1. Design a Low Pass Digital Filter by Windowing (M=3, rectangular window)

1a) Filter coefficients are

Coeffs = [0.8, 0.1015, 0.1611, 0.2457, 0.3363, 0.4131, 0.4589, 0.4635, 0.4251, 0.3505, 0.2530, 0.1493, 0.0554, -0.0167, -0.0671, -0.0774, -0.0727, -0.0559, -0.0364, -0.0219, -0.158];

1b)

$$H(e^{j\omega}) = \sum_{n=0}^{3}\left[0.54 - 0.46\cos\left(\frac{2\pi n}{M-1}\right)\right] \times \frac{\sin(\omega_c n)}{\pi n}$$

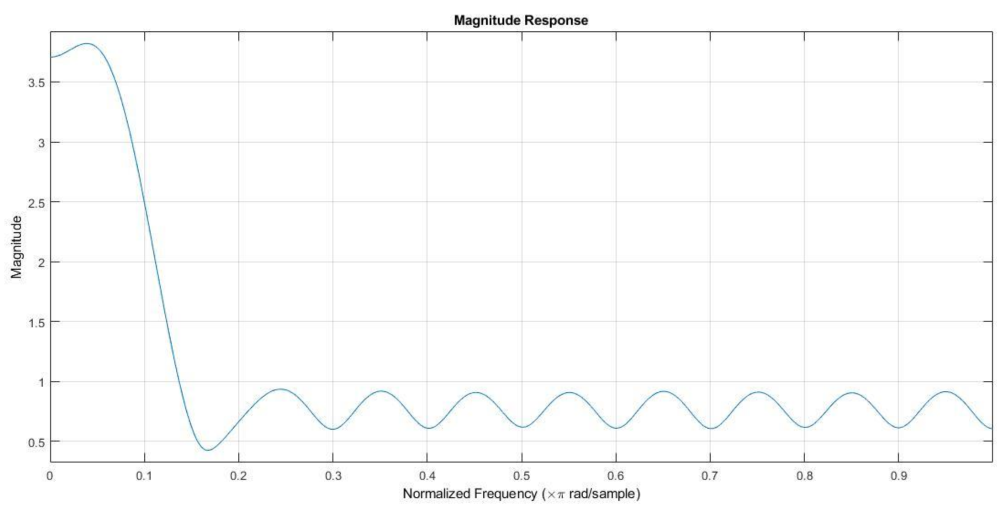
$$H(e^{j\omega}) = 0.54\sum_{k=0}^{n}\delta(\omega - 2\pi k) - \sum_{n=0}^{3}\left[0.46\cos\left(\frac{2\pi n}{M-1}\right)\right] \times \frac{\sin(\omega_c n)}{\pi n}$$

1c) First zero occurs at 0.8.

1d) Actual cut-off frequency is 1840 Hz from the frequency response

1e) %age error = $\frac{2000-1840}{2000} \times 100 = 0.0543\%$

1f) fvtool() frequency response plot from MatLab - linear magnitude scale



2. Implement a Low Pass Filter in Python (M=3, rectangular window)

```
#!/usr/bin/python

import sys
import time
import math

import base64
import random as random
```

