VALENCIA COLLEGE

Electrical & Computer Engineering Technology

EET 4190C – DIGITAL SIGNAL PROCESSING

PROGRAMMING PROJECTS

BY

MASOOD EJAZ

PROGRAMMING PROJECT # 1

Quantization of Analog Signals and Calculation of Quantized noise

Objective: This is a MATLAB based exercise. Objective of this exercise is to take user generated analog signal data and calculate the quantized value for each data point using Analog-to-Digital converter specifications, also given by the user. Finally, plot both the sampled signal and quantized signal and calculate signal-to-quantized noise ratio.

Program Requirements:

1. Give proper explanation of your program at the beginning of your script file. Also, include your name, class, semester and date in the first line of the script file. Hence, the first few lines will be comments
2. It is always a good idea to use *clear* and *clc* functions to clear the previous data and command window.
3. Ask user to enter sampled data vector (use *input* function)
4. Ask user to give you information about ADC, including, number of bits, maximum and minimum reference voltages (use *input* function). Ask all three quantities as a vector, i.e. [*N*, *Vref*+, *Vref*-]
5. From the ADC information, calculate the number of steps and the step size. Do not display the calculated values in the command window.
6. Analyze each value of the input sampled data vector and generate the equivalent quantized value. Hold all quantized values in a vector but do not display them in the command window.
7. Plot both original function from sampled data values (use *plot* function) and quantized function from quantized data vector that you generated through your program (use *stairs* function) in the same figure. Properly label your x-axis and y-axis, give a proper title, and add a legend on the plot to display the information about original signal and quantized signal plotting patterns.
8. Calculate the value of signal-to-quantized noise in both regular and logarithmic (dB) scales (use equations (2.10) and (2.11) from the class notes). Properly display these values in the plot window (you can use *text* function).

NOTE: Make sure to use comments to describe different steps of your program.

Submission Information:

Submit the following on Canvas by the due date:

1. Your MATLAB script file. File naming format should be *Your first name_last name_Project1*.
2. A paper with proper explanation of analog-to-digital conversion process with proper expressions to go from input analog voltage to output digital code. Discuss sampling, its requirements, and effects, quantization and its effects, and digitization. Support your explanation with proper figures. This should be a pdf file with the same naming format as described above. Do not explain you program steps in this paper.

Sample Program Output:

```
Enter the sampled data vector to be quantized: 2*cos(5*pi*(0:99)*0.01) + 5*sin(7*pi*(0:99)*0.01)
Enter number of bits, positive, and negative reference voltages of the ADC as a vector: [8 8 -8]
```

Figure 1: Command Window Input and Output

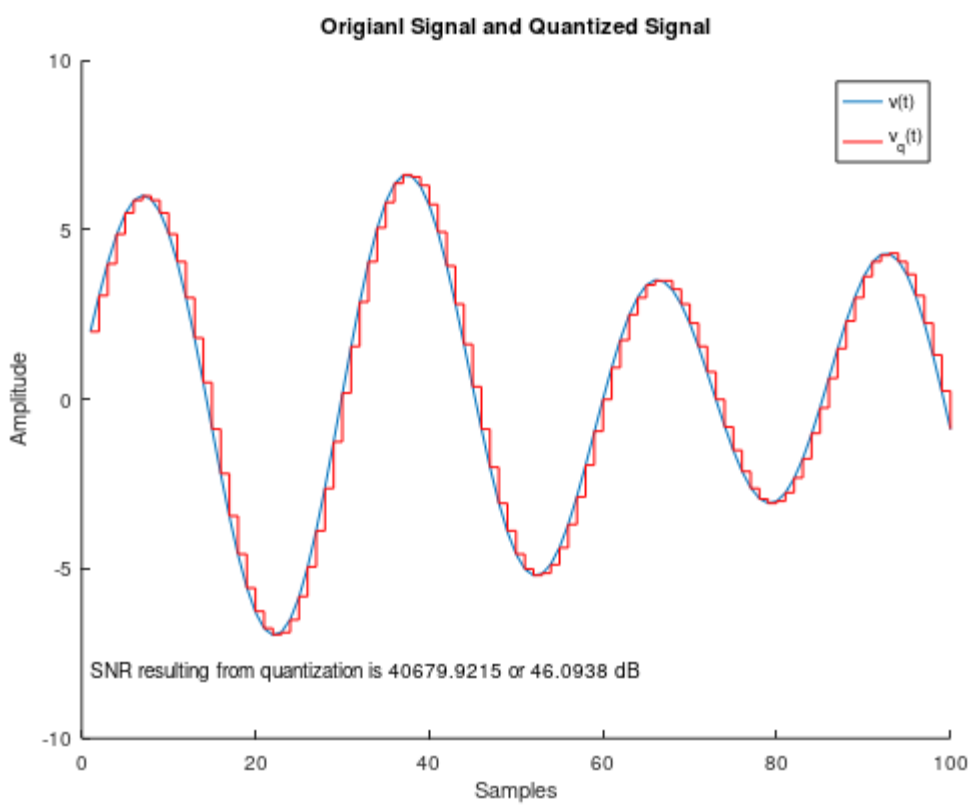


Figure 2: Plot of Original and Quantized Signals with SNR calculation

PROGRAMMING PROJECT # 2

Digital Convolution

Objective: This is a MATLAB based exercise. Objective of this exercise is to create a program that takes user given input and impulse response vectors and produces output of a linear time-invariant system. Also, program should generate a plot of all three quantities, i.e. input, impulse response and output

Program Requirements:

1. Give proper explanation of your program at the beginning of your script file. Also, include your name, class, semester and date in the first line of the script file. Hence, the first few lines will be comments
2. Clear the command window through your program (*clc*).
3. Ask user to enter input data vector *x(n)* for $n \geq 0$ (use *input* function)
4. Ask user to enter impulse response vector *h(n)* for $n \geq 0$ (use *input* function)
5. Calculate output vector *y(n)* using convolution between *x(n)* and *h(n)* as follows,

$$y(n) = x(n) * h(n) = \sum_{k=0}^{\infty} x(k)h(n-k)$$

   Note that if length of *x* is *N* and length of *h* is *M* then length of the output vector *y* will be *N+M-1*
6. Check your output values using MATLAB convolution function *conv()*. Make sure *conv()* is just to check your values, it shouldn't be part of the code.
7. Create three subplots with *x(n)*, *h(n)*, and *y(n)*.

