

LAB TITLE AND CODE - SIGNAL PROCESSING LAB 19CLE281

EXPERIMENT NUMBER - 10

DATE - 28/12/2021

DESIGN OF FIR FILTER* AIM :

For a given frequency response $H(e^{j\omega})$, design a Finite Impulse Response (FIR) low pass filter using rectangular and Hanning window techniques and determine the impulse and frequency response.

* SOFTWARE REQUIRED :

Synder IDE - Anaconda 3 2021.11 (Python 3.9.7 64-bit)
Publisher - Anaconda, Inc.

* THEORY :

FIR filters are filters having a transfer function of a polynomial in z and is an all-zero filter in the sense that the zeroes in the z -plane determine the frequency response magnitude characteristic. The z transform of a N -point FIR filter is given by -

$$H(z) = \sum_{n=0}^{N-1} h(n) z^{-n}$$

FIR filters are particularly useful for applications where exact phase response is required. The FIR filter is generally implemented in a non-recursive way which guarantees a stable filter.

→ RECTANGULAR WINDOW METHOD:-

In this method, from the desired frequency response specification $H_d(w)$, corresponding unit sample response $h_d(n)$ is determined using the following relation -

$$h_d(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} H_d(w) e^{jwn} dw$$

where, $H_d(w) = \sum_{n=-\infty}^{\infty} h_d(n) e^{-jwn}$

In general, unit sample response $h_d(n)$ obtained from the above relation is infinite in duration, so it must be truncated at some point say $n=m-1$ to yield an FIR filter of length m (i.e. 0 to $m-1$). This truncation of $h_d(n)$ to length $m-1$ is same as multiplying $h_d(n)$ by the rectangular window defined as -

$$w(n) = \begin{cases} 1, & 0 \leq n \leq m-1 \\ 0, & \text{otherwise} \end{cases}$$

Thus, the unit sample response of the FIR filter becomes -

$$\begin{aligned} h(n) &= h_d(n) w(n) \\ &= \begin{cases} h_d(n), & 0 \leq n \leq m-1 \\ 0, & \text{otherwise} \end{cases} \end{aligned}$$

Now, the multiplication of the window function $w(n)$ with $h_d(n)$ is equivalent to convolution of $H_d(w)$ with $w(w)$, where $w(w)$ is the frequency domain representation of the window function -

$$W_d(w) = \sum_{n=0}^{m-1} w(n) e^{-jwn}$$

③

Thus, the convolution of $H_d(w)$ with $w(w)$ yields the frequency response of the truncated FIR filter -

$$H(w) = \frac{1}{2\pi} \int_{-\pi}^{\pi} H_d(v) w(w-v) dv$$

The frequency response can also be obtained using the following relation -

$$H(w) = \sum_{n=0}^{M-1} R(n) e^{-jwn}$$

HANNING WINDOW METHOD:-

Similar to rectangular window,

$$w[n] = \sum_{k=0}^K (-1)^k a_k \cos\left(\frac{2\pi k n}{N}\right), \quad 0 \leq n \leq N.$$

In most cases, including the given question, all coefficients $a_k \geq 0$. These windows have only $2K+1$ non-zero N -point DFT coefficients.

The customary cosine-sin windows for case $k=1$ have the form -

$$w[n] = a_0 - \underbrace{(1-a_0)}_{a_1} \cdot \cos\left(\frac{2\pi n}{N}\right), \quad 0 \leq n \leq N$$

which is easily (and often) confused with its zero-phase version -

$$w_0(n) = w\left(n + \frac{N}{2}\right)$$

$$= a_0 + a_1 \cdot \cos\left(\frac{2\pi n}{N}\right), \quad -\frac{N}{2} \leq n \leq \frac{N}{2}$$

Setting $a_0 = 0.5$ produces a Hann window -

$$w[n] = 0.5 \left[1 - \cos\left(\frac{2\pi n}{N}\right) \right] = \sin^2\left(\frac{\pi n}{N}\right)$$

* GRAPH PLOTTING ALGORITHM:

The following steps were followed-

- ① Define the x-axis and corresponding y-axis as lists.
- ② Plot them on canvas using `plot()` function.
- ③ Give a name to x-axis and y-axis using `xlabel()` and `ylabel()` functions.
- ④ Give a title to your plot using the `title()` function.
- ⑤ Finally, to view your plot, we use the `show()` function.
- ⑥ To set the visibility of the grid inside the figure to on, we use the `grid()` function.

* THEORETICAL CALCULATION :

Given -

Number of coefficients, $N = 9$ (also, indicated by m)

Frequency response, $H(e^{jw}) = \begin{cases} 1, & -2\pi/5 \leq w \leq 2\pi/5 \\ 0, & \text{otherwise} \end{cases}$

Desired filter coefficient ($h_d[n]$) is obtained by taking Inverse Discrete Time Fourier Transform (IDTFT) on $H(e^{jw})$ and yields $h_d[n] = \sin\left(\frac{2\pi n}{5}\right)$

$$\Rightarrow \text{for } n=0, h_d[0] = \frac{1}{2\pi} \int_{-\pi}^{\pi} 1 \times e^{j0w} dw = 0.4$$

