('J_1','J_2');
            end

            Q_arr(u)=Q;
            u=u+1;

            for j=1:length(segment(i,:))
                X_=PHI*X;
                t1(:,:)=P(j,:,:);
                P_=(PHI*t1*PHI')+(G*Q*G');
                K=(P_*H')*(inv(H*P_*H'+R));
                t1=(eye(order(i))-K*H)*P_;
                P(j+1,:,:)=t1(:,:);
                e=segment(i,j)-(H*X_);
                X=X_+K*e;
                cleanspeech(i,j)=X(end);

            end
            %adjust a posteriori error covariance matrix dimensions
            if(i< totseg)
                t2 = zeros(order(i),order(i));
                t2(:,:) = P(j-1,:,:);
                Y = adjustDimensions(t2, order(i+1));
            end

            %second iteration of Kalman filter with lpc calculated from
            %cleaned speech
            [A,Q]=lpc(cleanspeech(i,:),order(i));
            PHI=[temp(2:order(i),:);-fliplr(A(2:end))];
            X=cleanspeech(i,1:order(i))';

            if i==1
                P0=R*eye(order(i));
            else
                P0 = Z(:,:);
            end

            for j=1:length(segment(i,:))
                X_=PHI*X;
                P_=(PHI*P0*PHI')+(G*Q*G');
                K=(P_*H')*(inv(H*P_*H'+R));
                P0=(eye(order(i))-K*H)*P_;
                e=segment(i,j)-(H*X_);
                X=X_+K*e;
                cleanspeech(i,j)=X(end);
            end

            if(i< totseg)
                Z = adjustDimensions(P0, order(i+1));
            end

            %calculate Segmental SNR
            segsnr_before=segsnr_before+log10(rms(zseg(i,:))/...
                rms(zseg(i,:)-segment(i,:)));
            segsnr_after=segsnr_after+log10(rms(zseg(i,:))/...
                rms(zseg(i,:)-cleanspeech(i,:)));
        end

        %overlap add
        cleanSpeech(1:l)=cleanspeech(1,1:l);
        start=l+1;
        for i=2:totseg-1
            cleanSpeech(start:start+(l-overlap))=...
                cleanspeech(i,overlap:end);
            start=start+l-overlap-1;
        end
        cleanSpeech(start:length(y))=cleanspeech(totseg,...
            1:(length(y)-start)+1);
        cleanSpeech=cleanSpeech(1:length(y));

        %normalizing
        z=z./abs(1.2*max(z));
        y=y./abs(1.2*max(y));
        cleanSpeech=cleanSpeech./abs(1.2*max(cleanSpeech));

        %qualitative measure of noise removed
        figure(2);ylabel('Normalised amplitude');xlabel('Time in seconds');
        subplot(3,1,1);plot((1:length(z))/fs,z);title('original speech');
        axis([0, length(z)/fs, -1, 1]);
        subplot(3,1,2);plot((1:length(y))/fs,y,'k');
        title('corrupted speech');axis([0, length(y)/fs, -1, 1]);
        subplot(3,1,3);plot((1:length(cleanSpeech))/fs,cleanSpeech,'r');
        title('cleaned speech');axis([0, length(cleanSpeech)/fs, -1, 1]);
        wavplay(y,fs);
        wavplay(cleanSpeech,fs);
        wavwrite(cleanSpeech,fs,strcat(saveToPath,filename,'_Q',...
            num2str(m),'_',noiseType,'_sn',num2str(SNR(snri)),...
            '_enhanced.wav'));
        saveas(figure(2),[saveToPath,'Waveform_',filename,'_Q',...
            num2str(m),'_',noiseType,'_sn',num2str(SNR(snri))]);

        figure(3);
        subplot(3,1,1);spectrogram(z,64,16,1024,fs,'yaxis');
        title('Spectrum of clean speech');
        subplot(3,1,2);spectrogram(y,64,16,1024,fs,'yaxis');
        title('Spectrum of noisy speech');
        subplot(3,1,3);spectrogram(cleanSpeech,64,16,1024,fs,'yaxis');
        title('Spectrum of cleaned speech after processing');
        saveas(figure(3), [saveToPath,'Spectrogram_',filename,'_Q',...
            num2str(m),'_',noiseType,'_sn',num2str(SNR(snri))]);

        %quantitative measure of noise removed - Seg SNR
        disp('The segmental snr before processing is :');
        segsnr_before = 20*segsnr_before/totseg
        disp('The segmental snr after processing is :');
        segsnr_after = 20*segsnr_after/totseg

