Lab 5 – FIR Filtering

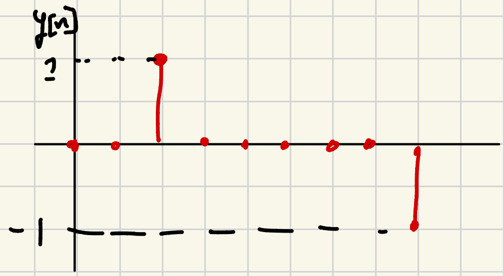
3.1 a) $\delta[n-3] * \delta[n-5] \Rightarrow \delta[n-8]$

$\delta[n-a] * \delta[n-b] \Rightarrow \delta[n-(a+b)]$

3.1 B) $x[n] = -3\{0, 0, 1, 1, 1, 1, 1, 1, 0, 0,\}$

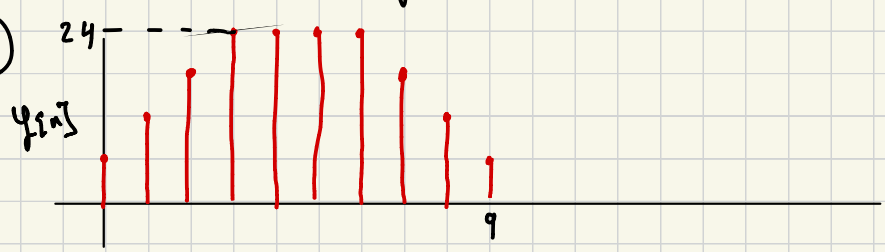
$h[n] = \{1, -1\}$

$$y[n] = \begin{cases} 1 & n = 2 \\ -1 & n = 8 \\ 0 & \text{otherwise} \end{cases} \quad \text{or} \quad y[n] = \delta[n-2] - \delta[n-8]$$



3.1 C) The filter computes the difference between consecutive values ⇒ any consecutive (1, 1) pair will convolve to zero except for the edges of the signal.

