COMMUNICATION SYSTEMS MATLAB ASSIGNMENT NOTES

AKSHANSH CHAUDHARY

Communication Systems MATLAB Assignment Notes, First Edition

Presented by: Akshansh Chaudhary
Graduate of BITS Pilani, Dubai Campus
Batch of 2011

Course content by: Dr. Jagadish Nayak
Then Faculty, BITS Pilani, Dubai Campus

Layout design by: AC Creations © 2013

The course content was prepared during Fall, 2013.
More content available at: www.Akshansh.weebly.com

AKSHANSH CHAUDHARY
2011AAPS300U

Assignment 1: Getting Started With Matlab

1 Generating Array elements

a) Create an array of 3 elements having values 100 200 and 300. Name the array x.

Instruction: `X= [100 200 300]`

b) Append the same array with 3 other elements all having values 1.

Instruction: `X=[X ; 1 1 1]`

c) Pad the array with 4 zeros.

Instruction: `Z = Zeros(2); x = [X Z]`

d) Limit the number of elements in the array to 5 elements.

Instruction: for a matrix $x = \begin{bmatrix} 1 & 1 & 5 & 6 & 7 & 8 & 9 \\ 9 & 10 & 11 & 5 & 6 & 9 & 8 \end{bmatrix}$ &, `y = [x(2,1:5)]`

e) Display the 2nd and the 3rd element of the array at the command prompt.

Instruction: `x(1,2:3)` or `x(1,[2,3])`

f) Make a two dimensional array of size 3*3. Call it x3d.

Instruction: `x3d = [1 2 3 ; 4 5 6 ; 7 8 9]`

g) Display the 2nd row 3rd column element of the array at the command prompt

Instruction: `x3d(2,3)`

h) Find the transpose of this matrix and display both the 3*3 matrices. Call it x3dc.

Instruction: `x3dc = x3d'` OR `x3dc = transpose(x3d)`

i) Find the sum of the array elements in x (the one -dimensional array created before).

Instruction: `S = sum(x)`

What happens when I use the same command to the 3d array x3d.

If x3d is in a multidimensional array, sum(x3d) creates the value along the 1st non-singleton dimension as vectors, returning an array of row vectors.

j) Make an array containing 100 elements from 1 to 100 in steps of 1

Instruction: `y = [1 : 1 : 100]`

k) Make an array containing 100 elements from 1 to 500 in steps of 5

Instruction: `a = [1 : 5 : 500]`

l) Find the lengths of both the recently created arrays.

Instruction:

```
Size(y);
Size(z);
```

11 Getting used to functions to abs, sqrt and exp

a) Find the square root of a positive number and store it in a variable called xsqrt. Obtain the positive number from the user.

Instruction: `x = input("Enter a positive number")`
`xsqrt = sqrt(x)`

Result: 4.7958 (input = 23)

b) Find the square root of a negative number and store it in a variable called xsqrt1. Obtain the number again from the user.

Instruction: `l = input("Enter a negative number")`
`xsqrt1 = sqrt(l)`

Result: 0 + 2.0000i (input = -4)

c) Note the difference in the values stored in xsqrt and xsqrt1.

For xsqrt : only real part is shown
For xsqrt1 : Both real & imaginary parts are shown

d) Find the abs value of xsqrt1 and store that in the same variable

Instruction: `xsqrt1 = abs(xsqrt1)`

Result: 2

e) Store the velocity of light in a variable called c. (Using exp or e)

Instruction: `c = 3e8`

f) Obtain 10 random elements using rand function

Instruction: `rand(5,2)`

(generates a matrix of size 5×2 with 10 random elements)

(III)

(d) As per MI of part (a)

```
plot (x, sin(x))
hold
plot(x, sin(2*x))
plot (x, sin(3*x))
```

As per M2 of part(a)

```
a = sin(2*2*pi*f*t);
b = sin(3*2*pi*f*t);
plot (t, x)
hold
plot(t, a, 'g')
plot (t, b, 'r')
```

(e) As per M1 of part(d)

```
subplot(3, 1, 1)
plot (x, sin(x))
hold
Subplot (3, 1, 2)
plot (x, sin(2*x))
hold
subplot (3, 1, 3)
plot (x, sin(3*x))
```

As per M2 of part(d)

```
Subplot (3, 1, 1)
Plot (t, x)
Subplot (3, 1, 2)
Plot (t, a)
Subplot (3, 1, 3)
Plot (t, 3)
```

III Create a simple sine function and Plot it

a) Create a sine wave and assign it to a variable and plot it. Obtain the frequency of the sine wave for this question from the user. M2

Instructions:

M1

```
x = 0:0.01:2*pi
y = sin x
plot(x,y)
```

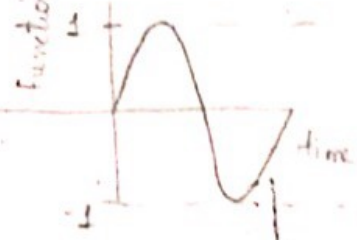
```
x = input("Enter frequency of sine wave")
fs = 10000;
dt = 1/fs ; t = [0:dt:0.10-dt];
x = sin(2*pi*f*t);
plot(t,x);
hold;
```

→ ~~x-label~~ title('signal vs time');

b) Label your plot and label the axes. Sketch the result obtained on screen here.

Instructions:

```
xlabel('time (in seconds)')
ylabel('signal , y = sin(2πft)
```



c) Obtain grids on your plot.

Instructions: grid on

d) Obtain multiple sine plots of different frequency on the same plot.

Instruction: (On adjacent page)

e) Obtain these multiple plots on different subplots in the same figure.

Instruction: (on adjacent page)

f) What is the maximum frequency of the sine wave that you can plot for a given number of sample/period? If T_s is the sampling period.

$\Rightarrow F_s$ is the sampling frequency ($F_s = \frac{1}{T_s}$)

According to Nyquist sampling theorem,

$F_s \geq F$: sine wave frequency.

$\therefore F_{max}$ of sine wave $= F_s$.

V To get used to control functions

a) Generate a random array of 100 elements. Call it x. Also generate another 100-element array containing values from 1 to 100 in steps of 1 and call it x1.

Instructions:
```
x = rand(1,100)
x1 = [1:1:100]
```

b) If the 50th or the 100th element of x is odd, then make parity=1

Instructions:
```
k = [x(50), x(100)];
remain = rem(k,2);
if remain == 0
parity = 0
else
parity = 1
end;
```

c) Set a counter that counts the number of odd elements in x and x1.

Instructions:
```
z = [x ; x1]
c = 0;
for i = 1:2
for j = 1:100
if rem(z(i,j),2) not 0
c = c+1
end
end
end
```

d) Check how many elements in x and x1 are multiples of 6

Instructions:
```
c = 0;
for i = 1:2
for j = 1:100
if rem(z(i,j),6) == 0
c = c+1
end
end
end
```

e) Generate a random sequence of binary digits using rand function and applying 0.5 as threshold value.

Instructions:
```
seq = randi([0,1],4)
```

V To Create functions and call and execute them

a) Create a function that counts the number of multiples of 3 in an array. Call it mult3.

Instructions:

```
function c = mult3(num)
i = length(num);
c = 0
for k = 1 : i;
if (rem(num(k),3)==0
    c = c+1
end
end
end
```

b) Call this function from a main function

Instructions:

```
a = [1 3 6 9 10]
x = mult3(a)
```

c) Pass x and x1 arrays created in the last problem to the function from main.

Instructions:

```
f = mult3(x);
g = mult3(x1);
```

Signature of the instructor: _____

Date: 26|9|2013

AKSHANSHI CHAUDHARY, 2011AAPS300U

Assignment 2: Amplitude Modulation

Write a Matlab code that would perform Amplitude modulation of a carrier signal based on a message signal. The modulation results have to be observed in time and frequency domain. Follow the instructions given below.

1. Obtain the sampling frequency, carrier frequency and message signal frequency from the user.
   Conditions: Take i) fc >10fm and ii) fs > 2* fc.
2. Also obtain the amplitude of the message signal and the carrier signal from the user.
3. Allow the user to choose one of the following cases of modulation.
   a. Normal AM with transmitted carrier (Ordinary AM)
   b. DSB-SC
   c. Single sideband suppressed carrier.
4. Compare the time domain and frequency domain waveforms of the three types of modulation.
5. Change the sampling frequency, message signal frequency and carrier frequency and see what happens.

Example of command line inputs

(AM Transmitted Carrier)

>> am2

Enter the type of AM: 0

Enter the carrier frequency: 1000

Enter the frequency of the modulating signal: 100

Enter the sampling frequency of the signal: 10000

Enter the amplitude of modulating signal: 1

Enter the amplitude of carrier signal: 1

List down a set of chosen values:

The Carrier frequency: 100 Hz The Frequency of the modulating signal: 5 Hz

The Sampling frequency of the signal: 1000 Hz

The Amplitude of modulating signal: 2 The Amplitude of carrier signal: 4

(1,2) Taking user input for f_m, f_s, f_c, A_m and A_c

```
values = inputdlg({'Sampling frequency (fs)', 'Carrier frequency
         (fc)', 'Message signal frequency (fm)', 'Amplitude
         of message signal (Am)', 'Amplitude of
         carrier signal (Ac)'}, 'Enter values');
```

```
input('Note : Enter values such that fc > 10fm and
       fs > 2*fc');
```

```
fs = str2num(values{1}); fc = str2num(values{2});
fm = str2num(values{3}); Am = str2num(values{4});
Ac = str2num(values{5});
```

(2). Making user choose type of modulation

```
type = menu('Choose type of modulation', 'Normal AM
       with Transmitted carrier (Ordinary AM)', 'DSB-SC',
       'SSB-SC');
```

% shows and assigns a value to "type" based on what user chooses.

