Implementing Linear Predictive Coding:

2.1: Set Up

Q1:

In a 20 ms block, there are 8000 samples/sec * 0.02 sec = 160 samples

Voiced file contains 5405 samples and therefore there are 5405/160 = 33 blocks
Unvoiced file contains 7675 samples and therefore there are 7675/160 = 47 blocks

Q2:

There is a value of one in the pulse train in each period, 1/200.
If we sample the pulse train at 8000 sample/sec, there is a value of one at each, 8000*1/200 , 40 samples in the sampled train.

2.2 Implementing the Model

2.2a) Linear Prediction

We can create a transfer function from every difference equation relating the input to an output.

The output $\hat{s}$[n] is related to the $s[n]$ input by the difference equation

$$\hat{s}[n] = \sum_{i=1}^{p} (\alpha_i * s[n-i])$$

In this equation, $\hat{s}$[n] is the output and $s[n]$ is the input. The coefficients of $\hat{s}$[n] are 1 and the coefficient of s[n] are α_i. Therefore,

a = 1 and b = α_i

2.2a) Prediction Error

Similarly, we get the a and b coefficients from the difference equation relating e[n] to s[n].

$$e[n] = s[n] - \sum_{i=1}^{p} (\alpha_i * s[n-i])$$

The input is s[n] and the output is e[n]. Therefore,

a = 1 and b = [1 -alpha]

2.3a) Synthesis

The input e[n] and the output $\bar{s}[n]$ are related by the difference equation

$\bar{s}[n] - \sum_{i=1}^{p} (\alpha_i * \bar{s}[n-i]) = e[n]$

a =[1 -alpha]
b= 1

2.3b) Reconstruction

The input Gx[n] and the output $\bar{s}[n]$ are related by the difference equation

$\bar{s}[n] - \sum_{i=1}^{p} (\alpha_i * \bar{s}[n-i]) = Gx[n]$

Here, a = [1 -alpha] and b = G

2) **Methodology**

The methodology is similar for all cases.

In the first case, to calculate the error, we use the original signal, X1 as the input. In a loop, we process a segment of the signal. The vectors, a and b calculated in the previous problem are used in Matlab's filter function to estimate the prediction error for the filtered block. The initial conditions for each block are updated in each loop and are calculated from the filter function. To create the final filtered signal, we add up all the estimated errors.

Similarly, to create the reconstructed signal, we only change the input to an impulse train and to create the synthesized sound we use the error prediction as the input.

Q3: Impulse Train

```
pulse_hz = 200;
period = 40;

E1 = zeros(1,N_per_block);
E1(1) = 1;
for index = 1:N_per_block-1
   if  mod(index,40) == 0
      E1(index) = 1;
   end
end
```

Q4:

The synthesized sound exactly sounds like the original signal since we used the exact error. On the other hand, the reconstructed signal sounds very unnatural and robotic.

2.3) Extending Implementing the Model

Q1:

The synthesized signal sounds exactly like the original unvoiced. However, the reconstructed signal sounds like pitched-noise and lacks any audible unvoiced sound.

Q2:

```
E2 = wgn(160,1,0);
```

Q3:

By using the noise as the excitation signal, we construct better unvoiced sound than the previous attempt. Similar to the voiced reconstruction, it sounds robotic.

