Abstract

This paper reports on signal sampling and multiple methods of convolution. The input signal was a randomly generated noise signal of 1024 samples. This project had two different transfer signals (Hk). The first signal sampled was $0.54 - 0.46 \cos(2\pi n/256)$ [256 samples]. The second function or signal that was sampled was $0.42 - 0.5 \cos(2\pi n/256) + 0.08 \cos(4\pi n/256)$ [256 samples]. Both signals contained 256 samples and were zero padded to meet differing requirements. The first thing done was to plot the noise signal and the hamming window for the given functions. To perform the first convolution method, the signals were padded to N+M-1 being 1024+256-1 = 1279. From padding the signals, the conjugates were found. Then the equation $\sum x[k]h[(i - k)\mod(N)]$ was used to perform the linear convolution of the two signals, x[n] and h[n]. Next, the fourier transform of the signals was found. From the transform, the linear convolution was found by multiplying the transformed signals. Then the inverse fourier was taken and plotted for y[n]. From comparison of the two signals, the results were the same, but the second, was shortened down to 1024 samples. This procedure was performed for both given signals.
1 Technical Discussion

Matlab was the main tool used in this project. All programming was done in the form of Matlab’s ‘m files.’ To sample the random signal, a loop was designed to generate 1024 random integers between 0 and 10. For all cases, n’s value was between 1 and 1025. The hamming was done the same way except the for loop went through the given equation instead of randomly generating the samples. The total number of samples was 256.

To convolve the signals, the signals were first padded with zeros. Then the circular convolution equation was used in the form of a nested for loop. To obtain the N-point fourier transform, the built in matlab function $\textit{fft()}$ was used for the signal transformation. Once transformed, a for loop was used to multiply the transformed functions together. Then the inverse fourier transform function $\textit{ifft()}$ was used to take the inverse transform of the convolved signal. After all these results were plotted, the exact process was used again for the other $h[n]$ signal.
2 Discussion of Results

The results of sampling the signals match the expected results. The hamming window functions were as expected. The biggest differences that was seen between the different convolution methods was that the circular convolution equation limited the end signal when compared to the other methods. It looked more “cut-off” than the others.

When the signals were multiplied together in the frequency domain, the resulting time domain graph did not wholly resemble the linear convolution obtained previously. The beginning and end of the graph were “cut-off.” Initially, this was not expected. After inspecting the formulas used and methods involved, it became clear that signals being multiplied together in the frequency domain did not contain enough samples. These signals were padded up from 1024 samples to 1279 samples. After the signal length was increased, the convolution graph perfectly matched the graph previously found by the circular convolution form of linear convolution. These results can be seen below.
3 Results

3.1 Generated Graphs

Figure 1: Random Noise Signal

Figure 2: Hamming Window
Figure 3: Linear Convolution

Figure 4: Fourier Transform of the Transfer Function (H)
Figure 5: Fourier Transform of Input Function (X)

Figure 6: Convolution in Frequency Domain
Figure 7: Convolution in Time Domain

Figure 8: Modified Linear Convolution
Figure 9: Random Noise Signal for 2nd Function

Figure 10: Hamming Window for 2nd Function
Figure 11: Linear Convolution for 2nd Function

Figure 12: Fourier Transform of Transfer Function (H) for 2nd Function
Figure 13: Fourier Transform of Input Function (X) for 2nd Function

Figure 14: Convolution in Frequency Domain for 2nd Function
Figure 15: Convolution in Time Domain for 2nd Function

Figure 16: Modified Linear Convolution for 2nd Function
4 Appendix

4.1 Matlab Code

%Questions 1-6
Lab3.m

close all;
clear all;
clcc;

%Question 1
N = 1024;
x=zeros(1,N);
for i = 1 : N+1
  x(i)=round(rand*10);
end
figure;
plot(x);
title('Random Noise Signal');
xlabel('Samples (time)');
ylabel('Magnitude');

%Question 2
M = 256;
h=zeros(1,M);
for i = 1 : M+1
  h(i)=0.54-0.46*cos(2*pi*i/M);
end
figure;
plot(h);
ylabel('0.54-0.46cos(2Pi*n/M)');