- Creating modulating signal and giving conditions.

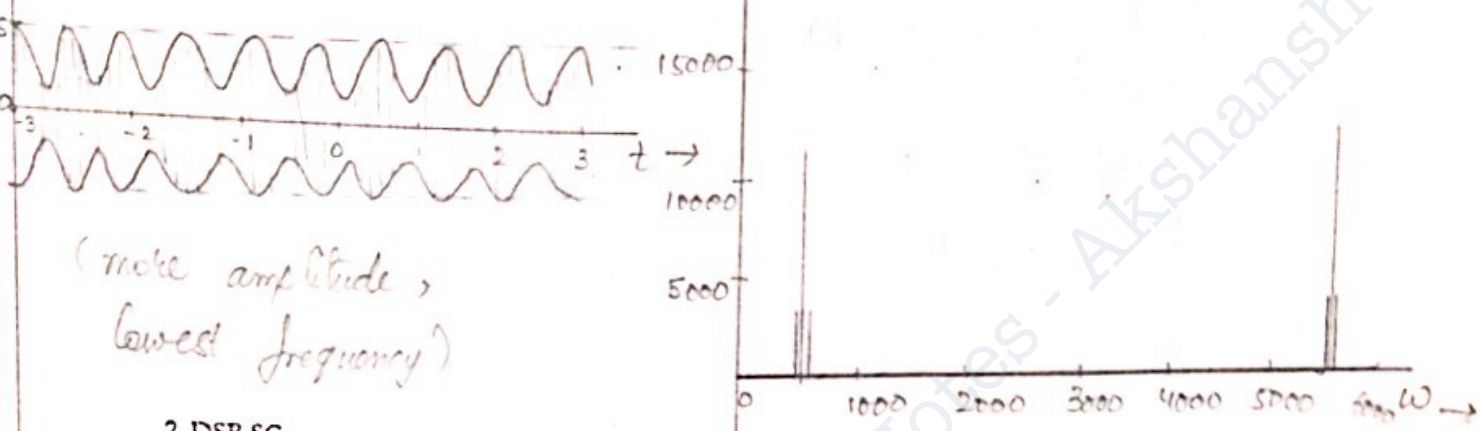
```
t = [-3 : 1/fm : 3]
m = Am * cos(2*pi*fm*t)
```

continued.

Sketch the Modulated signal in time domain and the spectrum of the modulated signal

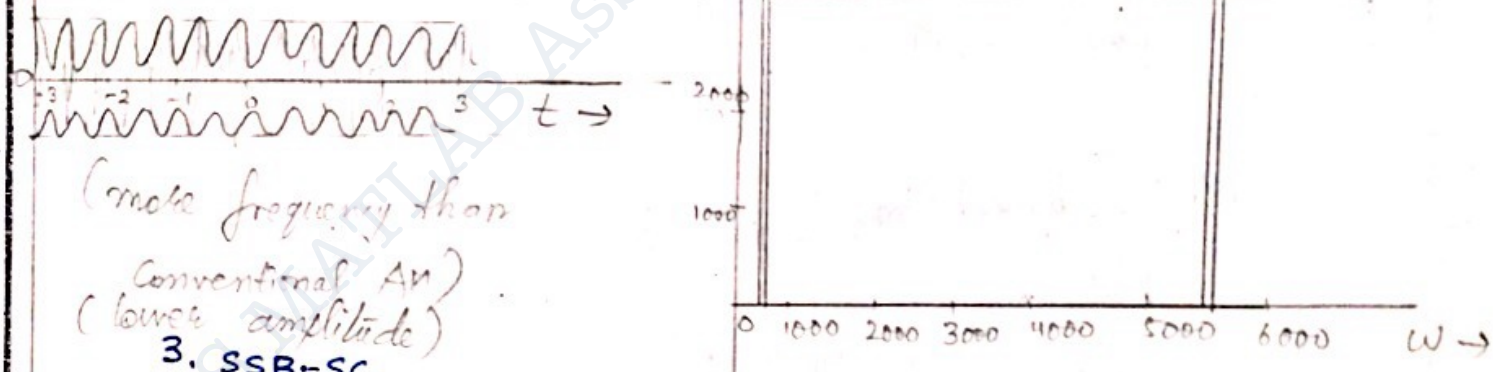
- The figures appearing on screen should be copied on to the workspace provided below.
- Remember to label all the figures

1. AM Transmitted Carrier



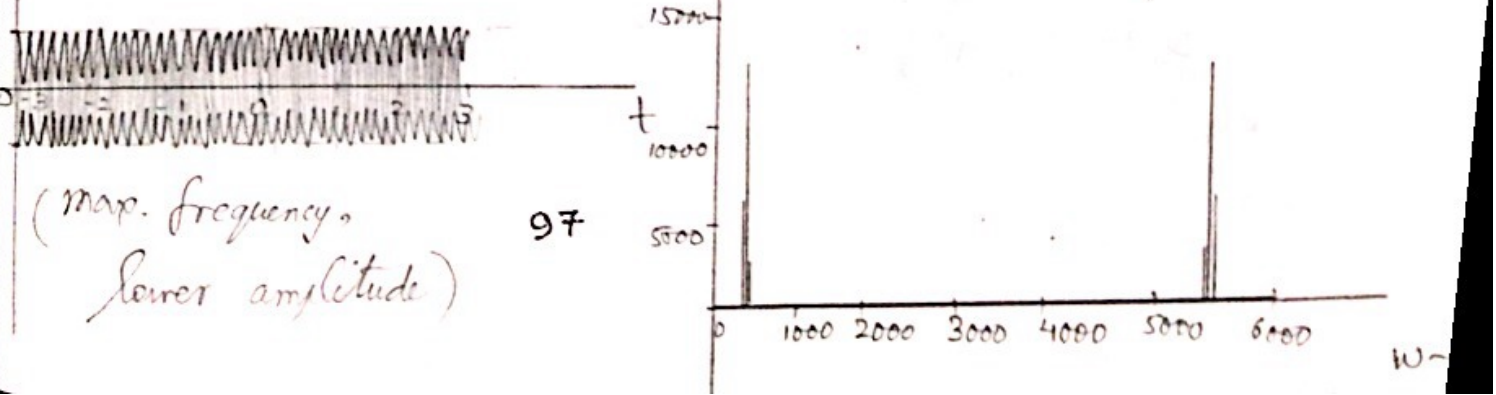
(more amplitude, lowest frequency)

2. DSB SC.



(more frequency than Conventional AM)
(lower amplitude)

3. SSB-SC



(max. frequency, lower amplitude)

97

(3). Giving conditions.

- continued

```
>> if type == 1
    M = ammod(m, fc, fs, 0, Ac);
    subplot(2,1,1)
    plot(abs(fft(M))); % frequency domain waveform
    hold
    subplot(2,1,2)
    plot(t, M); % time domain waveform.
  else if type == 2
    M = ammod(m, fc, fs);
    subplot(2,1,1)
    plot(abs(fft(M)));
    hold
    subplot(2,1,2)
    plot(t, M);
  else
    M = ssbmod(m, fc, fs);
    subplot(2,1,1)
    plot(abs(fft(M)));
    hold
    subplot(2,1,2)
    plot(t, M);
  end
end.
```

Instruction set 4

% everytime the user chooses the type of modulation, type "Instruction set 4" to automatically take the condition and plot the graphs.

% for choosing the type of modulation, type instruction 2. followed

Answer the following questions based on the figures obtained.

1. What is the transmission bandwidth of the modulated waveforms for each of the modulation schemes?

for DSB-SC and Conventional AM, BW = 2 f_m
for SSB-SC, BW = f_m

2. What waveform do you expect in time domain for AM if the modulating signal amplitude is greater than the carrier amplitude? What do you call such a modulation case as?

We get a distorted waveform. Its called o-modula

3. What is the effect of increasing or decreasing sampling frequency with respect to the carrier frequency?

Increasing f_s gives a demodulated waveform close to m(t)

4. Explore the MATLAB inbuilt commands for modulation and demodulation in the communication toolbox.

ammod; amdemod; ssbmod; ssbdemod;

[illegible] of the instructor.

[illegible]

20.10.2013

Assignment 3: FM and Tool box functions

1. Frequency Modulation

Write a Matlab code that would perform Frequency modulation of a carrier signal based on a message signal. The modulation results have to be observed in time and frequency domain. Follow the instructions given below.

a. Obtain the amplitude and frequency of the message signal and that of the carrier signal from the user. Also obtain the sampling frequency from the user and check for conditions.

b. Obtain the frequency-modulated response in time domain and frequency domain and interpret that for different modulation indices.