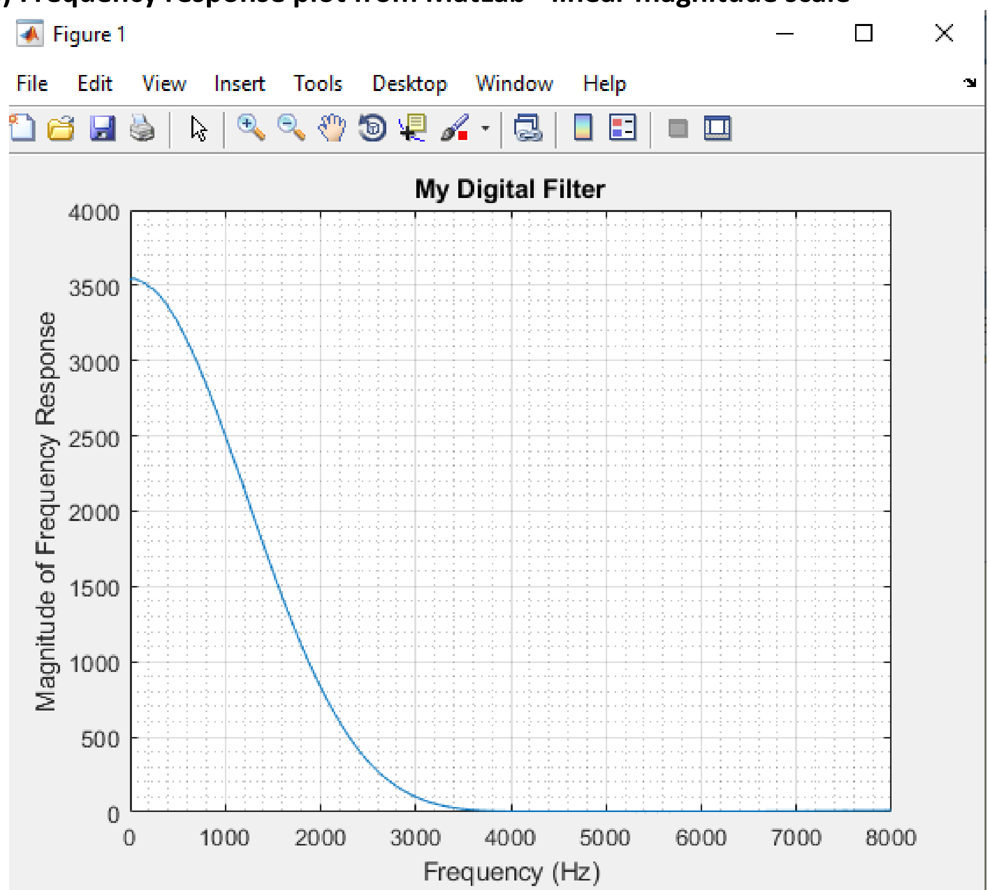
```
import datetime
import time

from cpe367_wav import cpe367_wav
from my_fifo import my_fifo



############################################
############################################
# define routine for implementing a digital filter
def process_wav(fpath_wav_in,fpath_wav_out):
	"""
	: this example implements a very useful system:  y[n] = x[n]
	: input and output is accomplished via WAV files
	: return: True or False
	"""

	# construct objects for reading/writing WAV files
	#  assign each object a name, to facilitate status and error reporting
	wav_in = cpe367_wav('wav_in',fpath_wav_in)
	wav_out = cpe367_wav('wav_out',fpath_wav_out)

	# open wave input file
	ostat = wav_in.open_wav_in()
	if ostat == False:
		print('Cant open wav file for reading')
		return False

	# setup configuration for output WAV
	# num_channels = 2
	# sample_width_8_16_bits = 16
	# sample_rate_hz = 16000
	#
wav_out.set_wav_out_configuration(num_channels,sample_width_8_16_bits,sample_rate_hz)

	# configure wave output file, mimicking parameters of input wave (sample rate...)
	wav_out.copy_wav_out_configuration(wav_in)

	# open WAV output file
	ostat = wav_out.open_wav_out()
	if ostat == False:
		print('Cant open wav file for writing')
		return False

	# students - allocate your fifo, with an appropriate length (M)
	M = 3
```

```
fifo = my_fifo(M)

# students - allocate filter coefficients, length (M)
# students - these are not the correct filter coefficients
bk_list = 1/M#[1/M, 1/M, 1/M]

# process entire input signal
xin = 0
while xin != None:

    # read next sample (assumes mono WAV file)
    #  returns None when file is exhausted
    xin = wav_in.read_wav()
    if xin == None: break


    ####################################################################
    ####################################################################
    # students - go to work!

    # update history with most recent input
    fifo.update(xin)

    # evaluate your difference equation
    yout = 0
    h_ideal = 0
    W_hamm = 0
    fc=2000
    for k in range(M):

        # use your fifo to access recent inputs when evaluating your diff eq
        # y[n] = b[k] * x[n-k]
        h_ideal = math.sin(2*math.pi*fc*k)/math.pi*k
        W_hamm = 0.54-(0.46*math.cos((2*math.pi*k)/(M-1)))
        print(W_hamm*h_ideal)
        yout += W_hamm*h_ideal * fifo.get(k)

    # students - well done!
    ####################################################################
    ####################################################################


    # convert to signed int
    yout = int(round(yout))

    # output current sample
    ostat = wav_out.write_wav(yout)
    if ostat == False: break
```

```
        # close input and output files
        #  important to close output file - header is updated (with proper file size)
        wav_in.close_wav()
        wav_out.close_wav()

        return True
############################################
############################################
# define main program
def main():

        # check python version!
        major_version = int(sys.version[0])
        if major_version < 3:
                print('Sorry! must be run using python3.')
                print('Current version: ')
                print(sys.version)
                return False

        # grab file names
        fpath_wav_in = 'in_noise.wav'
        fpath_wav_out = 'out_noise.wav'



        ############################################
        ############################################
        # test signal history
        #  feel free to comment this out, after verifying

        # allocate history
        M = 3
        fifo = my_fifo(M)

        # add some values to history
        fifo.update(1)
        fifo.update(2)
        fifo.update(3)
        fifo.update(4)

        # print out history in order from most recent to oldest
        print('signal history - test')
        for k in range(M):
                print('hist['+str(k)+']='+str(fifo.get(k)))

        ############################################
        ############################################
```

```
        # let's do it!
        return process_wav(fpath_wav_in,fpath_wav_out)


################################################
################################################
# call main function
if __name__ == '__main__':

        main()
        quit()
```

