EE387 – SIGNAL PROCESSING

LAB 2

GUNARATHNE K.G.G.C.B

E/14/108

SEMESTER 6

12/11/2018

Discrete Time Signals

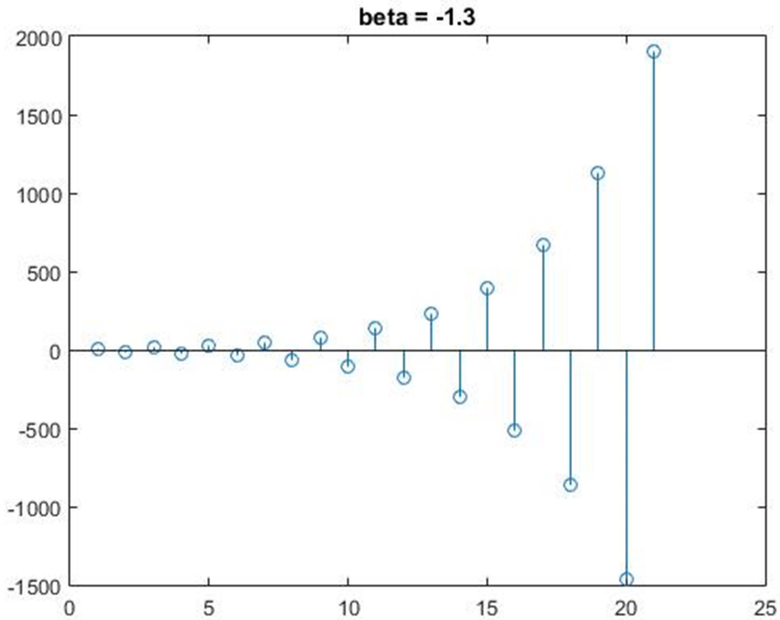
1) Understanding properties of Discrete Time Sinusoidal signals

a.

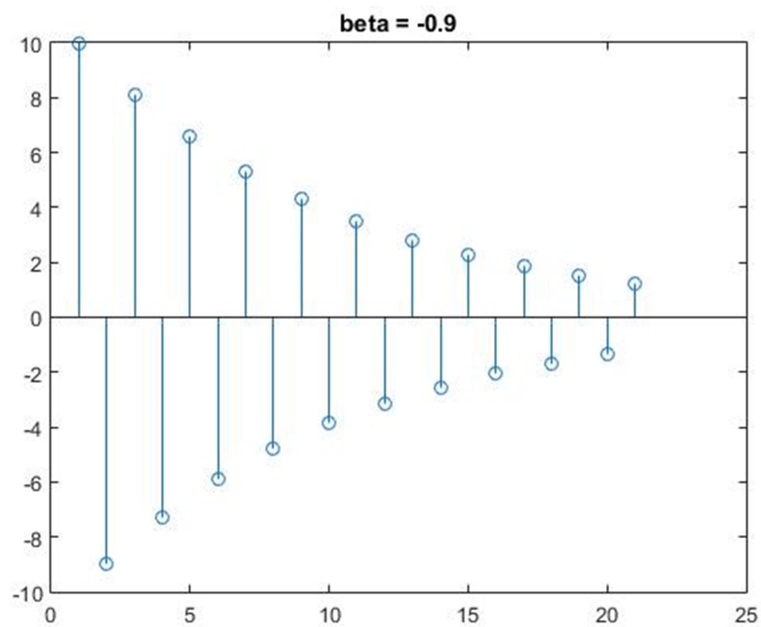
```
close all;

figure;
beta = -2;
fun_beta(beta);
title('beta = -2');

figure;
beta = -0.5;
fun_beta(beta);
title('beta = -0.5');

figure;
beta = 0.5;
fun_beta(beta);
title('beta = 0.5');

figure;
beta = 2;
fun_beta(beta);
title('beta = 3');
```

```
function fun_beta(beta)

n = 0:20;
x = 10*(beta.^n);
stem(x);
```

