ELECENG 3TP3

Lab #2 Report

Instructor: Dr. Kiruba

Jasmine Dosanjh – dosanj5 – 400531879

Warisha Noushad – noushadw – 400519903

November 4, 2025

Question 1: Convolution

Part 1. (a)

To manually find the convolution $y[n] = x[n] * v[n]$ for all $n \geq 0$, $v[m]$ was flipped to $v[-m]$ and shifted by n to get $v[n-m]$. Then element by element multiplication and summation were performed to sketch the final result $y[n]$.

(a)

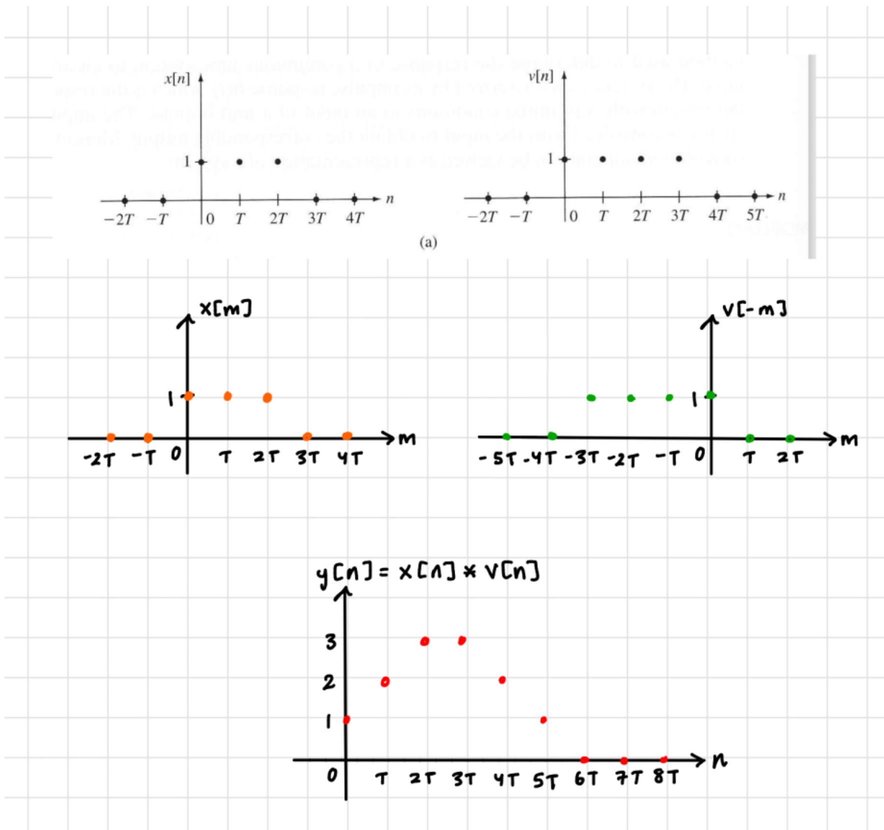

Figure 1. Convolution for (a)

$$y[0] = 1$$

$$y[T] = 1 + 1 = 2$$

$$y[2T] = 1 + 1 + 1 = 3$$

$$y[3T] = 1 + 1 + 1 = 3$$

$$y[4T] = 1 + 1 = 2$$

$$y[5T] = 1$$

$$y[n] = 0 \; otherwise$$

(b)

