Linear Time Invariant (LTI) and Convolution Assignment 2:

Colt Thomas

June 19, 2020

MATLAB Problems Section

Problem 1

In Homework problem 2.1 *and* 2.2 *we focus on the creation and display of discrete signals. Start by finding and down loading the file DSP_mFiles2019.zip in Bb and extract the archived m-files in a directory and include the directory in your MATLAB path. These files consist of is an m-files library created by our text author to assist in DSP MATLAB computations. You also need to find and down load the file "DSP m-file Help Manual.pdf." Study this manual before addressing the homework problems since it contains selected instructions that are helpful in creating and operating with discrete signals*
Problem 1 has three parts, A, B, and C. Plot the three graphs described in A, B, and C in a single figure using the MATLAB subplot command. In all cases assume $n1 = 20, n2 = 39, f = .08$ *cycles per sample and the index* n *is an array defined by the MATLAB command* $n = [0 : n2]$. . *All plots should contain appropriate (readable size) x-axis, y-axis, and title labels with data displayed by a symbol like "o," and not connected with lines(Hints for MATLAB commands are shown at the end of the problem 1 statement). All of plots shown in this problem statement are conceptual representations and not to scale.*

Part A

Create and document a MATLAB script file to generate the graphs for w1(n), w2(n),w3(n),but display only the discrete signal w3(n)in subplot(3,1,1).Graphs for parts B and C will be displayed in subsequent subplot(3,1,2)and subplot(3,1,3).

This subplot will create a step function $w_3(n)$ that results from summing $w_1(n)$ and $w_2(n)$. See the top plot in Figure 1 and the MATLAB code section at the very end for more details.

Part B

Create and document a MATLAB script file to generate and display the discrete signal $s_1(n)$

The second plot is of sinusoid $s_1(n)$. See the middle plot in Figure 1 and the MATLAB code section for more details.

Part C

Create and display a MATLAB script file to and create and display the discrete signal $s_2(n)$ *referred to as a discrete pulsed sine wave signal or a pulsed sine wave sequence.*

The third plot is of the pulsed sinusoid sequence $s_2(n)$. See the bottom plot in Figure 1 and the MATLAB code section for more details.

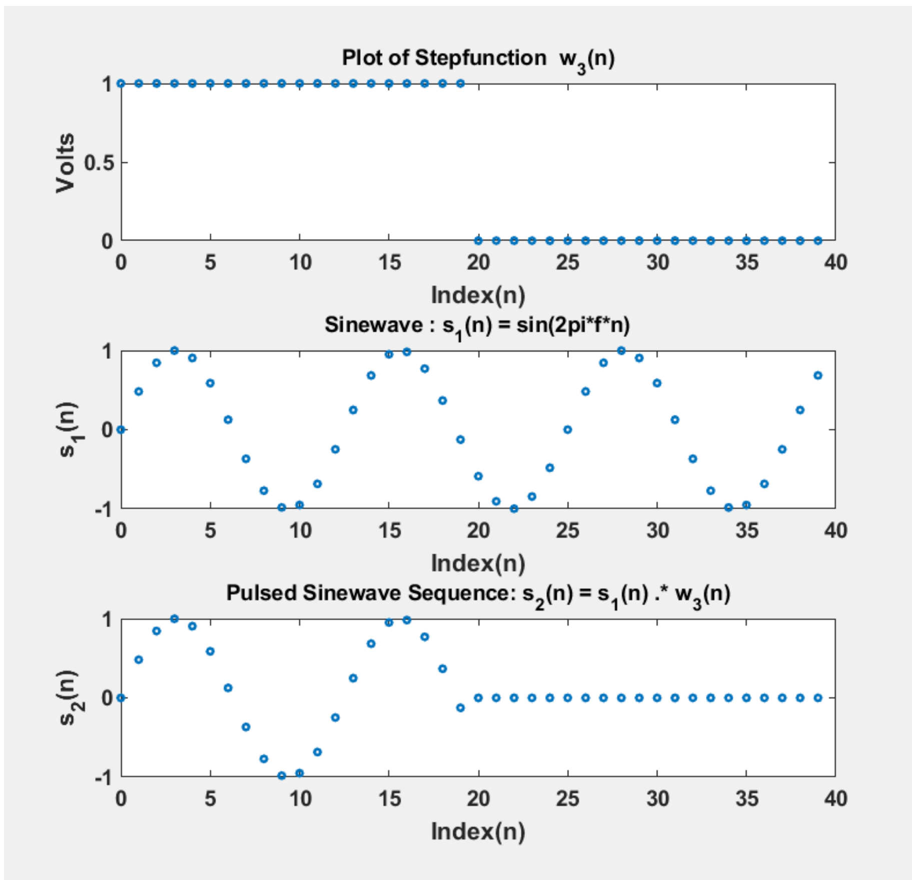


Figure 1: Subplots of functions from Parts A, B, and C.

Problem 2

In this problem create and document a MATLAB script file to generate a discrete signal s2(n)consisting multiple replicas of s1(n)as depicted below. In this problem you are to produce 3 replicas of the sequence s1(n)using the technique shown on Extraction page 15 in the DSP m-file help guide. Here xtilde is replaced by $s_1(n)$ *and* $P = 3$ *to get the MATLAB display below. This discrete signal could referred to as a train of pulsed sine waves or a finite periodic sequence of sine wave pulses.*

This task can be accomplished by concatenating the signal repeatedly in an array, or doing an index trick. See the MATLAB code section below for details. The resulting signal is in Figure 2

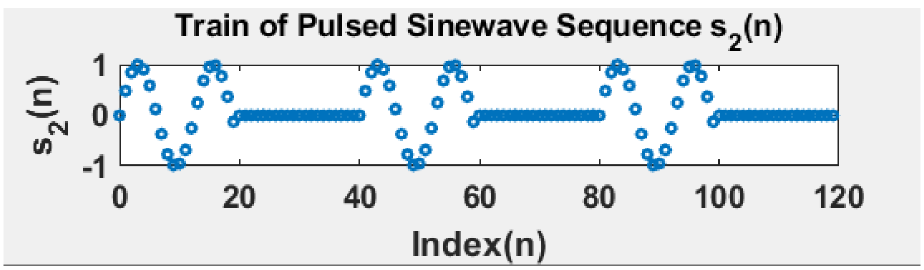


Figure 2: A train of pulsed sine waves created by using concatenation in MATLAB.

