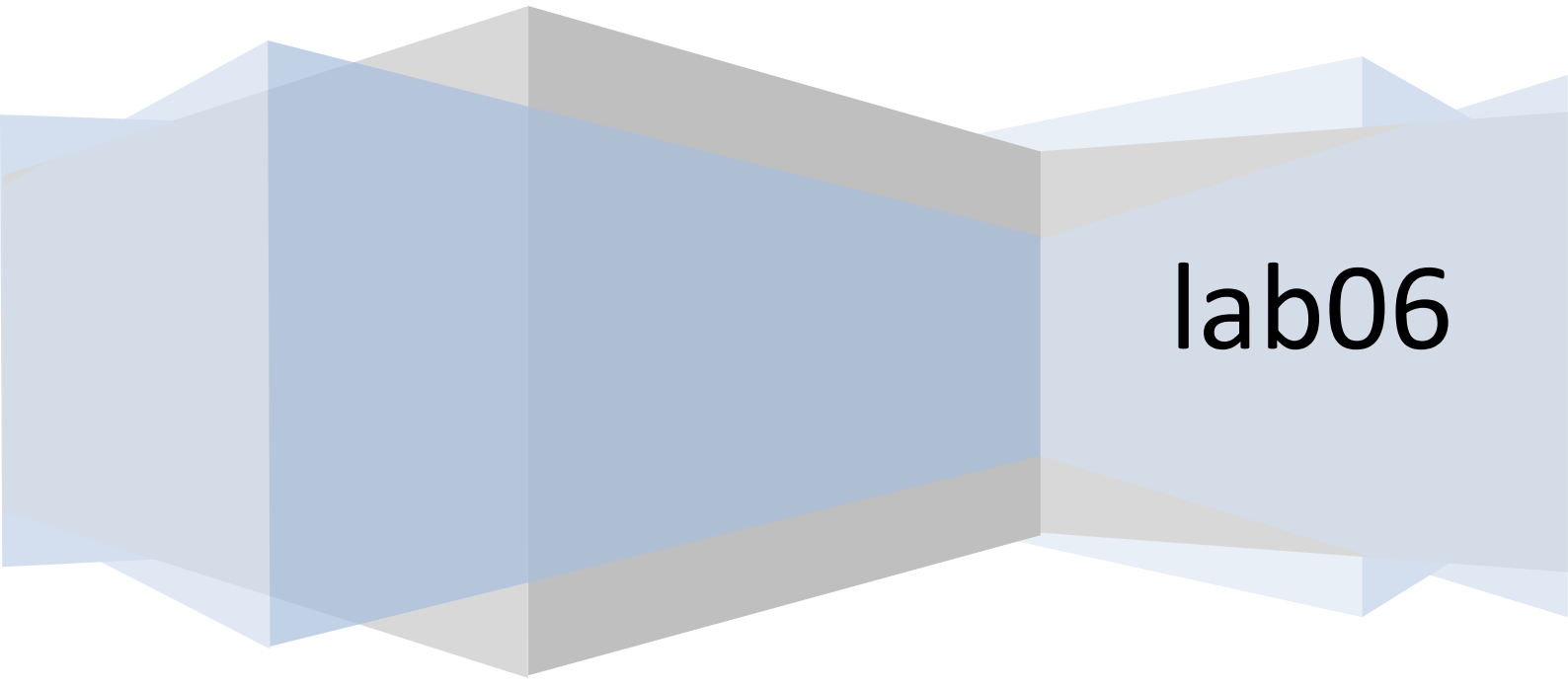
New York University

Real Time Digital Signal Processing Lab

lab06

Drumil Mahajan, dm3804



Problem 1.