Similarly, for $n=1, h_d[1] = 0.3027$

for $n=2, h_d[2] = 0.0935$

for $n=3, h_d[3] = -0.06236$

for $n=4, h_d[4] = -0.0757$

for $n=5, h_d[5] = -0.06236$

for $n=6, h_d[6] = 0.0935$

⑤

$$\text{for } n=7, h_2[7] = 0.3027$$

$$\text{for } n=8, R_2[8] = 0.4$$

① RECTANGULAR WINDOW TECHNIQUE:-

Expected filter coefficients ($R[n]$) on multiplying with rectangular window function ($w[n]$) yields the following -

$$w[n] = \begin{cases} 1, & 0 \leq n \leq M-1 = 0 \leq n \leq 8 \\ 0, & \text{otherwise} \end{cases}$$

$$h[n] = h_2[n] \times w[n]$$

$$= \begin{cases} h_2[n], & 0 \leq n \leq 8 \\ 0, & \text{otherwise} \end{cases}$$

Determine the frequency response $H(e^{j\omega})$ by direct substitution of the expression (even symmetry and length is odd integer) -

$$H(\omega) = H_2(\omega) e^{-j\omega \left(\frac{M-1}{2}\right)} = H_2(\omega) e^{-j\omega \left(\frac{8}{2}\right)},$$

where $H_2(\omega)$ is a real function of ω and can be expressed as -

$$H_2(\omega) = 2 \left(\frac{M-1}{2} \right) + 2 \sum_{n=0}^{\frac{(M-1)}{2}} h(n) \cos \left(\frac{m-1}{2} - n \right); \text{ when } m \text{ is odd}$$

$$\text{and } H_2(\omega) = 2 \sum_{n=0}^{\frac{(M-1)}{2}-1} a(n) \cos \omega \left(\frac{m-1}{2} - n \right); \text{ when } m \text{ is even}$$

Taking ω values from 0 to 180° with a difference of 10° at regular intervals, we obtain 18 values corresponding for $H(\omega)$, which is converted to decibels by -

$$\text{Decibel Representation} = 20 \times \log_{10} |H(\omega)|$$

(6)

→ HANNING WINDOW TECHNIQUE:-

Expected filter coefficients ($h[n]$) on multiplying with Hanning window function ($w[n]$) yields the following -

$$w[n] = 0.5 \left(1 - \cos\left(\frac{\pi n}{m-1}\right) \right) = 0.5 \left(1 - \cos\left(\frac{\pi n}{N_f}\right) \right)$$

$$h[n] = h_r[n] \times w[n]$$

$$h[n] = \begin{cases} 0.00, & \text{for } n=0 \\ 0.0443, & \text{for } n=1 \\ 0.0468, & \text{for } n=2 \\ -0.0532, & \text{for } n=3 \\ -0.0757, & \text{for } n=4 \\ -0.0532, & \text{for } n=5 \\ 0.068, & \text{for } n=6 \\ 0.0443, & \text{for } n=7 \\ 0.00, & \text{for } n=8 \\ 0, & \text{otherwise} \end{cases}$$

Determine the frequency response, similar to rectangular window technique.

* PROGRAM WITH COMMENTS :-

- 1 import matplotlib.pyplot as plt # Provides an implicit way of plotting
- 2 import numpy as np # Support for large, multi-dimensional arrays and matrices
- 3 from scipy import integrate # Compute a definite integral
- 4
- 5 N = int(input("Enter the number of coefficients for FIR filter:"))
- 6 fd_w = int(input("Enter the frequency response value:"))

```

    7
    8 # Inverse discrete-time Fourier transform:
    9 Rd_n = []
    10 for n in range (N//2 + 1):
    11     expression = lambda w: Rd_w + np.exp (f1j * n * n)
    12     # Expression within the integral
    13     temp = integrate.quad (expression, (-2 + np.pi)/5,
    14                               (-2 + np.pi)/5)
    15     integral = temp[0] / (2 * np.pi) # quad() gives a tuple,
    16     integral_value[0] + constant[1]
    17     Rd_n.append (integral)
    18
    19 # Compute desired filter coefficients:
    20 for n in reversed (range (N//2)): # Backward iteration will start
    21     occurring from the rear
    22     symmetric = Rd_n[n]
    23     Rd_n.append (symmetric)
    24 print ("The desired filter coefficients a-d[n] is as follows:\n"
    25     f"{{}}.format (Rd_n))\n"
    26
    27 # Design FIR filter using rectangular window technique
    28 # (Impulse Response):
    29
    30 # Rectangular window function:
    31 wn_rectangular = []
    32 for n in range (N):
    33     if n < N:
    34         temp = 1
    35     else:
    36         temp = 0
    37     wn_rectangular.append (temp)

```

(8)

```

34 # Expected filter coefficients obtained using rectangular
35 window technique:
35 hn_rectangular = []
36 for n in range (N):
37     hn_rectangular.append (hd_n[n] + wn_rectangular[n])
38
39 # sketch expected filter coefficients obtained using
40 rectangular window techniques:
40 plt.xlabel ('n')
41 plt.ylabel ('magnitude of h[n]')
42 plt.title ("Impulse Response Using Rectangular Window
43 Technique")
43 plt.stem (np.arange (0, len (hn_rectangular)), hn_rectangular)
44 plt.grid (True)
45 plt.show ()
46 print ("The expected filter coefficients h[n] obtained using
47 rectangular window technique is as follows : In fig "
47 format (hn_rectangular))
48
48 # Design FIR filter using rectangular window technique
49 (Frequency Response):
49
50 hr_w_rectangular = [] # Real function of w
51 h_w_rectangular = [] # Frequency response (magnitude)
52 h_w_rectangular_decibels = [] # Frequency response (decibels)
53 index_rectangular = 0 # Index variable to iterate through
54 "hr_w_rectangular" list
54 x_axis_rectangular = [] # Define the angle axis
55
56 for degrees in range (0, 180, 10):
57
58     w_rectangular = degrees + (np.pi / 180)