Textbook Questions

Problem 3

Book - 2.1

A discrete-time signal x(n) is defined as

$$x(n) = \begin{cases} 1 + \frac{n}{3}, & -3 <= n <= -1 \\ 1, & 0 <= n <= 3 \\ 0, & \text{elsewhere.} \end{cases} \tag{1}$$

(a) Determine its values and sketch the signal x(n)

The plot of x(n) can be found in 3.

(b) Sketch the signals that result if we:

1. First fold $x(n)$ and then delay the resulting signal by four samples.

2. First delay $x(n)$ by four samples and t hen fold the resulting signal.

Figure 3 shows $x(n)$ undergoing the respective operations in (1) and (2).

(c) Sketch the signal $x(-n + 4)$.

Figure 4 shows signals $x(n)$ and $x(-n + 4)$. Notice how $x(-n + 4)$ is $x(-n)$ delayed by 4. This is also equivalent to the middle graph in Figure 3.

(d) Compare the results in parts (b) and (c) and derive a rule for obtaining the signal $x(-n + k)$ from $x(n)$.

There is a note on pg 52 of our book stating that the operations of folding and time delaying a signal are not commutative. A rule of thumb is to first fold, then delay. Let's suppose we have $x(n)$ that we want to modify. $x(-n+4)$ and $x(-(n + 4))$ are the mathematical equivalents of folding/delaying and delaying/folding respectively.

(e) Can you express the signal $x(n)$ in terms of signals $\delta(n)$ and $u(n)$?

$$x(n) = \frac{1}{3}\delta(n + 2) + \frac{2}{3}\delta(n + 1) + u(n) - u(n - 4)$$

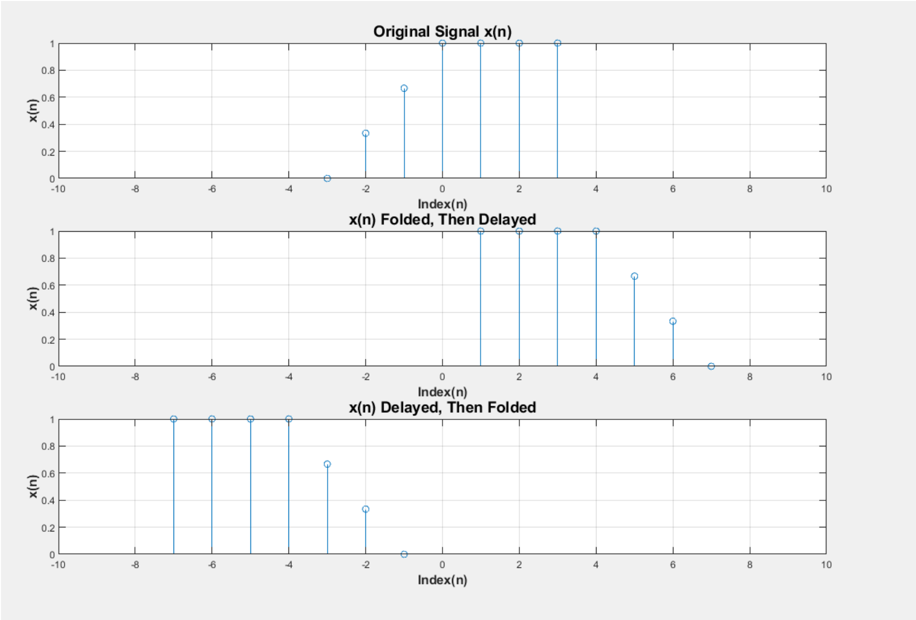


Figure 3: $x(n)$ undergoes folding and delay operations. Notice how order of operation matters.

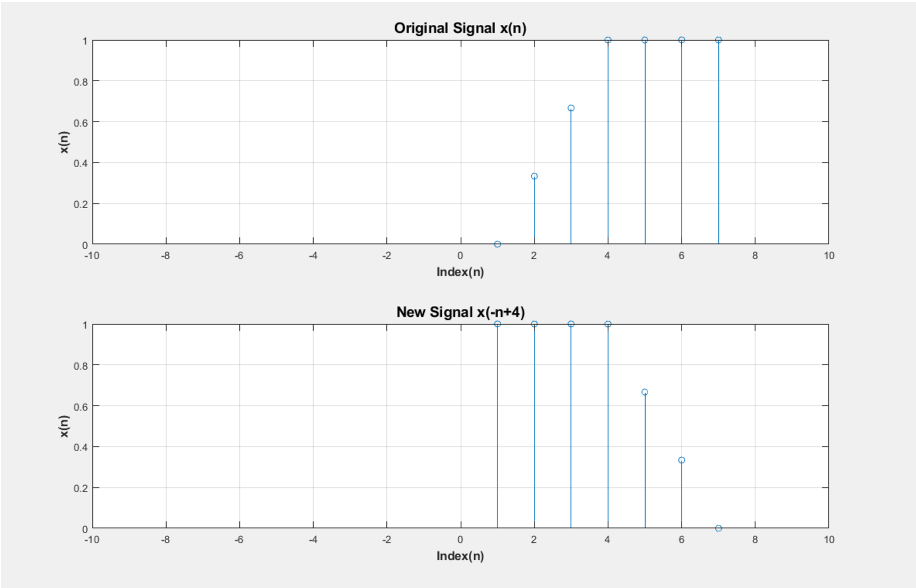
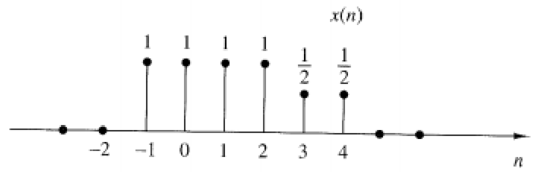


