LAB CODE AND NAME - 19CLE284 SOFTWARE DEFINED RADIO LAB
EXPERIMENT NUMBER - 5
DATE - 30/05/2022

DESIGN AND ANALYZE FILTERS FOR GIVEN SPECIFICATIONS

* **AIM:**

Given the order, cutoff frequency, and filter type, analyze the filter response and magnitude response. Also, include finite word length effects in the analysis.

* **SOFTWARE REQUIRED:**

Spyder IDE - Python 3.9

* **CODE:**

```
import matplotlib.pyplot as plt # Provides an implicit way of plotting
import numpy as np # support for large, multi-dimensional arrays and matrices
import time # Handle time-related tasks

#Compute DFT coefficients using linear transformation method:-
def DFT(x, plot=name, round_off):

    start = time.time()

    # Compute W(N) 1D Array:-
    xl = cl = len(x)
    wn = []
    for i in range(xl):
        for j in range(cl):
            wn.append(np.exp(-2j * np.pi * i * j / len(x)))
```

```
# numpy.reshape() is used to give a new shape to an array without changing its data.

wn_multidim = np.reshape(wn, (r1, c1)) # An N*N W(N) matrix
r2 = len(x); c2=1
x_multidim = np.reshape(x, (r2, c2)) # An N*1 x(N) matrix

# compute X(N) = W(N) * x(N), an N*1 matrix
fourier_transform_multidim = [[0]*c2]*r1 # Null Multidimensional Array Declaration
fourier_transform_1_t = [] # Convert Multidimensional Array to 1D
for i in range(r1):
    for j in range(c2):
        fourier_transform_multidim[i][j] = 0
        for k in range(c1):
            fourier_transform_multidim[i][j] += wn_multidim[i][k] * float(x_multidim[k][j])
        fourier_transform_1_t.append(round(abs(fourier_transform_multidim[i][j]), round_off))

plt.xlabel("Frequency in Hertz")
plt.ylabel("A(f) in Volts")
plt.title("" + str(plot_name) + " in Frequency Domain")
plt.plot(np.arange(0, len(fourier_transform_1_t)), fourier_transform_1_t)
plt.grid(True)
plt.show()

end = time.time()
print("In The time taken for execution of " + str(plot_name) + " Design (in seconds) is :", end-start)
```

MAIN MENU

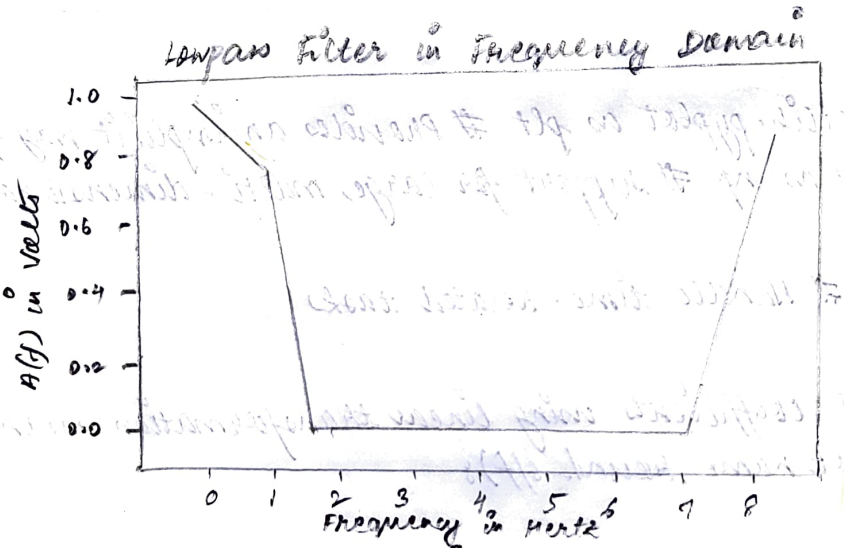
1. Lowpass Filter
2. Highpass Filter
3. Bandpass Filter
4. Bandstop Filter
5. Exit