2a) Frequency response plot from MatLab - linear magnitude scale



3. Design a Bandpass Digital Filter and Implement in Python (M=21, Hamming window

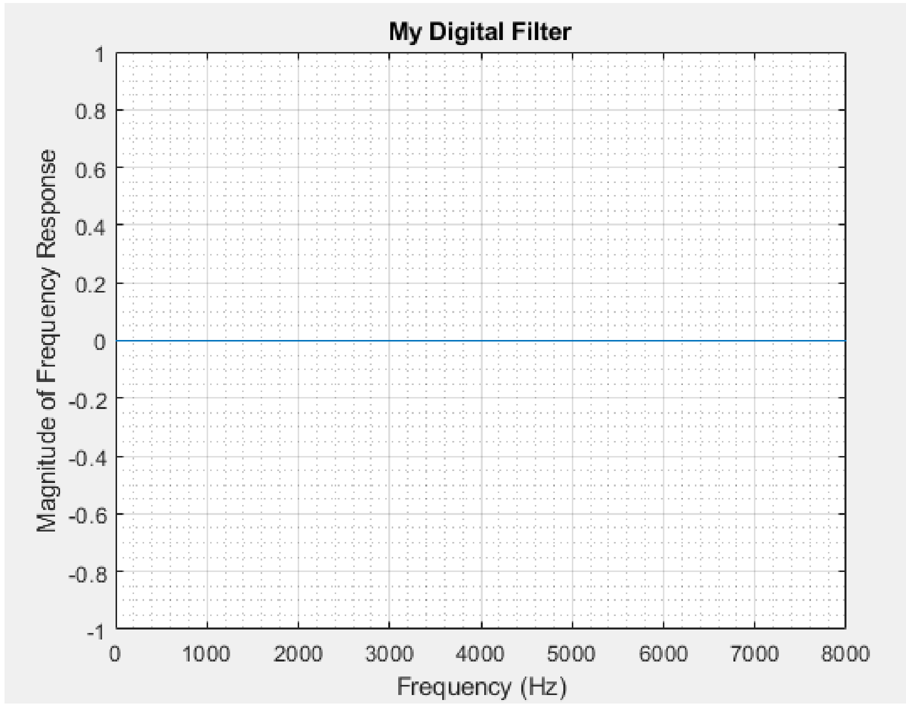
3a) What are the coefficients of your bandpass filter

