Lab06_MattBachmeier

Table of Contents

ECE 203 April 9, 2017 Tasks to be performed in Lab06

Download and install GUIs

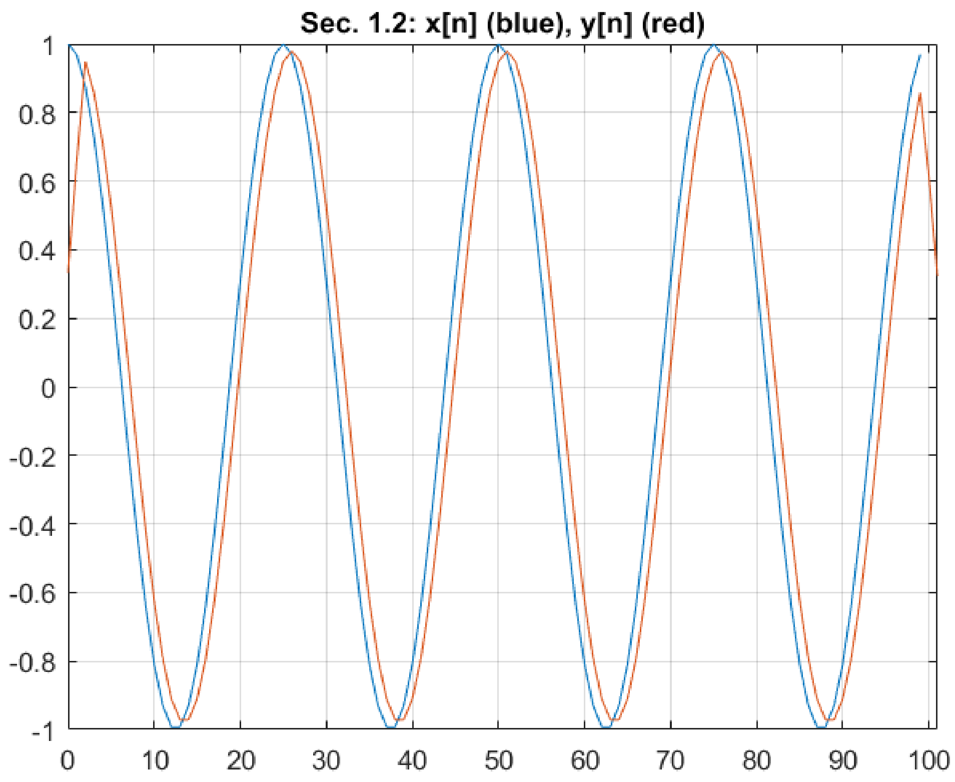
Two GUI (graphic user interface) will be used: con2dis, and dconvdemo They are part of the Matlab package spfirst_v168 that you can download from the course website at URL: These two files can be downloaded from textbook website:

http://spfirst.gatech.edu/matlab/

To install these GUIs, you should first decompress the zip files and place the files into folder (dafult names) con2dis and dconvdemo Then use the "set path" tab within Matlab to add paths to these folders so that your matlab can find files in these folders. Please use "add with subfolders" option.

Section 1.2

```
nn = 0:99; %<--Time indices
xx = cos( 0.08*pi*nn ); %<--Input signal
bb = [1/3 1/3 1/3]; %<--Filter coefficients
yy = firfilt(bb, xx); %<--Compute the output

n2 = 0:(numel(yy)-1);
% **** enter matlab commands below to plot yy (y[n]) and xx (x[n]).
 The
% index range of yy are integers from 0 to lenght of output yy.
% <- Enter code here
mm = 0:length(yy)-1;
figure(1)
plot(nn,xx), hold on
plot(mm,yy), grid on

title('Sec. 1.2: x[n] (blue), y[n] (red)')
axis([0 n2(end) -1 1])
```



Section 1.4

Sampling and Aliasing Demo

```
% Enter "con2dis" in the command window to bring up the demo GUI
% a) set the input to x(t)=cos(40*pi*t) in the GUI
% b) set the sampling frequency f_s = 24 samples/sec.
% c) determine the locations of spectrum lines of the discrete time
 signal
% x[n], found in the lower middle panel. Click the "Radian" button to
% change the x-axis unit from f^ to omega^.
% d) determine the formula for the output signal y(t) shown in the
 upper right
% panel. What is the output frequency in Hertz?
%
% ****** Give answers to c) and d) below
% Answer: Discrete time signal has spectrum lines at approximately
 -1.2,
% -0.8, -0.2, 0.2, 0.8, and 1.2. The output frequency is 4 Hz, showing
 us
% there is aliasing.
% <- Enter answer here
```