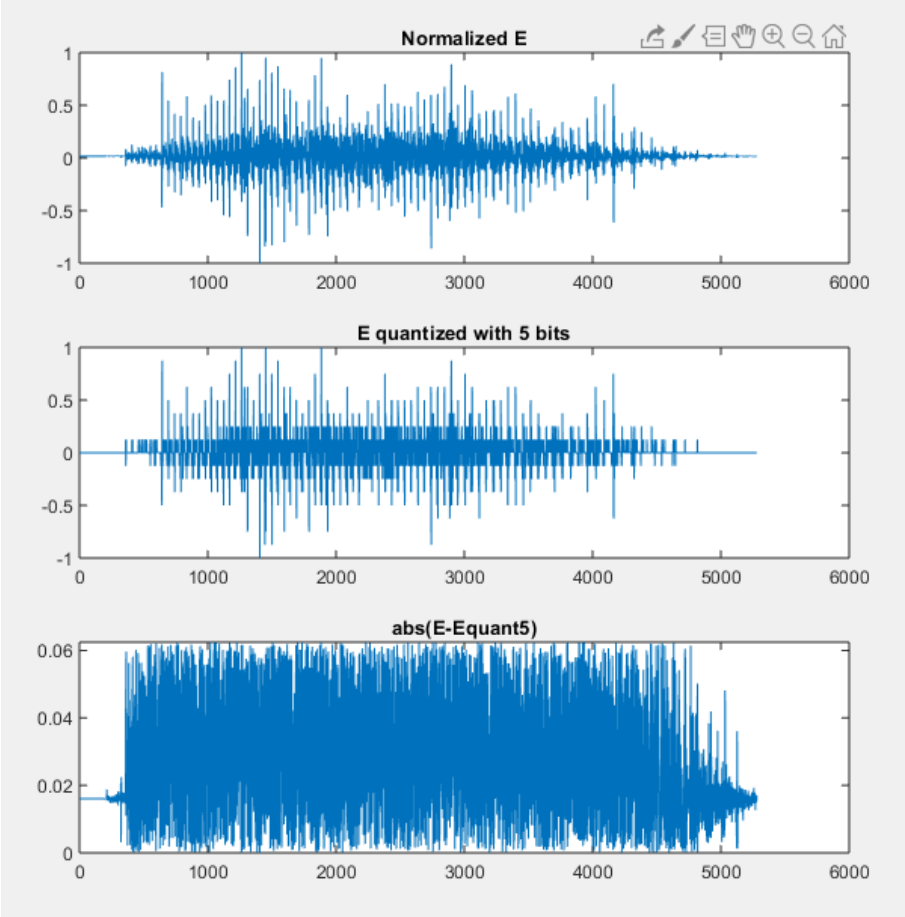
Q4:

To create the voiced sounds, the system would need an impulse train as an excitation signal and noise for unvoiced.