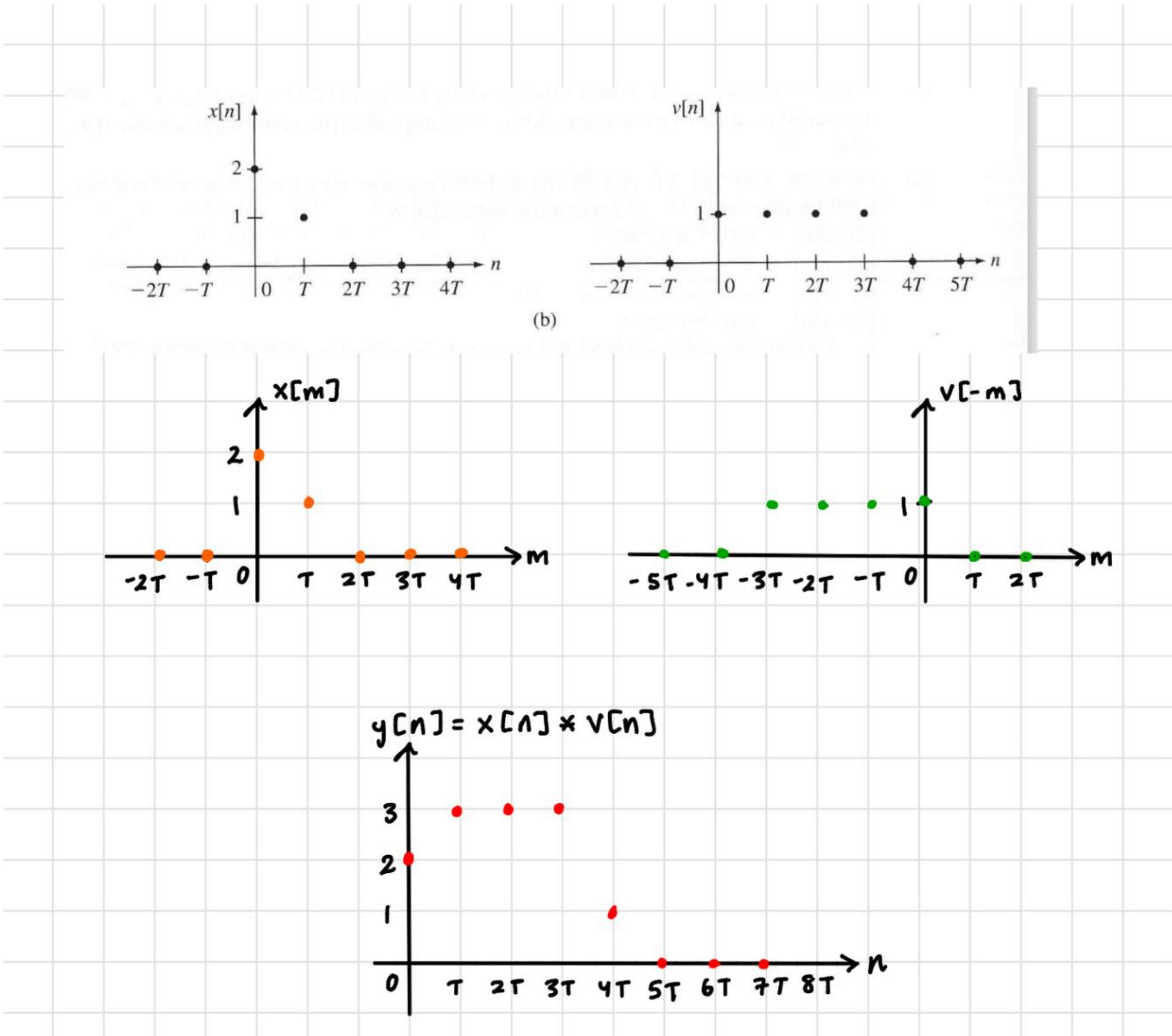

Figure 2. Convolution for (b)

$$y[0] = 1(2) = 2$$

$$y[T] = 1(2) + 1(1) = 3$$

$$y[2T] = 1(2) + 1(1) = 3$$

$$y[3T] = 1(2) + 1(1) = 3$$

$$y[4T] = 1(1) = 1$$

$$y[n] = 0 \; otherwise$$

(c)