```

①

59 sum_rectangular = 0

60 if ($\pi/2 \leq \theta$) :61 for n in range ($((N-3)/2) + 1$) :

62 summation = hn_rectangular[n] +

63 np.cos(w_rectangular) + ((N-1)/2) - n

64 sum_rectangular = sum_rectangular + summation

65 hr_w_rectangular.append (hn_rectangular[N-1]/2) +
(2 * sum_rectangular))

66

67 else :

68

69 for n in range ($(N/2) - 1$) :

70 summation = hn_rectangular[n] +

71 np.cos(w_rectangular) + ((N-1)/2) - n

72 sum_rectangular = sum_rectangular + summation

73 hr_w_rectangular.append (2 * sum_rectangular)

74

75 hr_w_rectangular.append (hr_w_rectangular[Index_rectangular] +
np.exp (-j * w_rectangular * ((N-1)/2)))

76

77 hr_w_rectangular_decibels.append (20 * np.log10 (abs (hr_w_rectangular[Index_rectangular])))

78

79 Index_rectangular = Index_rectangular + 1

80

81 x_axis_rectangular.append (degrees)

82

83 # sketch frequency response obtained using rectangular
window technique:

84 plt.xlabel ('w in degrees')

85 plt.ylabel ('magnitude of H[w]')

```

86 plt.title ("Frequency Response Using Rectangular Window
Technique")
87 plt.plot (x-axis-rectangular, h-w- rectangular- decibels)
88 plt.grid (True)
89 plt.show ()
90 print ("The frequency response H[w] obtained using,
rectangular window technique is as follows: In {} ,
format (h-w- rectangular _decibels) ")
91
92 # Design FIR filter using Hanning window technique
(Impulse Response):
93
94 # Hanning window function:
95 wh-hanning = []
96 for n in range (N):
97     wh-hanning.append (0.5 + (1 - np.cos ((2 * np.pi * n) /
(N-1))) )
98
99 # expected filter coefficients obtained using Hanning window
technique:
100 hn-hanning = []
101 for n in range (N):
102     hn-hanning.append (hd-n[n] + wh-hanning [n])
103
104 # sketch expected filter coefficients obtained using Hanning
window technique:
105 plt.xlabel ('n')
106 plt.ylabel ('magnitude of h[n]')
107 plt.title ("Impulse Response Using Hanning Window
Technique")
108 plt.stem (np.arange (0, len (hn-hanning)), hn-hanning)
109 plt.grid (True)
110 plt.show ()

```

111

111 print ("The expected filter coefficients $h[n]$ obtained using
Hanning window technique is as follows: In f").
format ('h[n]=hanning'))

112

113 # Design FIR filter using Hanning window technique
(Frequency Response):

114

115 hr_w_hanning = [] # Real function of w

116 h_w_hanning = [] # Frequency response in magnitude

117 h_w_hanning - decibels = [] # Frequency response (decibels)

118 index_hanning = 0 # Index variable to iterate through
'hr_w_hanning' list

119 x_axis_hanning = [] # Define the angle axis

120

121 for degrees in range (0, 190, 10):

122

123 w_hanning = degrees + (np.pi/180)

124 sum_hanning = 0

125

126 if (N%2 == 1):

127

128 for n in range (((N-3)/2) + 1):

129 summation = hr_w_hanning[n] + np.cos(w_hanning) *
(((N-1)/2) - n)

130 sum_hanning = sum_hanning + summation

131 hr_w_hanning.append(hr_w_hanning[((N-1)/2) + 1 +
sum_hanning])

132

133 else:

134

135 for n in range ((N-2) - 1):

136 summation = hr_w_hanning[n] + np.cos(w_hanning) +
(((N-1)/2) - n)

(P)

```

137 sum_hanning = sum_hanning + summation
138 h_w_hanning.append(2 * sum_hanning)
139
140 h_w_hanning.append(h_w_hanning[index_hanning] +
    np.exp(j * w_hanning * ((N-1)/2)))
141
142 h_w_hanning_decibels.append(20 + np.log10(abs(
    h_w_hanning[index_hanning])))
143
144 Index_hanning = Index_hanning + 1
145
146 x_axis_hanning.append(degrees)
147
148 # sketch frequency response obtained using Hanning window
technique:
149 plt.xlabel('w in degrees')
150 plt.ylabel('H[w] in decibels')
151 plt.title("Frequency Response using Rectangular Window
Technique")
152 plt.plot(x_axis_hanning, h_w_hanning_decibels)
153 plt.grid(True)
154 plt.show()
155 print("The frequency response H[w] obtained using
Hanning window technique is as follows: In f3",
format(h_w_hanning_decibels))
156

```

* **INFERENCES:**

Compute the impulse and frequency response for a FIR lowpass filter using rectangular and Hanning window techniques.

RESULTS
VERIFIED.