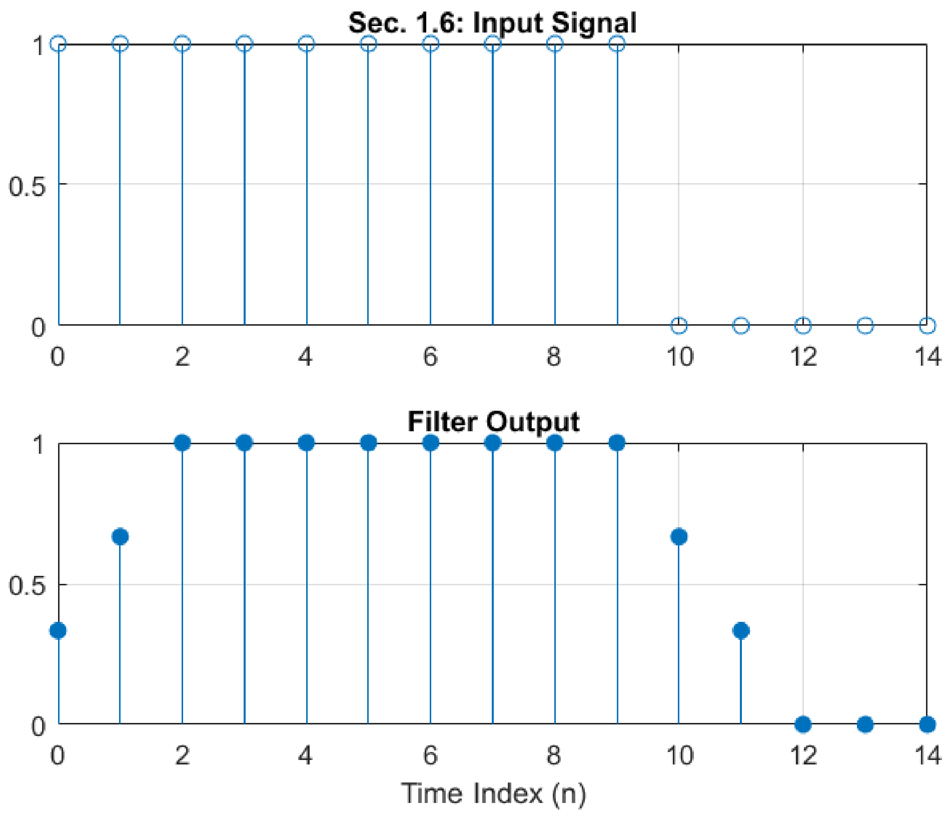
Section 1.5

practice as instructed. There is nothing to be submitted.

Section 1.6

Do convolution using firfilt (included in downloaded spfirst_v168.

```
bb = repmat(1/3,1,3);
xx = [ ones(1,10) zeros(1,5) ];
% ***********  Call firfilt to compute output yy below
% <- Enter code here
yy = firfilt(bb,xx);

% 1.6 (b) plot the output
first = 0;
last = numel(xx)-1;
nn = first:last;
figure,
subplot(2,1,1);
stem(nn,xx); grid on
title('Sec. 1.6: Input Signal')
subplot(2,1,2);
stem(nn,yy(nn+1),'filled'); grid on %--Make black dots
title('Filter Output')
xlabel('Time Index (n)')

% 1.6 (c) Explain the filtering action of the 3-point averager by
 comparing
% the plot in the previous part. This filter might be called a
 "smoothing"
% filter. Note how the transitions in x[n] from zero to 1 and from one
 back
% to zero, have been "smoothed".
% There is nothing to turn in in 1.6. (c)
```



Warm up

2.1 Sampling and Aliasing

```
% Use the con2dis GUI to do the following problem:
% Given (a) input frequency = 12 Hz, (b) fs = 15 Hz,
%
%  ********(c) Determine frequency of reconstructed output signals
% Answer: The GUI shows the frequency of y(t) is 3 Hz
% <- Enter answer here
%
%  ********(d) determine the locations in omega^ of the lines in the
%  spectrum of the discrete time signal, give numerical value
% Answer:
% digital frequency of the aliased signal is -7/6, 5/6, -1/6, 1/6,
 5/6, 7/6
% <- Enter answer here
% Translating to radian = -7pi/3, -5pi/3, -pi/3, pi/3, 5pi/3, 7pi/3
% <- Enter answer here
%
%  ******** (e) change the sampling frequency to fs = 12 Hz, the
 reconstructed
%  signal  has an aliased frequency = 0 Hz
% <- Enter answer here
```

2.2 Discrete Time Convolution

Use dconvdemo to do the following task: (a) Set x[n] = (0.9)^n (u[n]-u[n-10]) (b) Set h[n] = delta[n] - 0.9*delta[n-1].

```
*********** (c) Illustrate output y[n] and Explain why its values are
almost all zeros. Compute the numerical value of the lat point in y[n]
that is non-zero.
 There is a value of 1 at index 0 and -0.3486 and index 9. They are
 mostly zero because all but the edge values will essentially cancel
 each other out by subtracting the previous value of 0.9^n with its
 positive value of the same degree.
<- Enter answer here
```

2.3 Loading data

labdat.mat can be downloaded from lab6 data in the Moodle page.

```
clear all; close all;
load labdat.mat;
whos; % check variables loaded from labdat, should have x1, xtv, x2,
 h1, h2
% There is nothing to turn in in this subsection.
```

```
  Name            Size                 Bytes  Class     Attributes

  h1              1x45                   360  double              
  h2              1x45                   360  double              
  x1              1x100                  800  double              
  x2          24576x1                 196608  double              
  xtv           256x1                   2048  double              
```