%Question 3
%Linear Convolution
arrLen=N+M-1;
tmp=zeros(1,arrLen-N);
x1=[x tmp];
tmp=zeros(1,arrLen-M);
h1=[h tmp];
for i = 1 : arrLen
  tmp = zeros(1,arrLen);
  for j = 1 : arrLen
    tmp(j)=x1( 1 + mod( (i-j),(arrLen+1)))*h1(j);
  end
  y(i) = sum(tmp);
end
figure;
plot(y);
title('Linear Convolution');
xlabel('Samples (time)');
ylabel('Magnitude');

%Question 4
%tmp = zeros(1,N);
tmp = fft(x);
tmpX = tmp;
X = abs(tmp)/max(abs(tmp));
figure;
plot(-512:512,fftshift(X));
title('1024 Point FFT of x[n]');
xlabel('X[k]');
ylabel('Magnitude');

%Question 5
Y = tmpX.*tmpH;
figure;
plot(-512:512,fftshift(Y));
tmp = ifft(Y);
ylabel('Magnitude');

%Question 6
13
%tmp = zeros(1,N);
tmp = fft([x zeros(1,arrLen-N)]);
tmpX = tmp;
figure;
plot(-512:512,fftshift(tmpX));
tmp = fft([h zeros(1,arrLen-M)]);
tmpH = tmp;
figure;
plot(-512:512,fftshift(tmpH));
Ya = tmpX.*tmpH;
figure;
plot(-512:512,fftshift(Ya));
tmp = ifft(Ya);
y1a = abs(tmp);
figure;
plot(y1a);
title('Modified Fourier Transforms used to find Linear convolution');
xlabel('Samples (time)');
ylabel('Magnitude');
axis([0 1300 0 800]);

%Question 7
Lab3q7.m
clear all;
close all;
cic;

%Question 1
N = 1024;
x=zeros(1,N);
for i = 1 : N+1
    x(i)=round(rand*10);
end
figure;
plot(x);
title('Random Noise Signal');
xlabel('Samples (time)');
ylabel('Magnitude');

%Question 2
M = 256;
h=zeros(1,M);
for i = 1 : M+1
    h(i)=0.42 - 0.5*cos(2*pi*i/M) + 0.08*cos(4*pi*i/M);
end
figure;
plot(h);
title('Hamming Window Signal');
xlabel('Samples (time)');
ylabel('0.42 - 0.5cos(2*Pi*n/M)');

%Question 3
%Linear Convolution
arrLen=N+M-1;
tmp=zeros(1,arrLen-M);
x1=[x tmp];
tmp=zeros(1,arrLen-M);
h1=[h tmp];
for i = 1 : arrLen
    tmp = zeros(1,arrLen);
    for j = 1 : arrLen
        tmp(j)=x1(1+mod((i-j),(arrLen+1)))*h1(j);
    end
    y(i) = sum(tmp);
end
figure;
plot(y);
title('Linear Convolution');
xlabel('Samples (time)');
ylabel('Magnitude');

%Question 4
X = abs(tmp)/max(abs(tmp));
figure;
plot(-512:512,fftshift(X));
title('1024 Point FFT of x[n]');
xlabel('Samples (frequency)');
ylabel('X[k]');

H = abs(tmp)/max(abs(tmp));
figure;
plot(-512:512,fftshift(H));
title('1024 Point FFT of h[n]');
xlabel('Samples (frequency)');
ylabel('H[k]');

%Question 5
Y=tmpX.*tmpH;
figure;
plot(-512:512,fftshift(Y));
tmp = ifft(Y);
title('X[k]*H[k]');
xlabel('Samples (frequency)');
ylabel('Y[k]');
y1 = abs(tmp);
figure;
plot(y1);
title('y[n] found by Inverse of Y[k]');
xlabel('Samples (time)');
ylabel('Magnitude');
axis([0 1200 0 800]);

%Question 6

tmp = zeros(1,N);
tmp = fft([x zeros(1,arrLen-N)]);
tmpXa = tmp;
figure;
plot(-512:512,fftshift(Xa));
tmp = abs(tmp)/max(abs(tmp));
figure;
plot(-512:512,fftshift(Xa));
tmp = abs(tmp)/max(abs(tmp));
figure;
plot(-512:512,fftshift(H));

Ya=tmpXa.*tmpHa;
figure;
plot(-512:512,fftshift(Ya));
tmp = ifft(Ya);
y1a = abs(tmp);
figure;
plot(y1a);
title('Modified Fourier Transforms used to find Linear convolution');
xlabel('Samples (time)');
ylabel('Magnitude');
axis([0 1300 0 800]);