NOTE: Make sure to use comments to describe different steps of your program.

Submission Information:

Submit the following on Canvas by the due date:

1. Your MATLAB script file. File naming format should be *Your first name_last name_Project2*.

2. A paper with proper explanation of digital convolution process and different methods to calculate digital convolution. This should be a Microsoft Word file with same naming format as described above. Do not explain you program steps in the paper.

Sample Program Output:

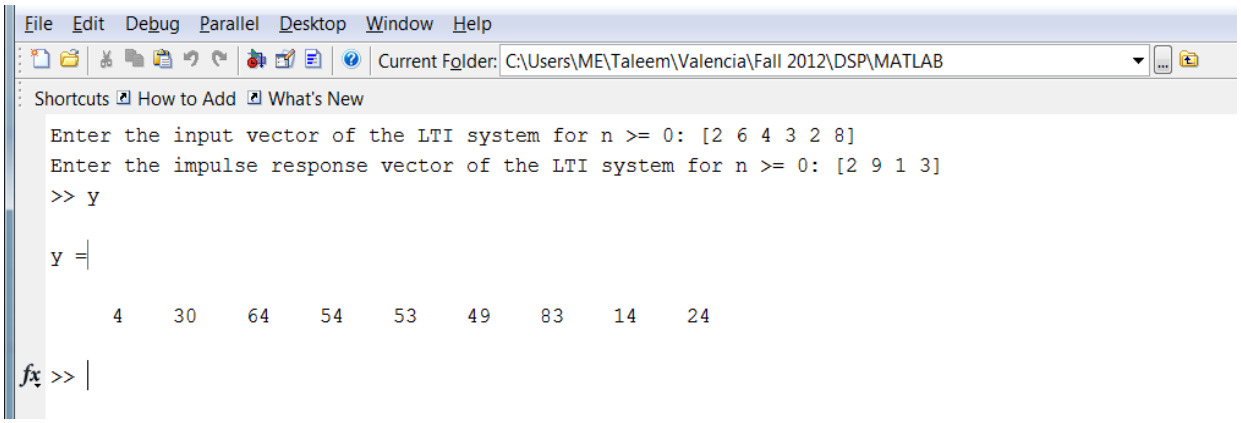


Figure 1: Command Window Input and Calculated Output

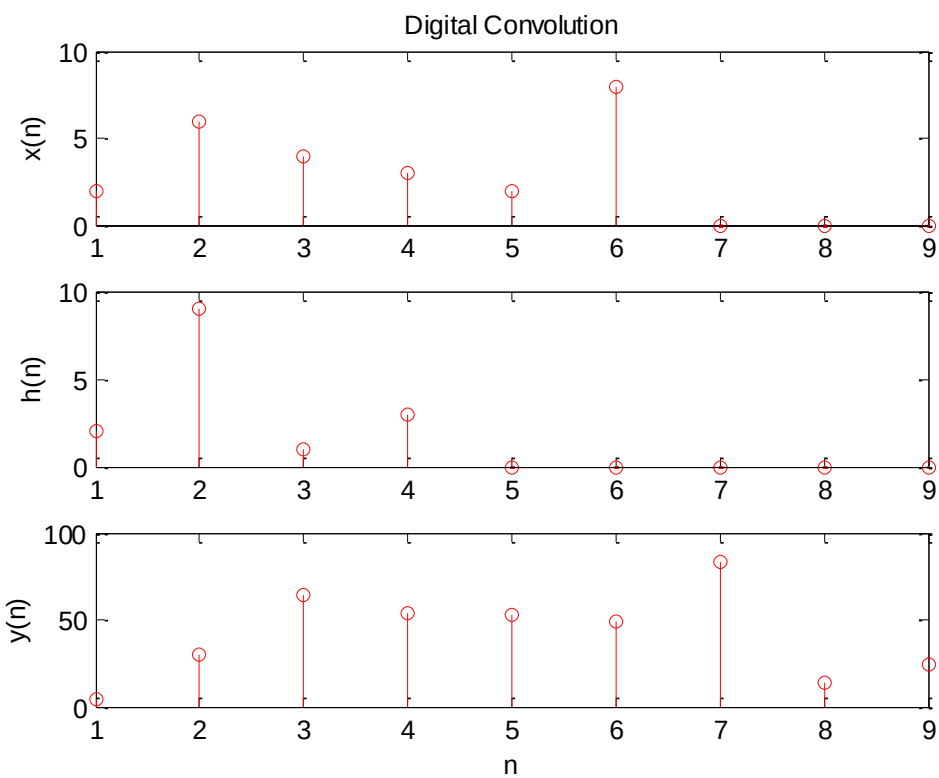


Figure 2: Plot of Input, Impulse Response, and Output Vectors

PROGRAMMING PROJECT # 3

Generation of Digital Signals and Signal Spectral Analysis

Objective: This is a MATLAB based exercise. The objective of this exercise is to generate different digital signals and to investigate about their signal spectra and bandwidth using Discrete Fourier Transform.

Program Requirements:

1. Include your name, class, semester and date in the first line of the script file followed by a proper explanation of your program. Hence, the first few lines of your script file should be comments.
2. Clear the command window through your program (*clc*).

