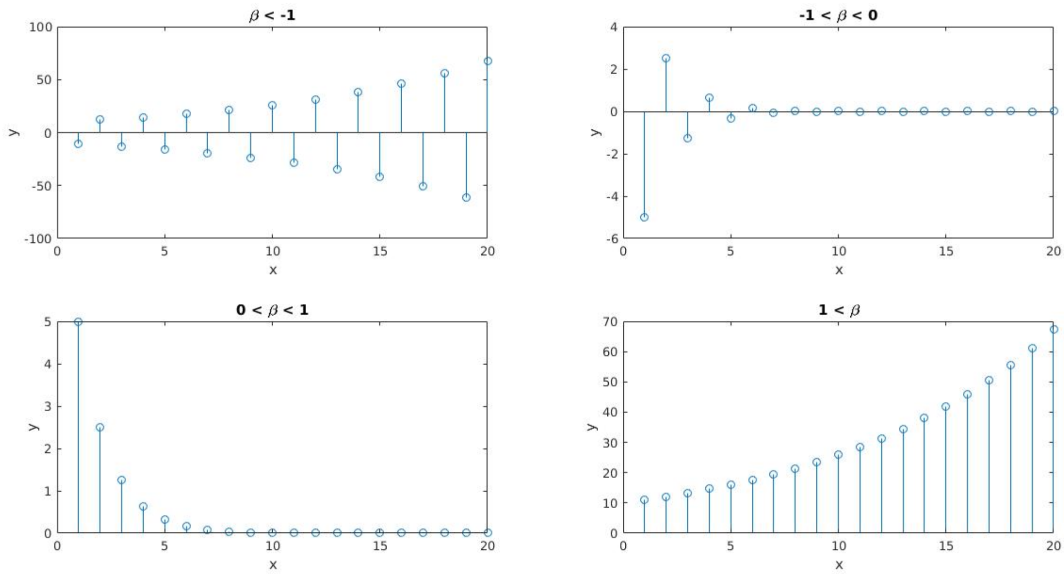
EE387 – DISCRETE TIME SIGNALS

JAYATILAKA G.C.
E/14/158
SEMESTER 06
11/12/2018

(1) Understanding the properties of the Discrete Time Signals

(a)

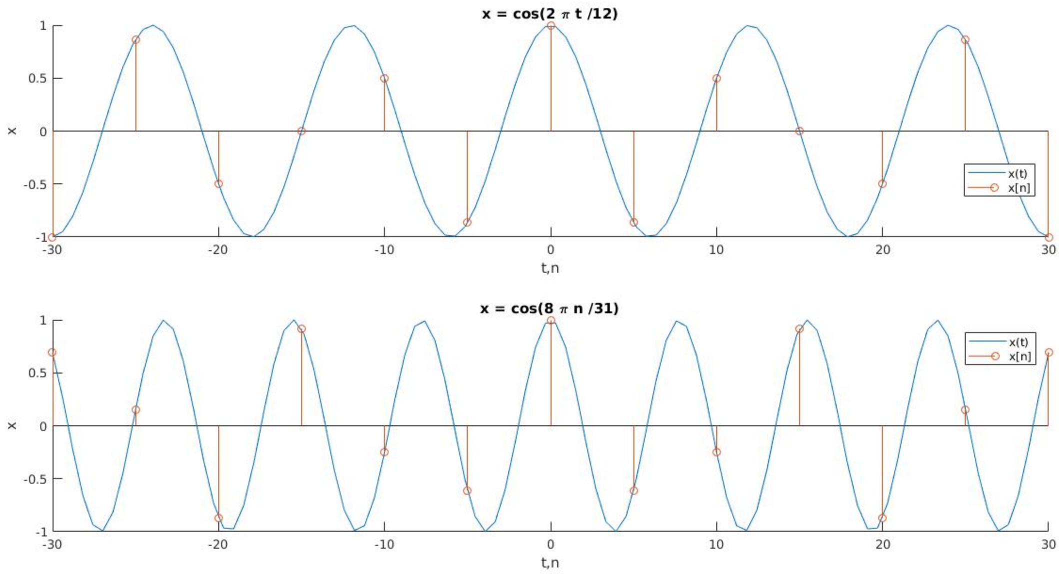
```
% EXERCISE (1) (A) CODE
figure;
title('Ex 01 : A');
hold on;
b=[-1.1,-0.5,0.5,1.1];%Different beta values
n=1:20;
for idx=1:4
    x=10*(b(idx).^n);
    subplot(2,2,idx);
    stem(n,x);
    xlabel('x');
    ylabel('y');
    labels=["\beta < -1","-1 < \beta < 0", "0 < \beta < 1","1 < \
beta"];title(labels(idx));
end
```