3.1 D)



3.1 E) max amplitude: 24, length: $7+4-1 = 10$

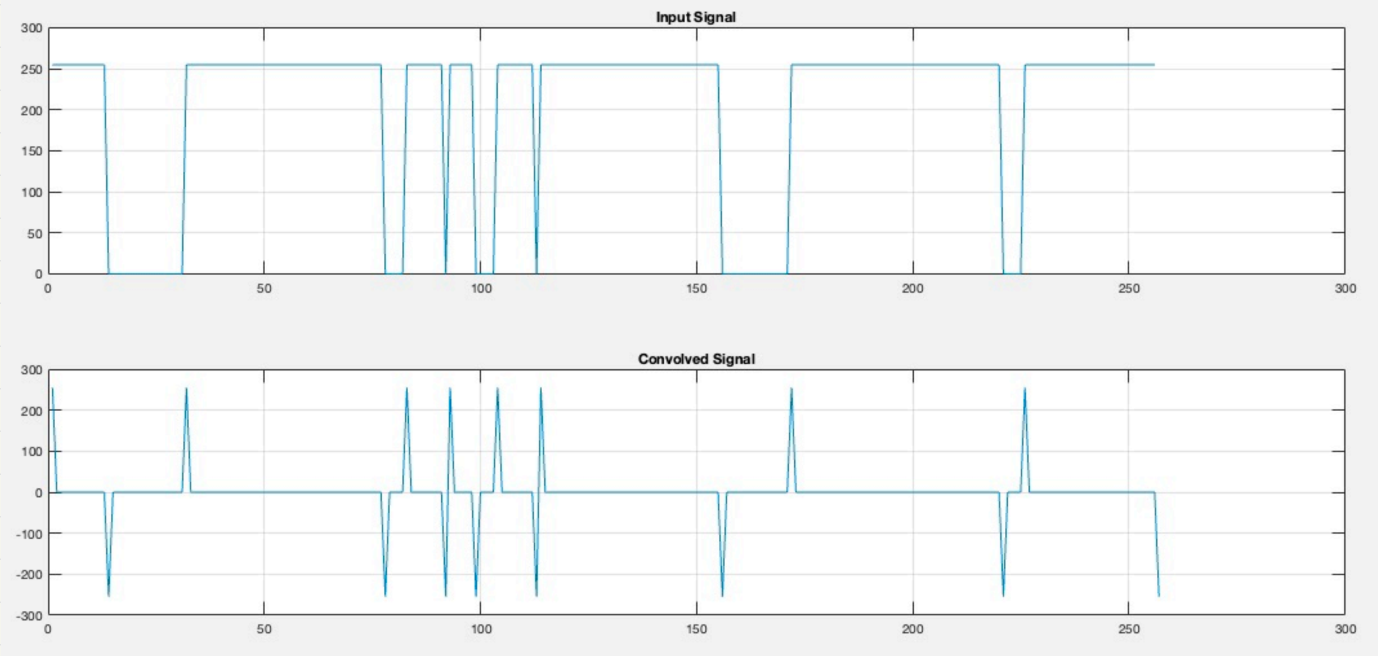
3.1 F) $y[n] =$ 1 @ $n = 7, 20, 32, 45$

-1 @ $n = 13, 25, 38,$

@ $y[n] = 1$, the signal $x[n]$ transitions from $0 \to 1$ (a pos − 0 ⇒ +1)

@ $y[n] = -1$, the signal $x[n]$ transitions from $1 \to 0$ (a neg − 0 ⇒ −1)

3.2 c)



impulses Negative indicate a white -> Black transition + vice versa Positive => a Black -> white transition

indices = find(yyi) => indices(3) - indices(2)

3.4 b) @ $\omega = .2\pi$, $|H(e^{j\omega})| = .374$

$\angle H(e^{j\omega}) = -1.885$ or $-.6\pi$

3.4 c)

