Assignment 10:Linear and Circular Convolution

Abhishek Sekar
EE18B067

May 11,2020

1 Introduction

Convolution is ubiquitous in DSP.One of the main uses of the DFT is to implement Convolution. The equation for the convolution between two signals is described as follows.

$$y[n] = x[n] * h[n] \tag{1}$$

$$\Rightarrow \sum_{k=-\infty}^{k=\infty} x[n] \cdot h[n-k] \tag{2}$$

In the frequency domain, this operation becomes a multiplication as follows.

$$Y[m] = X[m] \cdot H[m] \tag{3}$$

Similarly, we can define a circular convolution by extending x[n] into an infinite periodic sequence. In this assignment, we experiment with linear convolution and circular convolution on a few signals and also attempt to perform linear convolution using a modified circular convolution.
Moreover, we look at the Zadoff Chu Sequence and study some of its special properties.

2 Coefficients of the FIR filter

Given below is the stem plot of the FIR filter in time domain.

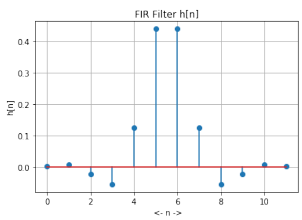

Figure 1: The FIR filter h[n]

From the plot, we see that the filter coefficients resemble a sinc function and thus we expect it's frequency magnitude response to be rect(w). Shown below are the magnitude and the phase response of the filter.

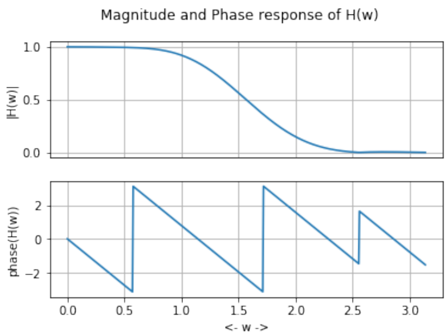

Figure 2: The magnitude and phase response of H(w)

Since, the FIR filter's response is real, the frequency response will be even, and thus, the response resembles that of a low pass filter. Additionally, we see that the response indeed resembles rect(w) as we had earlier hypothesized.

2.1 Code

```
# importing the necessary libraries

import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as sig

#Question 1

#reading the file h.csv and extracting the filter coefficients

h_n = np.genfromtxt('h.csv') #filter coefficients

#Question 2

plt.stem(h_n)
plt.title('FIR Filter h[n]')
```

```
plt.xlabel(' <- n ->')
plt.ylabel('h[n]')
plt.grid()
plt.show()

#magnitude and phase response
w,H_w = sig.freqz(h_n,1)

plt.subplot(2,1,1)
plt.plot(w,np.abs(H_w))
plt.grid()
plt.ylabel('|H(w)|')
plt.tick_params(axis='x',which='both',bottom=False,top=False,labelbottom=False

plt.subplot(2,1,2)
plt.plot(w,np.angle(H_w))
plt.grid()
plt.ylabel('phase(H(w))')
plt.xlabel('<- w ->')
plt.suptitle('Magnitude and Phase response of H(w)')
plt.show()
```

3 Linear convolution with x[n]

We consider an input signal x[n] $= \cos(0.2\pi n) + \cos(0.85\pi n)$ shown below.

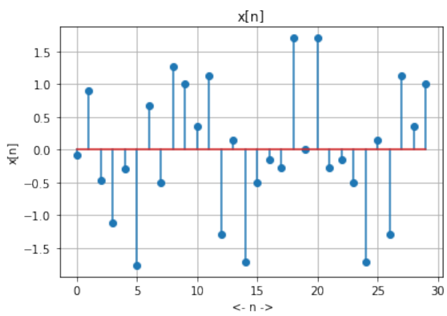

Figure 3: x[n]

We then, pass this signal through the FIR filter described in the previous section and the resulting output can be described as a linear convolution. The output of the filter is shown in the plot below. We'd expect the higher frequency component to be relatively absent as the frequency response of the filter looks like a lowpass one.

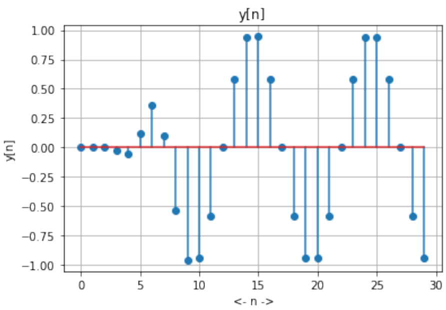

Figure 4: $y[n] = x[n] * h[n]$

The above plot confirms our hypothesis as we see a sinusoid which approximately has a frequency of 0.2π. The higher frequency cosine has been damped by the FIR filter as we had expected.

3.1 Code

```
#Question 3
n = np.linspace(1, 2**10, 2**10)
x_n = np.cos(0.2*np.pi*n) + np.cos(0.85*np.pi*n)

plt.stem(x_n[:30])
plt.title('x[n]')
plt.xlabel(' <- n ->')
plt.ylabel('x[n]')
plt.grid()
plt.show()


#Question 4
y_n = np.convolve(x_n,h_n)
plt.stem(y_n[:30])
plt.title('y[n]')
```

```python
plt.xlabel(' <- n ->')
plt.ylabel('y[n]')
plt.grid()
plt.show()
```

4 Circular convolution with x[n]

We implement the circular convolution in the frequency domain by making use of the signals' DFT. In order to perform the circular convolution, we zero pad the filter coefficients with additional set of zeros to make its length the same as that of the signal x[n].
The output of the circular convolution is shown below.