(b)

```
% EXERCISE (1) (B) CODE
clear all;close all;clc;

tStart=-30;
tEnd=30;

w=[pi/6,  8*pi/31];
t=linspace(tStart,tEnd,100);%CT variable
T=5;
k=(tStart/T):(tEnd/T);
n=k*T;%DT variable
```

```
figure;
title('Ex 01 : B');
hold on;


for idx=1:2
    x_t=cos(w(idx)*t);
    x_n=cos(w(idx)*n);

    subplot(2,1,idx);
    hold on;
    plot(t,x_t);%Plot the CT
    stem(n,x_n);%Plot the DT
    xlabel('t,n');
    ylabel('x');
    legend(["x(t)","x[n]"]);
    labels=["x = cos(2 \pi t /12)","x = cos(8 \pi n /31)"];title(labels(idx));
end
```



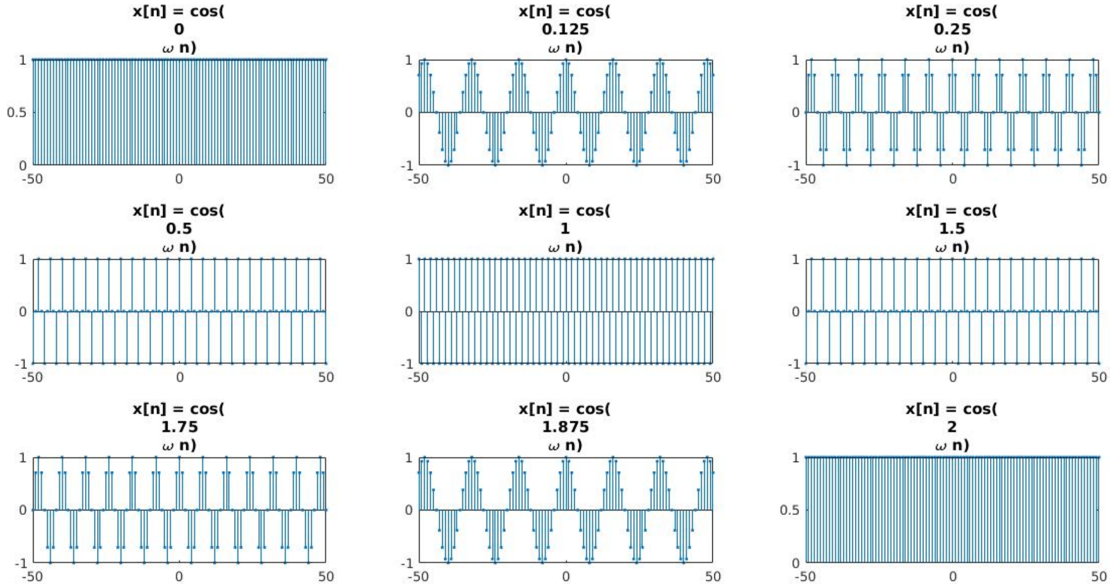
CT Signal	x(t)=cos(2*pi*t /12)	x(t)=cos(8*pi*t /31)
Theoretical time period	12 time units	31/4 time units

DT Signal	x[t]=cos(2*pi*n /12)	x[n]=cos(8*pi*n /31)
Theoretical period	12	31

The observed period is same as the theoretical period for both the CT and DT forms for the first signal.
The second signal's observed period is the DT form's theoretical period.

