Assignment 10: Linear and Circular Convolution

Siddharth D P
EE18B072

May 20, 2020

1 Abstract

The goal of this assignment is the following.

- To implement Linear and Circular Convolution to find the output of a digital filter for causal signals.
- Assess the computational cost of direct summation for Linear convolution and directly implementing the `np.convolve` function.
- To see how linear convolution can be implemented using circular convolution with aliasing.
- To analyse the correlation of the Zadoff-Chu sequence.

2 Assignment

2.1 Importing all the standard libraries

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

2.2 Reading the filter coefficients from a file and plotting the transfer function

We read the coefficients of the transfer function of the filter from the file "h.csv" and plot the FIR filter's magnitude (in dB scale) and phase (after unwrapping the angle) is plotted using the *scipy.signal.freqz()* function.

```
with open("h.csv") as file: # Use file to refer to the file object
   data = file.read()
data=np.fromstring(data, sep='\n')

w, h = signal.freqz(data)

fig = plt.figure()
plt.subplot(2,1,1)
plt.title('Digital filter frequency response')
plt.plot(w, 20*np.log10(abs(h)), 'b')
plt.ylabel('Magnitude', color='b')
plt.xlabel('Frequency [rad/sample]')
plt.grid()

plt.subplot(2,1,2)
angles = np.unwrap(np.angle(h))
#angles=np.pi/180*angles
plt.plot(w, angles, 'g')
plt.ylabel('Angle in radians(unwrap)', color='g')
plt.xlabel('Frequency [rad/sample]')

plt.grid()
plt.axis('tight')
fig.tight_layout()
```

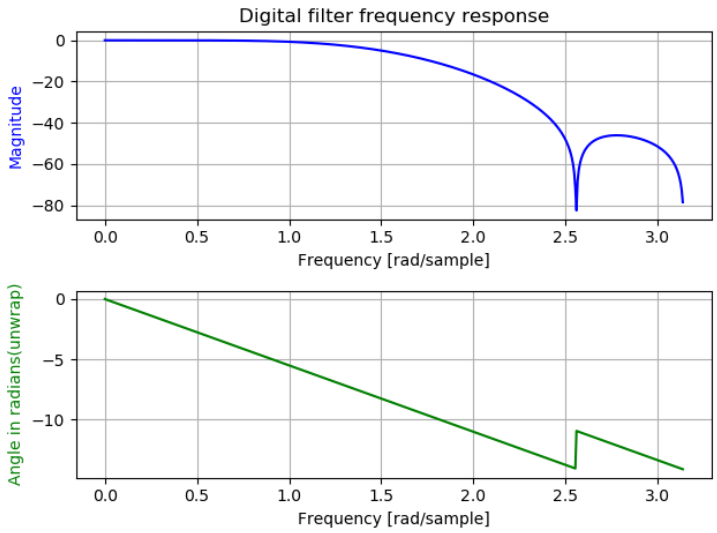

Figure 1: Frequency response of the filter

We notice from the magnitude plot that the system is indeed a lowpass

filter. From the phase plot, we infer that the filter introduces a constant group delay.

2.3 Generating the given function sequence and plotting it

We generate the function $cos(0.2\pi n) + cos(0.85\pi n)$ for values ranging from 1 to 2^{10} and plot it, limiting the observable values to 200 points for ease of viewing.

```
n=np.arange(1,2**10+1,1)
x=np.cos(0.2*np.pi*n)+np.cos(0.85*np.pi*n)
plt.figure()
plt.plot(n,x)
plt.title("Plot of cos(0.2$\pi$ n)+cos(0.85$\pi$ n)")
plt.xlabel("n")
plt.ylabel("f(n)")
plt.grid()
plt.xlim([0,200])
```

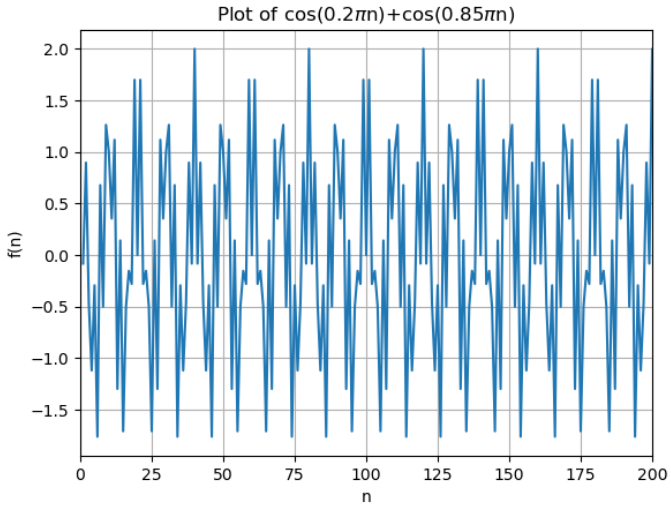

Figure 2: Graph of the function $cos(0.2\pi n) + cos(0.85\pi n)$

We see that the figure is extremely spiky and periodic for the range of values given.