2.4 Filtering a Signal

```
Do a 5-point average on input x1 and plot stem plot of both input and
output
```

2.4 (a) filter coefficient of 5 point average

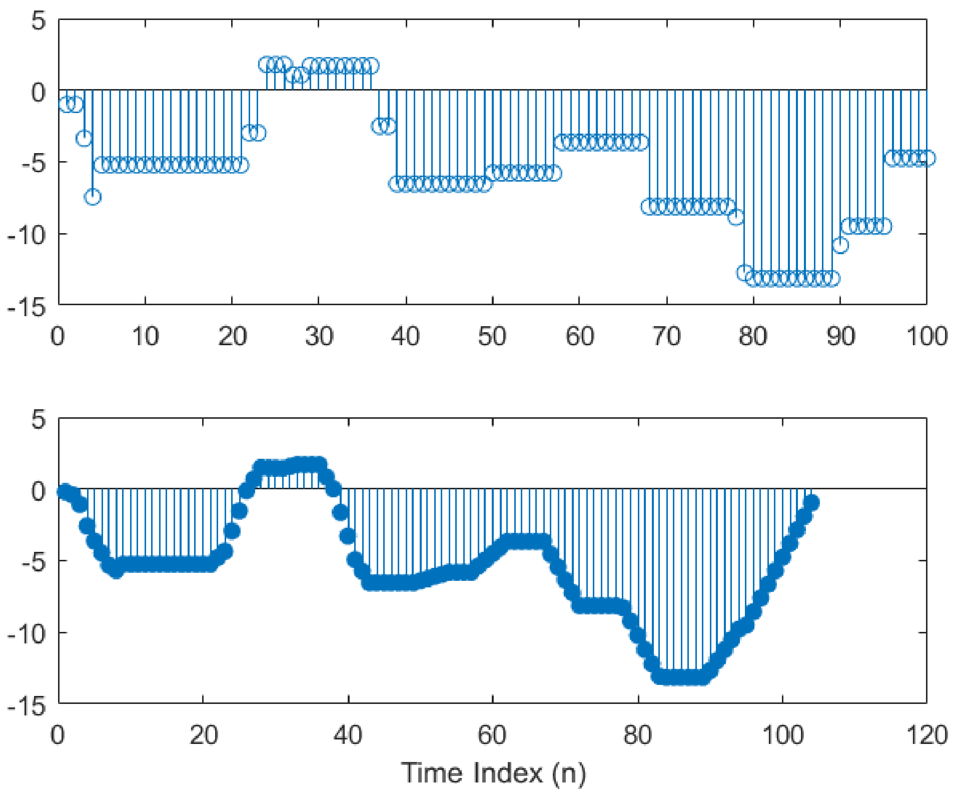
```
bb = (1/5)*ones(1, 5);
%
%  ******** Use firfilt to process x1 using bb. Denote output as y1
y1 = firfilt(bb,x1);
% <- Enter code here


%  How long are the input and output signals?
inputLength = numel(x1)
outputLength = numel(y1)
```

```
% <- Enter code here
```

```
inputLength =

   100


outputLength =

   104
```

2.4 (b) Stem plot

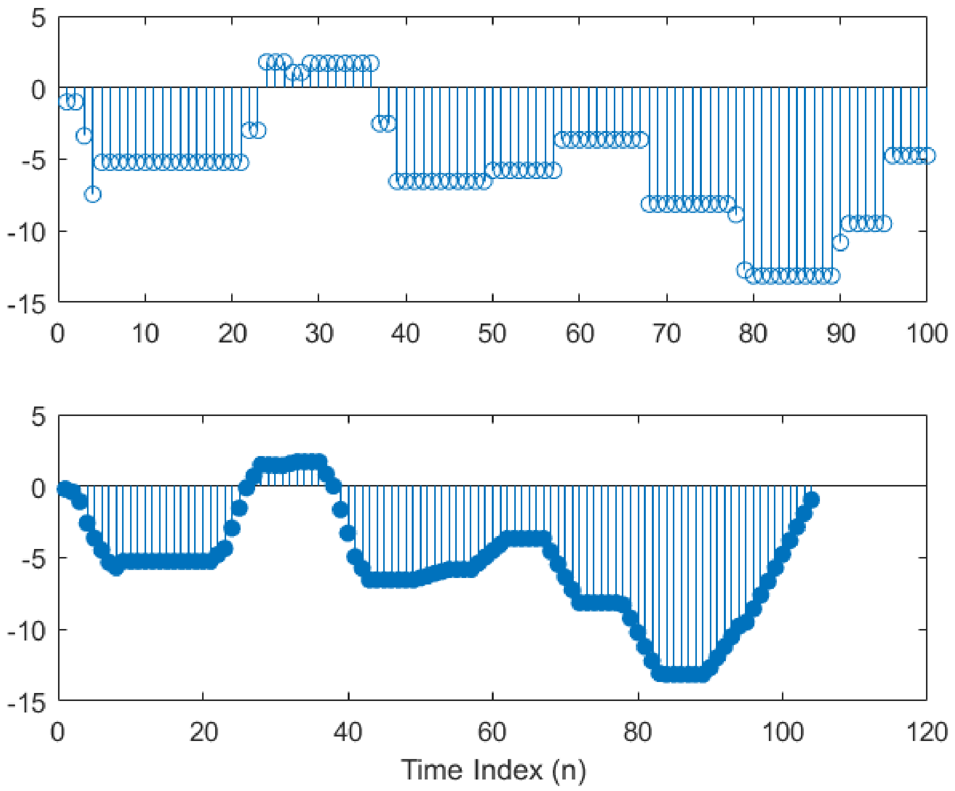
```
% ******* set first and last to the beginning and ending of input x1
 first = 0; %<- Enter code here
 last =  inputLength - 1; %<- Enter code here

nn = first:last;
subplot(2,1,1);
stem(x1)
subplot(2,1,2);
stem(y1,'filled') %--Make black dots
xlabel('Time Index (n)')
```



2.4 (c) a zoomed stem plot of 30 points in the middle

Nothing to turn in 2.4(c)

```
first = ceil(length(x1)/2)-15;
last = first + 30 -1;
nn = first:last;
figure,
subplot(2,1,1);
stem(x1)
subplot(2,1,2);
stem(y1,'filled') %--Make black dots
xlabel('Time Index (n)')
```



2.4 (d)