(c)

```
% EXERCISE (1) (C) CODE
clear all;close all; clc;
tStart=-500;
tEnd=500;

w=[0,pi/8,pi/4,pi/2,pi,3*pi/2,7*pi/4,15*pi/8,2*pi];
t=linspace(tStart,tEnd,1000);%Continuous variable
T=5;
k=(tStart/T):(tEnd/T);
n=k*T;%Discrete variable

for idx=1:length(w)
    subplot(3,3,idx);
    stem(n,cos(w(idx).*n),'.');
    title((["x[n] = cos(",num2str(w(idx)/pi),"\omega n)"]));
end
```



(d)
for x[n]=cos(0ωn) the observed waveform is constant.
When ω increases, th rate of oscillation of the observed waveform increases.
It peaks at x[n]=cos(0ωn).
Then the rate of oscillation of the observed waveform decreases again.

(2) Discrete Convolution

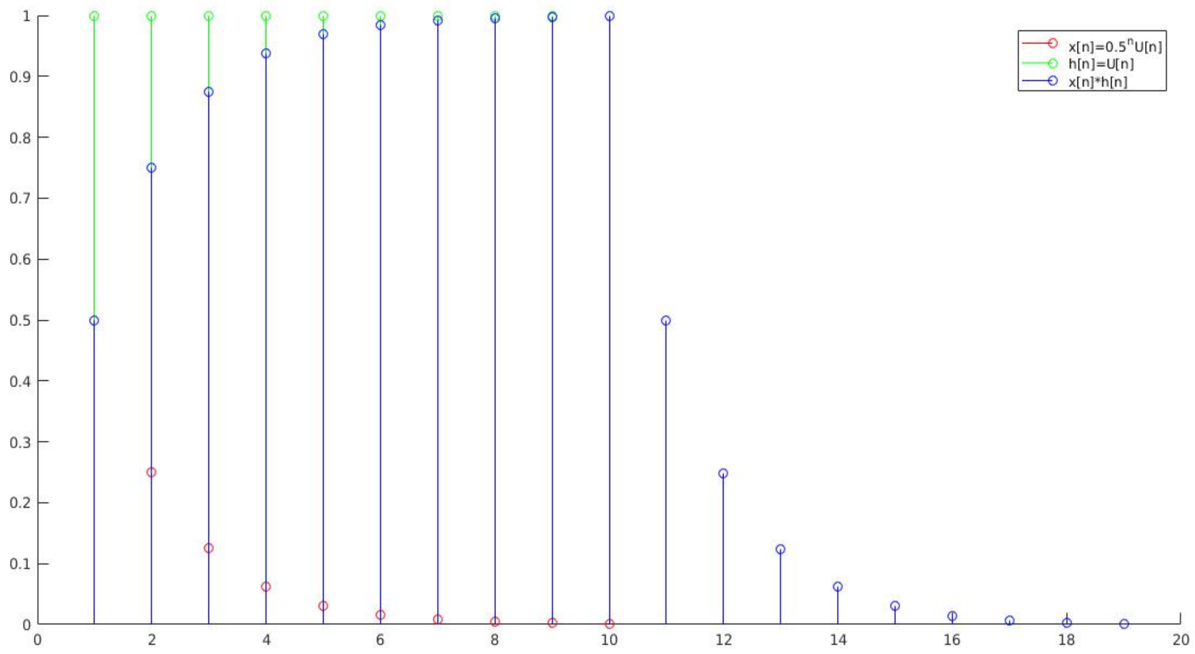
(a)

% EXERCISE (2) (A) CODE

```
% EXERCISE (2) (A) CODE
function [ y] = myConv( x1,x2)
    y=zeros(1,length(x1)+length(x2));%resulting vector
    N=length(y);

    for n=1:N
        for k=1:N
            if (k <= length(x1)) && (n-k >= 1) && (n-k <=length(x2))
                %Checking to see if the variables goes out of the finite
                %array (in which case they are zero)
                y(n)=y(n)+x1(k)*x2(n-k);
            end
        end
    end

    y=y(1,2:length(y));

%     disp('DEBUG OUTPUT:')
%     disp(['        x1 = ',num2str(x1)]);
%     disp(['        x2 = ',num2str(x2)]);
%     disp(['        x1CONVx2 = ',num2str(y)]);

end
```