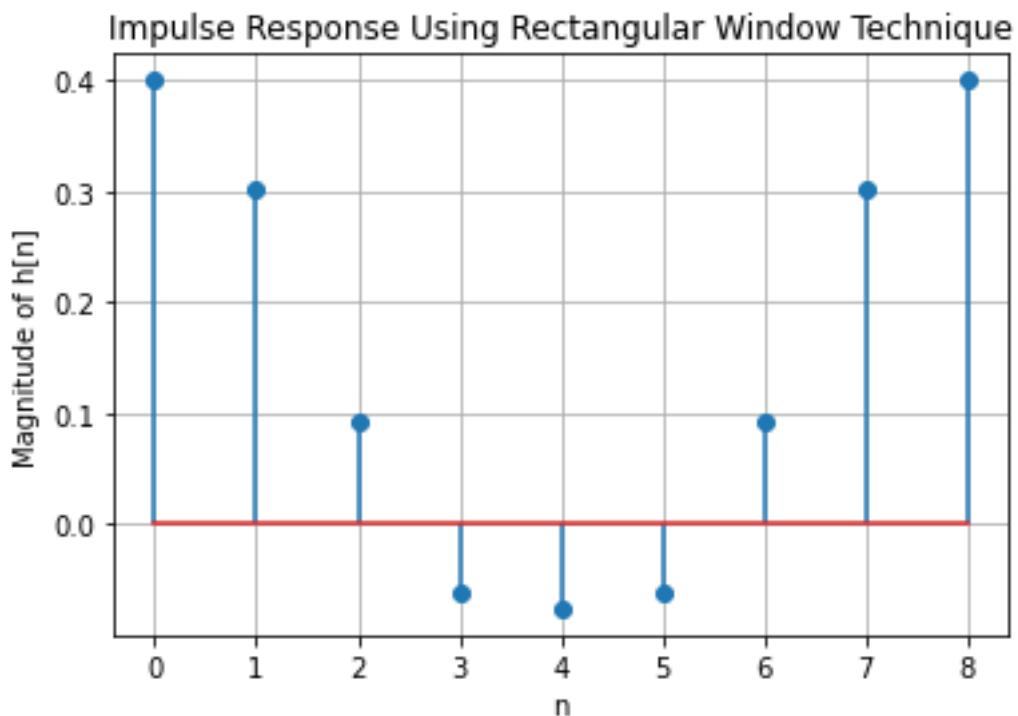
Impulse Response Using Rectangular Window Technique

The screenshot shows the Spyder Python IDE interface. On the left, the code editor displays 'Experiment 10.py' with Python script content. On the right, there are two plots: one titled 'Impulse Response Using Rectangular Window Technique' showing discrete-time Fourier transform magnitude, and another plot below it. Below the plots is a console window showing Python version information and command-line inputs.

```

1 import matplotlib.pyplot as plt # Provides an implicit way of plotting
2 import numpy as np # Support for large, multi-dimensional arrays and matrices
3 from scipy import integrate # Compute a definite integral
4
5 N = int(input("\nEnter the number of coefficients for FIR Filter: "))
6 hd_w = int(input("Enter the frequency response value: "))
7
8 # Inverse discrete-time Fourier transform:
9 hd_n = []
10 for n in range(N//2 + 1):
11     expression = lambda w: hd_w * np.exp(-1j * w * n) # Expression within the integral
12     temp = integrate.quad(expression, (-2*np.pi)/5, (2*np.pi)/5)
13     integral = temp[0] / (2*np.pi) # quad() gives a tuple, integral_value[0] + constant[1]
14     hd_n.append(integral)
15
16 # Compute desired filter coefficients:
17 for n in reversed(range(N//2)): # Backward iteration will start occurring from the rear
18     symmetric = hd_n[n]
19     hd_n.append(symmetric)
20
21 print("\nThe desired filter coefficients h_d[n] is as follows:\n {}".format(hd_n))
22
23 # Design FIR filter using rectangular window technique (Impulse Response):
24
25 # Rectangular window function:
26 wn_rectangular = []
27 for n in range(N):
28     if n < 0:
29         temp=1
30     else:
31         temp=0
32     wn_rectangular.append(temp)
33
34 # Expected filter coefficients obtained using rectangular window technique:
35 hn_rectangular = []
36 for n in range(N):
37     hn_rectangular.append(hd_n[n] * wn_rectangular[n])
38
39 # Sketch expected filter coefficients obtained using rectangular window technique:
40 plt.xlabel('n')
41 plt.ylabel('Magnitude of h[n]')

```



Step 1: Import library source files, NumPy, matplotlib.pyplot and SciPy.

Step 2: Enter the number of coefficients for the FIR filter along with the frequency response value.

