3TP3 Lab 2 Report

Rui Qiu (400318681)

Shabab Rahman (400291441)

Instructor: Terry Todd

Section: T01

Q1:

Do Question 2.7 from the textbook. In this question you are given three sets of discrete-time signals x[n] and v[n]. In Part (a) you manually compute the convolutions of x[n] and v[n]. Then in Part (b) you use the Matlab conv function to verify your results. In Part (b), write Matlab code that do stem plots of x[n], v[n] and x[n] * v[n] (Use the subplot command to plot them above each other). Include the Matlab code and your plots in your writeup.

2.7. For the discrete-time signals $x[n]$ and $v[n]$ shown in Figure P2.7, do the following:

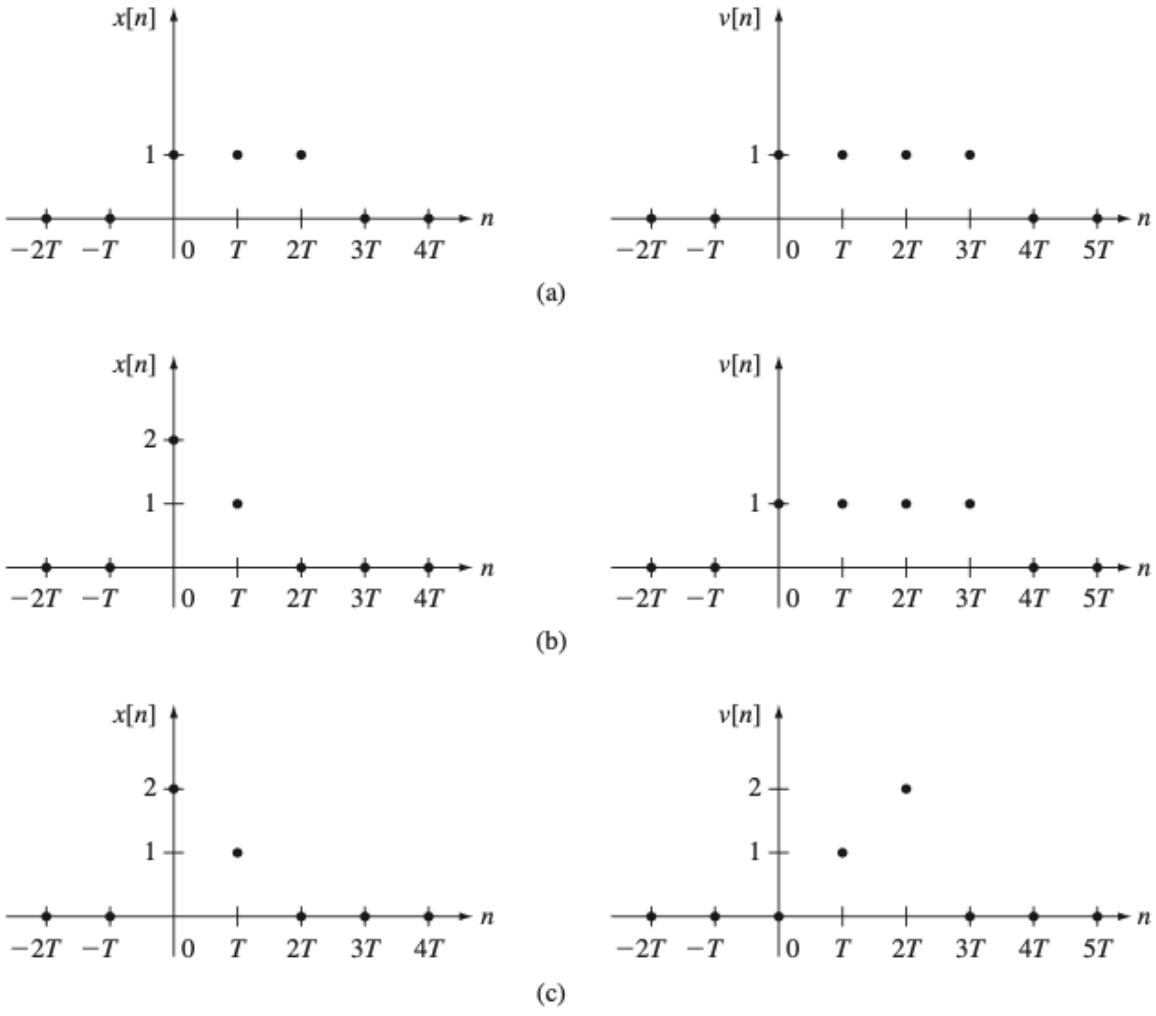
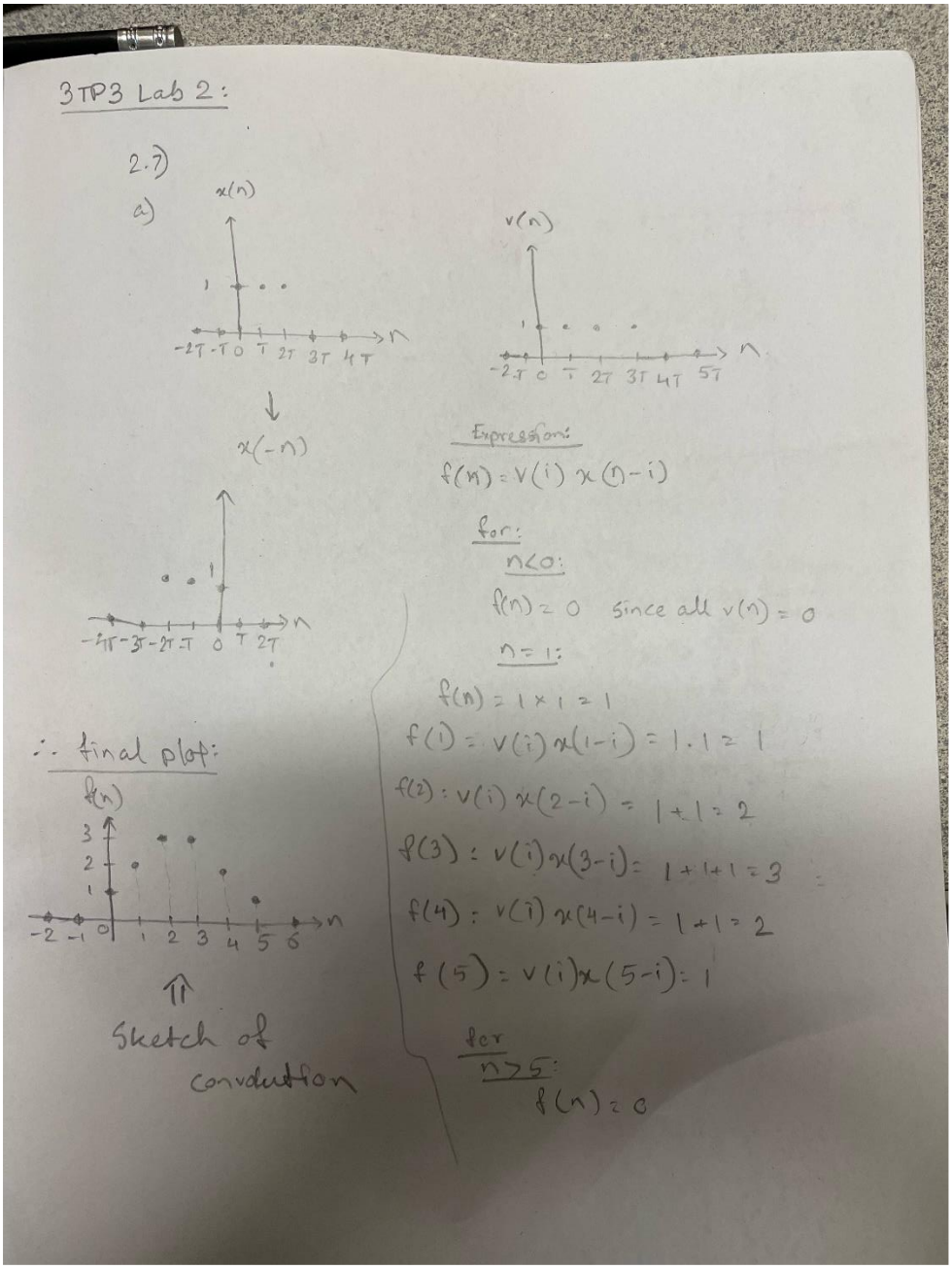


FIGURE P2.7

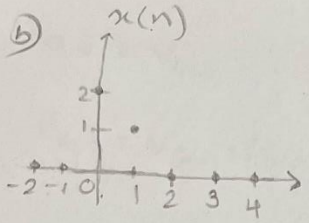
(a) Compute the convolution $x[n] * v[n]$ for all $n \geq 0$. Sketch the results.
(b) Repeat part (a), using the M-file conv.

