Assignment 10: Linear and circular convolution

Manognya Param Koolath

April 16, 2019

The task

In this assignment, we learn to convolve signals using linear convolution as well as circular convolution.

First, we download `h.csv` file and read it into the program. This file contains coefficients for an FIR filter.

We use `scipy.signal.freqz()` to convert it to frequency domain and the magnitude and phase response is as shown in Figure 1. We observe that this is a low pass filter.

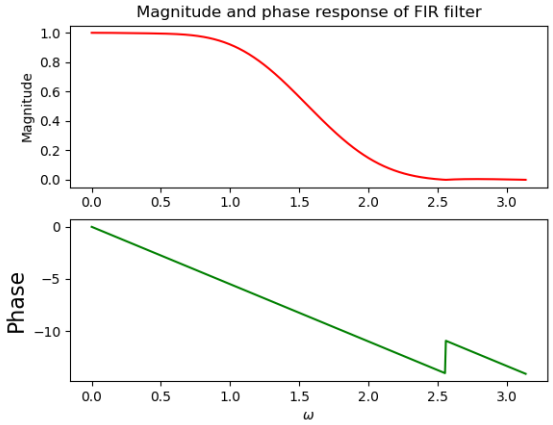

Figure 1: Magnitude and phase response of the FIR filter

Next we generate the following sequence where n is 1, 2, 3, ..., 2^{10}.

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

This sequence (x) is plotted in Figure 2.

Now we have to pass the sequence through the FIR filter that we have. For this, we convolve the two signals in the time domain.

The output obtained by doing linear convolution is as shown in Figure 3.

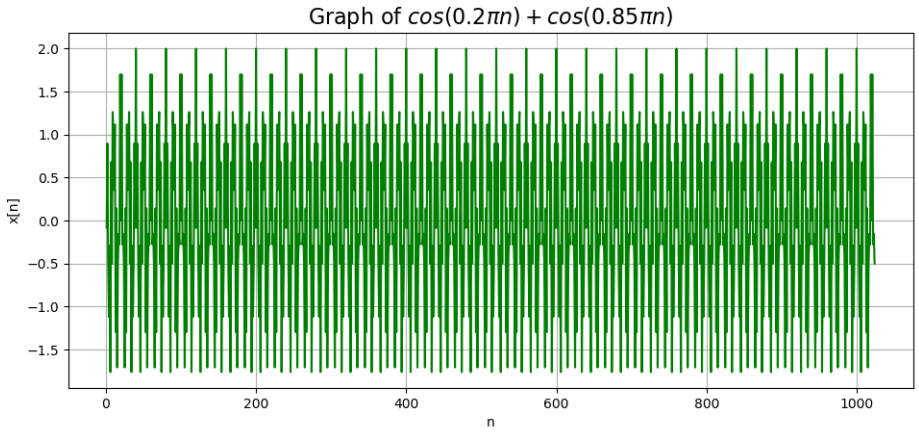

Figure 2: The sequence $x[n]$

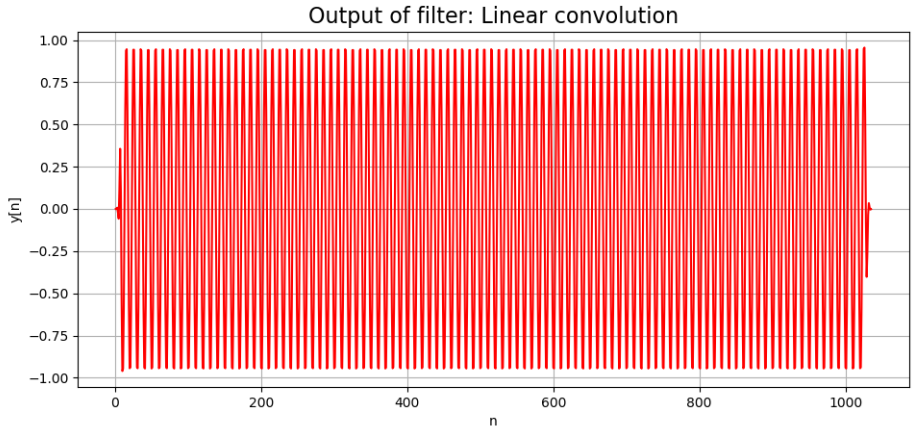

Figure 3: Output $y[n]$ using linear convolution

The output obtained by doing circular convolution is as shown in Figure 4.