Signal # 1: White Noise with Gaussian Distribution

3. Generate a random *White Noise with Gaussian/Normal Distribution* signal vector, *x(n)*, comprised of 4096 points having zero mean and standard deviation of five, using *randn* function. Check help of the function to further your knowledge.
4. Assume that 4096 (2^{12}) samples of the above signal that you generated are obtained at a sampling frequency of 8KHz. Obtain the signal spectrum *X(k)* (*fft*).
5. Plot the sampled signal *x(n)* vs. *n* (*stem)* and *amplitude* spectrum, A_X, of the signal *X(f)* vs. *f* (*plot*) (Make sure to convert the x-axis for *X(f)* from discrete *k* to continuous *f*) using two subplots. For better visualization of the original signal, plot only first 100 samples of the input signal. Properly label your plots. This will be *figure 1*.
6. Print out the value of the frequency resolution, Δf in the plot window (use *text* function)
7. Include *sound* function in your program to play a sound corresponding to the scaled value of white noise vector,

   ```
   sound(x/max(abs(x)),fs)
   ```

   where *fs* is sampling frequency.

Signal # 2: Sum of Sinusoidal Signals

8. Generate the following three sinusoidal signals sampled at 8KHz up to time length of 0.1 second.

(i) $x_1(t) = 5\cos[2\pi(500)t]$

(ii) $x_2(t) = 5\cos[2\pi(1200)t + 0.25\pi]$

(iii) $x_3(t) = 5\cos[2\pi(1800)t + 0.5\pi]$

9. Create a signal *x(t)* that is the sum of x_1, x_2, and x_3. Note that *x(t)* is also a sampled signal with sampling rate of 8KHz up to the time length of 0.1 second.
10. Calculate the frequency spectrum of *x(n)*, i.e. *X(k)*
11. Plot two figures; *figure 2* with four subplots with x_1, x_2, x_3, and *x*, and *figure 3* with two subplots with *amplitude* of the spectrum, *X(k)* and *X(f)*, i.e. discrete scale *k* converted into continuous scale *f*. Plot only the first 30 samples of the time signals x_1, x_2, x_3, and *x* for a better visualization. Properly label your plots.
12. Properly print the following in the plot window through your program
    (i) Frequency resolution
    (ii) Frequency components that you observe in the spectrum.
13. Include *sound* function in your program to play a sound corresponding to the scaled values of x_1, x_2, x_3, and *x*. Use *pause* at the end of each *sound* function to create a pause between the successive signal sounds,

```
sound(x21/max(abs(x21)),fs),pause
```

Signal # 3: Digital Speech Waveform

14. Download the audio file, which is sampled at 8KHz, from this link: speech.wav
15. Read the file in your program as follows:

```
x = audioread('speech.wav')
```

16. Obtain the frequency spectrum of the sampled data *x* (*fft*).
17. Calculate the amplitude spectrum from the frequency spectrum.
18. Create two subplots, one with the given samples vs. time and other with the *amplitude* of the frequency spectrum of samples vs. frequency in Hz. This will be *figure 4*.
19. From the plots, observe the maximum frequency (bandwidth) of the signal (highest frequency with substantial magnitude)
20. Print the following in the plot window through your program,
    (i) Frequency resolution
    (ii) Important frequency components in the signal (from your observation)
21. Include *sound* function in your program to play a sound corresponding to the scaled value of the speech vector, as you did earlier for the other signals.

Submission Information:

Submit your MATLAB script file saved in the file-naming format, *Your first name_last name_Project3*, on Canvas by the due date.

Reference

Lab 1: Generation of Digital Signals and Signal Spectral Analysis Using MATLAB by Li Tan.

http://www.elsevierdirect.com/v2/companion.jsp?ISBN=9780123740908

PROGRAMMING PROJECT # 4

Z-transfer Functions, Difference Equations, and Filter Implementation

Objective: This is a MATLAB based exercise. The objective of this exercise is to create two programs to implement a fourth-order band-pass filter in two different ways. The fourth-order band-pass filter is defined by the following transfer function:

$$H(z) = \frac{0.0201 - 0.0402z^{-2} + 0.0201z^{-4}}{1 - 2.1192z^{-1} + 2.6952z^{-2} - 1.6924z^{-3} + 0.6414z^{-4}}$$

Description of each of the program is given in the *program requirements* section as follows.

Program Requirements:

1. Include your name, class, semester and date in the first line of the script file followed by a proper explanation of your program. Hence, the first few lines of your script file should be comments.
2. Clear the command window through your program (*clc*).
3. Load the *speech.wav* file using *audioread* function. Assume that the sampling frequency (f_s) is 8KHz. This is your input vector, *x*.
4. Plot the filter's magnitude response (in decibel; $20\log_{10}(|H(e^{j\Omega})|)$ and phase response (in degrees) equations from *H(z)* by taking $z = e^{j\Omega}$, from zero to the folding frequency ($f_s/2$). Magnitude response axis should be set from -40db to 0db and phase response axis should be set between -200° to 200°. This will be *figure 1* with two subplots.

First Method:

5. Create difference equation to represent system output *y*(*n*) using the transfer function *H*(*z*). In the difference equation, input *x*(*n*) is the samples of the speech signal.
6. Through your program, implement the difference equation to calculate the output *y*(*n*). Length of *y*(*n*) should be the same as the length of the speech signal *x*(*n*).
7. Plot the input *x*(*n*) and output *y*(*n*) in two subplots. *x-axis* of both the sub-plots should be in time instead of sample number, *n*. Remember, $t = nT_s$. This will be *figure 2.*
8. Calculate the spectral contents corresponding to the input *x(n)* and output *y(n)* using *fft()* function.
9. Plot the amplitude spectra of input and output in two sub-plots. *x-axis* for each of the sub-plots should be in *hertz* going from zero to the folding frequency, $f_s/2$. This will be *figure 3*.
10. Include *sound* function in your program to play sounds corresponding to the scaled values of your input *x(n)* and output *y(n)*:

```
sound(x/max(abs(x)), fs), pause
sound(y/max(abs(y)), fs)
```

where *fs* is the sampling frequency.

