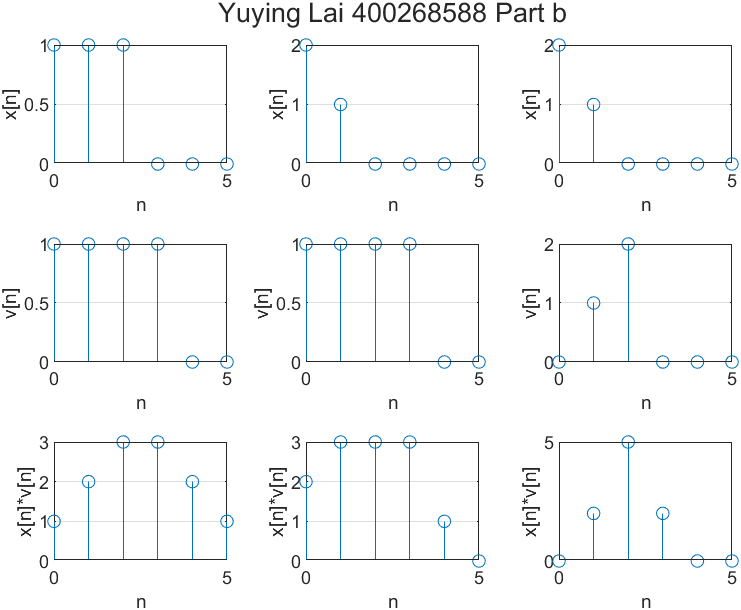
3tp3 lab2

Yuying Lai Laiy24 400268588

1. First use unit step function to create function of x[n] and v[n] for each question. Then, manually using for loop to multiply and add result to the sum for questions in part a. Use conv() for part b to check the result.

The result graph: a(colume 1), b(colume 2), c(colume 3). For range of n out of the graph are all equal to zero.

图片包含 图示

描述已自动生成

Code:

n = 0:10;

f=SimpleFunctions();

% add function

q1ax = f.unitstep(n)-f.unitstep(n-3);

q1av = f.unitstep(n)-f.unitstep(n-4);

q1bx = (-n+2).\*f.unitstep(n)-(-n+2).\*f.unitstep(n-2);

q1bv = f.unitstep(n)-f.unitstep(n-4);

q1cx = (-n+2).\*f.unitstep(n)-(-n+2).\*f.unitstep(n-2);

q1cv = n.\*f.unitstep(n)-n.\*f.unitstep(n-3);

%manually do a

tempsum = 0;

q1a\_con = zeros(1,10);

for a = 1:10

disp(a)

for b = 1:a

disp(b)

tempsum = tempsum+ (q1ax(1,b)\*q1av(1,a-b+1));

end

q1a\_con(1,a) = tempsum;

tempsum = 0;

end

%manually do b

tempsum = 0;

q1b\_con = zeros(1,10);

for a = 1:10

disp(a)

for b = 1:a

disp(b)

tempsum = tempsum+ (q1bx(1,b)\*q1bv(1,a-b+1));

end

q1b\_con(1,a) = tempsum;

tempsum = 0;

end

%manually do c

tempsum = 0;

q1c\_con = zeros(1,10);

for a = 1:10

disp(a)

for b = 1:a

disp(b)

tempsum = tempsum+ (q1cx(1,b)\*q1cv(1,a-b+1));

end

q1c\_con(1,a) = tempsum;

tempsum = 0;

end

% create figure

x = 0:5 % only need the first 6 element

fig = figure('units', 'normalized', 'Name',...