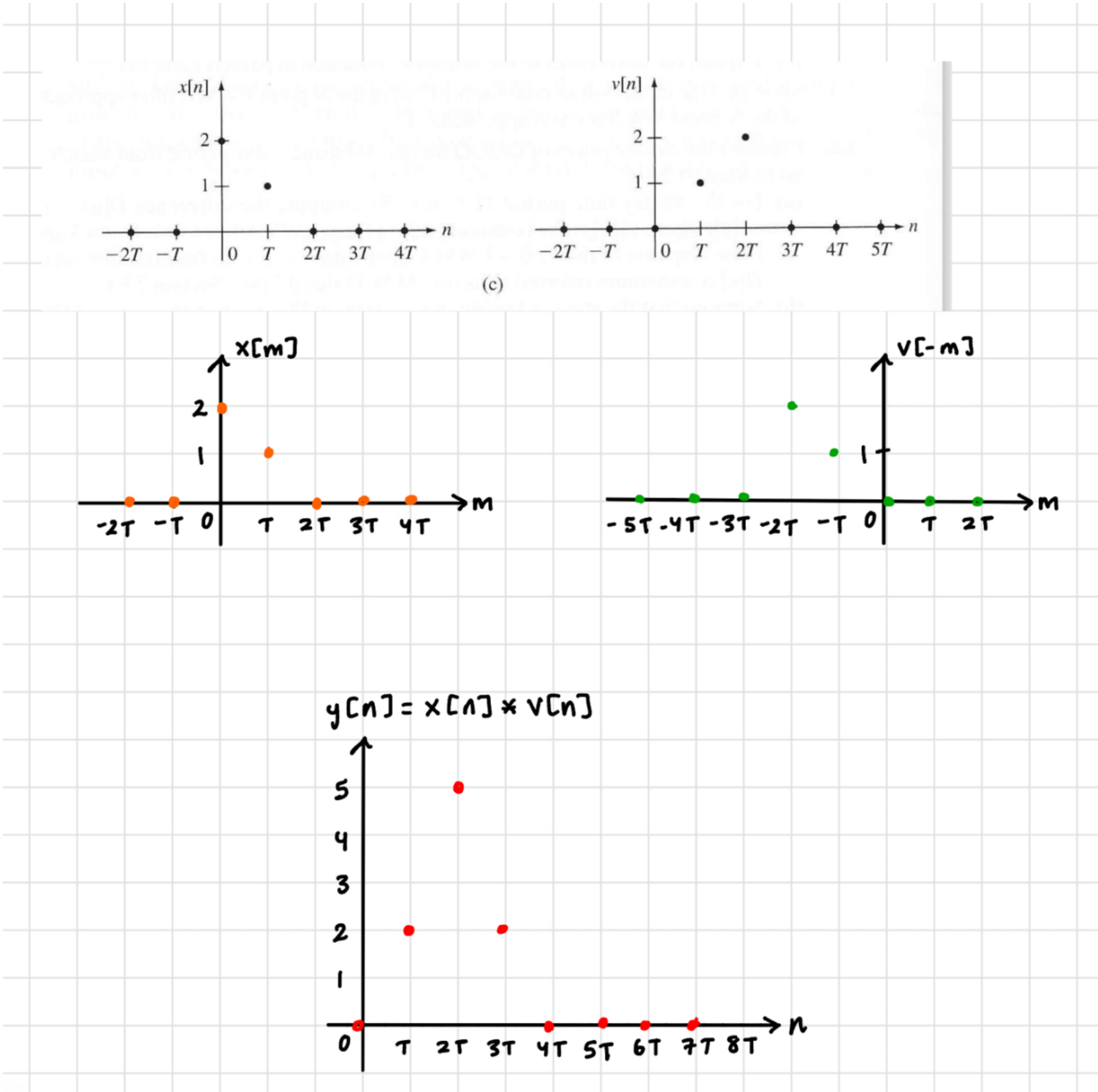

Figure 3. Convolution for (c)

$$y[T] = 1(2) = 2$$

$$y[T] = 2(2) + 1(1) = 5$$

$$y[2T] = 1(2) = 2$$

$$y[n] = 0\ otherwise$$

Part 1. (b)

The MATLAB code to verify the results from Part (a) is included below. The script assigns the data for each signal into a vector and finds the convolution of the vectors $x[n] * v[n]$.

Listing 1: Assigning Variables

```
xa = [1 1 1]; xb = [2 1]; xc = xb;
va = [1 1 1 1]; vb = va; vc = [0 1 2];

ya = conv(xa,va);
yb = conv(xb,vb);
yc = conv(xc,vc);
```

The MATLAB code to plot and export each vector is included below. It uses the MATLAB subplot feature to output a plot per vector and display it.

Listing 2: Stem Plots

```
t = tiledlayout(3, 3);
title(t, {'Jasmine Dosanjh 400531879', 'Warisha Noushad 400519903'});

nexttile; % Plot 1
stem(0:length(xa)-1,xa); title('(a) x[n]'); axis([0 3 0 2]);

nexttile; % Plot 2
stem(0:length(va)-1,va); title('v[n]'); axis([0 4 0 2]);

nexttile; % Plot 3
stem(0:length(ya)-1,ya, 'r'); title('x[n]*v[n]'); axis([0 6 0 4]);

nexttile; % Plot 4
stem(0:length(xb)-1,xb); title('(b) x[n]'); axis([0 2 0 3]);

nexttile; % Plot 5
stem(0:length(vb)-1,vb); title('v[n]'); axis([0 4 0 2]);

nexttile; % Plot 6
stem(0:length(yb)-1,yb, 'r'); title('x[n]*v[n]'); axis([0 5 0 4]);

nexttile; % Plot 7
stem(0:length(xc)-1,xc); title('(c) x[n]');  axis([0 2 0 3]);

nexttile; % Plot 8
stem(0:length(vc)-1,vc); title('v[n]'); axis([0 3 0 3]);

nexttile; % Plot 9
stem(0:length(yc)-1,yc,'r'); title('x[n]*v[n]'); axis([0 4 0 6]);
yticks(0:2:6); % Force 0,1,2,3,4,5

exportgraphics(gcf, 'Q1_stem_plots.jpg');
```

