ELECENG 3TP3 Lab 2

Raeed Hassan
hassam41
McMaster University

December 12, 2020

1. Part a)

(a)

$$x[i] = [1, 1, 1],\ v[i] = [1, 1, 1, 1]$$
$$y[n] = \sum_{i=-\infty}^{\infty} x[i]\, v[n-i]$$
$$y[0] = x[0]\, v[0] = 1 \cdot 1 = 1$$
$$y[1] = x[0]\, v[1] + x[1]\, v[0] = 1 + 1 = 2$$
$$y[2] = x[0]\, v[2] + x[1]\, v[1] + x[2]\, v[0] = 3$$
$$y[3] = x[0]\, v[3] + x[1]\, v[2] + x[2]\, v[1] = 3$$
$$y[4] = x[1]\, v[3] + x[2]\, v[2] = 2$$
$$y[5] = x[2]\, v[3] = 1$$
$$y[6] = 0$$
$$y[n] = \begin{cases} 1, & n = 0, 5 \\ 2, & n = 1, 4 \\ 3, & n = 2, 3 \\ 0, & \text{otherwise} \end{cases}$$

(b)

$$x[i] = [2, 1],\ v[i] = [1, 1, 1, 1]$$
$$y[n] = \sum_{i=-\infty}^{\infty} x[i]\, v[n-i]$$
$$y[0] = x[0]\, v[0] = 2$$
$$y[1] = x[0]\, v[1] + x[1]\, v[0] = 2 + 1 = 3$$
$$y[2] = x[0]\, v[2] + x[1]\, v[1] = 3$$
$$y[3] = x[0]\, v[3] + x[1]\, v[2] = 3$$
$$y[4] = x[1]\, v[3] = 1$$
$$y[5] = 0$$
$$y[n] = \begin{cases} 1, & n = 4 \\ 2, & n = 0 \\ 3, & n = 1, 2, 3 \\ 0, & \text{otherwise} \end{cases}$$

(c)

$$x[i] = [2, 1],\ v[i] = [0, 1, 2]$$

$$y[n] = \sum_{i=-\infty}^{\infty} x[i]\, v[n-i]$$

$$y[0] = x[0]\, v[0] = 2 \cdot 0 = 0$$
$$y[1] = x[0]\, v[1] + x[1]\, v[0] = 2$$
$$y[2] = x[0]\, v[2] + x[1]\, v[1] = 5$$
$$y[3] = x[1]\, v[2] = 2$$
$$y[4] = 0$$

$$y[n] = \begin{cases} 2, & n = 1, 3 \\ 5, & n = 2 \\ 0, & \text{otherwise} \end{cases}$$

Part b)

The MATLAB code used to verify these results is shown in Listing 1. The script simply assigns the data for each signal into an array, and calculates the convolution of each pair of $x[n]$ and $v[n]$.

Listing 1: Part 1b

```
x_a = [1 1 1]; x_b = [2 1]; x_c = [2 1];
v_a = [1 1 1 1]; v_b = [1 1 1 1]; v_c = [0 1 2];

conv_a = conv(x_a,v_a);
conv_b = conv(x_b,v_b);
conv_c = conv(x_c,v_c);
```

The MATLAB code used to plot and export the stem plots is shown in Listing 2. The portion of the script creates subplots for each plot, then plots a stem plot into the subplot. Each stem plot has been given appropriate axis limits for viewing the data. The plots are shown in Figure 1.

Listing 2: Part 1b stem plots

```
fig = figure('Name','Question 1 Part B');
t = tiledlayout(3,3); title(t,'Raeed Hassan');
nexttile;
stem(0:length(x_a)-1,x_a); title('x[n]');axis([-1 3 -1 2]);
nexttile;
stem(0:length(x_b)-1,x_b); title('x[n]');axis([-1 2 -1 3]);
nexttile;
stem(0:length(x_c)-1,x_c); title('x[n]');axis([-1 2 -1 3]);
nexttile;
stem(0:length(v_a)-1,v_a); title('v[n]');axis([-1 4 -1 2])
```

```
nexttile;
stem(0:length(v_b)-1,v_b); title('v[n]');axis([-1 4 -1 2])
nexttile;
stem(0:length(v_c)-1,v_c); title('v[n]');axis([-1 3 -1 3])
nexttile;
stem(0:length(conv_a)-1,conv_a); title('x[n]*v[n]'); axis
   ([-1 6 -1 4]);
nexttile;
stem(0:length(conv_b)-1,conv_b); title('x[n]*v[n]'); axis
   ([-1 5 -1 4]);
nexttile;
stem(0:length(conv_c)-1,conv_c); title('x[n]*v[n]'); axis
   ([-1 4 -1 6]);
exportgraphics(fig,'..\Figures\question1b.png');
```

