ECE 320 Signals & Systems II

# Matlab Project I Solutions

Fall 2008

## Overview

This project reviews discrete-time convolution using Matlab. In particular it focuses on how to index the input and output vectors, and how to deal with finite-length approximations to the convolution of infinite-length signals. Section 2 develops and tests an overlap-add filter function called `oafilt`. In Section 3 `oafilt` is used to filter a noisy audio signal.

The report below summarizes the solutions for each part of the project and contains plots of the various output signals. Analytical work and copies of the Matlab code are appended to this report.

## 1 Basic Convolution Exercises

(a) Figure 1 shows the convolution of the signals $h[n] = \delta[n+1]+\delta[n-1]$ and $x[n] = \delta[n]-3\delta[n-2]$. See the appendix for an analytical solution to this problem. The Matlab solution and the analytical solution match.

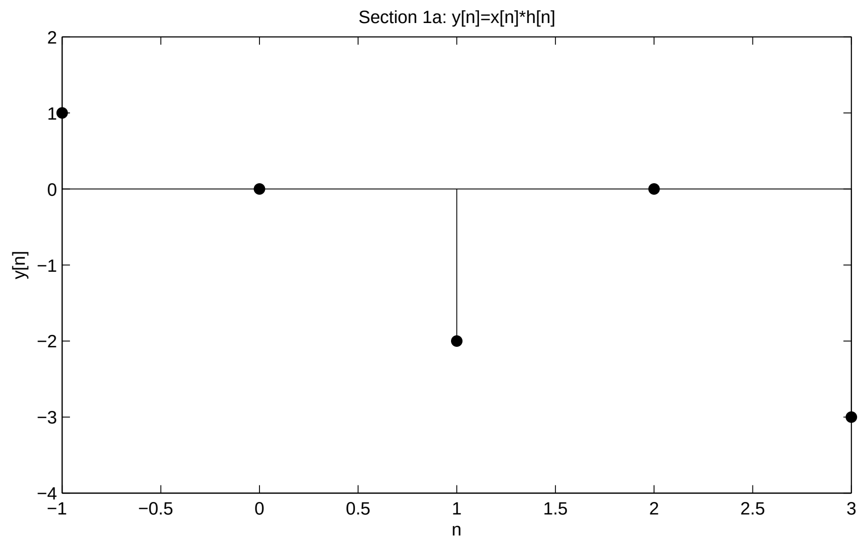


Figure 1: Convolution of $x[n]$ and $h[n]$ as defined in Section 1a.

(b) The goal of this analytical problem is to determine the starting and ending points of a convolution result based on the starting and ending points of the input signal and the impulse response.

(i) The appendix shows the analytical convolution of the following signals:

$$h[n] = \delta[n-a] + \delta[n-b];$$

$$x[n] = \delta[n-c] + \delta[n-d].$$

The result is $y[n] = \delta[n-c-a] + \delta[n-d-a] + \delta[n-c-b] + \delta[n-d-b]$. In other words there are four unit impulses located at $a+c$, $a+d$, $b+c$, and $b+d$.

(ii) The vector of indices for the output signal $y[n]$ should start at $a+c$ and end at $b+d$, *i.e.*, `ny=a+c:b+d`.

(iii) The length of the output vector will be $(b+d)-(a+c)+1$ (the difference in the endpoints plus 1). If it is known that $a=0$, $b=N-1$, $c=0$, and $d=M-1$, the length of the output vector will be $(N-1+M-1)-(0+0)+1=N+M-1$. This agrees with the given information.

(c) This problem considers convolving the following signals:

$$h[n]=u[n+2];$$

$$x[n]=\left(\frac{3}{5}\right)^{n-2}u[n-2].$$

(i) The appendix shows the analytical solution to this convolution.

(ii) Figure 2 shows the result of convolving these signals when $h[n]$ is defined for $-2\le n\le 14$ and $x[n]$ is defined for $0\le n\le 24$. Since x is a truncated version of the true input, the output vector y only contains the correct values for $n\le 16$. The plot marks the good and bad points of the output. See the analytical solution for more details.

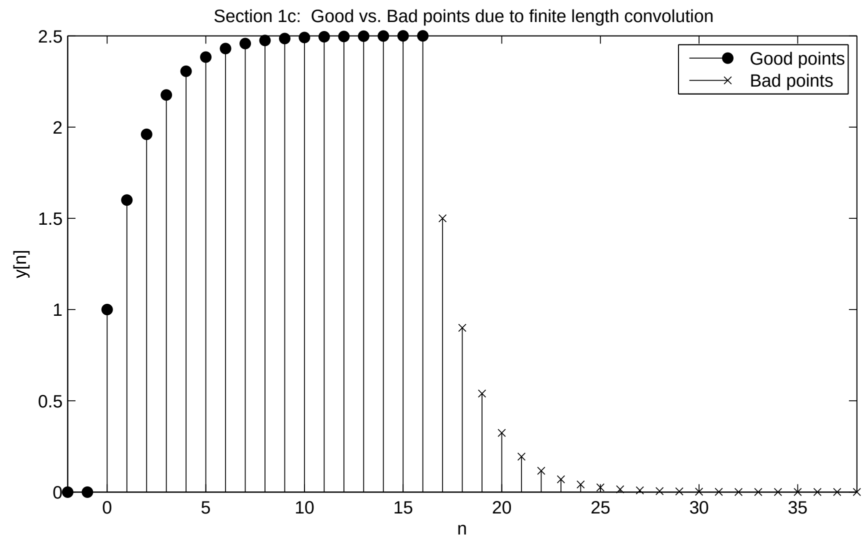


Figure 2: Convolution of $x[n]$ and $h[n]$ as defined in Section 1c.