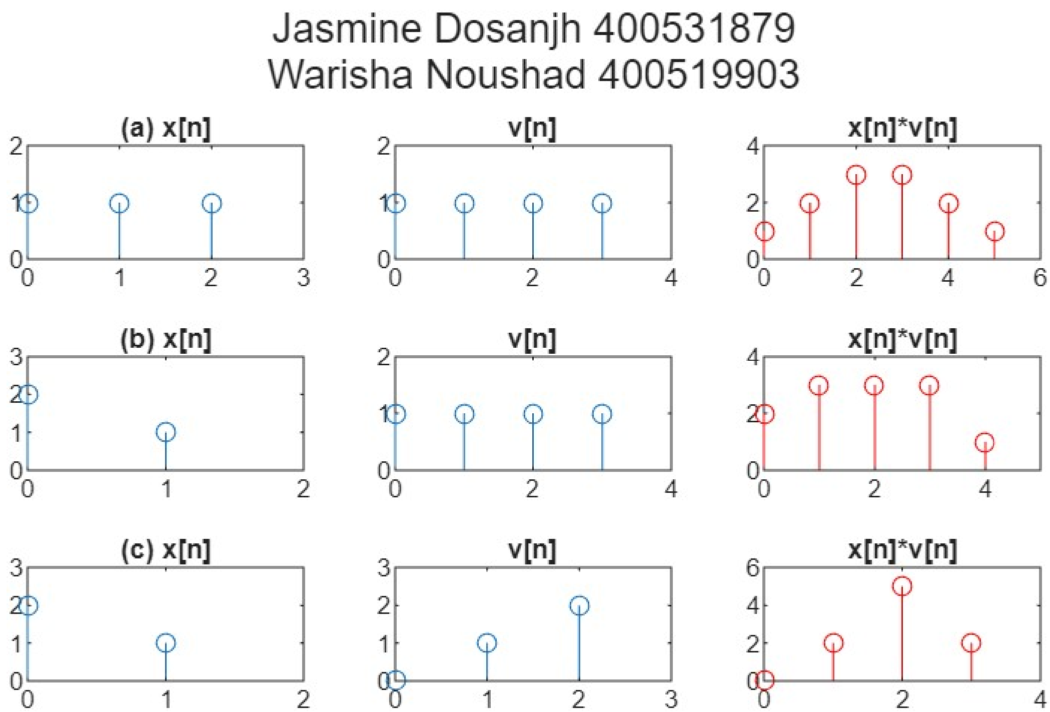

Figure 4. Part 1b Stem Plots

Question 2: Audio Recording

4. The MATLAB code is included below; it generates the following distorted signal:

$$f_r(t) = f_s(t) + af_s(t - t_e) \tag{1}$$

First, the audio file was stored in the vector signal, with its sample frequency set to Fs. The signal collapsed from a stereo vector (`688128×2`) to a mono vector (`688128×1`).

Listing 3: Original Signal

```
1. [signal, Fs] = audioread('my_speech_clip.wav')
2. signal = mean(signal, 2);
3.
4. L = length(signal) % Number of samples in the signal.
5. T = 1/Fs            % Sampling period in seconds.
6. t = [0:L-1]*T    % Time vector in seconds.
```

Next, the echo delay was set to 1 second. Using the echo delay value, the number of samples being delayed `L_delay` was found. The echo volume alpha was set to half of the original. A vector called `signalplusecho` was created, which makes a zero-filled vector large enough to hold both the original and delayed echo. The original signal was copied into samples 1 to L of `signalplusecho`. Finally, the scaled copy of the signal was inserted with a 1-second delay, which created the distorted echo effect. The output was rescaled to avoid clipping and then exported.

Listing 4: Delayed Signal

```
1. Te = 1000 % Echo delay in msec
2. Te_sec = Te / 1000
3. alpha = 0.5; % echo volume
4.
5. L_delay = round(Te_sec * Fs) % Number of samples being delayed
6. signalplusecho = zeros(L+L_delay, 1) % create vector
7. signalplusecho(1:L) = signal   % insert original signal
8. signalplusecho(L_delay+1:end) = signalplusecho(L_delay+1:end) +
alpha*signal; % add delayed echo
11. signalplusecho = signalplusecho/max(abs(signalplusecho)); % re-scale to
avoid clipping
12. audiowrite('speechwithecho.wav', signalplusecho, Fs) % new wav file
```

