Digital Signal Processing Laboratory

Experiment 5

By: Hardik Tibrewal (18EC10020)

Adaptive Line Enhancer

Objective:

To design an adaptive line enhancer (adaptive filter)

Theoretical Background:

An adaptive line enhancer (ALE) is used to detect a low-level sine wave of unknown frequency in presence of noise. An ALE is an adaptive digital filter, capable of adapting its filter coefficients in a manner that allow it to become a band-pass filter centred at the frequency of the sine wave present in the input. The coefficients of the filter are initialised to zero, and then the input signal is passed through it. An error estimate of choice is used to compare the actual and desired output, depending on which the filter coefficients are updated.

The Least Mean Square algorithm is implemented here, which uses the error estimate to suitably update the filter coefficients till the relative change in the parameters in less than a chosen threshold. The parameters are updated using the gradient descent approach, which tries to ensure that the error in successive steps converges to the least possible error. The rate at which it converges is decided by the convergence factor chosen by us, and the error signal. We continue iterating till the relative change is greater than the error threshold, and stop when it goes below the threshold.

$$\overrightarrow{x[n]} = [x[n]\ x[n-1]\ \cdots\ x[n-m+1]]$$
$$\overrightarrow{h_t} = [h[0]\ h[1]\ \cdots\ h[m-1]]$$

$$y[n] = \overrightarrow{x[n]}^T \overrightarrow{h}$$
$$e[n] = x[n] - y[n]$$

$$\overrightarrow{h_{t+1}} = \overrightarrow{h_t} + \mu(\overrightarrow{x[n]})e[n]$$
$$\text{Continue till } \frac{||\overrightarrow{h_{t+1}} - \overrightarrow{h_t}||^2}{||\overrightarrow{h_t}||^2} \geq \epsilon$$

Pseudocode:

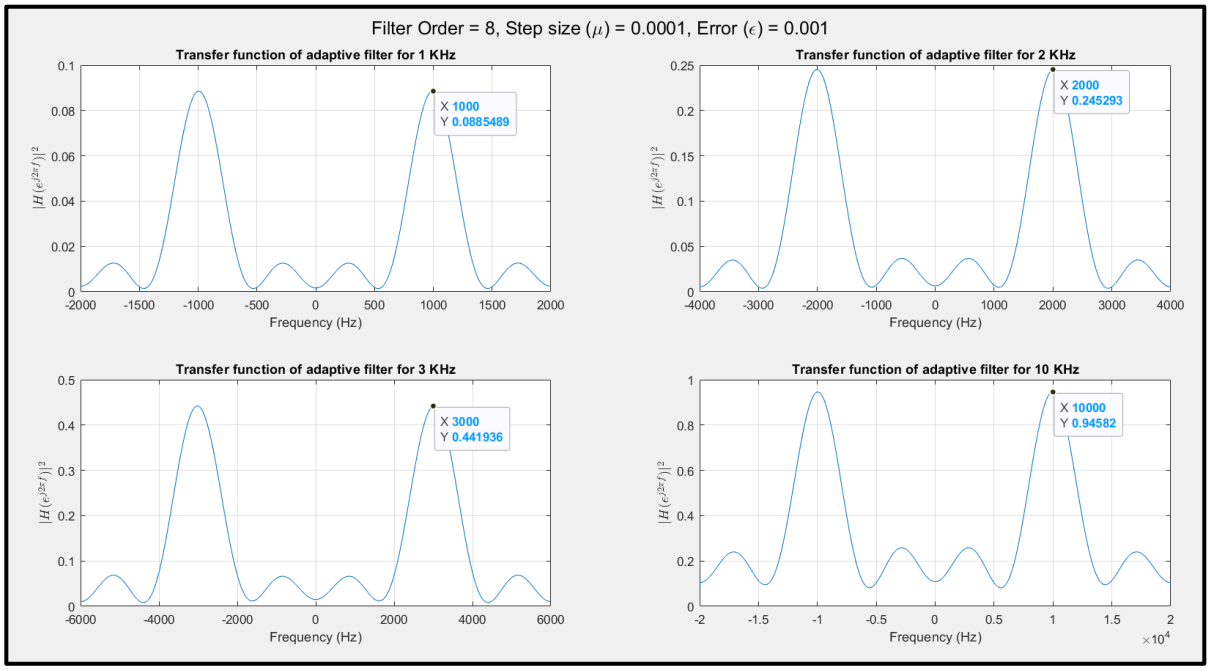
The parameters (filter order, convergence rate, error threshold) to be chosen for the ALE are set, and a sine wave of a chosen frequency is generated and corrupted with Gaussian noise of zero mean and unity variance to obtain the input signal.