## 2 Block Convolution

In this part we consider the problem of computing the output $y[n]$ of an LTI system for $0\le n\le 99$ when the impulse response and input signal are defined below:

$$h[n]=(0.9)^n(u[n]-u[n-10]),$$

$$x[n]=u[n].$$

(a) (i) Figure 3 shows the results of using the conv command to compute the output for $0\le n\le 99$.

(ii) In this part we consider breaking the input into two segments of length $L=50$ and using the conv command, compute the convolution of $h[n]$ with each of the segments. In other words we compute:

$$y_0[n]=h[n]*x_0[n]\qquad\qquad y_1[n]=h[n]*x_1[n],$$

where $x_0[n]$ contains the first 50 samples of $x[n]$ and $x_1[n]$ contains the last 50 samples of $x[n]$. The output $y[n]$ may be computed as follows

$$y[n] = x[n] * h[n] = y_0[n] + y_1[n - M].$$

The value of $M$ can be determined by looking at what the starting point of the convolution with the second segment should be. The first input segment starts at $n = 0$ and ends at $n = L - 1$=49. The second segment starts at $n = L = 50$ and ends at $n = 99$. Note that the filter starts at $n = 0$. To obtain the starting point of a convolution we add the starting points of $x[n]$ and $h[n]$. For the second segment that will be $0 + L = 50$. Thus the value of $M$ in the equation above should be $M = L = 50$. The code in the script file `proj1.m` shows how this 2-block convolution was implemented. Figure 4 shows the result of implementing the convolution this way. The results are the same as for part 2a-i.

**Matlab note:** It is important to remember that Matlab indexes start at 1, not 0. Thus when we index into the output y vector, we have to add 1 to the n values indicated in the previous paragraph. See the project code `proj1.m` in the appendix for additional details.

(b) The Matlab code for the `oafilt` function is given in the appendix.

(c) Figure 5 shows the results of convolving these two signals using the `oafilt` function with a block size of $L = 50$. The bottom plot confirms that the `oafilt` function produces the same result as the `conv` function that was used in part 2a. Note that the same results are obtained for other block sizes.

## 3 Filtering of a Noisy Audio Signal

The top plot of Figure 6 shows the noisy audio signal contained in the `proj1_data.mat` file available on the course website. The bottom plot in Figure 6 shows the audio signal after filtering using the `oafilt` function with the 61-point FIR filter contained in the `proj1_data.mat` file. The filter effectively removed the high frequency noise, revealing that the audio signal is a clip from the Hallelujah Chorus by Handel.

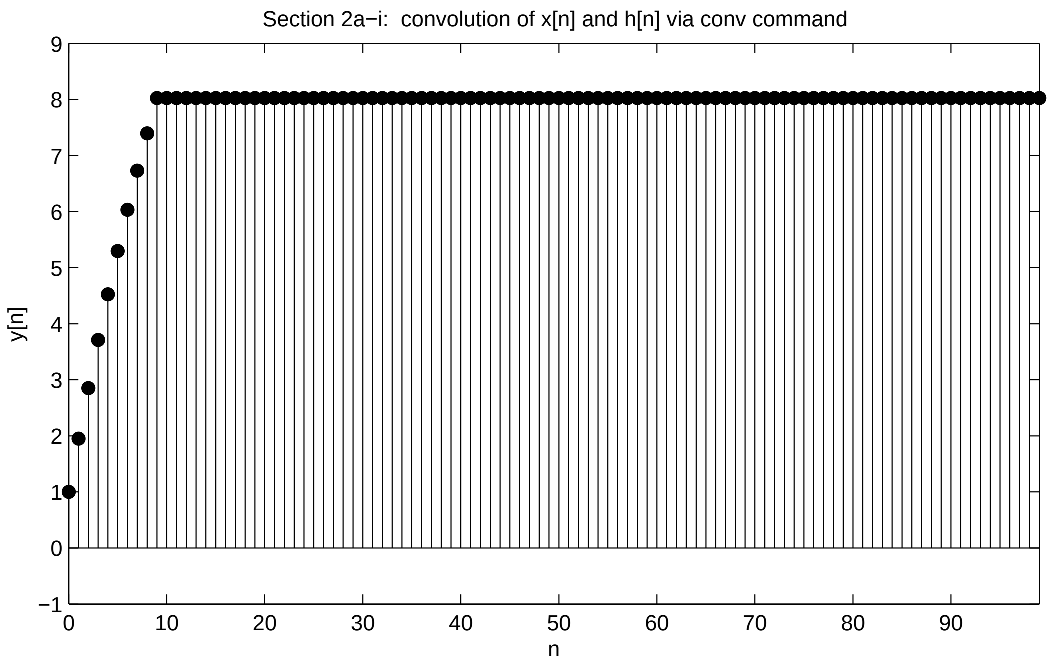


Figure 3: Convolution of $x[n]$ and $h[n]$ as defined in Section 2a-i.