2.4 Passing the sequence through the filter using linear convolution and plotting the output

We first do linear convolution using as follows-

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

using the inbuilt python function `convolve()`and plot it.

```
y=convolve(x, data)

plt.figure()
plt.plot(range(len(n)+len(data)-1), y)
plt.title("Linear convolution using direct summation")
plt.xlabel("n")
plt.ylabel("f(n)")
plt.grid()
plt.xlim([0,200])
```

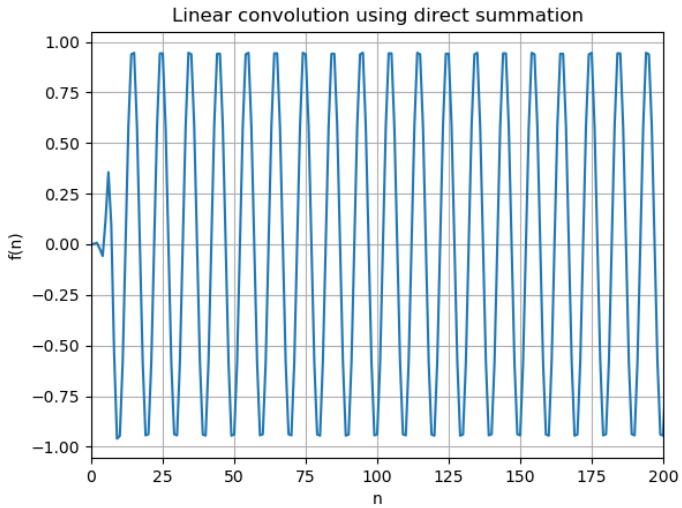

Figure 3: Linear convolution using the *convolve* function

Notice that the high frequency variations in x[n] have been attenuated by the system. The low frequency component remains.

2.5 Using DFT to perform Circular Convolution

The length of the convolution output y[n] is expected to be len(x)+len(h)-1. We also know that the number of frequencies in the DFT of a signal is equal

to the number of samples in the time domain. Thus, the input x[n] and the filter h[n] are zero padded, so that the DFT of the y[n] has the appropriate number of samples. The following code snippet does the job :

```
x_=np.concatenate((x, zeros(len(data)-1)))
y1=ifft(fft(x_)*fft(concatenate((data,zeros(len(x_)-len(data))))))

plt.figure()
plt.plot(range(len(y1)), y1)
plt.title("Output of circular convolution")
plt.xlabel("n")
plt.ylabel("f(n)")
plt.grid()
plt.xlim([0,200])
```

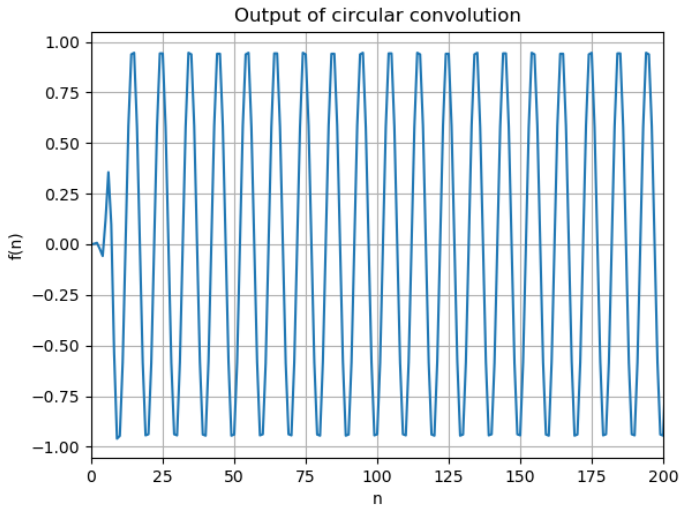

Figure 4: Circular convolution performed using DFT and inverse DFT

We see that the signals obtained using *np.convolve* and the DFT are identical.