Figure 4: $x(-n+4)$ plotted in comparison to the original $x(n)$

Book - 2.2

A discrete-time signal $x(n)$ is shown in Fig. P2.2. Sketch and label carefully each of the following signals:

Figure P2.2



(a)$x(n-2)$ **(b)**$x(4-n)$ **(c)**$x(n+2)$ **(d)**$x(n)u(2-n)$ **(e)**$x(n-1)\delta(n-3)$ **(f)**$x(n^2)$ **(g)**even part of $x(n)$ **(h)**odd part of $x(n)$

In Figure 5 are several variations of $x(n)$ as specified in parts a-h for this problem. Notice that eqn (f) is scaled differently from the others due to non-linear $x(n^2)$

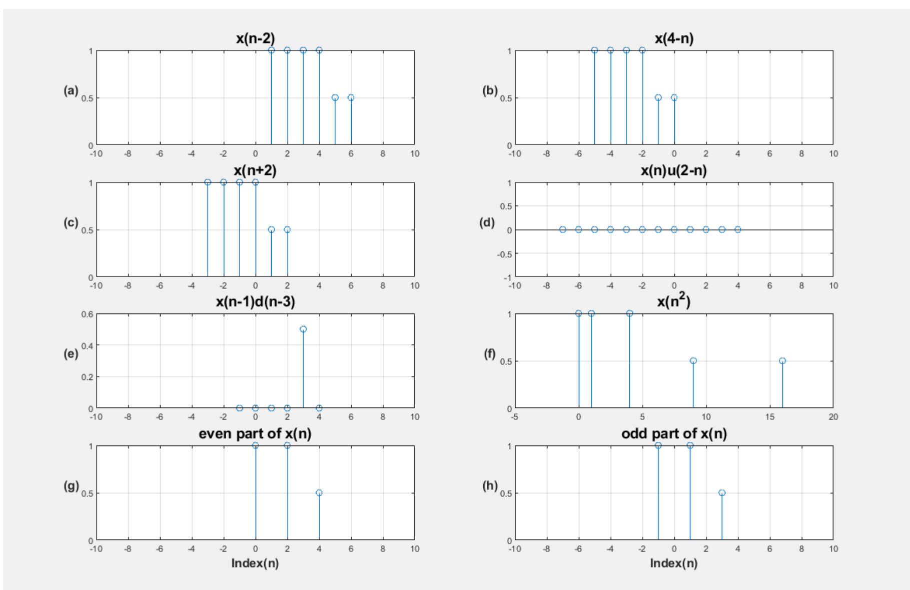


Figure 5: Parts a-h of problem 2.2; Various operations on $x(n)$

Problem 4

Book 2.3

Show that

(a) $\delta(n) = u(n) - u(n-1)$

Let's start with the definitions of the Unit Step Function and the Dirac Delta function. The definition of the Dirac Delta function is:

$$\delta(n) = \begin{cases} 1, & \text{for } n = 0 \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The Unit Sep function is defined as:

$$u(n) = \begin{cases} 1, & \text{for } n >= 0 \\ 0, & \text{for } n < 0. \end{cases} \tag{3}$$

We can define $u(n)$ in terms of $\delta(n)$ as follows:

$$u(n) = \sum_{k=0}^{\infty} \delta(n-k) \tag{4}$$

Plug into our initial equation:

$$\begin{gathered} \delta(n) = u(n) - u(n-1) \\ \delta(n) = \sum_{k=0}^{\infty} \delta(n-k) - \sum_{k=1}^{\infty} \delta(n-k) \\ \delta(n) = \delta(n) + \sum_{k=1}^{\infty} \delta(n-k) - \sum_{k=1}^{\infty} \delta(n-k) \\ \delta(n) = \delta(n) \end{gathered}$$

(b) $u(n) = \sum_{k=-\infty}^{n} \delta(k) = \sum_{k=0}^{\infty} \delta(n-k)$

By definition of Equation 2 , we notice that $\delta(n)$ is 1 for $n = 0$ only. The previous problem shows that a unit step function can be defined as a series of Dirac delta functions in Equation 4. Lets sort out the middle portion of the question:

$$u(n) = \sum_{k=-\infty}^{n} \delta(k)$$

Expand the summation and we get something like:

$$u(n) = \delta(-\infty) + ... + \delta(n-k) + ... + \delta(n-2) + \delta(n-1) + \delta(n)$$

We can see that this is a series of delayed deltas that actually spans from $0 <= n <= \infty$, rather than the $-\infty$ that it appears to go to at first. This matches what we have for Equation 4 above.