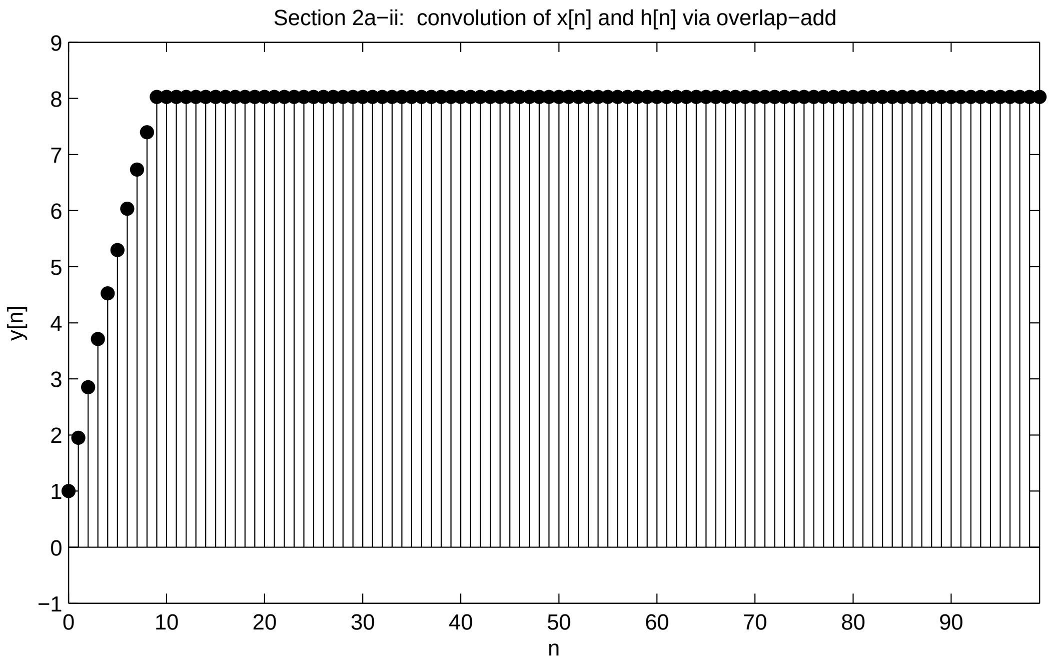
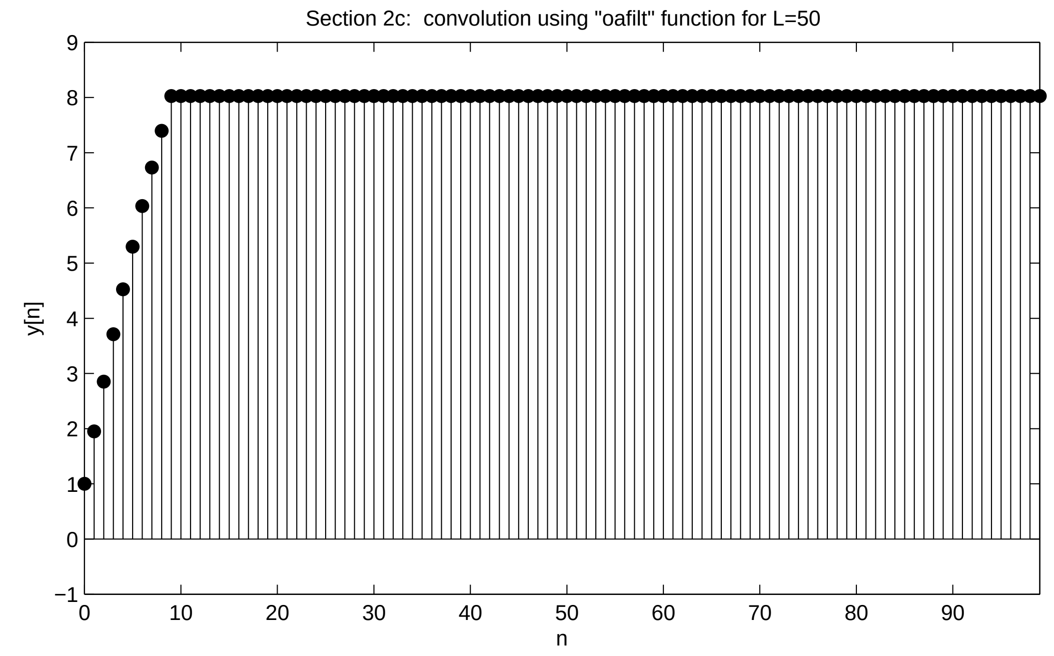


Figure 4: Result of convolving $h[n]$ and $x[n]$ using overlap-add with two 50-point blocks.



