Anton Khelou
260425637
MUMT-501
Final Project

Linear Prediction

Introduction

Linear prediction is a mathematical operation commonly used in signal processing in order to analyse discrete-time signals. Linear prediction essentially describes a way to estimate future values of a discrete-time signal based on the previous samples of that signal.

In the following report, I will be going over the general theory behind linear prediction, and elaborating on all-pole linear prediction. I will then explain the differences between two of the classical methods for parameter estimation of all-pole models: the autocorrelation method and the covariance method.

A section will be dedicated to my experiences implementing and analyzing the results of both methods. I will be going over a two cases, white noise and a sustained vowel as the inputs to the model, as well as describing the issues I ran into while working on this project.

Overview

Linear prediction is a technique that involves the estimation of future values of a discrete-time signal as a linear function of previous samples. As the name suggests, linear predictions makes use of properties of linear systems to achieve the expected results. Linear systems are defined as a system whose output is a linear combination of its previous outputs and its current and previous inputs. Further, a linear system can be defined as being time-invariant if a shift in the input causes a corresponding shift in the output. And so the general difference equation of such a linear system can be defined as follows:

$$\sum_{k=0}^{p} a_k y[n-k] = G \sum_{j=0}^{q} b_j x[n-j]$$

$$y[n] + \sum_{k=1}^{p} a_k y[n-k] = G \sum_{j=0}^{q} b_j x[n-j]$$

$$y[n] = G \sum_{j=0}^{q} b_j x[n-j] - \sum_{k=1}^{p} a_k y[n-k]$$

Where y is the output signal, x is the input signal, and scalars b_j and a_k exist according to their summation ranges (and $a_0 = 1$, $b_0 = 1$). This difference equation can be rewritten in the Z-domain as a transfer function H(z):

$$\sum_{k=0}^{p} a_k y[n-k] = G \sum_{j=0}^{q} b_j x[n-j]$$

$$\sum_{k=0}^{p} a_k z^{-k} Y(z) = G \sum_{j=0}^{q} b_j z^{-j} X(z)$$

$$H(z) = \frac{Y(z)}{X(z)} = G \frac{1 + \sum_{j=0}^{q} b_j z^{-j}}{1 + \sum_{k=0}^{p} a_k z^{-k}}$$

H(z) is the general pole-zero model – The roots of the numerator are the zeros and the roots of the denominator are the poles of the model, which are defined by the coefficients of the input and output signals. Makhoul describes three different types of models based on H(z):

- All-zero model, Moving Average (MA) model (generally non-linear approach). Assumes constant denominator.
- All-pole model, Autoregressive (AR) model (well understood, linear approach). Assumes constant numerator.
- Pole-zero model, Autoregressive Moving Average (ARMA) model. Makes no assumptions.

Since the purpose of this project is to explore two classical all-pole methods, autocorrelation and covariance, for parameter estimation, we will not be going over the MA and ARMA models.

As for the AR model, the problem statement involves a missing input signal sequence, and so we do not use it in the derivation. Also, all-pole models give systems of equations that can usually be solved efficiently. The all-pole model assumes that we have the linear combinations of past outputs and some scaled input:

$$y[n] = -\sum_{k=1}^{p} a_k y[n-k] + Gx[n]$$

However, in many cases, x[n] is totally unknown, which means that we can only give an approximation of the predicted signal based on the past values:

$$\tilde{y}[n] = -\sum_{k=1}^{p} a_k y[n-k]$$

We can then express the error between the actual value and the predicted value (called the *residual*) as:

$$e[n] = y[n] - \tilde{y}[n]$$
$$e[n] = y[n] + \sum_{k=1}^{p} a_k y[n-k]$$

We can also specify the total squared error over an unspecified range of signal samples as:

$$E = \sum_{n} e[n]^2$$
$$E = \sum_{n} \left(y[n] + \sum_{k=1}^{p} a_k y[n-k] \right)^2$$

The range will be specified depending on the method we use. The value of E gives a value of the energy of the residual. We can minimize E by using differential calculus (i.e. setting the derivative of E with respect to each predictor coefficient equal to 0):

$$\frac{dE}{da_i} = 0, \qquad 1 \le i \le p$$

The result is a set of equations, known as the normal equations:

$$\sum_{k=1}^{P} a_k \sum_{n} y[n-k]y[n-i] = -\sum_{n} y[n]y[n-i] \ \ where\ 1 \le i \le p$$

So, for any signal y[n], using the above, we can solve a set of p equations with p unknowns for the predictor coefficients a_k where $1 \le k \le p$. By expanding the above, we get a correlation matrix, which can be used to solve the linear system of equations

by using any of the standard numerical computing algorithms such as Gaussian elimination. This in turn gives us the necessary coefficients to create an all-pole filter to create a spectral envelope with respect to the signal used in the model.

When estimating the parameters, the way we set the bounds for summation of the total squared error result in different results in the prediction process – different normal equations which result in different predictor coefficients.

Autocorrelation Method

The autocorrelation method of linear prediction makes the assumption that the total squared error is minimized over an infinite duration $-\infty < n < \infty$. One thing that we need to consider when dealing with signals of infinite duration, we need to apply windowing in order to truncate the signal. Generally, a Hanning window, of size N, is applied to the signal to make all samples values outside a range equal to 0. We get the following squared error summation:

$$E = \sum_{n=0}^{N+p-1} e[n]^2$$

However, when applying a window, we can run into issues with the accuracy of the estimated parameter – If the signal being processed is a portion of an impulse response (i.e truncated in the during an impulse response), and not the entire impulse response, accurate parameter estimation cannot be ensured. However, this method ensures that the resulting filter after estimating the parameters will be stable.

Another important property of this method is exhibited by the fact that the error signal is zero outside of the analysis interval (due to the windowing) – the correlation matrix generated by the normal equations is Toeplitz symmetrical, which allows for efficient (due to the symmetrical redundancies along the right diagonal) solving of the system of linear equation with the use of Levinson-Durbin algorithm.

Covariance Method

In contrast to the autocorrelation method, the covariance method assumes that the total squared error is minimized over a finite interval $0 < n < N - 1$. We get the following squared error summation:

$$E = \sum_{n=0}^{N-1} e[n]^2$$

Because we do not apply any windowing techniques to the signal, the samples outside the prediction error interval are not equal to 0. We require –p samples outside the interval in order to perform the prediction (i.e. sample at n=0 needs –p samples before it). However, this method does not ensure that the resulting filter will be stable (though in practice, if N is sufficiently large, then the filter will be stable).

Linear prediction coding

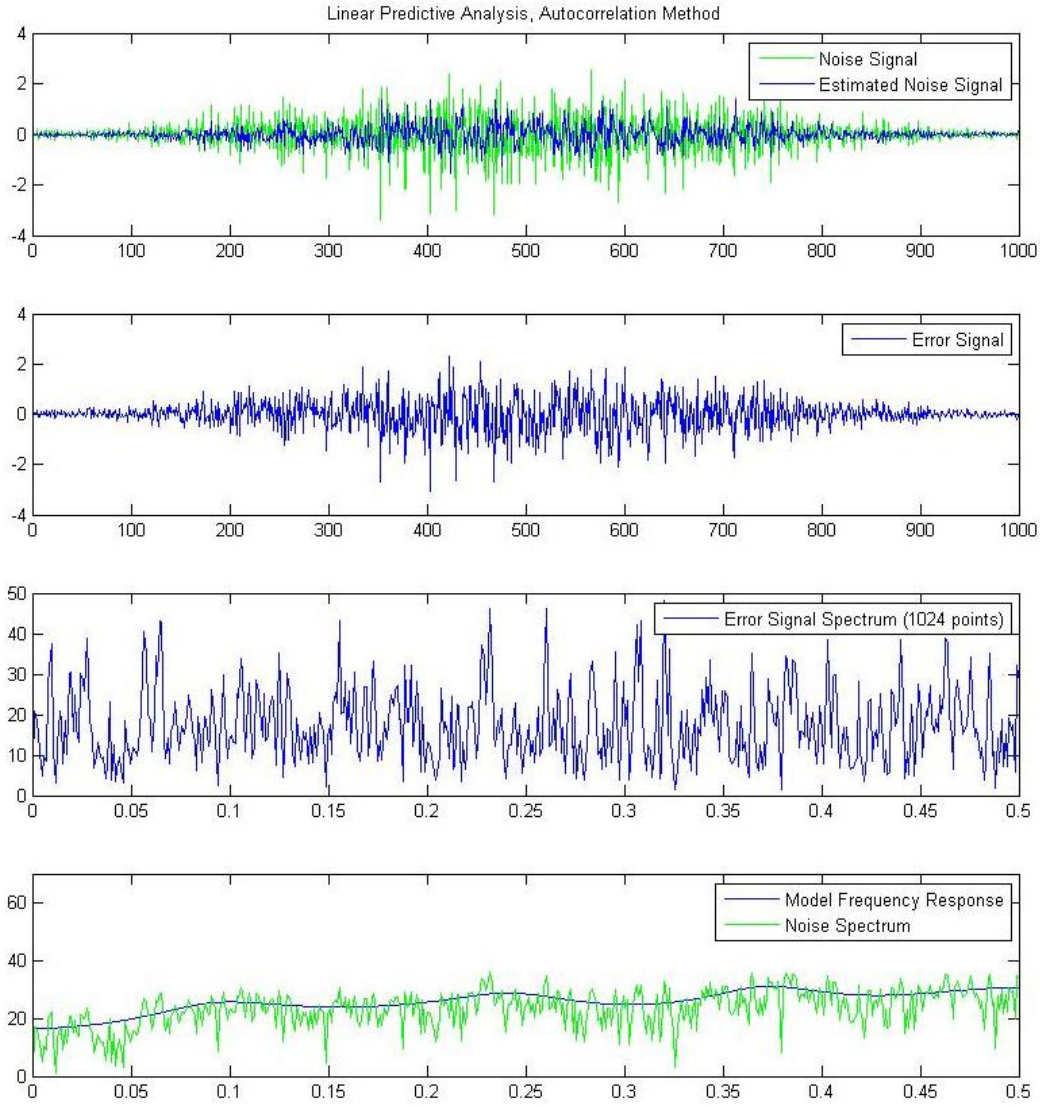
An important relationship arises from the residual signal with respect to the original signal – The residual signal is essentially the compliment to the predictor coefficients used in the filter. This means that, if we use the residual signal to excite the generated filter, we can reconstruct the original signal. This relationship is at the core of codecs using LPC. Essentially, the error signal is encoded and attached with it is the set of coefficients generated from the linear prediction process. Because the error signal contains less information, the encoding will be more compact, yet the quality will remain the same. LPC is generally used in lossless formats, such as FLAC.