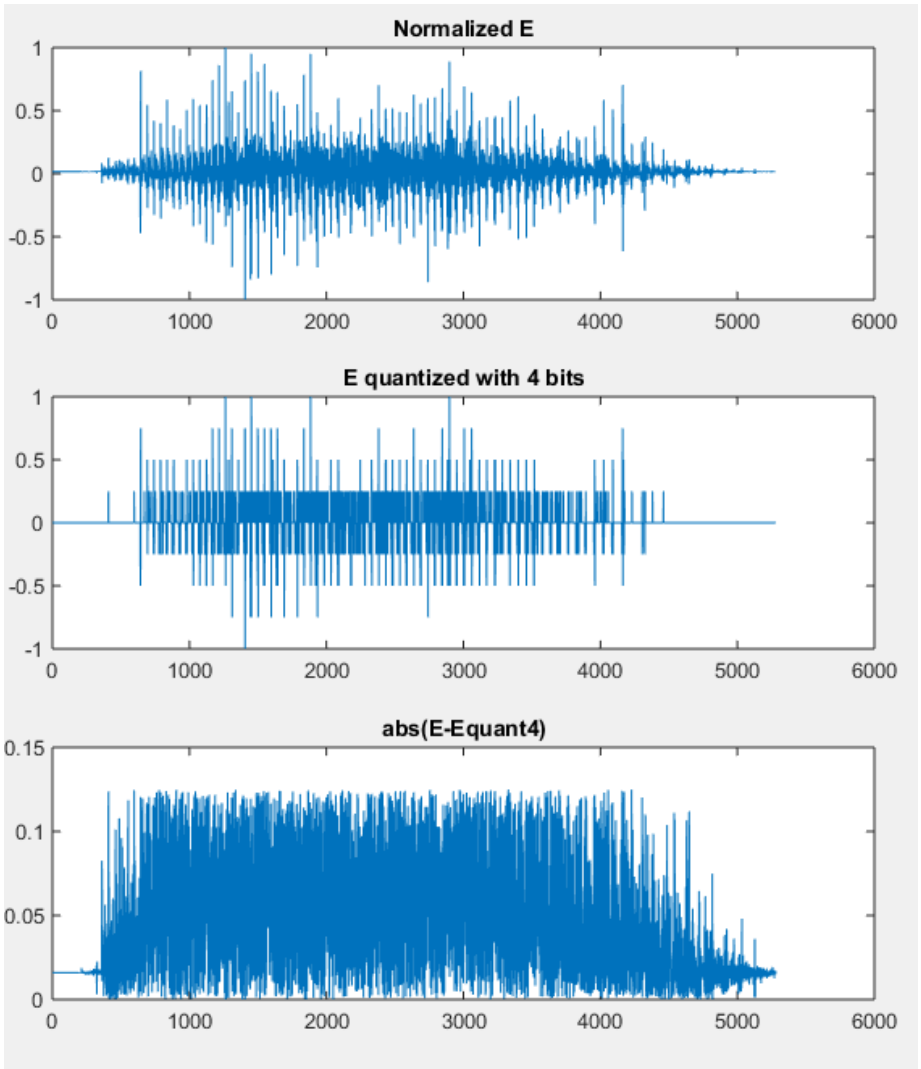
2.3) Compression Using LPC

Q1:

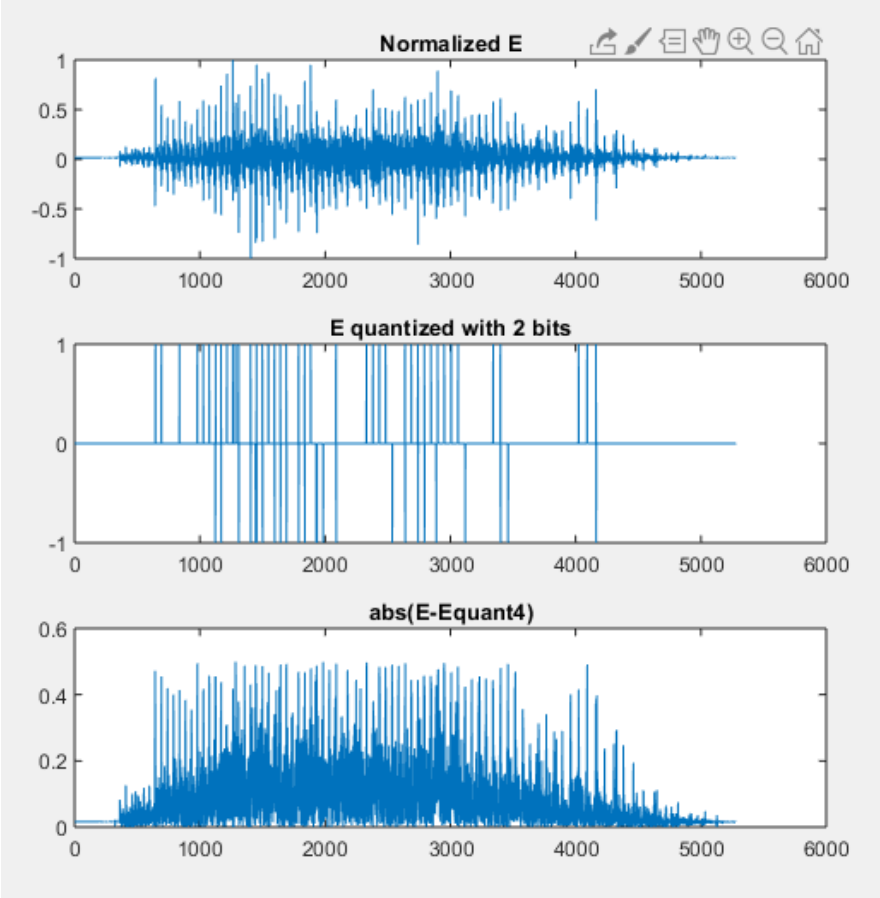
For 5 bits, we get the following plots with a maximum difference of 0.0624.



For 4 bits, the maximum difference is 0.1249.



For 2 bits, the maximum difference is 0.4990.



Q2:

The rescaled quantized error produces a good reconstruction of the original signal. Although this is a simple compression, it works well.