Second Method:

11. Calculate the frequency response of the filter using *freqz* function

```
[H,W] = freqz([0.0201, 0, -0.0402, 0, 0.0201], [ 1, -2.1192, 2.6952, -
1.6924, 0.6414], K);
```

where *H* is the frequency response evaluated at the values of *W*. *W* is the normalized frequency vector, Ω, between 0 to π divided into *K* points. Take *K* = 512 points. Observe that the two vectors in *freqz* functions correspond to the coefficients of z^{-k} in the numerator followed by the denominator.

12. Plot the filter's magnitude (in decibel) and phase response (in degrees) from *H*. Magnitude response axis should be set from -40db to 0db and phase response axis should be set between -200° to 200°. This will be *figure 4* with two subplots.
13. Calculate the output of the filter using *filter* function,

```
y = filter([0.0201, 0, -0.0402, 0, 0.0201], [ 1, -2.1192, 2.6952, -1.6924,
0.6414], x);
```

Observe that the two vectors in *filter* function correspond to the coefficients of z^{-k} in the numerator followed by the denominator.

14. Plot the input *x*(*n*) and output *y*(*n*) in two subplots. *x-axis* of both the sub-plots should be in second (time). This will be *figure 5*.
15. Calculate the spectral contents corresponding to the input *x(n)* and output *y(n)* using *fft()* function. Calculate amplitude response for each of the spectra.
16. Plot the amplitude spectra of input and output in two sub-plots. x-axis for each of the sub-plots should be in *hertz* going from zero to folding frequency. This will be *figure 6*.
17. Include *sound* function in your program to play sounds corresponding to the scaled values of your input *x(n)* and output *y(n)*.

```
sound(x/max(abs(x)), fs), pause
sound(y/max(abs(y)), fs)
```

where *fs* is the sampling frequency.

PROGRAMMING PROJECT # 5

FIR Filter Design Using Window Functions

Objective: This is a MATLAB based exercise. The objective of this exercise is to create a program that calculates the frequency response of a low-pass, high-pass, band-pass, or band-stop filter given the number of taps, sampling frequency, and cut-off frequency for low-pass and high-pass filters and upper and lower cut-off frequencies for band-pass and band-stop filters. Also, four window functions, Triangular, Hanning, Hamming, and Blackman will be used to smooth out the original filter's response and to eliminate the oscillations in the pass and stop bands due to the Gibbs effect.

Description of the program is given in the *program requirements* section as follows.

Program Requirements:

1. Include your name, class, semester and date in the first line of the script file followed by a proper explanation of your program. Hence, the first few lines of your script file should be comments.
2. Clear all the variables (*clear)* and the command window through your program (*clc*).
3. Ask user to enter the following information:
    a. Number of taps (2M + 1)
    b. Filter type (lowpass, highpass, bandpass, bandstop)
    c. Sampling frequency in hertz
    d. Window type (*triangular, hanning, hamming, blackman*)

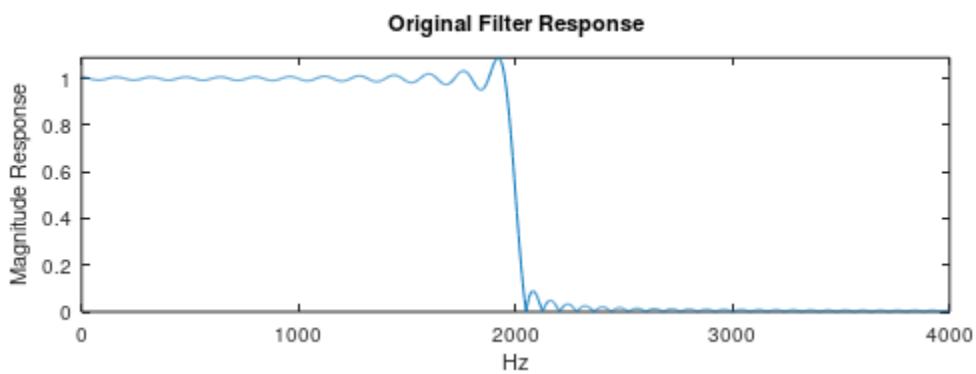
4. Use *switch* or *if* routine to select one of the four filter types; *lowpass, highpass, bandpass, and bandstop.*
5. If user selects *lowpass* or *highpass*, ask user to enter the cutoff frequency in hertz, and if the selection is *bandpass* or *bandstop*, ask for the lower and upper cutoff frequencies.
6. Calculate the corresponding normalized frequencies
7. Calculate the causal filter coefficients, b_o to b_{2M}, total 2M + 1, from non-causal filter coefficients, *h(-M)* to *h(M)* calculated from *table 7.1* of your text book.
8. Calculate the frequency response using *freqz* function. Use 1024 points between $\Omega = 0$ to π.
9. Use another *switch* or *if* routine, this time for one of the four window types; triangular, Hanning, Hamming, and Blackman.
10. Calculate window function coefficients, *w*, from $-M$ to M. Multiply *h* and *w* to get new non-causal filter coefficients, h_w.
11. From h_w, calculate the causal filter coefficients b_w
12. Calculate windowed filter frequency response using *freqz* function. Use 1024 points for normalized frequency Ω.

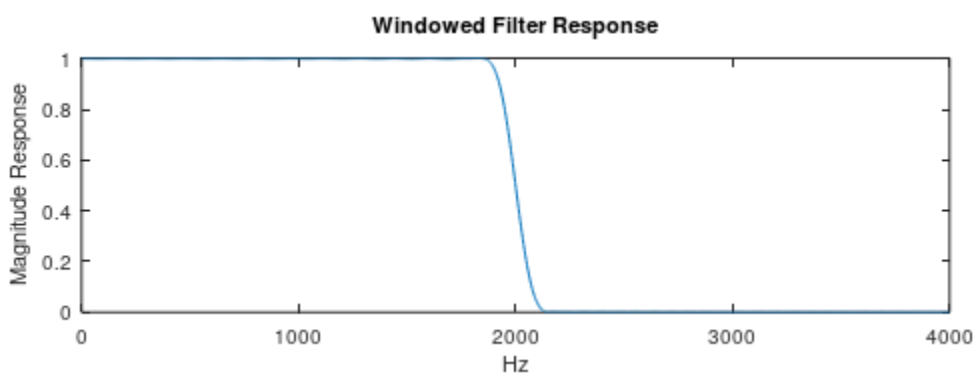
13. Create two figures, each with two sub-plots; *figure (1)* with sub-plots for the original and windowed filter magnitude responses from zero to *fs*/2 hertz, and *figure(2)* with two phase responses, in degrees, from the original and windowed filter frequency responses, from zero to *fs*/2 hertz.