Enter the number corresponding to the menu to implement the choice : 1

Enter the order of the filter : 9

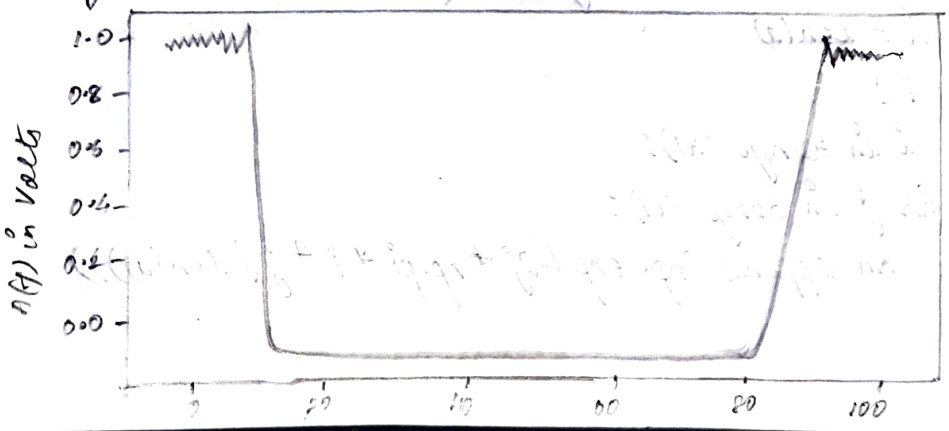
The truncation value for finite word analysis is taken to be 4

Enter the value of WC coefficient for the Lowpass Filter : 0.3



The time taken for execution of 'Lowpass Filter' Design (in seconds) is : 0.10024905204772949

Similarly, for order of the filter : 100



```
def plot_desired_impulse_response(hd_n, plot_name):

    plt.xlabel("Frequency in Hertz")
    plt.ylabel("A(f) in Volts")
    plt.title("Desired Impulse Response of " + str(plot_name))
    plt.stem(np.arange(0, len(hd_n)), hd_n)
    plt.grid(True)
    plt.show()

def lowpass_filter(N, round_off):

    title = "Lowpass Filter"
    w_c = float(input("Enter the value of wC coefficient for the "
+ str(title) + ": "))
    w_c = w_c * np.pi

    hd_n = [] # Desired Frequency Response - NULL Array Declaration
    for n in range(-N//2+1, N//2+1, 1):
        if n==0:
            hd_n.append((w_c * np.sinc(w_c * n)) / np.pi) # Geometric
Interpretation
        else:
            hd_n.append(np.sin(w_c*n) / (np.pi * n)) # computation
Purposes.

    plot_desired_impulse_response(hd_n, title)
    DFT(hd_n, title, round_off

def highpass_filter(N, round_off):

    title = "Highpass Filter"
    w_c = float(input("Enter the value of wC coefficient for the " +
str(title) + ": "))
```