        %PESQ
        psq = pesq(fs,strcat(soundpath,'clean\',filename,'.wav'),...
            strcat(saveToPath, filename,'_Q',num2str(m),'_',noiseType,...
            '_sn',num2str(SNR(snri)),'_enhanced.wav'));
        fprintf(fileID,'%f %s %s %d %f %f %f %d\r\n', R,...
            ['Q',num2str(m),'=',num2str(mean(Q_arr))],...
            ['n=',num2str(log10(mean(Q_arr)))], SNR(snri),...
            segsnr_before, segsnr_after, psq, round(mean(order)));
        close all;
    end
end
fclose('all');
end
```

Listing A.1: KF_speech.m

A.2 Function to determine R

```
function [R] = measurementNoiseNew(xseg,fs)
%new method of calculating measurement noise variance based on PSD

numFrame = size(xseg,1);
noise_cov = zeros(1,numFrame);
spectral_flatness = zeros(1,numFrame);
%order estimation for voiced and silent frames
for k = 1:numFrame

    [c, lag] = xcorr(xseg(k,:),'coeff');
    %calculating power spectral density from ACF
    psd = (fftshift(abs(fft(c))));
    psd = psd(round(length(psd)/2):end);
    freq = (fs * (0:length(c)/2))/length(c);
    %keeping positive lags only since ACF is symmetrical
    c = c(find(lag == 0):length(c));
    lag = lag(find(lag == 0):length(lag));
    %keep frequencies from 100Hz to 2kHz
    freq_2kHz = find(freq>= 100 & freq<=2000);
    psd_2kHz = psd(freq_2kHz);
    spectral_flatness(k) = geomean(psd_2kHz)/mean(psd_2kHz);

end

normalized_flatness = spectral_flatness/max(spectral_flatness);
threshold = 0.707;
for k = 1:numFrame
    if normalized_flatness(k) >= threshold
        noise_cov(k) = var(xseg(k,:));
    end
end
R = max(noise_cov)
end
```

Listing A.2: measurementNoiseNew.m

A.3 Function to estimate order

```matlab
function [ order ] = findOrder(noisy, dB, type, saveToPath)
%estimates order of each frame of noisy signal
totseg = size(noisy,1);
order = zeros(totseg,1);
%we assume maximum order to be 100
T = 100;

for i = 1:totseg
    [arcoefs,noisevar,reflection_coefs] = aryule(noisy(i,:),T);
    pacf = -reflection_coefs;
    cpacf = cumsum(abs(pacf));
    %estimated order = lag at which CPACF is 70% of range of CPACF
    dist = abs(cpacf - 0.7*(range(cpacf)));
    order(i) = find(dist == min(dist),1,'first');

    if i == 4 || i == totseg - 1
        if i == 4
            figure(5);
            heading = 'PACF plot for Voiced Frame';
        else
            figure(6);
            heading = 'PACF plot for Silent Frame';
        end
        title(heading);
        subplot(211);
        stem(pacf,'filled','MarkerSize',4);
        xlabel('Lag');ylabel('Partial Autocorrelation coefficients');
        xlim([1 T]);
        uconf = 1.96/sqrt(size(noisy,2));
        lconf = -uconf;
        hold on;
        plot([1 T],[1 1]'*[lconf uconf],'r');
        hold off;
        subplot(212);
        text = ['Estimated order = ',num2str(order(i))];
        stem(cpacf,'filled','MarkerSize',4);
        xlabel('Lag');ylabel('Cumulative PACF');title(text);
        grid on;
        hold on;
        plot(0.7*range(cpacf)*ones(1,T),'r');
        hold off;
        xlabel('Lags');ylabel('Cumulative PACF');
    end
end

saveas(figure(5),[saveToPath,'PACF_plot_voiced_frame_',...
    type,'_',num2str(dB),'dB']);
saveas(figure(6),[saveToPath,'PACF_plot_silent_frame_',...
    type,'_',num2str(dB),'dB']);

end
```

Listing A.3: findOrder.m

A.4 Function to adjust matrix dimensions

```
function [Y] = adjustDimensions(X,p)
%Adjust the dimensions of X to pxp
m = size(X,1);
Y = zeros(p,p);
if(p > m)
    newRows = zeros(p-m,m);
    newCols = zeros(p,p-m);
    temp = [X;newRows];
    Y = [temp newCols];
else
    if(p == m)
        Y = X;
    else
        Y(:,:) = X(1:p,1:p);
    end
end
end
```

Listing A.4: adjustDimensions.m

A.5 Function to add noise of desired SNR to signal

```
function [desiredNoise,snr] = makeSNR(x,actualNoise,dB)
%make noise of SNR 'dB' when clean signal and noise are given