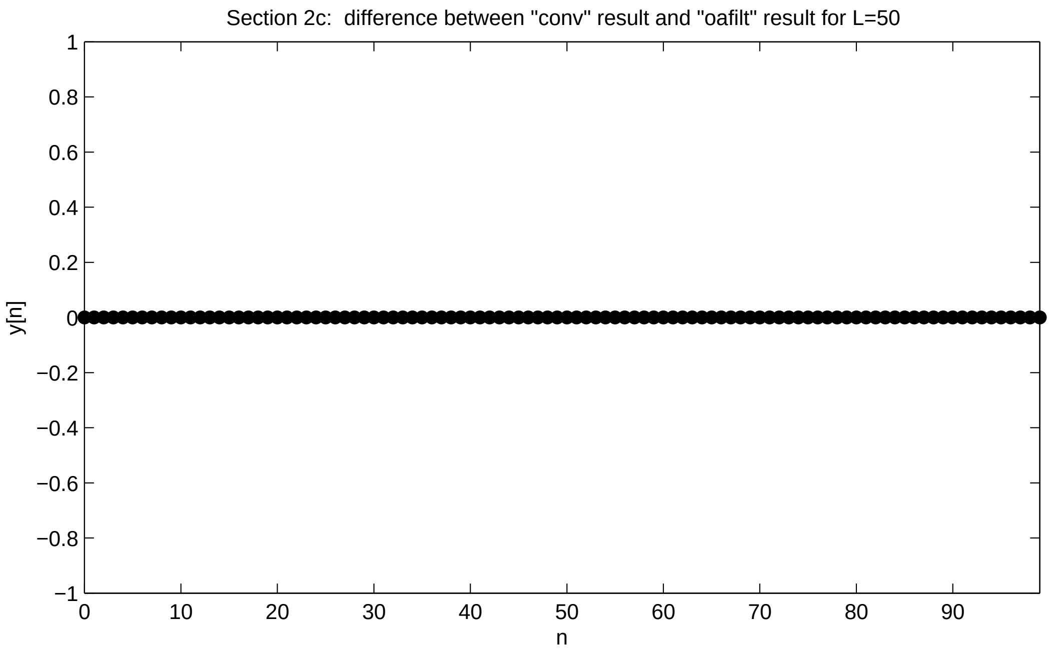
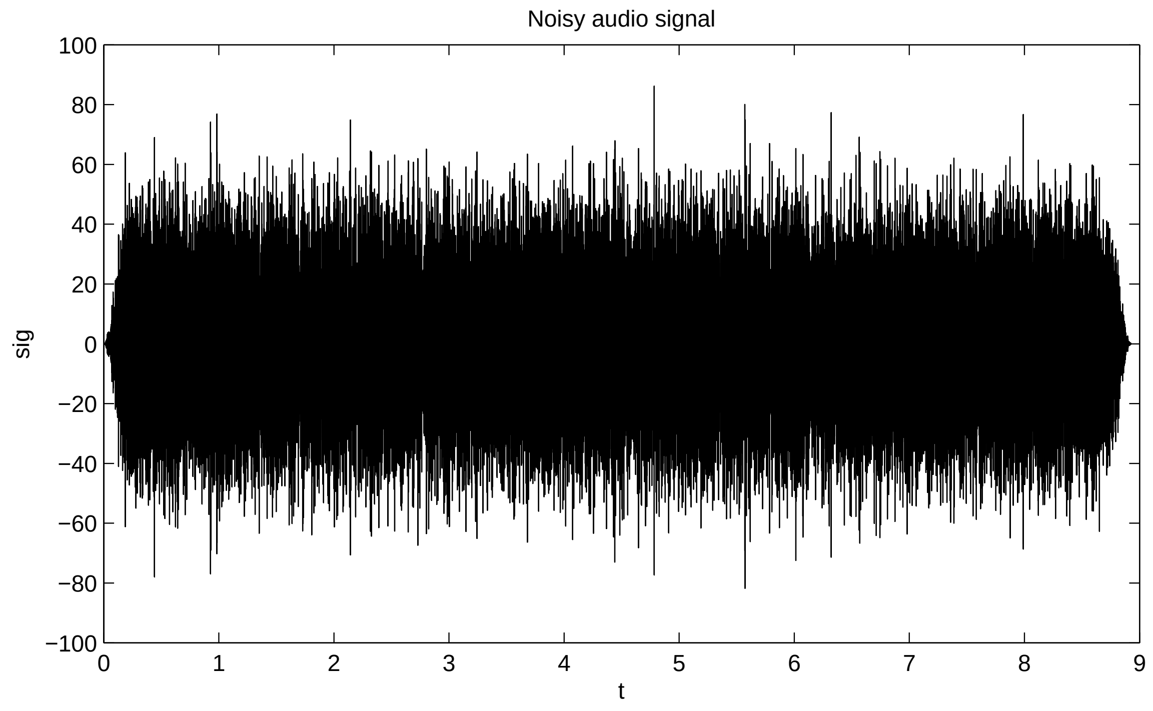


Figure 5: Convolution of $x[n]$ and $h[n]$ using the `oafilt` function. The bottom plot confirms that `oafilt` produces the same result as `conv` in part 2a.