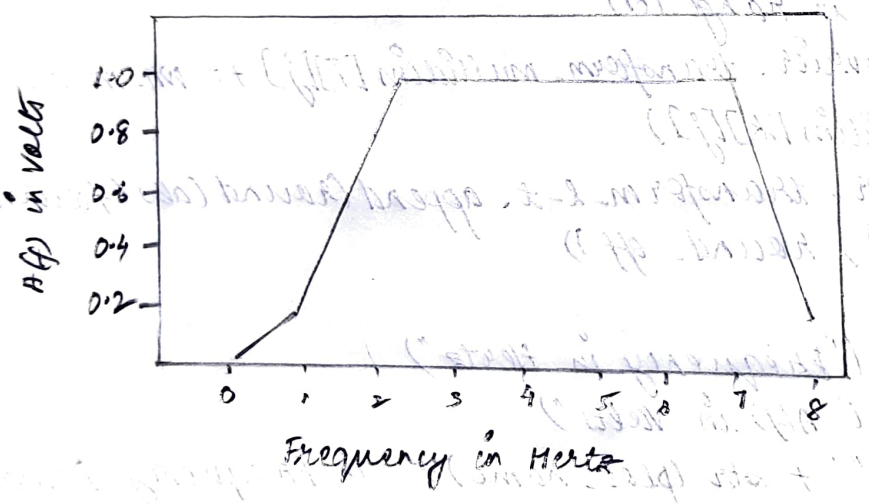
MAIN MENU

1. Lowpass Filter
2. Highpass Filter
3. Bandpass Filter
4. Bandstop Filter
5. Exit

Enter the number corresponding to the menu to implement the choice : 2

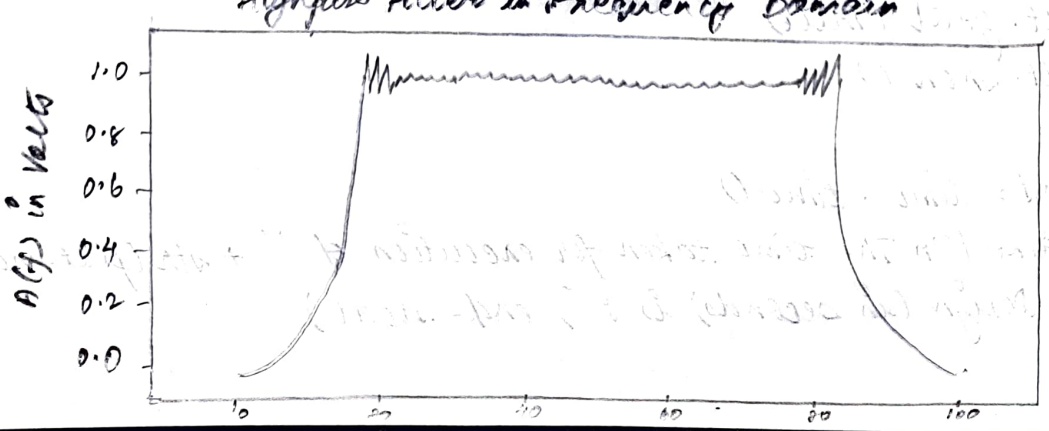
Enter the order of the filter : 9
The truncation value for finite word length analysis is taken to be 4.
Enter the value of wc coefficient for the Highpass Filter : 0.3

Highpass filter in Frequency Domain



Similarly, for order of the filter : 100

Highpass Filter in Frequency Domain



```
    w_c = w_c * np.pi

    hd_n = [] # Desired Frequency Response - NULL Array Declaration
    for n in range (-N//2+1, N//2+1, 1):
        if n==0:
            hd_n.append ((np.sinc ((np.pi * n)) - ((w_c * np.sinc (w_c * n)) / (np.pi))) # Geometric Interpretation
        else:
            hd_n.append ((np.sin ((np.pi) * n)) - (np.sin (w_c * n)) / ((np.pi) * n)) # computation purposes

    plot_desired_impulse_response (hd_n, title)
    DFT (hd_n, title, round_off)

def bandpass_filter (N, round_off):

    title = "Bandpass Filter"
    w_h = float (input ("Enter the value of wH coefficient for the " + str (title) + " : "))
    w_h = w_h * np.pi
    w_l = float (input ("Enter the value of wL coefficient for the " + str (title) + " : "))
    w_l = w_l * np.pi

    hd_n = [] # Desired Frequency Response - NULL Array Declaration
    for n in range (-N//2 + 1, N//2+1, 1):
        if n==0:
            hd_n.append ((w_h * np.sinc (w_h * n) / np.pi) - (w_l * np.sinc (w_l * n) / np.pi)) # Geometric Interpretation
        else:
```