Time Delay of output signal $y[n]$

$(\phi_{y[n]} - 2\pi \ell) = \phi_{x[n]} + \angle H(e^{j\omega})$

3.4 D) $y[n] = 1.8\,(|H(e^{j\omega})|)\cos(2\pi(.1)(n - \phi_x + \frac{\angle H(e^{j\omega})}{2\pi}))$

$= .673\cos(.2\pi(n-6)) \Rightarrow n_d = 6$

3.3.2

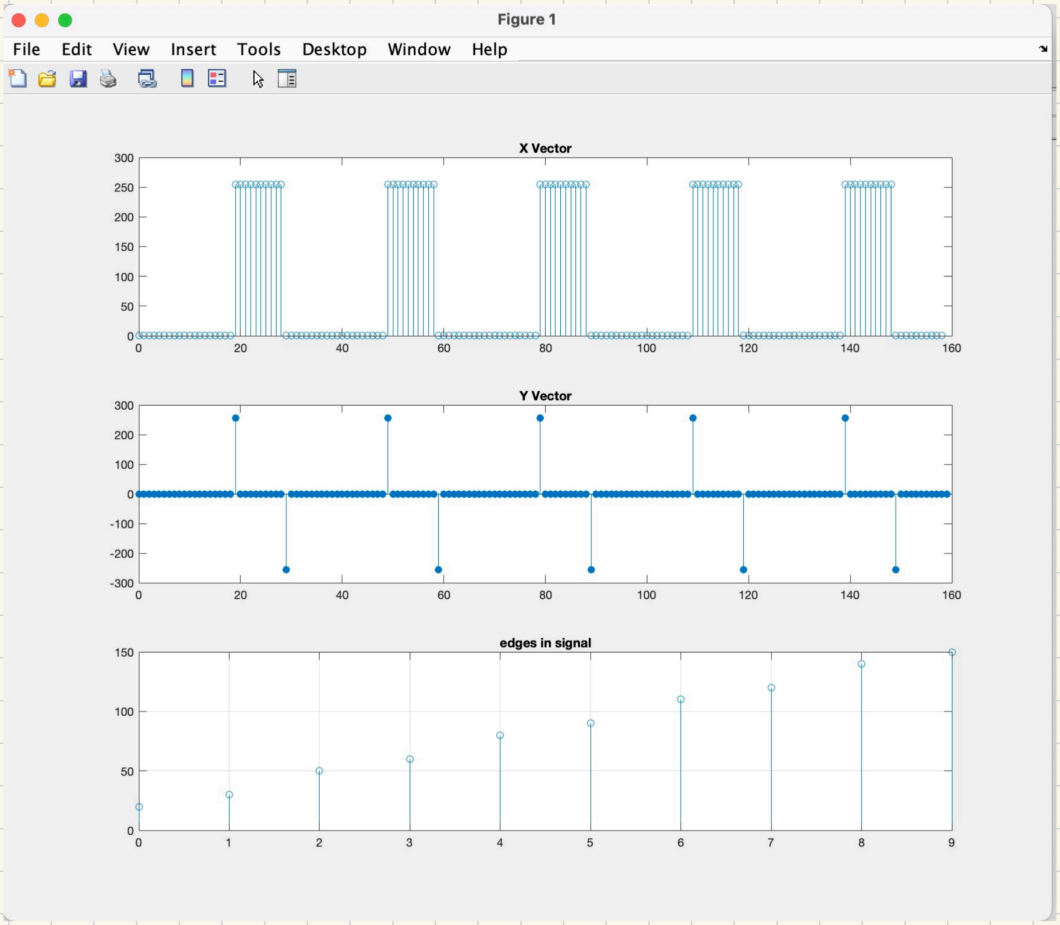


Figure 1
File Edit View Insert Tools Desktop Window Help
X Vector
Y Vector
edges in signal

3.3.4

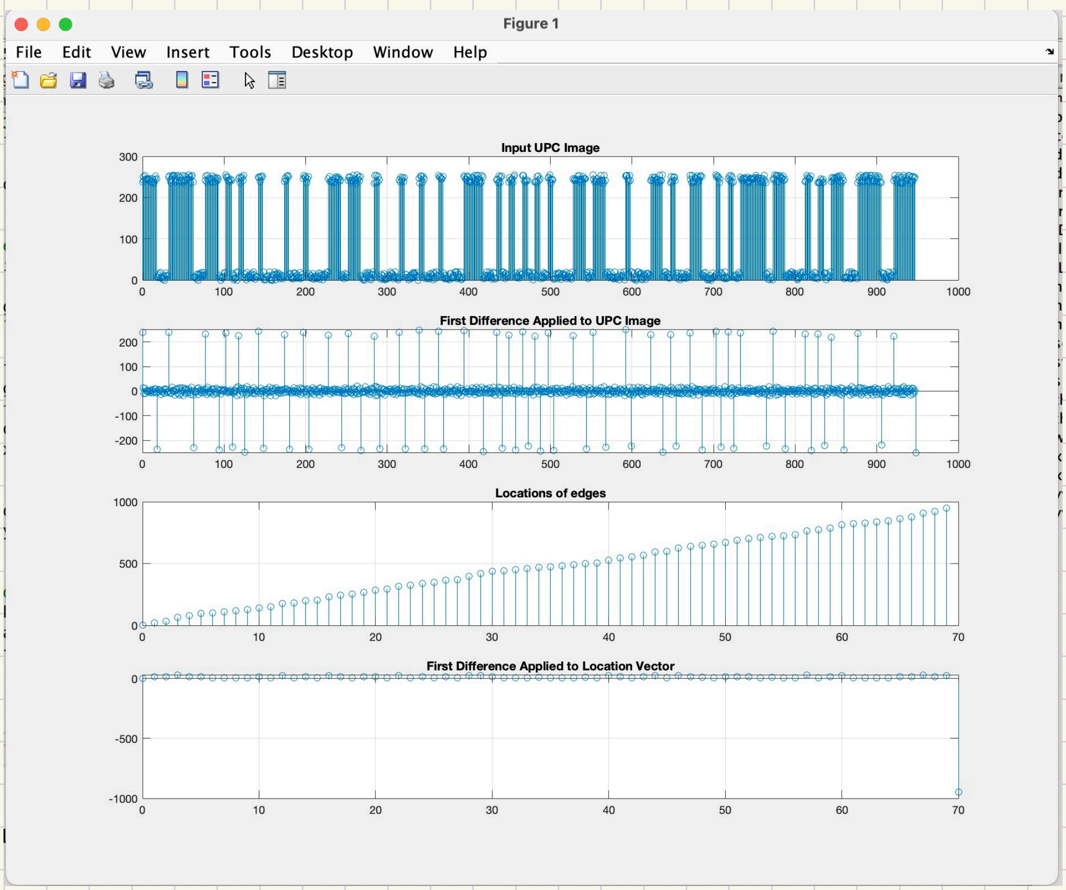
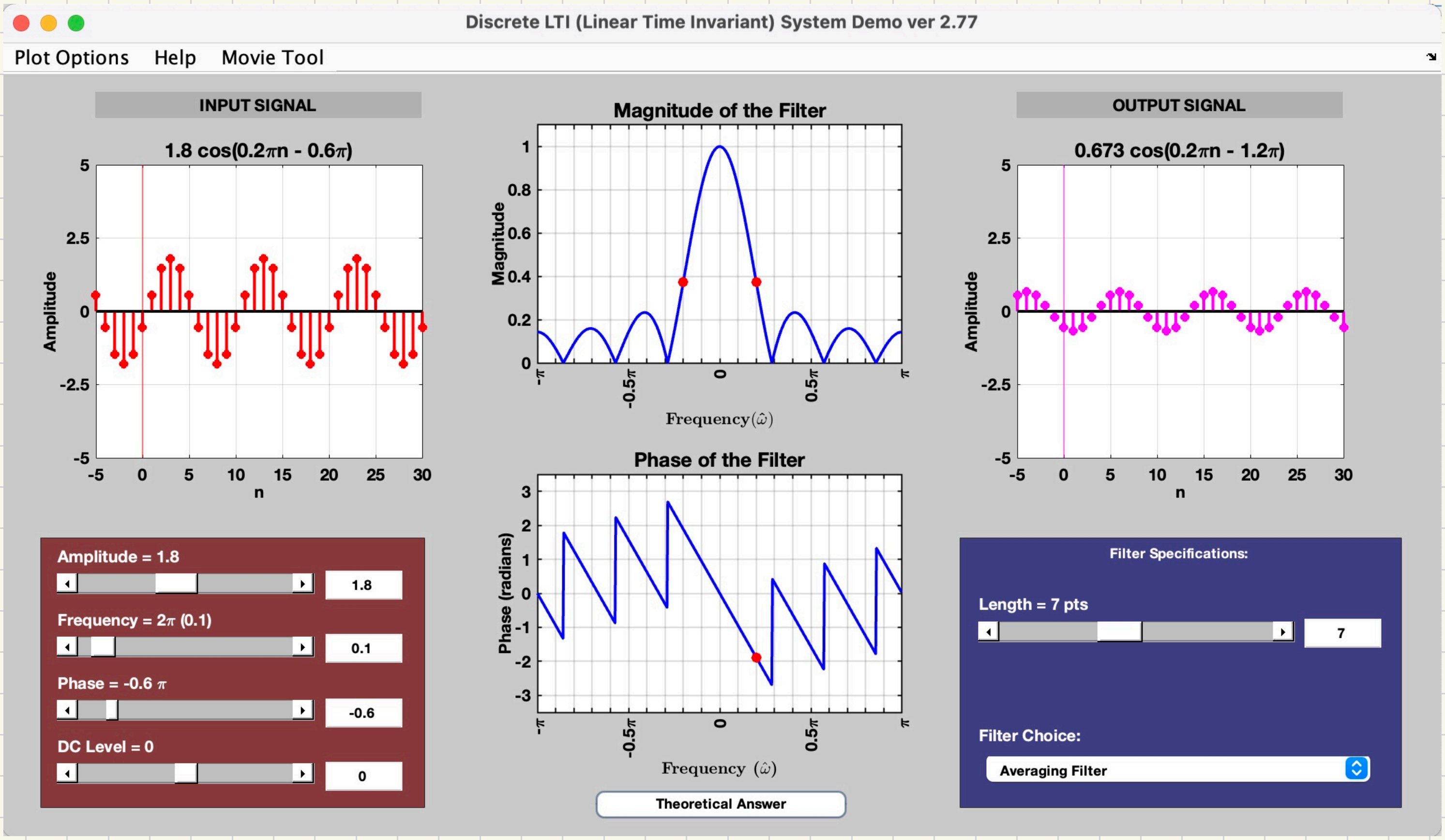


