EE2703: Applied Programming Lab
Assignment 10
Convolution using DSP

G Abhiram, EE20B037

May 16, 2022

1 Abstract

We will explore linear and circular convolutions and their implementation using the DFT. The goal of this assignment is to:

- To use Linear and Circular Convolution to find the output of a low pass FIR filter for a given input.
- To see how linear convolution can be interpreted as circular convolution with aliasing
- To analyse the correlation of the Zadoff-Chu sequence.

2 Assignment

2.1 Helper Functions

I am using two helper functions to plot the signal and it's spectrum respectively. They are as follows:

```
    figNum = 0
def signal(t, x, figTitle=None, style='b-', blockFig=False, showFig=False, saveFig=True,
    global figNum
    plt.figure(figNum)
    plt.title(figTitle)
    plt.grid()
    plt.ylabel(yLabel)
    plt.xlabel(xLabel)
    if(stemPlot):
        plt.stem(t, x, linefmt='b-', markerfmt='bo')
    else:
        plt.plot(t, x, style)
    if(xLimit):
        plt.xlim(xLimit)
    if(yLimit):
        plt.ylim(yLimit)
```

```
    if(saveFig):
        plt.savefig(str(figNum)+".png")
    if(showFig):
        plt.show(block=blockFig)
    figNum+=1

def spectrum(w, Y, figTitle=None, magStyle='b-', phaseStyle='g', xLimit=None, yLimit=Nor
    global figNum
    plt.figure(figNum)
    plt.suptitle(figTitle)
    plt.subplot(211)
    plt.grid()
    plt.plot(w, abs(Y), magStyle, lw=2)
    plt.ylabel(r"$\| "+type+"\|$")
    if (xLimit):
        plt.xlim(xLimit)
    if (yLimit):
        plt.ylim(yLimit)
    plt.subplot(212)
    plt.grid()
    plt.plot(w, np.angle(Y), phaseStyle, lw=2)
    plt.xlim(xLimit)
    plt.ylabel(r"$\angle "+type+"$")
    plt.xlabel(r"$\omega\ \to$")

    if(saveFig):
        plt.savefig(str(figNum)+".png")
    if(showFig):
        plt.show(block=blockFig)
    figNum+=1
```

2.2 Part 1: Low Pass FIR Filter

We are given the coefficients of a FIR Low Pass Filter in the **h.csv** file, which can be plotted to get the filter as shown: If we observe carefully, we can see that the envelope of the filter looks like a $sinc(x)$ function, which means that it's DTFT will look similar to a $rect(\omega)$. The magnitude and phase response of the given filter is given by:

$$H(e^{j\omega}) = \sum_{n=0}^{11} h[n]e^{-j\omega n} \tag{1}$$

We read the coefficients and using the scipy.freqz function. The code is as follows:

```
filter = np.genfromtxt("h.csv")
signal(range(len(filter)), filter, "FIR Filter ($h[n]$)", showFig=True, yLabel=r"$h[n]$"
w, H = sp.freqz(filter, 1)
spectrum(w, H, "Frequency Response of FIR Filter ($H(e^{j\omega}))$", type="H", showFig=
```

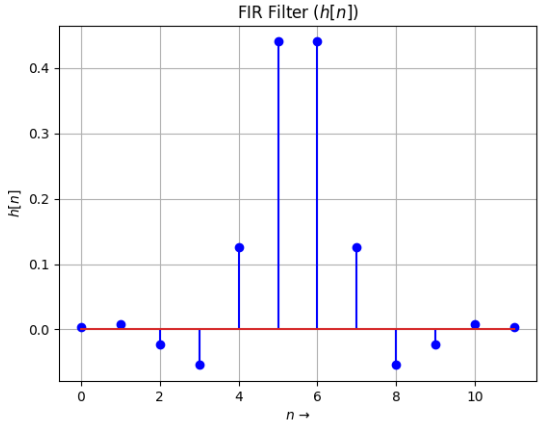


Figure 1: The FIR Filter

We compute the frequency response of the FIR filter and plot the corresponding bode plots, which are as follows:

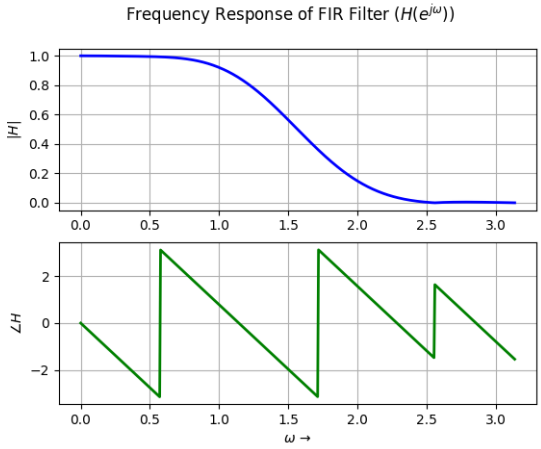


Figure 2: Magnitude and Phase Response of the Filter

From the magnitude plot, we notice that the system acts a low pass filter, and acts as a linear phase filter in some intervals, and the system introduces a constant group delay.