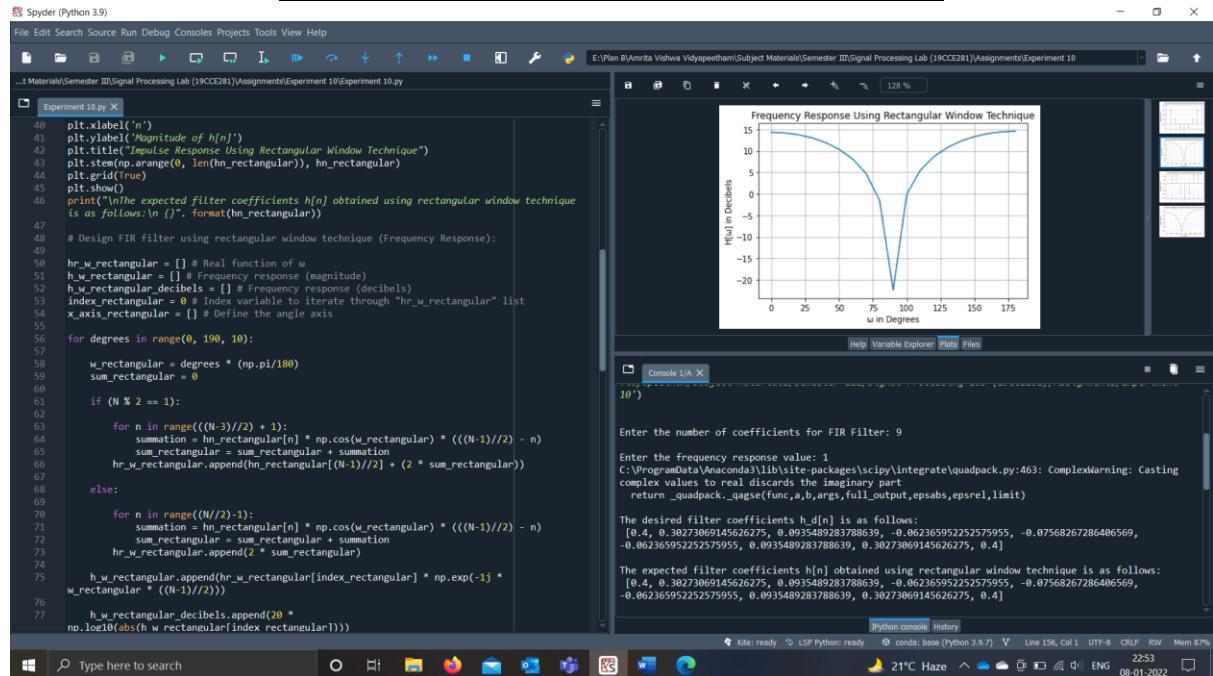
Step 3: Calculate inverse discrete-time Fourier transform and compute desired filter coefficients.

Step 4: Design FIR filter using rectangular window technique (Impulse Response).

Step 5: Compute the rectangular window function.

Step 6: Evaluate expected filter coefficients obtained using the rectangular window technique and sketch the same.

Frequency Response Using Rectangular Window Technique



The screenshot shows the Spyder Python IDE interface. On the left, the code editor displays a script named 'Experiment 10.py' containing Python code for calculating the frequency response of an FIR filter using a rectangular window. The code includes imports for numpy and matplotlib, initializes arrays for window coefficients and frequency response, and plots the magnitude in decibels against the angle axis in degrees. On the right, the IPython console shows the resulting plot titled 'Frequency Response Using Rectangular Window Technique' and the calculated filter coefficients.

```

40 plt.xlabel('ω')
41 plt.ylabel('Magnitude of h[n]')
42 plt.title("Impulse Response Using Rectangular Window Technique")
43 plt.stem(np.arange(0, len(hn_rectangular)), hn_rectangular)
44 plt.grid(True)
45 plt.show()
46 print("The expected filter coefficients h[n] obtained using rectangular window technique is as follows: In {} format(hn_rectangular)")
47
48 # Design FIR filter using rectangular window technique (Frequency Response):
49
50 hr_w_rectangular = [] # Real function of ω
51 h_w_rectangular = [] # Frequency response (magnitude)
52 h_w_rectangular_decibels = [] # Frequency response (decibels)
53 index_rectangular = 0 # Index variable to iterate through "hr_w_rectangular" list
54 x_axis_rectangular = [] # Define the angle axis
55
56 for degrees in range(0, 190, 10):
57
58     w_rectangular = degrees * (np.pi/180)
59     sum_rectangular = 0
60
61     if (N % 2 == 1):
62
63         for n in range((N-3)//2 + 1):
64             summation = hn_rectangular[n] * np.cos(w_rectangular) * (((N-1)//2) - n)
65             sum_rectangular = sum_rectangular + summation
66             hr_w_rectangular.append(hn_rectangular[(N-1)//2] + (2 * sum_rectangular))
67
68     else:
69
70         for n in range((N//2)-1):
71             summation = hn_rectangular[n] * np.cos(w_rectangular) * (((N-1)//2) - n)
72             sum_rectangular = sum_rectangular + summation
73             hr_w_rectangular.append(2 * sum_rectangular)
74
75     h_w_rectangular.append(hr_w_rectangular[index_rectangular] * np.exp(-1j *
76     w_rectangular * ((N-1)//2)))
77
78     h_w_rectangular_decibels.append(20 *
79     np.log10(np.abs(h_rectangular[index_rectangular])))

```

Frequency Response Using Rectangular Window Technique

H[ω] in Decibels

ω in Degrees

Help Variable Explorer Plots Files

Console 1/A

10°

Enter the number of coefficients for FIR Filter: 9

Enter the frequency response value: 1

C:\ProgramData\Anaconda3\lib\site-packages\scipy\integrate\quadpack.py:463: ComplexWarning: Casting complex values to real discards the imaginary part