3538.778683 3545.872747 3547.401477 3541.588431 3535.44815 3530.302934
3528.008447 3515.172452 3514.507572 3498.417155

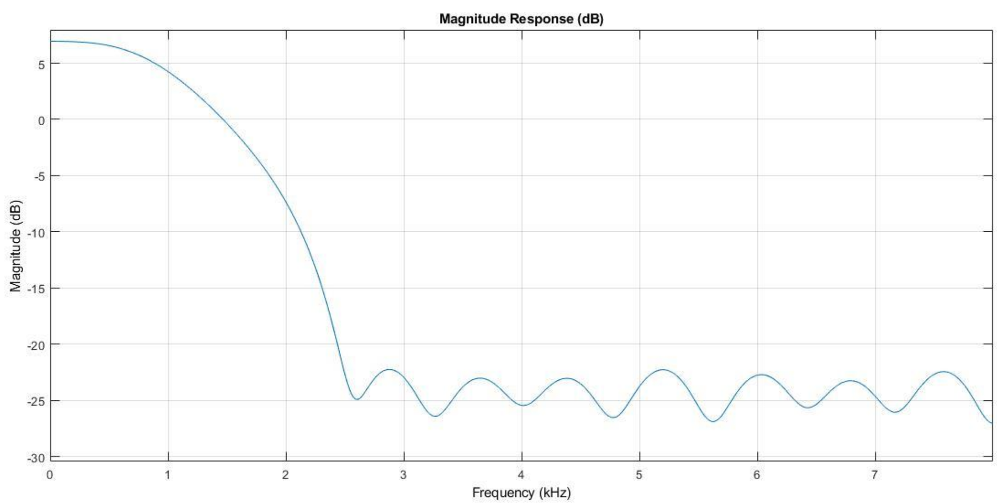
3b) Upload the portion of your Python code that computes the filter coefficients and then modulates the impulse response to yield the bandpass filter

```
##################################################################
        ##################################################################
                # students - go to work!
                # update history with most recent input
                fifo.update(xin)
                # evaluate your difference equation
                yout = 0
                h_ideal = 0

                W_hamm = 0
                fc=0
                for k in range(M):

                        # use your fifo to access recent inputs when evaluating your diff
eq
                        # y[n] = b[k] * x[n-k]
                        #yout += bk_list * fifo.get(k)
                        h_ideal = math.sin(2*math.pi*fc*k)/math.pi*k
                        W_hamm = 0.54-(0.46*math.cos((2*math.pi*k)/(M-1)))
                        yout = W_hamm*h_ideal

                # students - well done!
        ##################################################################
        ##################################################################
```

3c) Upload Welch Periodogram from MatLab



4. Use MatLab's DSP Toolbox to Find an Optimized Filter (M=21)



4a) How much can the stopband attenuation be increased while maintaining a filter length of M=21 or less (dB)? (2) Keep Fp = 1kHz, Fst = 3.5kHz and increase Ast.

As we increase the dBs with the increment of 5 dB, it was observed the dBs can be increased upto 65 dB. Above this value, and the filter length increases than 21.

4b) How much can the stopband frequency be reduced while maintaining a filter length of M=21 or less (Hz)? (2) Keep Ast = 50, Fp = 1kHz, and decrease Fst

As we decrease the Fst with the decrement of 100 Hz, it was observed the it can be decreased upto 3100 Hz. Below this value, and the filter length increases than 21.