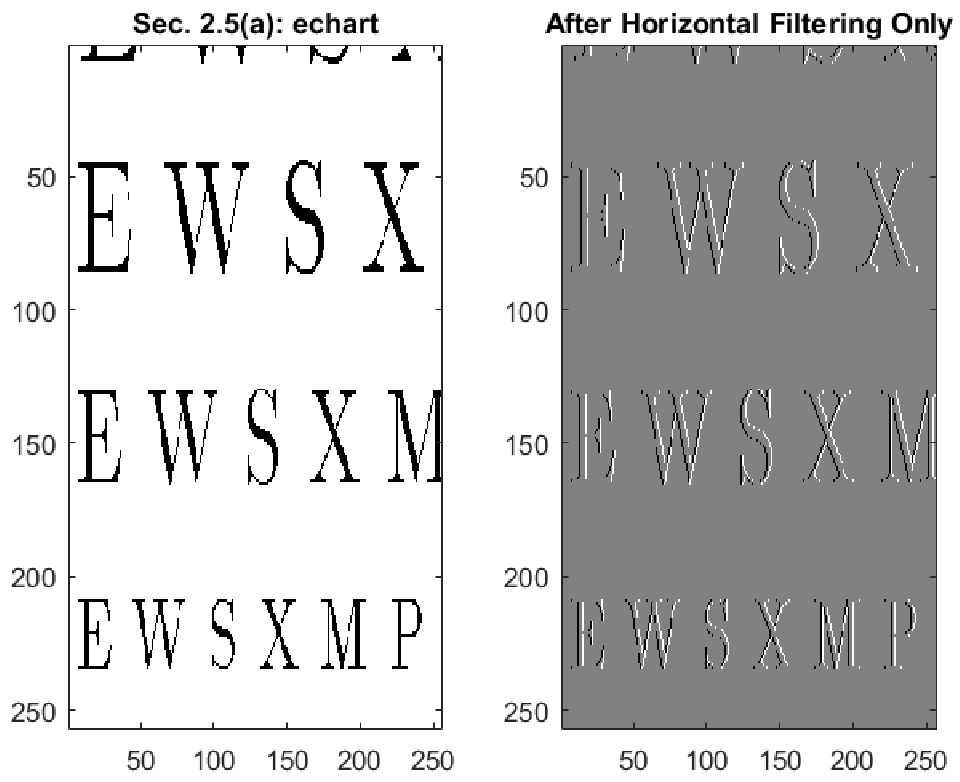
Compare stem plots from 2.4(b) and 2.4(c) and note the smoothed transition from one level to another. This is why it is a "smoothed" filter.

There is nothing to turn in in 2.4(d)

2.5 Filtering Images: 2-D Convolution

using echart.mat and conv2()

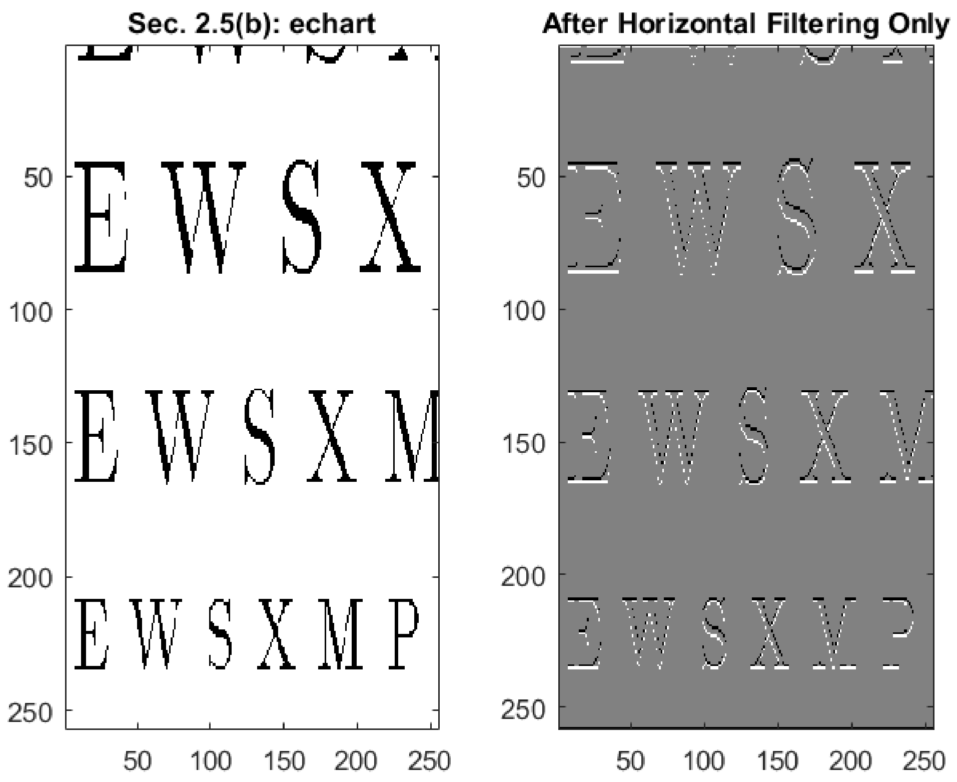
```
clear all; close all;
load echart; % variable name echart is 257 by 256
[e1, e2] = size(echart);
bdiffh = [1, -1];
yy1 = conv2(echart, bdiffh);

%  ******* Display the input image echart and output image yy1 on the
%  screen at the same time.

figure,
subplot(1,2,1)
imagesc(echart);% <- Enter code here
title('Sec. 2.5(a): echart')
gry = true; % Set Up Color Scheme
if gry
colormap(gray)
end
subplot(1,2,2)
imagesc(yy1);% <- Enter code here
title('After Horizontal Filtering Only')


% ********* Compare and give a qualitative description of
%  what you see
% Echart is simply black letters typed on a white board with different
 sizes. The
% convoluted image looks like everything is now just a gray color but
 where
% the letters were appear to be raised or stamped into a sheet of gray
% metal. Can not see raised edges on straight horizontal lines from
 the
% letters.
% <- Enter your answer here
```



2.5 (b)

do first difference filter in the VERTICAL direction.

```
% Call the filter bdiffv, input is echart, and output yy2.
% Plot the input and output at the same time on the screen.

bdiffv = [1, -1]';
yy2 = conv2(echart, bdiffv);

%  ******* Display the input image echart and output image yy2 on the
%  screen at the same time.

figure,
subplot(1,2,1)
imagesc(echart);% <- Enter code here
title('Sec. 2.5(b): echart')
gry = true; % Set Up Color Scheme
if gry
colormap(gray)
end
subplot(1,2,2)
imagesc(yy2);% <- Enter code here
title('After Horizontal Filtering Only')
```