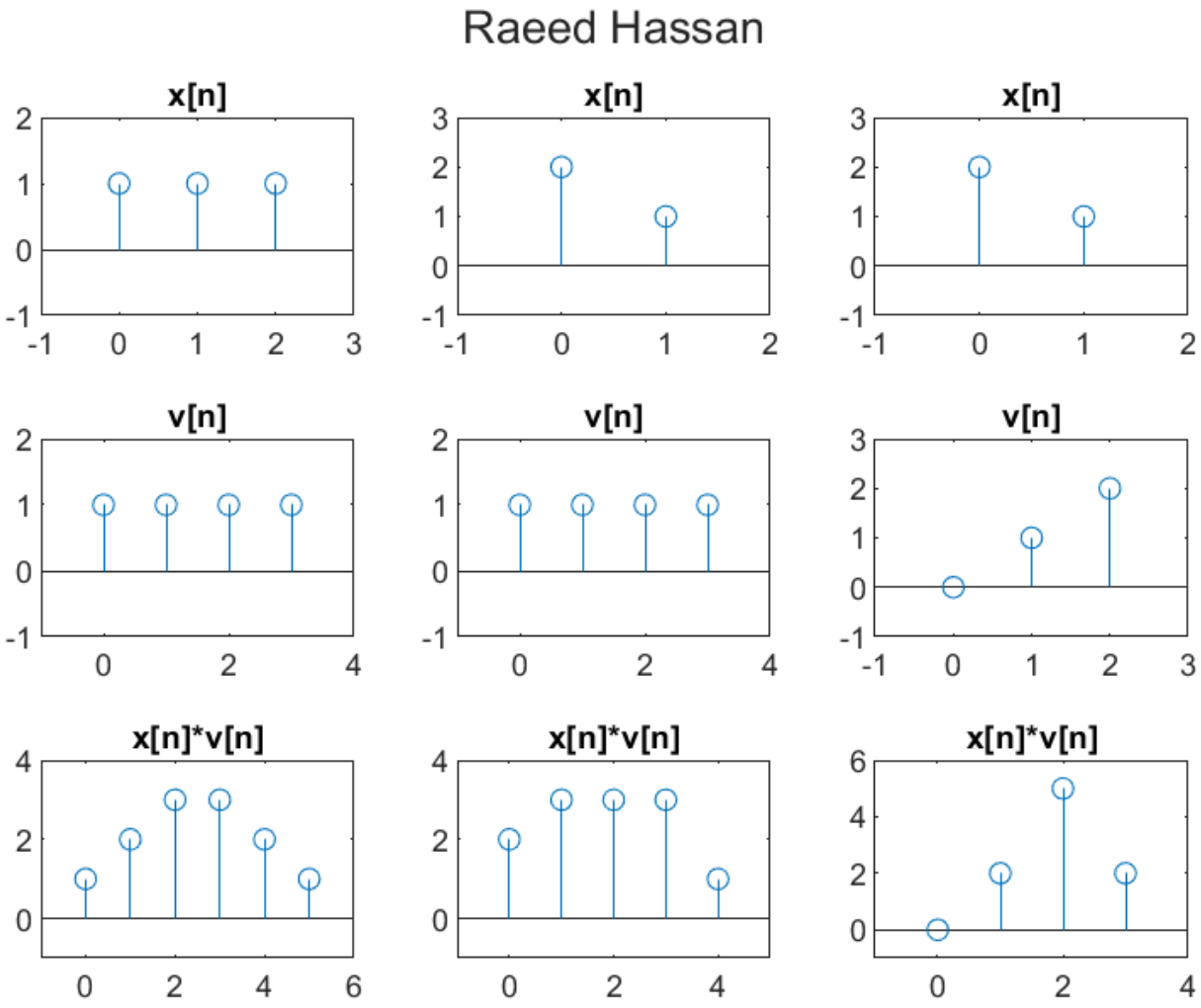

Figure 1: Part 1b Plots

4. The MATLAB code used to generated the distorted signal is shown in Listing 3. The script calls the MATLAB code described in part 3 to read the `speech.wav` audio file. The script calculates the echo delay, `Te`, in seconds and stores this value in `Te_sec`. The number of sample points on the original signal that the echo would be delayed by is calculated by dividing the echo delay in seconds, `Te_sec`, by the sampling period of the original signal, `T`, and stored in the variable `L_delay`. The value of `L_delay` is rounded to ensure that the sample delay is always an integer.

   The signal `signalplusecho` is initialized as a column vector of size `L+L_delay` filled with 0s. The start of the new signal is set to the original signal. The amplified echo signal is added to the new signal at appropriately shifted indices. The new signal was re-scaled to ensure the signal does not clip, then written to a new wav file `speechwithecho.wav`.