'Question 1');

t = tiledlayout(3, 3);

nexttile;

stem(x, q1ax(:,1:6)); grid on;

xlabel("n");

ylabel("x[n]");

nexttile;

stem(x, q1bx(:,1:6)); grid on;

xlabel("n");

ylabel("x[n]");

nexttile;

stem(x, q1cx(:,1:6)); grid on;

xlabel("n");

ylabel("x[n]");

nexttile;

stem(x, q1av(:,1:6)); grid on;

xlabel("n");

ylabel("v[n]");

nexttile;

stem(x, q1bv(:,1:6)); grid on;

xlabel("n");

ylabel("v[n]");

nexttile;

stem(x, q1cv(:,1:6)); grid on;

xlabel("n");

ylabel("v[n]");

nexttile;

stem(x, q1a\_con(:,1:6)); grid on;

xlabel("n");

ylabel("x[n]\*v[n]");

nexttile;

stem(x, q1b\_con(:,1:6)); grid on;

xlabel("n");

ylabel("x[n]\*v[n]");

nexttile;

stem(x, q1c\_con(:,1:6)); grid on;

xlabel("n");

ylabel("x[n]\*v[n]");

title(t, "Yuying Lai 400268588 Part a");

exportgraphics(fig, "lab2\_question1a.png");

**For part B replace all manually do for loop part with**

%conv

q1a\_con = conv(q1ax, q1av);

q1b\_con = conv(q1bx, q1bv);

q1c\_con = conv(q1cx, q1cv);

1. After read the signal, use delay time/period to find the number of delay sample. For signal at n, add value (alpha \* signal[n-delay]) to the current signal. The value (alpha \* signal[n-delay]) will be the echo.

Code:

clc; clear;

% read signal

[signal, Fs] = audioread('my\_speech\_clip.wav');

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

% process

Te = 200; % in ms

alpha = 1;

delay = round((Te / 1000) / T);

signalplusecho = [signal; zeros(delay, 1)]; % init, add delay time to the end

signalplusecho(delay + 1:end) =alpha .\* signal + signalplusecho(delay + 1:end) ;

% write into new file

signalplusecho = signalplusecho / max(abs(signalplusecho));

audiowrite('speechwithecho.wav', signalplusecho, Fs);

1. To use convolution, we set IR to be 1 at IR[0] and alpha at IR[delay] for the echo. Since matlab index starts at 1, it will than be IR[1] and IR[delay+1]. Using the convolution conv(signal, IR) allows the result signal to have original sound when multiply with IR[1] and the echo when multiply the alpha at IR[delay+1].
2. For question 6, Te will generally need to be less than 200ns for a clear sound. But it depends on the speaking speed of the audio. If the user speaks fast, it required less delay like 120ns to avoid much superposition.

Code:

clc; clear;

% read signal

[signal, Fs] = audioread('my\_speech\_clip.wav');

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

% process

Te = 200; % in ms

alpha = 1;

delay = round((Te / 1000) / T);

IR = [1;zeros(delay,1)];

IR(delay+1,1) = alpha;

signalplusecho = conv(signal,IR);

% write into new file

signalplusecho = signalplusecho / max(abs(signalplusecho));

audiowrite('speechwithecho\_con.wav', signalplusecho, Fs);

1. There is total of Ne echo and the echo will appears at the delay time when it is the multiplex of Te (1\*Te, 2\*Te, ect). Therefore, we still set the value of IR at multiplex of delay be decay with factor alpha. For example, IR[delay+1] will be alpha^1, IR[delay\*2+1] will be alpha^2. Using conv(signal, IR) can get the echo result.

Code:

clc; clear;

% read signal

[signal, Fs] = audioread('my\_speech\_clip.wav');

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

% process

Te = 70; % in ms

alpha = 0.9;

Ne = 5;

delay = round((Te / 1000) / T);

IR = [1;zeros(Ne\*delay,1)];

for i = 1:Ne

IR(i \* delay + 1) = alpha ^ i;%put echo into response function

end

signalplusecho = conv(signal,IR);

% write into new file

signalplusecho = signalplusecho / max(abs(signalplusecho));

audiowrite('speechwithecho\_rev.wav', signalplusecho, Fs);

To if we set alpha to be 1, the echo does not decay. It means that the echo will always have the original amplitude as the signal. If the number of echoes is large, the echo repeats fast. When all echo added together it will be hard to determine the original content due to the superposition. For example, if the delay time is 100ms and number of echoes is larger than 3, we will hear signal at 0ns, 100ns, 200ns and 300ns all added up together at 300ns.

To get a clear result signal, it required either less repeating times or less delay time. If the superposition of each word end before the next word appears, it will be possible to know the content. In my example, if set Ne be 5, the delay time should be less than 70.