Section 3. Lab Exercise: FIR Filters

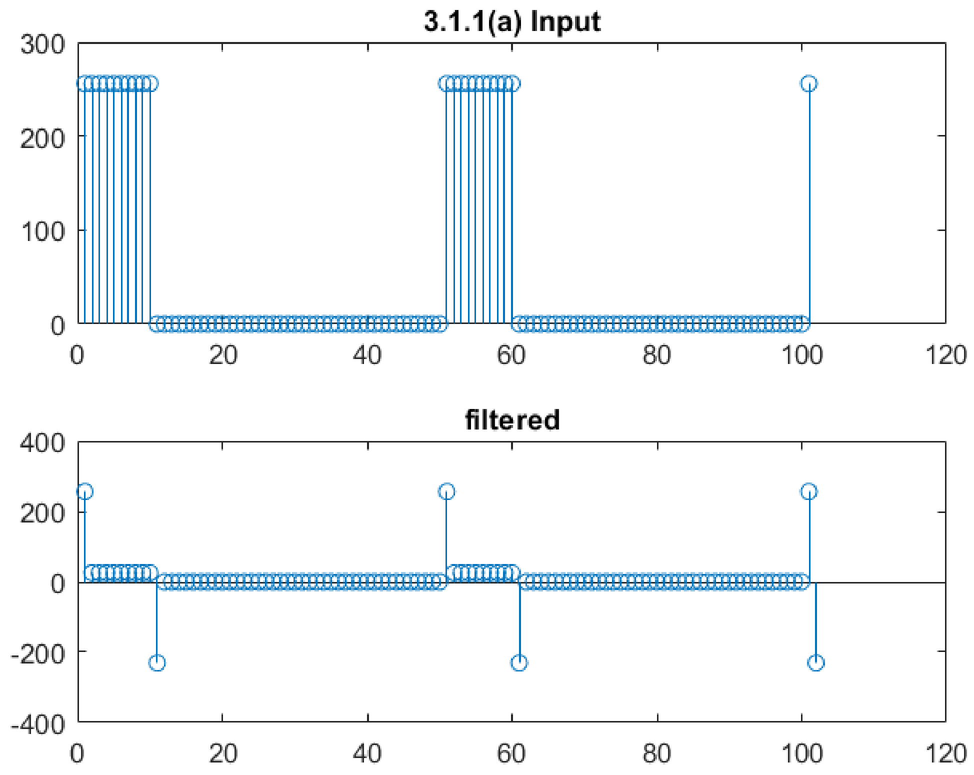
Use FIR filter to produce echo and deconvolution

3.1 Deconvolution Experiment for 1-D Filters

```
clear all; close all;
xx = 256*(rem(0:100,50)<10);
% ********* implement a filter w[n]=x[n]-0.9*x[n-1];
bb = [1,-0.9]% <- Enter code here
wn = firfilt(bb, xx);


%  ********** (a) Stem plot x[n], w[n] on the same figure using
 subplot,
%  restricting horizontial axis in the range 0 <= n <= 75.

nn = 0:75;
figure,
subplot(2,1,1),
stem(xx);% <- Enter code here
title('3.1.1(a) Input'),
subplot(2,1,2),
stem(wn);% <- Enter code here
title('filtered')
```

```
% ******* Explain why the output appears the way it does
% Answer: The filter used will only allow the first part of a signal
 to be
% output and the end of a signal will be reversed and reduced in
 magnitude
% slightly. This happens because of the type of filter that is being
 used
% to convolute the signal.
% <- Enter Answer here.
```

```
bb =

    1.0000   -0.9000
```