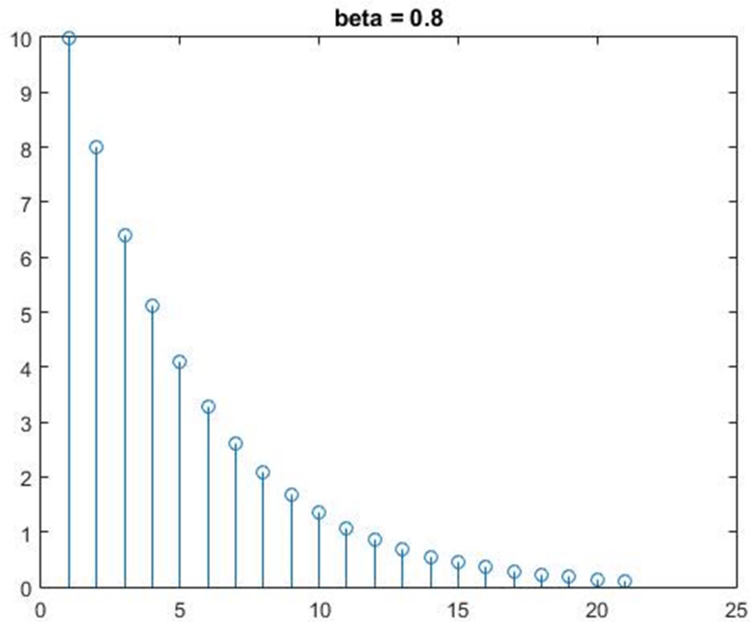
i. $\beta < -1$



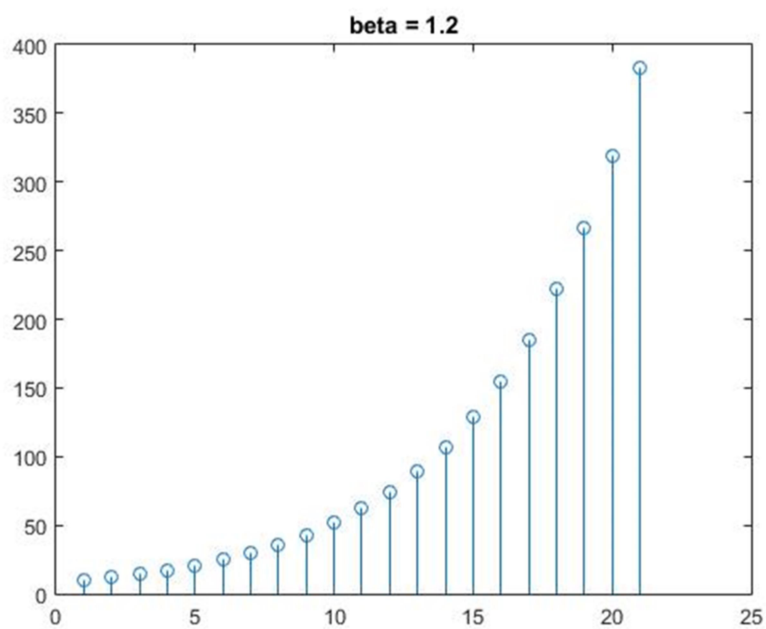
ii. $-1 < \beta < 0$



iii. $0 < \beta < 1$



iv. $\beta > 1$



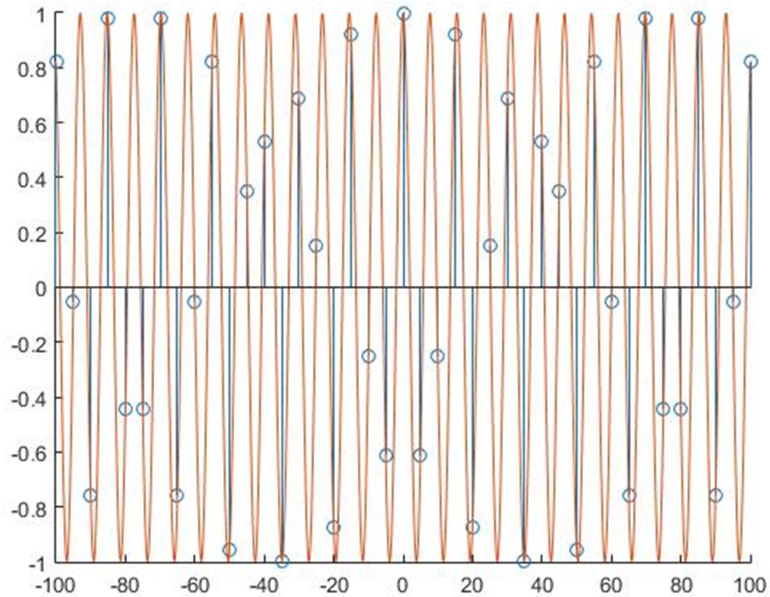
b.

code

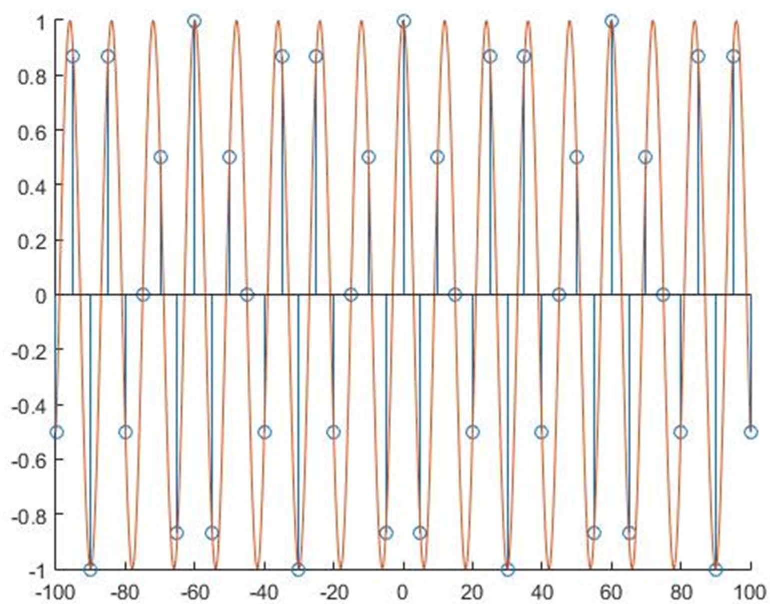
```
function fun_b(eq)

k = -20:20;
T = 5;
n = k*T;
t = linspace(-20*5,20*5,1000);

figure;
hold on;

xnval = subs(eq, n);
xtval = subs(eq, t);

plot(n, xnval, 'r');
plot(t, xtval);

hold off;
```

```
close all;

syms x;
xn = cos((2*pi*x)/12);

fun_b(xn);

xn = cos((8*pi*x)/31);

fun_b(xn);
```

Output



X[n] = cos((8*pi*x)/31)



