Julian Jarzebowski

Speech Processing

Homework 6

Part one: Define an over lap add (OLA) error function

$$e(t) \equiv \frac{R}{W(0)} \sum_{n=0}^{N-1} w(t - nR) - 1,$$

where w(t) is a window function (i.e., Hamming, hanning or Kaiser window), W (0) = sum(w[n],n) is the sum over w(t), n is an integer running from 0 to N – 1 and R will be the decimation parameter, which is an integer between 1 and the length of the window L. For this problem let L = 128 and N = 1024, and take w to be a Kaiser window having β = 10.5. Let t = mT with T = 1/(2 × 10^4).

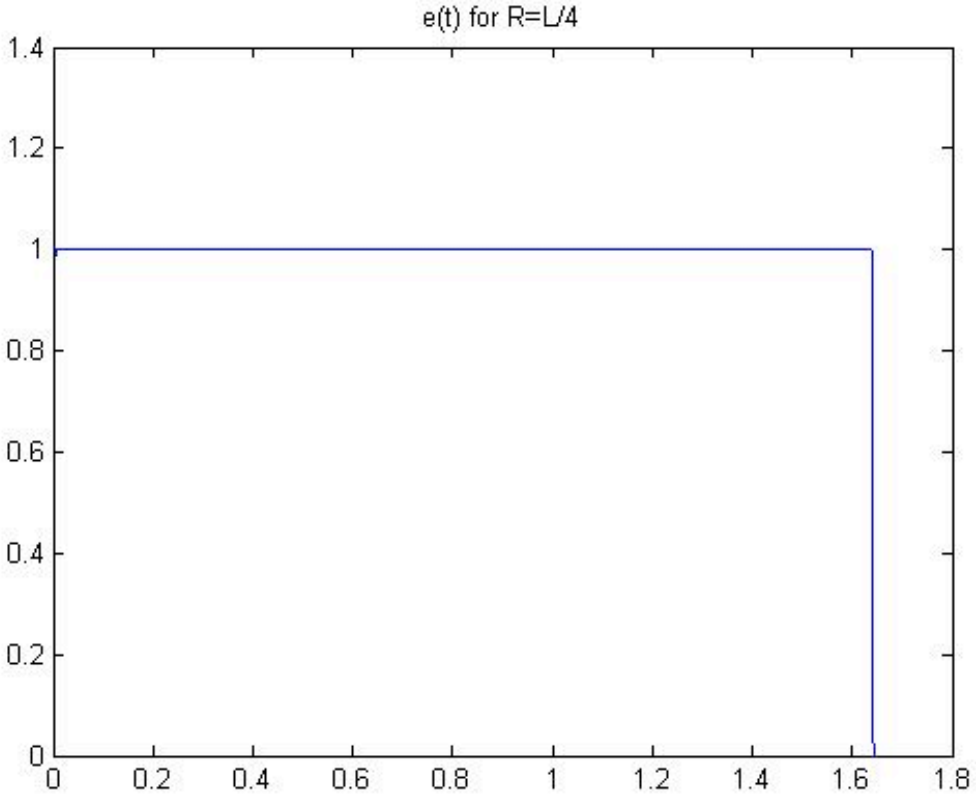
My Matlab code for this part:

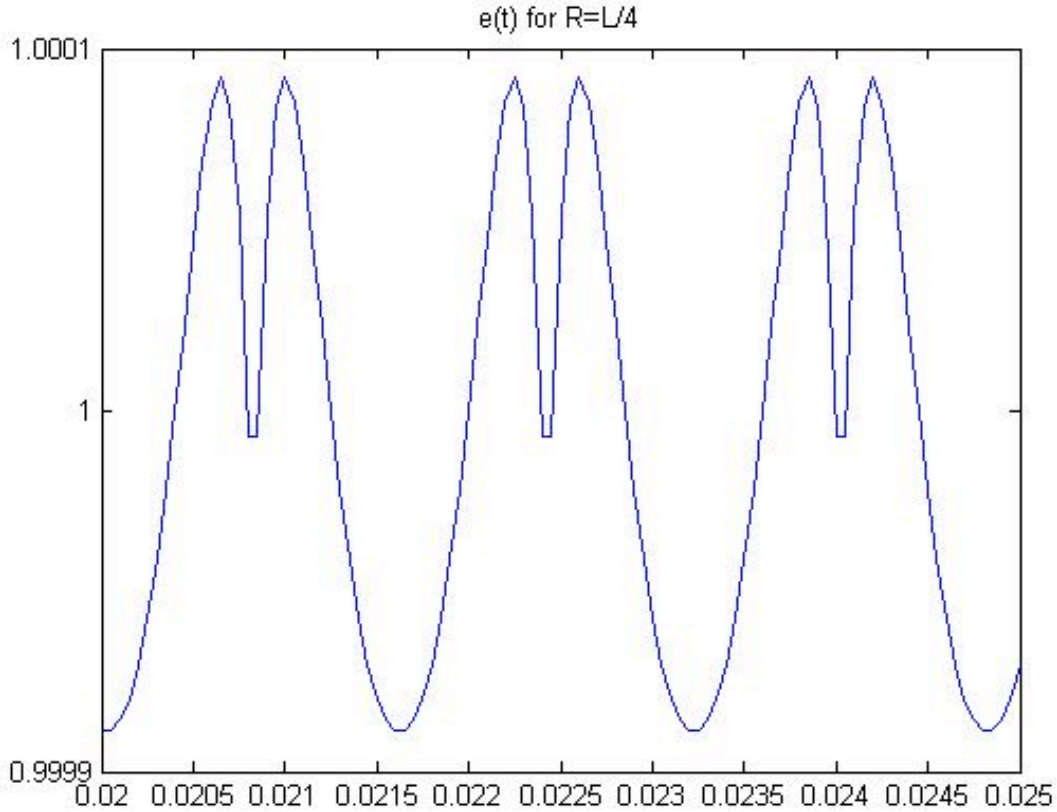
```
%HM 6 part 1
clear all;
T=1/(2*10^4);
L=128;
N=1024;
BTA=10.5;
W = kaiser(L,BTA);
W0=sum(W);
z=[];
t1=1;
CC=2;
R=L/CC;
t2=1023*R+128;



for t=t1:t2
    for n=0:N-1
        if((t-n*R)>0 && (t-n*R)<(L+1))
            z(n+1)=W(t-n*R);
        else
            z(n+1)=0;
        end
    end
    e(t)=(R/W0)*sum(z);
end
```

Julian Jarzebowski

```
plot([t1:t2].*T,e(t1:t2)), title(['e(t) for R=L/' num2str(CC)])
```