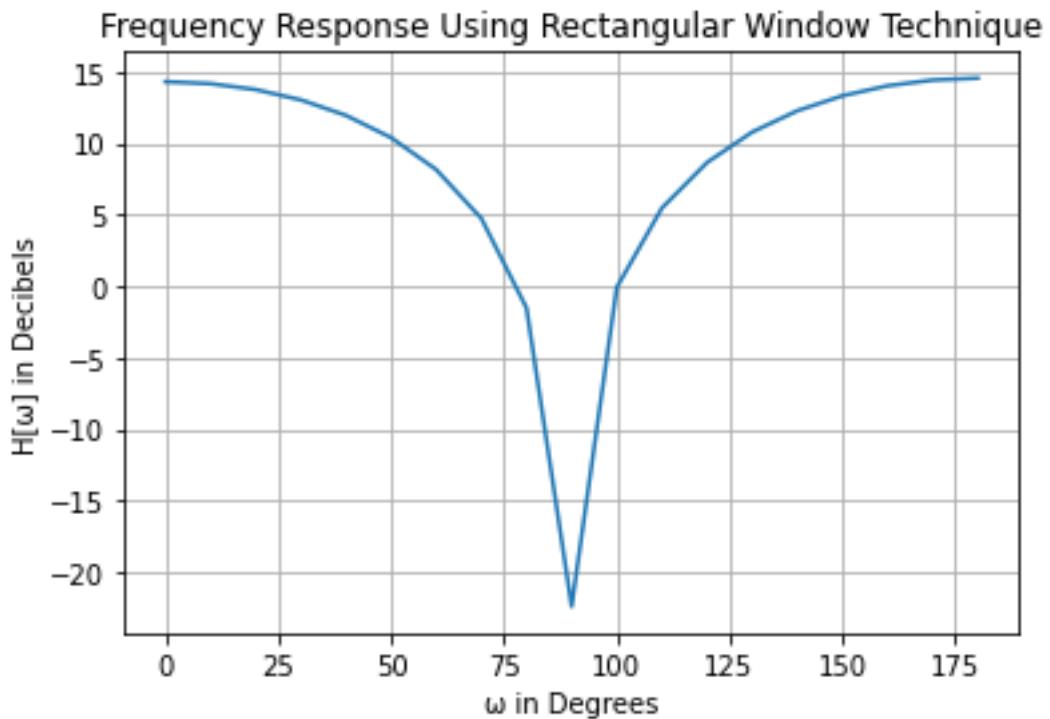
return _quadpack._qngufunc,a,b,args,full_output,epsabs,epsrel,limit

The desired filter coefficients h[n] is as follows:

[0.4, 0.30273069145626275, 0.0935489283788639, -0.062365952252575955, -0.07568267286406569, -0.062365952252575955, 0.0935489283788639, 0.30273069145626275, 0.4]

The expected filter coefficients h[n] obtained using rectangular window technique is as follows:

[0.4, 0.30273069145626275, 0.0935489283788639, -0.062365952252575955, -0.07568267286406569, -0.062365952252575955, 0.0935489283788639, 0.30273069145626275, 0.4]

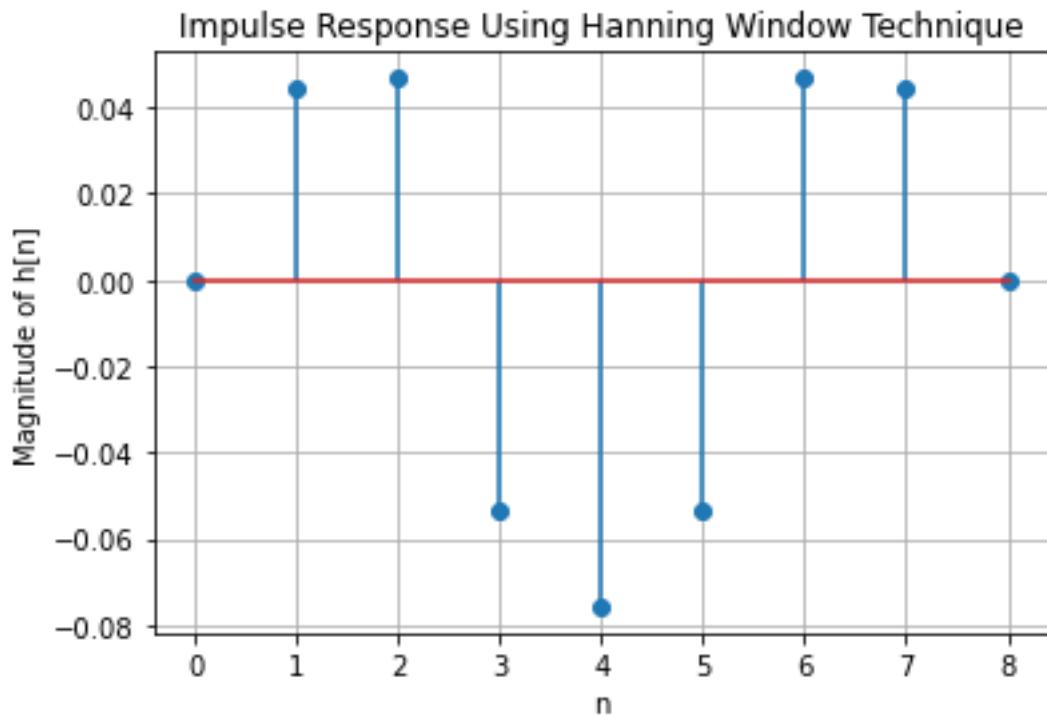
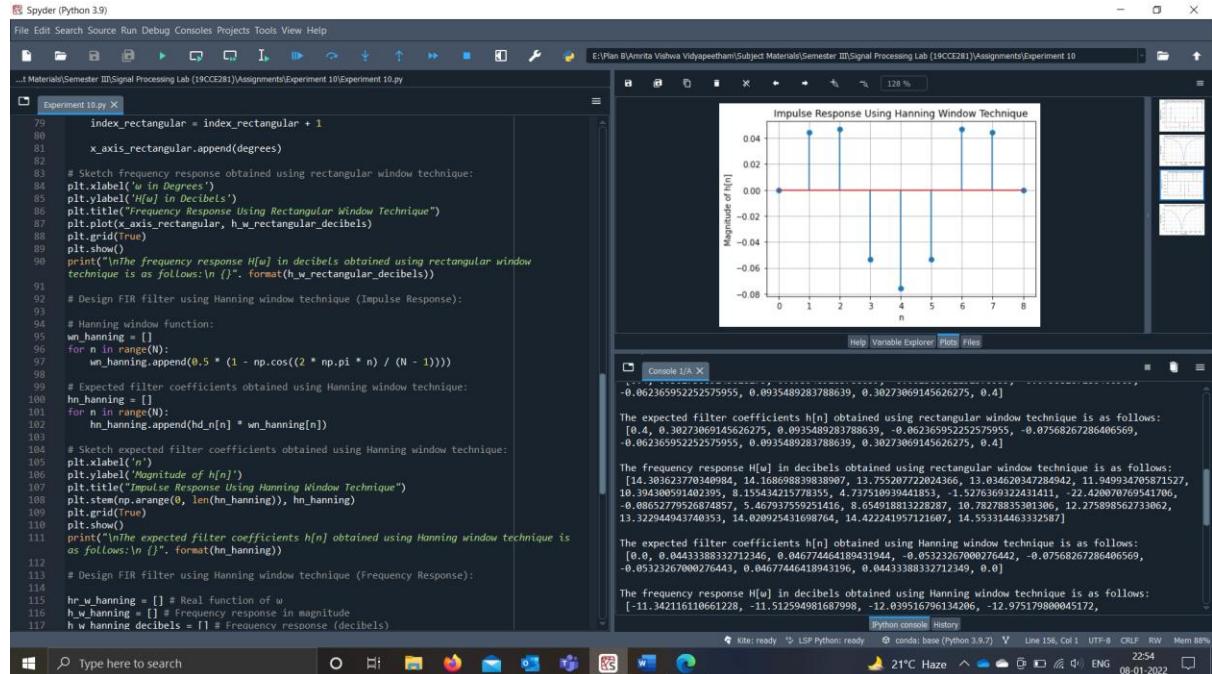