The output obtained by doing the linear convolution using circular convolution is as shown in Figure 5.

Note that the filter has eliminated the high frequency component from the input, which is expected because the filter is a low pass filter.

Next we have to do circular correlation. We take the *Zadoff-Chu* sequence as an example. The coefficients of this sequence (say, x_1) is read from `x1.csv`. We take care to make sure that each coefficient is read as a complex number into the program. This sequence is as shown in Figure 7.

We know correlation can be written as $a[n] * b[n]$. In this, we correlate x_1 with a cyclic shifted version of x_1, with a shift of 5 to the right. The

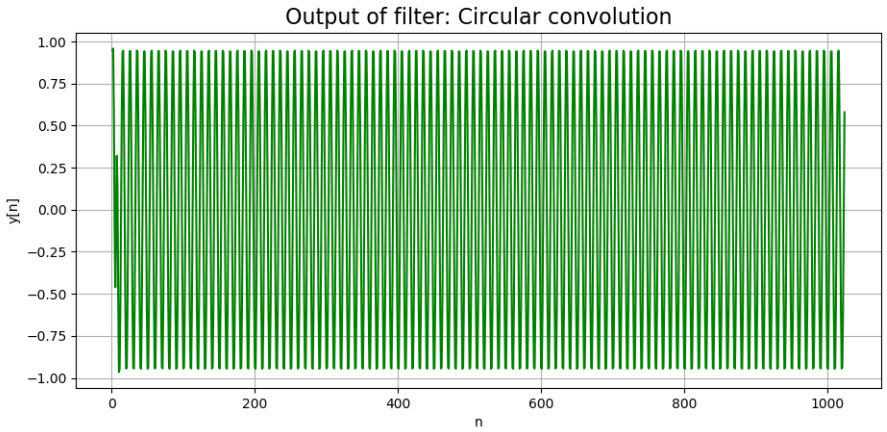

Figure 4: Output $y[n]$ using circular convolution

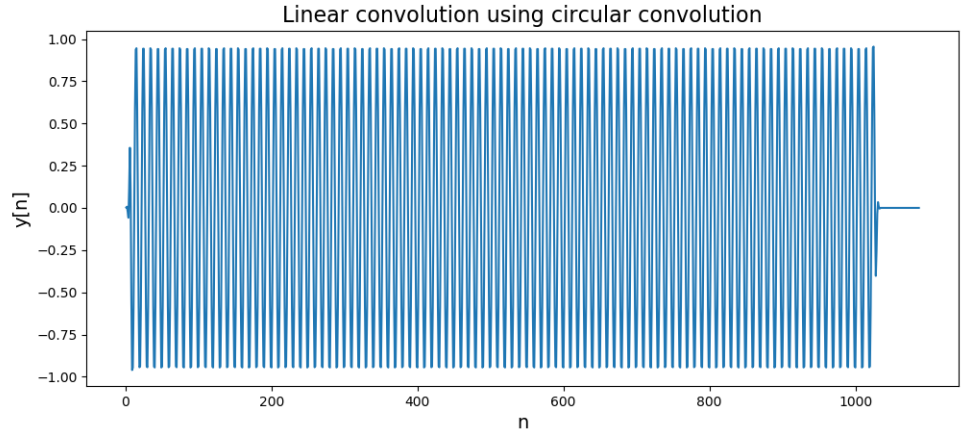

Figure 5: Output of doing linear convolution using circular convolution

output of this is as shown in Figure 8.

Python code

The code is properly commented and completely vectorised.

Assignment 10 Code

```
#Assignment 10
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as sp
```