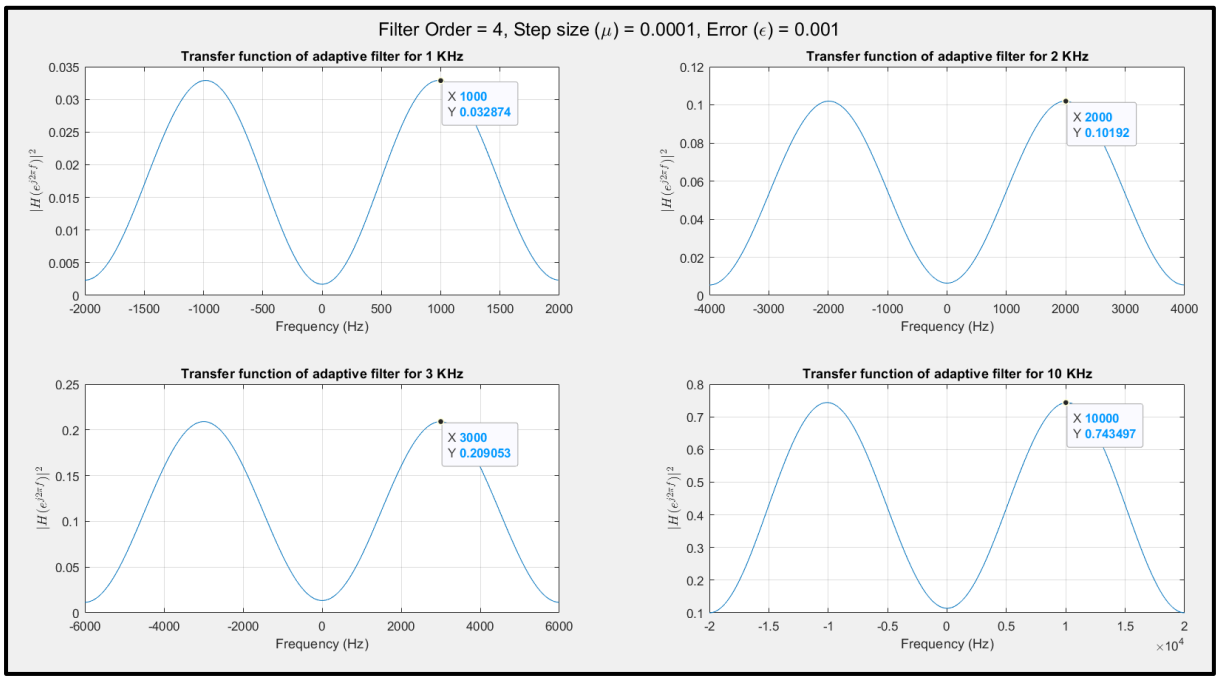
The filter coefficients are initialised to zero, and the input is split into blocks of suitable size (i.e., same as the number of filter coefficients) so that the vector operations are valid. The input signal is passed through the ALE. The error between the desired and actual outputs is calculated, multiplied to the input vector, and added to the filter coefficients to update them. The relative change between the old and new coefficients is calculated used the square of the Euclidean distance between the vectors. This is done for the input vector corresponding to each sample of the input.