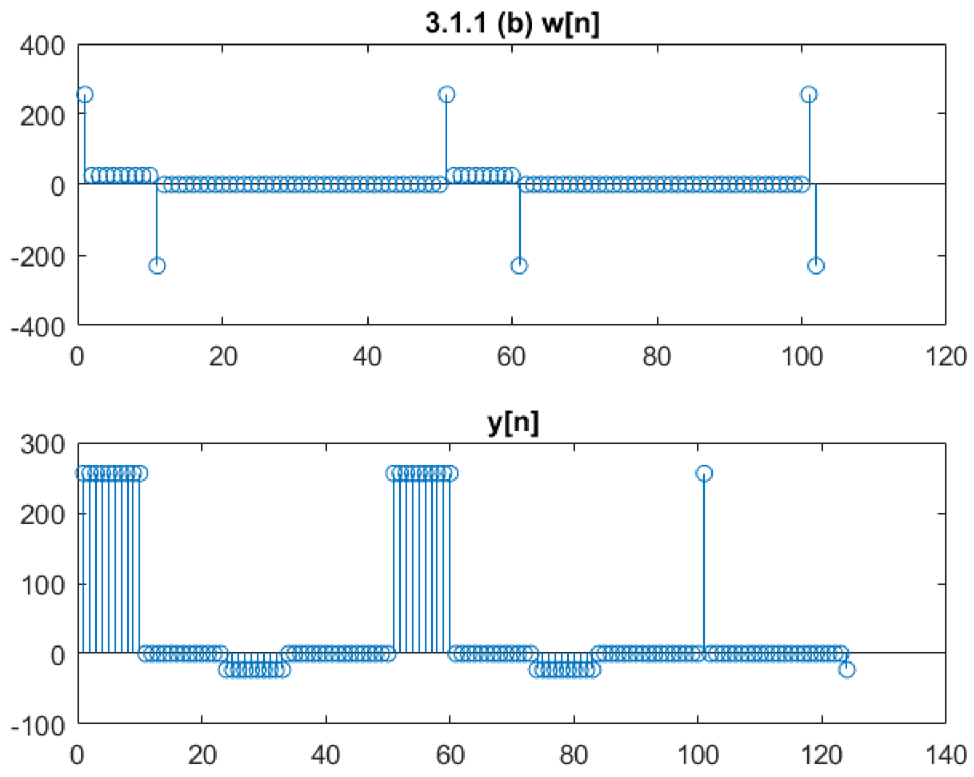


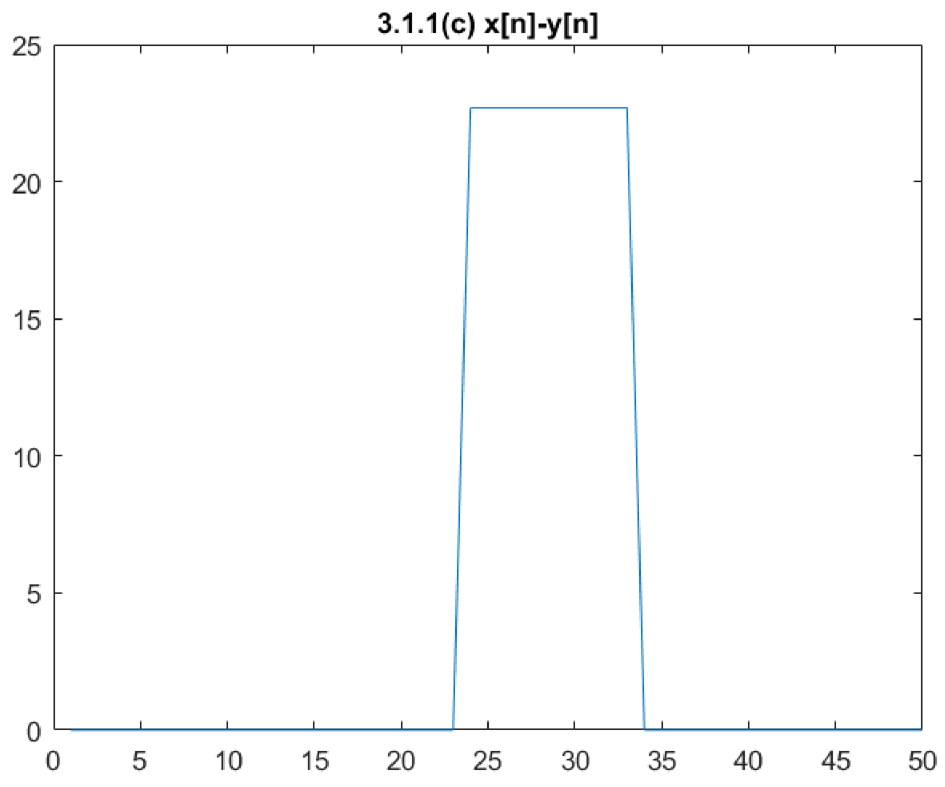
3.1.1 Resotration Filter

HINT: The impulse response of FIR Filter 2 in this section for r = 0.9, M = 22 is h[n] = (0.9)^n for 0 <= n <= 22 and 0 elsewhere. The matlab commend to generate the fir filter is (.^ means element by element operation) bb2 = 0.9.^[0:22];

```
% ***** compute bb2
M = 22;
r = 0.9;
```

```
bb2 = r.^[0:M];% <- Enter code here

% ***** (a) filter w with bb2
yn = firfilt(bb2,wn);% <- Enter code here

% ****** (b) stem plots of w[n] and y[n] in the same plot with
 subplots.
%    using index range nn that is the same for both signals.

figure,
subplot(2,1,1)
stem(wn);% <- Enter code here
title('3.1.1 (b) w[n]');
subplot(2,1,2)
stem(yn);% <- Enter code here
title('y[n]')

%  ****** (c) plot the difference x[n] - y[n] over 0 <= n <= 50
nn=1:50;
error = abs(abs(xx(nn))-abs(yn(nn)));% <- Enter code here
figure,
plot(nn,error);% <- Enter code here
title('3.1.1(c) x[n]-y[n]');
```





3.1.2 Worst Case Error

find worst case error and discussion

```
% **************3.1.2(a) Find the worst case error using Max and
% absolution error between x[n] and y[n] over 0 <= n <= 50
worstCase = max(error)% <- Enter code here


% 3.1.2 (b) What does this work case error tell you about the quality
 of
% the resotration filter?
% ********* skip this question
```

```
worstCase =

   22.6891
```

3.1.3 An Echo Filter

```
clear all; close all;
```

3.1.3(a)

Determine r and P in eq. (4) ******* You need to be familiar with the derivation below: fs = 8000 Hz, time delay = 0.2 sec. strength = 0.8 of original If input is x(t), output y(t) = x(t) + 0.9*x(t - 0.2). Now, T = 1/fs, and set t = nT. So, y(nT) = x(nT) + 0.9*x(nT - 0.2/T) or y[n] = x[n] + 0.9*x[n - 0.2*fs] Therefore, r = 0.9, P = 0.2*fs = 1600

```
fs=8000;
r = 0.9;
P = 1600;% <- Enter code here to set r and P
```

3.1.3 (b)

describe filter coefficient of the FIR filter in eq. (4) using r and P from part (a) and determine its length

```
% the impulse response of eq. (4) is h1[n] = delta[n] + r*delta[n-P]
% ******* generate the filter coefficient b1
b1 = zeros(1,P);
b1(1) = 1;
b1(end) = r;%<- Enter code here;

fprintf('echo filter length = %g\n', length(b1));
```

```
echo filter length = 1600
```

3.1.3 (c)

use x2 in labdat.mat as input to compute echo output yecho using eq. (4) and results from (a) and (b) above.