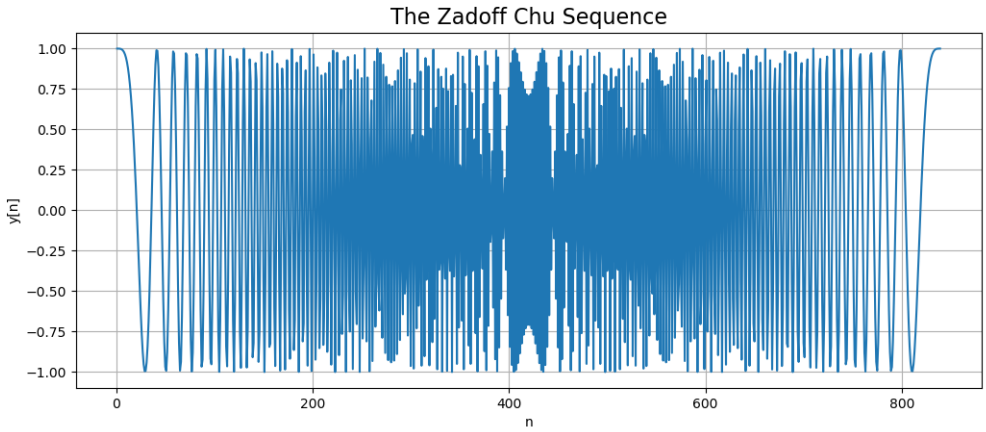

Figure 6: The *Zadoff-Chu* sequence

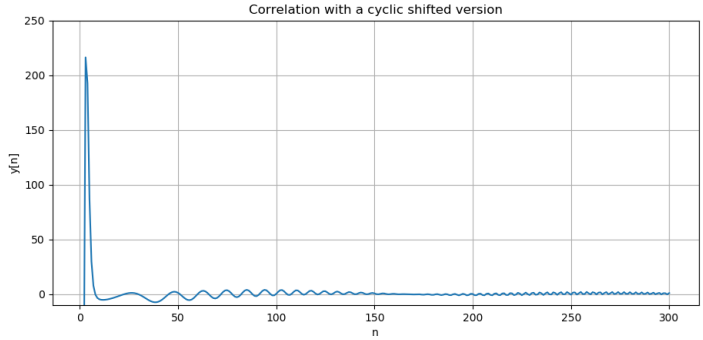

Figure 7: Correlation output of *Zadoff-Chu* sequence

```
from scipy.linalg import lstsq
import scipy.integrate as spint
import mpl_toolkits.mplot3d.axes3d as p3
from scipy import ndimage
import sympy as spy
# import pandas as pd
import csv

# Q1:
b = []#np.full(0,0.0)
with open("h.csv") as csv_file:
        csv = csv.reader(csv_file, delimiter = ',')
        for row in csv:
                # b=np.concatenate(b,float(row[0]))
```

```
                b.append(float(row[0]))
                # print(type(row[0]))
        print(b)

# Q2:
w,h = sp.freqz(b)
# plt.figure()
# plt.plot(w, abs(h), 'r')
## plt.title("Magnitude response of FIR filter")
# plt.ylabel("Magnitude")
# plt.xlabel(r"$\omega$")
# plt.show()
plt.subplot(2,1,1)
plt.plot(w,np.abs(h),'r')
# plt.xlim([-10,10])
plt.ylabel("Magnitude")
plt.title("Magnitude and phase response of FIR filter")
# plt.grid(True)
plt.subplot(2,1,2)
plt.plot(w, np.unwrap(np.angle(h)),'g')#np.unwrap
# plt.xlim([-10,10])
plt.ylabel("Phase",size=16)
plt.xlabel(r"$\omega$")
# plt.grid(True)
# savefig("fig9-1.png")
plt.show()

# Q3:
n= np.arange(1025)
n = n[1:]
# print(n)
x = np.cos(0.2*np.pi*n) + np.cos(0.85*np.pi*n)

plt.figure()
plt.plot(n,x,'g-')
plt.title("Graph of $cos(0.2\pi n) + cos(0.85\pi n)$", size=16)
plt.ylabel("x[n]")
plt.xlabel("n")
plt.grid(True)
plt.show()

# Q4:
# x = np.concatenate((np.array([0]*(11), dtype = float),x))
y = np.array([0]*1035, dtype = float)
```

```
b = np.array(b)
# print(x)
# print(b)
# print(y)
y2 = np.convolve(x,b)
# for i in range(11,1035):
#         y[i]=0
#         for k in range(12):
#                 y[i] += b[k]*x[i-k]
plt.figure()
# print(n.shape, y.shape, x.shape)
plt.plot(np.concatenate((n,np.arange(1025,1036))),y2,'r-')
plt.title("Output of filter: Linear convolution", size=16)
plt.ylabel("y[n]")
plt.xlabel("n")
plt.grid(True)
plt.show()
# Note that only the low frequency component will remain.

# Q5:
# x1=np.fft.fft(x)
# print(len(x))
# x2 = np.fft.fft(np.concatenate((b,np.array([0]*(len(x)-len(b)), dtype =
# print(x1.shape, x2.shape)
y1 = np.fft.ifft(np.multiply(np.fft.fft(x) , np.fft.fft(np.concatenate((b
print(y1.shape)
plt.figure()
# plt.plot(np.concatenate((n,np.arange(1025,1036))),y1,'g-')
plt.plot(n,y1,'g-')
plt.title("Output of filter: Circular convolution", size=16)
plt.ylabel("y[n]")
plt.xlabel("n")
plt.grid(True)
plt.show()

# Q6:
P = len(b)
L = 2**6

# y_2 = np.full(len(x)+P-1,0.0)
y_2 = np.full(len(x)-1+L,0.0+0.0j)
# for k in np.arange(0,2**10,L):
#         w = np.convolve(b, x[k:k+L])
#         y_2[k:k+P-1] += w[0:P-1]
```