Book 2.4

Show that any signal can be decomposed into an even and an odd component. Is the decomposition unique? Illustrate your arguments using the signal

$$x(n) = \{2, 3, 4, 5, 6\}$$

We can use the principle of superposition to decompose $x(n)$ into even and odd components. From the book equations 2.1.26 and 2.1.27 the even and odd components can be expressed as:

$$x_e(n) = \frac{1}{2}[x(n) + x(-n)]$$

$$x_o(n) = \frac{1}{2}[x(n) - x(-n)]$$

For $x_e(n)$:

$$x_e(n) = \frac{1}{2}[\{\underset{\uparrow}{2}, 3, 4, 5, 6\} + \{6, 5, 4, 3, \underset{\uparrow}{2}\}]$$

$$x_e(n) = \frac{1}{2}[\{6, 5, 4, 3, \underset{\uparrow}{4}, 3, 4, 5, 6\}]$$

$$x_e(n) = \{3, 2.5, 2, 1.5, \underset{\uparrow}{2}, 1.5, 2, 2.5, 3\}$$

For $x_o(n)$:

$$x_o(n) = \frac{1}{2}[x(n) - x(-n)]$$

$$x_o(n) = \frac{1}{2}[\{\underset{\uparrow}{2}, 3, 4, 5, 6\} - \{6, 5, 4, 3, \underset{\uparrow}{2}\}]$$

$$x_o(n) = \frac{1}{2}[\{-6, -5, -4, -3, \underset{\uparrow}{0}, 3, 4, 5, 6\}]$$

$$x_o(n) = \{-3, -2.5, -2, -1.5, \underset{\uparrow}{0}, 1.5, 2, 2.5, 3\}$$

Now we will combine the even and odd components:

$$x(n) = x_e(n) + x_o(n)$$

$$x(n) = \{3, 2.5, 2, 1.5, \underset{\uparrow}{2}, 1.5, 2, 2.5, 3\} + \{-3, -2.5, -2, -1.5, \underset{\uparrow}{0}, 1.5, 2, 2.5, 3\}$$

$$x(n) = \{2, 3, 4, 5, 6\}$$

Book 2.7: a,b,c,d,g,h,k

A discrete-time system can be

(1) Static or dynamic
(2) Linear or nonlinear
(3) Time invariant or time varying
(4) Causal or noncausal
(5) Stable or unstable

Examine the following systems with respect to the properties above.

(a)$y(n) = cos[x(n)]$

(1) Static/memoryless - does not require a previous input (n-1, etc)
(2) Non-Linear - $cy(n) \neq cos[cx(n)]$ for constant c
(3) Time-Invariant - $y(n-k) = cos[x(n-k)]$ for all n, k
(4) Causal - no future input (n+k) required
(5) Stable - $y(n)$ is bounded such that $0 <= y(n) <= 1$ for all n

(b)$y(n) = \sum_{k=-\infty}^{n+1} x(k)$

(1) Dynamic - Relies on 1 future input and infinite history of inputs
(2) Linear - Scalable: $cy(n) = \sum_{k=-\infty}^{n+1} cx(k) = c\sum_{k=-\infty}^{n+1} x(k)$ and summations follow superposition property.
(3) Time Varying - relies on infinite history
(4) Noncausal - relies on future input x(n+1)
(5) Unstable - The summation is does not converge to a finite value.

(c)$y(n) = x(n)cos(\omega_o n)$

(1) Static - does not require past or future values
(2) Linear - Scalable and follows superposition property
(3) Time-Varying - $y(n-1) \neq x(n-1)cos(\omega_o(n-1))$ the cosine also varies when time is advanced/delayed
(4) Causal - does not require future input
(5) Stable - With a bounded input $x(n)$, we get a bounded output $|y(n)| < \infty$ since $x(n)$ is multiplied by a constant $0 <= cos(\omega_o n) <= 1$

(d)$y(n) = x(-n+2)$

(1) Dynamic - it requires future values
(2) Linear - $y(n) + y(u) = x(-n+2) + x(-u+2)$ for any integer u, and $cy(n) = cx(-n+2)$ for constant c
(3) Time invariant - $y(n-k) = x(-(n-k)+2)$ for all k
(4) Non-Causal - relies on future input (-n+2)

(5) Stable - if the input $x(n)$ is bounded, so is the output. This is because y(n) only folds and advances the signal. It doesn't do any non-linear modifications to the input.

(g) $y(n) = |x(n)|$

(1) Static - doesn't require past or future values
(2) Nonlinear - this follows the superposition principle, but $cy(n) \neq |cx(n)|$ for negative values of c
(3) Time invariant - $y(n-k) = |x(n-k)|$ for all integers k
(4) Causal - does not rely on future input
(5) Stable - The absolute value of anything is a finite value as long as the input is finite.

(h) $y(n) = x(n)u(n)$

(1) Static - does not require past or future input
(2) Linear - The unit step function follows superposition and is scalable
(3) Time invariant - $u(n)$ will shift in time at the same rate $x(n)$ does.
(4) Causal - does not require future input
(5) Stable - the unit step function will multiply $x(n)$ by 1 for all $n >= 0$ and 0 for $n < 0$. If $x(n)$ is bounded, $y(n)$ will be too.

(k) $y(n) = \begin{cases} x(n), & \text{if } x(n) >= 0 \\ 0, & \text{if } x(n) < 0 \end{cases}$
(1) Static - does not rely on past or future input
(2) Nonlinear - $cy(n) \neq cx(n)$ for $c < 0$
(3) Time invariant - The output varies depending on the sign of $x(n)$, but not n
(4) Causal - does not require future input
(5) Stable - If $x(n) >= 0$ for all n and $x(n)$ is stable, then $y(n)$ is stable. 0 is finite for $x(n) < 0$

Problem 5

Book 2.13

Show that the necessary and sufficient condition for a relaxed LTI system to be BIBO stable is

$$\sum_{n=-\infty}^{\infty} |h(n)| <= M_n < \infty$$

for some constant M_n

$h(n)$ is the impulse response of a system, such that output $y(n) = h(n)$ when

$x(n) = \delta(n)$ for system $y(n) = x(n) * h(n)$. This system can also be expressed at:

$$y(n) = \sum_{k=-\infty}^{\infty} h(k)x(n-k)$$

Which is the definition of convolution. if we take the absolute value of both sides of the equation we get:

$$|y(n)| = \Big|\sum_{k=-\infty}^{\infty} h(k)x(n-k)\Big|$$

The absolute value of a summation will always be less than or equal to the original value. We can also distribute the absolute value operation. This turns our equation into an inequality:

$$|y(n)| <= \sum_{k=-\infty}^{\infty} |h(k)||x(n-k)|$$

If $x(n)$ is bounded then there is a finite value M_x such that $|x(n)| <= M_x$. If we substitute this in to our equation we get:

$$|y(n)| <= M_x \sum_{k=-\infty}^{\infty} |h(k)|$$

In order for the output $y(n)$ to be bounded, $\sum_{k=-\infty}^{\infty} |h(k)|$ must be bounded such that:

$$\sum_{n=-\infty}^{\infty} |h(n)| <= M_n < \infty$$

Where M_n is the upper bound of our impulse response. The take-away from this problem is that $h(n)$ must be <u>absolutely summable</u> in order to have a bounded output, or an LTI system.