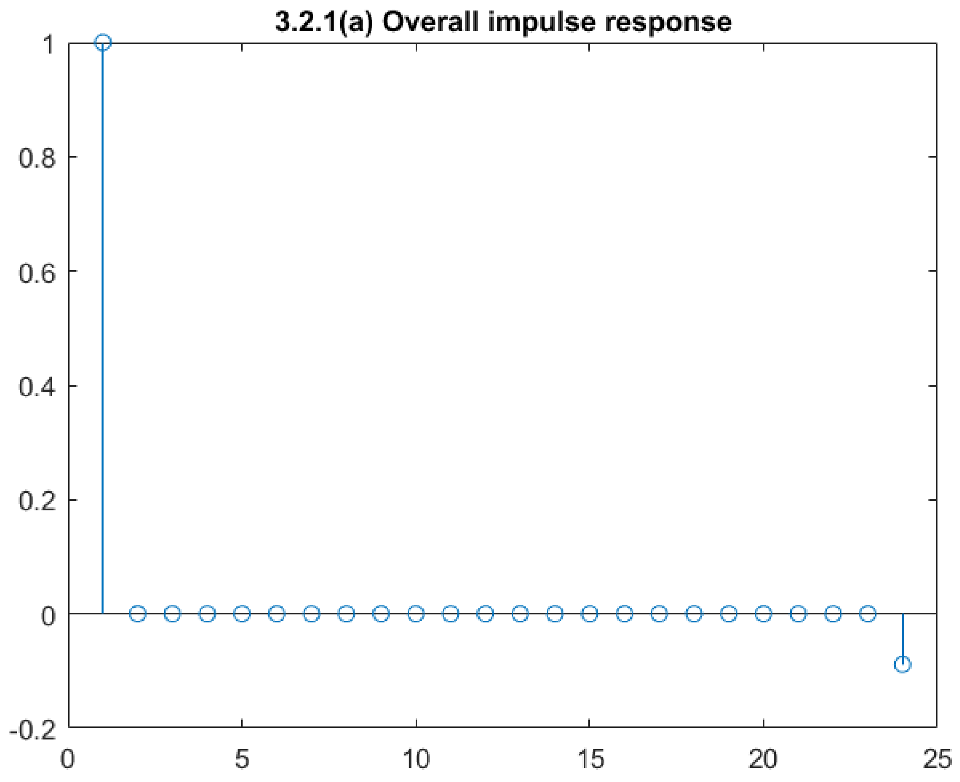
```
load('labdat.mat');
yecho = firfilt(b1,x2);%<- Enter code here;

soundsc(yecho,fs);

% section 3.1.3 (d) will be skipped
```

3.2 Cascading Two Systems

3.2.1 (a) Overall Impulse Response

```
% Overall impulse response with q = 0.8, r = 0.9, M = 22.
% Also plot the impulse response.
%
q=0.9; r=0.9; M=22;
% *********** Filter 1: impulse response h1
h1 = [1,-q];% <- Enter code here

% *********** Filter 2: impulse response h2
h2 = r.^[0:M];% <- Enter code here

%  ********** Overall impulse response h = h1*h2
```

```
h = firfilt(h1,h2);% <- Enter code here

% Plot the overall impulse reponse
nn=1:length(h);
stem(h),
title('3.2.1(a) Overall impulse response')
```



3.2.1 (b) Manual verification of result in (a)

No need to submit anything. Refer to the supplement materials posted.

3.2.1 (c)

Discussions: the overall system response differs from perfect reconstruction in the last term -r^(M+1). As M increases, this term converges to 0, and the filter approaches perfect reconstruction.

% No need to submit anything.

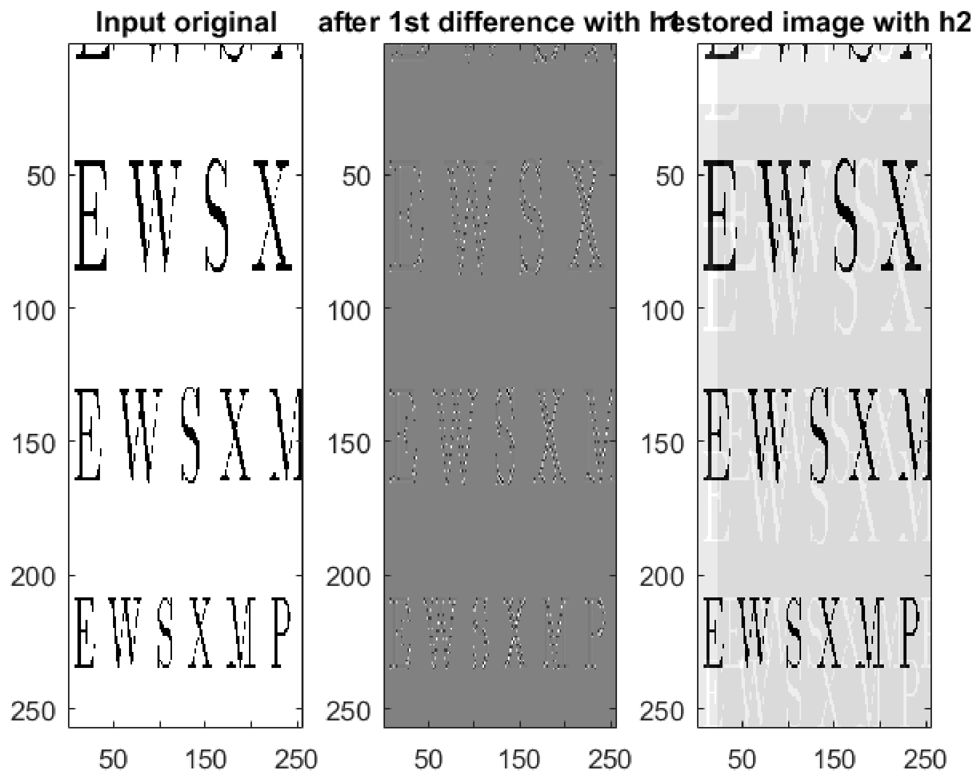
3.2.2 Distorting and Restoring Images

```
clear all; close all;
% 3.2.2 (a)
% Load echart.mat
load('echart.mat'); % variable echart 257 x 256
```

```
[e1, e2]=size(echart);
subplot(1,3,1),
imagesc(echart),colormap('gray')
title('Input original')

% ****** 3.2.2 (b)
% q = 0.9; % first order difference filter h1 is [1 -q]
% filter echart by h1 horizontally and then vertically using conv2
% Note that if h1 is a row vector, convole with h1 is filtering
% horizontally. Also, h1' will be a column vector and convole with h1'
 will
% apply this filter on each column

q = 0.9;
h1 = [1 -q];
ech90 = conv2(echart, h1);% <- Enter code here
ech90 = conv2(ech90, h1');% <- Enter code here

subplot(1,3,2)
imagesc(ech90(1:e1,1:e2)),colormap('gray')
title('after 1st difference with h1')


% *******  3.2.2 (c)
% Deconvolve ech90 with fir filter II with r = 0.9, M = 22 using conv2

r = 0.9; M = 22;
h2 = r.^[0:M];
echrest = conv2(ech90, h2);% <- Enter code here
echrest = conv2(echrest, h2');% <- Enter code here

subplot(1,3,3)
imagesc(echrest(1:e1,1:e2)),colormap('gray')
title('restored image with h2')

% describe visual appearance of the output. Determine where are
 "ghosts"
% and worst case error and estimate how big the ghost's magnitude
 relating
% to the black-white transition from 0 to 255.
%
% ANSWER: the restored image suffers from "ghost" artifacts. This
 indicates
% that the combined filter h=h1*h2 is an echo fir filter whose length
 is 2
% + M - 1 = 23. Thus, the ghost
% images are seperated from the original images by P pixels both
% horizontally and vertically by P = 23 - 1 = 22 pixels in each
 direction.


%
% *********  worst case error:
eworst= max(abs(echart(1:256) - echrest(1:256)))% <- Enter code here
```