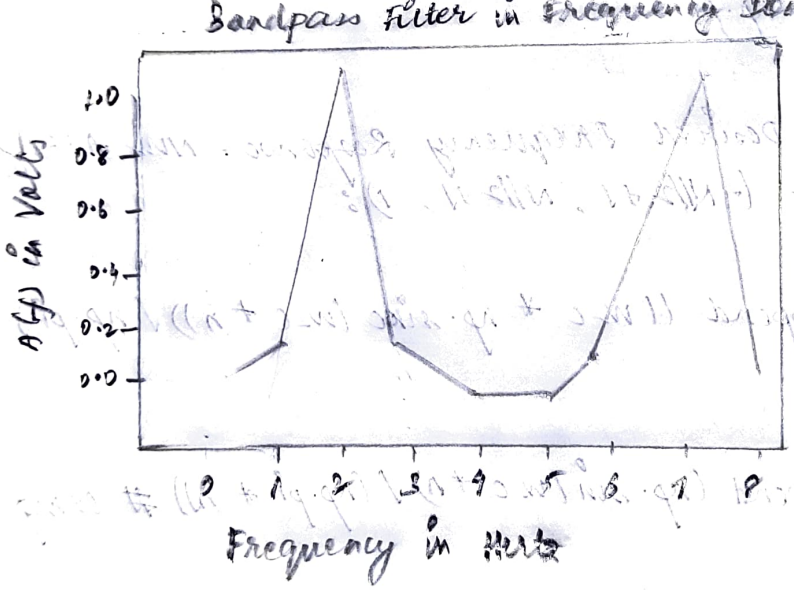
MAIN MENU

1. Lowpass filter
2. Highpass filter
3. Bandpass filter
4. Bandstop filter
5. Exit

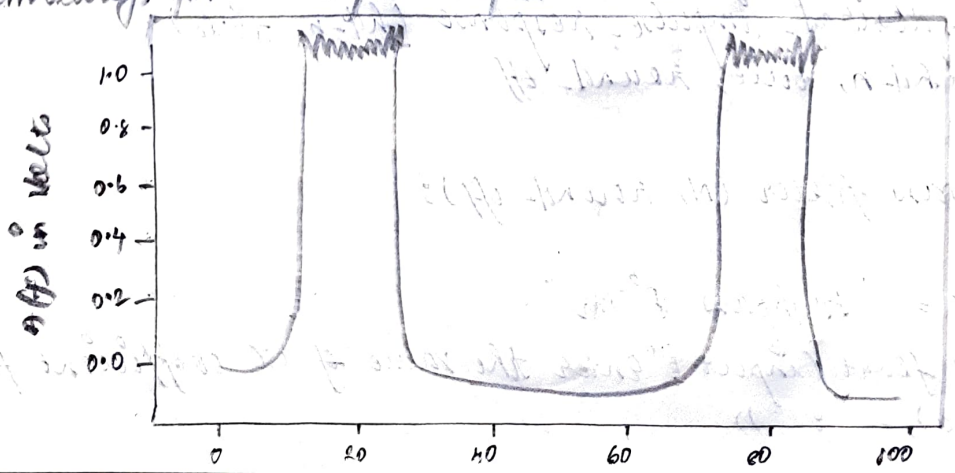
Enter the number corresponding to the menu to implement the choice : 3

Enter the order of the filter: 9
The truncation value for finite word length analysis is taken to be 4.

Enter the value of w1 coefficient for the Bandpass filter : 0.6
Enter the value of w2 coefficient for the Bandpass filter: 0.3



Similarly, for order of the filter : 100



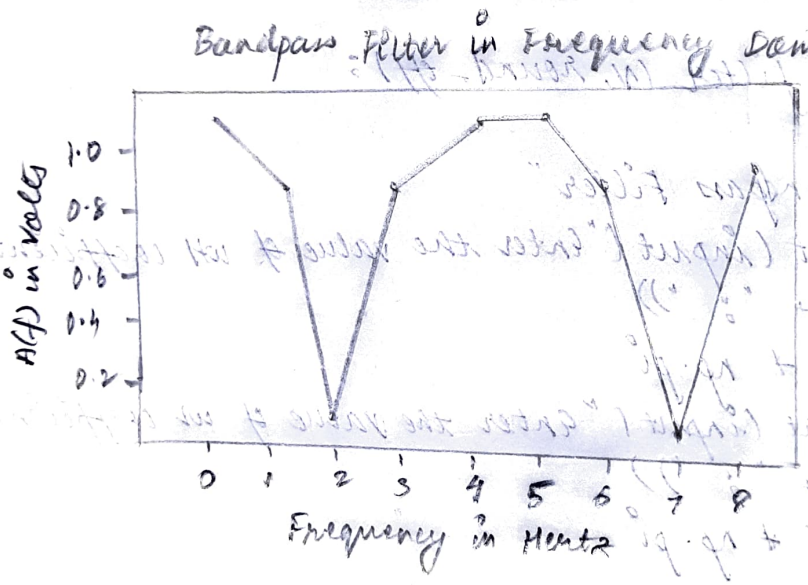
```
            hd_n.append(((np.sin(w_h*n))-(np.sin(w_l*n)))/(np.pi*n)) # Computation Purposes
    plot_desired_impulse_response(hd_n,title)
    DFT(hd_n,title,round_off)

def bandstop_filter(N,round_off):

    title="Bandstop Filter"
    w_h=float(input("Enter the value of wH coefficient for the "+str(title)+": "))
    w_h=w_h*np.pi
    w_l=float(input("Enter the value of wL coefficient for the "+str(title)+": "))
    w_l=w_l*np.pi

    hd_n=[] # Desired Frequency Response - NULL Array Declaration
    for n in range(-N//2+1,N//2+1,1):
        if n==0:
            hd_n.append((np.sinc(np.pi*n))+(w_l/np.pi)-(w_h/np.pi)) # Geometric Interpretation
        else:
            hd_n.append(((np.sin(np.pi*n))+(np.sin(w_l*n))-(np.sin(w_h*n)))/(np.pi*n)) # Computation Purposes

    plot_desired_impulse_response(hd_n,title)
    DFT(hd_n,title,round_off)

# Driver Code: main(); Execution starts here.
```