1. Plot e(t) for R = L/4.

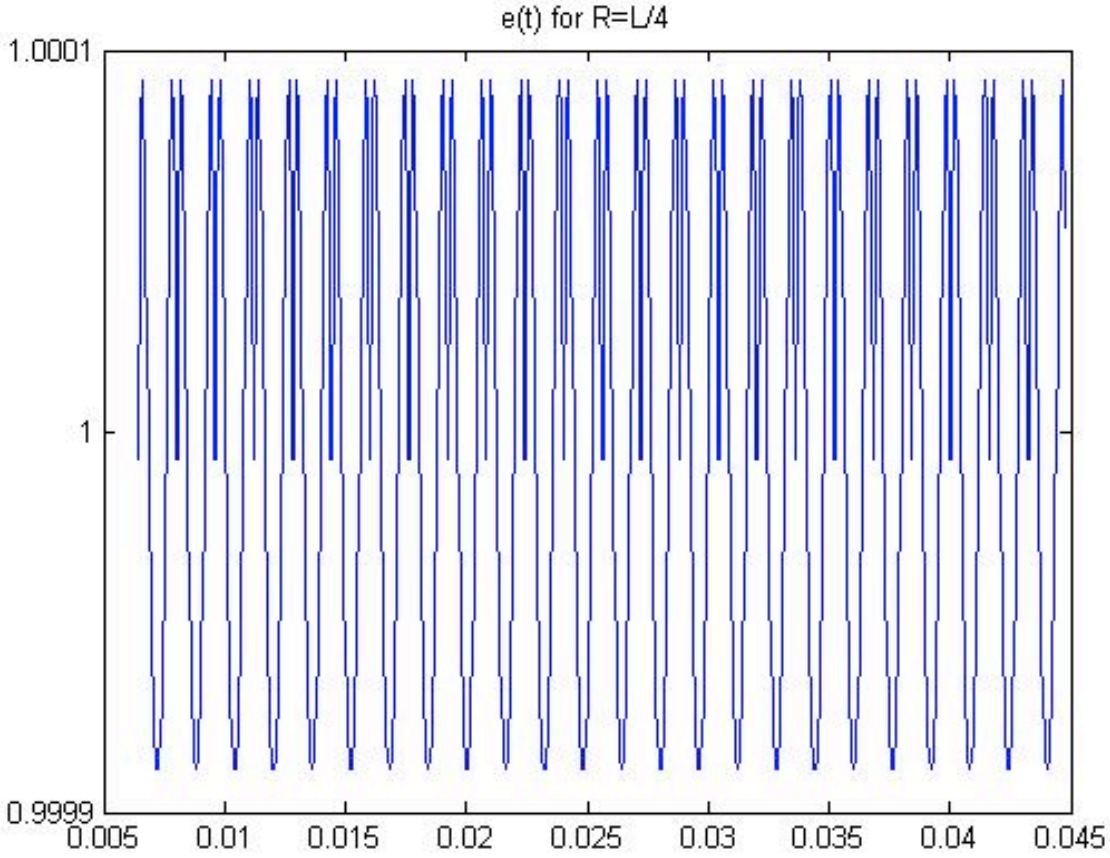




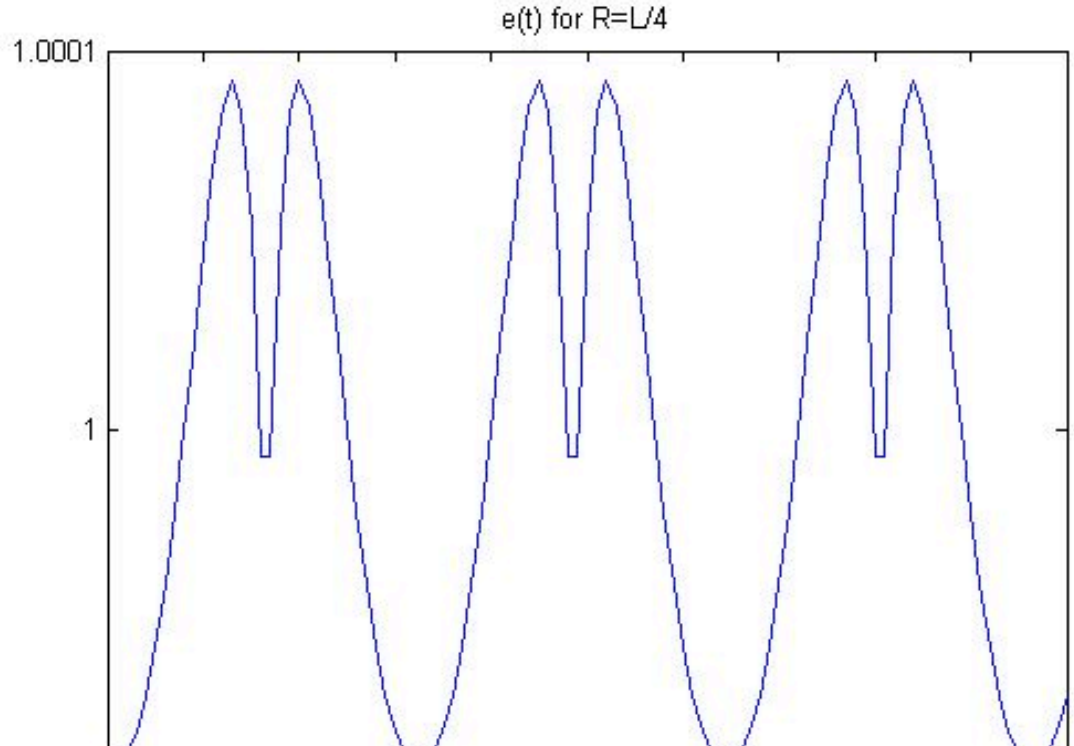
We can see that the signal is periodic with a period of 1.6 ms