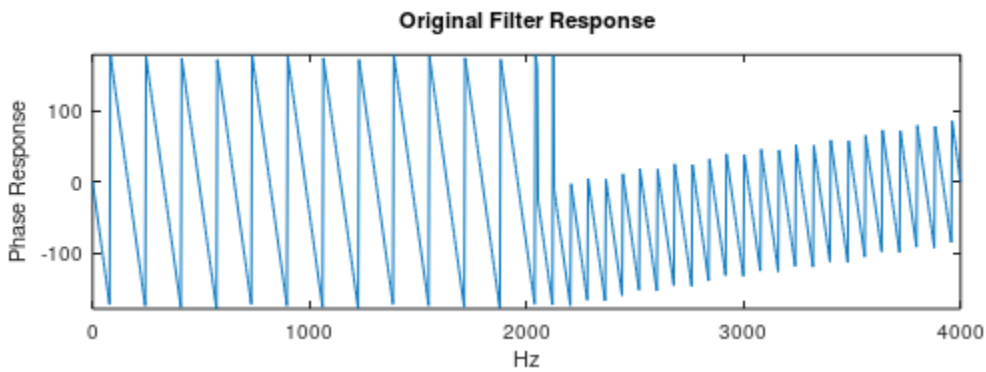
Program Output Samples

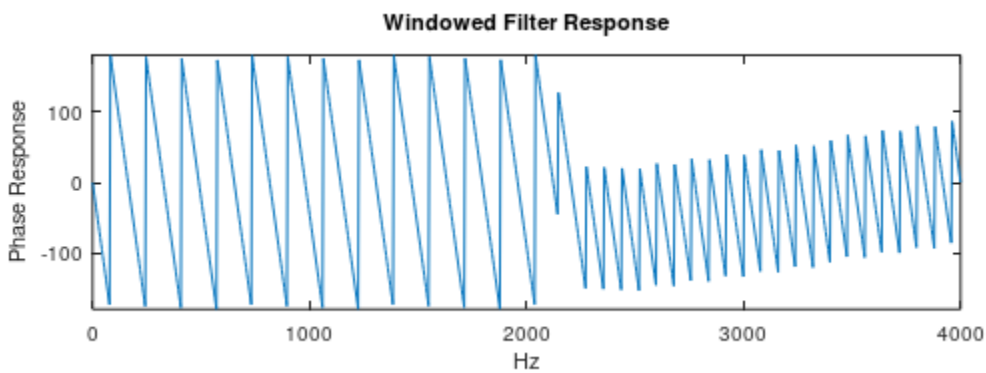
Low-pass Filter

```
Enter Number of taps:99
Enter Filter Type; lowpass, highpass, bandpass, or bandstop:lowpass
Enter Window Function Name; triangular, hamming, hanning, blackman:hamming
Enter sampling rate in Hz: 8000
Enter cut-off frequency in Hz: 2000
>>
```