Experiments and Observations

Experiments were run on both of these methods with both white noise and sustained vowel as the inputs to the model. These experiments were implemented in MATLAB. The following section contains one set of graphs. The full set of graphs for all experiments ran, can be found at the appendix.

White noise experiment

Similar conclusions can be made for both the autocorrelation and covariance methods. Firstly, the major observation is the fact that the estimated signal does not seem to improve as we increase the order of the model. Though, it can be noticed that the order of the model improves the shape of the filters' frequency response (using the predicted coefficients) with respect to the frequency spectrum of the original noise signal. It can also be noted that the spectrum of the error signal, is white noise (which is what is expected), with all the orders tested. In fact, the error signal's spectrum contains nearly the same frequencies as the spectrum of the white noise inputted into the model. This indicates that the prediction of coefficients was in fact reasonably successful, as there is an error in, visibly, most frequencies that were present in the input noise signal. But besides that, it can be visually seen that the estimated signal, in all values of order, is very similar in shape to the original noise signal. The following shows a set of graphs with various tests used in the above observations:

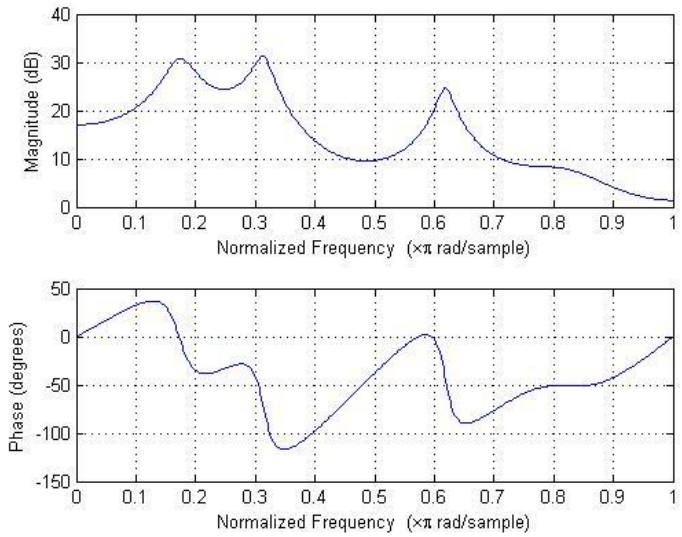


Sustained vowel experiment

Once again, similar conclusions can be made for both the autocorrelation and covariance methods. All tests were run on a recorded sample of me sustaining the ‘a’ vowel. The wave file was recorded at a sampling rate of 8kHz. Although other sampling rates were tested as well, I was not able to understand the results - this will be talked about in a later section.

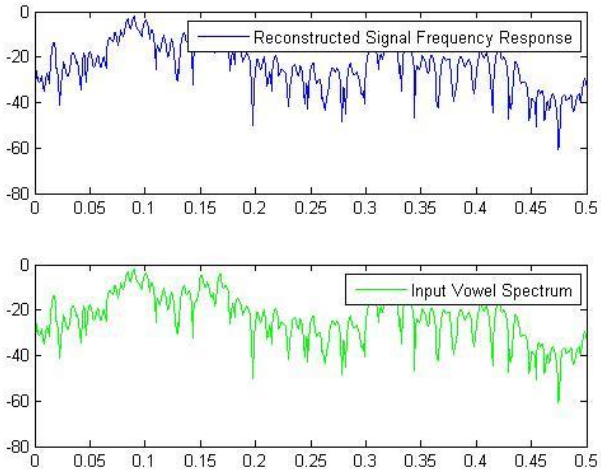
In the case of inputting a sustained vowel sound, the order of the filter affected the results dramatically. The most important observation to be made is the importance of resonances in the frequency response of the filter associated with the increase in order of the filter. As the order increases, it can be observed that formants associated with the vowel ‘a’ start appearing in the frequency response of the filter that we are creating.

These formants seem to be the most defined in between the range for 10 to 20 as the order of the model, and after passing an impulse train through the filter, they sound the best as well. Beyond that, as the order of the filter increases, into values over 30, we can see that the frequency response starts becoming polluted with formants that are seemingly harmonics of the input vowel sound – This can be noticed especially well when an impulse train sound is passed through the filter. However, if we set the order to a value that is too low, the prediction will not use enough previous samples in its estimation of parameters which will cause the resulting filter to lack resonances that provide the necessary frequency shaping to the signal inputted into the generated filter. The following is the frequency response of a filter created with linear prediction autocorrelation method of order 10:



After listening to the estimated vowel sound and the error signal (between the original and estimated signal) from an order 10 linear prediction, I noticed that the estimated vowel retained most of the vowel information and the error signal had most of the buzzing, nasal sound of the original signal. I believe this is a good sign, as this indicates that the prediction process is able to detect the right formants associated with the vowel ‘a’ used in the model, resulting in a more accurate spectral envelope in the filter.

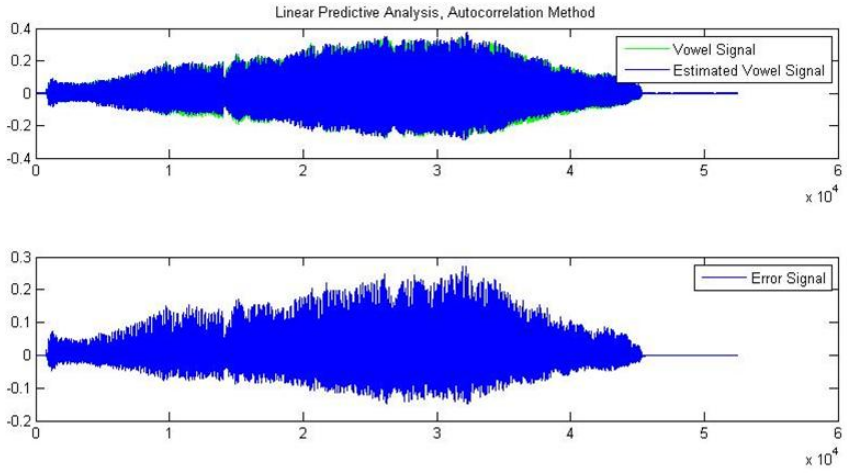
Linear predictive coding was tested as well. The residual signal was passed into the filter, and as expected, the original signal was outputted. The following graph shows the reconstruction using the autocorrelation method of order 10:

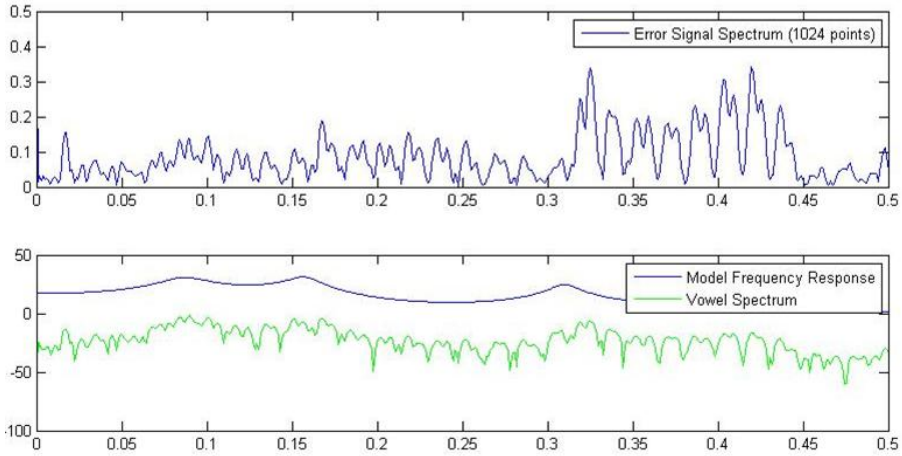


When looking at the graphs comparing the estimated vowel signal and the original vowel signal, we can notice that the signal is very similar for all order values tested. However, in the case of vowels, we can clearly notice that the estimated signal is improving as we increase the order of the filter, just by visually looking at the graph illustrating the vowel signal and the estimated vowel signal. Further, we can notice that the energy across frequencies in error signal spectrum becomes more evenly distributed as the order of the model increases – This is due to the more accurate prediction of the signal (more resonances appear in the filters' frequency response, which lead to a more faithful filter with respect to the spectrum of the original vowel signal).

The only noticeable difference between the autocorrelation and covariance method, is the exact shape of the formants – In both cases, the general shape is the same, and so the same resonances appear in both cases, but the magnitude of the peaks is difference and the frequencies around the peaks are slightly different.

The following set of graphs show the results of the autocorrelation method of order 10:





General difficulties, issues and possible improvements

To be honest, it was difficult for me to wrap my head around this topic and getting to this point in my understanding. It wasn't clear to me what the purpose of linear prediction (calling it prediction confused me quite a bit) was and why it was used. I think everything started to come together once I started implementing the systems, playing around with it and analyzing the results.

The graphs helped quite a bit with solidifying the theory, and the results were somewhat convincing, but I still wasn't completely sure if the results I was getting were correct. The reason being is that, I knew that the idea behind linear prediction was essentially to create a filter, and I knew that the resulting filter should shape an input signal according to the modeled signal. The issue that kept me from believing my results was the lack of sound tests when I had initially started implementing everything. Once I started experimenting with all the signals generated during the entire process is when everything started making sense. Also, it took me a good amount of time to get the implementation up and running. The code is based on some code I had found online, but there were issues with the code (did not seem to match the theory suggested in the Makhoul paper nor the documentation provided by MATLAB). Another thing I am not totally sure about is whether or not the windowing is necessary for the autocorrelation method, as the matlab documentation seems to indicate that it's implicit, but I could not see it in the resulting graphs. It also took me a while to figure out that *arcov* is the MATLAB implementation of linear prediction with the covariance method (according to some code I found on Gary Scavone's website).

Another issue I ran into was when I decided to use a sustained vowel signal at a higher sampling rate. None of it was making any sense to me. I could not understand why the prediction coefficients were so off with respect to the spectrum of the input signal. After

further research, I found online that when working with higher sampling rates, some form of frequency warping needed to be applied in order to get correct results.

As far as improvements go, I think the next step would be to implement some sort of way to figure out what is the optimal order for a model, algorithmically, perhaps based on error signal thresholds. Implementing frequency warping would be another important thing to implement to support high sampling rates.

Conclusion

Although I still feel that my understanding of linear prediction is quite limited, I feel that I have learnt a significant amount. I can see why linear prediction is such a powerful tool in signal processing but there are still many unanswered question as to the exact reasons why it works the way it does. I hope to eventually be able to put some of this knowledge to use in the areas of study of Computer Science that I will be pursuing in the future.

References

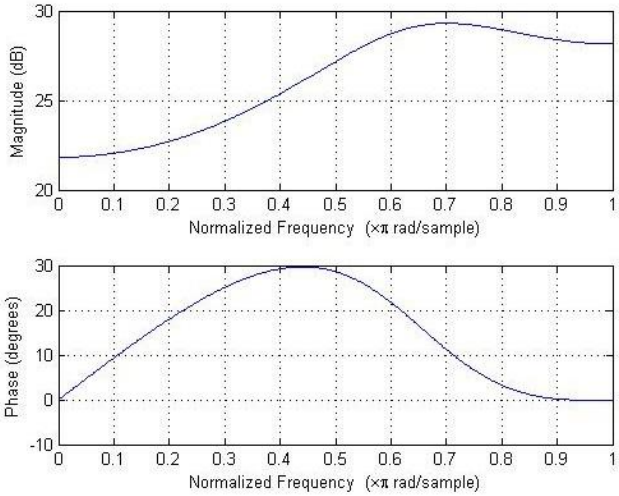
J. Makhoul.
Linear prediction: A tutorial review.
Proceedings of the IEEE, 63 (4): 561-580, Apr. 1975.

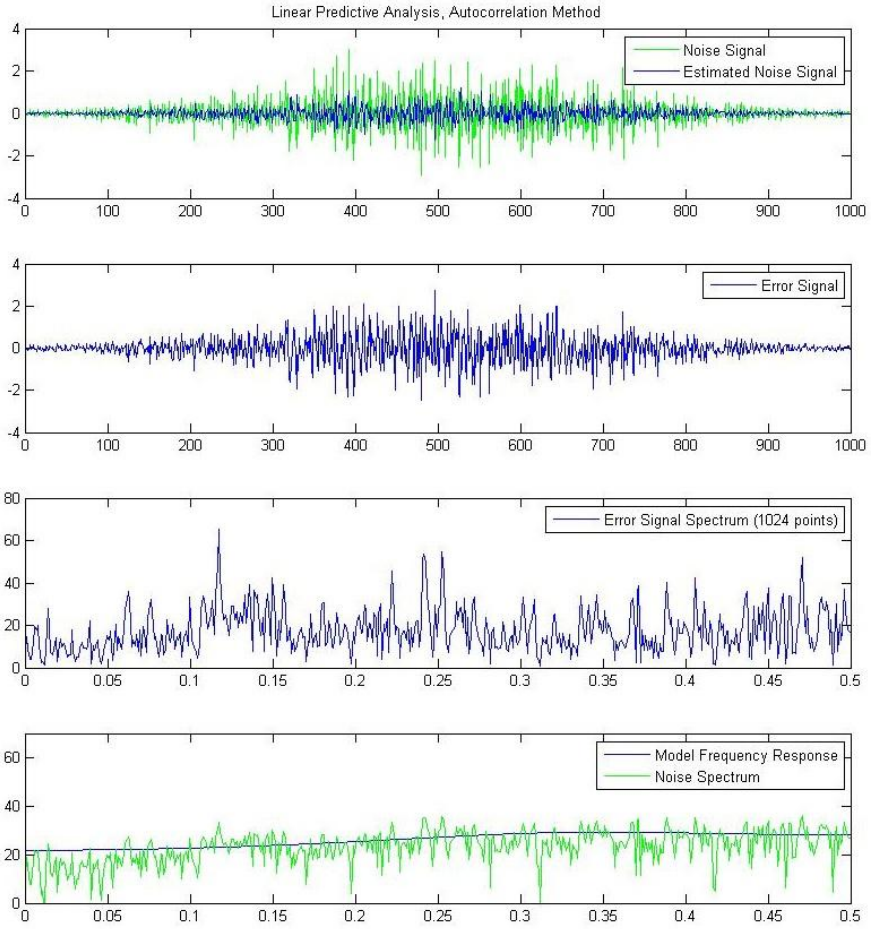
G. Scavone.
Linear prediction.
http://www.music.mcgill.ca/~gary/307/week9/lpc.html

N. Katsamanis
Speech Processing using MATLAB.
http://cvsp.cs.ntua.gr/~nassos/resources/speech_course_2004/OnlineSpeechDemos/speechDemo_2004_Part1.html