Example of command line inputs

```
>> fm_code

Enter the carrier frequency: 10
Enter the modulating frequency: 1
Enter the sampling frequency of the signal: 1000
Enter the carrier amplitude: 1
Enter the value for modulation index: 100
```

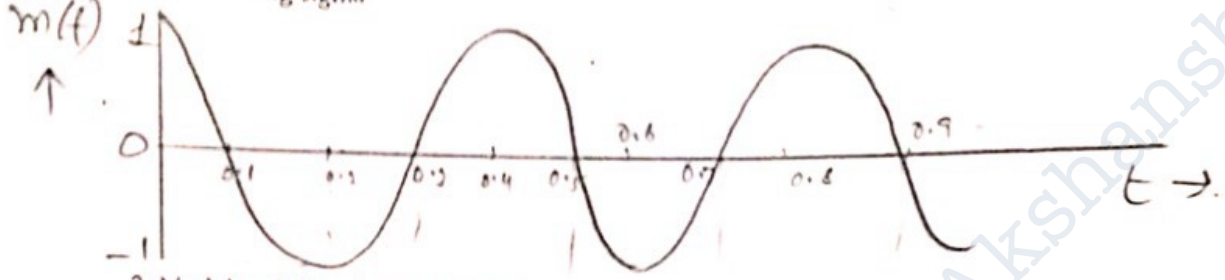
List down a set of chosen values:

The Carrier frequency: 400 The Frequency of the modulating signal: 25
The Sampling frequency of the signal: 1000
The Amplitude of carrier signal: 1 Value of the modulation index: 10

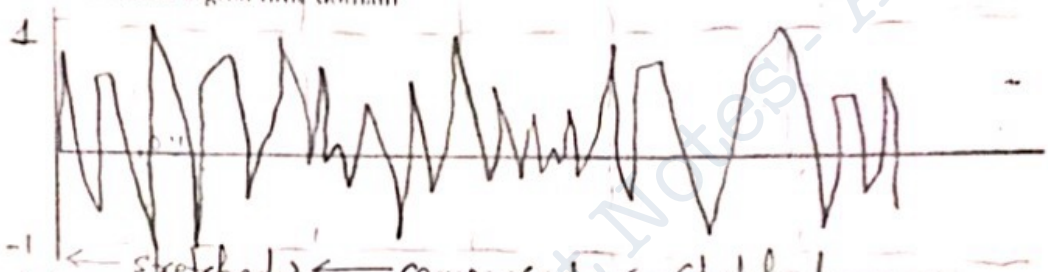
Sketch the Modulated signal in time domain and the spectrum of the modulated signal

- The figures appearing on screen should be copied on to the workspace provided below.
- Remember to label all the figures

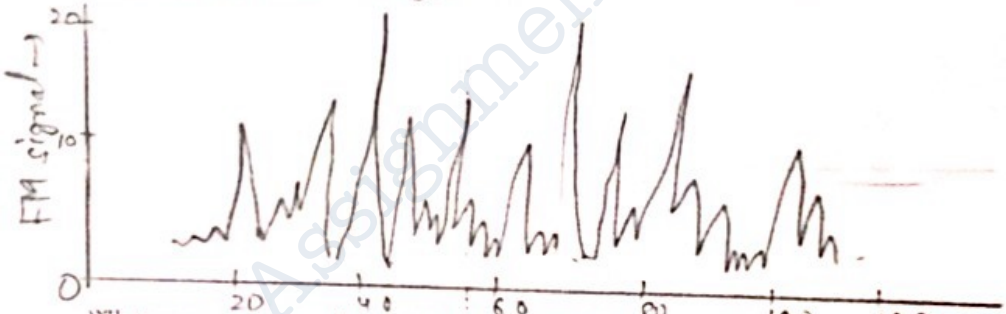
1. Modulating signal



2. Modulated Signal time domain



3. Spectrum of the modulated signal



What can you say about NBFM and WBFM from the spectrums obtained for different modulation index

When modulation index is changed (i.e, value of β) correspondingly, we get NBFM & WBFM.
Seeing the spectrum, for NBFM (taken at 0.1), the no. of spectral lines are less.
whereas, for WBFM (taking β = 1000), we see more no. of spectral lines.

ASSIGNMENT 3.

CS - MATLAB

Akshansh
2011AAPS300U

Q. 1)

```
>> values = inputdlg({'Message Amplitude, Am','Message frequency, fm','Carrier amplitude, Ac','Carrier frequency, fc','Sampling frequency, fs','Modulation index, mi'}, 'Enter values');
>> Am = str2num(values{1}); fm = str2num(values{2}); Ac = str2num(values{3}); fc = str2num(values{4}); fs = str2num(values{5}); mi = str2num(values{6});
>> t = 0:0.001:0.1;
>> m = Am * cos(2*pi*fm*t);
>> c = Ac* cos(2*pi*fc*t);
>> subplot(3,1,1);
>> plot(t,m);
>> xlabel('time');
>> ylabel('amplitude');
>> title('Message signal');
>> grid on;
>> subplot(3,1,2);
>> plot(t,c)
>> xlabel('time');
>> ylabel('amplitude');
>> title('carrier signal');
>> grid on;
>> Sfm = cos(2*pi*fc*t + (mi*cos(2*pi*fm*t)))
>> subplot(3,1,3);
>> plot(t, Sfm); % Plotting modulated signal
>> xlabel('time');
>> ylabel('amplitude');
>> grid on;
```

```
>> plot(abs(fft(m)));
>> plot(abs(fft(c)));
>> plot(abs(fft(sfm)));   % Plotting spectrum
```

% Note: Values entered, $f_m = 25$, $f_c = 400$, $m_i = 10$

% Changing values of modulation index :-

For modulated signal & spectrum: Decreasing mod m_i increases ideality. The spectrum nearly has only 2 bands at $m_i = 0.1$, whereas, it has a range of many bands at higher values of m_i.

2. Toolbox functions

This is to help you learn about the modulation related function in the communication toolbox.

Explore the functions amod, amodce, ademodce, ademod.

The commands were studied and analyzed.

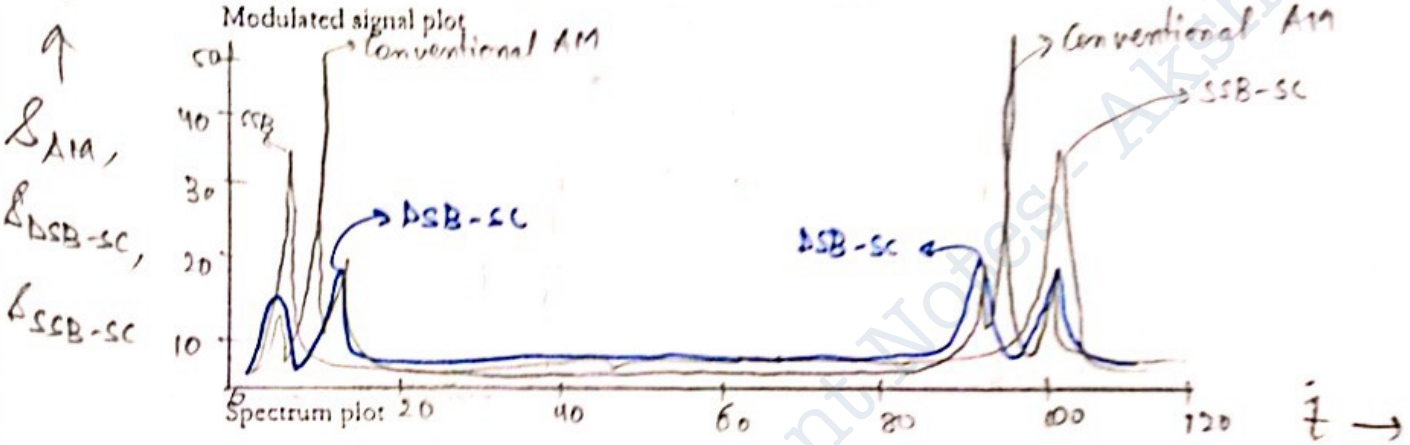
3. Amplitude Modulation using toolbox functions

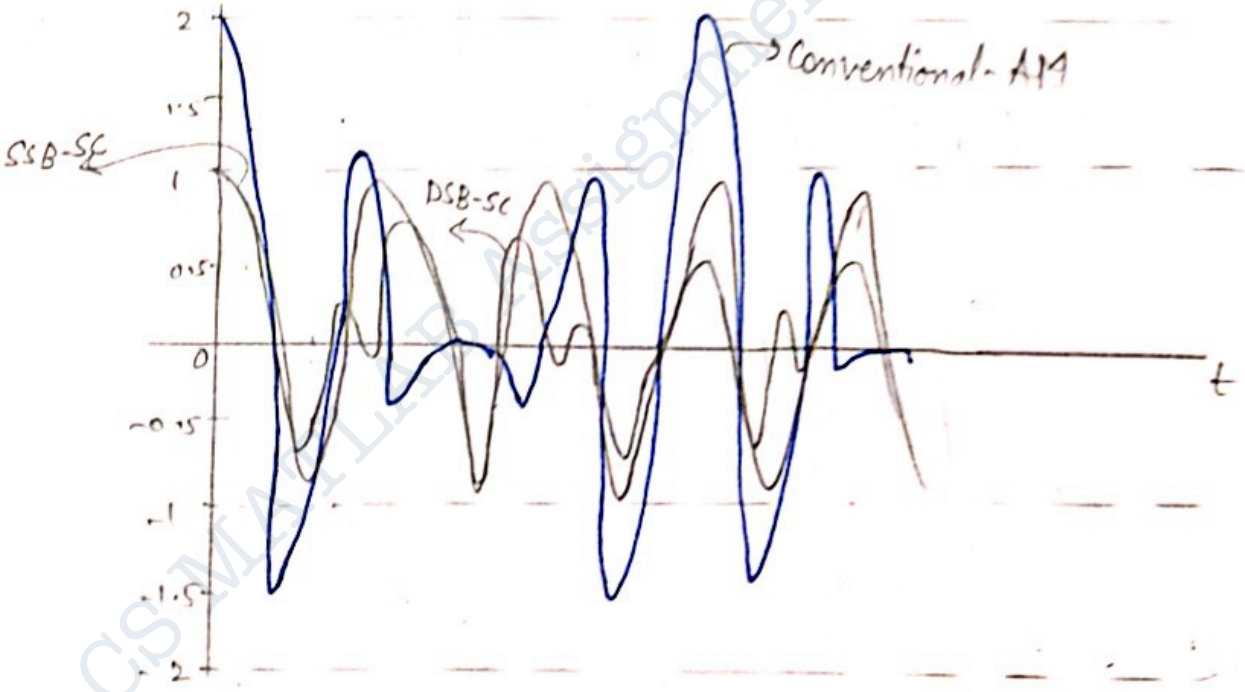
Obtain the amplitude-modulated responses in time domain for a given signal and carrier frequency using toolbox functions. Given, Fs=100,Fc=10 and Fm=1.