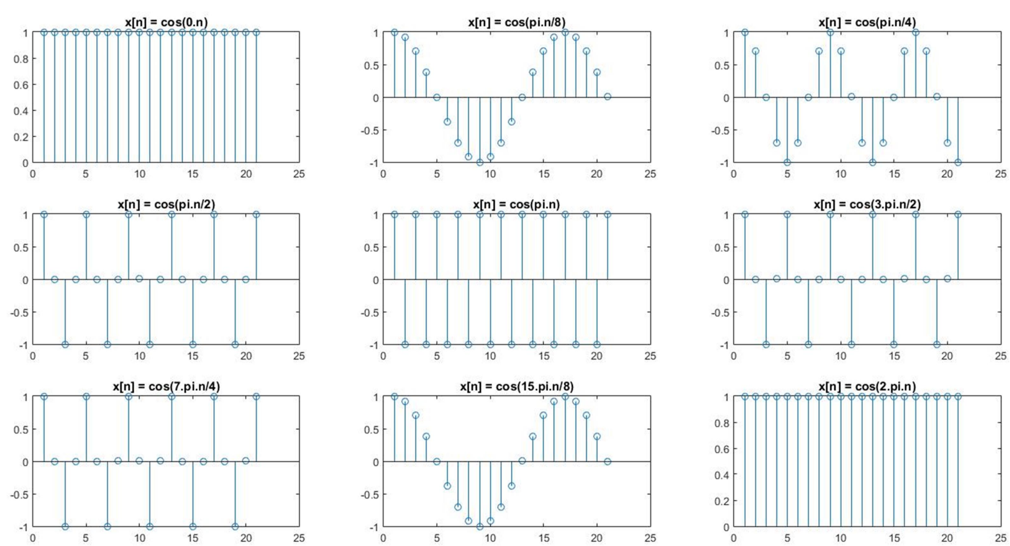
X[n] = cos((2*pi*x)/12

c.

code

```
clear all;

x = 0:20;

subplot(3,3,1)
yl= cos(0*x);
stem(yl);
title('x[n] = cos(0.n)');

subplot(3,3,2)
yl= cos(pi*x/8);
stem(yl);
title('x[n] = cos(pi.n/8)');

subplot(3,3,3)
yl= cos(pi*x/4);
stem(yl);
title('x[n] = cos(pi.n/4)');
```

Output



d. By observing the plots we can identify that the pattern will repeat at 2*pi cycles.

Discrete convolution

a.

```
function y = myconv(x,h)

a = length(x);
b = length(h);
n = a+b-1;
y = zeros(1,n);

for i = 0:n
    for j = 0:n
        if((i-j+1)>0 && (i-j+1)<=b && (j+1)<=a)
             y(i+1) = y(i+1)+ x(j+1).*h(i-j+1);
        end
    end
end
```

b.

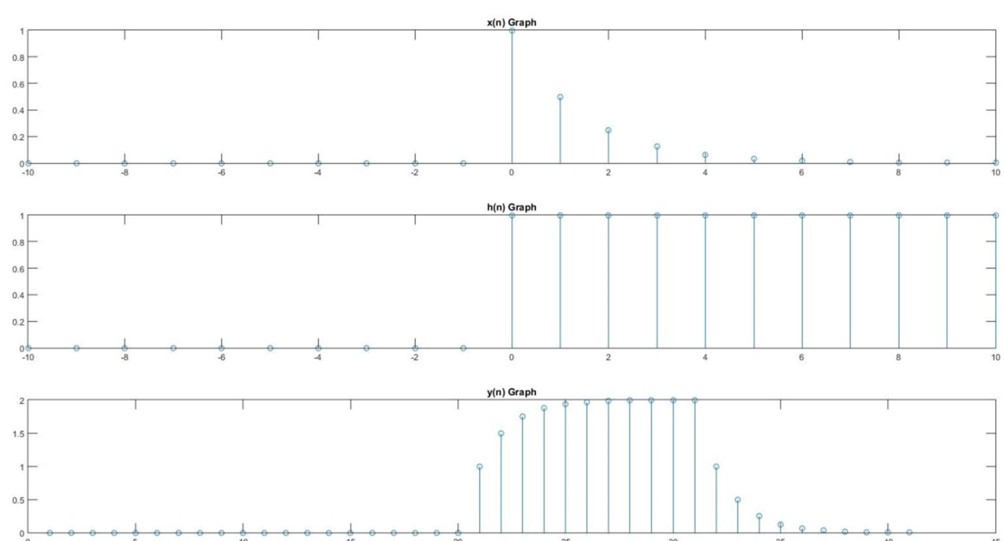
code

```
function x = fun_x(n)

u = unitStep(n);

x = zeros(1,length(n));

for i = 1:length(n)
    x(i) = (0.5^n(i))*u(i);
end
```

```
function u = unitStep(n)

u = [];

for i = n
    if (i>=0)
        u = [u 1];
    else
        u = [u 0];
    end
end
```

```
close all;

n = -10:10;
h = unitStep(n);
x = fun_x(n);
y = myconv(x,h);

subplot(3,1,1);
stem(n,x);
title('x(n) Graph');

subplot(3,1,2);
stem(n,h);
title('h(n) Graph');

subplot(3,1,3);
stem(y);
title('y(n) Graph');
```