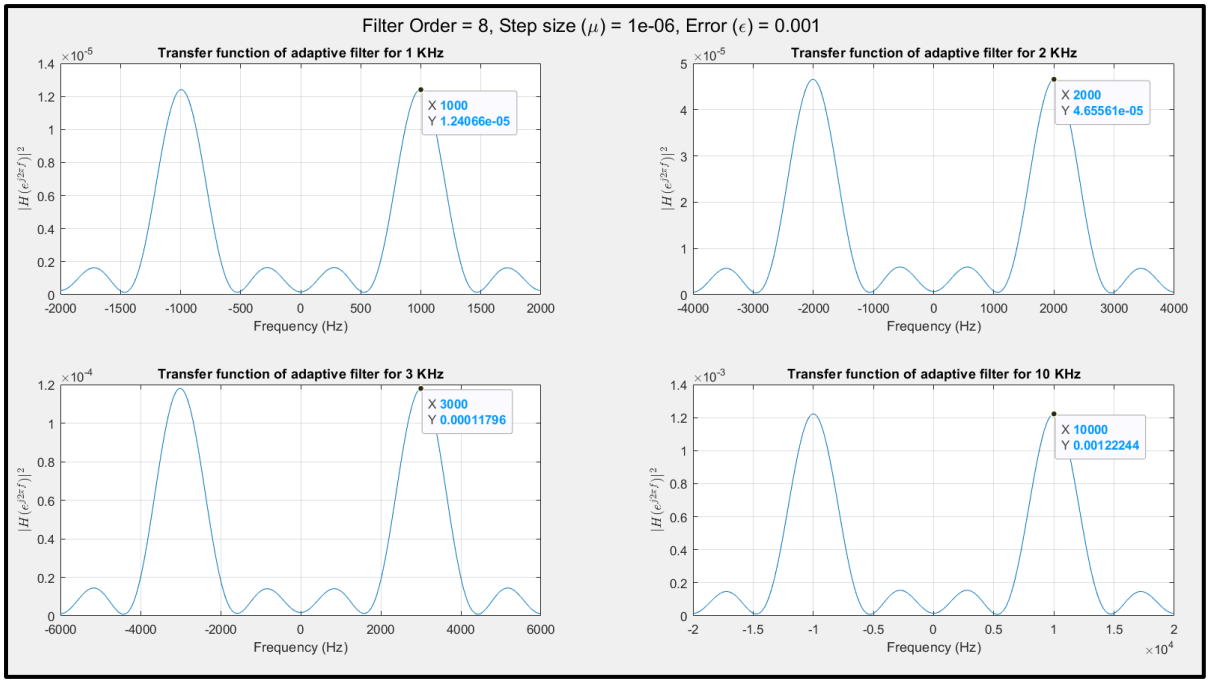
This process is repeated till the change due to the update falls below our chosen threshold. The transfer function of the ALE is also plotted after the training, to verify if the filter is a bandpass filter centred around the input frequency. The effect of varying the hyperparameters (filter order, convergence rate, and error threshold) is also observed by changing their values.

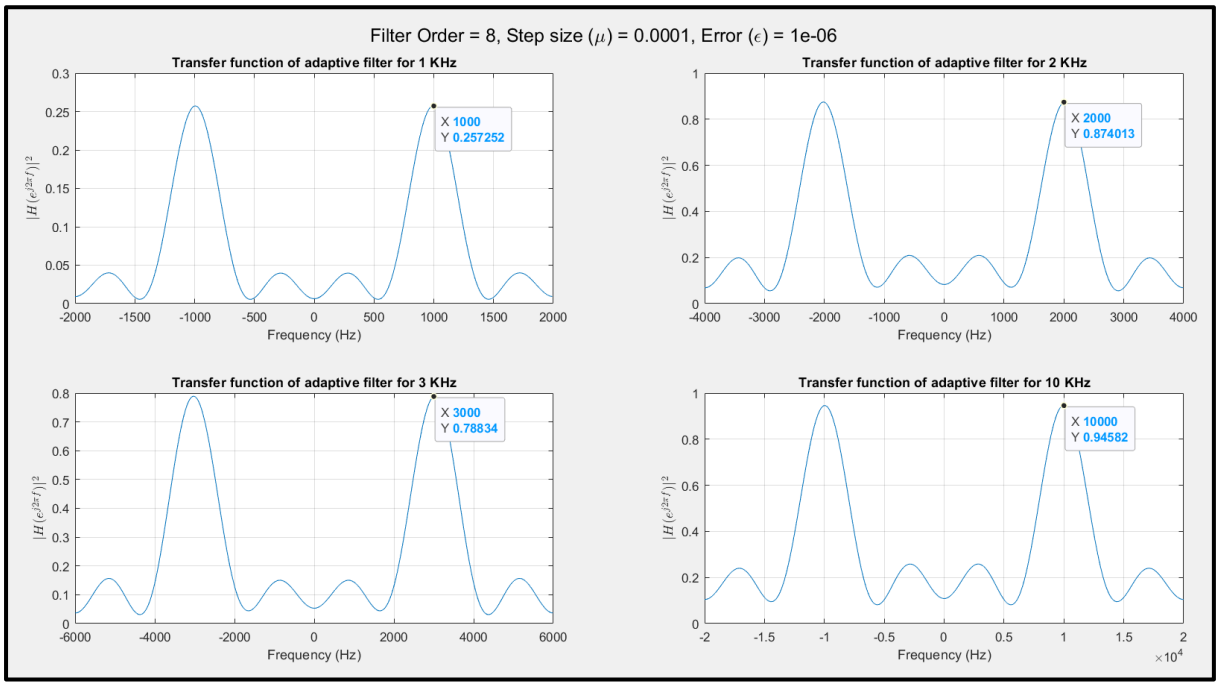
Also, for some examples, the input and output signals are plotted in the time domain and frequency domain to see the effect of filtering. (Just a few examples since the plots for others will be similar. The code will generate all the examples for a set value of hyperparameters.)