Listing 3: Part 4

```
question3;

Te = 1000; % delay in msec
Te_sec = Te / 1000;
L_delay = round(Te_sec/T);
alpha = 0.05;

signalplusecho = zeros(L+L_delay,1);
signalplusecho(1:L) = signal;
signalplusecho(L_delay+1:end) = signalplusecho(L_delay+1:
   end) + signal*alpha;

signalplusecho = signalplusecho/max(abs(signalplusecho));
audiowrite('..\Speech Files\speechwithecho.wav',
   signalplusecho, Fs);
```

5. The MATLAB code used to generated the distorted signal using convolution is shown in Listing 4. The script calculates the number of samples the echo is delayed by. The impulse response, `IR`, was simply chosen to be the starting samples for the original signal and the echo. A column vector filled with 0s of size `L_delay`+1 was created, and the impulses for the original signal (value of 1) and the echo (value of reduced amplitude factor, α) were assigned at indices 1 and `L_delay`+1. The distorted signal, `signalplusecho_conv`, was determined by calculating the convolution of the original signal and the impulse response. The distorted signal was re-scaled to ensure the signal does not clip, then written to a new wav file `speechwithecho_conv.wav`.

Listing 4: Part 5

```
question3;

Te = 1000; % delay in msec
Te_sec = Te / 1000;
```

```
L_delay = round(Te_sec/T);
alpha = 0.05;

IR = zeros(L_delay+1,1);
IR(1) = 1; IR(L_delay+1) = alpha;
signalplusecho_conv = conv(signal,IR);

signalplusecho_conv = signalplusecho_conv/max(abs(
   signalplusecho_conv));
audiowrite('..\Speech Files\speechwithecho_conv.wav',
   signalplusecho_conv, Fs);
```

6. When the value of `alpha` is equal to 1, the value of `Te` must be around 120 ms before the quality of speech is acceptable. At higher values of `Te`, it is either difficult to identify that there is only one signal that is supposed to be transmitted, or the echo interferes with the receiver's ability to clearly understand the original signal. At values of `Te` less than 120 ms, the receiver can both identify that there is a single signal that was intended to be transmitted and understand the contents of the signal relatively clearly.

   The value of `Te` for the quality of the speech to be acceptable does not change from 120 ms as the value of `alpha` decreases, until the value of `alpha` is sufficiently low that the receiver can tune out the echo (`alpha` ≈ 0.05). The echo can still be interpreted as a second signal and interfere with the user's ability to clearly understand the contents of the original signal, even while adjusting the value of `Te`. It is only when the echo is quiet enough the user can largely ignore it, can the value of `Te` be increased to any value with the quality of speech remaining acceptable.

7. The MATLAB code used to generated the distorted signal using convolution is shown in Listing 5. The MATLAB script is very similar to the one used for part 5, with the only differences being an additional variable `num_echos` to store the number of echos, and the impulse response `IR` being generated differently. The size of `IR` is `num_echos*L_delay`+1 instead of `L_delay`+1 as there are multiple additional echos that have to be accounted for at the end of the signal. The exponentially decaying impulses are added to the impulse response using a for loop. The convolution of the original signal and impulse response is calculated and stored in `signalplusreverb`. The distorted signal was re-scaled to ensure the signal does not clip, then written to a new wav file `speechwithreverb.wav`.

Listing 5: Part 7

```
question3;

Te = 0.2; % delay in msec
Te_sec = Te / 1000;
L_delay = round(Te_sec/T);
alpha = 1;
```

```
num_echos = 1000; % number of echos

IR = zeros(num_echos*L_delay + 1,1);
IR(1) = 1;
for i = 1:num_echos
    IR(i*L_delay + 1) = alpha^i;
end
signalplusreverb = conv(signal,IR);

signalplusreverb = signalplusreverb/max(abs(
   signalplusreverb));
audiowrite('..\Speech Files\speechwithereverb.wav',
   signalplusreverb, Fs);
```

When the value of N_e was equal to 1, the signal was identical to the signal examined in part 6, and the observations about `alpha` and `Te` remain the same.

When the value of N_e increased and the value of `alpha` was 1, the value of `Te` to make the quality of speech acceptable depended on the value of N_e. As the value of N_e increases, the value of `Te` required to make the quality of speech acceptable decreases. It was observed that the quality of speech remained acceptable as long as the delay between the last echo and the original signal remained around the 120 ms value (`Te` $\times$ `num_echos` $\approx$ 120 ms) observed in part 6. This was observed with values of N_e as high as 1000. As you increased the delay above the 120 ms delay, the quality of speech sharply degraded as you would begin to hear multiple distinct speeches or high-pitched noise in the signal.

When the value of N_e increased and the value of `alpha` was decreased, the value of `Te` or N_e do not change until the value of `alpha` is sufficiently low that the receiver can tune out all the echos (`alpha` ≈ 0.05), similarly to what was observed in part 6. The echo can still be interpreted as a second signal and interfere with the user's ability to clearly understand the contents of the original signal, even while adjusting the values of `Te` and `num_echos`. The value of `Te` $\times$ `num_echos` needed to keep the quality of speech acceptable remains at around 120 ms, until the value of `alpha` is low enough the user can ignore the echos. At values of `alpha` lower than this point, the values of `Te` and `num_echos` can be changed freely without having a significant effect on the quality of speech.