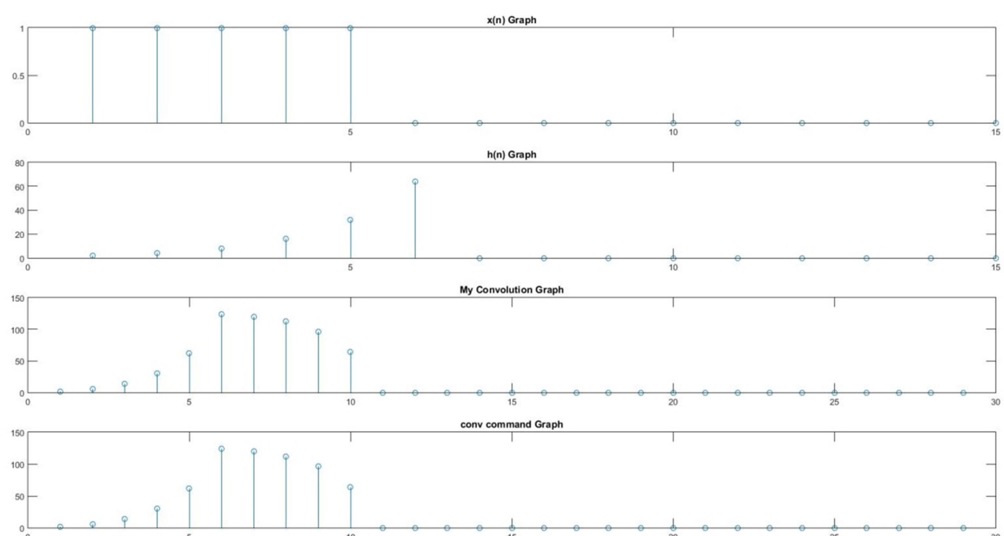
Output



c.

```
close all;

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];

y = myconv(x, h);
y1 = conv(x, h);

subplot(4,1,1);
stem(x);
title('x(n) Graph');

subplot(4,1,2);
stem(h);
title('h(n) Graph');

subplot(4,1,3);
stem(y);
title('My Convolution Graph');

subplot(4,1,4);
stem(y1);
title('conv command Graph');
```

Output



LTI Systems

a.

i)

```
function newbalance = function_investor(b, p, n)

newbalance = [];

for i = 1:length(n)
    b = (b*1.01+p(i));
    newbalance = [newbalance b];

end
```

ii)