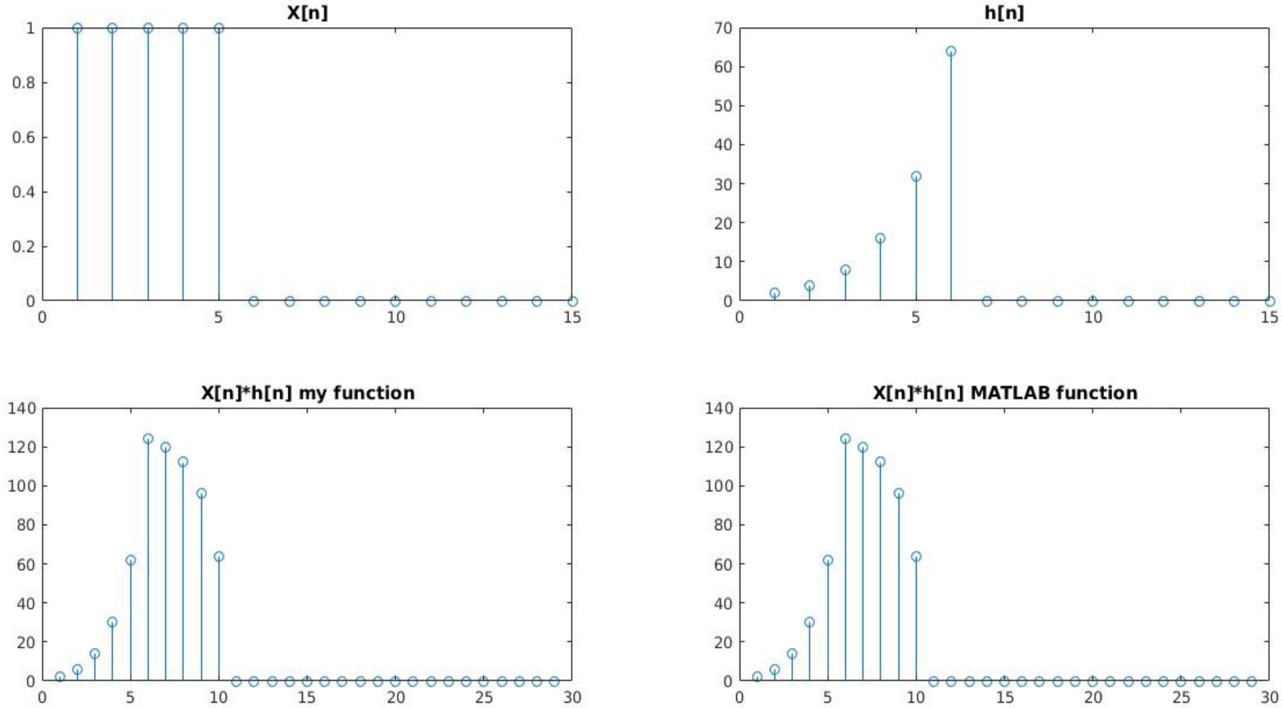
(b)

```
% EXERCISE (2) (B) CODE
close all;clear all; clc;
n=1:10;
x=0.5.^n .* heaviside(n);
h=heaviside(n);

xh=myConv(x,h);

figure;
hold on;
stem(n,x,'r')
stem(n,h,'g');
stem((1:length(xh)),xh,'b');
legend(["x[n]=0.5^nU[n]","h[n]=U[n]","x[n]*h[n]"]);
```



(c)

```
% EXERCISE (2) (C) CODE
close all;clear all; clc;

X=[1   1   1   1   1   0   0   0   0   0   0   0   0   0   0];
h=[2   4   8   16  32  64  0   0   0   0   0   0   0   0   0];
Xh=myConv(X,h);%My implementation
Xhh=conv(X,h);%MATLAB function

subplot(2,2,1);
stem(X);
title("X[n]");

subplot(2,2,2);
stem(h);
title("h[n]");

subplot(2,2,3);
stem(Xh);
title("X[n]*h[n] my function");

subplot(2,2,4);
stem(Xhh);
title("X[n]*h[n] MATLAB function");
```



(iv) The convolution has acted like a range sum for the h[n] sequence.

(1) LTI Systems

(a)

P[n], the net savings per month is the input sequence.
B[n], the bank balance at the end of every month is the output sequence.