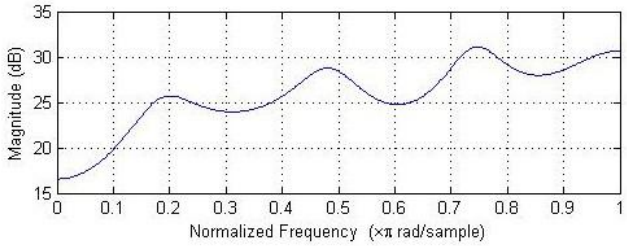
Appendix

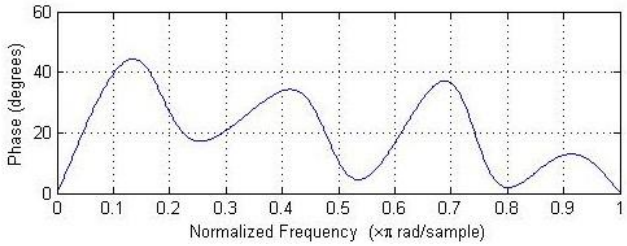
Noise Input, Autocorrelation, Order 2

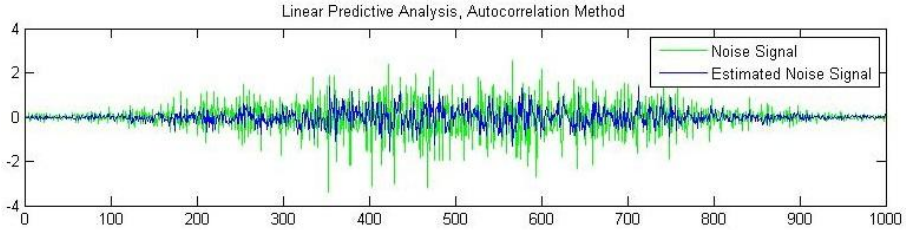


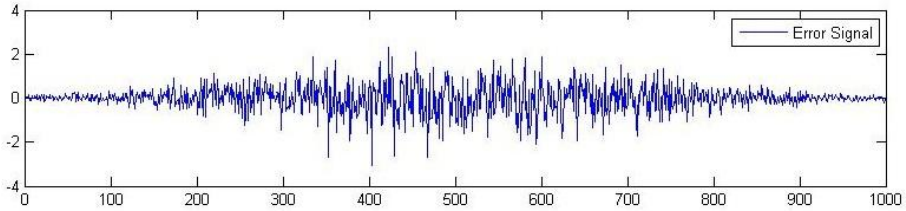


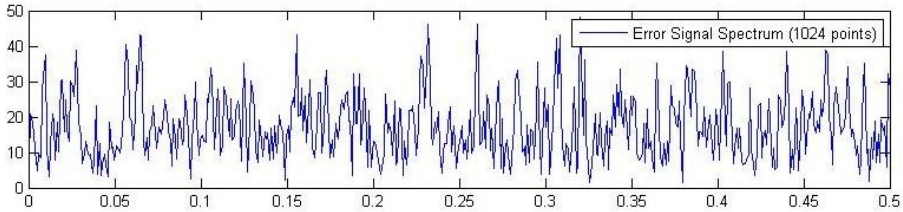
Noise Input, Autocorrelation, Order 10

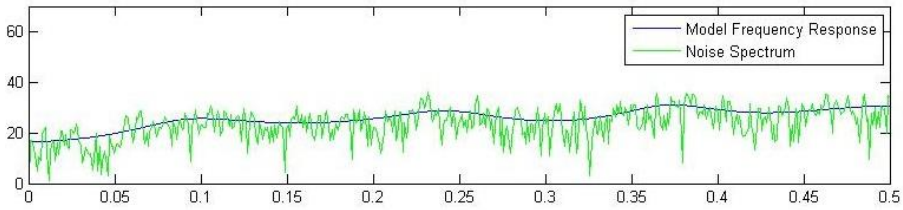




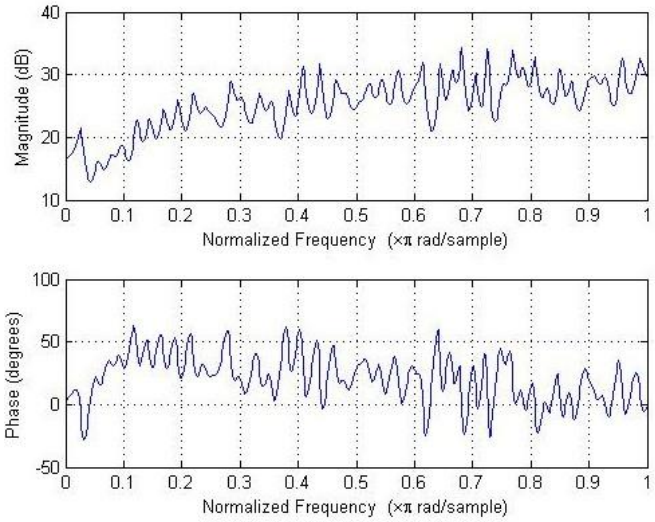


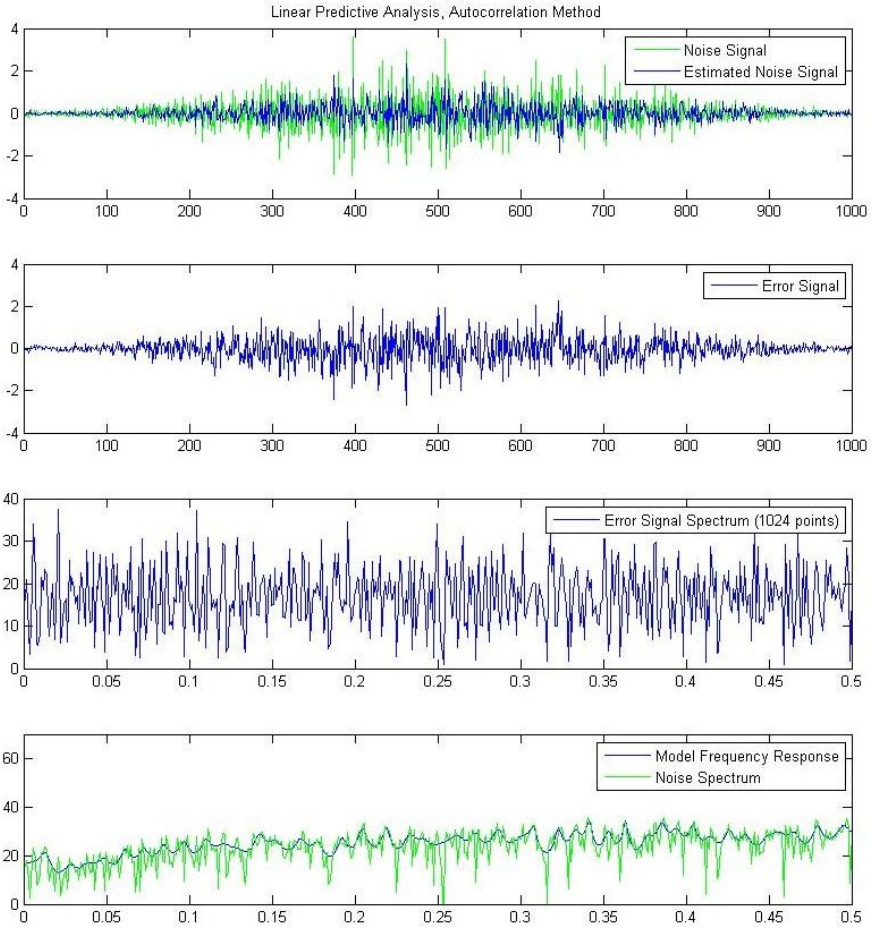