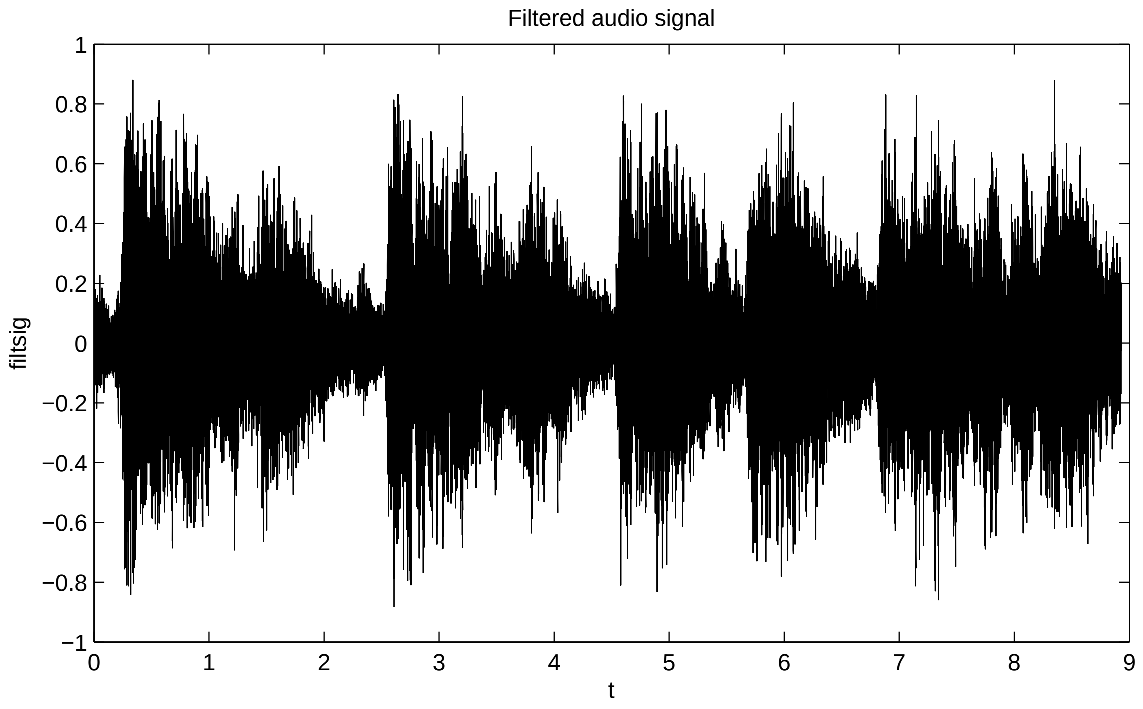
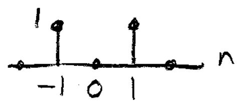


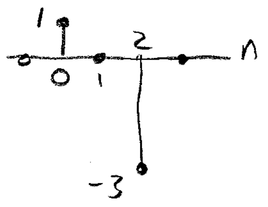
Figure 6: Plots of the noisy signal and the filtered signal. The filtered signal was generated using the `oafilt` command with a block size of 4096.

KEW

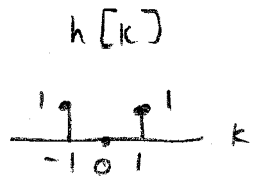
## SECTION 1 BASIC CONVOLUTION EXERCISES

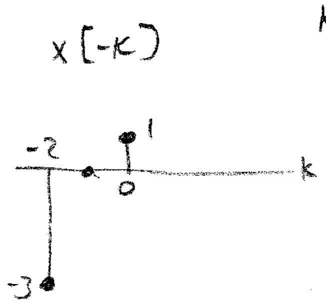
a) $h[n] = \delta[n+1] + \delta[n-1]$ $x[n] = \delta[n] - 3\delta[n-2]$

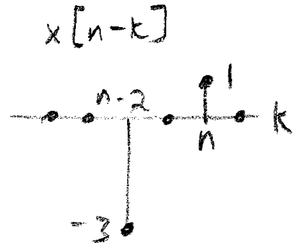




ANALYTICAL CONVOLUTION! $y[n] = \sum_{k=-\infty}^{+\infty} h[k]\,x[n-k]$







REGIONS : NO OVERLAP WHEN $n < -1$

$n = -1$ : 1 POINT OVERLAP

$y[-1] = 1$

$n = 0$ : NO OVERLAP $y[0] = 0$

$n = 1$ : 2-POINT OVERLAP

$y[1] = 1 \cdot 1 + 1(-3) = -2$

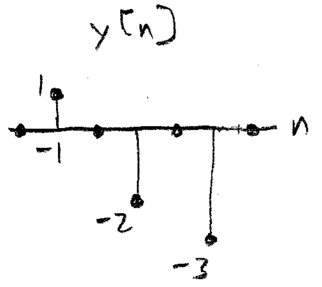
$n = 2$ : NO OVERLAP $y[2] = 0$

$n = 3$ : 1-POINT OVERLAP

$y[3] = -3$

$n > 3$ : NO OVERLAP

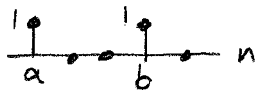
RESULT

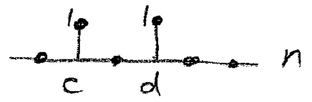


INDEX FOR OUTPUT VECTOR $\boxed{ny = -1 : 3}$

SECTION 1    BASIC CONVOLUTION EXERCISES

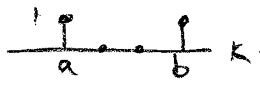
b) $h[n] = \delta[n-a] + \delta[n-b]$    $x[n] = \delta[n-c] + \delta[n-d]$



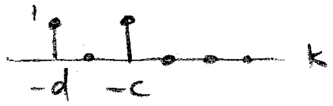


ANALYTICAL CONVOLUTION: $y[n] = \sum_{k=-\infty}^{+\infty} x[k]\,h[n-k]$

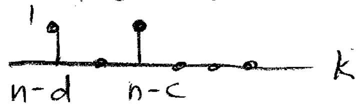
$x[k]$



$h[-k]$



$h[n-k]$



NOTE: ONLY GET OVERLAP AT A DISCRETE # OF POINTS

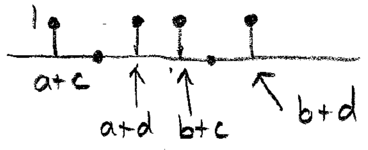
$n-c=a$
$n=a+c$  $\Rightarrow$  $y[a+c]=1$