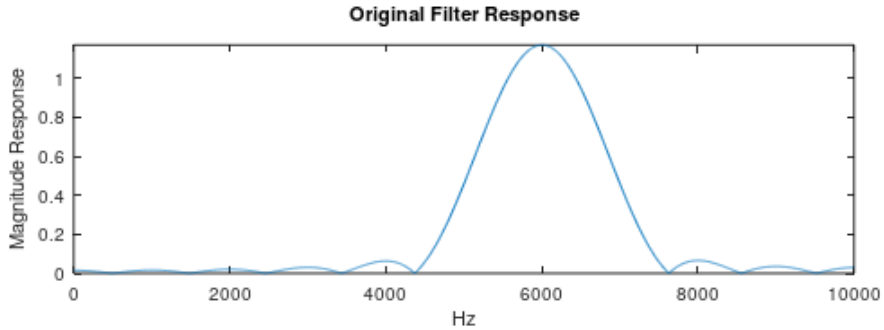


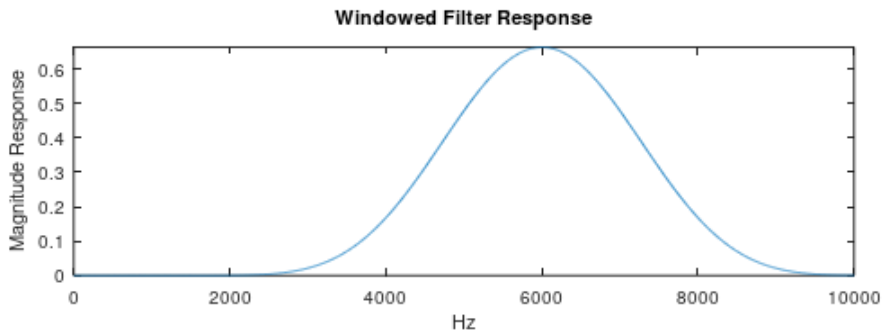


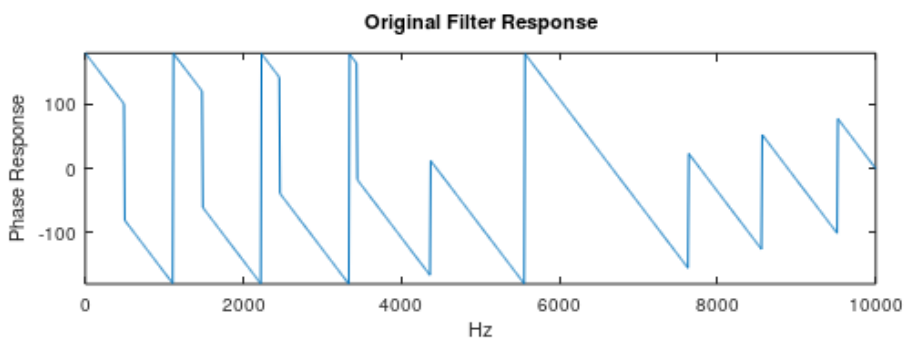


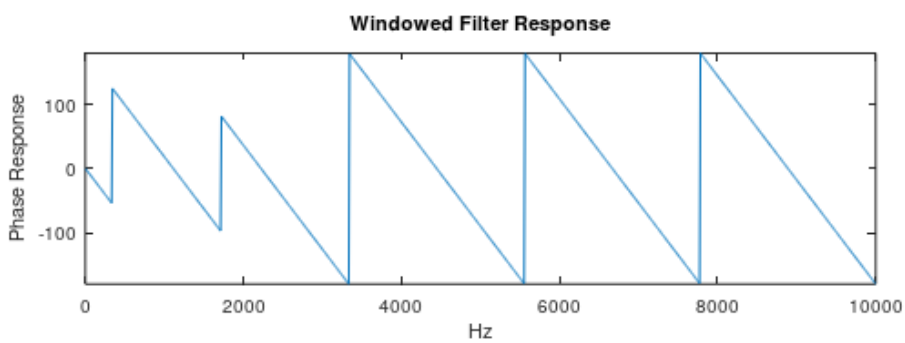
Band-pass Filter

```
Enter Number of taps:19
Enter Filter Type; lowpass, highpass, bandpass, or bandstop:bandpass
Enter Window Function Name; triangular, hamming, hanning, blackman:blackman
Enter sampling rate in Hz: 20e3
Enter lower cut-off frequency in Hz: 5e3
Enter upper cut-off frequency in Hz: 7e3
>>
```









<u>Submission Information:</u>

Submit your MATLAB script file saved in the file-naming format, *Your first name_last name_Project5*, on Canvas by the due date.

PROGRAMMING PROJECT # 6

FIR Filter Application: Noise Reduction

Objective: This is a MATLAB based exercise. The objective of this exercise is to create a program to design a low-pass filter according to the given specifications. This filter is then used to curb noise from a noisy tone signal. Responses of the original signal and its frequency spectrum, and filtered signal and its frequency spectrum will be plotted and studied.

Description of the program is given in the *program requirements* section as follows.

Program Requirements:

1. Include your name, class, semester and date in the first line of the script file followed by a proper explanation of your program. Hence, the first few lines of your script file should be comments.
2. Clear all the variables (*clear)* and the command window through your program (*clc*).
3. Produce 250 samples of a noisy tone signal sampled at 8KHz, *yn = x + n*, from the following clean tone signal,

$$x(t) = 2\sin(1000\pi t)$$

   and normalized Gaussian noise signal generated using MATLAB function *randn,*

   noise = randn(1,250); n = noise/max(abs(noise));

4. Calculate the frequency spectrum of the noisy signal *yn* using *fft* function. Calculate the amplitude spectrum from the frequency spectrum.
5. Create *figure(1)* with two sub-plots; one with the noisy signal against the number of samples, and second with the amplitude spectrum of the noisy signal against the cyclic frequency in Hz. Keep your cyclic frequency range from 0 to the folding frequency, *fs*/2.
6. Design a lowpass *FIR* filter with f_{pass} = 800Hz and f_{stop} = 1KHz. Maximum pass-band ripple should not exceed 0.02dB and minimum stop-band attenuation should be 50dB.
   (*Design Hints: Calculate normalized transient bandwidth, Δf. Using Δf and ripple information, decide which window function satisfies the design requirements and then calculate number of taps. Calculate cut-off frequency. Using number of taps, cut-off frequency and window type, use experiment 5 to generate windowed LPF coefficients*)
7. Apply designed filter on the noisy signal *yn* to get the filtered output signal. Use *filter* function of MATLAB.

8. Use *fft* function to calculate the frequency spectrum of the filtered signal. Calculate amplitude spectrum from the frequency spectrum.
9. Create *figure(2)* with two sub-plots; one with the filtered signal against the number of samples, and second with the amplitude spectrum of the filtered signal against the cyclic frequency in Hz. Keep your cyclic frequency range from 0 to the folding frequency.
10. Include comments in your script file regarding the technical reason behind the delay that you see in the filtered signal graph. Also, comment on the difference between the two amplitude spectra, noisy signal and the filtered signal.