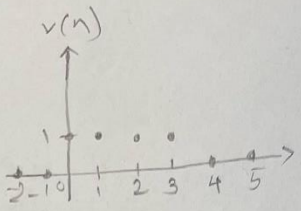
Part 1:

Written Work:



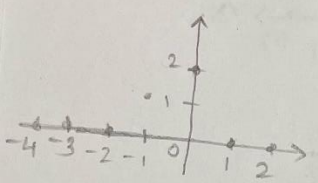
b)





$\Downarrow$

$x(-n)$



for $n < 0$:
$f(n) = 0$

$f(n)$:

$f(0) = v(i)\,x(0-i) = 2$

$f(1) = v(i)\,x(1-i) = 2+1 = 3$

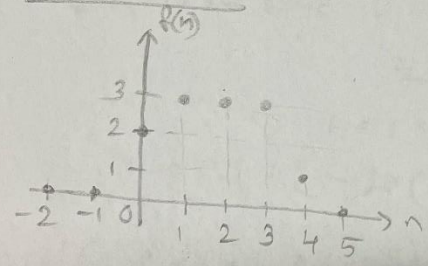
$f(2) = v(i)\,x(2-i) = 2+1 = 3$

$f(3) = v(i)\,x(3-i) = 2+1 = 3$

$f(4) = v(i)\,x(4-i) = 1$

$f(5 \text{ or } >) = 0$

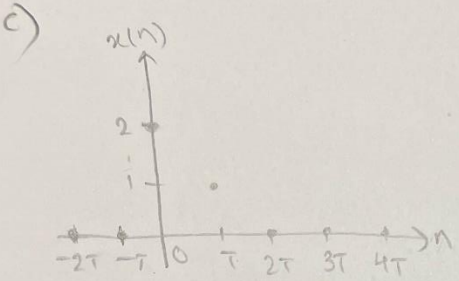
$\therefore$ final plot:

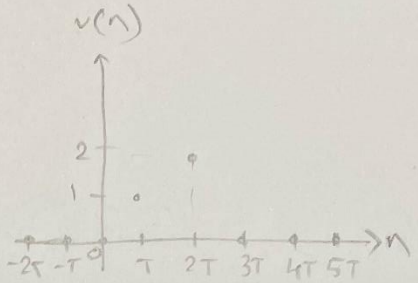


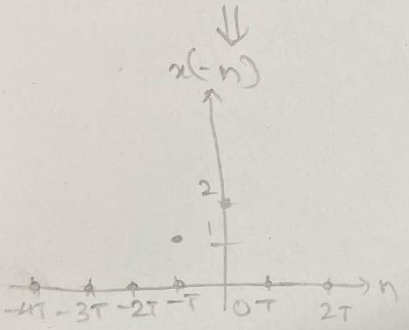
$\Uparrow$

Sketch of convolution

c)







f(n):

f(n≤0) = 0 since v(n) = 0

f(1) = v(i) x(1-i) = 2×1 = 2

f(2) = v(i) x(2-i) = 2×2+1×1 = 5

f(3) = v(i) x(3-i) = 2×1+0 = 2

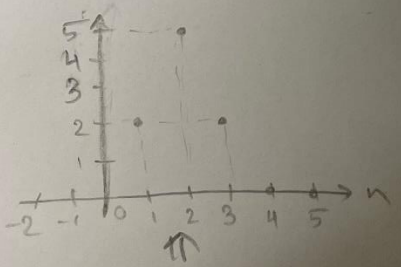
f(4) = v(i) x(4-i) = 0

f(5or>) = 0

↑ no intersection

final plot:

f(n)



