EE679: Speech Processing
Computing Assignment 3

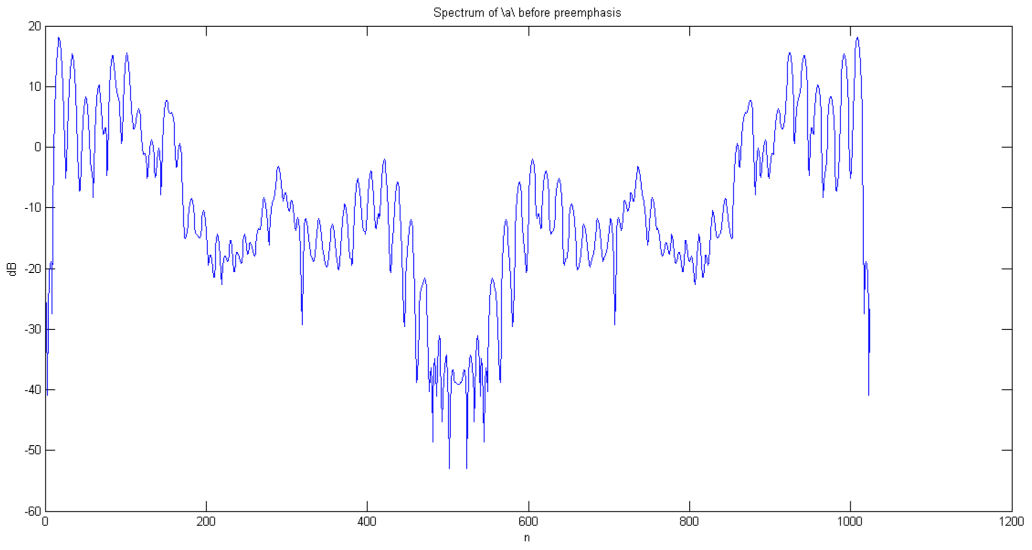
Ashwin Kachhara
10d070048

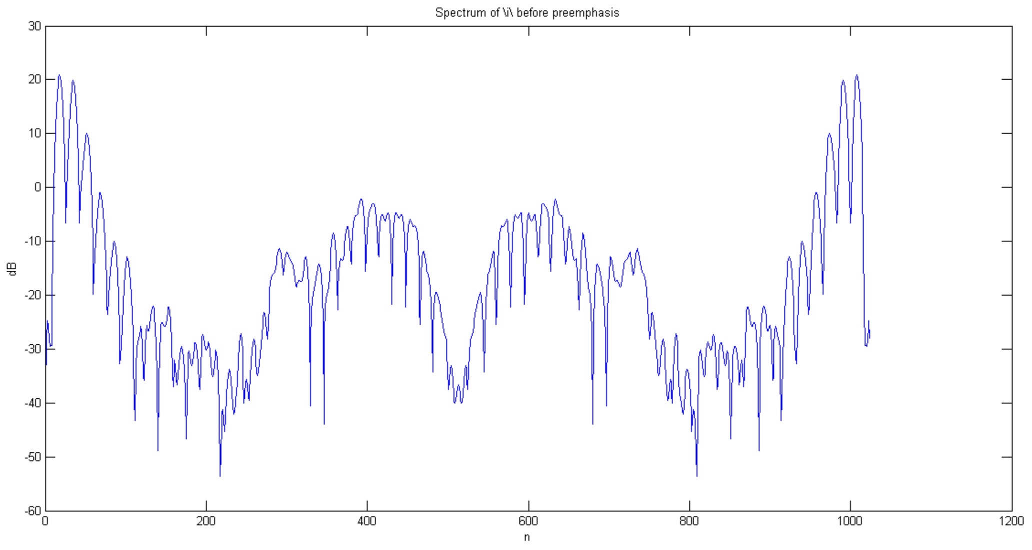
Q1

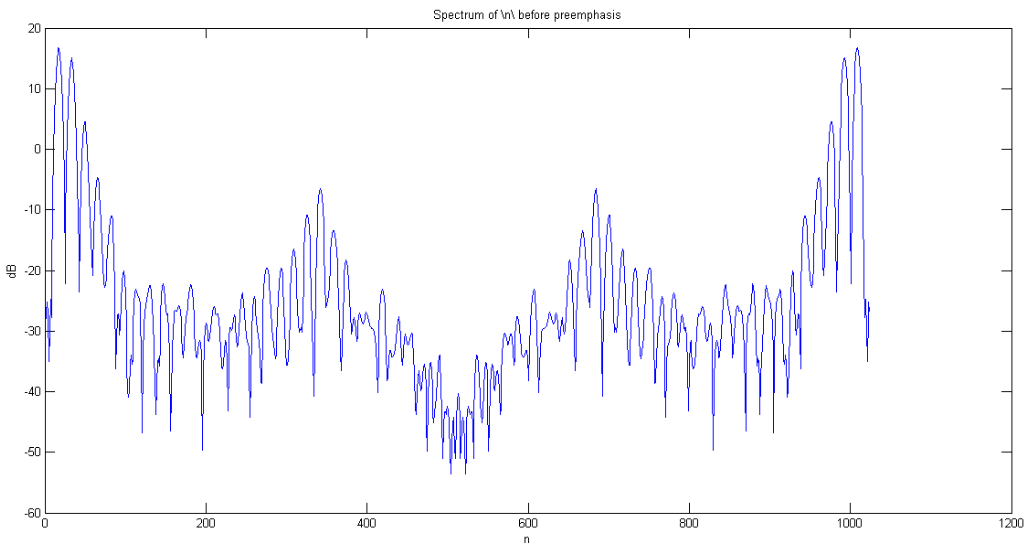
Narrowband spectrum of following sounds using Hamming of duration 30ms

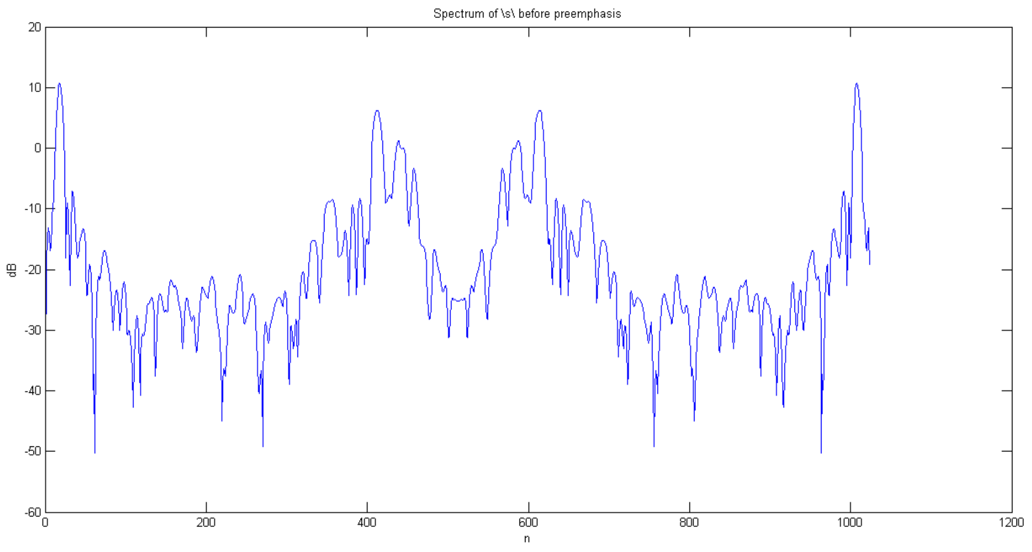
30ms duration and 8000 Hz sampling frequency, means that the window length turns out to be 240 samples

Before Pre-emphasis







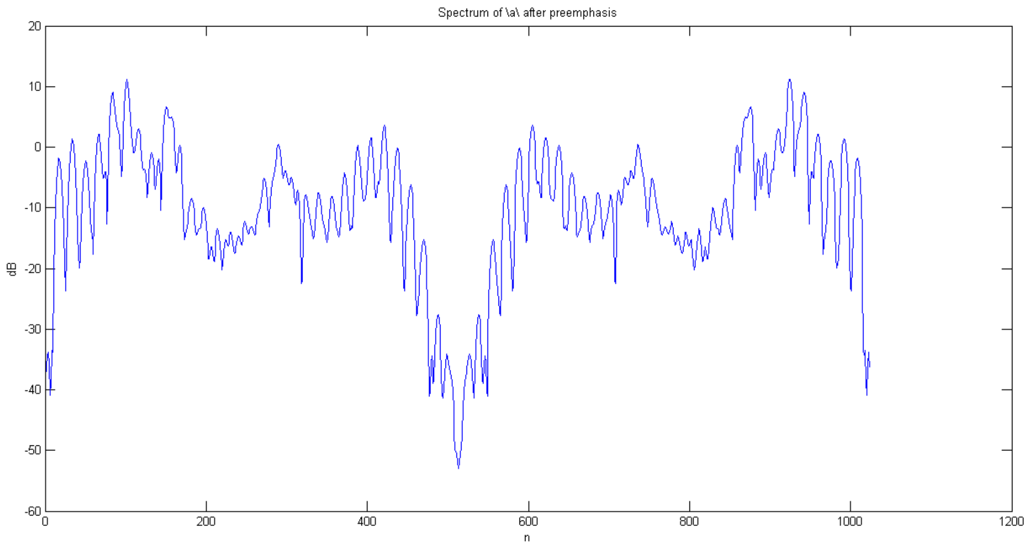


After Pre-emphasis

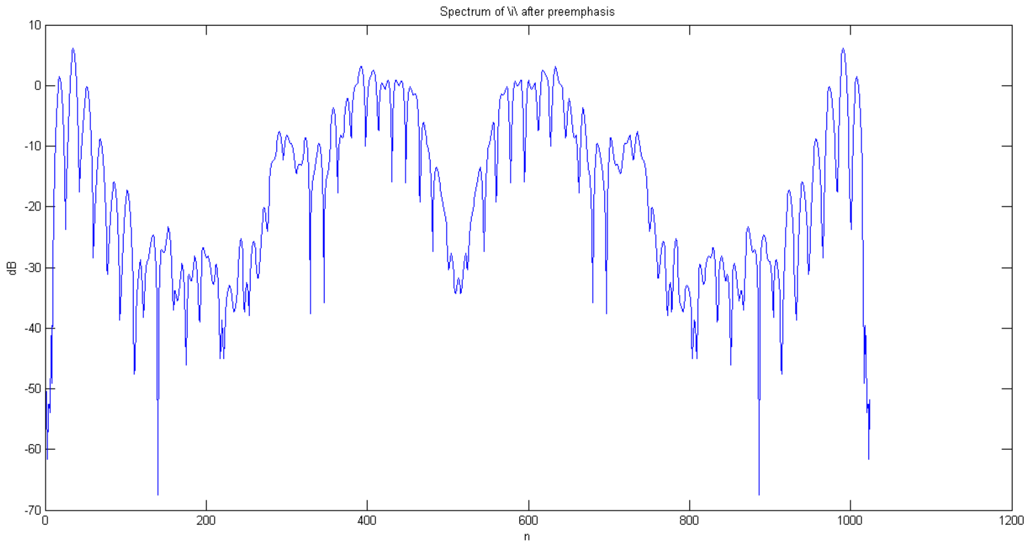
Pre-emphasis is basically a first-order high pass filter. This is done to emphasize the higher frequency energy in speech.

