EE2703 Applied Programming Lab - Assignment No 10

Name: Atishay Ganesh
Roll Number: EE17B155

April 13, 2019

1 Abstract

The goal of this assignment is the following.

- To use Linear and Circular Convolution to find the output of a low pass filter.
- To see how linear convolution is the same as circular convolution with aliasing.
- To analyse the correlation of the Zadoff-Chu sequence.

2 Assignment

2.1 Setting up the variables

Importing the standard libraries

```
from pylab import *
import scipy.signal
```

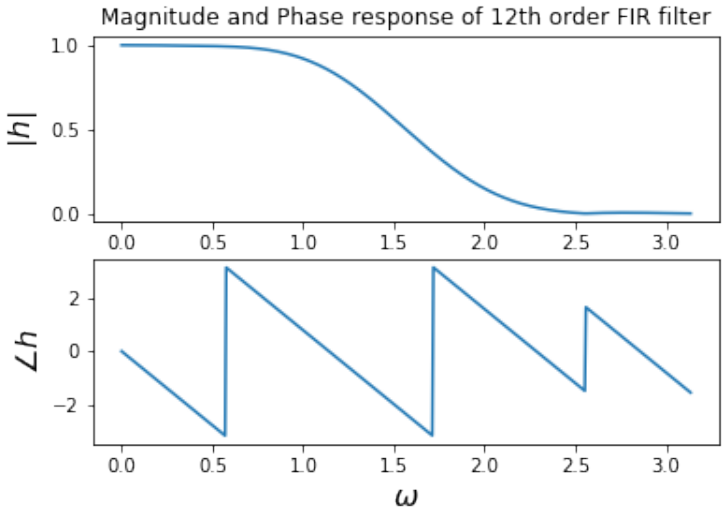
2.2 Reading the low pass filter

We read the transfer function and verify that it is indeed a low pass filter

```
f = open('h.csv')
lpf = f.readlines()
f.close()
lpf = asarray([float(i[:-1]) for i in lpf])
w,h = scipy.signal.freqz(lpf)
fig, (ax1, ax2) = plt.subplots(2, 1)
title("Magnitude and Phase response of 12th order FIR filter",pad = 125)
```

```
ax1.plot(w,abs(h))
ax2.plot(w,angle(h))
ax1.set_ylabel(r"$|h|$",size=16)

ax2.set_ylabel(r"$\angle{h}$",size=16)
ax2.set_xlabel(r"$\omega$",size=16)
plt.show()
```



2.3 Defining a Linear Convolution Function

We define a linear convolution function We vectorize the inner for loop for better efficiency.

```
def con(a,b):
    X = len(a)
    K = len(b)
    b = b[::-1]#because time reverse the second signal
    y = zeros(X+K-1)
    #we initally define as zeros and then later reassign each value
    for i in range(1,X+K):
        y[i-1] = sum(a[max(0,i-K):min(X,i)]*b[max(0,K-i):min(K+X-i,K)])
        #vectorised convolution operation
    return y
```

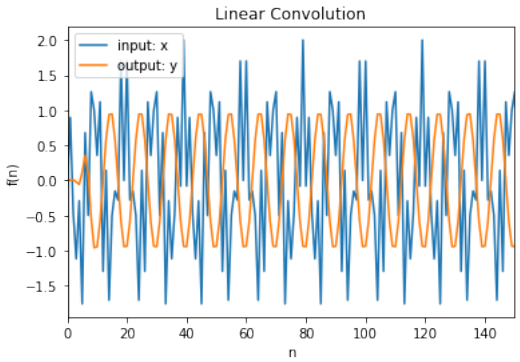
2.4 Linear Convolution using summation

We define the input function which is the sum of two sinusoids, one inside and outside the passband of the lpf. We perform linear convolution of the input with the low pass filter.

```
n= arange(1,2**10+1)
x = lambda n: cos(0.2*pi*n)+cos(0.85*pi*n)
inp = x(n)

out = con(inp,lpf)
#out = np.convolve(inp,lpf)
figure(1)
plot(inp)
plot(out)
xlabel("n")
ylabel("f(n)")
title("Linear Convolution")
legend(("input: x",'output: y'))
xlim(0,150)# just to make the result understandable
show()
```

The first few values are obviously wrong, but the output after 16 values is a delayed version of the input filtered by the lpf. The delay is clear from the fact that there is a phase lag.