2.3 Part 2: Plotting the input signal $x[n]$:

We are given an input signal $x[n] = cos(0.2\pi n) + cos(0.85\pi n)$ to the filter, which can be plotted using the code shown below to get the following plot:

```
n = np.linspace(1, 2**10, 2**10)
x = np.cos(0.2*pi*n) + np.cos(0.85*pi*n)
signal(n, x, figTitle="$x[n] = cos(0.2\pi n) + cos(0.85\pi n)$", xLimit=[0, 50], showFig
```

The input signal has frequency components at around 0.63 rad/s and 2.670 rad/s. Therefore, by passing it through a matching low pass filter, we can extract the 0.2π component alone.

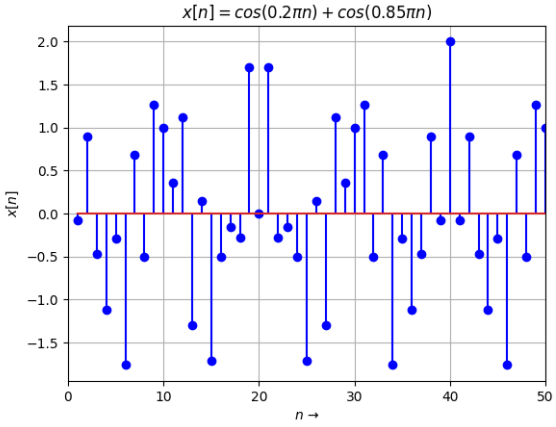


Figure 3: Input Signal $x[n]$

2.4 Part 3: Linear Convolution of $x[n]$ and $h[n]$

To get the output of passing the input signal into the FIR filter, we can convolve the input signal and the impulse response of the filter using the **np.convolve** command.

$$y[n] = x * h = \sum_{\{k:\ h[k]\neq 0\}} x[n-k]h[k]$$

Using the above command to get output, and then plotting the output signal we get:

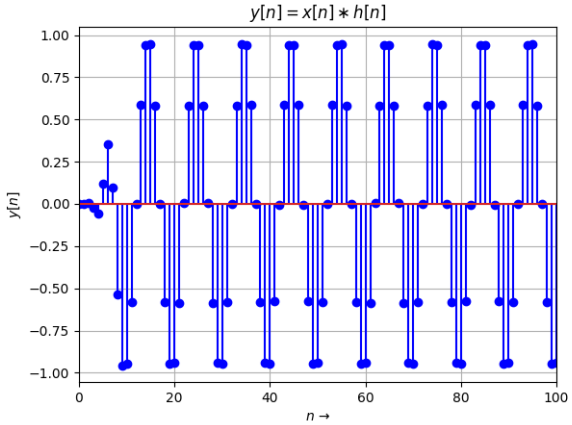


Figure 4: Output of Filter, $y[n]$

To look at the output in CT domain, we make use of the following code:

```
y1 =  convolve(x,filter)
figure()
plot(range(len(n)+len(filter)-1),y1,'r')
xlabel(r'$n\rightarrow$',size=15)
ylabel(r'$y\rightarrow$',size=15)
title(r'CT plot of output $y=x*h$')
xlim([1,100])
savefig("CT Output Linear Convolution.png")
```

The CT output signal we get, is as follows:

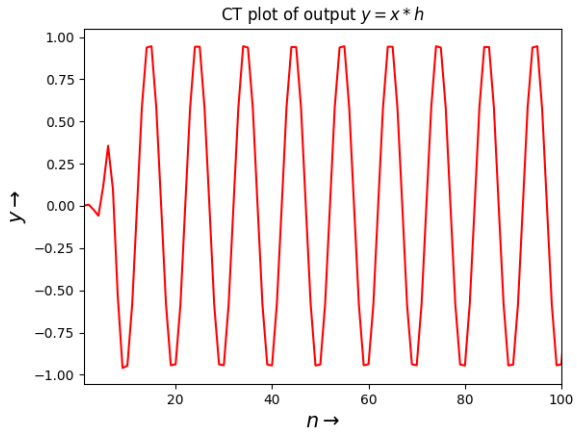


Figure 5: CT Output of Filter, $y[n]$

As expected, we see a sinusoid of frequency approximately 0.2π. This is because the filter has damped the high frequency component $cos(0.85\pi n)$ greatly. We can see from the magnitude response that $|H(e^{j\omega})|\Big|_{\omega=0.85\pi}$ is close to 0.