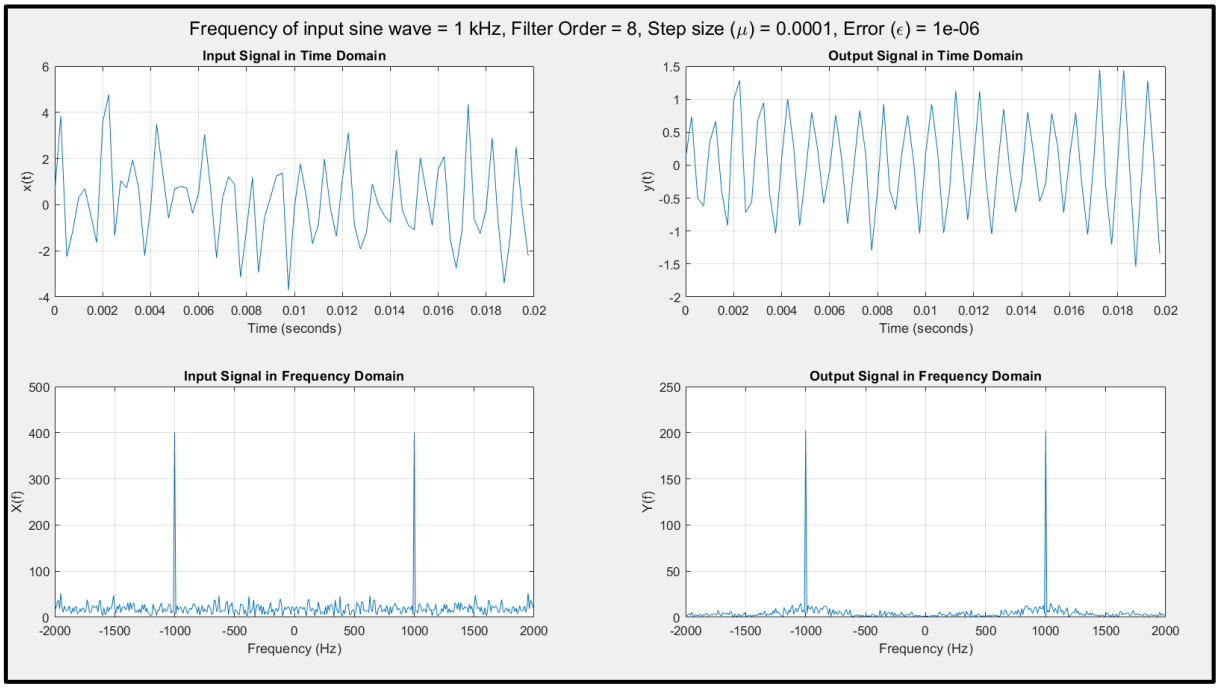
Simulation Results:

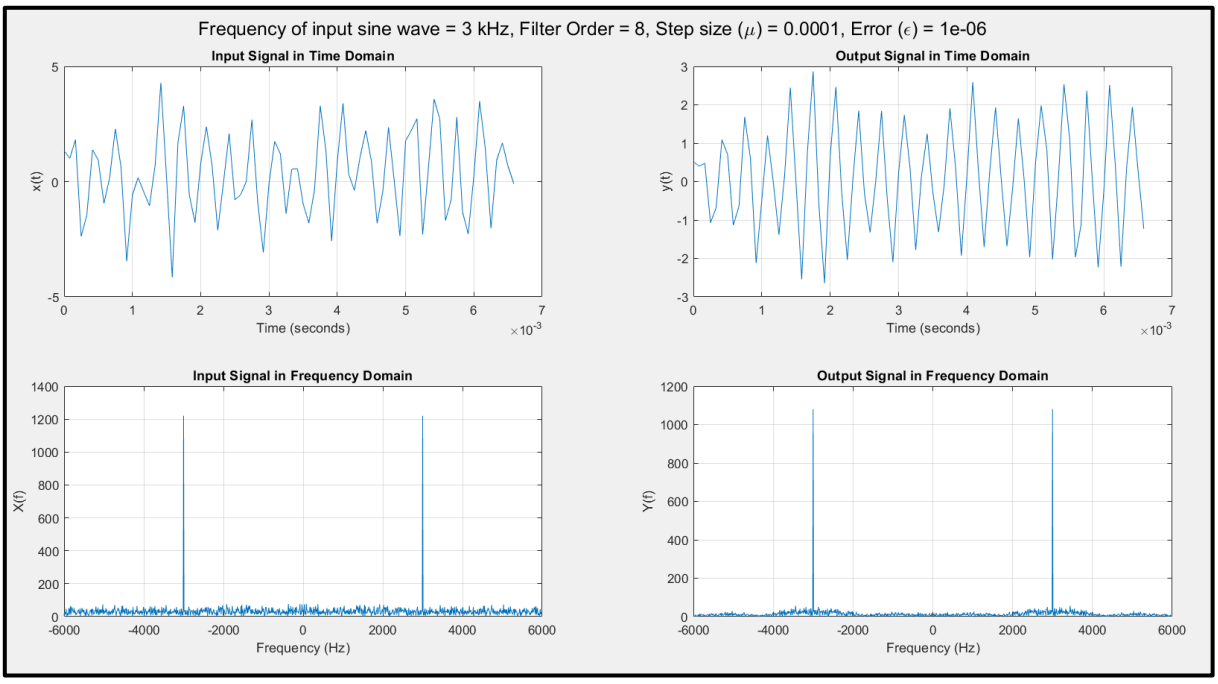





Filter Order = 8, Step size (μ) = 1e-06, Error (ϵ) = 0.001
Transfer function of adaptive filter for 1 KHz
X 1000
Y 1.24066e-05
Transfer function of adaptive filter for 2 KHz
X 2000
Y 4.65561e-05
Transfer function of adaptive filter for 3 KHz
X 3000
Y 0.00011796
Transfer function of adaptive filter for 10 KHz
X 10000
Y 0.00122244
Frequency (Hz)


Filter Order = 8, Step size (μ) = 0.0001, Error (ϵ) = 1e-06
Transfer function of adaptive filter for 1 KHz
X 1000
Y 0.257252
Transfer function of adaptive filter for 2 KHz
X 2000
Y 0.874013
Transfer function of adaptive filter for 3 KHz
X 3000
Y 0.78834
Transfer function of adaptive filter for 10 KHz
X 10000
Y 0.94582
Frequency (Hz)


Frequency of input sine wave = 1 kHz, Filter Order = 8, Step size (μ) = 0.0001, Error (ϵ) = 1e-06
Input Signal in Time Domain
x(t)
Time (seconds)
Output Signal in Time Domain
y(t)
Time (seconds)
Input Signal in Frequency Domain
X(f)
Frequency (Hz)
Output Signal in Frequency Domain
Y(f)
Frequency (Hz)


Frequency of input sine wave = 3 kHz, Filter Order = 8, Step size (μ) = 0.0001, Error (ϵ) = 1e-06
Input Signal in Time Domain
x(t)
Time (seconds)
Output Signal in Time Domain
y(t)
Time (seconds)
Input Signal in Frequency Domain
X(f)
Frequency (Hz)
Output Signal in Frequency Domain
Y(f)
Frequency (Hz)

Results:

The Adaptive Line Enhancer designed using the Least Mean Square Algorithm was able to adapt its filter coefficient in order to transform into a band-pass filter centred at the frequency of the input signal. The bandwidth, magnitude, and number of side lobes of the filter and its transfer function were seen to be varying with the hyperparameters.

Discussion:

An Adaptive Line Enhancer is an adaptive filter, capable of detecting a low-level sine wave of unknown frequency in the presence of noise. From the plots of the transfer function of the ALE for inputs of varying frequency shows that it is a band-pass filter centred around the frequency of the sine wave present in the input signal. The number of peaks, the magnitude of the highest peak, and the width of the pass-band vary on the different hyperparameters, but the centre of the pass-band is just dependent on the frequency of the input sine wave. The time and frequency domain plots of the input and the output reinforce this conclusion since the time domain output now varies more uniformly, whereas the frequencies introduced due to noise are highly attenuated in the frequency domain (except those close to the frequency of the sine wave). This proves that the ALE works as desired.