MAIN MENU

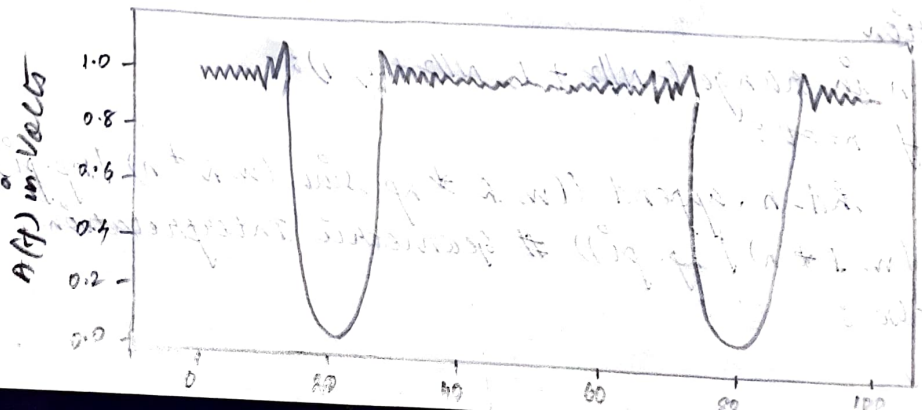
1. Lowpass filter
2. Highpass filter
3. Bandpass filter
4. Bandstop filter
5. Exit

Enter the number corresponding to the menu to implement the choice: 4

Enter the order of the filter: 9
The truncation value for finite word length analysis is taken to be 4.
Enter the value of w1 coefficient for the Bandstop filter: 0.6
Enter the value of w2 coefficient for the Bandstop filter: 0.3



Similarly for order of the filter: 100



```
while True: # This simulates a DO Loop

    print("\n")
    heading = "MAIN MENU"
    print('{:.>53}'.format('\u0332'.join(heading.center(50))))

    choice = input("
        1. Lowpass Filter\n    2. Highpass Filter\n    3. Bandpass Filter\n    4. Bandstop Filter\n    5. Exit\n Enter the number corresponding to the menu to implement the choice: ")
# Menu Driven Implementation

    # str() returns the string version of the variable 'choice'
    if (choice == str(1) or choice == str(2) or choice == str(3) or choice == str(4)):

        N= int(input("\n Enter the order of the filter: "))
        print("The truncation value for finite word length analysis is taken to be 4.")
        round_off = 4

        if choice == str(1):
            lowpass_filter(N, round_off)
        elif choice == str(2):
            highpass_filter(N, round_off)
        elif choice == str(3):
            bandpass_filter(N, round_off)
        elif choice == str(4):
            bandstop_filter(N, round_off)

    elif choice == str(5):
        break # Exit loop
```

```
else:
    print("Error: Invalid Input! Please try again.")
```

* **RESULT:**

For the given order, cutoff frequency, and filter type, analyzed the filter response and magnitude response. Also, included finite word length effects in the analysis. All the simulation results were verified successfully.