Julian Jarzebowski

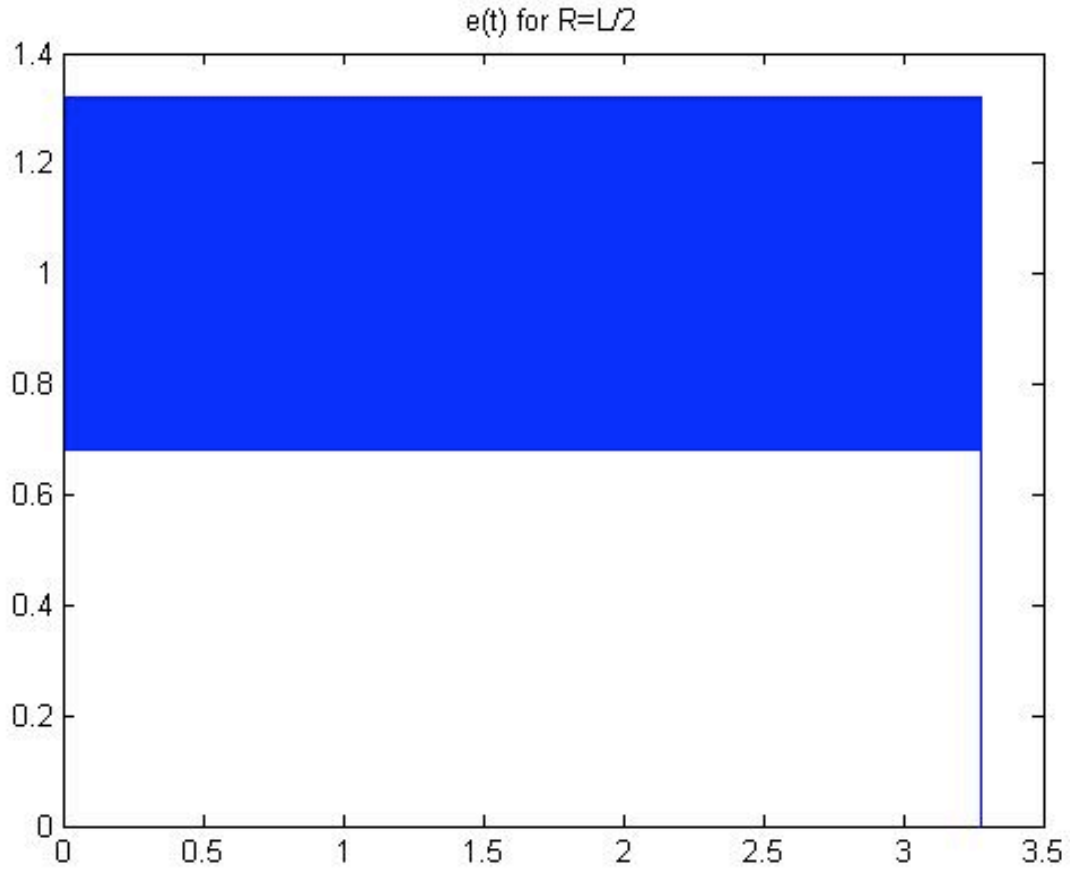
2. Plot e(t) in the range of $L \leq t \leq N - 1 - L$.
Hopefully we obtain the same plot when we zoom on this plot than what we had in 1).



3. Plot e(t) with R = L/2.



We can zoom on this plot to observe that the signal is again periodic with a period of 3.2 ms which is two times the first period.



4. Why is e(t) periodic? What is the period? Why?

As we use the same window which we schift by a constant number of points along the signal, we shall observe a periodicity related to this schifting coefficient.
Thus we observe for R = L/2 = 128/2 = 64 with 64/(2*10^4) = 3.2 ms.
The same results appears for R = L/4 = 128/4 = 32 with 32/(2*10^4) = 1.6 ms.

5. Explain what is going on. What is happening at the ends of e(t) in the first plot?

To obtain this formula:

$$e(t) \equiv \frac{R}{W(0)} \sum_{n=0}^{N-1} w(t - nR) - 1,$$

we apply the window function to the signal in a special way. First we apply it a the begining of the signal, then we schifts it by R samples and we apply it again. We do this until the end of the window reaches the end of the signal.

Then we sum all the windowed frames to obtain the resulting signal.

At the end of the first plot, the resulting final ultimatly falls to zero.

Part two:

The idea of this problem is to make a very simple lossy speech codec. A codec encodes and then decodes speech, reducing the number of bits for the representation. The idea is to reduce the bit rate as much as possible, without reducing the perceptual quality of the speech.

1. First read the wave file into memory using matlab's wavread command `[s,Fs]=wavread('WhenAllElse8k.wav');`
Verify that F_s = 8000 [Hz] and that the speech is read in properly. Plot it, and write it back out in another file, to verify this, for example.

```
clear all;
[s,Fs]=wavread('WhenAllElse8k.wav');
sound('WhenAllElse8k.wav');
wavwrite(s,Fs,'TestWhenAllElse8k.wav');
sound('TestWhenAllElse8k.wav');

C=10
M=15/C;
disp(sprintf('%g bits per speech sample',M+1));
```

2. Next strip off and save the sign from the speech waveform, using the Matlab `b16=sign(s)` command, which saves the sign in array `b16`. This counts as one bit/sample.

```
b16=sign(s);
```