$$y[n] = x[n] - a * x[n-1]$$

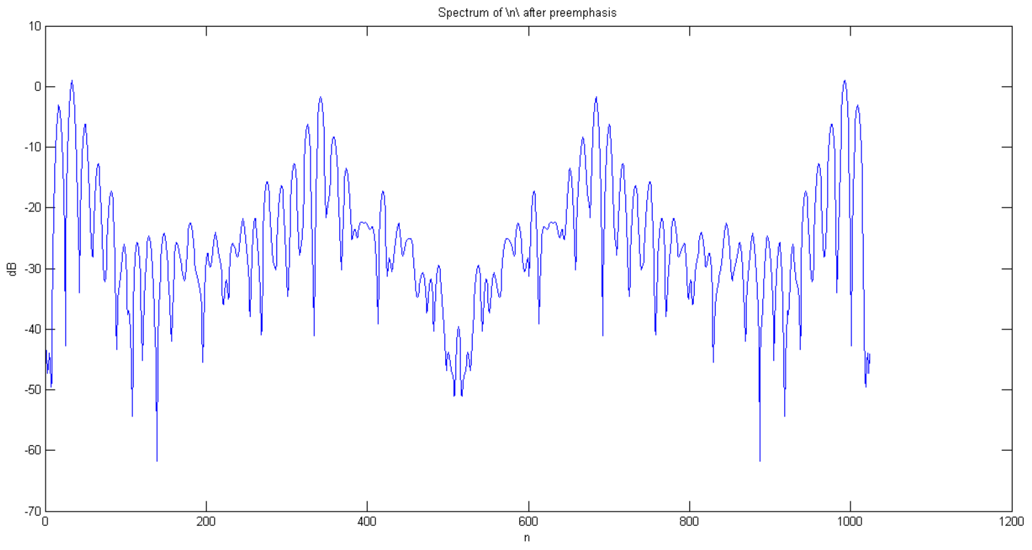
Here 'a' is typically a value in the interval [0.95, 1). In my code, I have taken 'a' to be 0.975. The resultant spectrum of each phone is as follows:

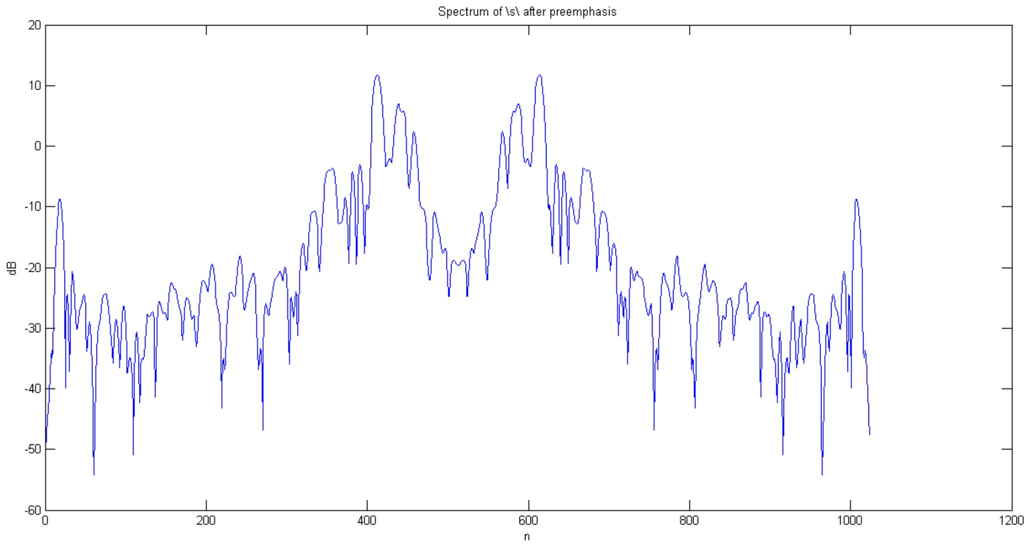


Spectrum of \a\ after preemphasis
dB
n
20
10
0
-10
-20
-30
-40
-50
-60
0
200
400
600
800
1000
1200



Spectrum of \i\ after preemphasis
dB
n
10
0
-10
-20
-30
-40
-50
-60
-70
0
200
400
600
800
1000
1200





Q2

First of all, after pre-emphasizing the waveform, we multiply it with a centered hamming window of length Lw. Also, we calculate its power spectrum spect_a

```
W = hamming(Lw);
Sw = zeros(Lw,1);
strt = floor(Ls/2-Lw/2);
Sw(1:end) = S(strt:strt+Lw-1).*w;
spect_a = (abs(fft(Sw, 1024)).^2)/1024;
```

The advantage of pre-emphasis here is that we can obtain better modeling of the spectrum, since pre-emphasis removes/reduces the spectral tilt.

To obtain a linear predictive model for the signal, first we need to calculate the autocorrelation sequence. We can do that as follows and calculate all 20 required estimates (for a max predictive model of order 20)

```
p = 20;
r=zeros(p+1, 1);
for i=0:p,
    r(i+1) = Sw(1:Lw-i)'*Sw(i+1:Lw);
end
```

To calculate the LPC, we use the Levinson-Durbin Recursion. This is implemented in the *levinson()* function in MATLAB, which returns the LPC coefficient and the error variance.

```
[a2 e2] = levinson(r, 2);
```

The above obtains order 2 LPC, where a2 is the LPC(2) coefficients, e2 is error variance and r is the autocorrelation sequence that we had calculated earlier. The same is repeated for orders p = 4, 6, 8, 10, 12, 20.

Magnitude spectrum of LPC = square root of the Power spectrum of LPC

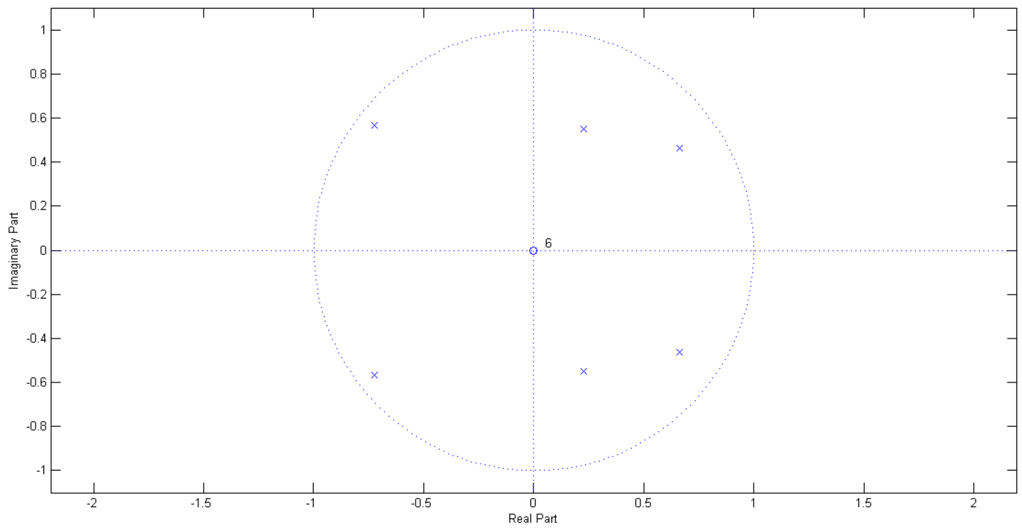
```
Pspec(k+1) = e/(abs(expo*a')).^2
```

Square root of it, is the LPC magnitude spectrum.

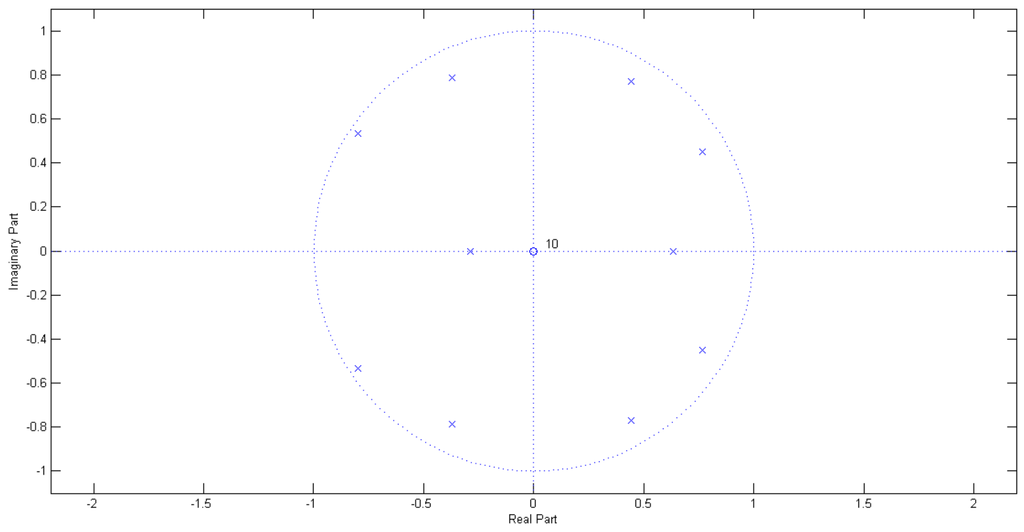
As you will see soon, the p=20 model appears to best follow the shape of the original spectrum. Also, the error signal energy decreases as the model order increases, as we can more accurately model the signal if given more no. of poles to use.