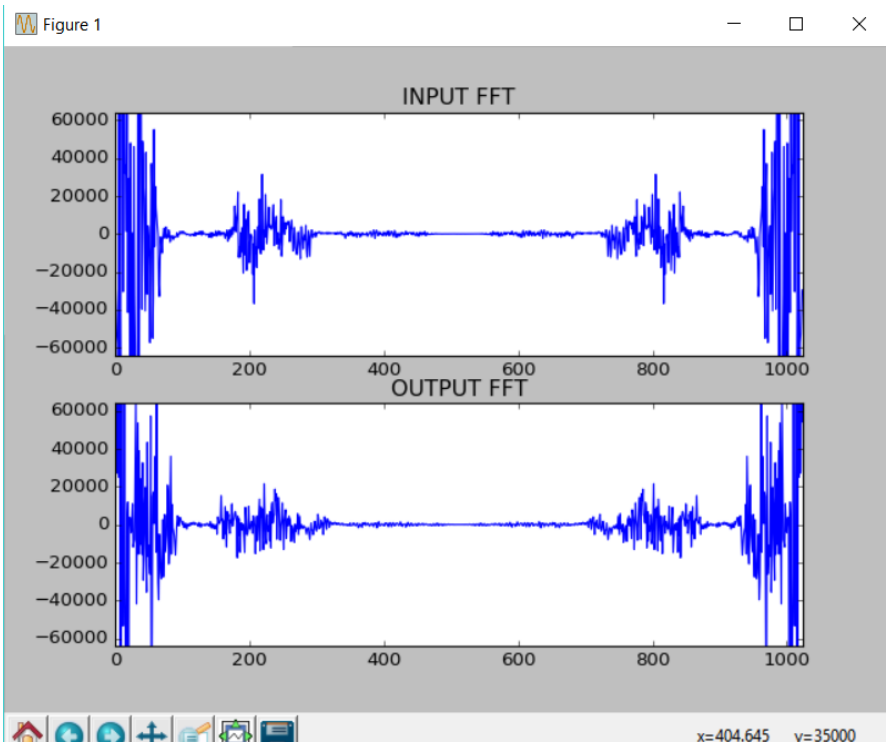
Folder: **Q1**

No written comments asked. The code is well commented for ease of reading.

Problem 2.

Folder: **Q2**

The figure below shows a snippet of output asked in question 2.



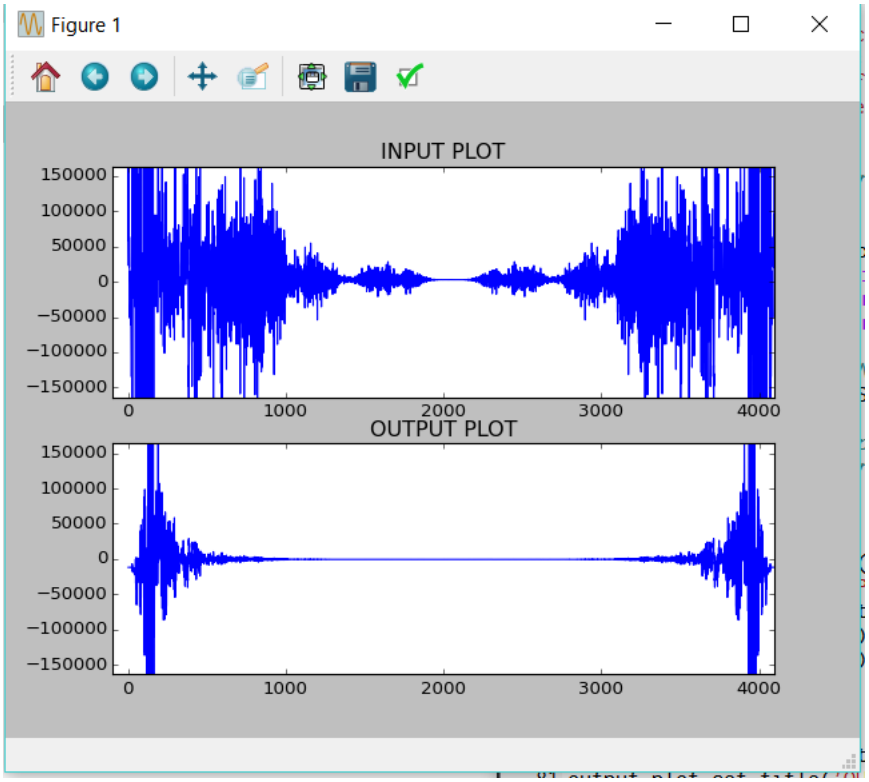
Relation between the two spectra: The two spectra almost appear same but is shifted.
Is the shift clear: It is difficult to analyse shift in real time although the program does show some shift.

Problem 3.

Folder: **Q3**

No written comments asked: The filter implemented is a bandpass filter with frequency range 100 to 500. It can be seen that lower frequency components are lost.