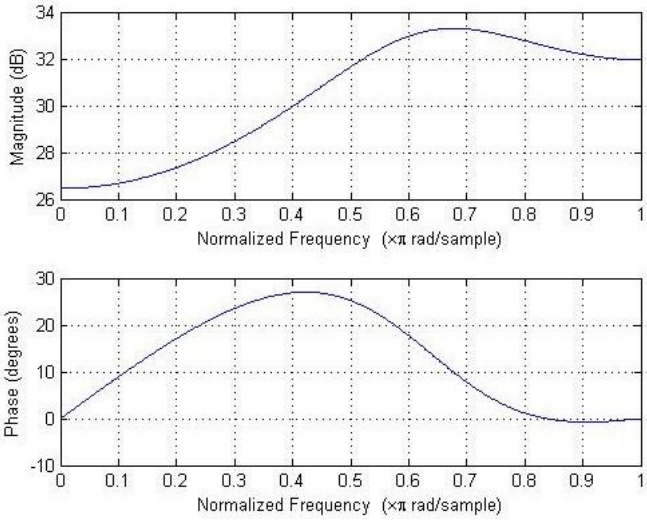


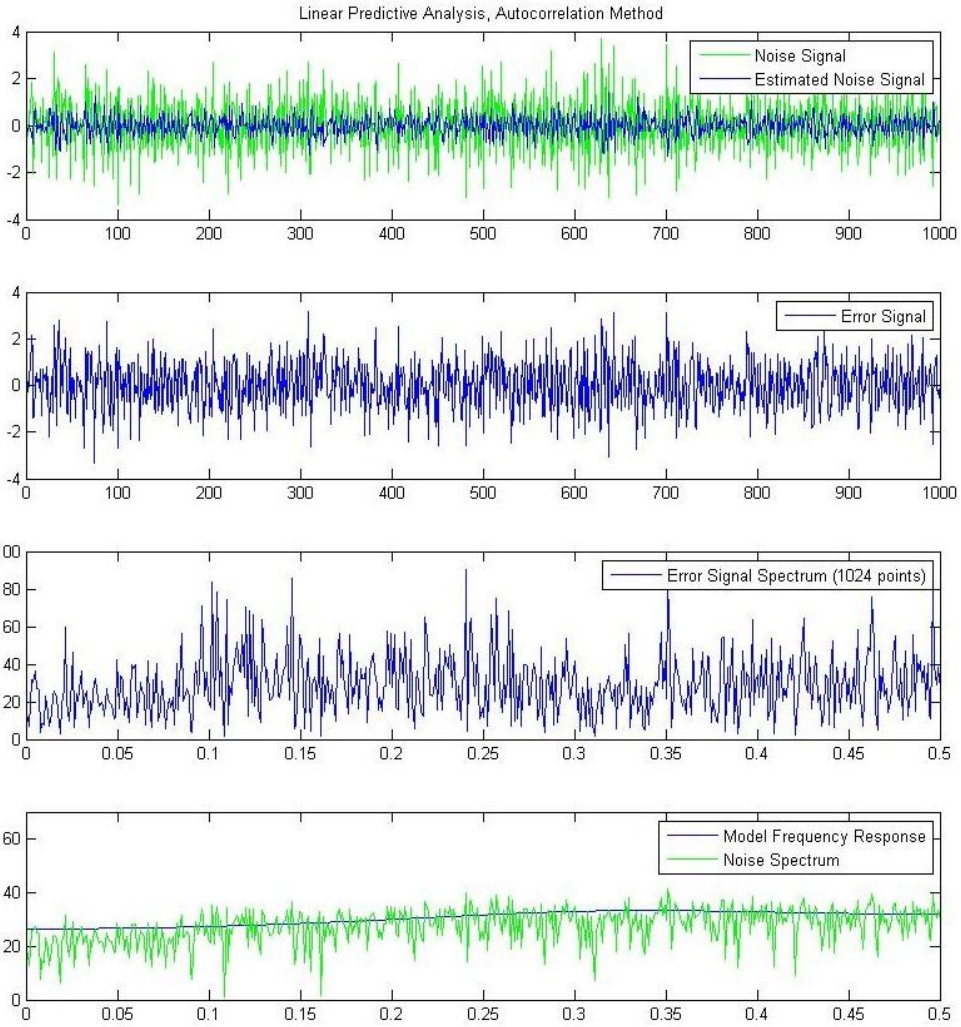
Noise Input, Autocorrelation, Order 100



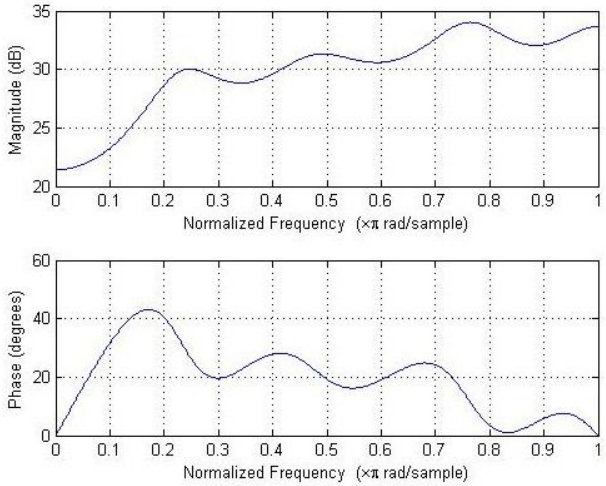


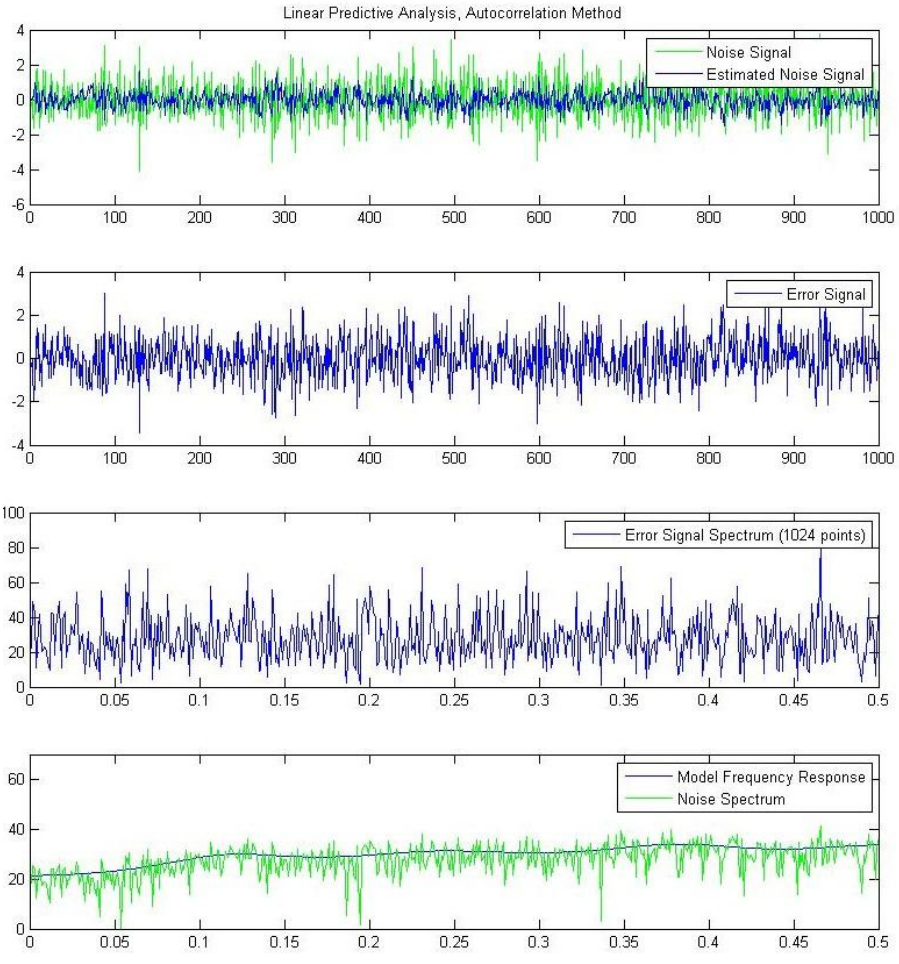
Noise Input, Covariance, Order 2



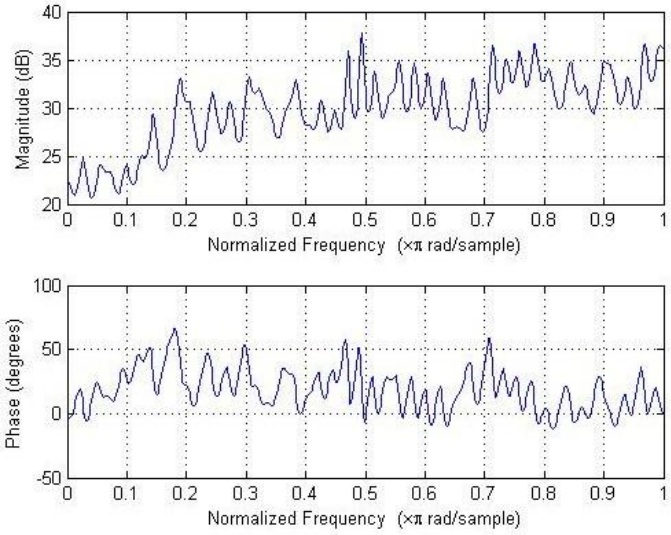


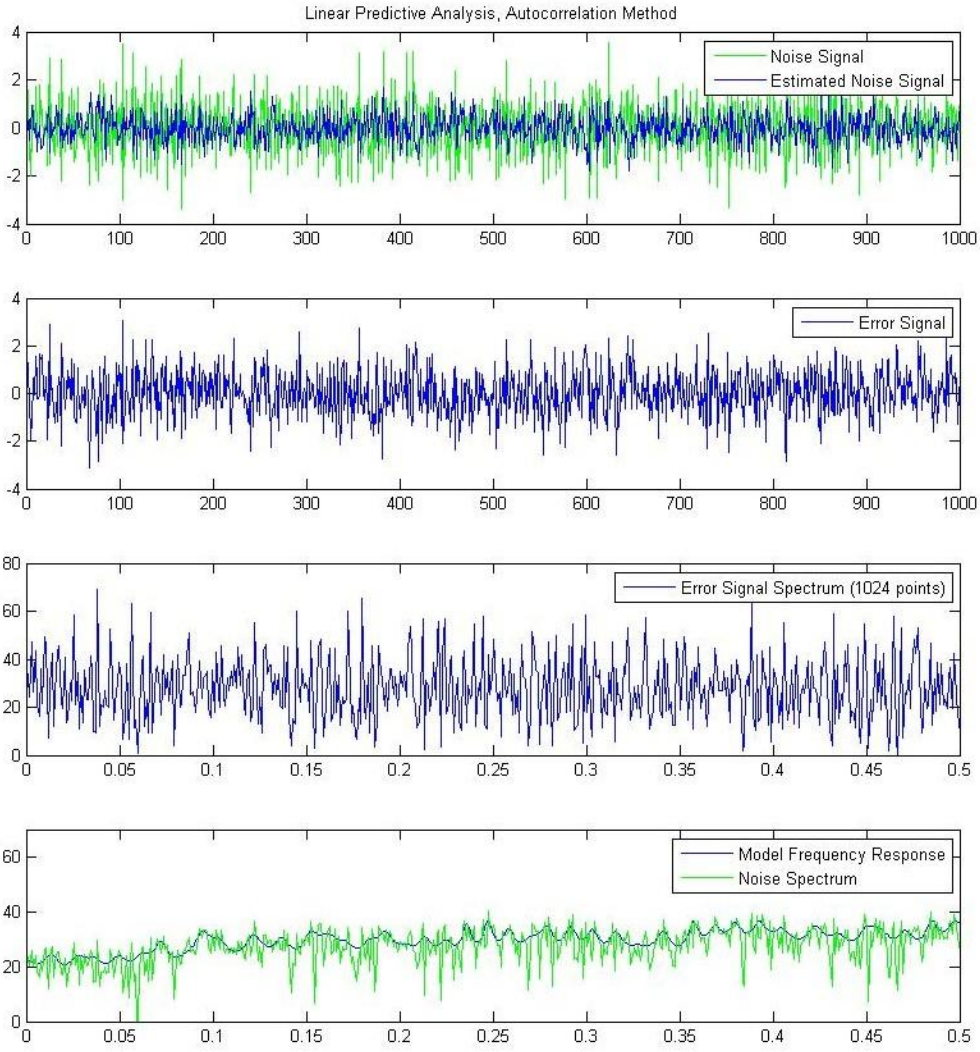
Noise Input, Covariance, Order 10



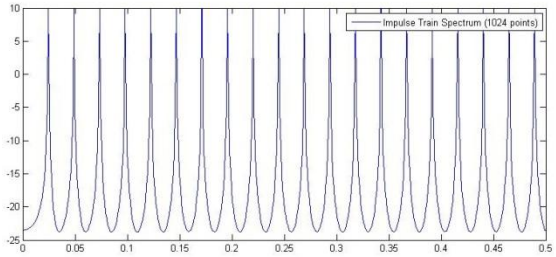


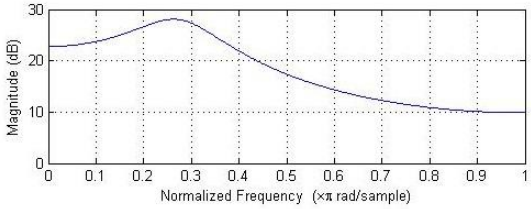