Obtain the AM TC, DSB-SC and SSB-SC responses. Obtain all the three modulated signals in a single plot and all the spectrums in another plot.

Sketch the modulated signal and its spectrum as it appears on the screen.

Modulated signal plot





Question 3)

```
>> fs = 100; fc = 10; fm = 1
>> Sdsb = ammod(m, fc, fs); % Plot of DSB spectrum
>> plot(abs(fft(Sdsb)));
>> hold on;
>> Sam = ammod(m, fc, fs, 0, Ac); % Plot of conventional AM spectrum
>> plot(abs(fft(Sam)), 'red');
>> hold on;
>> Sssb = ssbmod(m, fc, fs);
>> plot(abs(fft(Sssb)), 'green'); % Plot of SSB-SC spectrum.
% Plotting modulating signals
>> t = [0 : 1/fm : 100]
>> plot(t, Sdsb)
>> hold on;
>> plot(t, Sam, 'red');
>> hold on;
>> plot(t, Sssb, 'green');
```

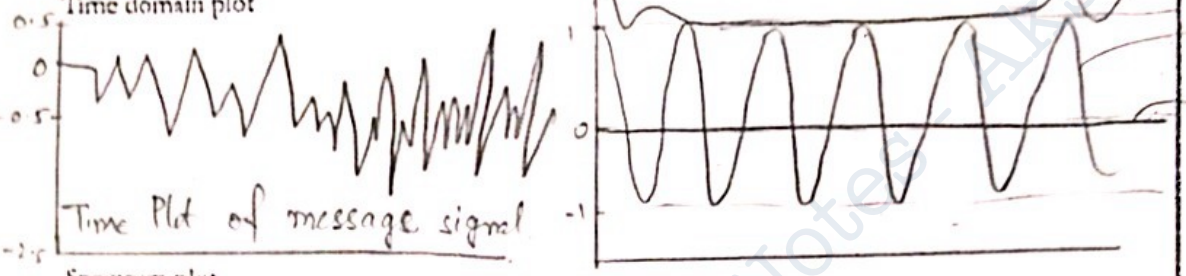
4. Modulated signal

Add two sinusoidal signals of frequency 1 and 2 Hz and the using that as the signal perform base-band modulation and demodulation using toolbox functions. Take Fs=100.

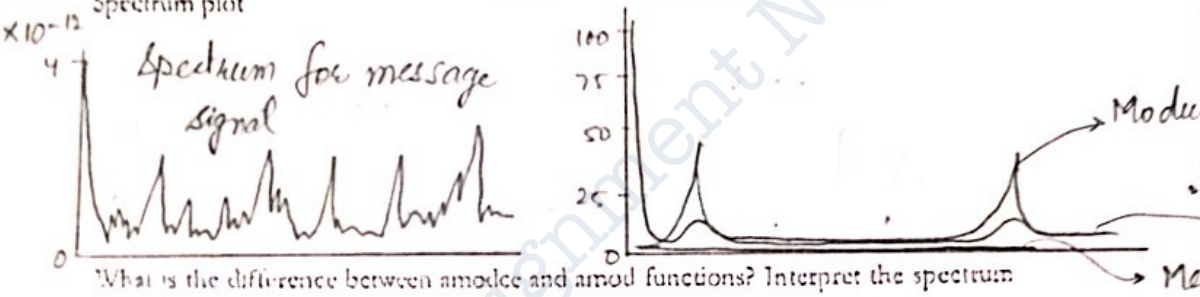
Obtain the message signal, baseband modulated signal and demosulated signal in one single plot and all spectrums in another plot.

Sketch the modulated signal and its spectrum as it appears on the screen.

Time domain plot



Spectrum plot



What is the difference between amodce and amod functions? Interpret the spectrum obtained using these functions.

On the attached sheets.

Try the demodulator, which uses the Costas loop option. Note the change in the demodulated response

Question - 4)

```
% Making the plot assuming conventional AM
>> m1 = sin(2*pi*1*t); m2 = sin(2*pi*2*t);
>> m = m1 + m2;
>> s = ammod(m, fc, fs, 0, Ac);
>> sd = amdemod(s, fc, fs);
% Signal plotting
>> plot(t, m);
>> hold on;
>> plot(t, s, 'red');
>> hold on;
>> plot(t, sd, 'magenta');
% Spectrum plotting
>> plot(abs(fft(m)));
>> hold on;
>> plot(abs(fft(s)), 'red');
>> hold on;
>> plot(abs(fft(sd)), 'magenta');
```

% Difference between amod and amodce commands :

~~>> clc~~

% Function amod performs analog passband modulation whereas
% function amodce performs analog baseband modulation.

5. Frequency modulation using toolbox functions

Obtain the amplitude and frequency of the message signal and that of the carrier signal from the user. Also obtain the sampling frequency from the user and check for conditions.

Obtain the frequency modulated response in time domain and frequency domain using tool box function and interpret the results for different modulation indices.

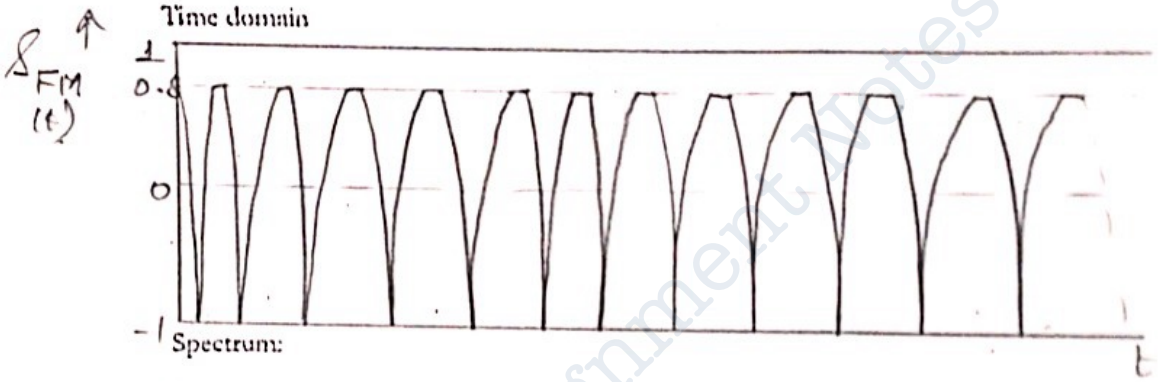
Chosen values:

$A_m = 1.$, $f_m = 25$, $A_c = 1$, $f_c = 400$, $f_s = 1000$, $m_i = 10$

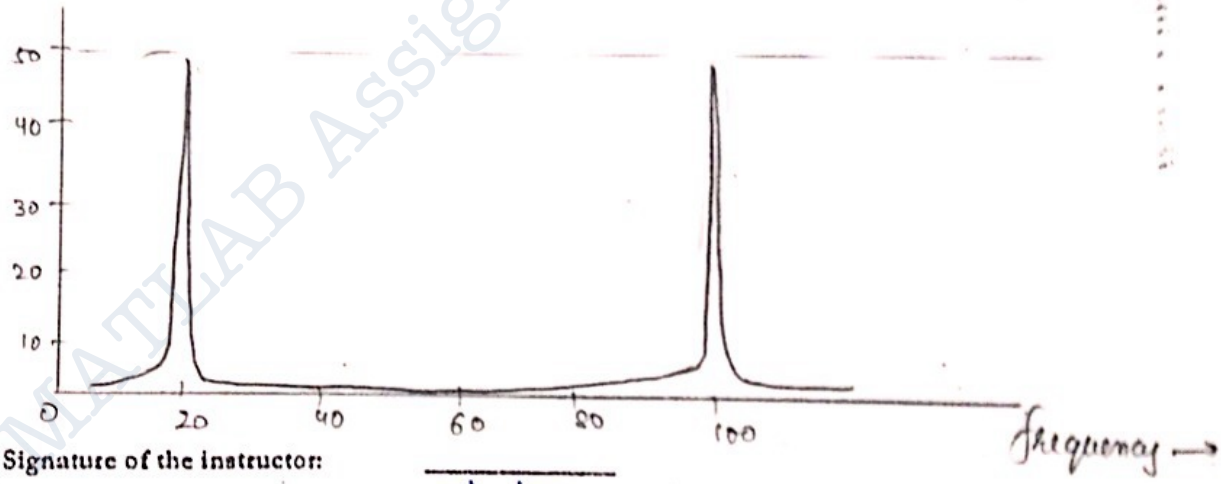
(values are same as in question 1)

Plots:

Time domain



Spectrum:



Signature of the instructor:

Date: 7/11/13

Question - 5)

```
% Taking values from user
>> data = inputdlg ({'Message Amplitude, Am','Message frequency, fm',
   'Carries amplitude, Ac','Carries frequency, fc','Sampling
   frequency, fs','Modulation index, mi'},'Enter values');
>> Am = str2num(values{1}); fm = str2num(values{2}); Ac = str2num
   (values{3}); fc = str2num(values{4}); fs = str2num(values{5});
   mi = str2num(values{6});
% Assuming max. freq. deviation (± kf Am) = 10.
>> ml = Am* cos(2*pi*fm*t);
% Modulated signal plotting
>> freq = fmmod(ml, fc, fs, 10);
>> plot(t, freq)
% The graph expands when max. freq. deviation is increased.
% Spectrum plot.
>> plot(abs(fft(freq)));
```

Akshansh
2011AAPS300U

Assignment 4: Line codes

This assignment is to generate the various line codes and view their power spectral density

The function 'linecodes' should take the bit period; sampling frequency and the bit sequence as input from the user and provide the user with a choice to choose from the six linecodes.