Problem 6

Book 2.17

Compute and plot the convolutions $x(n) * h(n)$ and $h(n) * x(n)$ for the pairs of signals shown in Fig. P2.17.

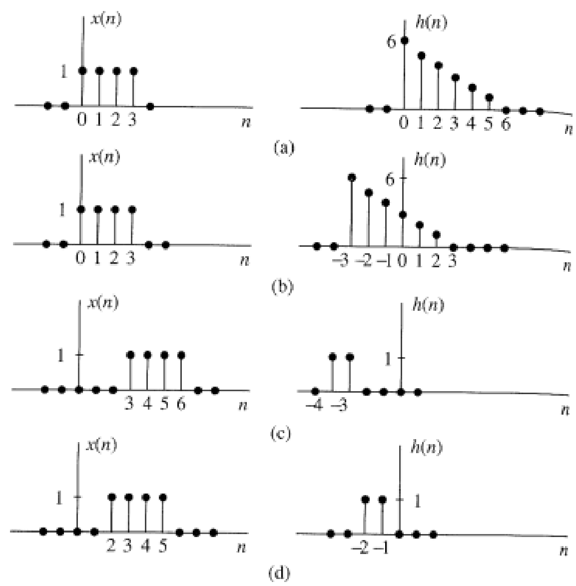


Figure P2.17

One of the properties of the convolution is the commutative law, where $y(n) = h(n) * x(n) = x(n) * h(n)$ In MATLAB we can use the conv() function to perform these convolutions, but for kicks I've tried swapping $x(n)$ and $h(n)$ in the MATLAB code below (see the section for problem 6). In Figure 6 you can see the plots of each convolution. Also note that because of the shifting property of the convolution, a shifted input $h(n)$ or $x(n)$ by k will also shift the output $y(n)$ by k.

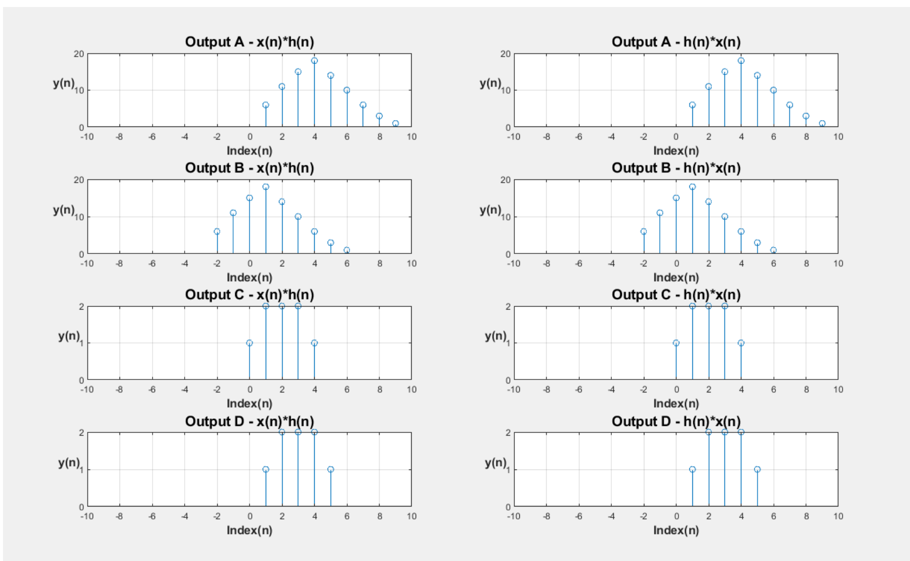


Figure 6: Plots of the above signals convolved with each other. Note that the order of convolution doesn't change $y(n)$

Problem 7

Book 2.34

Consider a system with impulse response

$$h(n) = \begin{cases} (\frac{1}{2})^n, & 0 <= n <= 4 \\ 0, & \text{elsewhere} \end{cases}$$

Determine the input $x(n)$ for $0 <= n <= 8$ that will generate the output sequence

$$y(n) = \{\ 1, 2, 2.5, 3, 3, 3, 2, 1, 0, ...\}$$

We can solve for $x(n)$ in $y(n) = x(n) * h(n)$ to find the input of an LTI system. This is tricky since we are trying to find $x(n)$ instead of $y(n)$. One way to think of convolution is by breaking the input $x(n)$ into a series of impulses, and summing the impulse responses $h(n-k)$ over time:

$$v_o = x(0) * h(n-0) = x(0)[1, 0.5, 0.25, 0.125, 0.0625, 0, 0, 0, 0, 0, 0, 0]$$

$$v_1 = x(1) * h(n-1) = x(1)[0, 1, 0.5, 0.25, 0.125, 0.0625, 0, 0, 0, 0, 0, 0]$$
$$v_2 = x(2) * h(n-2) = x(2)[0, 0, 1, 0.5, 0.25, 0.125, 0.0625, 0, 0, 0, 0, 0]$$
$$v_3 = x(3) * h(n-3) = x(3)[0, 0, 0, 1, 0.5, 0.25, 0.125, 0.0625, 0, 0, 0, 0]$$
$$v_4 = x(4) * h(n-4) = x(4)[0, 0, 0, 0, 1, 0.5, 0.25, 0.125, 0.0625, 0, 0, 0]$$
$$v_5 = x(5) * h(n-5) = x(5)[0, 0, 0, 0, 0, 1, 0.5, 0.25, 0.125, 0.0625, 0, 0]$$
$$v_6 = x(6) * h(n-6) = x(6)[0, 0, 0, 0, 0, 0, 1, 0.5, 0.25, 0.125, 0.0625, 0]$$
$$v_7 = x(7) * h(n-7) = x(7)[0, 0, 0, 0, 0, 0, 0, 1, 0.5, 0.25, 0.125, 0.0625]$$
$$...$$
$$v_k = x(k) * h(n-k)$$