$n-c=b$
$n=b+c$  $\Rightarrow$  $y[b+c]=1$

$n-d=a$
$n=a+d$  $\Rightarrow$  $y[a+d]=1$

$n-d=b$
$n=b+d$  $\Rightarrow$  $y[b+d]=1$

THUS $y[n] =$



$= \delta[n-(a+c)] + \delta[n-(a+d)] + \delta[n-(b+c)] + \delta[n-(b+d)]$

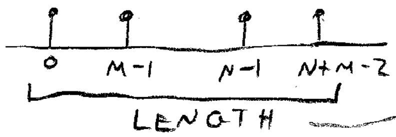
NOTE: IF $b-a = d-c$ THEN $a+d = b+c$ AND TWO IMPULSES OVERLAP

THIS CONDITION MEANS PULSES ARE SAME WIDTH APART

STARTING POINT: $n=a+c$

ENDING POINT: $n=b+d$

CHECK: $a=0$, $b=N-1$, $c=0$, $d=M-1$



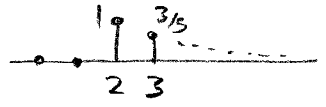
$\longrightarrow$ LENGTH $= (N+M-2-0+1)$
$= N+M-1$ ✓

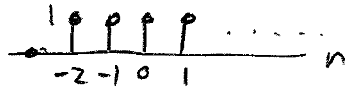
KEW

## SECTION 1 BASIC CONVOLUTION EXERCISES

c-i) ANALYTICAL CONVOLUTION

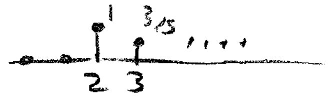
$x[n] = \left(\frac{3}{5}\right)^{n-2} u[n-2]$ $\qquad$ $h[n] = u[n+2]$

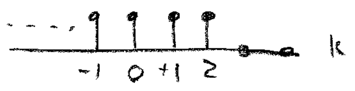




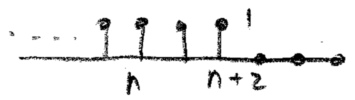
FLIP & SHIFT $h[n]$:

$x[k]$ $\qquad$ $h[-k]$





$h[n-k]$



NO OVERLAP IF $n+2 < 2$
$n < 0$

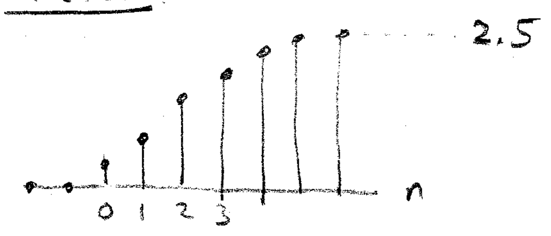
FOR $n \geq 0$

$$y[n] = \sum_{k=2}^{n+2} \left(\frac{3}{5}\right)^{k-2} = \left(\frac{3}{5}\right)^{-2} \sum_{k=2}^{n+2} \left(\frac{3}{5}\right)^{k}$$

$$= \left(\frac{5}{3}\right)^{2} \left[ \frac{\left(\frac{3}{5}\right)^{2} - \left(\frac{3}{5}\right)^{n+3}}{1-\frac{3}{5}} \right]$$

$$= \frac{5}{2}\left[1 - \left(\frac{3}{5}\right)^{n+1}\right]$$

ROUGH SKETCH:

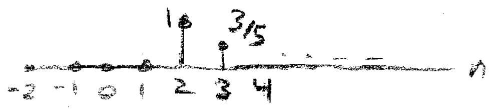


AS $n \to \infty$ $\quad y[n] \to \frac{5}{2}$

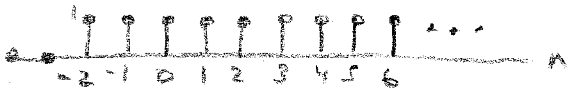
KEW

## SECTION 1 BASIC CONVOLUTION EXERCISES

c-ii) $x[n] = \left(\frac{3}{5}\right)^{n-2} u[n-2]$



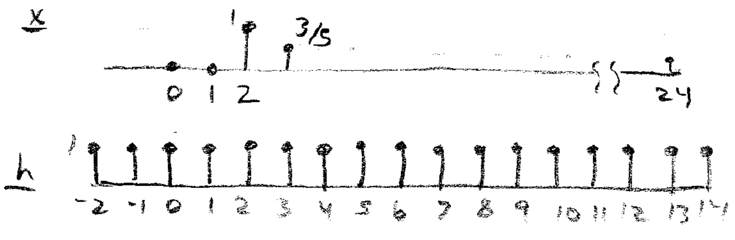
$h[n] = u[n+2]$



STARTING POINT OF $y[n] = h[n] * x[n] = -2+2 = 0$

ENDING POINT " " $= \infty + \infty = \infty$

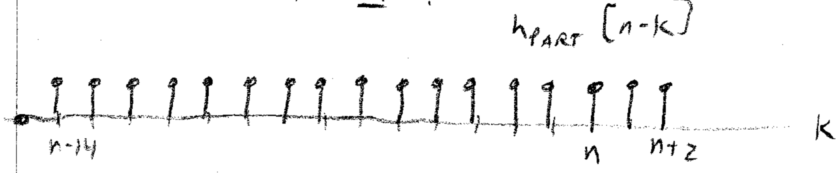
WE WILL USE MATLAB TO COMPUTE PARTIAL CONVOLUTION:



$\underline{x} = x[n]$ FOR $0 \leq n \leq 24$

$\underline{h} = h[n]$ FOR $-2$ TO $14$

FLIP PARTIAL $\underline{h}$:



WHEN $n-14 > 2$ THEN WE'LL START TO GET BAD POINTS BECAUSE WE'LL HAVE ZEROS BEING INCLUDED IN $h[n-k]$ INSTEAD OF ONES (DUE TO FINITE LENGTH OF $\underline{h}$).