The function 'linecodes' should use the user-defined functions mentioned below to perform the task.

1. codengen is the actual function that generates the particular line code and its plot in time domain. codegen will be called by linecodes.
2. psp is the function to plot the power spectral density of a particular waveform and can be called by codegen.

Example od command line inputs

```
>>linecodes
input the binary sequence within square braces
[1 1 0 1 0 1]
The time period of signalling is
1e-3
 Let the sampling frequency be atleast 10 times that of 1/Tb
The sampling frequency of the signal is
10000
    enter the code that you want to analyse
    enter 1 for Unipolar NRZ code
    enter 2 for Unipolar RZ code
    enter 3 for Polar NRZ code
    enter 4 for polar RZ code
    enter 5 for AMI code
    enter 6 for Manchester code
    enter 0 to exit
6
    enter the code that you want to analyse
    enter 1 for Unipolar NRZ code
```

enter 2 for Unipolar RZ code
enter 3 for Polar NRZ code
enter 4 for polar RZ code
enter 5 for AMI code
enter 6 for Manchester code
enter 0 to exit

0

Thank you

>>

Chosen values:

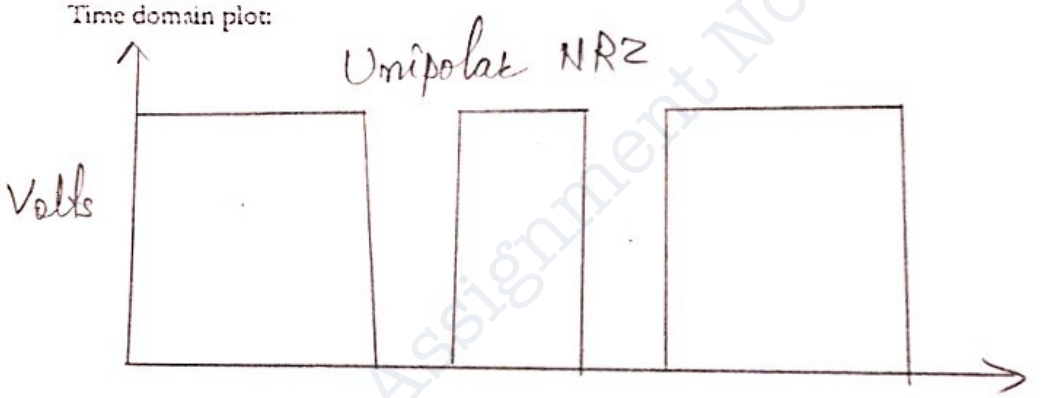
Time period of the signal:

Chosen Sampling frequency:

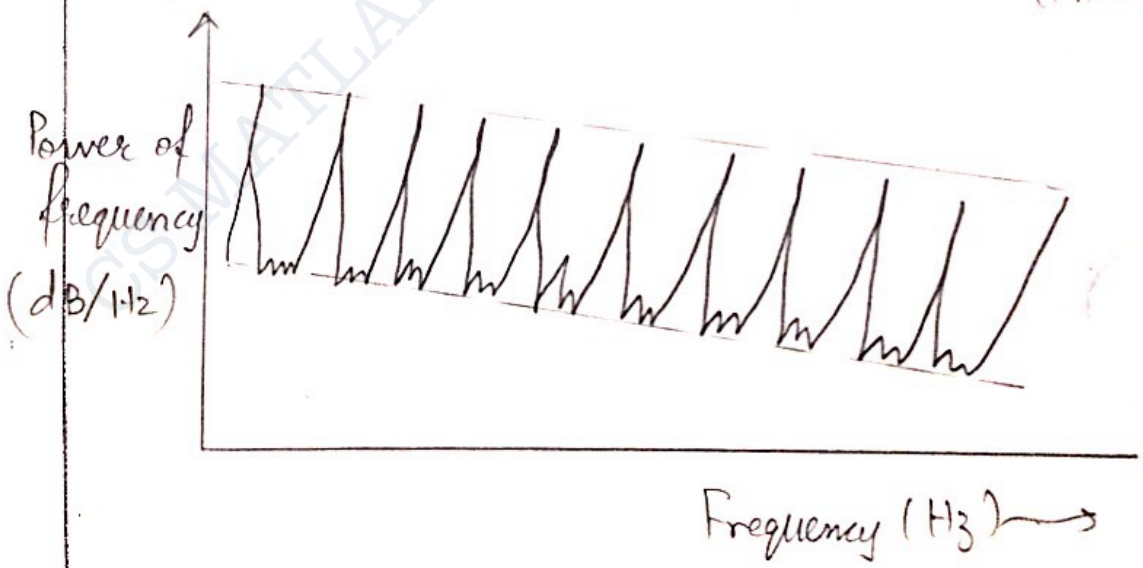
Plots:

Type:

Time domain plot:

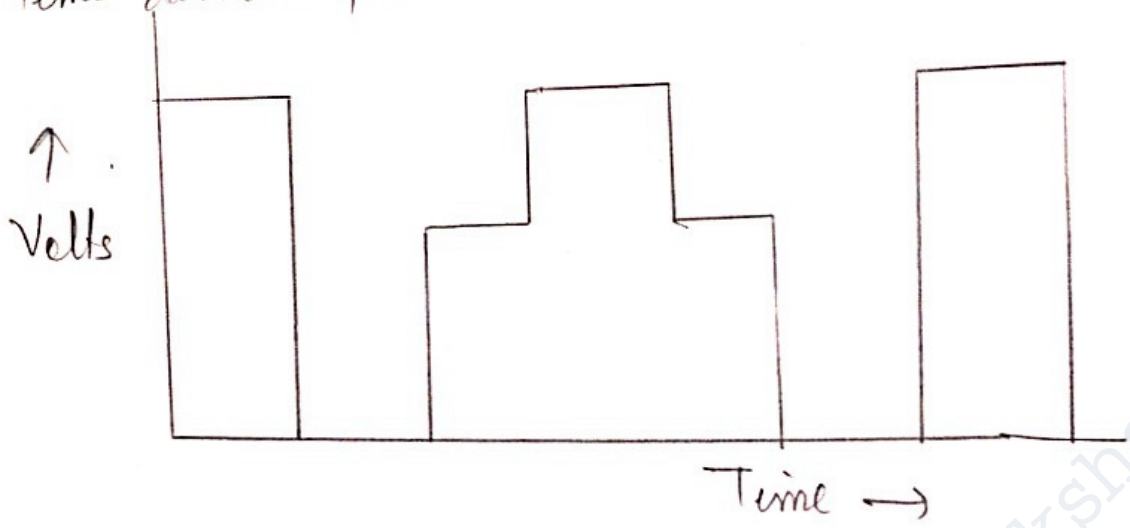


Spectrum:

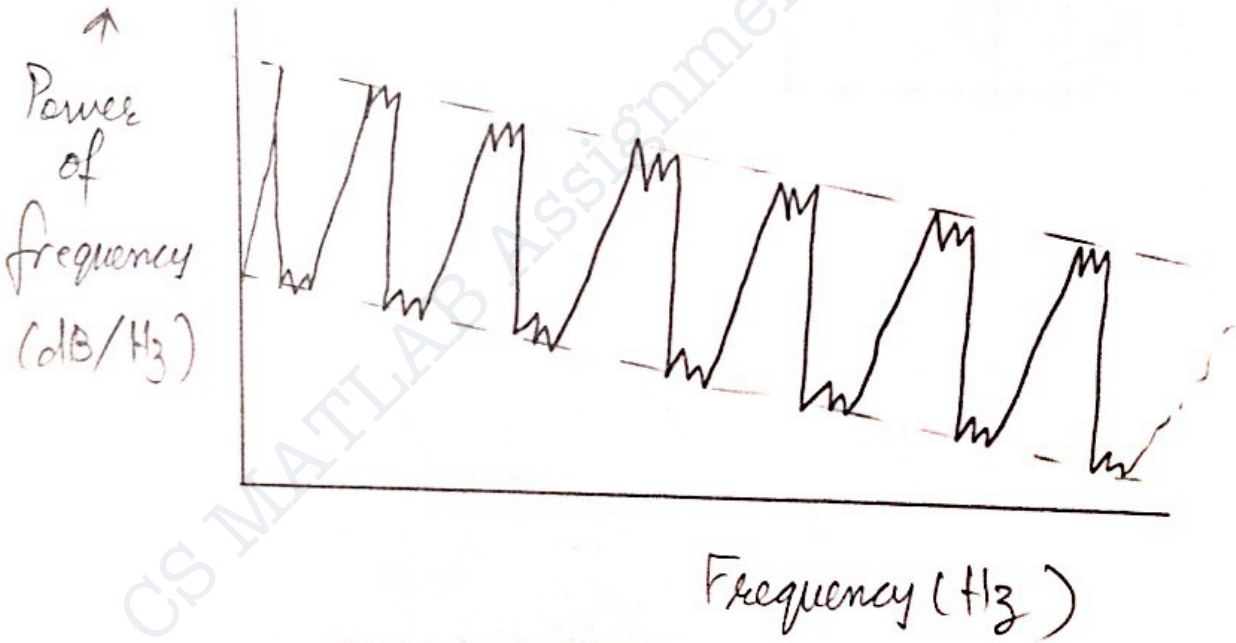


Type :- AMI

Time domain plot

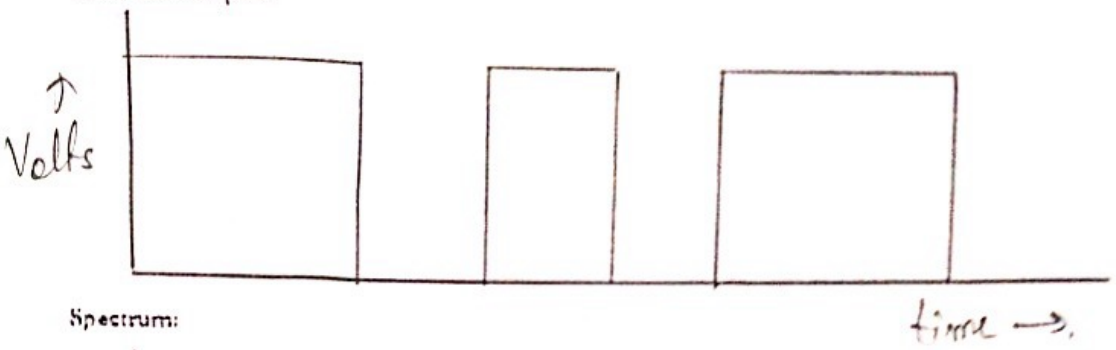


Spectrum :