2.5 Part 4: Circular Convolution of $x[n]$ and $h[n]$

An N-point circular convolution is defined as:

$$y[.] = x[.] \text{ N } h[.]$$

$$y[n] = \sum_{m=0}^{N-1} x[m]h[(n-m) \mod N],\ n \in [0, N-1]$$

2.5.1 Using Circular Convolution to perform Linear Convolution

- We zero pad the $h[n]$ to fit a 2^m window.
- $x[n]$ is broken into sections 2^m long.
- Now, each section of $x[n]$ is convolved with $h[n]$ using the corresponding DFTs. Appropriate padding is added to ensure that the length of the output is as expected.
- Each succeeding convolution adds to the already computed value of $y[n]$ at each index n, which may be updated by the previous convolution.

The intuition behind using circular convolution to get a linear convolution output is as follows:

- We can easily see that the $y[.]$ obtained as a result of circular convolution and that obtained by linear convolution will not be same always. However, if we take an P-point circular convolution, by appropriately zero padding the two signals, such that $P \geq len(y_{lin}[.])$, then, the two will match.
- This is because $y_{cir}[.]$ is the principal period of $\tilde{y}_{lin}[.]$, which is a periodic extension of $y_{lin}[.]$, with period P. So, if the period $P < len(y_{lin}[.])$, then there will be time-aliasing, which will distort the output.

- Otherwise, if $P \geq len(y_{lin}[.])$, then, there will not be any time-aliasing and so, $y_{cir}[.]$ and $y_{lin}[.]$ will have the same information.

We get the following graphs, depending on the length of the circular convolution output.

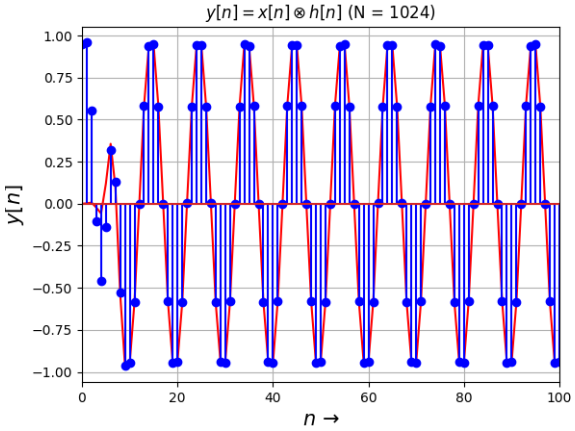


Figure 6: 1024-point circular convolution

If we take a $(1024 + 11 - 1) = 1034$-point circular convolution, there is no aliasing and we get the same output as linear convolution using circular convolution, which is as follows:

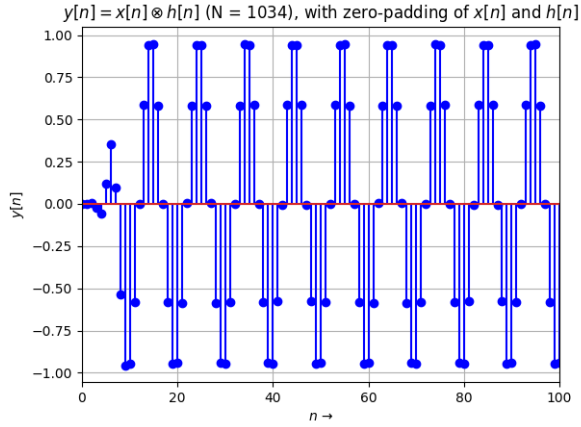


Figure 7: 1034-point circular convolution

But, to efficiently implement linear convolution through circular convolution, we have to take 2^m-point circular convolution. This is because, the underlying algorithm, the *Fast Fourier Transform* (FFT) is most efficient when it is used for 2^m-point signals.