$\Rightarrow$ BAD POINTS WHEN $n-14 > 2$

$\boxed{n > 16}$

THE PLOT IN FIG. 2 OF THE REPORT SHOWS THE RESULTS OF THE CONVOLUTION, WITH BAD + GOOD POINTS MARKED.

THE CODE IS IN proj1.m

KEW

SECTION 2 BLOCK CONVOLUTION

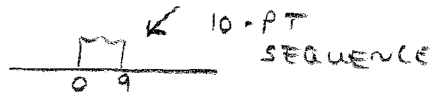
a-i)

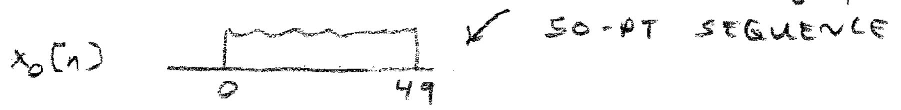
THE PLOT IN FIGURE 3 SHOWS THE CONVOLUTION OF $h[n]$ AND $x[n]$ OVER $0 \leq n \leq 99$.

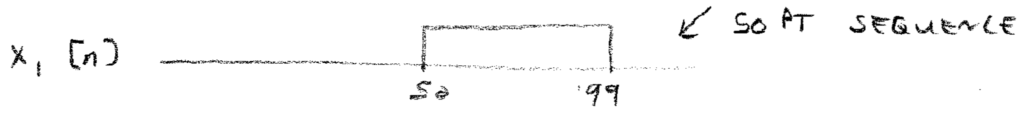
$\boxed{\text{CODE IS IN proj1.m}}$

a-ii)

OVERLAP - ADD







$y_0[n] = h[n] * x_0[n] = 50 + 10 - 1$ PT SEQUENCE.

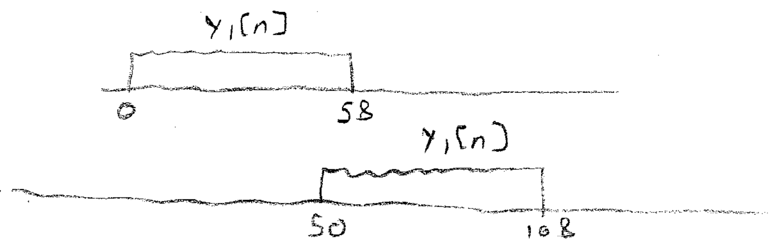
STARTING PT : $0 + 0 = 0$
ENDING PT : $49 + 9 = 58$

$y_1[n] = h[n] * x_1[n] = 50 + 10 - 1$ PT SEQUENCE

STARTING PT : $0 + 50 = 50$
ENDING PT : $9 + 99 = 108$

NOTE: AFTER CONVOLUTION IN MATLAB $y_0[n]$ + $y_1[n]$ ARE BOTH $50+10-1$ PT VECTORS.
WE HAVE TO ADD THEM TOGETHER AFTER PLACING THEM AT CORRECT INDICES.

$y[n] = h[n] * (x_0[n] + x_1[n])$



ADDING YIELDS $y[n]$.