Figure 1
File Edit View Insert Tools Desktop Window Help
Input UPC Image
First Difference Applied to UPC Image
Locations of edges
First Difference Applied to Location Vector

3.4



Discrete LTI (Linear Time Invariant) System Demo ver 2.77
Plot Options
Help
Movie Tool
INPUT SIGNAL
1.8 cos(0.2πn - 0.6π)
Amplitude
n
Magnitude of the Filter
Magnitude
Frequency(ω̂)
OUTPUT SIGNAL
0.673 cos(0.2πn - 1.2π)
Phase of the Filter
Phase (radians)
Frequency (ω̂)
Amplitude = 1.8
1.8
Frequency = 2π (0.1)
0.1
Phase = -0.6 π
-0.6
DC Level = 0
0
Theoretical Answer
Filter Specifications:
Length = 7 pts
7
Filter Choice:
Averaging Filter

3.2

```
%% 3.2
clc;clear;close all
load echart.mat

bdiffh = [1, -1];

imshow(echart)
m = 65; % 147, 221
yy1 = conv( echart(m,:), bdiffh );

%% Plot the input and output in the same figure using subplot
nn = 1:length(echart(m,:));

subplot(2, 1, 1);
plot(nn, echart(m,:)); grid on;
title('Input Signal');

subplot(2, 1, 2);
plot([nn, length(yy1)], yy1); grid on;
title('Convolved Signal');

%% Find the width of "E"

indices = find(yy1);

%ignore the first impulse

width = indices(3) - indices(2);
disp(width)
```

3.3.2

```matlab
%% 3.3.2
clc;clear;close all

%% Part a
xx = 255*(rem(1:159,30)>19);
bb = [1, -1];
yy = firfilt(bb, xx);

% Plot x and y using subplot
nn1 = 1:length(xx);
subplot(3, 1, 1);
stem(nn1-1, xx(nn1));

title("X Vector");

nn2 = 1:length(yy);
subplot(3, 1, 2);
stem(nn2-1, yy(nn2), 'filled');
title("Y Vector");

%% Part b
% Explain the effect of the first-difference operator on this input signal.
% puts an impulse (positive) when the input signal transitions from 0 to
% HIGH and then gives a negative impulse when the input signal
% transitions from high to low
%% Part c
% Find length of xx and yy
lengthY = length(xx)+length(bb)-1;

%% Part d: find the edges
threshold = 255;
d = abs(yy)>=threshold;

%% Part e: find edges indices
edge_index = find(d)
num_edges = length(edge_index);
rangeIdx = 1:num_edges;

subplot(3, 1, 3);
stem(rangeIdx-1, edge_index(rangeIdx)); grid on;
title("edges in signal");
```

3.3.4

```
%% 3.3.4 (UPC)
clc;clear;close all
img = imread('HP110v3.png');
numPlots = 4;
% img = imread('OFFv3.png');    % Uncomment fo part j

% Take one row in the middle
xx = img( round(size(img, 1)/2) , : );

xxLength = 1:length(xx);
% Apply first difference filter
bb = [1, -1];
yy = firfilt(bb, xx);
yyLength = 1:length(yy);
% Plot input and output using subplot
subplot(numPlots, 1, 1);
stem(xxLength-1, xx(xxLength)); grid on;
title('Input UPC Image');

subplot(numPlots, 1, 2);
stem(yyLength-1, yy(yyLength)); grid on;
title('First Difference Applied to UPC Image');

% Find d[n] and l[n]
threshold = 200 ;
dd = abs(yy) >= threshold;
ll = find(dd);



num_edges = length(ll);
lLength = 1:num_edges;

subplot(numPlots, 1, 3);
stem(lLength - 1, ll(lLength)); grid on;
title('Locations of edges')


% Apply first difference filter to calculate bar widths
delta = firfilt(bb, ll)
lDelta = 1:length(delta);
% Plot l[n] and delta[n] using subplot
subplot(numPlots, 1, 4);

stem(lDelta-1, delta(lDelta)); grid on;
title('First Difference Applied to Location Vector');

% Part e
%  prove that the total width of a valid 12-digit bar code is equal to 95θ
%since there are 12 digits in a UPC, and each digit is encoded with bar
%widths of different orders all totaling 7, there are going to be 84 total
%bar width variations (12*7). But each UPC is delimited by 1-1-1 on each
%end adding 6 bar widths and then is separated in the middle by 1-1-1-1-1
%adding the last 5 to total 95 times the unit bar width.
```

```matlab
% Loop through all the subsets
for start_idx = 1:length(delta)-58+1
    % Take subset of length 59 starting with start_idx
    subset = delta(start_idx:start_idx+59-1);

    % Part f
    sorted_delta = sort(subset); %sort the delta signal
    num_smallest = 31;  %grab the smallest values
    theta = median(sorted_delta(1:num_smallest))+1; %take the average of
those small values
    % theta = 6;

    % Part g
    width_arr = round(subset / theta); %divide the delta signal by the
average width and then round

    % Part h (decodeUPC.p is provided)
    code = decodeUPC(width_arr);

    % Check for incorrect codes
    incorrect = any(code == -1); %if any element in codes is -1, incorrect is
true
    % Continue for loop if incorrect; break out if correct
    if (~incorrect)
        break;
    end

end

%% Printing Detected code
code
```