\a\

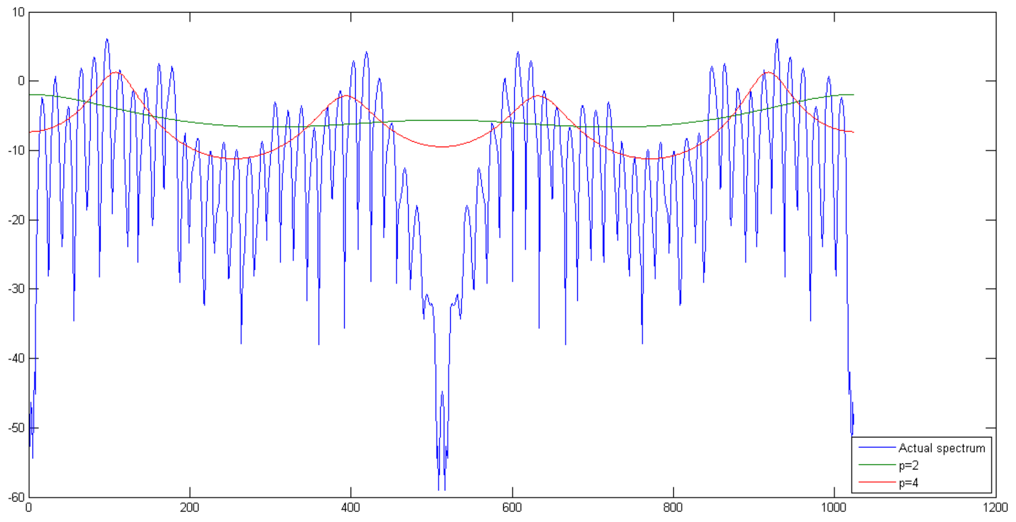
Pole Zero Plot| p=6

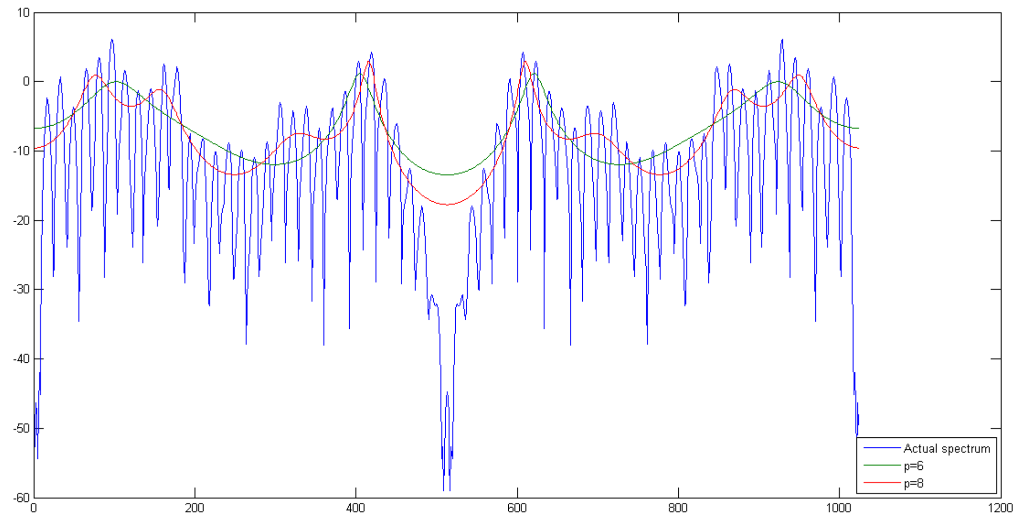


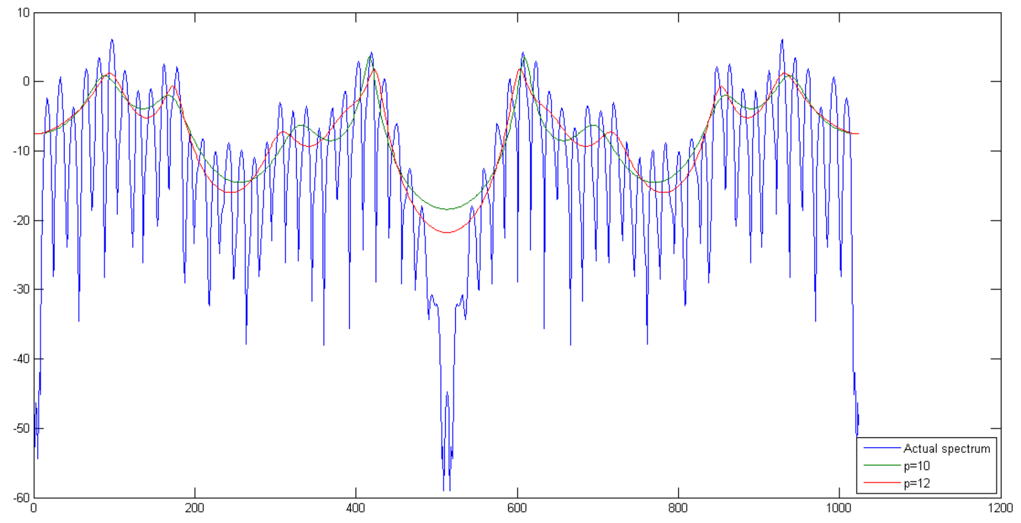
Pole Zero Plot | p=10

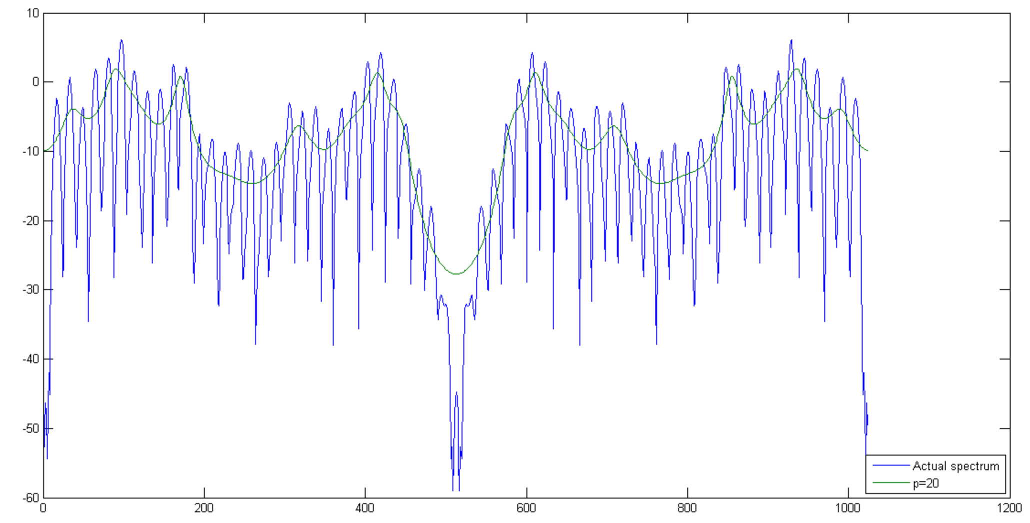


Magnitude Spectrum Plots for various LPC models of order p



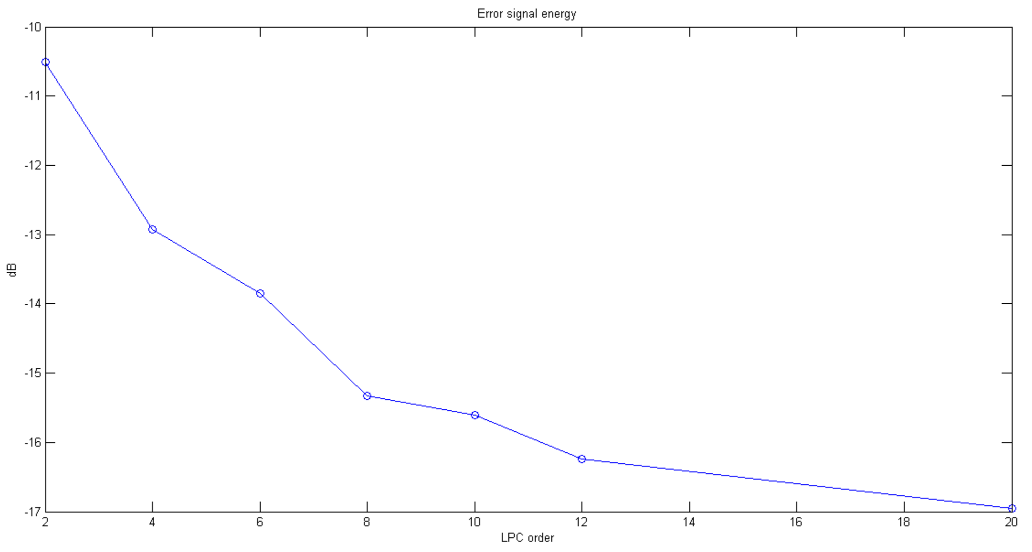

Actual spectrum
p=6
p=8


Actual spectrum
p=10
p=12



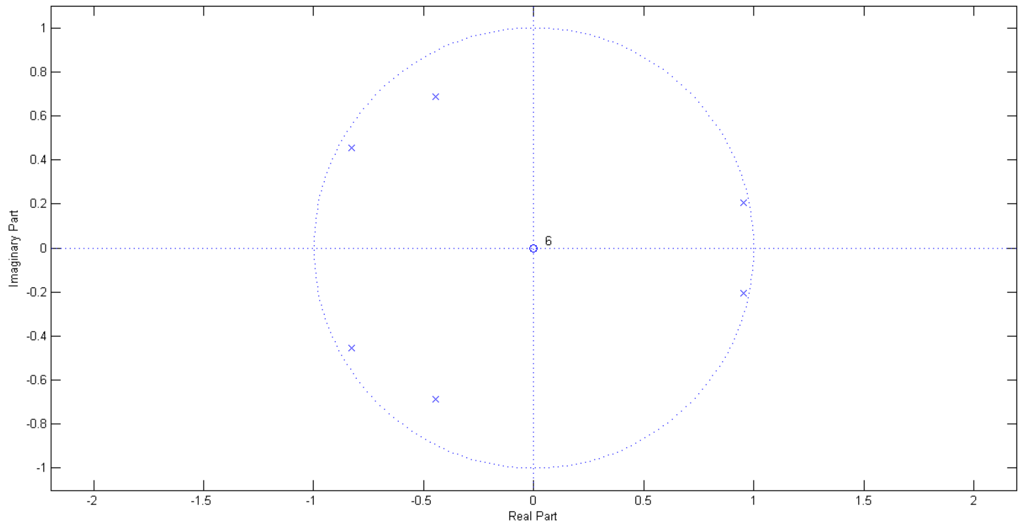
Error Signal Energy vs p

Error Signal Energy is simply the error variance, e_i

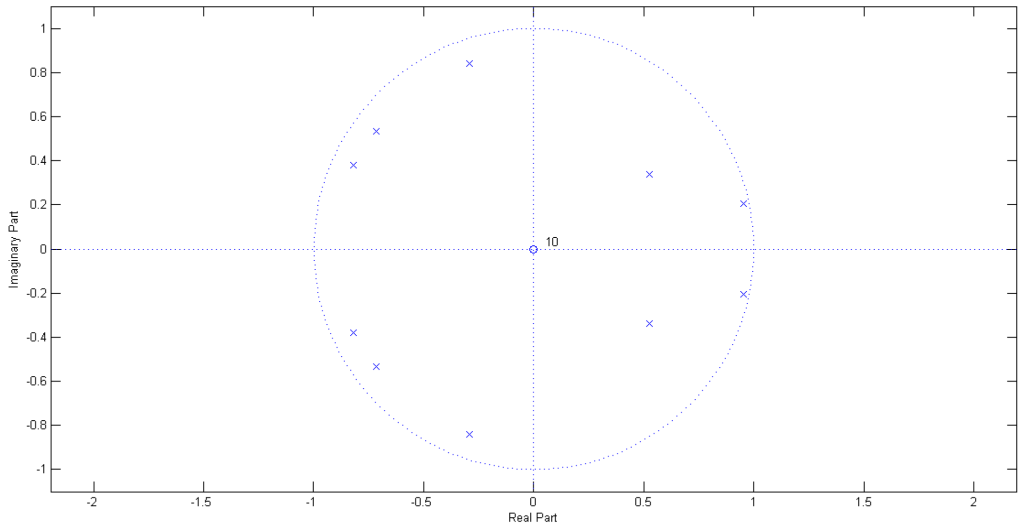


\i\

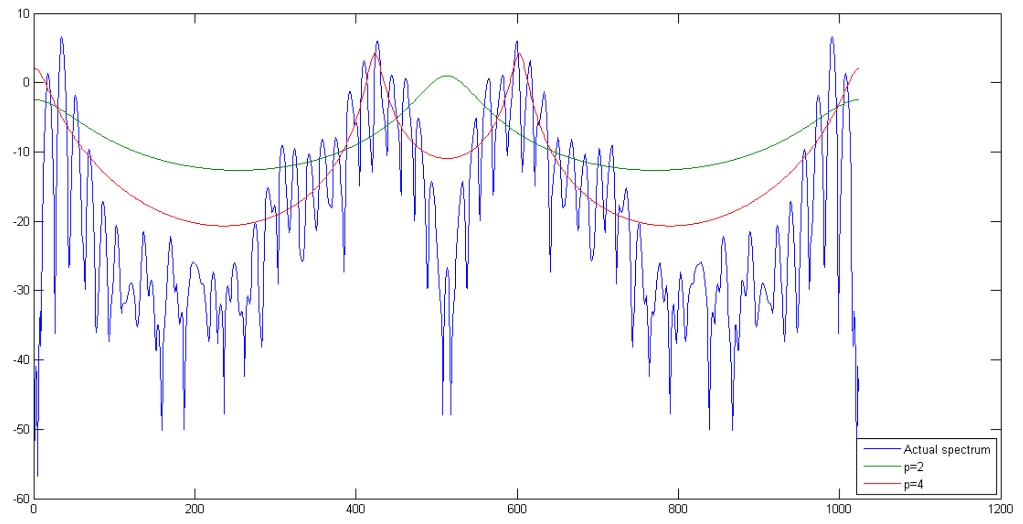