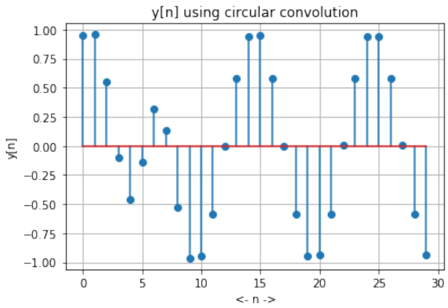

Figure 5: Output of circular convolution

Similarly, we can perform linear convolution using circular convolution by a slight modification in the way we zero pad the signals.
We just pad the signals by a lot more zeroes than before such that the number of points being considered for the circular convolution is $\geq$ the length of the output of the linear convolution which is len(x) + len(h) - 1.
The output of this shown below.

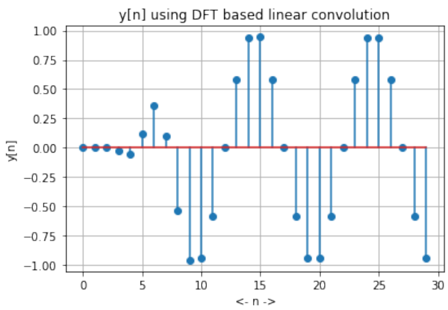

Figure 6: Linear convolution using padded circular convolution

As we can see, this plot is identical to what we had obtained earlier for the linear convolution.

4.1 Code

```
#Question 5
y_circ_n = np.fft.ifft(np.fft.fft(x_n)*np.fft.fft(np.concatenate((h_n, np.zero

plt.stem(y_circ_n[:30])
plt.title('y[n] using circular convolution')
plt.xlabel(' <- n ->')
plt.ylabel('y[n]')
plt.grid()
plt.show()

#Question 6
x_pad_n = np.concatenate((x_n, np.zeros(len(h_n)-1)))
h_pad_n = np.concatenate((h_n, np.zeros(len(x_n)-1)))
y_lin_n = np.fft.ifft(np.fft.fft(x_pad_n)*np.fft.fft(h_pad_n))

plt.stem(y_lin_n[:30])
plt.title('y[n] using DFT based linear convolution')
plt.xlabel(' <- n ->')
plt.ylabel('y[n]')
plt.grid()
plt.show()
```

5 Circular Correlation of the Zadoff Chu sequence

The Zadoff Chu Sequence is a special sequence satisfying the below properties.

- It is a complex sequence.
- It is a constant amplitude sequence.
- The auto correlation of a Zadoff-Chu sequence with a cyclically shifted version of itself is zero.

- Correlation of Zadoff-Chu sequence with a delayed version of itself will give a peak at that delay.

Shown below is a magnitude plot of the Zadoff-Chu sequence.

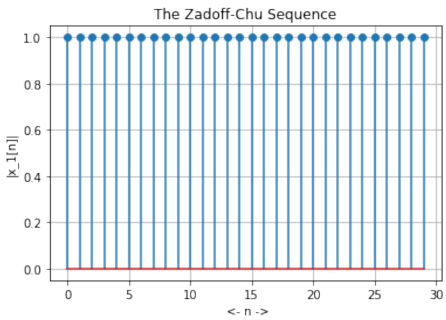

Figure 7: The magnitude plot of x_1[n]

We try verifying the last two properties of the Zadoff-Chu sequence by performing a correlation with a cyclically delayed version of itself which is delayed by 5 samples. The plot shown below verifies the properties.

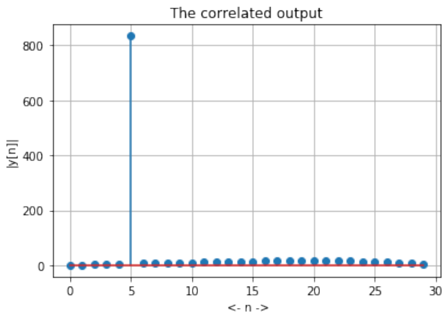

Figure 8: Output of the correlation

We see that the output is only non-zero at the place at the shift which results in a peak and it is zero everywhere else.

5.1 Code

```
#Question 7
with open("x1.csv",'r') as f: #open the csv file in read mode

    lines = f.readlines()

```

```
    f.close()

x_1_n = []
for i,line in enumerate(lines):
    line = line[:-1]
    if(i):
        line = line[:-1]
        line += 'j'

    x_1_n.append(complex(line))


x_1_n = np.asarray(x_1_n,dtype = np.complex)
plt.stem(np.abs(x_1_n[:30]))
plt.title('The Zadoff-Chu Sequence')
plt.xlabel(' <- n ->')
plt.ylabel('|x_1[n]|')
plt.grid()
plt.show()

#correlation
y_corr = np.fft.ifftshift(np.correlate(np.roll(x_1_n,5), x_1_n, "full"))
plt.stem(np.abs(y_corr[:30]))
plt.title('The correlated output')
plt.xlabel(' <- n ->')
plt.ylabel('|y[n]|')
plt.grid()
plt.show()
```

6 Conclusion

Linear Convolution is not optimal if we implement it in a summation based fashion. We should use the advantage of the frequency domain and implement it using a DFT operation of the input signals. This is a much faster and more efficient way to implement convolution numerically.
Additionally, as we saw, circular convolution can be used for linear convolu-

tion at a much faster speed by zero padding the input signals appropriately. Lastly, we learnt the properties of the Zadoff-Chu sequence and also verified them explicitly.