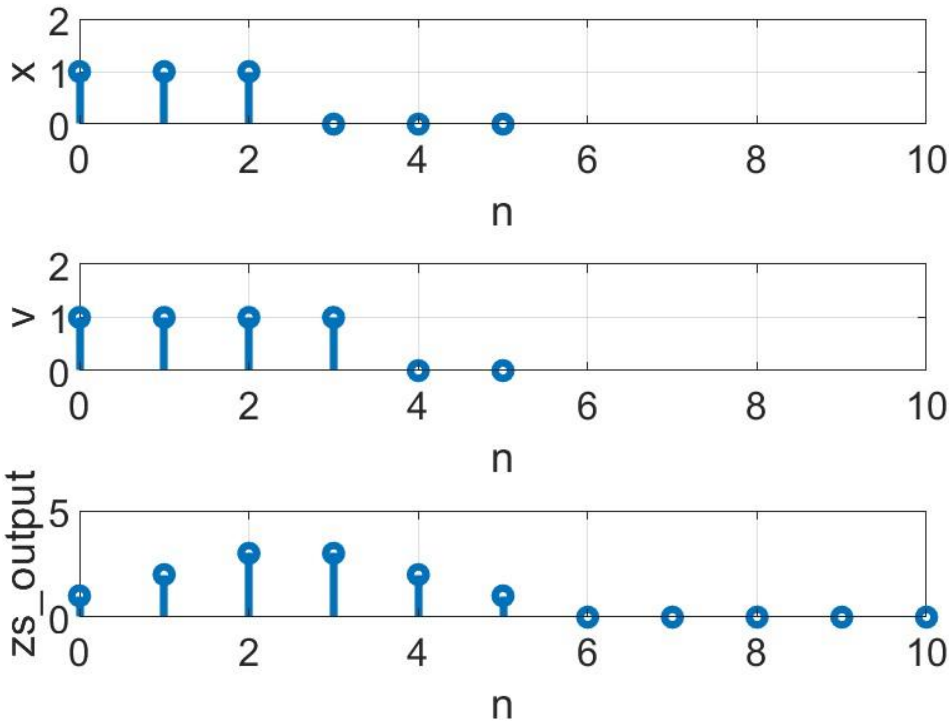
↑↑ sketch of convolution

Q:2.7:

a) Code:

```
%% (a)
%% x = [1 1 1]
%% v = [1 1 1 1]
%%
clear
f = SimpleFunctions();

% Assume time always starts at 0 for both signals.
n = 0:5;

x = f.unitstep(n) - f.unitstep(n-3);
v = f.unitstep(n) - f.unitstep(n-4);
% x and v have the same length.

% Create a correct discrete time vector.
zs_output = conv(x, v);
disp(zs_output)

time_axis = [0 length(zs_output)-1];

set(groot, 'DefaultStemLineWidth', 3)
set(groot, 'defaultAxesFontSize', 18)

subplot(3, 1, 1)
stem(n, x)
xlabel("n")
ylabel("x")
axis([time_axis 0 max(x)+1])
grid on;

subplot(3, 1, 2)
stem(n, v)
xlabel("n")
ylabel("v")
axis([time_axis 0 max(v)+1])
grid on;

subplot(3, 1, 3)
stem(0:length(zs_output)-1, zs_output)
xlabel("n")
ylabel("zs\_output")
axis([time_axis 0 max(zs_output)+2])
grid on;

exportgraphics(gcf, 'Q1a.jpg');
```

Plot:



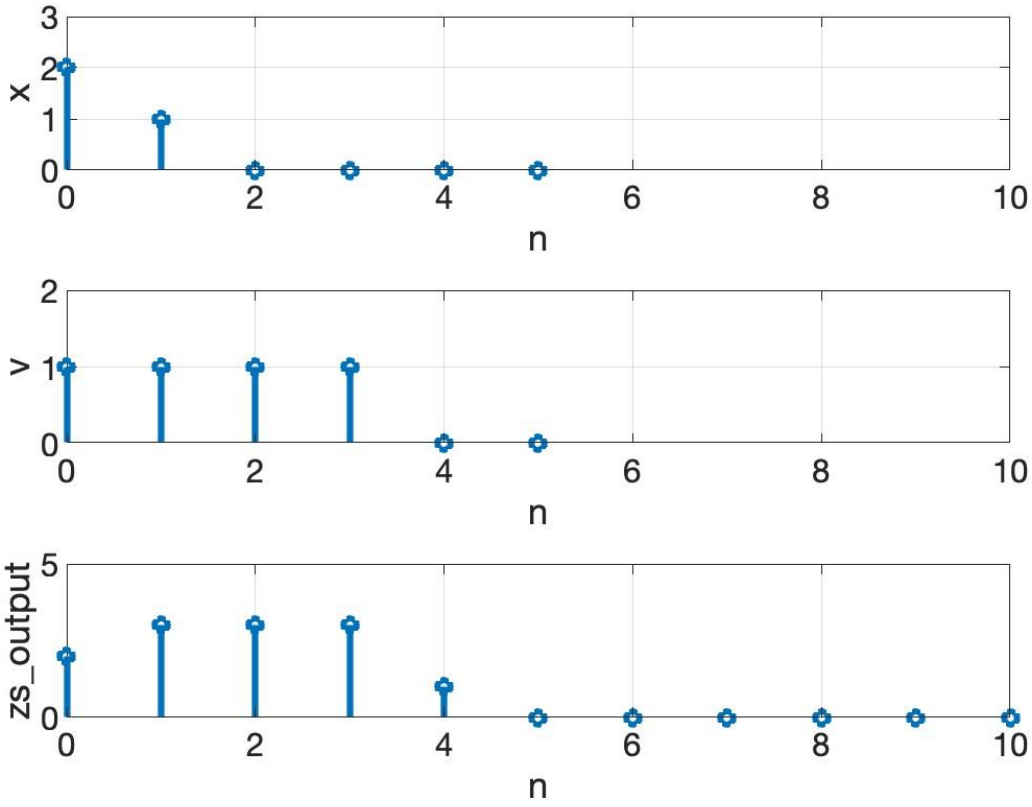
b)

Code:

```
%% (b)
%% x = [2 1]
%% v = [1 1 1 1]
%%
clear
f = SimpleFunctions();

% Assume time always starts at 0 for both signals.
n = 0:5;

x = 2*f.delta(n)+f.delta(n-1);
v = f.unitstep(n) - f.unitstep(n-4);
% x and v have the same length.

% Create a correct discrete time vector.
zs_output = conv(x, v);
disp(zs_output)

time_axis = [0 length(zs_output)-1];

set(groot, 'DefaultStemLineWidth', 3)
set(groot, 'defaultAxesFontSize', 18)

subplot(3, 1, 1)
stem(n, x)
xlabel("n")
```

```
ylabel("x")
axis([time_axis 0 max(x)+1])
grid on;

subplot(3, 1, 2)
stem(n, v)
xlabel("n")
ylabel("v")
axis([time_axis 0 max(v)+1])
grid on;

subplot(3, 1, 3)
stem(0:length(zs_output)-1, zs_output)
xlabel("n")
ylabel("zs\_output")
axis([time_axis 0 max(zs_output)+2])
grid on;
exportgraphics(gcf, 'Q1b.jpg');
```