Pole Zero plot | p=6

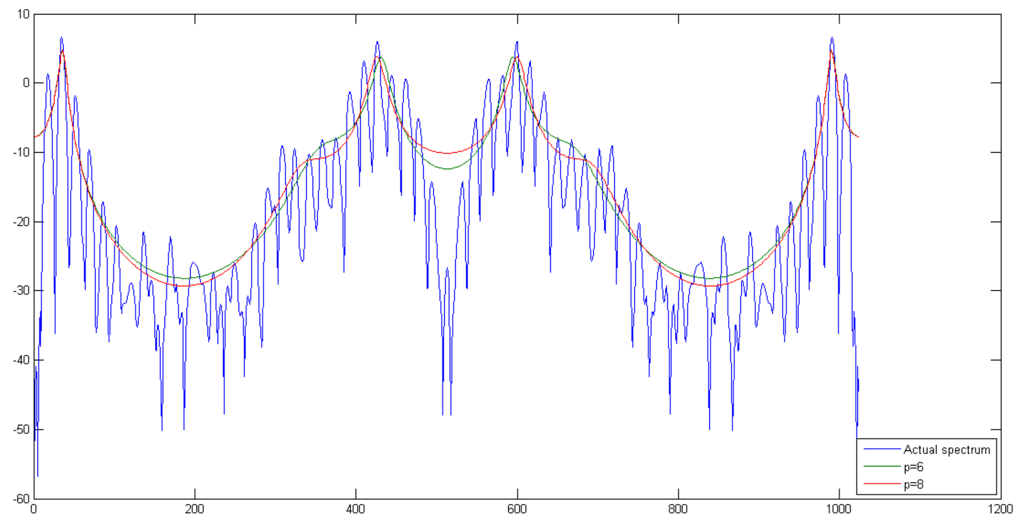


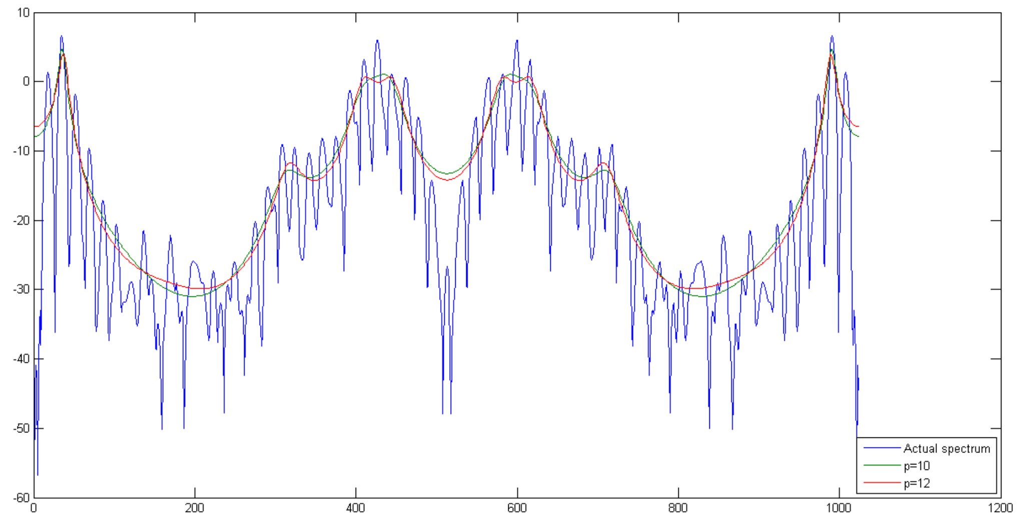
Pole Zero plot | p=10

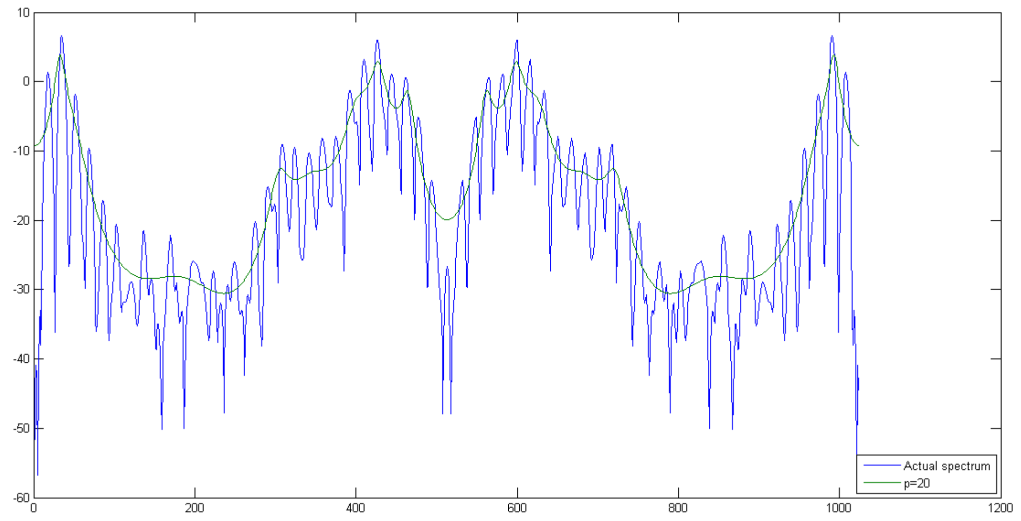


Magnitude Spectrum Plots for various LPC models of order p

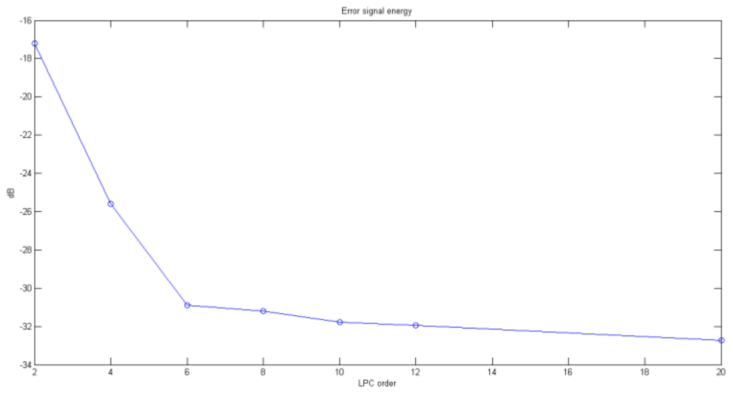






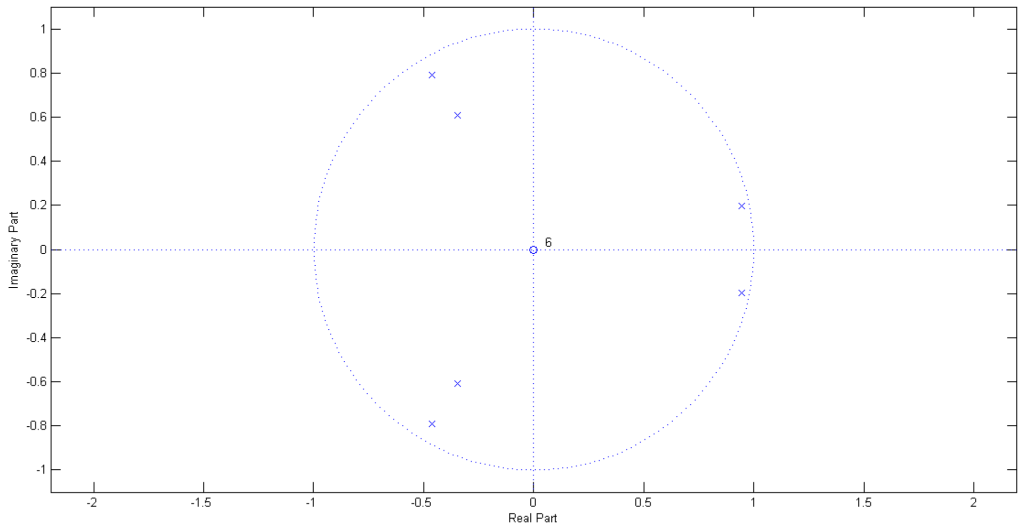


Error Signal Energy vs p

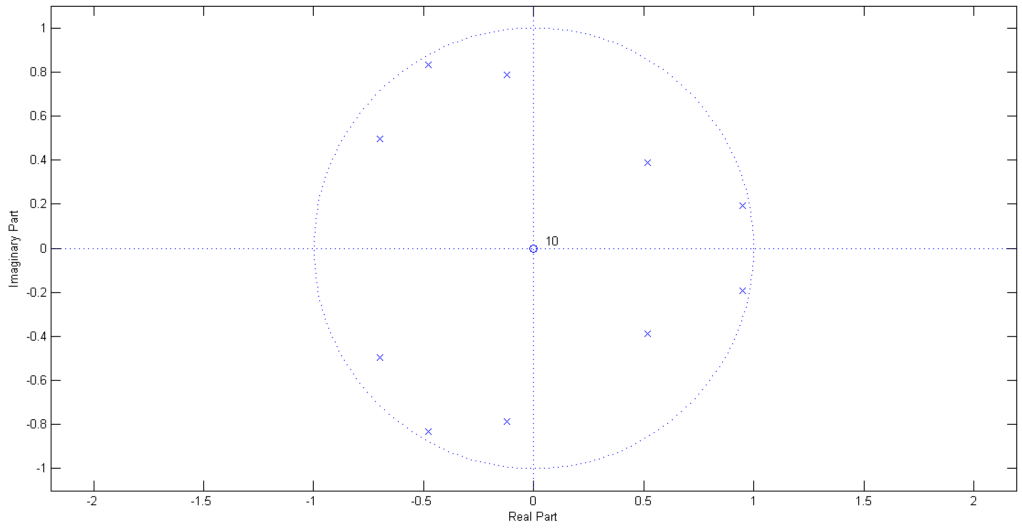


\n\

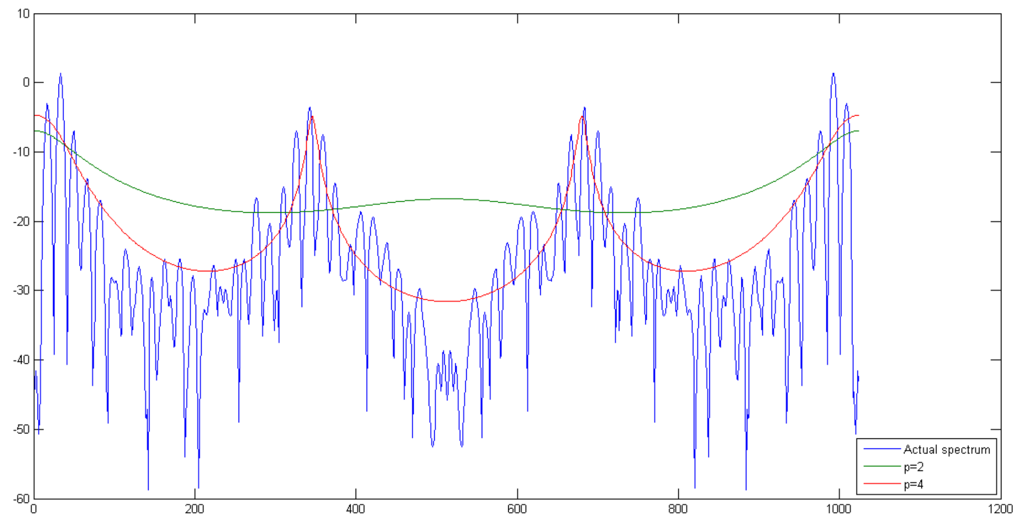
Pole Zero plots | p=6

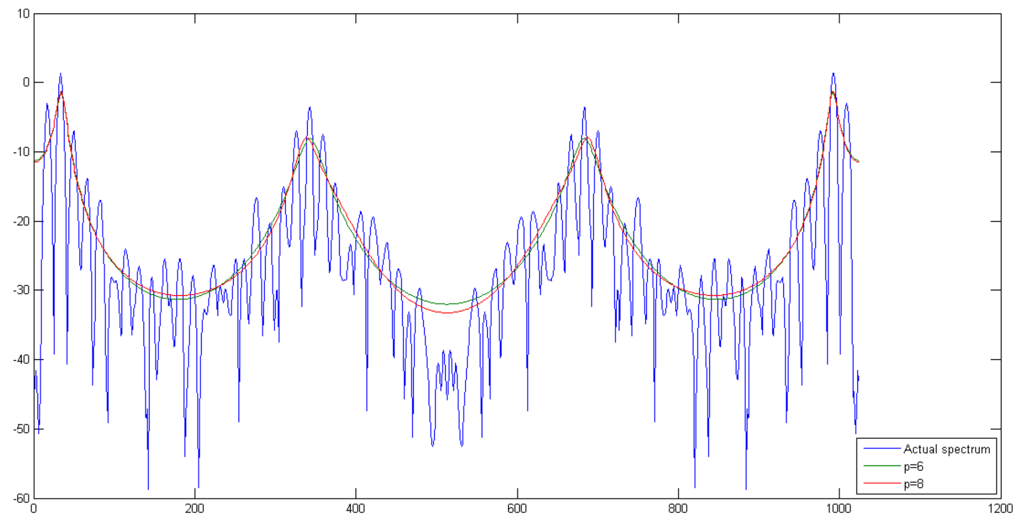


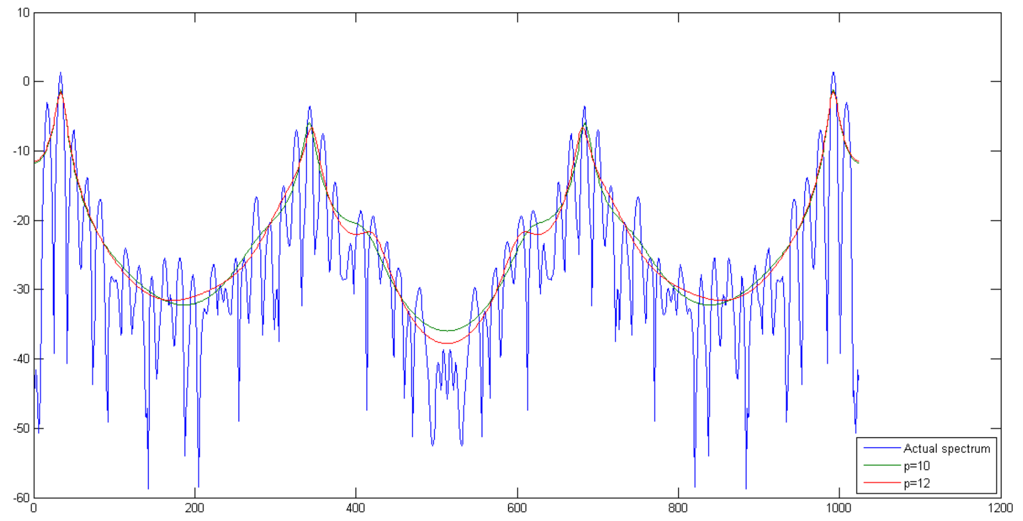