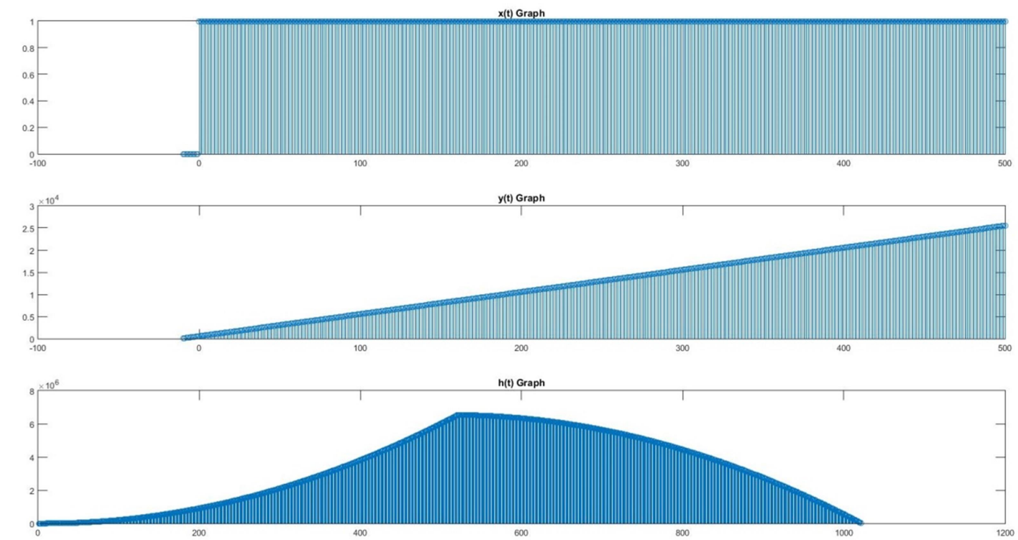
```
function newsaving = function_merchant(saving, m, n)

newsaving = [];

for i = 1:length(n)
    saving = saving+(m/2);
    newsaving = [newsaving saving];
end
```

b.

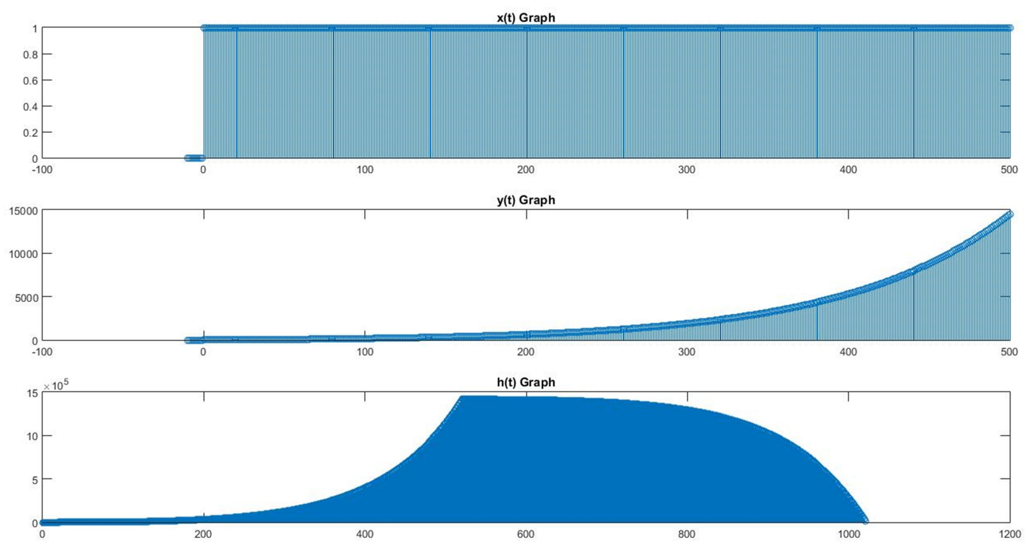
code

```
close all;

figure();
n = -10:500;

x = unitStep(n);
y = function_investor(0, x, n);
ht = conv(y, x);

subplot(3,1,1);
stem(n,x);
title('x(t) Graph');

subplot(3,1,2);
stem(n,y);
title('y(t) Graph');

subplot(3,1,3);
stem(ht);
title('h(t) Graph');

figure();
y = function_merchant(0, 100, n);
ht = conv(y, x);

subplot(3,1,1);
stem(n,x);
title('x(t) Graph');

subplot(3,1,2);
stem(n,y);
title('y(t) Graph');

subplot(3,1,3);
stem(ht);
title('h(t) Graph');
```

Output



Merchant Function Impulse Response



Investor Function Impulse Response

c. we can see that investor bank account function is an IIR system. This is because the function depends on both previous outputs and previous inputs. But in the merchant function only the input matters. So it is a FIR system. It is clearly shown in the graphs. In IIR system the graph grow exponentially while merchant function has a constant increment.