<u>Program Output Samples</u>

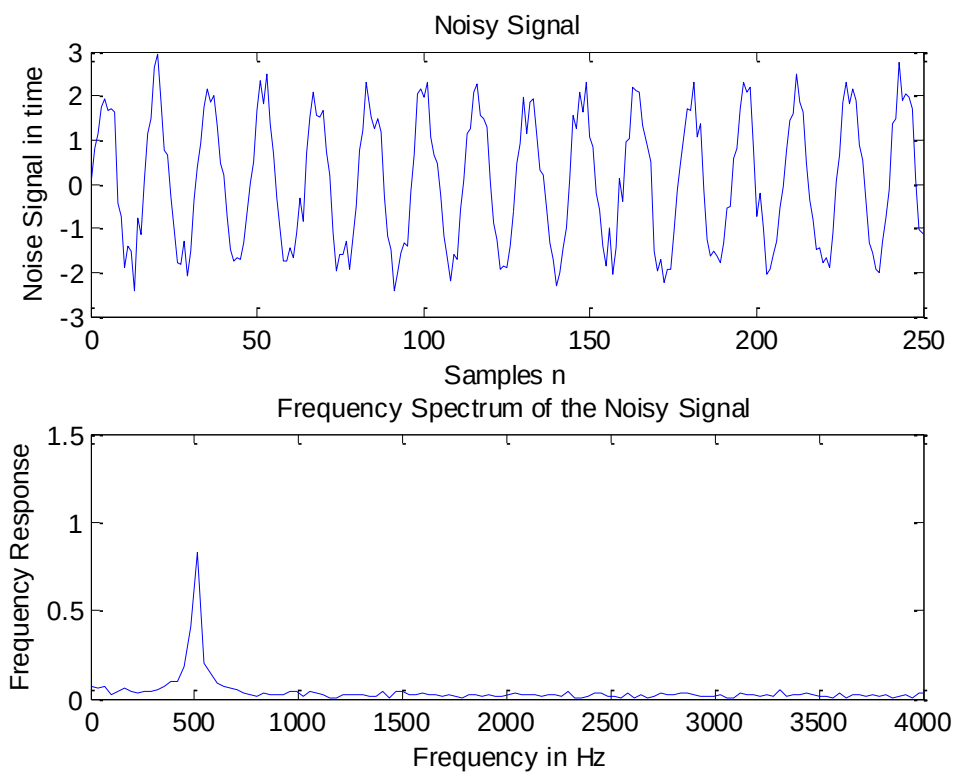


Figure 1: Noisy Signal and its Spectrum

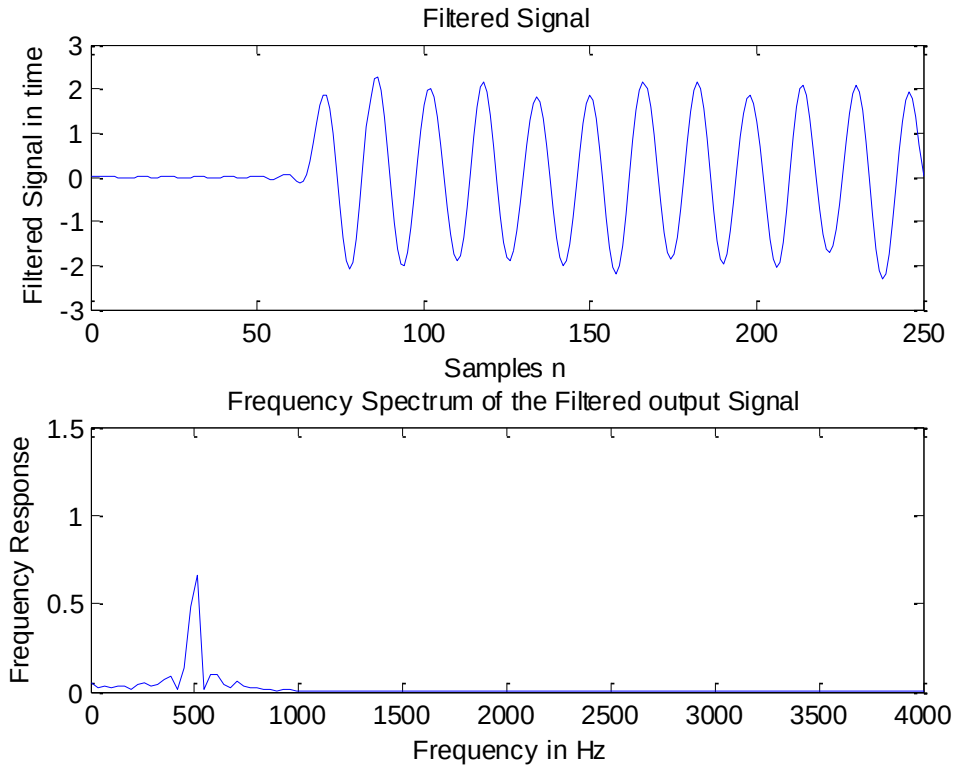


Figure 2: Filtered Signal and its Spectrum

<u>Submission Information:</u>

Submit your MATLAB script file saved in the file-naming format, *Your first name_last name_Project6*, on Canvas by the due date.

PROGRAMMING PROJECT # 7

IIR Filter Design by Bilinear Transformation Method

Objective: This is a MATLAB based exercise. The objective of this exercise is to create a program to design IIR filters using Bilinear Transformation (BLT) method and plot their magnitude and phase responses.

Description of the program is given in the *program requirements* section as follows.

Program Requirements:

1. Include your name, class, semester and date in the first line of the script file followed by a proper explanation of your program. Hence, the first few lines of your script file should be comments.
2. Clear all the variables (*clear)* and the command window through your program (*clc*).
3. Ask user about the filter type: *lowpass, highpass, bandpass,* or *bandstop.*
4. Ask user about the sampling frequency, *fs*.
5. Since we need to plot the frequency response, define the normalized frequency, Ω, vector from 0 to π divided into several points.
6. Map the normalized frequency onto *z*, i.e. $z = e^{j\Omega}$.
7. Map the digital transform variable *z* onto the analog transform (Laplace) variable *s*: $s = \frac{2}{T_s}\frac{z-1}{z+1}$. Since *z* is a vector, make sure to use 'periods' properly to perform array-wise operation. Note that to create the logic to plot the frequency response easily, we went backwards, i.e., from Ω to *s* instead of the other way round.
8. Use a *switch* statement for the filter type with four cases: one for each type of filter. You can also use *if-else-if* routine, but *switch* may be easier.
9. For *lowpass* and *highpass* filters, ask user to enter the digital cutoff frequency in hertz, *fc*.
10. From *fc* determine ω_c, and then determine the corresponding analog cutoff frequency using frequency warping: $\omega_a = \frac{2}{T_s}\tan(\frac{\omega_d T_s}{2})$.
11. Define the transfer function for the analog lowpass or highpass filter,