B[1]=P[1] since the bank balance at the end of the first month is only the net savings.
B[n]= B[n-1]*101% + P[n] for all other months,
the new bank balance is the previous bank balance, interest for it and the net savings of the month.

```
% EXERCISE (3) (A,i) CODE
function [ B ] = investor(P)
    %P[n], the net savings per month is the input sequence.
    %B[n], the bank balance at the end of every month is the output sequence.

    B=zeros(1,length(P));
    B(1)=P(1);%since the bank balance at the end of the first month is only the net savings.
    for m=2:length(P)
        B(m)=1.01*B(m-1)+P(m);%the new bank balance is the previous bank balance, interest for it and the net savings of the month.
    end

end
```

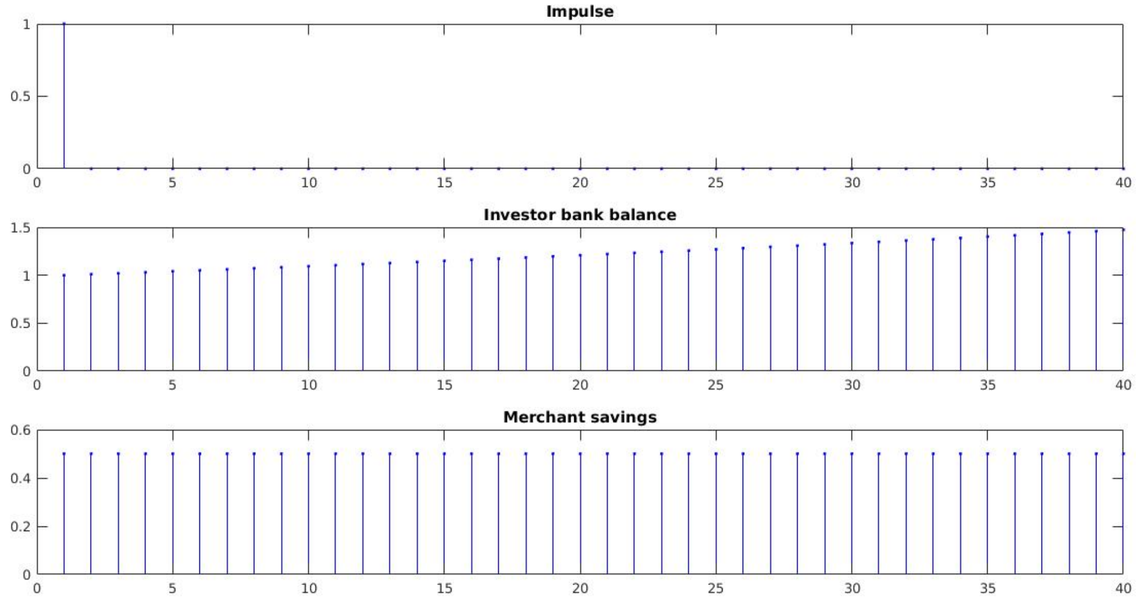
(a)
M[n], monthly earnings is the input sequence.
S[n], the savings at the end of every month is the output sequence.

S[1]= 0.5*M[1] since the balance brought forward is 0 and half the first moth's earnings is saved.
S[n]= S[n-1] + 0.5*M[n] for all other months,
since half of every months' earnings is added to the savings.

```
function [ S ] = merchant( M )
    %M[n], monthly earnings is the input sequence.
    %S[n], the savings at the end of every month is the output sequence.
    S=zeros(1,length(M));
    S(1)=0.5*M(1);
%since the balance brought forward is 0 and half the first mothâ€™s earnings
is saved.
    for m=2:length(M)
        S(m)=S(m-1)+0.5*M(m);
%since half of every monthsâ€™ earnings is added to the savings.
    end

end
```

(b)

```
% EXERCISE (3) (B) CODE
clear all;close all;clc;
im=zeros(1,40);
im(1)=1;
balance=investor(im);
savings=merchant(im);

subplot(3,1,1);
stem(im,'b.');
title("Impulse");
subplot(3,1,2);
stem(balance,'b.');
title("Investor bank balance");
subplot(3,1,3);
stem(savings,'b.');
title("Merchant savings");
```



(c)
Investor’s bank balance is IIR (the previous output is recursively added giving an infinite output with time.)
Merchant’s savings is FIR