Type: Polar NRZ

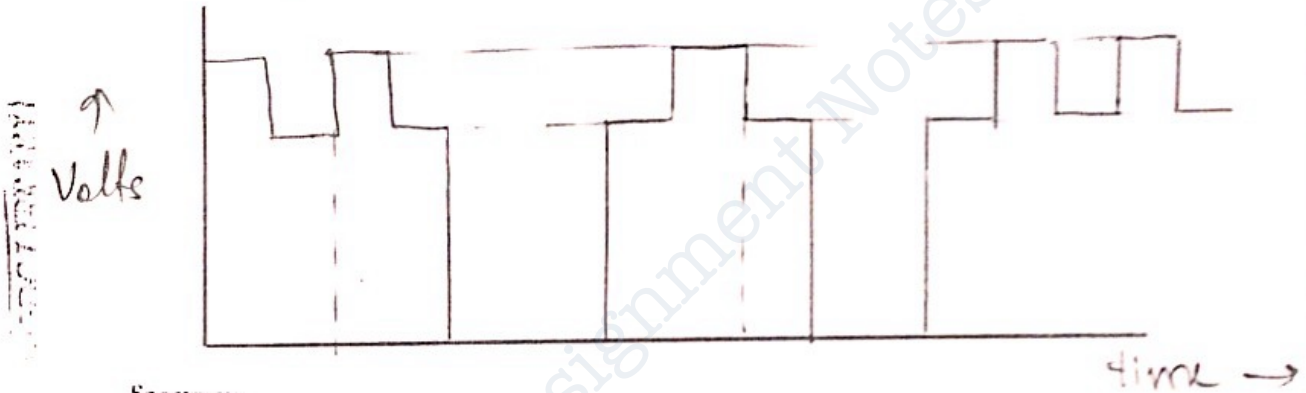
Time domain plot:



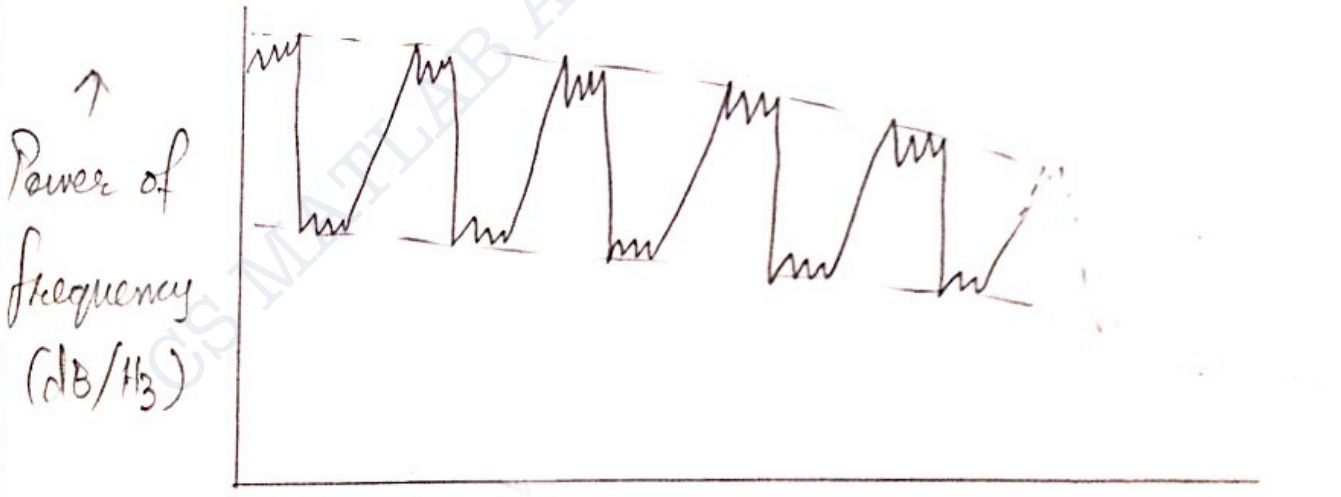
Spectrum:

Type: POLAR RZ

Time domain plot:

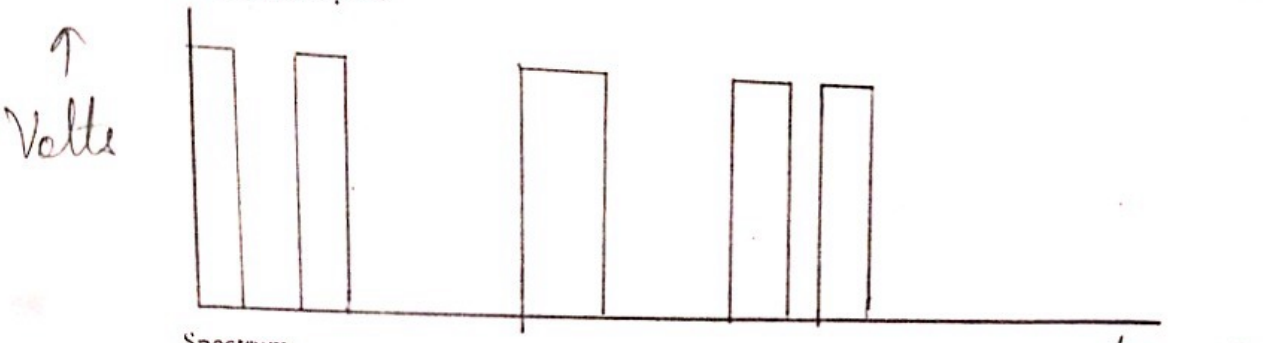


Spectrum:

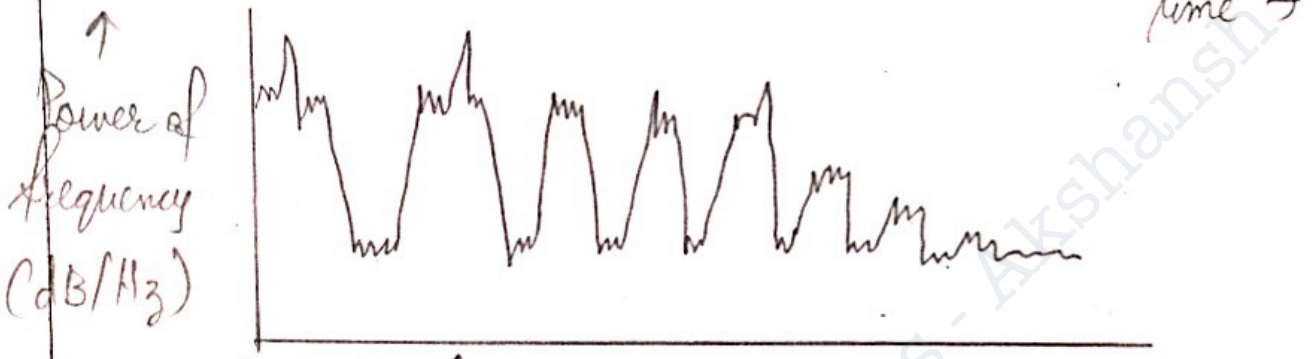


Type: UNIPOLAR RZ

Time domain plot:

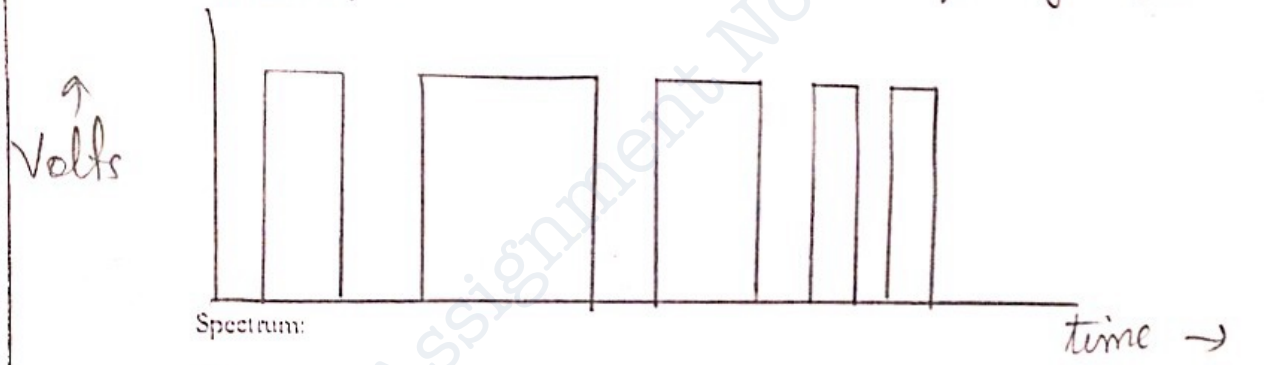


Spectrum:

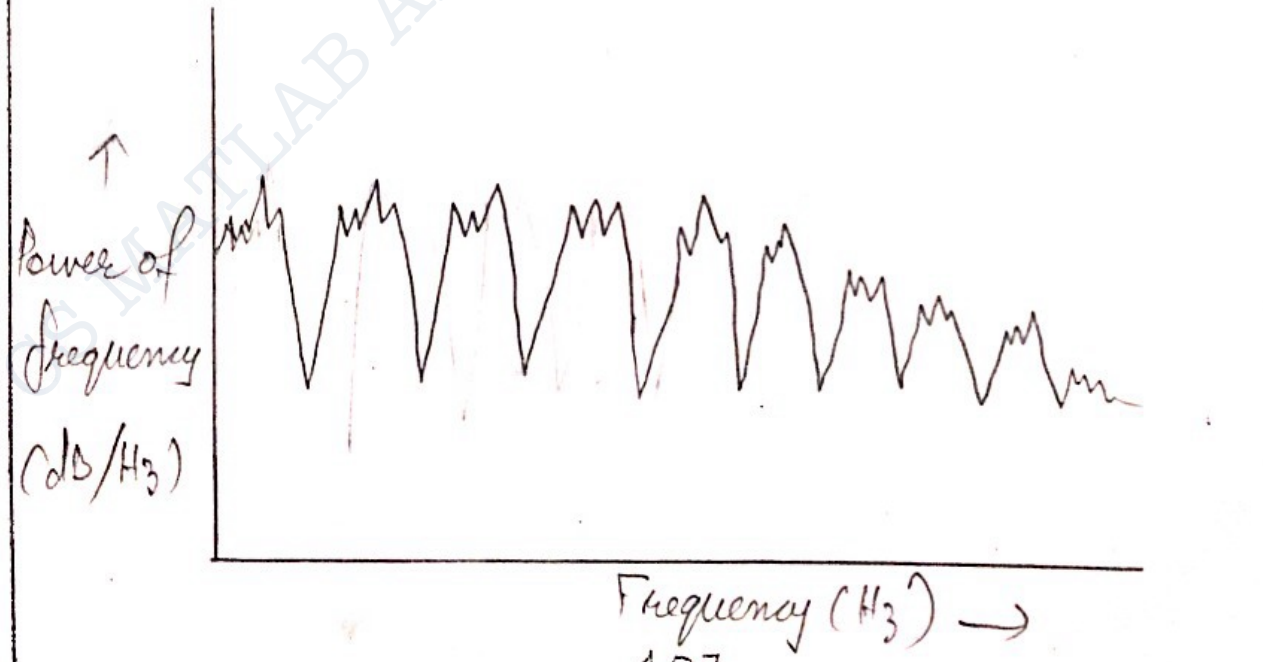


Type: Manchestar

Time domain plot:



Spectrum:



After generating all the six linecodes, analyse the time domain waveforms and the spectrums obtained.

Questions:

1. Which of these codes suppresses the DC component?

1. Manchester 2. AMI.
RZ has much less DC component than NRZ.

2. What is the disadvantage of Polar NRZ code?

It lacks clock information especially when a long sequence of 0's and 1's are transmitted.

3. Where is the first zero crossing of the psd in the Manchester code?

—

CS MATLAB ASSIGNMENT - 4 Akshansh Chaudhary


```
% Main Code of function --->(A)
function linecodes(~)
x = input('Enter binary sequence within brackets :');
Tb = input('Time period of signaling is :');
disp('Let sampling frequency be atleast 10 times that
of 1/Tb');
Fs = input('The sampling frequency of signal is :');
Ts = 1/Fs;
disp('Enter the code you want to analyse :');
disp('1. Unipolar NRZ');
disp('2. Bipolar NRZ');
disp('3. Unipolar RZ');
disp('4. Bipolar RZ');
disp('5. AMI');
disp('6. Manchester code');
disp('7. Exit');
c = input('Enter the code you want to analyse :');
y = code(c, Tb, x, Ts); % Jumps to (B) --> first func
psp(y, Fs);
end
```

% Function - 1 : Unipolar NRZ

```
function [x] = unrz(bitperiod, bits, samperiod)
T = length(bits)* bitperiod; % Total time for sequence.
t = 0 : samperiod : T- samperiod;
n = bitperiod / samperiod;
x = zeros(1, length(t)) ; % Initializing o/p variable.
for i=0 : length(bits) -1
        if bits(i+1) == 1
           x(1, i*n+1 : (i+1)*n) = 1;
        else x(1, i*n+1 : (i+1)*n) = 0;
        end
end
subplot (2, 1, 1)
plot (t, x)
hold
return % Third jump back to (B)
end
```

— x —

% Function 2 - Polar NRZ

```
function [x] = pnrz(bitperiod, bits, samperiod)
T = length(bits)* bit period;
n = bitperiod / samperiod;
t = [0 : samperiod : T- samperiod];
x = zeros(1, length(t))
```

```
for i=0 : length(bits) - 1
    if bits(i+1) == 1;
        x(1, i*n+1 : (i+1)*n) = 1;
    else x(1, i*n+1 : (i+1)*n) = -1;
    end
end
subplot (2, 1, 1)
plot (t, x)
hold
return % Third jump back to (B)
end.
```

— x —

```
% Function 3 : Unipolar RZ

function [x] = urz (bitperiod, bits, samperiod)
T = length(bits)* bitperiod;
t = [0 : samperiod : T - samperiod];
n = bitperiod/samperiod;
x = zeros(1, length(t));
  for i=0 : length(bits) - 1
      if bits(i+1) == 1
          x(1, i*n+1 : (i+0.5)*n = 1;
      else x(1, i* n+1 :(i+0.5)* n) = 0;
      end
  end
subplot(2, 1, 1)
plot (t, x)
hold
return % Third jump back to (B)
end.
```

% Function 4 - Polar RZ

```
function [x] = prz(bitperiod, bits, samperiod)
T = length(bits)* bitperiod;
t = [0: samperiod : T- samperiod];
n = bitperiod/ samperiod;
x = zeros(1, length(t));
for i=0: length(bits)-1
        if bits(i+1)==1
          x(1, i*n+1:(i+0.5)*n)= 1;
        else x(1, i*n+1:(i+0.5)*n)= -1;
        end
 end
subplot(2,1,1)
plot(t, x)
hold
return % Third jump back to (B)
end
```

→ —

% Function 5 : AMI

```
function [x]= ami(bitperiod, bits, samperiod)
T = length(bits)* bitperiod;
n = bitperiod/samperiod;
t = [0 : samperiod : T-samperiod];
  x = zeros(1, length(t));
   c = -1; % To track no. of 1's.
for i = 0 : length(bits)-1
        if bits(i+1)==1;
            c = c+1;
```

```
        if mod(c,2)==0
        x(1, i*n+1:(i+1)*n)=1;
        else x(1, i*n+1:(i+1)*n)=-1;
        end
    else x(1, i*n+1:(i+1)*n)=0;
    end
end
subplot(2,1,1)
plot(t,x)
hold
return % Third jump back to (B)
end
```

% Function 6: Manchester

```
function [x] = manchester(bitperiod, bits, samperiod)
T = length(bits)* bitperiod;
t = [0 : samperiod : T-samperiod];
n = bitperiod/samperiod;
X = zeros(1, length(t));
    for i=0 : length(bits)-1
        if bits(i+1)==1
            x(1, i*n+1:(i+0.5)*n)=-1;
            x(1, (i+0.5)*n+1:(i+1)*n)=1;
        else x(1, i*n+1:(i+0.5)*n)=1
            x(1, (i+0.5)*n+1:(i+1)*n)=-1;
        end
    end
```

```
subplot(2,1,1)
plot(t, x)
hold
return % Third jump back to (B)
end
```

```
function psp(y)
subplot(2,1,2)
plot(abs(fft(y)));
end
```

```
% First jump from main function --> (B)
function [y] = code(choice, Tb, x, Ts)
  c = choice; % Jump happens based on choice
  if c==1
     y = unrz(Tb, x, Ts); % Second Jump to Function-1
  else if c==2
     y = prz(Tb, x, Ts); % Second Jump to Function 2
  else if c==3
     y = urz(Tb, x, Ts); % Second Jump to function 3
  else if c==4
     y = prz(Tb, x, Ts); % Second Jump to function 4
  else if c==5
     y = ami(Tb, x, Ts); % Second Jump to function 5
  else if c==6
     y = manchester(Tb, x, Ts); % Second jump to
                                  function 6
```

```
else exit % Fourth jump back to (A) → last jump.
end
  end
  end
  end
  end
```

Akshansh
2011AAPS300U

Assignment 5: Pseudo Random Binary Sequence

Objective: Generation of PN sequence and verification of its properties

Write a Matlab code to generate a PN sequence. You will require the Initial State of the Shift register and a Generator polynomial. Let 'a' be the array containing the Initial Status of the shift register. The length of this array is as long as the number of flip-flops used. Every value in this array indicates the output of the individual flip-flop.

Let 'b' be the Generator Polynomial. The first and the last value in this generator polynomial have to be 1 as it specifies the involvement of the first flip-flop and last one. The other values in the array indicate the taps to be included to perform modulo addition that will result in the feedback value to the first flip-flop. The length of this array therefore has to be length(a)+1.

Generate two pn sequences and check its randomness properties. Check for the run, balance and correlation properties. Use the two pn sequences and find the cross correlation between them and make inferences.