%making lengths of noise and signal equal
if(length(actualNoise) > length(x))
    actualNoise = actualNoise(1:length(x));
else
    if(length(actualNoise) < length(x))
        start = length(actualNoise)+1;
        while(length(actualNoise) < length(x))
            actualNoise(start:start+length(actualNoise)-1) = actualNoise(1:end);
            start = length(actualNoise) + 1;
        end
        actualNoise = actualNoise(1:length(x));
    end;
end;

sumOfSquares_desired = (sum(x.^2))*(10^(-dB/20));
sumOfSquares_given = sum(actualNoise.^2);
ratio = sqrt(sumOfSquares_desired/sumOfSquares_given);
desiredNoise = ratio*actualNoise;
%snr should be equal to dB
snr = 20*log10(sum(x.^2)/sum(desiredNoise.^2));
end
```

Listing A.5: makeSNR.m

Appendix B

Bibliography

[1] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[2] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, Chapel Hill, NC, USA, 1995.

[3] B.S Atal. Speech analysis and synthesis by linear prediction of the speech wave. *Journal of the Acoustical Society of America*, 47(1):65, 1970.

[4] K.K. Paliwal and A. Basu. A speech enhancement method based on kalman filtering. In *Proc. ICASSP*, volume 12, 1987.

[5] J.S Lim and A. V. Oppenheim. Enhancement and bandwidth compression of noisy speech. In *Proc. IEEE*, volume 67, 1979.

[6] S. So and K.K. Paliwal. Suppressing the influence of additive noise on the kalman gain for low residual noise speech enhancement. *Speech Communication*, 53:355–378, 2011.

[7] J.D. Gibson, B. Koo, and S.D. Gray. Filtering of colored noise for speech enhancement and coding. *IEEE Trans. Signal Process.*, 39(8):1732–1742, 1991.

[8] Orchisama Das, Bhaswati Goswami, and Ratna Ghosh. Application of the tuned kalman filter in speech enhancement. In *2016 IEEE First International Conference on Control, Measurement and Instrumentation (CMI)*, pages 62–66. IEEE, 2016.

[9] K.K. Paliwal. Estimation of noise variance from the noisy ar signal and its application in speech enhancement. *IEEE Trans. Acoust., Speech, Signal Process.*, 36(2):292–294, 1988.

[10] Gidon Eshel. The yule walker equations for the ar coefficients. 2003.

[11] Rainer Martin. Noise power spectral density estimation based on optimal smoothing and minimum statistics. *Speech and Audio Processing, IEEE Transactions on*, 9(5):504–512, 2001.

[12] Mousumi Saha, Ratna Ghosh, and Bhaswati Goswami. Robustness and sensitivity metrics for tuning the extended kalman filter. *Instrumentation and Measurement, IEEE Transactions on*, 63(4):964–971, 2014.

[13] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control.* John Wiley & Sons, 2015.

[14] Peter J Brockwell and Richard A Davis. *Introduction to time series and forecasting.* Springer Science & Business Media, 2006.

[15] RG Bachu, S Kopparthi, B Adapa, and BD Barkana. Separation of voiced and unvoiced using zero crossing rate and energy of the speech signal.

[16] *Power Spectal Density*, chapter 10. MIT OpenCourseWare, 2010.

[17] Nilesh Madhu. Note on measures for spectral flatness. *Electronics letters*, 45(23):1195–1196, 2009.

[18] Lawrence R. Rabiner and Ronald W. Schafer. *Digital Processing of Speech Signals*, chapter 8 - Linear Predictive Coding of Speech, page 398. Prentice Hall, 1978.

[19] Gautam Vallabha and Betty Tuller. Choice of filter order in lpc analysis of vowels.

[20] Y. Hu and P. Loizou. Subjective evaluation and comparison of speech enhancement algorithms. *Speech Communication*, 49:588–601, 2007.

[21] Bernard Grundlehner, Johan Lecocq, Radu Balan, and Justinian Rosca. Performance assessment method for speech enhancement systems. Citeseer.

[22] B. Schwerin and K. Paliwal. Using stft real and imaginary parts of modulation signals for mmse-based speech enhancement. *Speech Communication*, 58:49–68, 2014.

[23] Y. Hu and P. Loizou. Evaluation of objective quality measures for speech enhancement. *IEEE Transactions on Speech and Audio Processing*, 16:229–238, 2008.

[24] O. Das, B. Goswami, and R. Ghosh. Issues in utilizing performance metrics of the kalman filter in music enhancement. In *Frontiers of Research in Speech and Music*, pages 90–95, 2015.