5. From Question 4, equation 1 has the following impulse response:

$$h[\mathrm{n}] = \delta[n] + a\delta(n - n_e) \tag{2}$$

This code creates the same distortion effect using convolution. Lines 1-8 remain the same as Question 4. A vector `IR` was created, which makes a zero-filled vector. At index 1, the original signal $\delta[n]$ is placed, and since the amplitude of the Kronecker delta is 1, then `IR=[1,0,0…]`. At index `L_delay+1,` alpha is placed, which represents the echo delayed by `L_delay` samples `IR=[1,0,0...0.5]`. Finally, the original and echoed signals are convolved, which combines the original signal and its delayed version. Again, the output was rescaled to avoid clipping and then exported.

Listing 5: Distorted Signal Using Convolution

```
[signal, Fs] = audioread('my_speech_clip.wav') % Read in og signal
signal = mean(signal, 2);

Te = 1000 % Echo delay in msec
Te_sec = Te / 1000
alpha = 0.5;  % echo volume (50% of og volume)

L_delay = round(Te_sec * Fs) % Number of samples being delayed

IR = zeros(L_delay+1, 1) % create impulse response vector
IR(1) = 1 % Original signal δ[n]
IR(L_delay+1) = alpha % Echo signal α·δ[t-te]

conv_echo_signal = conv(signal, IR); % Convolves original & echoed signal

conv_echo_signal = conv_echo_signal/max(abs(conv_echo_signal)); % re-scale to avoid clipping
audiowrite('speechwithecho_conv.wav', conv_echo_signal, Fs); % new wav file
```

6. Through experimenting with different values of Te when alpha=1, it was determined that the quality of speech is acceptable around 10ms. At this small delay, the two signals blend into one signal. As alpha decreases, the echo becomes quieter, reducing the effect of the echo. At alpha=0.3, a higher value of Te=100ms is more tolerable. This shows that the relationship between alpha and Te is inversely related. A louder signal requires a smaller delay to blend into the original signal, whereas a quieter signal can have a longer delay as it's less noticeable.

   The MATLAB code below creates a reverberation effect by creating several echoes, each at a decreasing amplitude. This code is the same as Listing 5 but includes an additional variable `num_echos`, which stores the number of echoes and `IR` being generated differently. The size of `IR` changed from `IR` being `L_delay+1` to `num_echos*L_delay+1`, as multiple echoes need to be accounted for instead of one. The echoes are added in a for loop through which `alpha` exponentially decreases, and at each index `i*L_delay+1` another echo is inserted i.e., `IR=[1,0.5,0.25...]`.

i	Index = `i*L_delay+1`	Value stored in IR=$alpha^i$
0	1	1
1	2	0.5
2	3	0.25

Finally, the original and reverb signal are convolved, which combines the original signal and its multiple delays. Again, the output was rescaled to avoid clipping and then exported.

Listing 6: Reverberated Signal

```
[signal, Fs] = audioread('my_speech_clip.wav') % Read in og signal
signal = mean(signal, 2);

Te = 20 % Echo delay in msec
Te_sec = Te / 1000;
alpha = 0.5;  % echo volume (50% of og volume)
L_delay = round(Te_sec * Fs); % Number of samples being delayed

num_echos = 1000; % number of echos Ne

IR = zeros(num_echos*L_delay+1, 1); % create impulse response vector
IR(1) = 1 % Original signal δ[n]

for i=1:num_echos
    IR(i*L_delay+1) = alpha^i; % Echo signal α·δ[n – Delay]
end

signalplusreverb = conv(signal, IR); % Convolves original & echoed signal

signalplusreverb = signalplusreverb/max(abs(signalplusreverb)); % re-scale to avoid clipping
audiowrite('speechwithreverb_conv.wav', signalplusreverb, Fs); % new wav file
```

When `Ne=1`, the signal was the same as that in Question 6, and the observations about Te and alpha remained the same.

Through experimenting with different values of `Te`, when `alpha=1` and `Ne=1000,` it was determined that the quality of speech is acceptable around 0.002ms. At this small delay, the multiple signals blend into one signal. As `alpha` decreases, the echo becomes quieter, reducing the effect of the echo. At `alpha=0.3,` a higher value of `Te=200ms` is more tolerable.

This shows the relationship between `Ne`, `alpha,` and `Te`. Consistent with part 6, `Te` and `alpha` are inversely related, and increasing `Ne` just amplifies this effect. Multiple louder signals require a very small delay to blend them all into the original signal, whereas multiple quieter signals can have longer delays as the reverberation is less noticeable.