Example of command line inputs

```
>>main
Initial Status within square brackets
[1 0 0 0]
Generator Polynomial within square brackets
[1 0 0 1 1]
For the second binary sequence
Initial Status within square brackets
[1 0 0 0]
Generator Polynomial within square brackets
[1 1 0 0 1]
If Both balance and run properties are not satisfied stat=0
If Balance property is satisfied stat=1
If Run Length property is satisfied stat=2
If Both the properties are satisfied stat=3
observe the figure to check if the correlation property has been satisfied
The obtained pn sequence is
```

To generate PN sequence using 'comm. PNsequence' function

```
hpn = comm.PNSequence('Polynomial', [3 2 0], ...
                      'SamplesPerFrame', 14, 'Initial Conditions', ...
                      [0 0 1]);
x1 = step(hpn);
```

To generate PN sequence using program code

```
function varargout = pnsequence(varagin)
clc
g = 0; % Verification of balance and run lengths properties
% Generation of PN sequance.
a = varagin{1};
x = length(a);
j = 1
i = (2^x)-1;
pn = zeros(1, x);
while i~=0;
   temp = xor(a(x,1), a(x-1), 1));
   a = circshift(a,1);
   temp1 = a(1,1);
   a(1,1) = temp;
   pn(1,j) = temp1;
   i = i-1;
   j = j+1;
end
varagout{1} = pn;
```

% Verification of Balance Property

```
zero = 0;
one = 0;
r = 0;
for k = 1:(2^x)-1
    r = pn(1,k);
    if (k == 0)
        zero = zero + 1;
    else one = one + 1;
    end
end
if (one > zero)
    disp('Balance Property Verified');
    g = g + 1;
else disp('Balance Property not verified');
end
```

% Verification of Run length Property

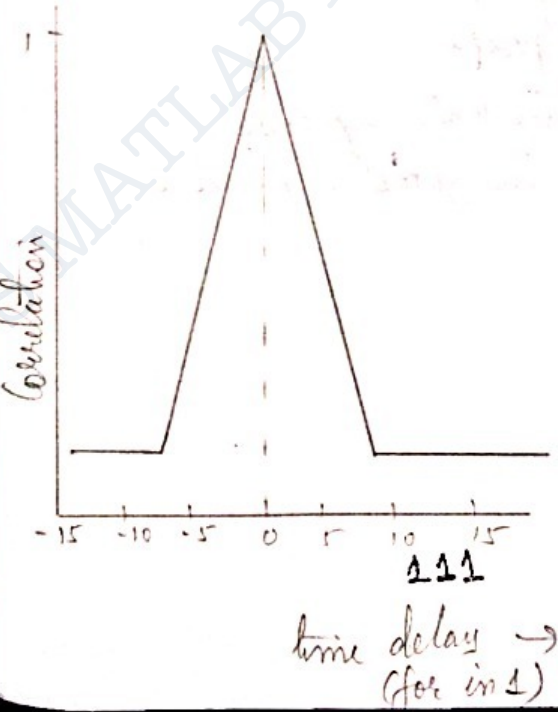
```
Co = 0; % To count no. of one's in a group
Cz = 0; % To count no. of zeros in a group
s = 0; % To keep a count of breaks in one's sequence.
t = 0; % To keep a count of breaks in zeros' sequence.
r = 0;
n = length(pn);
grps = zeros(1,n);
run = 0;
    for i = 1:n
        r = pn(i);
    if r == 0
        t = 1;
        cz = cz + 1;
      if t == s
          s = 0;
        grps(cc) = grps(co) + 1;
        co = 0
        run = run + 1;
    end
    end
    s = 1;
    co = co + 1;
    if t == s
      t = 0;
    grps(cz) = grps(cz) + 1;
    cz = 0;
    run = run + 1;
    end
    end
end
```

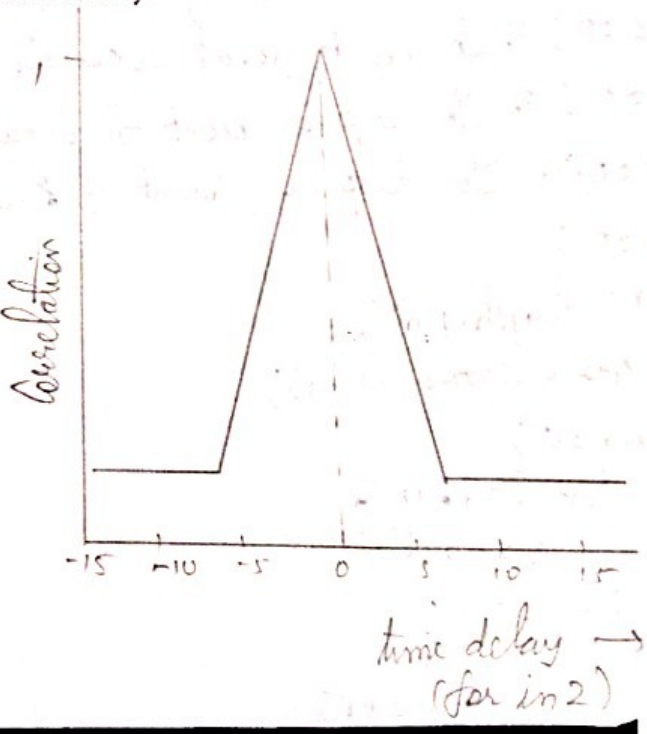
```
if t>s
grps(cz)=grps(cz)+1;
end
if s>t
 grps(co)=grps(co)+1;
end
run=run+1;
 c=1;
 y=1;
for i=1:n-1
 if grps(i)~=0
t=run/k^c);
 if t>1
 if grps(i)==t;
 y=y+1;
 else y=y-1;
 end
 else if grps(i)==1
 y=y+1;
 else y=y-1
  end
 end
c=c+1;
else break
 end
 end
 if y==c
 disp('Run length property
      verified');
 g=g+1;
 else disp('Run length property
          not satisfied');
  end
```

```
 if g==2
 disp('Both properties verified');
 disp('The PN sequence is:');
  pn
 end
% Correlation Calculation
t=length(pn);
m=1;
sum=0;
first=0;
second=0;
corarr
corarr=zeros(1,t);
delay=zeros(1,t);
for l=-t+1:t-1
 pn1=pn;
 pn1=[circshift(pn1',1)]';
for n=1:t
   if pn(1,n)==0
 first=1;
 else first=-1;
end
if pn1(1,n)==0
 second=1;
 else second=-1;
 end
Sum=sum+(first*second);
end
   corarr(1,m)=sum/t;
  delay(1,m)=l;
   m=m+1;
   sum=0;
end
  plot(delay,corarr);
title('Correlation variation with time delay');
xlabel('Time delay');
ylabel('Correlation');
 end
```

```
c =

 Columns 1 through 13

  0  0  0  1  0  0  1  1  0  1  0  1  1

 Columns 14 through 15

  1  1

The second pn sequence is

d =

 Columns 1 through 13

  0  0  0  1  1  1  1  0  1  0  1  1  0

 Columns 14 through 15

  0  1

stat1 =

  3

stat2 =

  3

>>
```

Plot the correlation of the sequence obtained as a function of delay.

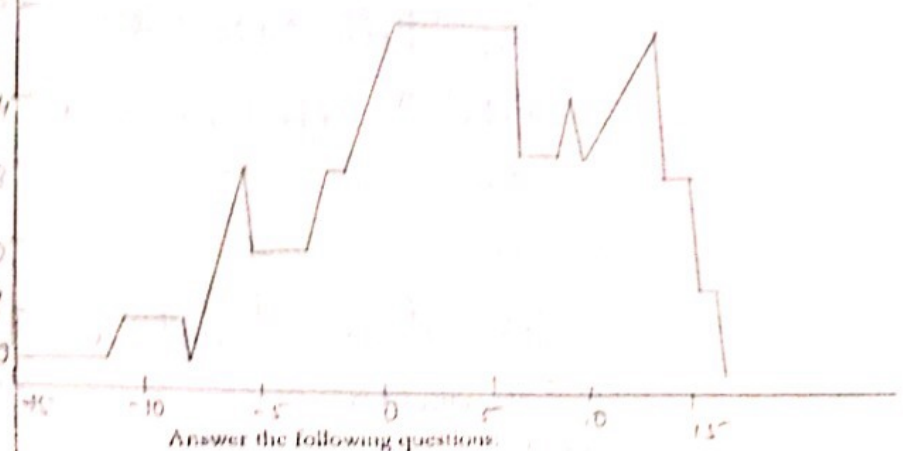




To get cross correlation graph:

```
>> a=[0 0 1 1];
>> pn1=pnsequence(a);
Balance property verified!
>> t=[1 0 0 0];
>> pn=pnsequence(t);
Balance property verified!
>> [b,t]=xcorr(pn,pn1);
>> plot(t,b);
```

Cross Correlation:



Answer the following questions:

1. What is the inference you are able to make with the correlation property and the corresponding plots drawn against delay?

Correlation is calculated for a PN sequence as:
$\frac{1}{N} \sum C_n \times C(n-k)$; $C(n-k)$ = delayed PN sequence
Correlation value decreases from 1 ($k=0$, no delay) to $-\frac{1}{N}$ ($k=n-1$, max delay)

2. What can you infer about the occurrence of maximal length sequence given the size of the shift register?

If no. of shift registers is D, max. length sequence is $2^D - 1$
(all zero state is not included as input to shift register)

3. For a shift register of size 4 and a given initial status [1 0 0 0], how many possible maximal length sequences exist.

one ; $2^D - 1$

Signature of the instructor: _____

Date: 24/11/13