The effect of varying hyperparameters on the training of the ALE was as follows:

1. Varying the filter order changes the number of peaks and the bandwidth in the frequency response of the ALE. Decreasing the filter order reduces the number of peaks, which is desirable since unwanted frequency components away from the centre of pass band will be attenuated further. But this increases the bandwidth, which is undesirable since the noise components near the pass-band centre will pass more easily. A compromise must be made based on the requirements and limitations of design.
2. Varying the rate of convergence (step size) varies the amplitude of the peaks of the frequency response of the ALE. The reason for this is that the change brought to the coefficients is lesser

for a lower rate of convergence results in the training period getting over as soon as the error threshold is reached, whereas for a larger step size the final result might give an error significantly lesser than the threshold. A decision regarding the rate of convergence is made based on the number of iterations and guarantee of convergence. A greater value of μ will result in faster convergence (lesser iterations), but increasing the value indefinitely might cause the result to diverge rather than converge.

3. Varying the error threshold also varies the amplitude of the frequency response. Lowering the value of the threshold increases the amplitude, which means that the pass-band attenuation is reduced, which is desirable. However, a lower threshold means more iterations must be made to meet the convergence criteria, so another compromise must be made while deciding the value of epsilon.

Appendix

Code:

```
clc
clear all
close all

m = 8; %Filter order
mu = 1e-4; %Convergence rate (step size)
epsilon = 1e-3; % Error threshold

rng('default'); %For reproducible random numbers
cntr = 1; % For plotting all transfer functions in one figure

for k = [1,2,3,10] %%Choosing frequencies in kHz
f = k*1000;
fs = 4*f; %Sampling above Nyquist rate
t = 0:1/fs:0.1-1/fs;
N = length(t); % for N-point FFT
f_range = -fs/2:fs/N:fs/2-fs/N;

% Generating noisy sine wave
x = 2*sin(2*pi*f*t)';
noise = randn(size(x));
x = x+noise;
x_n = buffer(x, m, m-1);
x_n = flip(x_n, 1);

% Initialising transfer function of ALE
h = zeros(m,1);
h_new = zeros(m,1);

% Passing the complete signal once
for ii = 1:size(x_n,2) %Passing vector of each sample
   h = h_new;
   y = x_n(:,ii)'*h; %Filtering (convolution)
   diff = x(ii)-y; %difference between desired and actual output
   update = x_n(:, ii)*diff;
   h_new = h + mu*update; %updating filter coefficients
   change = sum((h_new-h).^2)/sum(h.^2); %Finding the relative change
end

while change >= epsilon
   % Passing the complete signal till change >= error threshold
   for ii = 1:size(x_n,2) %Passing vector of each sample
      h = h_new;
      y = x_n(:,ii)'*h; %Filtering (convolution)
      diff = x(ii)-y; %difference between desired and actual output
      update = x_n(:, ii)*diff;
      h_new = h + mu*update; %updating filter coefficients
      change = sum((h_new-h).^2)/sum(h.^2); %Finding the relative change
   end
end

%Plotting transfer function of the filter
spectrum = fftshift(abs(fft(h, N)).^2);
figure(1);
sgtitle("Filter Order = "+m+", Step size (\mu) = "+mu+", Error (\epsilon) = "+epsilon);
```

```
subplot(2,2,cntr);
cntr = cntr+1;
plot(f_range, spectrum);
grid on;
xlabel('Frequency (Hz)'); title("Transfer function of adaptive filter for "+k+" KHz")
ylabel('$|H(e^{j2\pi f})|^2$', 'Interpreter', 'latex');

%Plotting input and output signals in time and frequency domain
figure();
sgtitle("Frequency of input sine wave = "+k+" kHz, Filter Order = "+m+", Step size (\mu) = "+mu+", Error
(\epsilon) = "+epsilon);
subplot(2,2,1);
plot(t(1:20*fs/f),x(1:20*fs/f));
grid on;
xlabel("Time (seconds)"); ylabel("x(t)"); title("Input Signal in Time Domain");

subplot(2,2,2);
y = x_n'*h;
plot(t(1:20*fs/f),y(1:20*fs/f));
grid on;
xlabel("Time (seconds)"); ylabel("y(t)"); title("Output Signal in Time Domain");

subplot(2,2,3);
plot(f_range, fftshift(abs(fft(x))));
grid on;
xlabel("Frequency (Hz)"); ylabel("X(f)"); title("Input Signal in Frequency Domain");

subplot(2,2,4);
plot(f_range, fftshift(abs(fft(y))));
grid on;
xlabel("Frequency (Hz)"); ylabel("Y(f)"); title("Output Signal in Frequency Domain");
end
```