Pole Zero plots | p=10

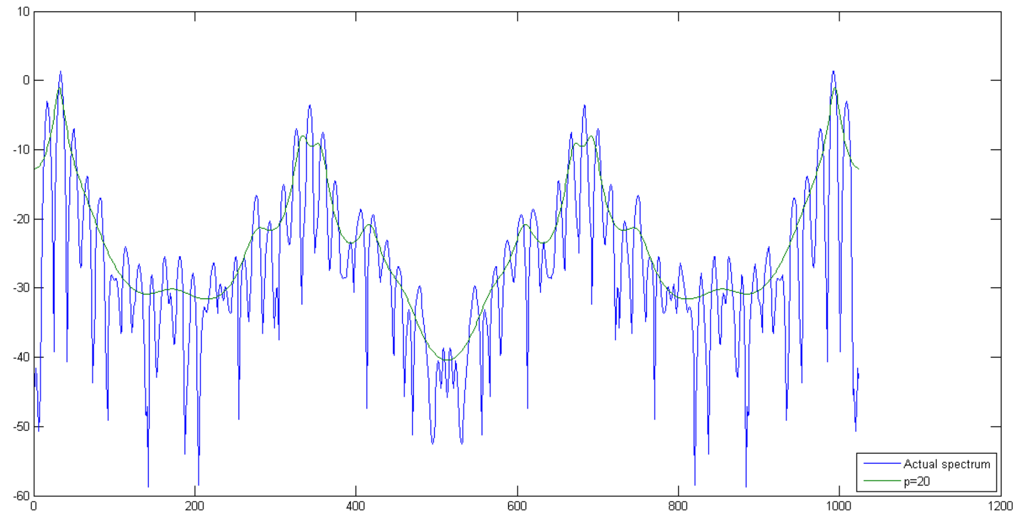


Magnitude Spectrum Plots for various LPC models of order p

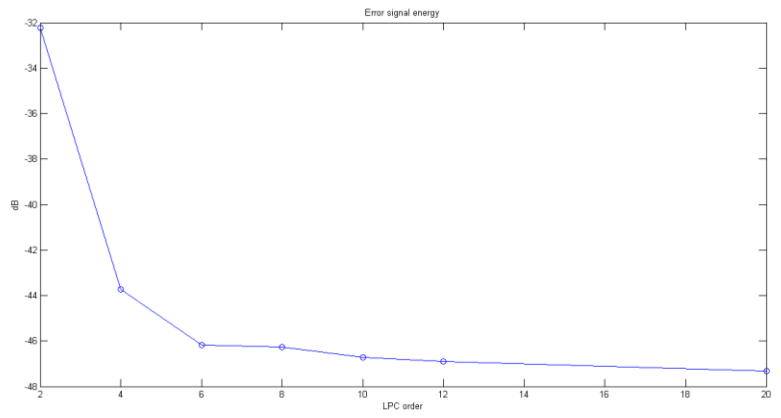






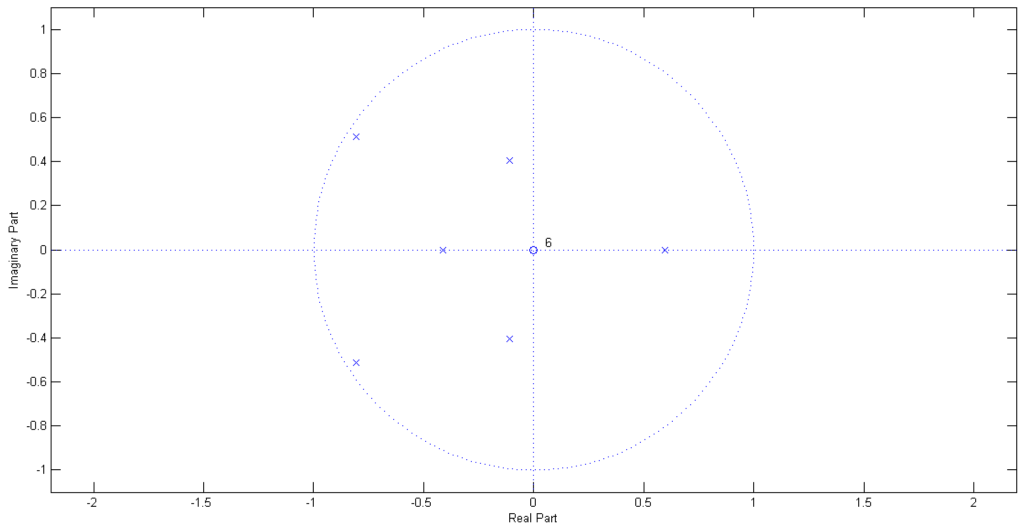


Error Signal energy vs p

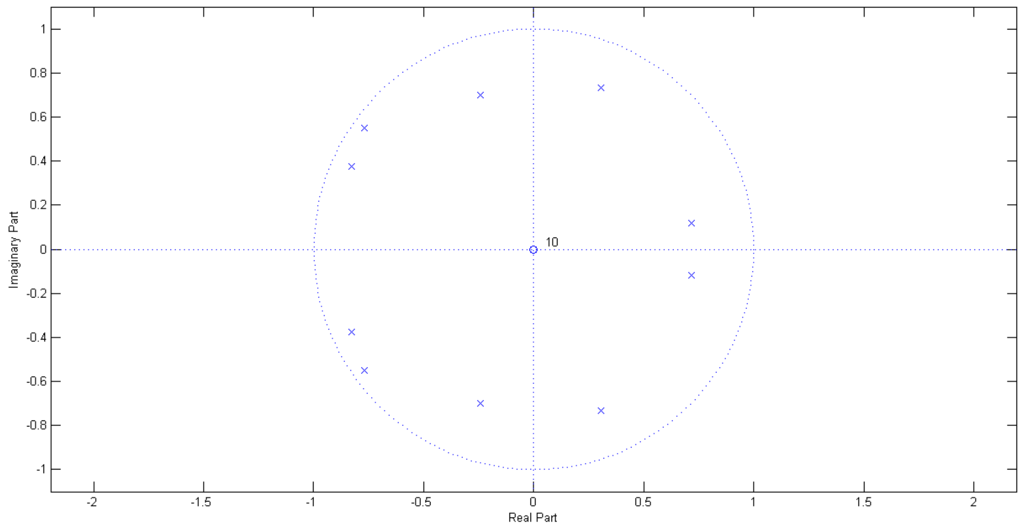


\s\

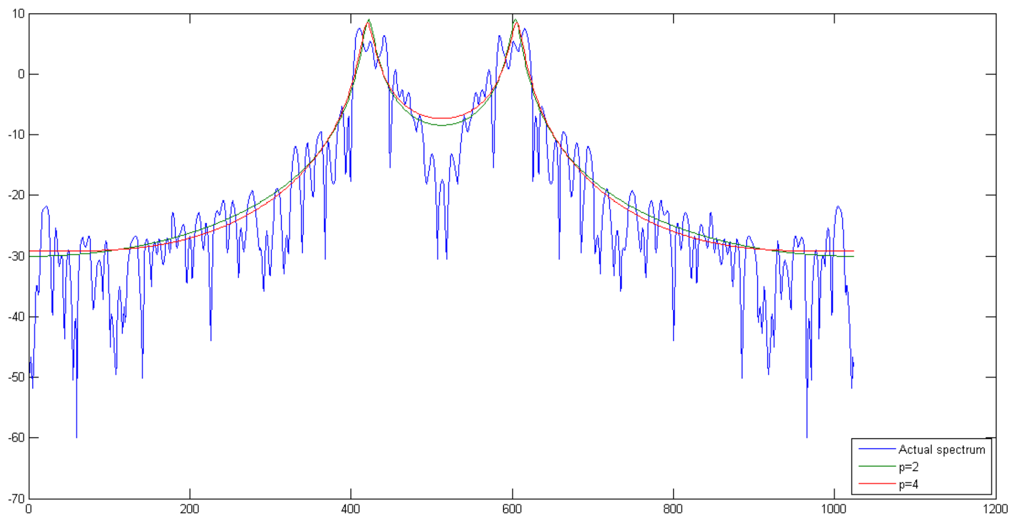
Pole Zero plot | p=6

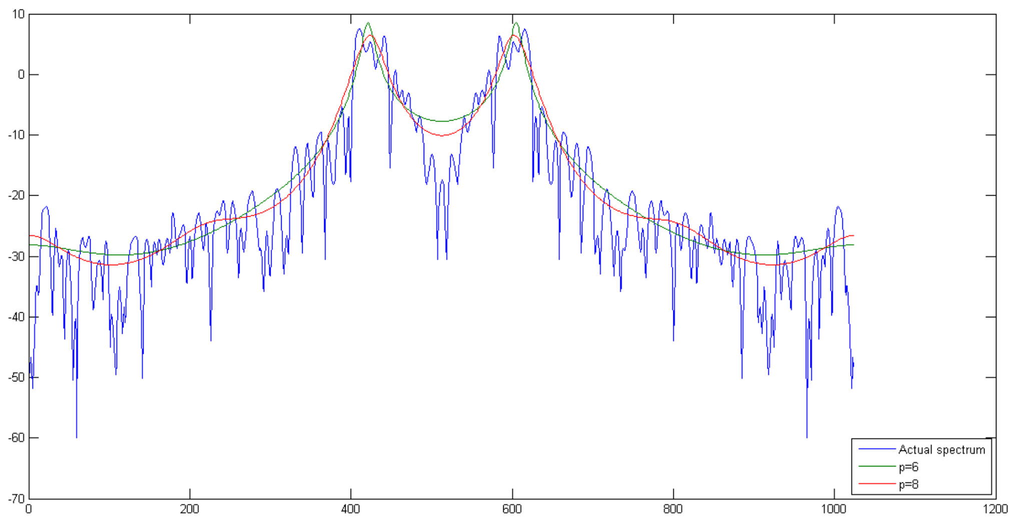


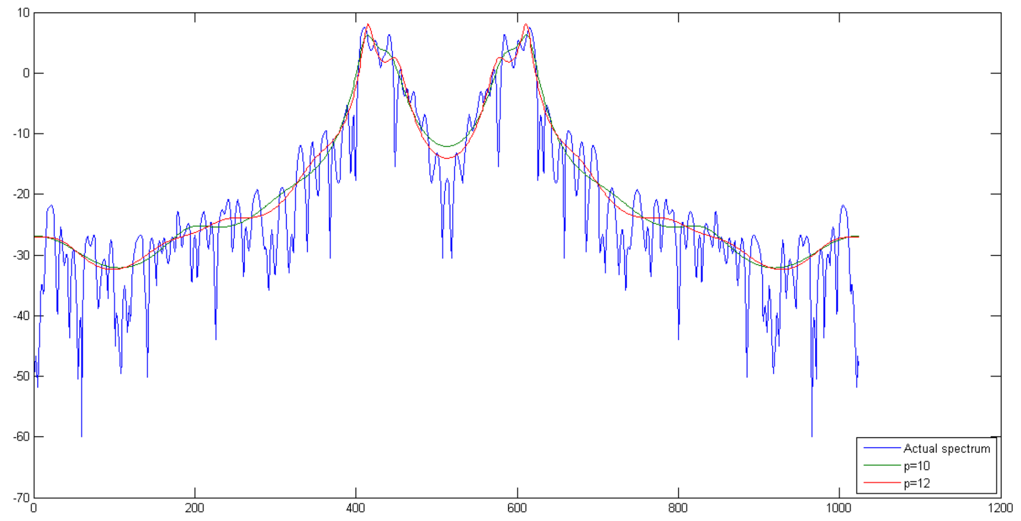
Pole Zero plot | p=10

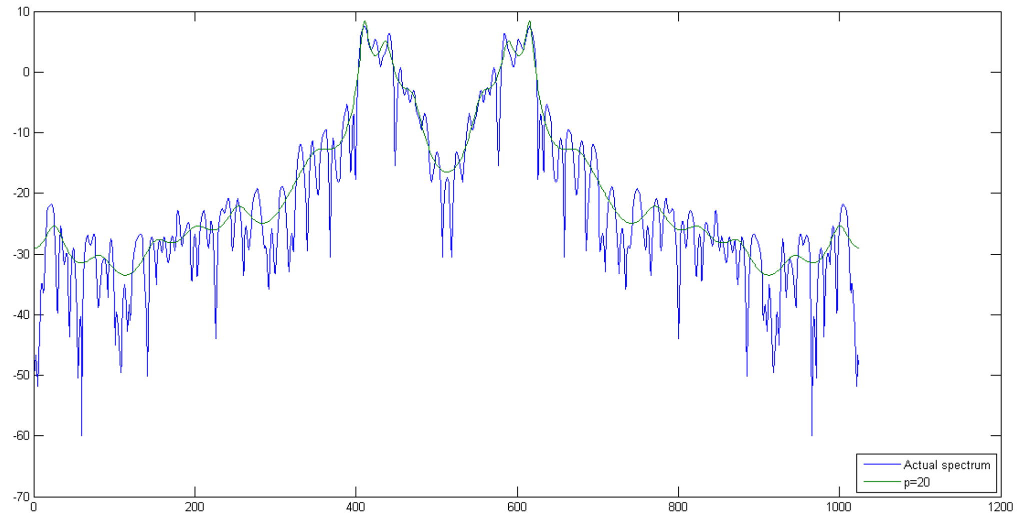


Magnitude Spectrum Plots for various LPC models of order p