3. Next take the magnitude of the speech and take the square root. By taking the square root, the sign-reduced speech is reduced from a 15 bit resolution to 8 bit resolution. This is because the original data was stored as sign and magnitude 16 bit numbers, and taking the square root reduces the dynamic range by 1/2.

```
S=abs(s);
SM=S.^(1/C); %compress
```

4. We next need to fix the numbers to be integers. This is the quantizing step. Scale the speech up by 7 bits (scale by 128) Use the "fix()" command. Then scale back down by 128. Verify that the resulting numbers are less than 1 after you have finished doing this.

```
sM=(1/2^ceil(M))*fix(SM*(2^ceil(M))); % with 2^ceil(M) = 128
max(sM)
```

5. Now square the result, and put the sign back on. You should now have a linear speech

signal again, but the compressed version should have 1/2 the dynamic range of the signal you started with, plus 1 bit (the sign bit).
We started with 16 bit/sample speech, but you should now have 9 bits/sample.

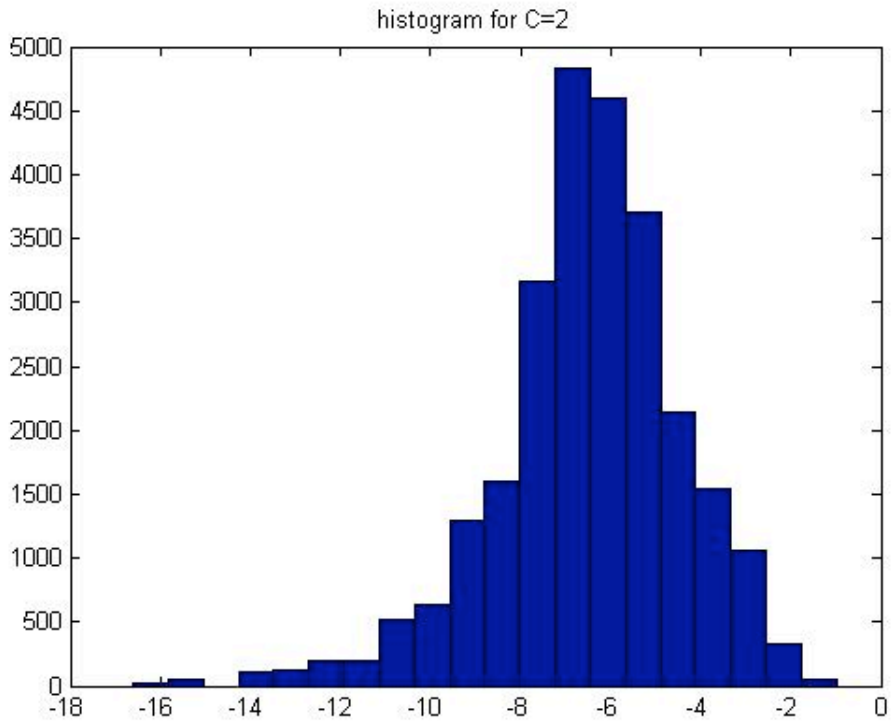
```
sM=b16.*(sM.^C);  % each componant of b16 and s_fixed is repectively
                  % 1 bit and 8 bits encoded. We thus have a 9bits code.
```

6. Make a histogram of `log2(abs(speech))`

```
hist(log2(abs(sM)+0.00001),20),title(['histogram for C=' num2str(C)])
wavwrite(sM,Fs,'gWhenAllElse8k_C7.wav');
sound('gWhenAllElse8k_C7.wav');
```

