EE2703 : Applied Programming Lab
ASSIGNMENT 10

RAKSHIT PANDEY
EE20B106

May 14, 2022

INTRODUCTION

In this week's assignment our main aim is to work with convolution of FIR signals. We perform linear convolution, circular convolution and circular convolution with the help of linear convolution. In the end we work with the Zadoff Chu sequence and analyze it's properties.

QUESTIONS

Importing important libraries

In order to do this assignment in python we need to import some important python libraries such as numpy, pylab and scipy.signal.

Numpy will allow us to work efficiently with arrays, pylab will enable us to plot curves and scipy.signal is a very useful library for digital signal processing.

Code

```
import numpy as np
import scipy.signal as sp
from pylab import *

```

Importing data from .csv files

In this week's assignment we need to import data from two .csv files, namely ”h.csv” and ”x1.csv”. The first file contains coefficients for a FIR(Finite impulse response) filter, and the second file contains coefficients of Zadoff-Chu sequence.

We will first read through the files and store corresponding data in arrays. The stored data will be strings, so we need to convert it into appropriate data types and store them in new arrays. This is done in python as follows:

1. Code (For ”h.csv” file)

```
with open("h.csv") as f:
  lines = f.readlines()

b = []
```

```python
for i in range(len(lines)):
  b.append(float(lines[i]))

```

2. Code (For "x1.csv" file)

```python
# Zadoff-chu sequence
with open("x1.csv") as f1:
  lines1 = f1.readlines()

b2 = []
for i in range(len(lines1)):
  x = complex((lines1[i].rstrip("\n")).replace("i","j")) # First we rstrip to
    remove "\n" at the end of the string, then we replace i with j for complex to
    understand the argument
  b2.append(x)

```

Plotting magnitude and phase response for FIR filter

We use freqz attribute of scipy.signal library to find the transfer function and corresponding values at different frequencies from $[0,\pi]$. Once we get the impulse response of the FIR filter, we can easily find the magnitude and phase response of our FIR filter.

Code

```python
w,h = sp.freqz(lines) # freqz is a scipy.signal attribute to generate the
    frequency response corresponding to filter, which in this case is FIR filter

# From the curve it is very clear that this frequency reponse is for a low pass
    filter

figure(1)
xlabel(r"$\omega\rightarrow$")
ylabel("Magnitude")
title("Magnitude response of Low pass filter")
grid()
plot(w,abs(h))

figure(2)
xlabel(r"$\omega\rightarrow$")
ylabel("Phase")
title("Phase response of Low pass filter")
grid()
```

```
plot(w,angle(h))

```

The magnitude and phase response plots are as follows:

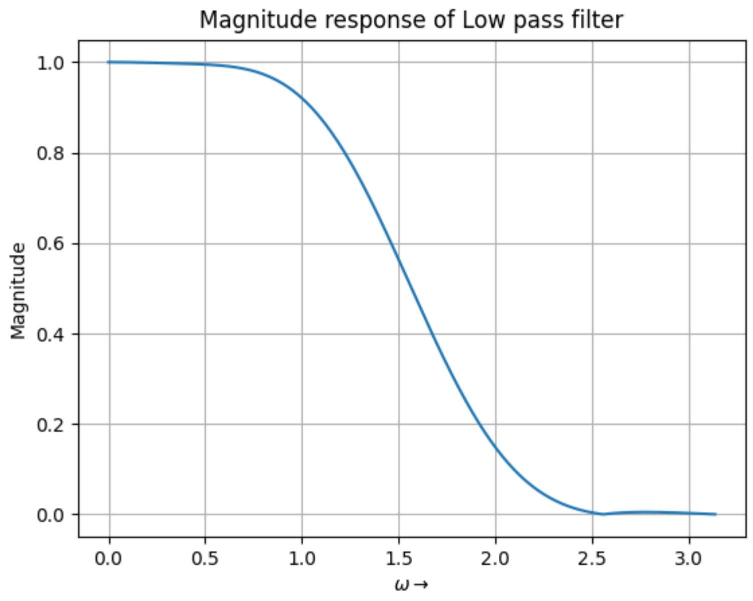

Figure 1: Magnitude response of Low pass filter

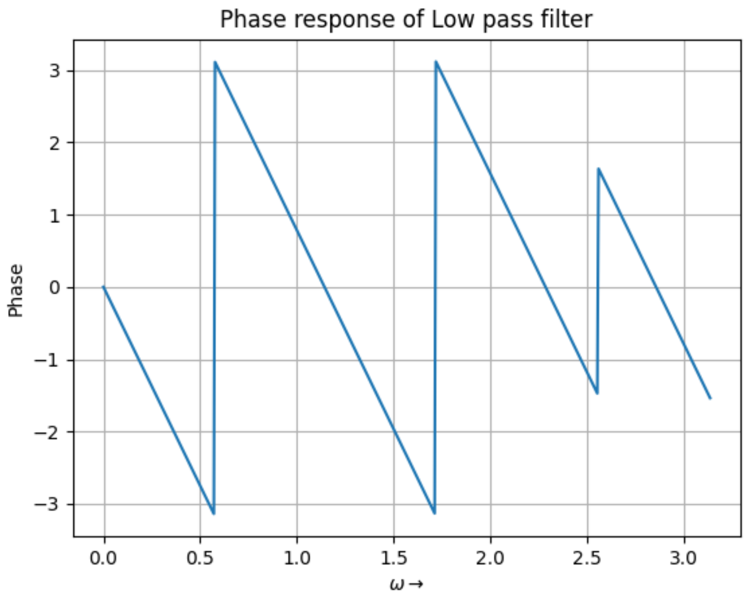

Figure 2: Phase response of Low pass filter

From the magnitude response it becomes very clear that the given filter is a Low pass filter. Hence, this filter will attenuate Higher frequencies and allow lower frequencies to pass through.

To generate sequence for x[n] and plot the output

In this question we are given a sequence x[n] which is defined as:

$$x[n] = cos(0.2\pi n) + cos(0.85\pi n)$$

where n = 1,2,3,...,2^{10}

We are supposed to plot this sequence for different samples of n, this is done in python as follows:

Code

```
n = arange(1,2**10+1)
x_n = cos(0.2*Pi*n) + cos(0.85*Pi*n)

figure(3)
xlabel("n")
ylabel("x[n]")
title("Input signal")
grid()
plot(n,x_n)
```

We get the following plot of x[n] vs n:

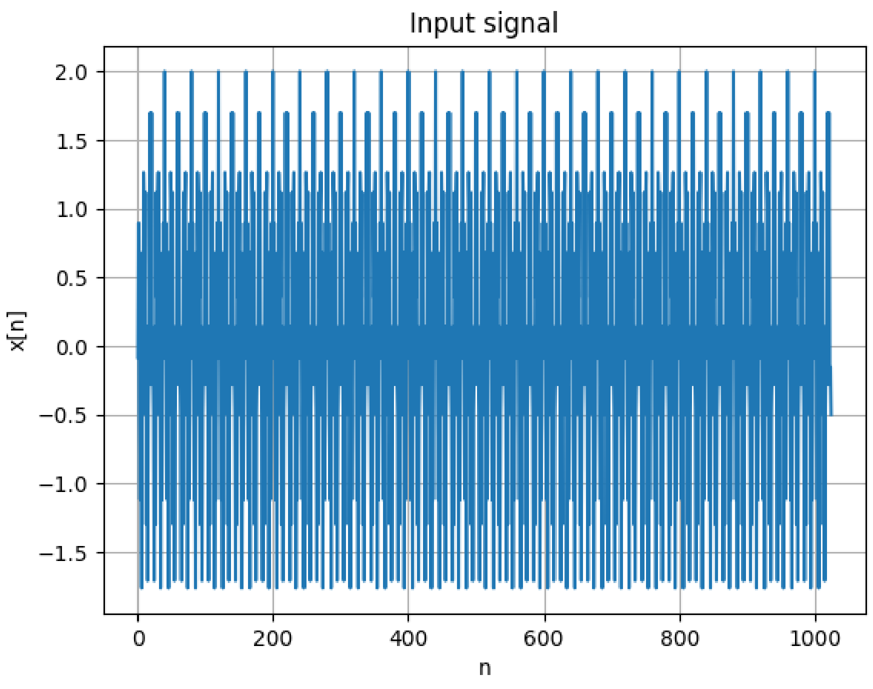

Figure 3: Input signal x[n] vs n

Ouput of filter by linear convolution

We will find the output of filter for input sequence x[n] with the help of linear convolution of x[n] with impulse response of FIR filter. The linear convolution of x[n] with h[n] is given by:

$$y[n] = \sum_{k=0}^{n-1} x[n-k]h[k]$$

We implement this in python as follows:

Code

```python
# Output using linear convolution
y_n = np.zeros(len(x_n))
for i in arange(len(x_n)):
    for k in arange(len(lines)):
        y_n[i]+=x_n[i-k]*b[k]

figure(4)
xlabel("n")
ylabel("y[n]")
title("Output signal as a result of linear convolution")
grid()
plot(n,real(y_n))
```

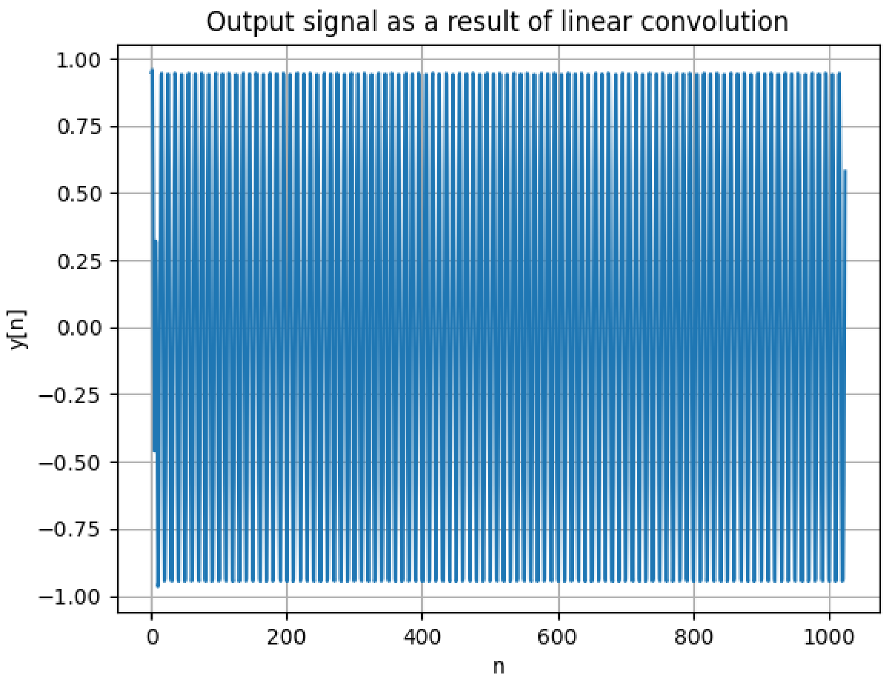

Figure 4: Output of filter using linear convolution

Output of filter using circular convolution

We now try to find the output of filter using circular convolution instead of linear convolution. The circular convolution of two sequences x[n] and h[n] is given by:

$$\tilde{x}[n] = \frac{1}{N}\sum_{k=0}^{N-1}\tilde{X}[k]e^{j\frac{2\pi}{N}kn}$$

We compute the circular convolution of x[n] with h[n] in python as follows:

Code

```
# Output using Circular convolution
y1=ifft(fft(x_n)*fft(concatenate((b,zeros(len(x_n)-len(b))))))

figure(5)
xlabel("n")
ylabel("y[n]")
title("Output signal as a result of circular convolution")
grid()
plot(n,real(y1))
```