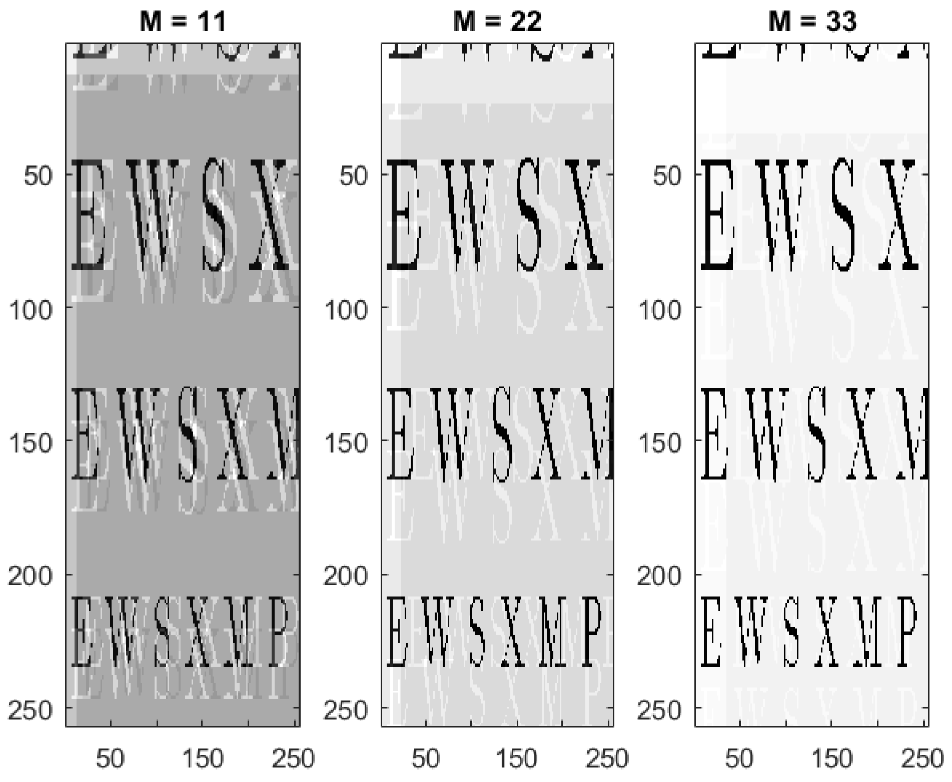
```
% the magnitude of the ghost image is proportional to the last
 coefficient
% of the combined echo filter h = h1*h2.
% ********* Compute the maximum magnitude of the ghost image,
 mag_ghost given that the
% pixel values can change from 0 (dark) to 255 (bright).
h = firfilt(h1,h2);% <- Enter code here
lastCoefProp = (r^M)*(0:2900);
possibleGhost = find(abs(lastCoefProp - round(lastCoefProp)) <
 0.0001);
mag_ghost = round(max(lastCoefProp(possibleGhost)));% <- Enter code
 here

fprintf('the max magnitude of ghost image = %g\n',mag_ghost);
```

```
eworst =

   22.6005

the max magnitude of ghost image = 97
```



3.2.3 A second restoration experiment

```
% *********** (a) deconvolve ech90 with r = 0.9, M = 11, 22, and 33.
% denote the three restored images as ech{k}, k = 1, 2, 3
% e1, e2 are dimensions of original image echart.
r = 0.9;
for k = 1:3,
    M = 11*k;
    h2 = r.^[0:M];
    ech{k} = conv2(ech90, h2);% <- Enter code here  % convolve rows
 with h2
    ech{k} = conv2(ech{k}, h2');% <- Enter code here  % convolve
 column with h2'
end


figure, colormap('gray')
subplot(1,3,1), imagesc(ech{1}(1:e1,1:e2)), title('M = 11');
subplot(1,3,2), imagesc(ech{2}(1:e1,1:e2)), title('M = 22');
subplot(1,3,3), imagesc(ech{3}(1:e1,1:e2)), title('M = 33');

% ************  Question: which value of M yields best result? Why?
% <- Enter your answer here
% ANSWER:
```



Published with MATLAB® R2016a