This figure is a real time plot of frequency spectrum. The x lims are nothing but the sample number. It can be converted to frequency by dividing by Fs.



Problem 3.3

Folder: **Q3.3**

No written comments asked.
The input is filtered through two filters. One filter is implemented mathematically in python and one filter is used from scipy.signal.

Both filters are given same filter coefficients and hence expected output must be same.

Filter implementations:

Filter **without** using scipy.signal

```
for n in range(0, BLOCKSIZE):

    # Set input to difference equation
    x0 = input_tuple[n]

    # Difference equation
    y0 = b0*x0 + b2*x2 + b4*x4 - a1*y1 - a2*y2 - a3*y3 - a4*y4

    # Delays
    x4 = x3
    x3 = x2
    x2 = x1
    x1 = x0
    y4 = y3
    y3 = y2
    y2 = y1
    y1 = y0

    x0_arr[n] = x0
    #print y0
    # Writing on the left channel values of the output_block
    output_block[2*n] = clip16(y0)
```

Filter using scipy.signal : It can be seen that I used the delay values so that I don't see transient effects.

```
# Filtering using lfilter
output_scipy, zf = signal.lfilter(b_array, a_array, x0_arr, axis = -1,  zi = zf)
```

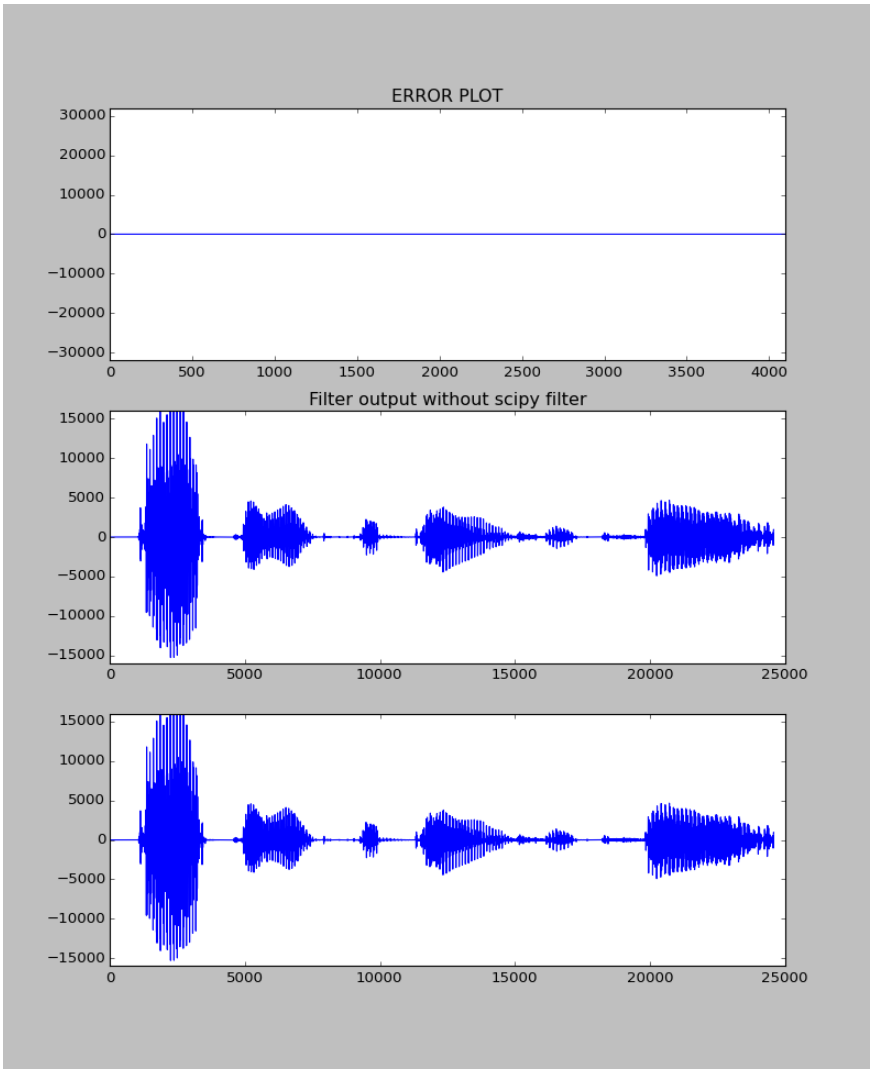
The code is well commented and can be easily read.