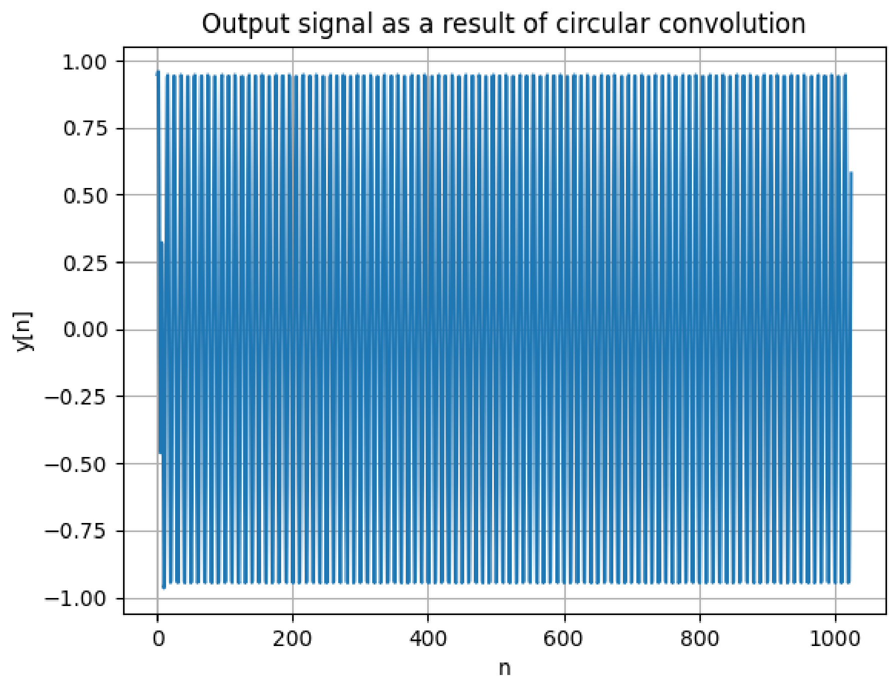

Figure 5: Output of filter using circular convolution

Output by circular convolution using linear convolution

We have realized that this efficient solution is non causal and requires the entire signal to perform a convolution. We now implement a linear method of circular convolution so that we need not depend on the next infinite values in the input but only a finite number of values(This is still non causal but the delay in the response is lower).

This is done in python as follows:

Code

```python
#Output of circular convolution using linear convolution

def circular_conv(x,h): # Function defined to calculate the circular convolution using the linear convolution
    P = len(h)
    n_ = int(ceil(log2(P)))
    h_ = np.concatenate((h,np.zeros(int(2**n_)-P)))
    P = len(h_)
    n1 = int(ceil(len(x)/2**n_))
    x_ = np.concatenate((x,np.zeros(n1*(int(2**n_))-len(x))))
    y = np.zeros(len(x_)+len(h_)-1)
    for i in range(n1):
        temp = np.concatenate((x_[i*P:(i+1)*P],np.zeros(P-1)))
        y[i*P:(i+1)*P+P-1] += np.fft.ifft(np.fft.fft(temp) * np.fft.fft( np.concatenate( (h_,np.zeros(len(temp)-len(h_))) ))).real
    return y

y_lc = circular_conv(x_n,b)

figure(6)
xlabel("n")
ylabel("y[n]")
title("Output of circular convolution using linear convolution")
grid()
plot(n,real(y_lc[:1024])) # While plotting we must take care to truncate the output values to take only first 1024 values.
```