Noise Input, Covariance, Order 100

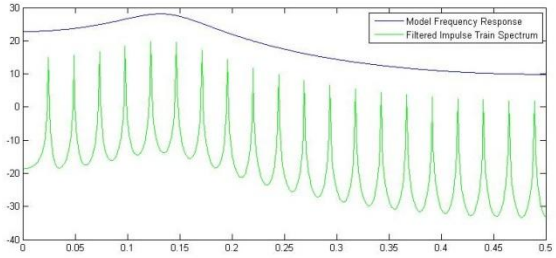


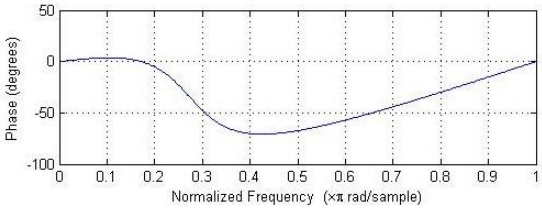


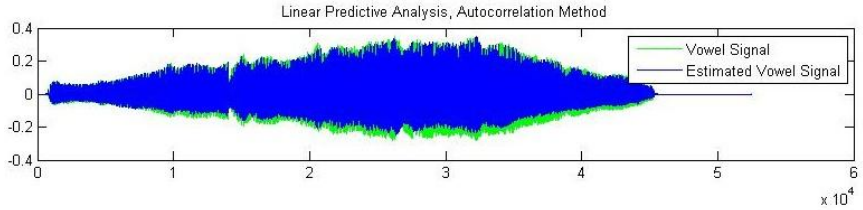
Vowel Input, Autocorrelation, Order 2

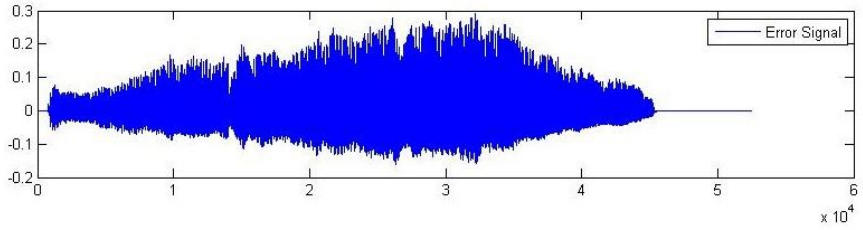


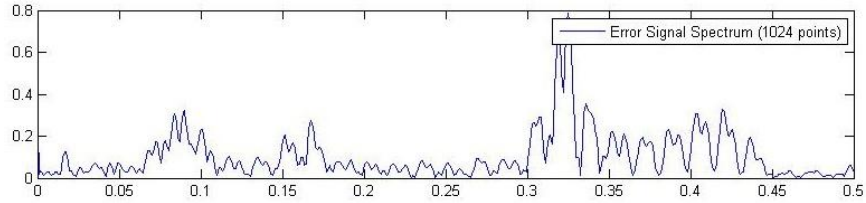


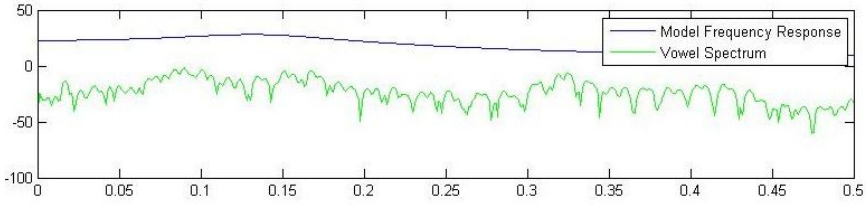




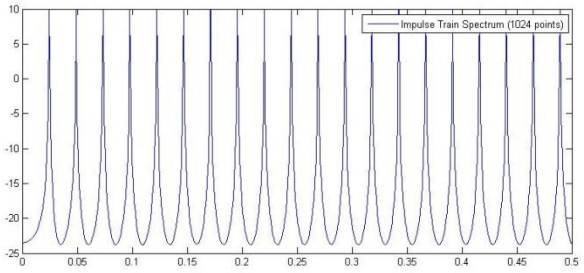


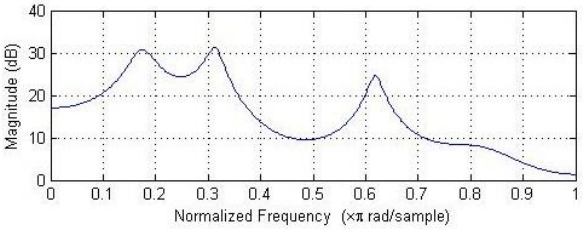


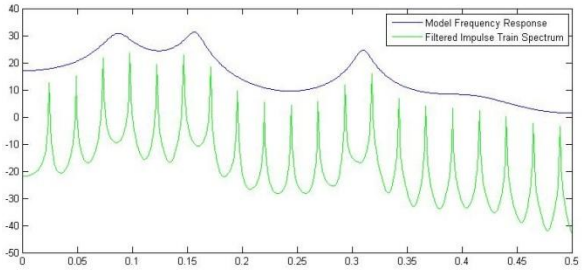