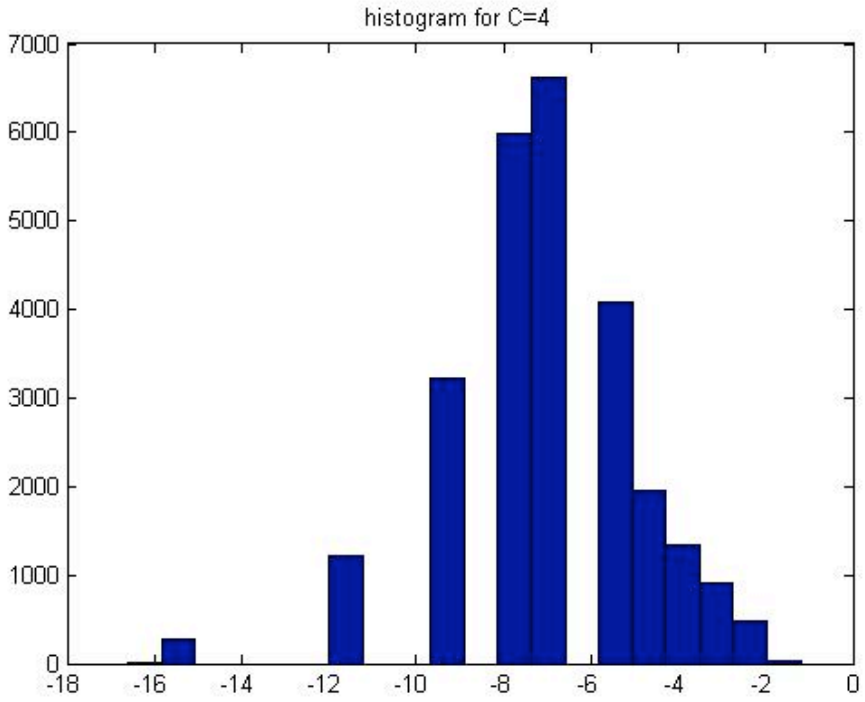
7. Then we change the values of C.