$y(n)$ will be a summation $\sum_{k=0}^{N} v_k$. To find $y(k)$, we sum all the values of the kth indices for each v_k. In this case $x(n)$ in unknown so we solve the system of equations. One trick we could do is if we think of the $h(n)$ vectors as a 2d array, we could transpose it and multiply it by a vector of $x(n)$ components. I calculated this line-by-line in my calculator starting with $x(0) = y(0)$, and came to an approximate solution:

$$x(n) = [1, 3/2, 3/2, 7/4, 3/2, 49/32, 35/64, 3/64, -57/128, 3/64, ...]$$

If you run the MATLAB script you will see that eventually you can converge to the exact $y(n)$ value as you increase k to ∞ in the convolution summation.

Matlab Code - Problem 1

```
% Assignment 2
addpath('./DSP_mFiles2019/');

fh = figure(1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Problem 1 - Graph w_3(n)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Mathematical assumptions for all parts
n1 = 20; % index of half way point
n2 = 39; % last index value
f = 0.08; % cycles per sample
n = (0:n2); % index range for plots

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ---- Part A : Plot Step Function ----
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% definitions of w_1(n) and w_2(n) step functions.stepseq
    () is a given
% MATLAB function requiring the step function range as
    input and outputs
% an array of the step function w_n and the index array n
    for plotting
[w1,n] = stepseq(0,0,n2);
[w2,n] = stepseq(n1,0,n2);
w3 = w1 - w2; % arrays can undergo mathematical
    operations

% subplot1 creation - subplots are used to create a
    single figure with
% several plots rather than several separate figures. In
    this case we have
% a 3 x 1 matrix of figures, and this is the first/top
    one.
subplot(3,1,1)

% The plot parameters create a stem plot and set plot
    characteristics
ph=plot(n,w3,'marker','o','markersize',4,'Linewidth',2,'
    LineStyle','none');

% labels and titles. set() will adjust parameters of gca
    in this case,
```

```
% which is an object for "get current axis"
set(gca,'FontSize',14,'FontWeight','Bold')
title('Plot of Stepfunction  w_3(n)','FontSize',14,'FontWeight','Bold');
xlabel('Index(n)','FontSize',16,'FontWeight','Bold')
ylabel('Volts','FontSize',16,'FontWeight','Bold')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%--- Part B : Plot Sinusoid ---
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% MATLAB comes with standard trig functions and variables such as pi and
% sin/cos. Notice that the input is assumed to be in radians.

s1 = sin(2*pi*f*n);

% subplot2 - Copied from Part A, except notice that the p in subplot(m,n,p)
% incremented. This is our second/middle figure.
subplot(3,1,2)
ph=plot(n,s1,'marker','o','markersize',4,'Linewidth',2,'LineStyle','none');

% labels
set(gca,'FontSize',14,'FontWeight','Bold')
title('Sinewave : s_1(n) = sin(2pi*f*n)','FontSize',14,'FontWeight','Bold');
xlabel('Index(n)','FontSize',16,'FontWeight','Bold')
ylabel('s_1(n)','FontSize',16,'FontWeight','Bold')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Part C : Plot Pulsed Sinewave Sequence ---
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Below we are doing a element-wise multiply (rather than a matrix
% multiplication) of the vectors s1 and w3. Note that s1 and w3 MUST be
%equal in length.
s2 = s1.*w3; % the .* makes it an element-wise multiply

% subplot3
subplot(3,1,3)
ph=plot(n,s2,'marker','o','markersize',4,'Linewidth',2,'
```

```
    LineStyle','none');

% labels
set(gca,'FontSize',14,'FontWeight','Bold')
title('Pulsed Sinewave Sequence: s_2(n) = s_1(n) .* w_3(n)','FontSize',14,'FontWeight','Bold');
xlabel('Index(n)','FontSize',16,'FontWeight','Bold')
ylabel('s_2(n)','FontSize',16,'FontWeight','Bold')


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Problem 2 - Graph series of pulsed sinewaves
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% To make a repeated signal, we can concatenate the original signal and
% make a series. Notice that I still use the same variable s2 without any
% issues
% s2 = [s2,s2,s2];

% A more dynamic way to do this is to work with the indexing:
P = 3; % The number of times we repeat the signal
s2 = s2' * ones(1,P);
s2 = s2(:); % long column vector
s2 = s2'; % transpose to long row vector
n = (0:length(s2)-1); % the length() command is useful for dynamic code

fh=figure(2); % creates a separate figure from problem 1