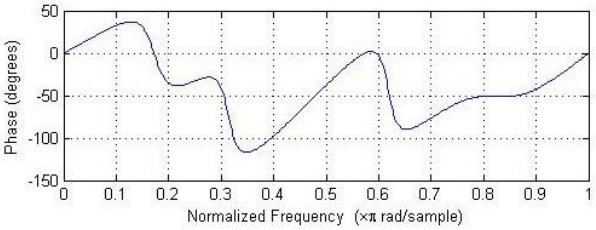


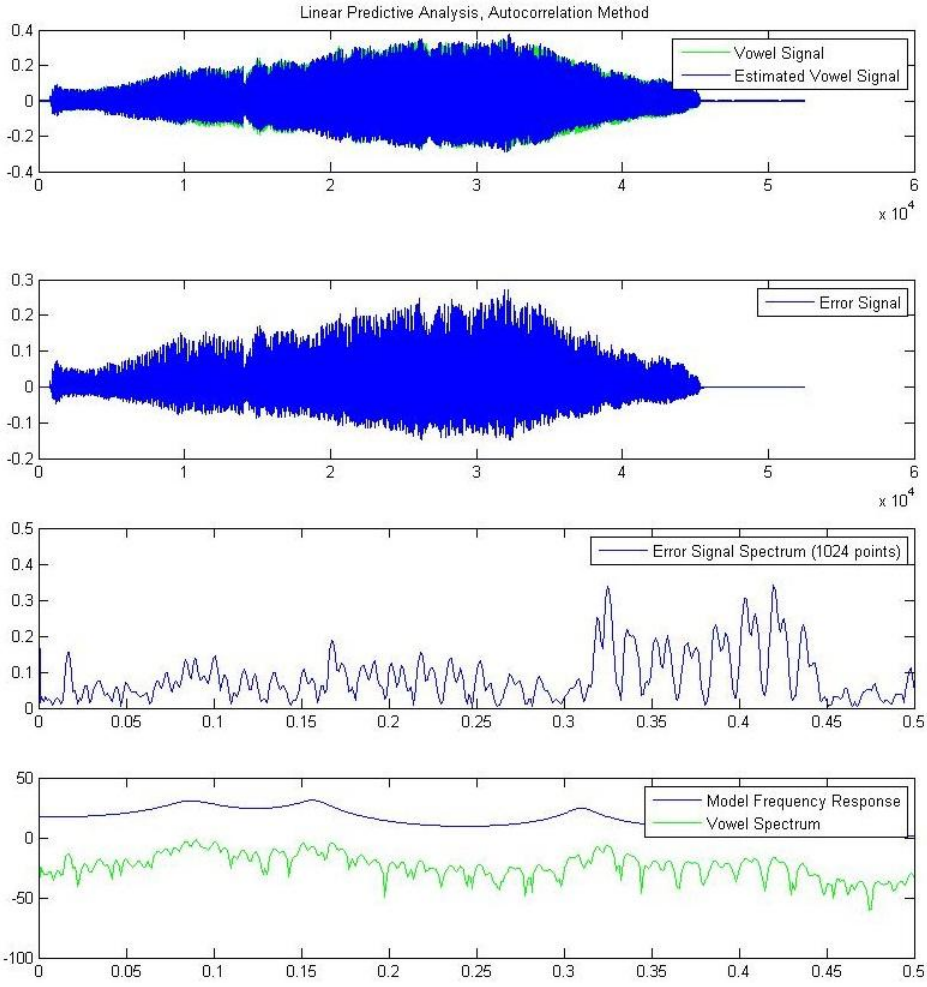
Vowel Input, Autocorrelation, Order 10



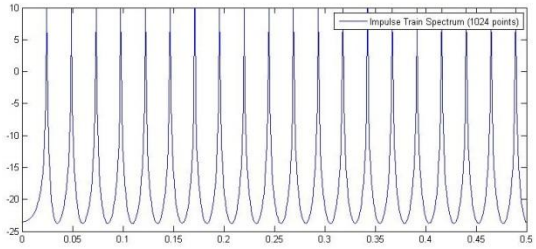


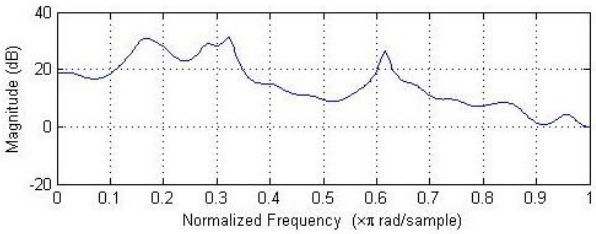


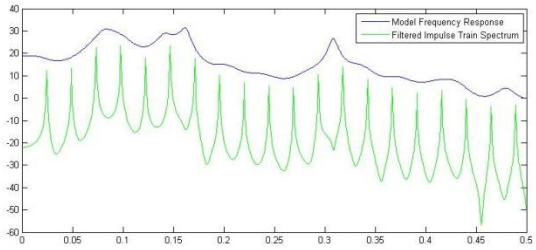


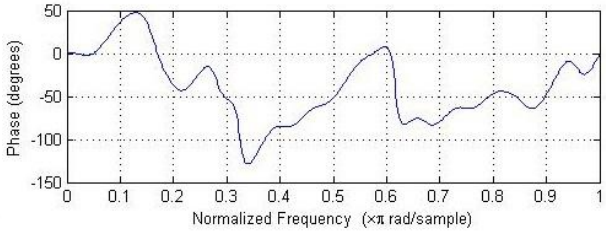


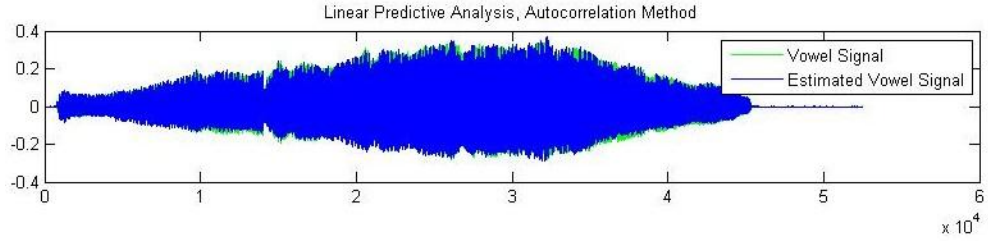
Vowel Input, Autocorrelation, Order 30

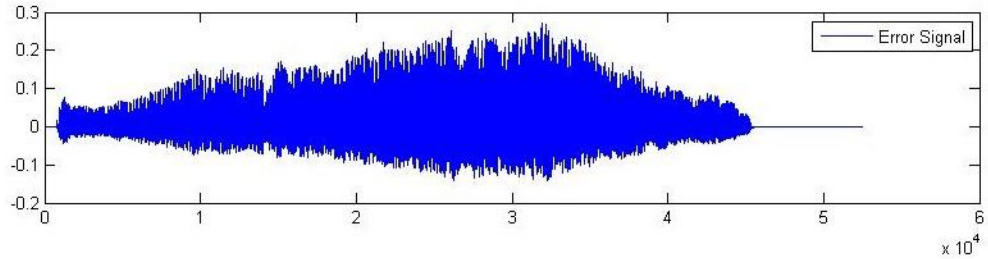