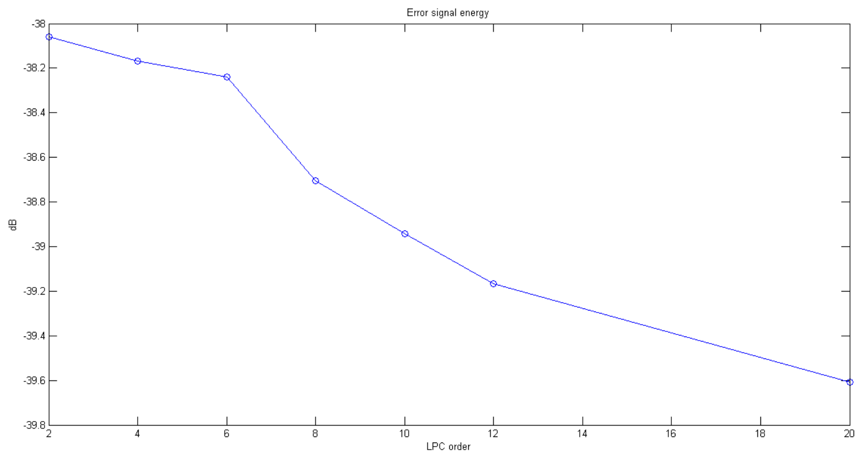








Error signal energy vs p



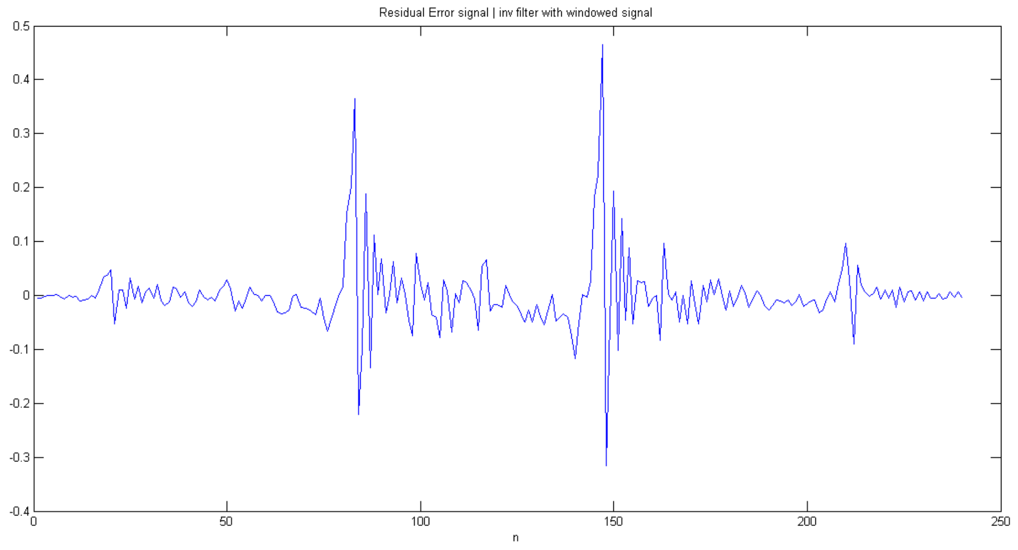
Q3

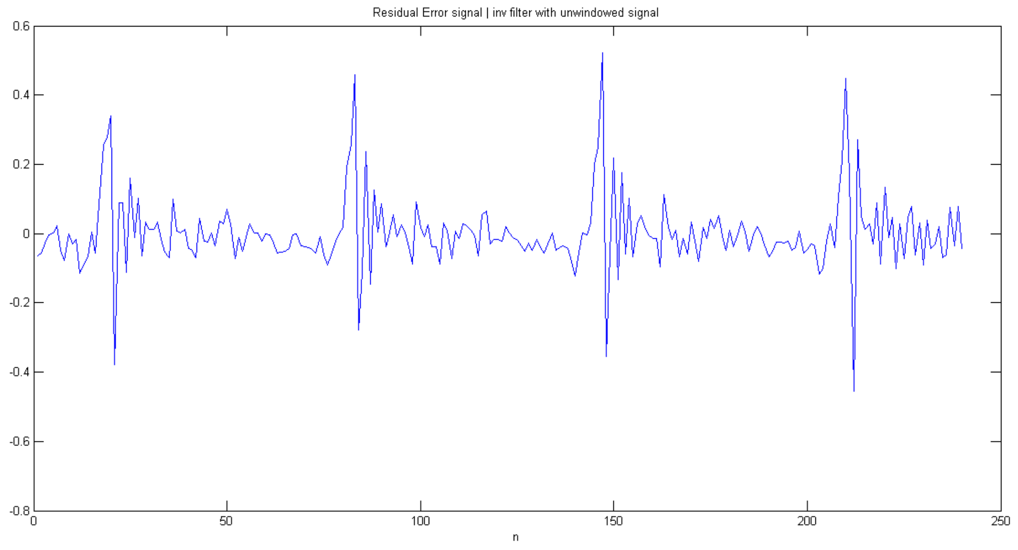
Residual Error signal e[n]

$$y[n] + \sum(a_k * y[n-k]) = e[n] * \sigma$$

The residual error signal can be solved by using the above difference equation for different values of n.

\a\





Residual error from the windowed signal is tapered at the edges because of the hamming window. But the residual error from the un-windowed signal is not. So, we would get a more reliable estimate of the pitch from the second graph.