ph=plot(n,s2,'marker','o','markersize',4,'Linewidth',2,'LineStyle','none');

% labels and style are the same as usual
set(gca,'FontSize',14,'FontWeight','Bold')
title('Train of Pulsed Sinewave Sequence s_2(n)','FontSize',14,'FontWeight','Bold');
xlabel('Index(n)','FontSize',16,'FontWeight','Bold')
ylabel('s_2(n)','FontSize',16,'FontWeight','Bold')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Problem 3 - Book Probs 2.1 and 2.2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%------------------------------
% Problem 2.1 - part a
```

```
%-------------------------------------------
padding = 0;
no = (-3:-1);
xo = 1+no/3;
no = ((-3):(3));
xo = [xo,ones(1,4)];

fh=figure(3);
%ph=plot(n,x,'marker','o','markersize',4,'Linewidth',2,'
    LineStyle','none');
subplot(3,1,1, 'align')
ph=stem(no,xo);
grid on;
xlim([-10 10])
title('Original Signal x(n)','FontSize',14,'FontWeight','
    Bold');
xlabel('Index(n)','FontSize',12,'FontWeight','Bold')
ylabel('x(n)','FontSize',12,'FontWeight','Bold')


%-------------------------------------------
% Problem 2.1 - part b
%-------------------------------------------

% (1) First fold the signal and then delay the result by
    4 samples
[x,n] = sigfold(xo,no);
[x,n] = sigshift(x,n,4); % delay/right shift by 4


subplot(3,1,2, 'align')
ph=stem(n,x);
grid on;
xlim([-10 10])
title('x(n) Folded, Then Delayed','FontSize',14,'
    FontWeight','Bold');
xlabel('Index(n)','FontSize',12,'FontWeight','Bold')
ylabel('x(n)','FontSize',12,'FontWeight','Bold')


% (2) First delay the signal by 4 samples and then fold

[x,n] = sigshift(xo,no,4); % delay/right shift by 4
[x,n] = sigfold(x,n);

subplot(3,1,3, 'align')
```

```
ph=stem(n,x);
grid on;
xlim([-10 10])
title('x(n) Delayed, Then Folded','FontSize',14,'
    FontWeight','Bold');
xlabel('Index(n)','FontSize',12,'FontWeight','Bold')
ylabel('x(n)','FontSize',12,'FontWeight','Bold')


%-----------------------------
% Problem 2.1 - part c
%-----------------------------
% plotting x(-n+4)

n = (-3:-1);
x = 1+n/3;
n = ((-3):(3));
x = xo;

[x,n] = sigfold(xo,n);
[x,n] = sigshift(x,n,4); % delay/right shift by 4


fh=figure(4);
subplot(2,1,1, 'align')
ph=stem(n,xo);

grid on;
xlim([-10 10])
title('Original Signal x(n)','FontSize',14,'FontWeight','
    Bold');
xlabel('Index(n)','FontSize',12,'FontWeight','Bold')
ylabel('x(n)','FontSize',12,'FontWeight','Bold')




subplot(2,1,2, 'align')
ph=stem(n,x);

grid on;
xlim([-10 10])
title('New Signal x(-n+4)','FontSize',14,'FontWeight','
    Bold');
xlabel('Index(n)','FontSize',12,'FontWeight','Bold')
ylabel('x(n)','FontSize',12,'FontWeight','Bold')
```

```
%%
%--------------------------------
% Problem 2.2 - part a
%--------------------------------
width = 10;
xo = [1,1,1,1,0.5,0.5];
no = (-1:4);

% [x,n] = sigfold(xo,n);
% [x,n] = sigshift(x,n,4); % delay/right shift by 4

% (a) - shift right/delay by 2
[x,n] = sigshift(xo,no,2);
fh=figure(5);
subplot(4,2,1, 'align')
ph=stem(n,x);
grid on;
xlim([-width width])
title('x(n-2)','FontSize',14,'FontWeight','Bold');
xlabel(' ','FontSize',12,'FontWeight','Bold')
ylabel('(a)      ','FontSize',12,'FontWeight','Bold','
    rotation',0)

% (b) - fold, and advance/shift left by 4
[x,n] = sigfold(xo,no);
[x,n] = sigshift(xo,no,-4);

subplot(4,2,2, 'align')
ph=stem(n,x);
grid on;
xlim([-width width])
title('x(4-n)','FontSize',14,'FontWeight','Bold');
xlabel(' ','FontSize',12,'FontWeight','Bold')
ylabel('(b)      ','FontSize',12,'FontWeight','Bold','
    rotation',0)

% (c) - advance/shift left by 2
[x,n] = sigshift(xo,no,-2);

subplot(4,2,3, 'align')
ph=stem(n,x);
grid on;
xlim([-width width])
title('x(n+2)','FontSize',14,'FontWeight','Bold');
xlabel(' ','FontSize',12,'FontWeight','Bold')
```

```
ylabel('(c)      ','FontSize',12,'FontWeight','Bold','rotation',0)

% (d) - multiply by step function that is folded and advanced by 2

[u,n] = stepseq(0,0,5);
[u,n] = sigfold(u,n);
[u,n] = sigshift(u,n,-2);
[x,n] = sigmult(xo,no,u,n);
subplot(4,2,4, 'align')
ph=stem(n,x);
grid on;
xlim([-width width])
title('x(n)u(2-n)','FontSize',14,'FontWeight','Bold');
xlabel(' ','FontSize',12,'FontWeight','Bold')
ylabel('(d)      ','FontSize',12,'FontWeight','Bold','rotation',0)

% (e) - delay x(n) by 1, multiply by delta delayed by 3
[d,n] = impseq(0,0,0);
[d,n] = sigshift(d,n,3);
[x,n] = sigmult(xo,no,d,n);

subplot(4,2,5, 'align')
ph=stem(n,x);
grid on;