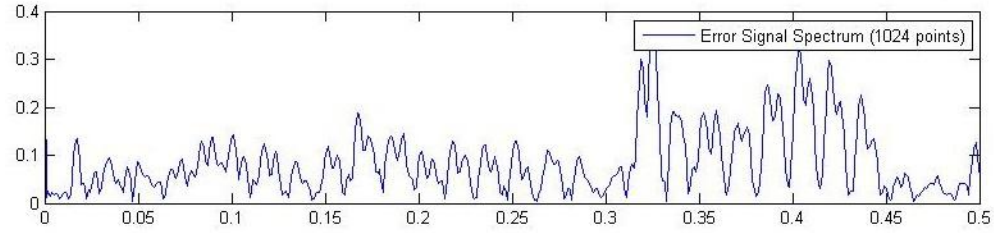


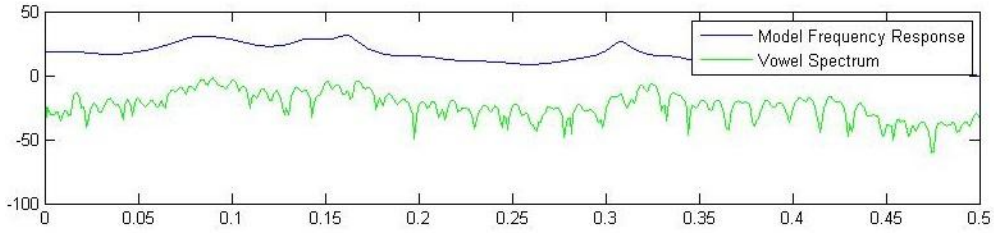




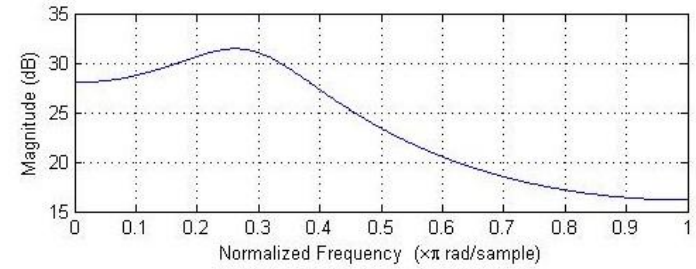


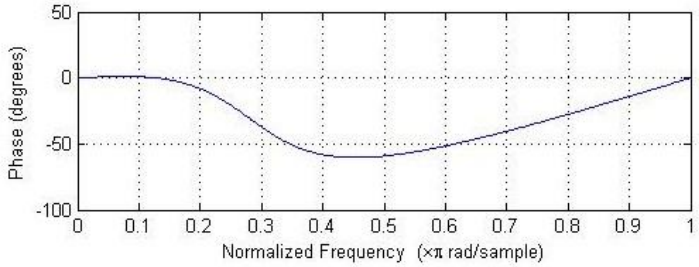


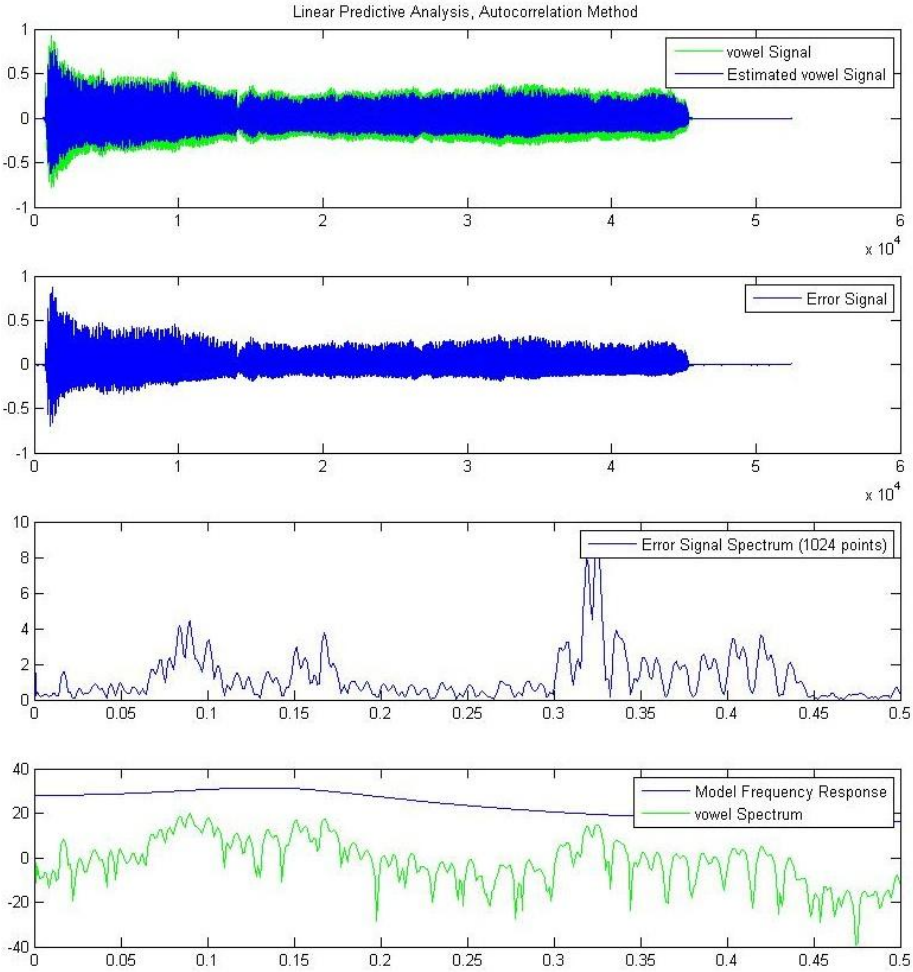




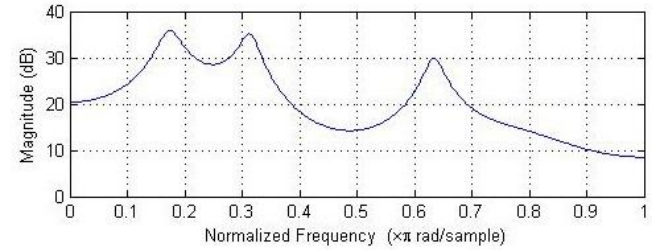
Vowel Input, Covariance, Order 2

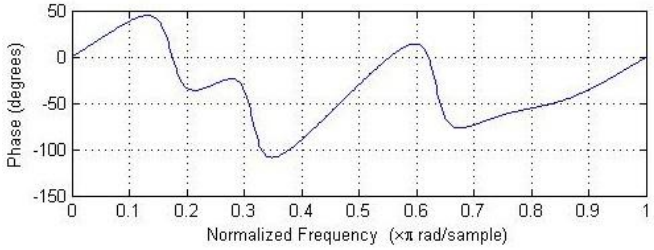


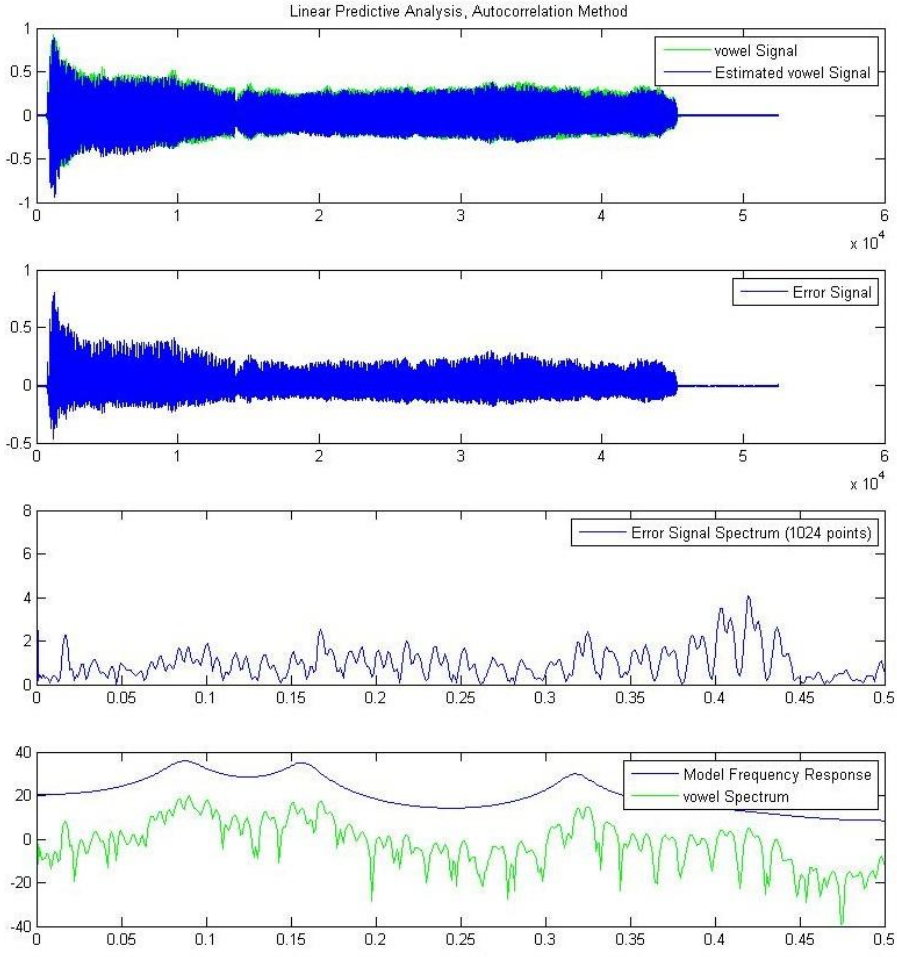




Vowel Input, Covariance, Order 10







Vowel Input, Covariance, Order 30