For C=2,

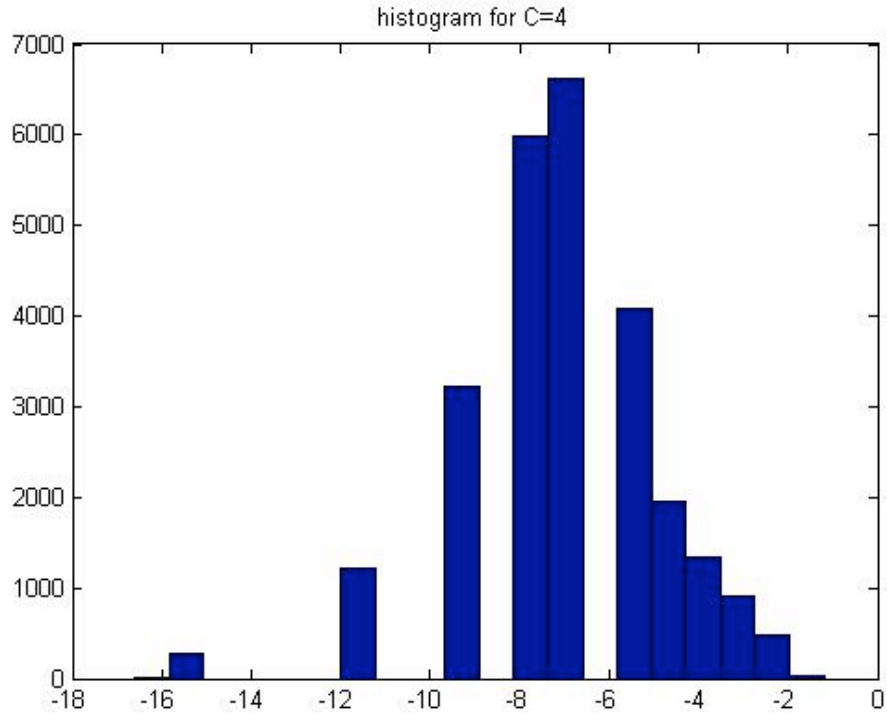


Julian Jarzebowski

For C=3,

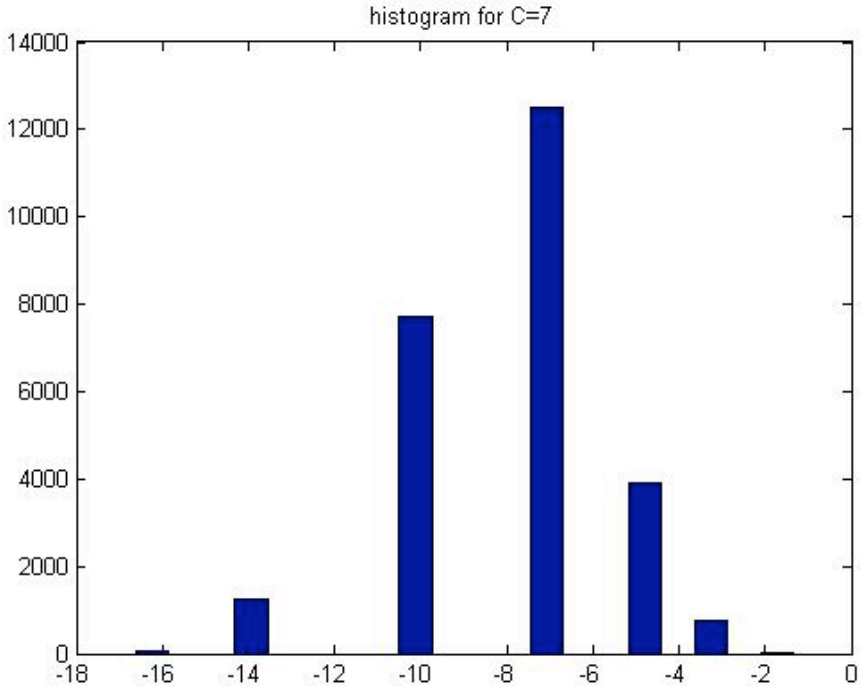


For C=4,

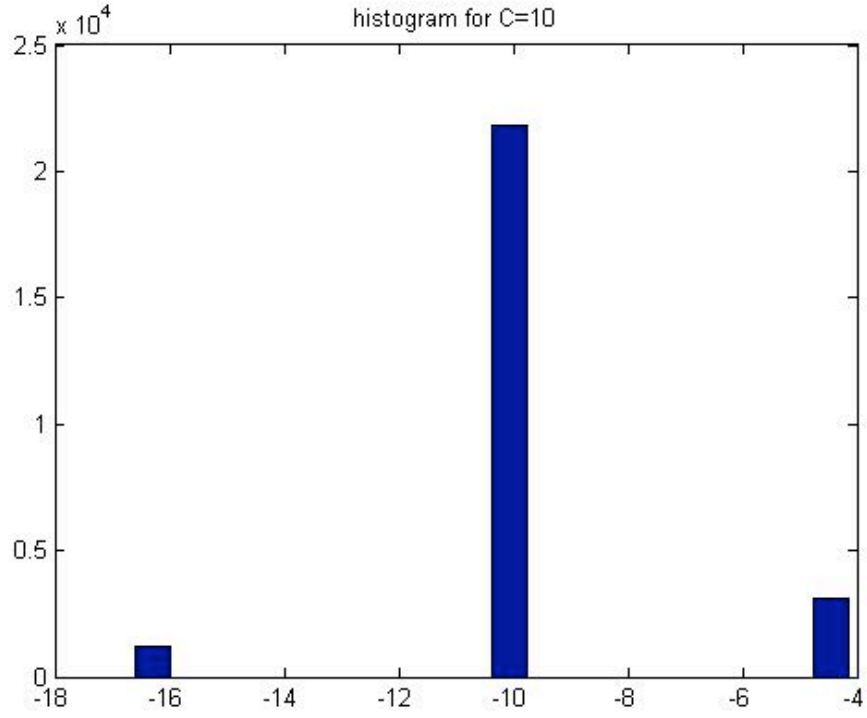


Julian Jarzebowski

For C=7,



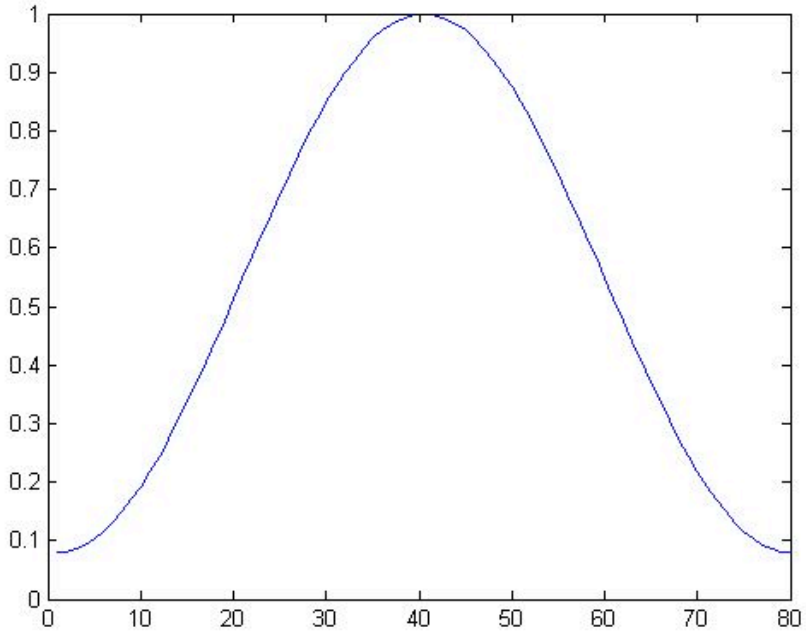
For C=10,



We observe that the signal gets worse and worse. At C=10, it becomes to be really bad, with a lot of noise.