xlim([-width width])
title('x(n-1)d(n-3)','FontSize',14,'FontWeight','Bold');
xlabel(' ','FontSize',12,'FontWeight','Bold')
ylabel('(e)      ','FontSize',12,'FontWeight','Bold','rotation',0)

% (f) - x(n^2)... nonlinear spacing
n = no.^2;

subplot(4,2,6, 'align')
ph=stem(n,xo);
grid on;
xlim([-width+5 width+10])
title('x(n^2)','FontSize',14,'FontWeight','Bold');
xlabel(' ','FontSize',12,'FontWeight','Bold')
ylabel('(f)      ','FontSize',12,'FontWeight','Bold','rotation',0)

% (g) - Even part
```

```
x = xo(2:2:end);
n = no(2:2:end);
subplot(4,2,7, 'align')
ph=stem(n,x);
grid on;
xlim([-width width])
title('even part of x(n)','FontSize',14,'FontWeight','
    Bold');
xlabel('Index(n)','FontSize',12,'FontWeight','Bold')
ylabel('(g)      ','FontSize',12,'FontWeight','Bold','
    rotation',0)

% (h) - Odd part
x = xo(1:2:end);
n = no(1:2:end);
subplot(4,2,8, 'align')
ph=stem(n,x);
grid on;
xlim([-width width])
title('odd part of x(n)','FontSize',14,'FontWeight','Bold
    ');
xlabel('Index(n)','FontSize',12,'FontWeight','Bold')
ylabel('(h)      ','FontSize',12,'FontWeight','Bold','
    rotation',0)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Problem 6 - Book Prob 2.17
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%--------------------------
% Signal Convolution
%--------------------------
% - pairs a
a_x = [1,1,1,1];
a_h = [6,5,4,3,2,1];
a_y = conv(a_x,a_h);
y_a = conv(a_h,a_x); % swapped h and x
a_n = (1:length(a_y));

% - pairs b
b_x = [1,1,1,1];
b_h = [6,5,4,3,2,1]; % advance by 3
b_y = conv(b_x,b_h);
y_b = conv(b_h,b_x);
b_no = (1:length(b_y));
```

```
[b_y,b_n] = sigshift(b_y,b_no,-3);
[y_b,] = sigshift(y_b,b_no,-3);

% - pairs c
c_x = [1,1,1,1]; % delay by 3
c_h = [1,1]; % advance by 4
c_y = conv(c_x,c_h);
y_c = conv(c_x,c_h); % swapped h and x
c_no = (1:length(c_y));
[c_y,c_n] = sigshift(c_y,c_no,-1); % delta n of 1
[y_c,] = sigshift(y_c,c_no,-1);

% - pairs d
d_x = [1,1,1,1]; % delay by 2
d_h = [1,1]; % advance by 2
d_y = conv(d_x,d_h);
y_d = conv(d_h,d_x);
d_n = (1:length(d_y));
% the delay and advance by 2 cancel



% ----------------------
% Convolution Plots
% ----------------------
width = 10;
subplot(4,2,1)
n = (1:length(a_y));
ph=stem(a_n,a_y);
grid on;
xlim([-width width])
title('Output A - x(n)*h(n)','FontSize',14,'FontWeight','Bold');
xlabel('Index(n)','FontSize',12,'FontWeight','Bold')
ylabel('y(n)      ','FontSize',12,'FontWeight','Bold','rotation',0)

subplot(4,2,2)
ph=stem(a_n,y_a);
grid on;
xlim([-width width])
title('Output A - h(n)*x(n)','FontSize',14,'FontWeight','Bold');
xlabel('Index(n)','FontSize',12,'FontWeight','Bold')
ylabel('y(n)      ','FontSize',12,'FontWeight','Bold','rotation',0)
```

```
subplot(4,2,3)
ph=stem(b_n,b_y);
grid on;
xlim([-width width])
title('Output B - x(n)*h(n)','FontSize',14,'FontWeight','Bold');
xlabel('Index(n)','FontSize',12,'FontWeight','Bold')
ylabel('y(n)      ','FontSize',12,'FontWeight','Bold','rotation',0)

subplot(4,2,4)
ph=stem(b_n,b_y);
grid on;
xlim([-width width])
title('Output B - h(n)*x(n)','FontSize',14,'FontWeight','Bold');
xlabel('Index(n)','FontSize',12,'FontWeight','Bold')
ylabel('y(n)      ','FontSize',12,'FontWeight','Bold','rotation',0)

subplot(4,2,5)
ph=stem(c_n,c_y);
grid on;
xlim([-width width])
title('Output C - x(n)*h(n)','FontSize',14,'FontWeight','Bold');
xlabel('Index(n)','FontSize',12,'FontWeight','Bold')
ylabel('y(n)      ','FontSize',12,'FontWeight','Bold','rotation',0)

subplot(4,2,6)
ph=stem(c_n,c_y);
grid on;
xlim([-width width])
title('Output C - h(n)*x(n)','FontSize',14,'FontWeight','Bold');
xlabel('Index(n)','FontSize',12,'FontWeight','Bold')
ylabel('y(n)      ','FontSize',12,'FontWeight','Bold','rotation',0)

subplot(4,2,7)
ph=stem(d_n,d_y);
grid on;
xlim([-width width])
title('Output D - x(n)*h(n)','FontSize',14,'FontWeight','
```

```
    Bold');
xlabel('Index(n)','FontSize',12,'FontWeight','Bold')
ylabel('y(n)       ','FontSize',12,'FontWeight','Bold','
    rotation',0)

subplot(4,2,8)
ph=stem(d_n,d_y);
grid on;
xlim([-width width])
title('Output D - h(n)*x(n)','FontSize',14,'FontWeight','
    Bold');
xlabel('Index(n)','FontSize',12,'FontWeight','Bold')
ylabel('y(n)       ','FontSize',12,'FontWeight','Bold','
    rotation',0)

%%%%%%%%%%%%%%%%%%%%%%
%% prob 7 Scratchwork
%%%%%%%%%%%%%%%%%%%%%%
h = [1,0.5,0.25,0.125,0.0625];
yo = [1,2,2,5,3,3,3,2,1];
x = [1,3/2,3/2,7/4,3/2,49/32,35/64,3/64,-57/128,3/64];

y = conv(h,x)
```