The figure below shows the snippet of the output. **PLEASE MAKE THE PLOT FULL SCREEN TO SEE THE TITLES PROPERLY.**

As expected, the error plot is always zero as both the outputs are same and hence the difference between them is zero.

After the signal is played, the complete output signals are also plotted from both the filters and it can be seen that they are same.

The ERROR signal is plotted in real time and shows zero throughout.



Problem 4:

Folder: **Q4**

It is given that x(t) = sin(Ø(t))
Also it is mentioned that the instantaneous frequency is given by

f-inst = (1/2pi)(Ø'(t)) , and for the given question f instantaneous increase from 200 to 500 in 5 seconds interval.

Writing the equation of straight line which whose slope is 300/5 and y intercept is 200, we get the equation for f-inst as 60t + 200.

Integrating with time and multiplying by 2pi gives us Ø(t) as 2pi(3t^2 + 200t)

To produce the signal in Python, we need to convert time into samples using equation t = n*Fs

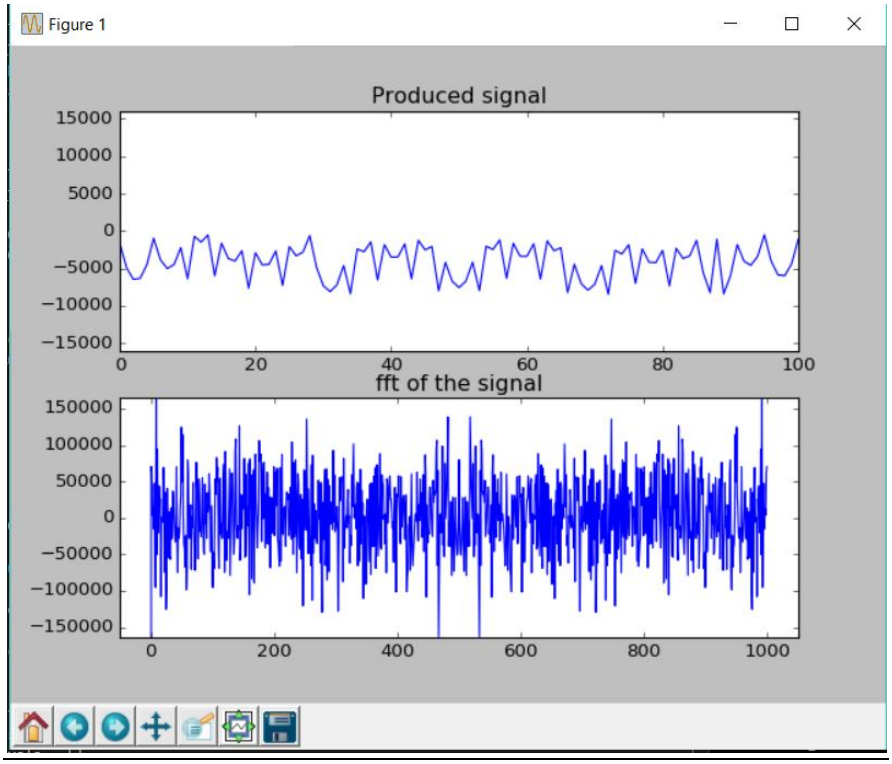
Therefore our equation in terms of samples becomes Fs*2pi(3t^2 + 200t).

This sine wave is produced in the program signal.py.

The figure shows the signal implementation in the python code. It is played for five seconds.

```
for n in range(0, 5*Fs):    # 5 second duration
    output_string = pack('h' , maxAmp * sin(Fs*2*pi*((30*(n**2))+(200*n))))
    #output_string = pack('h' , maxAmp * sin(n*2*pi*200/Fs) )
    stream.write(output_string) # left
```

The output can be seen the snapshot below.



Problem 5:

Folder: **No Folder for Question 5**

The difference equations are:

play_feedbackdelay1_test01.py :

y(n) = Gdp*x(n) + Gfb*y(n-delay_samples) + Gff*x(n-delay_samples)

play_feedbackdelay1.py

y(n) = x(n) + Gfb*y(n-5)