Part 3: Write a program in Matlab that performs overlap-add processing on the speech samples provided

We will need to use a window function such as the Hamming window. We use a window that has a duration of about 10 ms, and overlap the window by 4:1 (i.e., for every output sample, there will be 4 input samples that contribute).



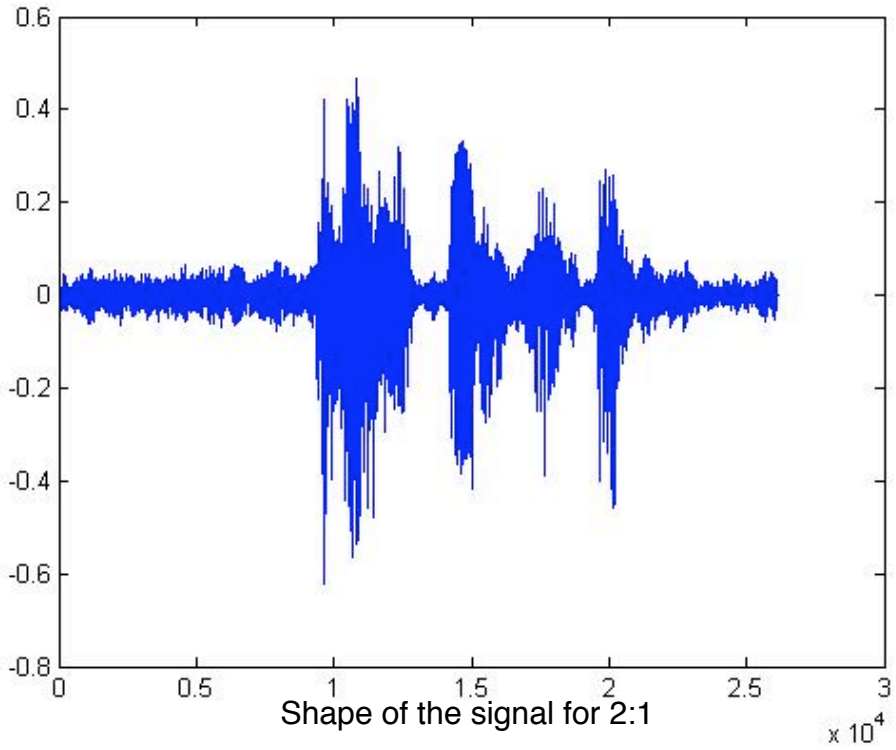
1. We try 4:1, 8:1, 2:1 and 1:1 overlap.

4:1

8:1

Julian Jarzebowski

2:1



Shape of the signal for 2:1

1:1

We used this code to do the previous plots and wav files:

```
clear all;
[s,Fs]=wavread('WhenAllElse8k.wav');
sound('WhenAllElse8k.wav');
L=10*10^(-3)*Fs;
W=hamming(L);

W0=sum(W);
z=[];
t1=1;
t2=length(s);
R=L/4;
N=floor((length(s)-L)/R);
```

```
 ss=zeros(length(s),1);
 for k=1:N
     tL=(k-1)*R+1;
     for t=tL:tL+L-1
         ss(t)=ss(t)+W(t-tL+1)*s(t);
     end
 end
 S=(R/W0).*ss;
 wavwrite(S,Fs,'S.wav');
```

2. We alternate the sign of the odd frames (each windowed piece is a frame), and repeat the process.

The sound is like an alien voice.

```
clear all;
[s,Fs]=wavread('WhenAllElse8k.wav');
sound('WhenAllElse8k.wav');
L=10*10^(-3)*Fs;
W=hamming(L);

W0=sum(W);
z=[];
t1=1;
t2=length(s);
R=L/4;
N=floor((length(s)-L)/R);

 ss=zeros(length(s),1);
 for k=1:N
     tL=(k-1)*R+1;
     for t=tL:tL+L-1
         ss(t)=ss(t)+(-1)^k*W(t-tL+1)*s(t);
     end
 end
 S=(R/W0).*ss;
 wavwrite(S,Fs,'S.wav');
```

Julian Jarzebowski

3. We FFT each frame of speech, conjugate it with Matlab's conj() function, then we overlap-add (OLA) the pieces. What does conj() do to the windowed pieces?

How does the resulting speech sound for each of the 4 overlaps?

The resulting sound is robotic.