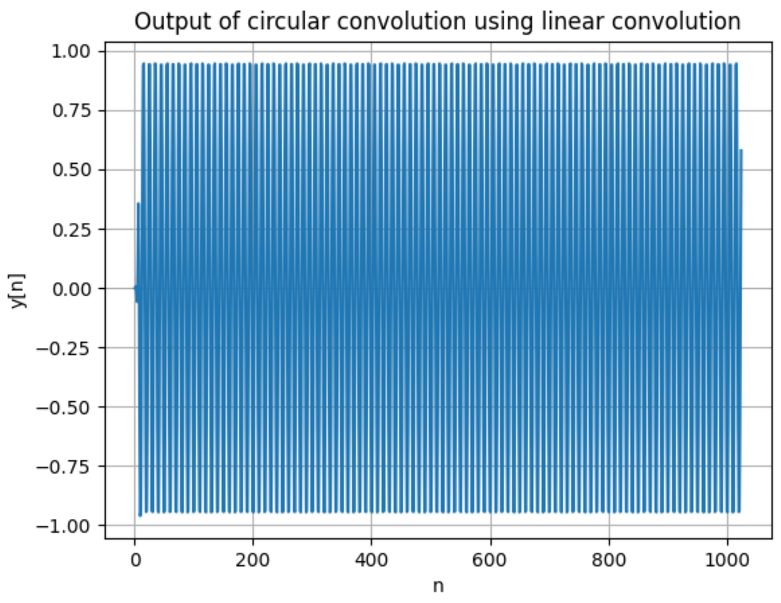

Figure 6: Output by circular convolution using linear convolution

Zadoff-Chu sequence

In order to study circular correlation, we use Zadoff-Chu sequence (which is very widely used in communications). A Zadoff-Chu sequence has following properties:

- It is a complex sequence.
- It is a complex 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 the delayed version of itself will give a peak at that delay.

In this question, we are supposed to correlate x1 with a cyclic shifted version of x1, with a shift of 5 to the right, and plot the correlation output. This is performed in python as follows:

Code

```
X = np.fft.fft(b2)
b3 = np.roll(b2,5)
cor = np.fft.ifftshift(np.correlate(b3,b2,'full'))

figure(7)
```

```
xlabel("t")
ylabel("Correlation")
title("Auto correlation")
grid()
xlim([0,20])
plot(np.arange(0,len(cor)),abs(cor))
show()
```

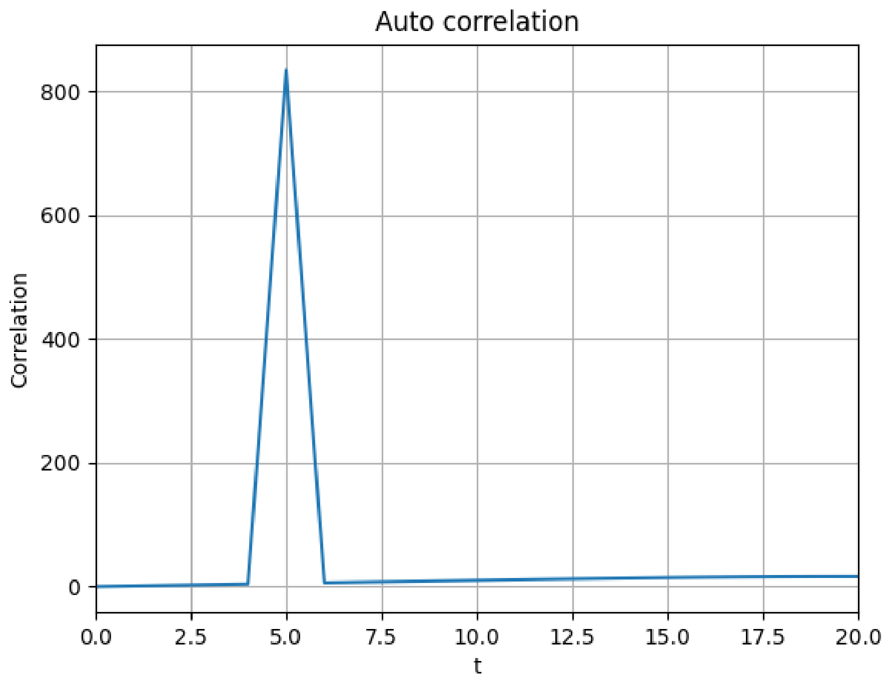

Figure 7: Correlation of Zadoff-Chu sequence with the delayed version of itself

From the plotted curve, it becomes clear that there is a peak at 5.0, which verifies the property that "correlation of Zadoff-Chu sequence with the delayed version of itself will give a peak at that delay", this delay in our case was 5.0.

CONCLUSION

So, In this week's assignment we worked with a FIR filter, we found output of filter by convolving input sequence with the impulse response of the filter. We used three different methods of convolution, namely 'linear convolution', 'circular convolution' and 'circular convolution using the linear convolution'. In the end we worked with Zadoff-Chu sequence and verified it's properties.