2.5 Circular Convolution using FFT

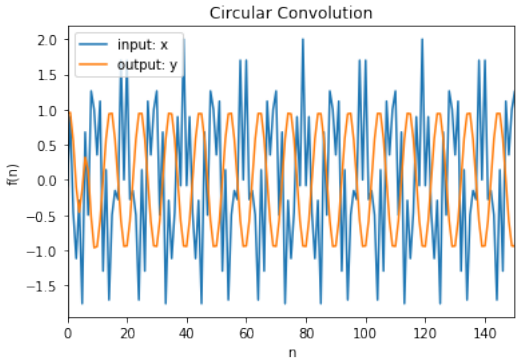
We use the FFT to calculate the output. Of course, convolution is multiplication in the frequency domain, and for multiplication both the vectors should be of the same size, so we zero pad the transfer function.

```
padded_a_lot_h = concatenate((lpf,zeros(len(inp)-len(lpf))))
y1 = ifft(fft(inp)*fft(padded_a_lot_h));
figure(2)
plot(inp)
plot(real(y1))
xlim(0,150)
xlabel("n")
```

```
ylabel("f(n)")
title("Circular Convolution")
legend(("input: x",'output: y'))

show()
```

It is pretty much the same as the previous case except for the edge condition of values from 0 to 16, which is quite different.



2.6 Linear Convolution using Circular Convolution

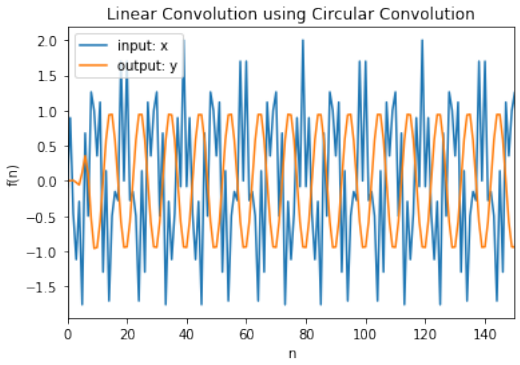
We now use Circular Convolution with Aliasing to get linear convolution.

```
padded_h = concatenate((lpf,zeros(4)))# to make it power of 2
x_split = asarray(split(inp,len(inp)//(2*len(padded_h))))
#samples come in groups of 32
out = np.zeros(len(padded_h)+len(inp)-1)
#Defining the output to be zeros
index = 0#starting index of the output where we add the convolution result
n_iter =len(x_split)
L = len(x_split[0])
P = len(padded_h)
N = L+P-1
#N,L,P as defined in the assignment
for i in range(n_iter):
    out[index:index+N] += con(x_split[i],padded_h)
    #this is why we defined the output as zeros to start.
    #we can just add the output for each batch at the right spot.
    #as every batch of L inputs comes in,
    #L correct outputs are found and hence
    index +=L
figure(3)
plot(inp)
```

```
plot(out)
xlim(0,150)#again for ease of understanding
xlabel("n")
ylabel("f(n)")
title("Linear Convolution using Circular Convolution")
legend(("input: x",'output: y'))

show()
```

Clearly the output is the same as linear convolution case, even the fringe values from 0-15. This way we can do online processing of the signal

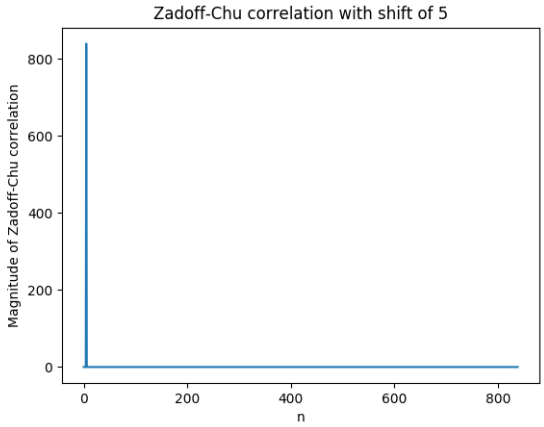


2.7 Correlation of Zadoff-Chu Sequence

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).
We read the sequence from a file and plot the correlation of the sequence with a circularly shifted version of itself.

```
f1 =open("x1.csv")
zc = f1.readlines()
zc  = asarray([complex(i[:-1].replace('i','j')) for i in zc],dtype = 'complex')
zcshift = np.concatenate((zc[-5:],zc[:-5]))
op = ifft(conj(fft(zc))*(fft(zcshift)))
#A*(jw)xB(jw) is the equivalent of correlation in the frequency domain.
plot(abs(op))
xlabel("n")
ylabel("Magnitude of Zadoff-Chu Correlation")
title("Zadoff-Chu correlation with shift of 5")
show()
```

As expected, the correlation of the Zadoff-Chu sequence with a cyclically shifted version of itself gives a peak at the delay, which in this case is 5. The peak is at 5.



3 Conclusions

- We used Linear and Circular Convolution to find the output of a low pass filter.
- We saw how linear convolution is the same as circular convolution with aliasing.
- We observed phase delay in all the cases.
- We analysed the correlation of the Zadoff-Chu sequence with a delayed version of itself and had a peak at the delay.