2.6 Part 5: Circular Correlation of the Zadoff-Chu Sequence

Zadoff-Chu sequences are important signals in communication. ZadoffChu sequences are used in the 3GPP LTE Long Term Evolution air interface in the Primary Synchronization Signal (PSS), random access preamble (PRACH), uplink control channel (PUCCH), uplink traffic channel (PUSCH) and sounding reference signals (SRS).
Consider the Zadoff-Chu sequence, a commonly used sequence in communication. The properties of the sequence are :

- It is a complex sequence.
- It is a constant amplitude sequence.
- The auto correlation of a ZadoffChu sequence with a cyclically shifted version of itself is zero, except at the shift.
- Correlation of ZadoffChu sequence with the delayed version of itself will give a peak at that delay.

Using the following code, we can extract the values from the given 'x1.csv' file and the Zadoff-Chu Sequence can be plotted.

```
file2 = "x1.csv"
f1 = open("x1.csv")
zChu = f1.readlines()
zChu  = asarray([complex(i[:-1].replace('i','j')) for i in zChu],dtype = 'complex')

spectrum(list(range(len(zChu))), np.asarray(zChu, dtype=np.complex),
r"Zadoff-Chu Sequence", phaseStyle='r-', showFig=True, type=r"zChu[n]", yLimit=[-0.5, 1.
```

The Zadoff-Chu sequence can be plotted to obtain the figure shown below:

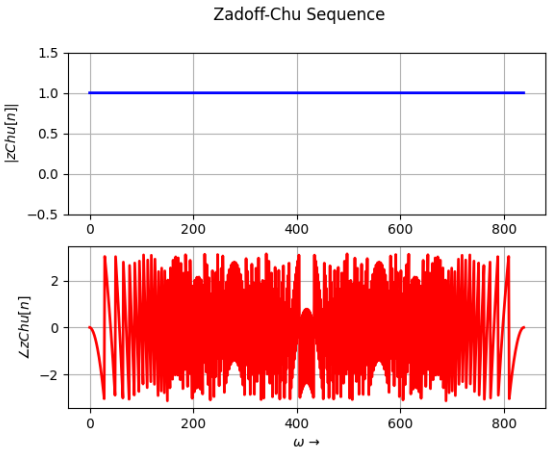
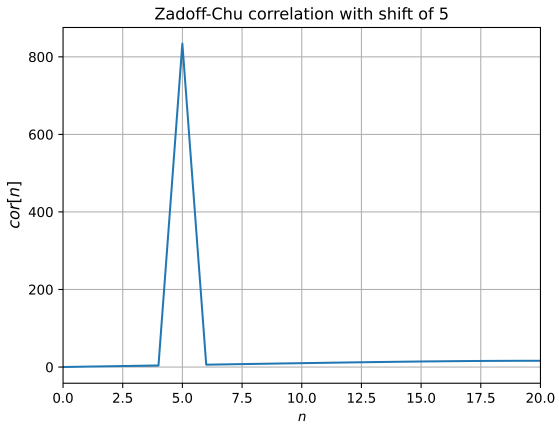


Figure 8: Zadoff-Chu Sequence

We can verify property (3) and (4) given above, by plotting the auto-correlation of $zChu[.]$ with a cyclically delayed version of itself. I have taken the auto-correlation with $zChu[.]$ cyclically rotated by 5. The code to do that is as follows:

```
zChuShifted = np.roll(zChu, 5)
y = np.fft.ifftshift(np.correlate(zChuShifted, zChu, "full"))
figure(9)
plot(abs(y))
xlabel("$n$")
ylabel("$cor[n]$", size = 12)
title("Zadoff-Chu correlation with shift of 5")
xlim([0,20])
grid(True)
savefig('8.png', dpi=1000)
```

The result is as follows:



In the above figure, we can see that the auto-correlation is non-zero and peaks only at $n = 5$, which corresponds to the delay value.

3 Conclusions

- We studied the magnitude and phase response of a 12^{th} order FIR Low-Pass Filter.
- To convolve two signals of length n each, the time complexity is of the order n^2 and thus is computationally very expensive. Instead, optimising using the FFT algorithm, we get the complexity to reduce to $nlog(n)$.
- A faster way to perform the convolution is to use the DFTs of the input and the filter. Circular convolution can be used for the implementation of linear convolution, with a much faster computation speed.
- We computed the output for an input signal from the FIR filter using three methods, Linear Convolution, Circular Convolution and Circular Convolution without any aliasing.
- For the Zadoff-Chu sequence, the correlation of the sequence with the cyclic shifted version of itself has a peak at the point equal to the shift, and also has a non-zero value only at that point.