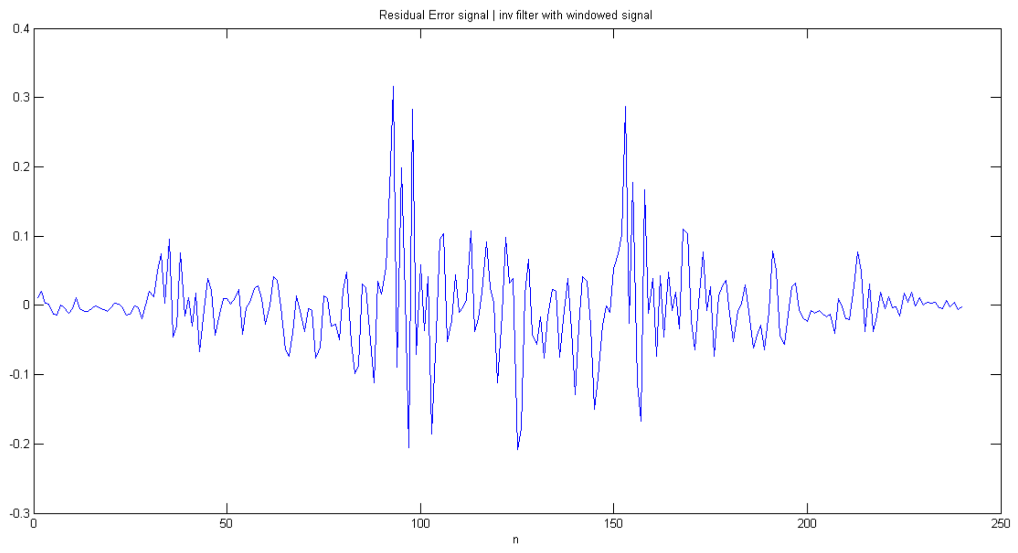
The peaks are at n = 20, 83, 147, 210

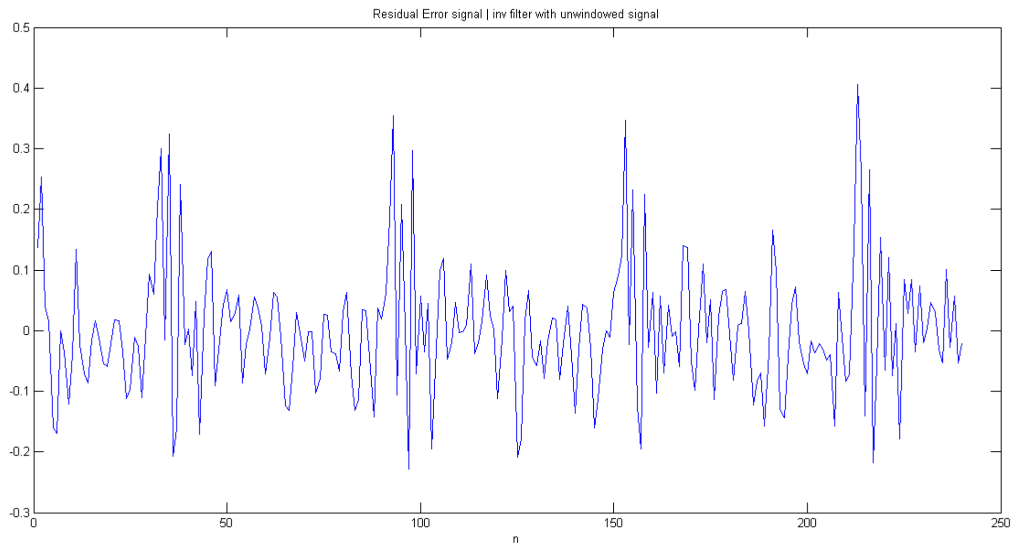
The difference between each peak is about 64, which corresponds to 8000/64 Hz = **125 Hz pitch**

The gain is simply the numerator of the LPC model. We obtained the error variance from Levinson-Durbin algorithm. Gain is the square root of the error variance. **Gain = 0.407359140844486**

LPC coefficients have already been calculated in the previous part.

\i\





Residual error from the windowed signal is tapered at the edges because of the hamming window. But the residual error from the un-windowed signal is not. So, we would get a more reliable estimate of the pitch from the second graph.

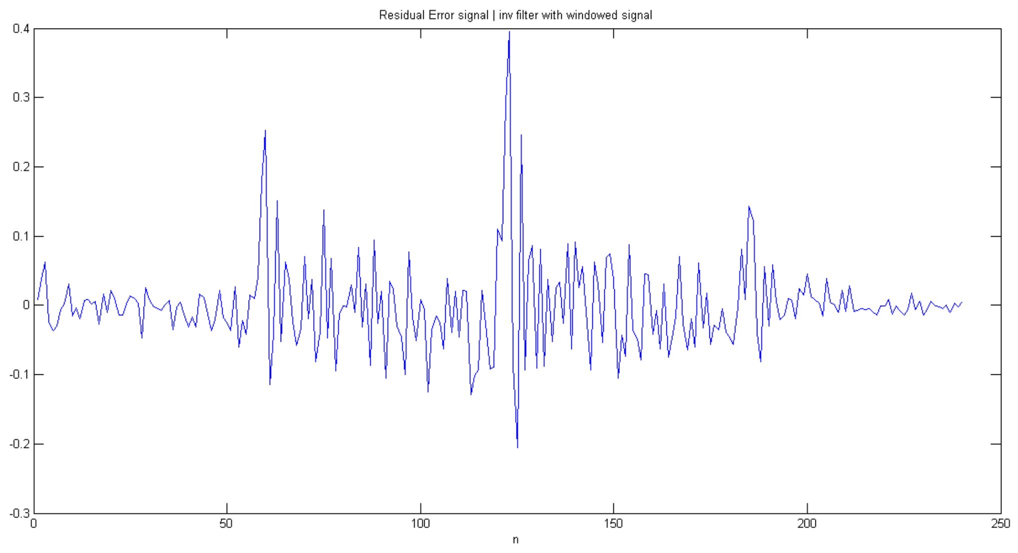
The peaks are at n = 35, 93, 153, 213

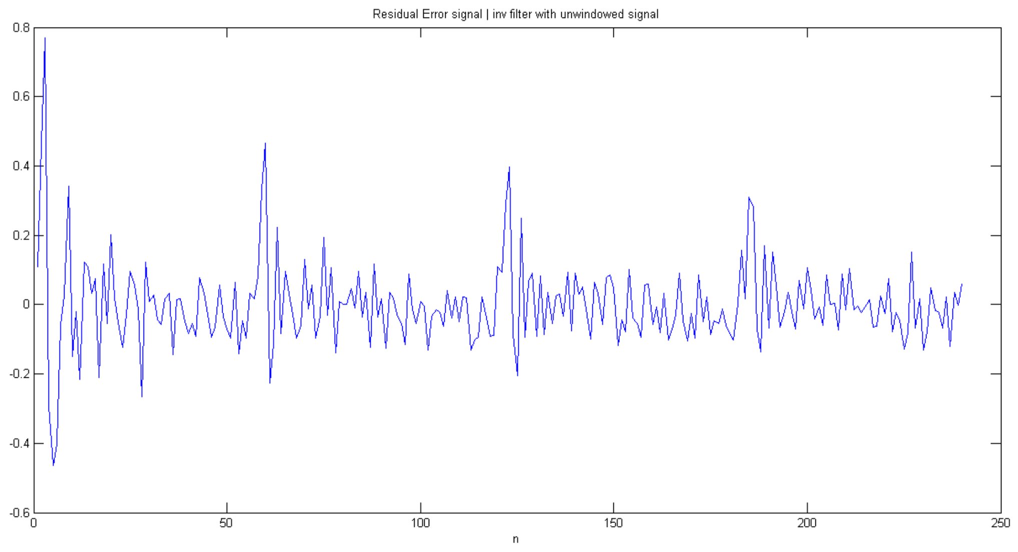
The difference between each peak is about 60, which corresponds to 8000/60 Hz = **133.33 Hz pitch**

The gain is simply the numerator of the LPC model. We obtained the error variance from Levinson-Durbin algorithm. Gain is the square root of the error variance. **Gain = 0.160576763399444**

LPC coefficients have already been calculated in the previous part.

\n\





Residual error from the windowed signal is tapered at the edges because of the hamming window. But the residual error from the un-windowed signal is not. So, we would get a more reliable estimate of the pitch from the *first* graph, in this case.

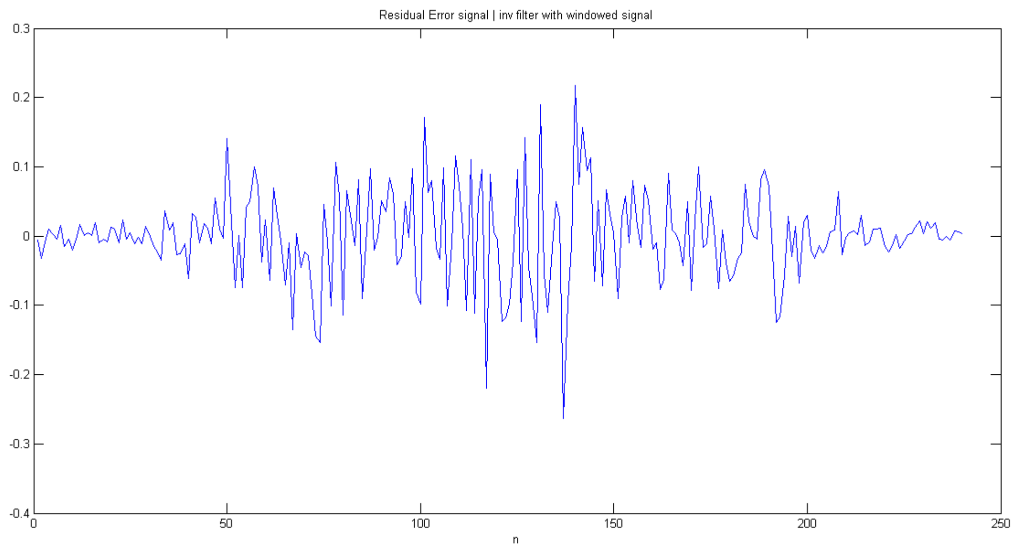
The peaks are at n = 60, 122

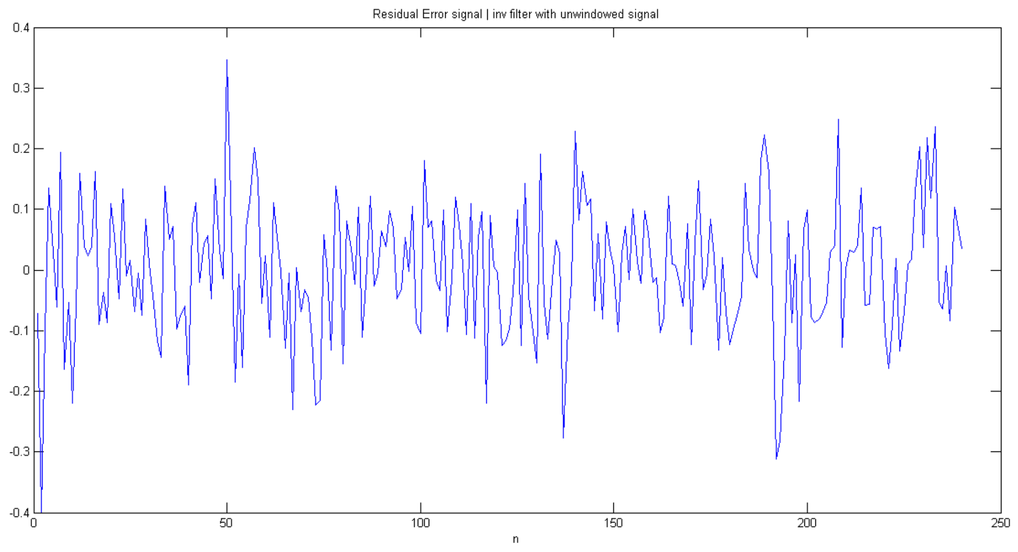
The difference between each peak is about 62, which corresponds to 8000/62 Hz = **129.03 Hz pitch**

The gain is simply the numerator of the LPC model. We obtained the error variance from Levinson-Durbin algorithm. Gain is the square root of the error variance. **Gain = 0.067988990815617**

LPC coefficients have already been calculated in the previous part.