Step 1: Design FIR filter using rectangular window technique (Frequency Response).

Step 2: Initialize the arrays for the real function of ω , frequency response (magnitude), frequency response (decibels) and define the angle axis.

Step 3: Sketch frequency response obtained using rectangular window technique.

Impulse Response Using Hanning Window Technique



Step 1: Design FIR filter using Hanning window technique (Impulse Response).

Step 2: Compute the Hanning window function.

Step 3: Evaluate expected filter coefficients obtained using the rectangular window technique and sketch the same.

Frequency Response Using Rectangular Window Technique

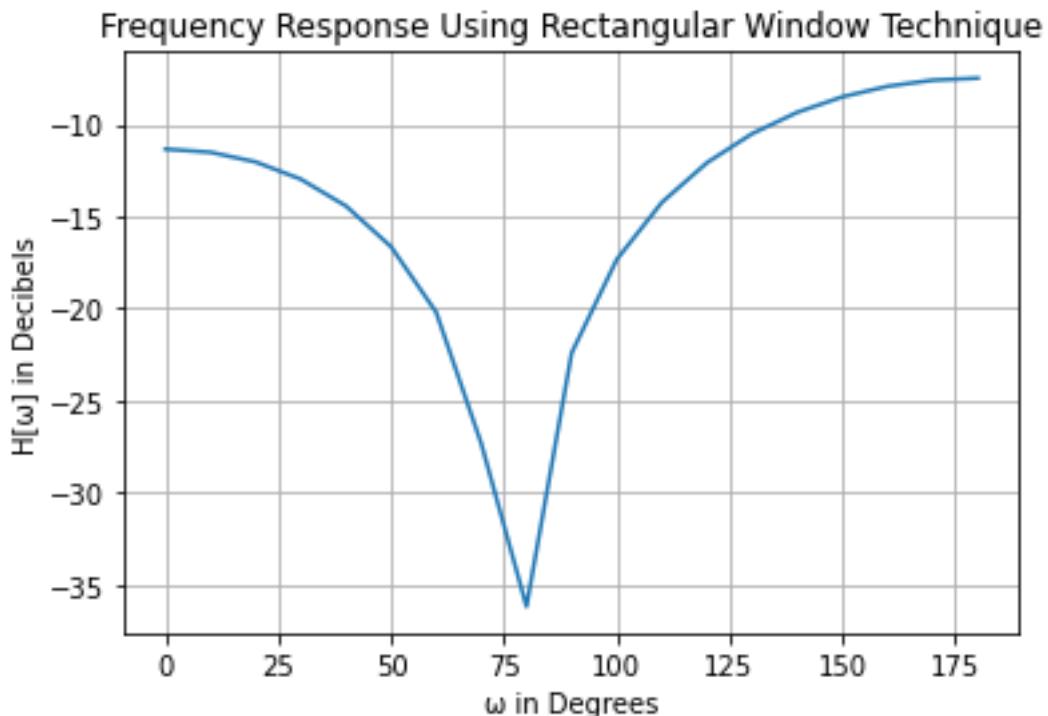
The screenshot shows the Spyder Python 3.9 IDE interface. On the left, the code editor displays a script named 'Experiment 10.py' containing Python code for calculating the frequency response using a rectangular window technique. The code involves defining arrays for the angle axis (w_hanning), sum_hanning, and h_w_hanning, and then iterating through degrees from 0 to 190 to calculate the summation part of the formula. It also includes a plot of the frequency response H[ω] in decibels versus ω in degrees.

```

118     index_hanning = 0 # Index variable to iterate through "hr_w_hanning" list
119     x_axis_hanning = [] # Define the angle axis
120
121     for degrees in range(0, 190, 10):
122
123         w_hanning = degrees * (np.pi/180)
124         sum_hanning = 0
125
126         if (N % 2 == 1):
127
128             for n in range(((N-3)//2) + 1):
129                 summation = h_n_hanning[n] * np.cos(w_hanning) * (((N-1)//2) - n)
130                 sum_hanning = sum_hanning + summation
131                 hr_w_hanning.append(h_n_hanning[((N-1)//2)] + (2 * sum_hanning))
132
133         else:
134
135             for n in range((N//2) - 1):
136                 summation = h_n_hanning[n] * np.cos(w_hanning) * (((N-1)//2) - n)
137                 sum_hanning = sum_hanning + summation
138                 hr_w_hanning.append(2 * sum_hanning)
139
140         h_w_hanning.append(hr_w_hanning[index_hanning] * np.exp(-1j * w_hanning * ((N-1)//2)))
141
142         h_w_hanning_decibels.append(20 * np.log10(abs(h_w_hanning[index_hanning])))
143
144         index_hanning = index_hanning + 1
145
146         x_axis_hanning.append(degrees)
147
148     # Sketch frequency response obtained using Hanning window technique:
149     plt.xlabel('ω in Degrees')
150     plt.ylabel('H[ω] in Decibels')
151     plt.title("Frequency Response Using Rectangular Window Technique")
152     plt.plot(x_axis_hanning, h_w_hanning_decibels)
153     plt.grid()
154     plt.show()
155
156 print("The frequency response H[ω] in decibels obtained using Hanning window technique is as follows:\n{}").format(h_w_hanning_decibels)