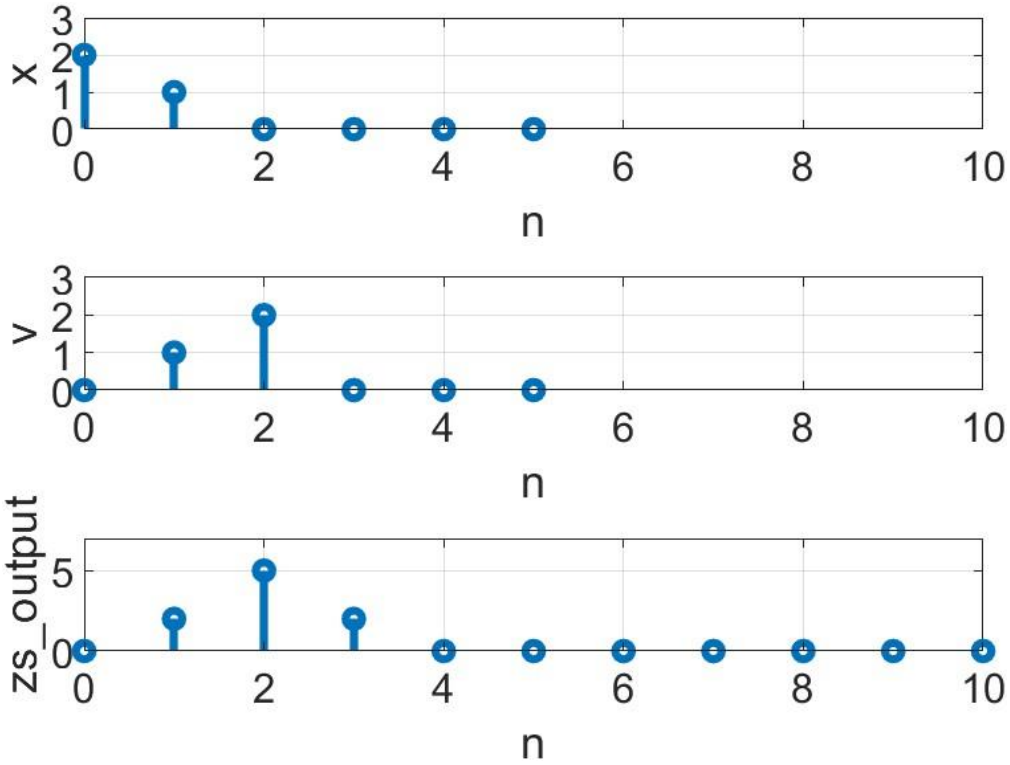
Plot:



c)

```
%% (b)
%% x = [2 1]
%% v = [1 2]
%%
clear
f = SimpleFunctions();

% Assume time always starts at 0 for both signals.
n = 0:5;

x = 2*f.delta(n)+f.delta(n-1);
v = f.delta(n-1)+2 .* f.delta(n-2);
% x and v have the same length.

% Create a correct discrete time vector.
zs_output = conv(x, v);
disp(zs_output)

time_axis = [0 length(zs_output)-1];

set(groot, 'DefaultStemLineWidth', 3)
set(groot, 'defaultAxesFontSize', 18)

subplot(3, 1, 1)
stem(n, x)
xlabel("n")
ylabel("x")
axis([time_axis 0 max(x)+1])
grid on;

subplot(3, 1, 2)
stem(n, v)
xlabel("n")
ylabel("v")
axis([time_axis 0 max(v)+1])
grid on;

subplot(3, 1, 3)
stem(0:length(zs_output)-1, zs_output)
xlabel("n")
ylabel("zs\_output")
axis([time_axis 0 max(zs_output)+2])
grid on;

exportgraphics(gcf, 'Q1c.jpg');
```

Plot:



Parts 2 and 3: Done as instructed in the manual.

Part 4)
Code:

```
clear
clc

[signal, Fs] = audioread('my_speech_clip.wav'); %signal is a column vector, Fs is
sampling rate
L = length(signal);  % Number of samples in the signal.
T = 1/Fs;            % Sampling period in seconds.
t = (0:L-1)*T;       % Time vector in seconds.


Te = 0.8;          %echo delayed in msec
amp = 0.4;
signalplusecho = echo_gen(signal,Fs,Te,amp);
audiowrite('speechwithecho.wav', signalplusecho, Fs);

function signalplusecho = echo_gen(signal,Fs,delay,amp)
    T = 1/Fs;       % Sampling period in seconds.
    % Calculate the increased sample spaces for the given delay
```

```
    N = round(delay/T);
    %create the original signal with alloacted spaces of the new dealyed signal equal
to zero
    signal_newspace = [signal;zeros(N,1)];
    %create the delayed signal with the delays equal to zero at the front and pad the
original signal multiply with the amplitude
    signal_echodelay = [zeros(N,1);amp.*signal];
    %added two signals
    signalplusecho = signal_newspace+signal_echodelay;
    %ensure the magnitude of any sample spaces does not exceed 1
    if max(abs(signalplusecho))>1
        signalplusecho = signalplusecho/max(abs(signalplusecho));
    end
end
```

Part 5:

Explanation: The choice of impulse response chosen is to set a delta pulse with delayed signal sample space index multiply by the amplitude to get a delayed signal. After generating the delayed signal, by assigning the extra spaces for the original signal, both signals can perform addition to create an echo effect.