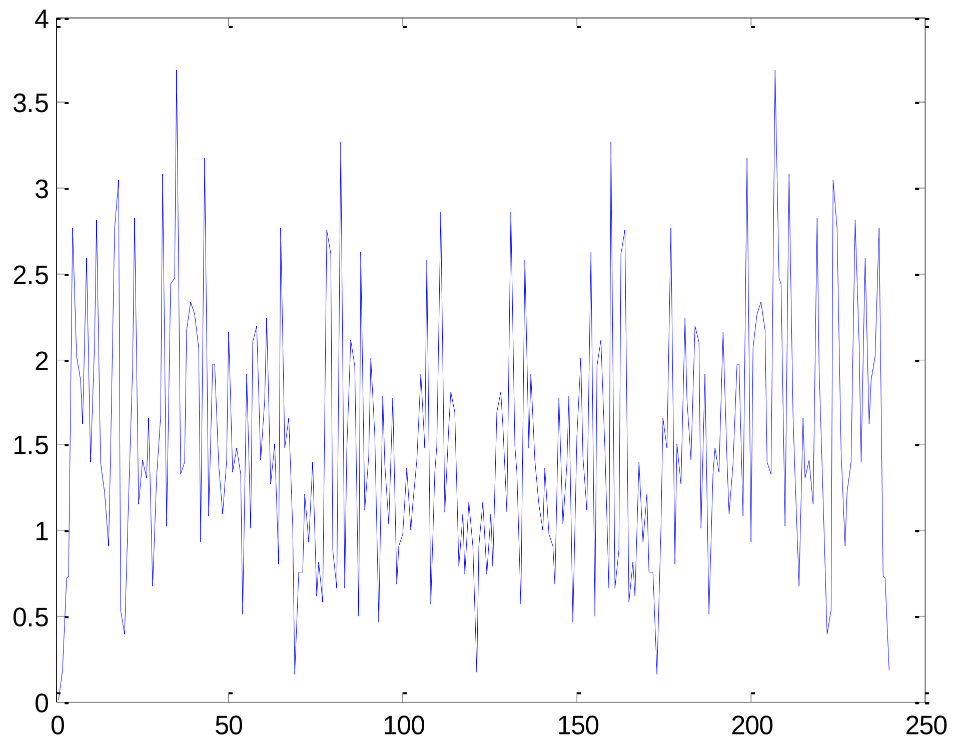
\s\





In the case of \s\, the signal is very 'noisy' in both cases, so it is quite difficult to discern any information about the pitch from the waveforms.
The gain is simply the numerator of the LPC model. We obtained the error variance from Levinson-Durbin algorithm. Gain is the square root of the error variance. **Gain = 0.106275777030210**
LPC coefficients have already been calculated in the previous part.

If we take the FFT of the un-windowed waveform, we get



From this we could crudely estimate the pitch to be **276 Hz**, though I am not sure about the correctness of this estimate.

Q4

Using the 10th order LP model that we have calculated before, and the pitch estimates from the residual error signal, we will attempt to re-synthesize an approximation of the original sound.

```
% Generating the impulse train
T0 = round(1/F0);
t = 0:1/Fs:1;
imptrain = zeros(size(t));
imptrain(1:Fs/F0:end) = 1;
imptrain = imptrain*sigma;
% Generating the white noise signal
wgnoise = wgn(8000,1,sigma.^2);

S = zeros(8000,1);
for i=11:8000,
    S(i) = imptrain(i)- S(i-1:-1:i-10)'*a(2:end)';
end
```

The reason for this particular range of iteration is for implementation convenience. The values S(1:10) are zero. Our actual S is S(11:8000). The above can be considered as a shift of S by 10 units in the +ve direction. This is necessary since, MATLAB does not support negative indices.

We apply the difference equation to obtain the output signal from the LP filter.

$$y[n] = e[n] - \sum a[k] * y[n-k],$$

e[n] = impulse train for voiced sounds(\a\, \i\, \n\) OR white noise signal for unvoiced sounds(\s\)

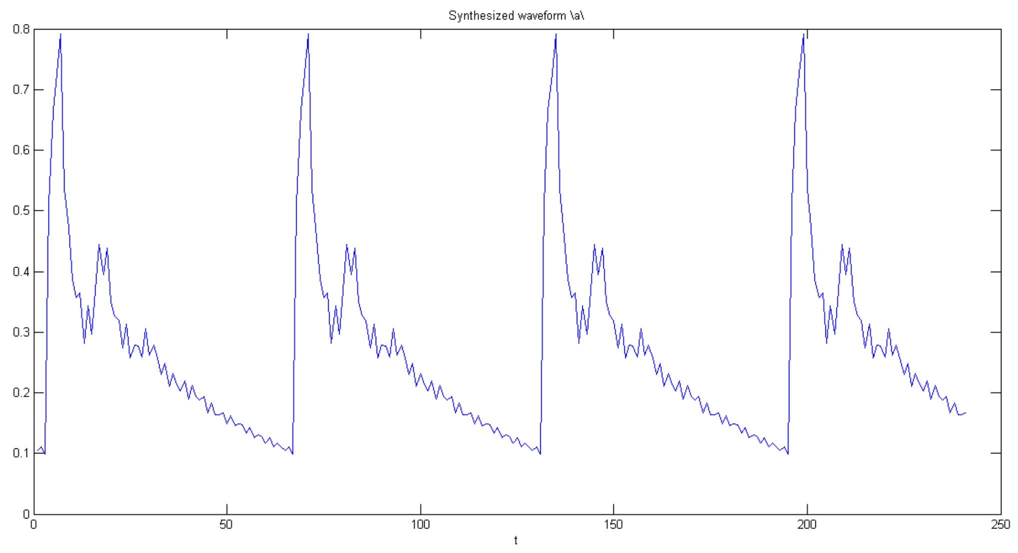
```
Sd = zeros(8000,1);
Sd(1) = S(i);
for i=11:8000,
    Sd(i) = 0.975*Sd(i-1) + S(i);
end

Ssyn = Sd(11:2410);
```

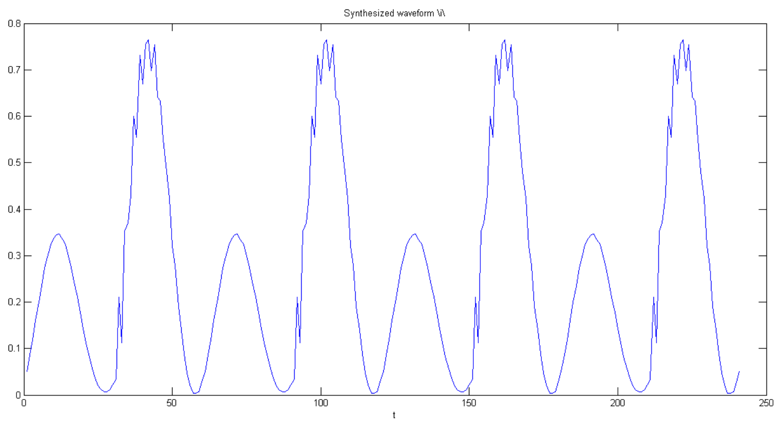
In the above code segment, we have performed de-emphasis. De-emphasis transfer function is simply the reciprocal of the pre-emphasis transfer function. So, the corresponding difference equation is

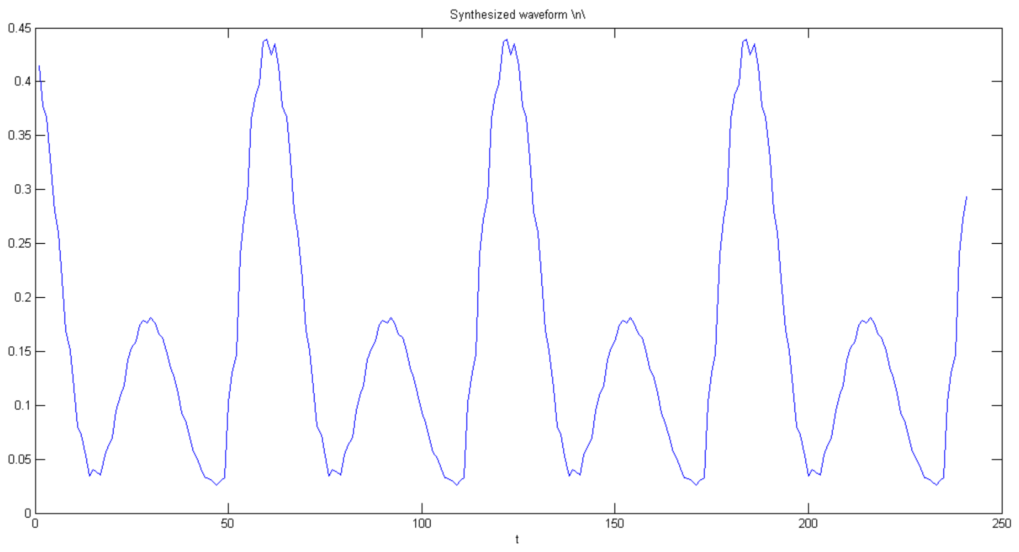
$$y_d[n] = y_d[n-1] * a + y[n], a = 0.975$$

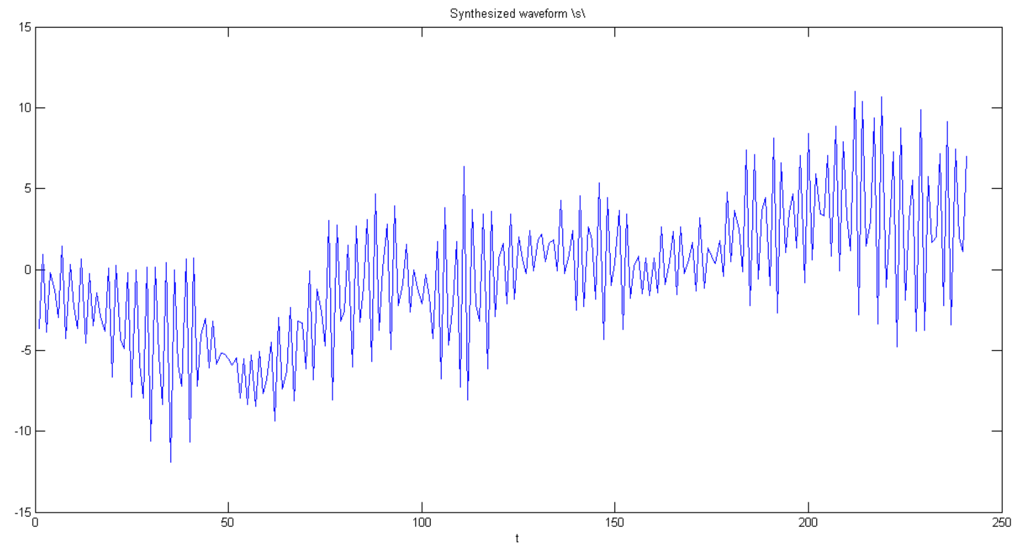
After De-emphasis, 300ms of output is stored in Ssyn

Waveforms of the synthesized sounds:









The synthesized sounds are easily identifiable with the corresponding phone. Though, since we are using a p=10 model, only some of the peaks in the spectrum can be modeled. Hence the sound of all the voiced phones (\a\, \i\, \n\) does not sound extremely natural. \s\ is synthesized using white noise signal, and it does sound quite accurate.