2.6 Using Circular Convolution to perform Linear Convolution

The main steps followed in the implementation of linear convolution using circular convolution are:

- 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 done 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 following code snippet implements the above algorithm:

```
ef lin_circular_conv(x,data):
    P = len(data)
    n_ = int(ceil(log2(P)))
    data_ = np.concatenate((data,np.zeros(int(2**n_)-P)))
    P = len(data_)
    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(data_)-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] += ifft(fft(temp) * fft( np.concatenate((data_,np.zeros(
    return y

y2 = lin_circular_conv(x,data)

plt.figure()
plt.plot(range(len(y2)), y2)
plt.title("Linear convolution using circular convolution")
plt.xlabel("n")
plt.ylabel("f(n)")
plt.grid()
plt.xlim([0,200])
```

The output obtained is thus, identical to the ones obtained above using the the *convolve()* function and using DFTs to perform circular convolution. In fact, the error is of the order of 10^{-16} possibly due to the finite accuracy in calculating the DFT using the in-built *ifft* function.

2.7 Circular Correlation

We consider the Zadoff-Chu sequence, a commonly used sequence in communication. We read the sequence from a file "x1.csv" and plot the correlation of the sequence with a circularly 5 units shifted version of itself. We notice that the following properties of the sequence:

- It is a complex sequence.

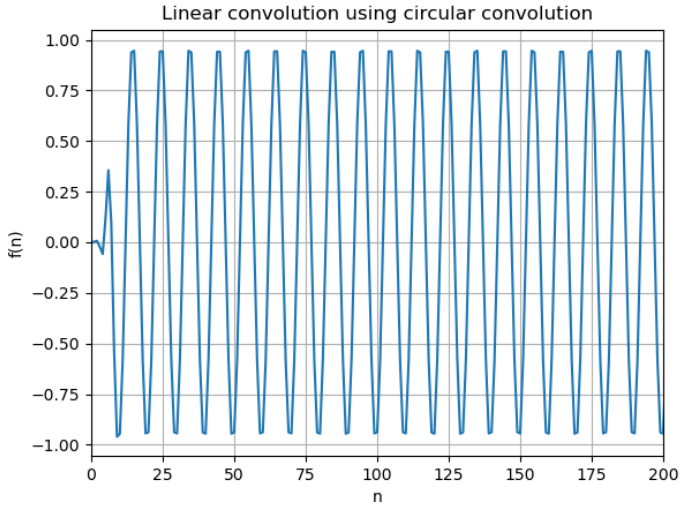

Figure 5: Linear convolution using circular convolution

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

The auto correlation property is verified using the following code snippet:

```
with open("x1.csv") as file2:
    data2=file2.readlines()
    data2=asarray([complex(i[:-1].replace('i','j')) for i in data2], dtype = 'comple

Dshifted=np.roll(data2, 5)
corr=ifft(conj(fft(data2))*(fft(Dshifted)))

plt.figure()
plt.title("Correlation of cyclically shifted Zadoff Chu sequence")
plt.stem(range(len(corr)), abs(corr))
plt.xlabel("n")
plt.ylabel("f(n)")
plt.grid()
xlim([0,30])

plt.show()
```

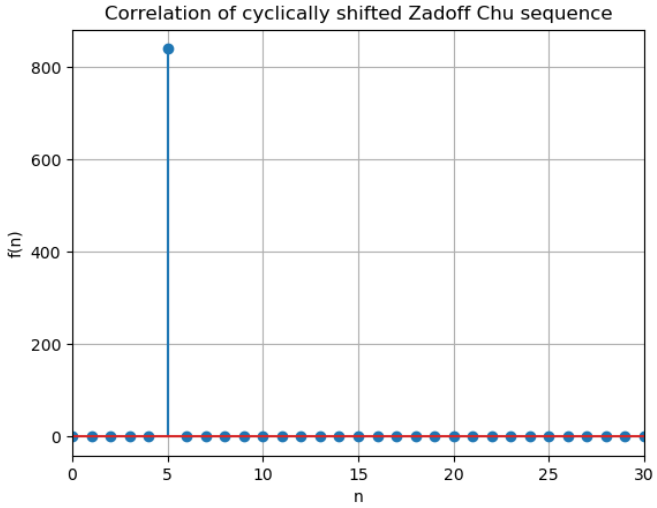

Figure 6: Correlation of cyclically shifted Zadoff-Chu sequence

We observe that the auto-correlation of x[n] and a cyclic shifted version of x[n], with a shift of 5 units to the right gives a peak at 5 and vanishes everywhere else.

3 Conclusion

- We used Linear and Circular Convolution to find the output of a low-pass filter.
- The linear convolution algorithm implemented using a direct summation is non-optimal and computationally expensive. A faster way to perform the convolution is to use the DFTs of the input and the filter. Circular convolution with aliasing can be used for the implementation of linear convolution, with a much faster computation speed.
- The magnitude and phase response of a low pass filter were studied and the system's output, for a mixed frequency signal was obtained through three different methods. We noticed how the three methods yielded similar graphs.
- For the Zadoff-Chu sequence, the auto correlation output with a cyclically shifted version of itself was found to be non-zero only at the point corresponding to the shift.