```
#         y_2[k+P:k+P+L-1] += w[P:L+P-1]
#         # y_2[k:k+P+L-1] += w[0:L+P-1]
pos=0
x=np.concatenate((np.array([0]*(P-1)),x))
x=np.concatenate((x,np.array([0]*(P-1+L))))
while(pos+L <=len(x)):
# for k in np.arange(0,2**10,L):
        # w = np.convolve(b, x[k:k+L])
        # abc1=np.fft.fft(x[k:k+L+P])
        # abc2=np.fft.fft(np.concatenate((b,np.array([0]*L, dtype = float
        # print(abc1.shape,abc2.shape)
        # if(abc1.shape==abc2.shape):
                # x1=x[k:k+L+1]; #x1[:P]=np.array([0]*P)
                # x1=np.concatenate((x[k:k+L+1],np.array([0]*(P-1))))
                # x1 = np.concatenate((np.array([0]*(P-1)),x1))
        x1=x[pos:pos+L]
        w = np.fft.ifft(np.multiply(np.fft.fft(x1) , np.fft.fft(np.concat
        # print(w.shape)
        # y_2[k:k+P-1] = y_2[k:k+P-1] + w[0:P-1]
        # y_2[k+P:k+P+L-1] += w[P:L+P-1]
        y_2[pos:pos+L-P]=w[P:]
                # y_2[k:k+L] += w[P:P+L]#L+P-1]
        pos+=(L-P-1)
        # y_2[k:k+L] += w[0:L]
        # print(pos)


# P = 12
# N = 1024
# L = 64
# n = np.arange(1,1025,1)
# x = np.cos(0.2*np.pi*n) + np.cos(0.85*np.pi*n)
# y_full = np.array([])
# x = np.concatenate((x,np.array([0]*48)))
# for i in range(20):
#         xi = x[(53*i):(53*i)+L]
#         yi1 = np.fft.ifft( np.fft.fft(xi)*np.fft.fft(np.concatenate((b,np
#         yi1 = yi1[(P-1):]
#         y_full = np.concatenate((y_full,yi1))
plt.figure()
# plt.plot(n, y_full[:1024])
plt.plot(np.arange(1,len(y_2)+1),y_2)
plt.xlabel("n",size=14)
plt.ylabel("y[n]",size=14)
```

```
plt.title("Linear convolution using circular convolution",size=16)
plt.show()
# plt.plot(n[:50], y_full[:50])
# plt.title("Linear convolution using circular convolution",size=16)
# plt.xlabel("n",size=14)
# plt.ylabel("y[n]",size=14)
# plt.show()


# Q7:

b = []
with open('x1.csv') as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=',')
        for row in csv_reader:
                a = row[0]
                a = a.replace(" ","")
                a = a.replace("i", "j")
                b.append(complex(a))
bshift = [0,0,0,0,0] + b
m = np.arange(1,840,1)
plt.figure()
plt.plot(m, b)
plt.title("The Zadoff Chu Sequence", size =16)
plt.xlabel("n")
plt.ylabel("y[n]")
plt.grid(True)
plt.show()

b = np.array(b)
c = np.array(bshift)
# y1=np.convolve(b,c)
y1 = np.fft.ifft( np.concatenate(( np.fft.fft(b),np.array([0]*5) ))*np.ff
plt.figure()
# plt.plot(np.arange(1682), y1)
plt.plot(m[:300], y1[:300])
plt.title("Correlation with a cyclic shifted version")
plt.ylim([-10,250])
plt.xlabel("n")
plt.grid(1)
plt.ylabel("y[n]")
plt.show()

# # END OF CODE
```