Code:

```
clear
clc

[signal, Fs] = audioread('my_speech_clip.wav'); %signal is a column vector, Fs is
sampling rate
% L = length(signal);  % Number of samples in the signal.
% T = 1/Fs;            % Sampling period in seconds.
% t = (0:L-1)*T;       % Time vector in seconds.


Te = 0.8;              % echo delayed in sec
delayed_space=Te*Fs;   % Delayed sample spaces


amp = 0.4;               % amplitude
% delayes_space + 1 is to match the correct timing convolution
h = zeros(1,delayed_space+1);    % create the spaces of the extra delayed samples
h(delayed_space+1) = amp;        % set the Impulse Response of the delayed signal for
convolution with the amplitude
dealyed_signal = conv(signal,h);      % Convolution of the origional signal and the
delayed impulse response


signal_newspace = zeros(length(dealyed_signal),1); % create the same length vector
space as the delayed signal
signal_newspace(1:length(signal)) = signal; % padding the original signal into the
new space
```

```
signalplusecho = signal_newspace + dealyed_signal; % adding both signals

audiowrite('Q5.wav', signalplusecho, Fs);
```

Part 6:
Code:

```
clear
clc

[signal, Fs] = audioread('my_speech_clip.wav'); %signal is a column vector, Fs is
sampling rate
% L = length(signal);  % Number of samples in the signal.
% T = 1/Fs;            % Sampling period in seconds.
% t = (0:L-1)*T;       % Time vector in seconds.


Te = 20e-3;              % echo delayed in sec
delayed_space=round(Te*Fs);   % Delayed sample spaces


amp = 1;               % amplitude
% delayes_space + 1 is to match the correct timing convolution
h = zeros(1,delayed_space+1);    % create the spaces of the extra delayed samples
h(delayed_space+1) = amp;        % set the Impulse Response of the delayed signal for
convolution with the amplitude
dealyed_signal = conv(signal,h);      % Convolution of the origional signal and the
delayed impulse response


signal_newspace = zeros(length(dealyed_signal),1); % create the same length vector
space as the delayed signal
signal_newspace(1:length(signal)) = signal; % padding the original signal into the
new space

signalplusecho = signal_newspace + dealyed_signal; % adding both signals

audiowrite('Q6.wav', signalplusecho, Fs);
```

Explanation:
By experimenting testing with different values of Te and alpha. The acceptable quality of speech for the Te when alpha is equal to 1 is Te = 20ms. The decreasing alpha will increase the acceptable time delay(Te) for the quality of speech.

Part 7:

Code:

```
clear
clc

[signal, Fs] = audioread('my_speech_clip.wav'); %signal is a column vector, Fs is
sampling rate
L = length(signal);  % Number of samples in the signal.
```

```
T = 1/Fs;             % Sampling period in seconds.
t = (0:L-1)*T;        % Time vector in seconds.


Te = 0.08;            %echo delayed in msec
amp = 0.5;           % amplitude
Ne = 3;

for i = 1:Ne

    delayed_space=round(i * Te*Fs);   % Delayed sample spaces
    disp(delayed_space);
    h = zeros(1,delayed_space+1);    % create the spaces of the extra delayed samples
    h(delayed_space+1) = amp.^i;        % set the Impulse Response of the delayed
signal for convolution with the amplitude
    delayed_signal = conv(signal,h);      % Convolution of the origional signal and
the delayed impulse response

    if i == 1
        signal_newspace = [signal;zeros(delayed_space,1)];
    else
        signal_newspace = zeros(length(delayed_signal),1); % create the same length
vector space as the delayed signal
        signal_newspace(1:length(signal_reverberation)) = signal_reverberation; %
padding the original signal into the new space
    end
    signal_reverberation = delayed_signal + signal_newspace;
end

audiowrite('Q7.wav', signal_reverberation, Fs);
```

Part7(q6)

When the alpha is equal to 1, the number of echoes is equal to 3, the time delay(Te) to make the quality of the speech acceptable is equal to 20ms. The decreasing of the alpha will increase the acceptable range of the time delay. The increasing number of echos will decrease the time delay to make the quality of the speech acceptable and vice versa.