    Lowpass: $H = \dfrac{1}{1+\dfrac{s}{\omega_a}}$

    Highpass: $H = \dfrac{1}{1+\dfrac{\omega_a}{s}}$

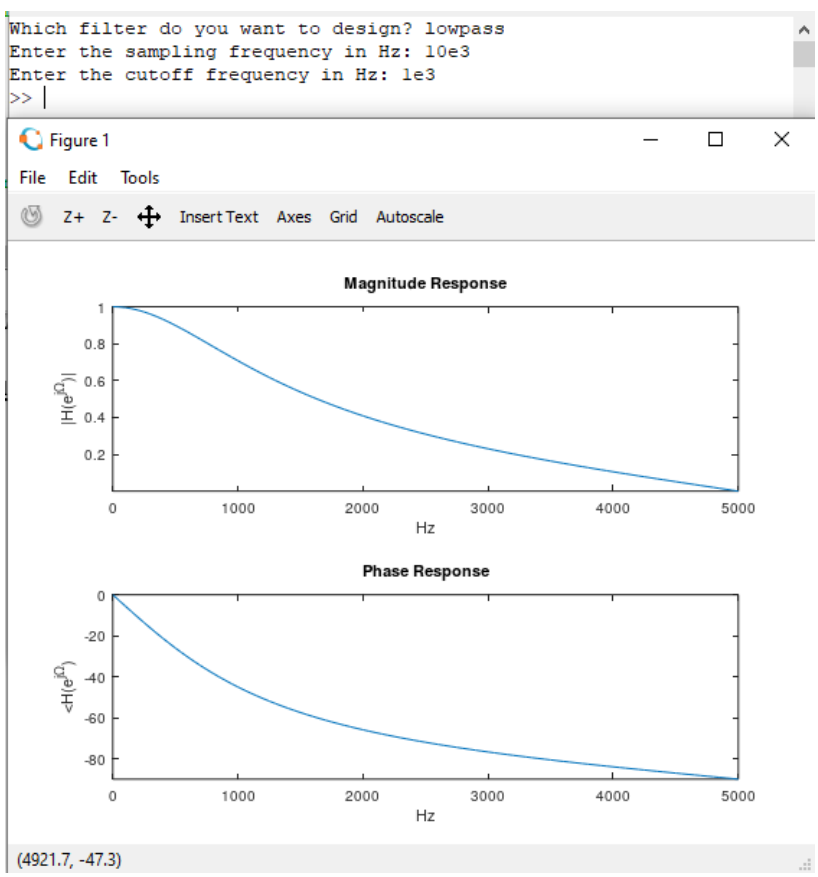
Since s is a vector defined in terms of z, which in turns is defined in terms of Ω, the analog equations will calculate the frequency response of the digital filter (remember, we are going backwards)

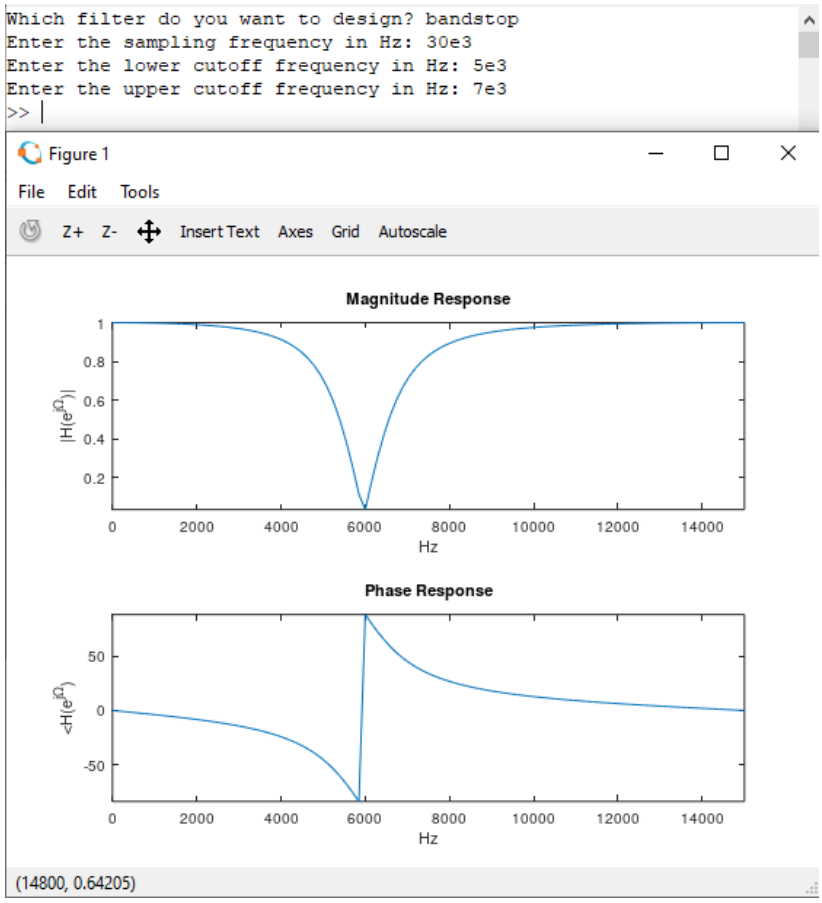
12. Write the other two cases, i.e., for *bandpass* and *bandstop* filters similarly. For each of these filters, you will ask user to enter the *lower* and *upper* cutoff frequencies in hertz.
13. Next, you will determine the corresponding analog filter *lower* and *upper* cutoff frequencies using frequency warping. These two frequencies will be used to calculate the center frequency ω_o and bandwidth W, which will be used in the transfer functions H of the two filters,

    Bandpass: $H = \dfrac{1}{1 + \dfrac{s^2 + \omega_o^2}{sW}}$

    Bandstop: $H = \dfrac{1}{1 + \dfrac{sW}{s^2 + \omega_o^2}}$
14. Once the transfer function H of the required filter is obtained, plot the magnitude and phase responses of H using *abs()* and *angle()* functions. Make sure that your *x-axis* is in hertz.

<u>Program Output Samples:</u>





Submission Information:

Submit your MATLAB script file saved in the file-naming format, *Your first name_last name_Project7*, on Canvas by the due date.