```

The right side of the interface shows the resulting plot titled "Frequency Response Using Rectangular Window Technique". The x-axis is labeled "ω in Degrees" and ranges from 0 to 175. The y-axis is labeled "H[ω] in Decibels" and ranges from -35 to -10. The plot shows a deep null at approximately 80 degrees, indicating a low-pass filter characteristic.



Step 1: Design FIR filter using Hanning window technique (Frequency Response).

Step 2: Initialize the arrays for the real function of ω , frequency response (magnitude), frequency response (decibels) and define the angle axis.

Step 3: Sketch frequency response obtained using Hanning window technique.

```
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]  
Type "copyright", "credits" or "license" for more information.
```

```
IPython 7.29.0 -- An enhanced Interactive Python.
```

```
Restarting kernel...
```

```
In [1]:      'E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/  
Signal Processing Lab (19CCE281)/Assignments/Experiment 10/Experiment 10.py'      = 'E:/  
Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester III/Signal Processing Lab  
(19CCE281)/Assignments/Experiment 10'
```

```
Enter the number of coefficients for FIR Filter: 9
```

```
Enter the frequency response value: 1
```

```
C:\ProgramData\Anaconda3\lib\site-packages\scipy\integrate\quadpack.py:463:  
ComplexWarning: Casting complex values to real discards the imaginary part  
    return _quadpack._qagse(func,a,b,args,full_output,epsabs,epsrel,limit)
```

```
The desired filter coefficients h_d[n] is as follows:
```

```
[0.4, 0.30273069145626275, 0.0935489283788639, -0.062365952252575955,  
-0.07568267286406569, -0.062365952252575955, 0.0935489283788639, 0.30273069145626275, 0.4]
```

```
The expected filter coefficients h[n] obtained using rectangular window technique is as follows:
```

```
[0.4, 0.30273069145626275, 0.0935489283788639, -0.062365952252575955,  
-0.07568267286406569, -0.062365952252575955, 0.0935489283788639, 0.30273069145626275, 0.4]
```

```
The frequency response H[w] in decibels obtained using rectangular window technique is as follows:
```

```
[14.303623770340984, 14.168698839838907, 13.755207722024366, 13.034620347284942,  
11.949934705871527, 10.394300591402395, 8.155434215778355, 4.737510939441853,  
-1.5276369322431411, -22.420070769541706, -0.08652779526874857, 5.467937559251416,  
8.654918813228287, 10.78278835301306, 12.275898562733062, 13.322944943740353,  
14.020925431698764, 14.422241957121607, 14.553314463332587]
```

```
The expected filter coefficients h[n] obtained using Hanning window technique is as follows:
```

```
[0.0, 0.04433388332712346, 0.046774464189431944, -0.05323267000276442,  
-0.07568267286406569, -0.05323267000276443, 0.04677446418943196, 0.04433388332712349, 0.0]
```

```
The frequency response H[w] in decibels obtained using Hanning window technique is as follows:
```

```
[-11.342116110661228, -11.512594981687998, -12.039516796134206, -12.975179800045172,  
-14.431524066355657, -16.64594370520493, -20.207868437905297, -27.356168782548593,  
-36.198974461181955, -22.42007076954167, -17.33718643619604, -14.233266653016974,  
-12.07599278330527, -10.501233150336287, -9.339282068886703, -8.49905963433598,  
-7.928154198669187, -7.596189324289906, -7.487198085148416]
```

```
In [2]:
```