THE PLOT IN FIGURE 4 SHOWS THAT THE CODE IN proj1.m IMPLEMENTS THIS ADDITION CORRECTLY.

```
%%% ECE 320
%%% Matlab Project I Solutions
%%% Fall 2008

%-----------------------------------------------------------------------
%%% Variable that decides whether to print figures or not

printfigs=true;
%-----------------------------------------------------------------------
%%% Basic Convolution Exercises

figure(1);
orient tall;

%%% Part a
nh=-1:1;                                     % indices for h
h=[1 0 1];                                   % h[n]=d[n+1]+d[n-1]
nx=0:2;                                      % indices for x
x=[1 0 -3];                                  % x[n]=d[n]-3d[n-2]
ny=-1:3;                                     % indices for y[n]
y=conv(h,x);                                 % y[n]=x[n]*h[n]
subplot(211)
stem(ny,y,'filled');                         % plot
axis([-1 3 -4 2])
xlabel('n');
ylabel('y[n]');
title('Section 1a: y[n]=x[n]*h[n]')

if printfigs
  print -deps figs/prob1a.eps
end


%%% Part c
figure(2)
orient tall;

nx=0:24;
x=(3/5).^(nx-2).*((nx-2)>=0);                % x[n]=(3/5)^(n-2)u[n-2]

nh=-2:14;
h=((nh+2)>=0);                               % h[n]=u[n+2]

ny=(min(nx)+min(nh)):(max(nx)+max(nh));      % indices for y[n]
y=conv(h,x);                                 % y[n]=h[n]*x[n]

badlo=find(ny==17);                          % bad points start at n=17
good_ind=1:badlo-1;
bad_ind=badlo:length(ny);

subplot(211)
hgood=stem(ny(good_ind),y(good_ind),'filled');
hold on;
hbad=stem(ny(bad_ind),y(bad_ind),'x');
hold off;
xlabel('n');
ylabel('y[n]');
title('Section 1c:  Good vs. Bad points due to finite length convolution')
legend([hgood(1),hbad(1)],'Good points','Bad points')
axis([min(ny) max(ny) 0 2.5])

if printfigs
  print -deps figs/prob1c.eps
end
```

```
%-----------------------------------------------------------------------
%%% Section 2:  Block Convolution

figure(3)
orient tall;

%%% Part 2a-i
nh=0:9;                                     % h[n]=(0.95)^n(u[n]-u[n-10])
h=(.95).^nh;

nx=0:99;                                    % x[n]=u[n]
x=nx>=0;

ny=(min(nh)+min(nx)):(max(nh)+max(nx)); % Vector of indices for y[n]
y=conv(x,h);                                % y[n]=x[n]*h[n]

subplot(211)                                % plot
stem(ny,y,'filled')
axis([0 99 -1 9]);
xlabel('n');
ylabel('y[n]');
title('Section 2a-i:  convolution of x[n] and h[n] via conv command')

if printfigs
  print -deps figs/prob2ai.eps
end

%%% Part 2a-ii
figure(4)
orient tall

P=10;
L=50;
x0=x(1:L);                                  % first section of x[n]
x1=x(L+1:length(x));                        % second section of x[n]
y0=conv(h,x0);                              % first section of y[n]
y1=conv(h,x1);                              % second section of y[n]

yaii=zeros(1,P+length(x)-1);                % combine sections
yaii(1:L+P-1)=yaii(1:L+P-1)+y0;             % y0 starts at 0 and ends at L+P-1
yaii(L+1:2*L+P-1)=yaii(L+1:2*L+P-1)+y1; % add y1 which starts at L and
                                            % ends at 2L+P-2
subplot(211)
stem(ny,yaii,'filled')
axis([0 99 -1 9]);
xlabel('n')
ylabel('y[n]')
title('Section 2a-ii:  convolution of x[n] and h[n] via overlap-add')

if printfigs
  print -deps figs/prob2aii.eps
end

%-----------------------------------------------------------------------
%%% Section 2c:  Redoing convolution using oafilt function

figure(5)
orient tall

%%% Part 2c
yc=oafilt(h,x,5);                           % Compute y using oafilt fxn

subplot(211)
```

```
stem(ny,yc,'filled')
axis([0 99 -1 9]);
xlabel('n')
ylabel('y[n]')
title('Section 2c:  convolution using "oafilt" function for L=50')

subplot(212)
stem(ny,yc-y','filled')
axis([0 99 -1 1]);
xlabel('n')
ylabel('y[n]')
title('Section 2c:  difference between "conv" result and "oafilt" result for L=50')

if printfigs
  print -deps figs/prob2c.eps
end

%-----------------------------------------------------------------------
%%% Section 3:  Filtering of a noisy audio signal

figure(6)
orient tall

load proj1_data                            % load data
filtsig=oafilt(h,sig,4096);                % filter the signal

%%% Play the noisy and filtered signals
soundsc(sig,fs);
soundsc(filtsig,fs);

subplot(211)
t=0:1/fs:(length(sig)-1)/fs;
plot(t,sig);
xlabel('t')
ylabel('sig')
title('Noisy audio signal')

subplot(212)
t=0:1/fs:(length(filtsig)-1)/fs;
plot(t,filtsig);
xlabel('t')
ylabel('filtsig')
title('Filtered audio signal')

if printfigs
  print -deps figs/prob3.eps
end
```

```
function y=oafilt(h,x,L)
%%  OAFILT  Function that implements overlap-add block convolution
%%          using Matlab's "conv" command for convolution of the
%%          individual blocks
%%
%%  Function Call:  y=oafilt(h,x,L)
%%
%%  Input Variables:
%%         h = vector of filter coefficients (length P)
%%         x = input signal to be broken into blocks
%%         L = blocksize (must be greater than P)
%%
%%  Output Variables:
%%         y = filtered sequence
%%

%%  File          :  oafilt.m
%%  Author        :  Kathleen Wage
%%  Date          :  April 8, 2002
%%  Updated       :  September 23, 2008
%-------------------------------------------------------------------
%%% Error checking
P=length(h);

if L<=P
  error('OAFILT:  Block length should be longer than the filter length')
end

%%% Preliminaries

h=h(:);                                      % make h a column vector
x=x(:);                                      % make x a column vector
nr=ceil(length(x)/L);                        % compute # of blocks
y=zeros(P+length(x)-1,1);                    % initialize output vector

%%% Filtering:  loop through first nr-1 blocks
for r=1:nr-1
  ind=(r-1)*L;
  xr=x(ind+(1:L));                           % get section of x
  yr=conv(h,xr);                             % compute section of y
  y(ind+(1:(L+P-1)))=y(ind+(1:(L+P-1)))+yr; % overlap with previous vectors
end

%%% Compute last section (which may be shorter than previous)
ind=(nr-1)*L;
xr=x((ind+1):length(x));
yr=conv(h,xr);
y(ind+(1:length(yr)))=y(ind+(1:length(yr)))+yr;

return
```