CODE : pn sequence

```
clc;
clear all;
close all;
x1=[1 1 1 1]
n1=length(x1)
len=(2^n1) - 1
p1(1,1)=x1(1,1);
disp(p1)
z1=x1
for y1= 2:len
    x1=z1;
for i = 1 : n1
    if i ==1
        z1(1,i) = xor(x1(1,1), x1(1,4));
    else
        z1(1,i) = x1(1, i-1);
    end
end
p1(1,y1)=z1(1,4);
end
subplot 211;
disp(p1);
stem(p1);
```

Code : effect of cluster size on system capacity

```
clc;
clear all;
close all;
total_bandwidth = 40e6;
channel_bandwidth = 30e3;
cell_radius = 2;
area_circle = 270;
cluster_size = input('num_cluster :');

num_channels =round(total_bandwidth / (2*channel_bandwidth));
area_hex = round(3 * sqrt(3) / 2 * cell_radius^2 );
num_cells = round( area_circle / area_hex);

disp(['Number of channels per cell: ' num2str(num_channels)]);
disp(['Number of cells in the system: ' num2str(num_cells)]);
disp(' ');

cluster_sizes = [cluster_size];
for i = 1:length(cluster_sizes)
   K = cluster_sizes(i);
   cluster_repetition = num_cells / K;
   system_capacity(i) = num_channels * cluster_repetition;
   disp(['System capacity for cluster size ' num2str(K) ': ' num2str(system_capacity(i)) '
channels']);
End
```

CODE : path loss exp on cluster size n sir

```
clc;
close all;
clear all;
n= input ('enter the value of n :');
SIR = 15;
powl=2/n;
SIR=power(10,SIR/10)
N=round((power(6*SIR,powl))/3);
fprintf('N=%d',N);
disp(' ');
valid=0;
for i=0:1:5
    for j=0:1:5
        x=power(i,2)+power(j,2)+i*j;
        if x==N
            valid=1;
            break;
        end
    end
end
arr=zeros();
p=1;
if valid==0
    for i=0:1:5
        for j=0:1:5
            x=power(i,2)+power(j,2)+i*j;
            arr(p)=x;
            p=p+1;
        end
    end
end
sorted_arr=sort(arr);
for i=1:1:15
    if sorted_arr(i)>N
        N=sorted_arr(i);
        break;
    end
end
fprintf('verification  N=%d',N)
```

CODE: Trunking and blocking probability

```
clc;
clear all;
close all;
Radius=1.56;
N=4;
total_ch=80;
A=15.25; %From Erlang C system chart
Au=0.029;
lambda=1;
prob=0.05;

disp('1. User per square kilometer')
disp('Number of channel per cell')
chno= total_ch/N
x=sqrt(3);
cell_a=((3*x)/2)*Radius *Radius
user_in_cell=A/Au
user_per_cell=user_in_cell/cell_a
disp('2. Probability for delay call will have to for more than given sec')
time=input('Enter the time ')
t=time/3600;
H=Au/lambda;
Pr =exp(-(chno-A)*(t/H))
disp('3. Probability in percentage for delay call will be delayed for more than given sec')
Probability =(prob* Pr)*100
```

CODE: propagation path loss using hatamodel

```
clc;
clear all ;
close all;
fc1=input('Enter the frequency fc1:');
fc2=input('Enter the frequency fc2:');
ht =100;
hr=4;
r=2:25;
alphar=3.2*((log10(11.75)*hr)^2)-4.97
Lph1= 68.75+26.16*log10(fc1)-13.82*log10(ht)-alphar+(44.9-6.55*log10(ht))*log10(r)
Lph2= 68.75+26.16*log10(fc2)-13.82*log10(ht)-alphar+(44.9-6.55*log10(ht))*log10(r)
p=plot(r,Lph1,r,Lph2);
ylabel('Lph(db)');
xlabel('Distance(Km)');
title('Propagation Path Loss using Hata Model')
```

OUTPUT:

```
Enter the frequency fc1:700
Enter the frequency fc2:900
```

CODE: Doppler shift

```
clc;
clear all ;
close all ;
v = 19.44 ;
fc = 900000000 ;
lambda=0.33
x = input('Enter the theta value')
fd = round ((v/lambda)*cos(x))
fcnew =round(fc + fd)
```

code: implement CDMA system where 2 users transmit data over single channel.

INPUT:
```
clc
clear all;
close all;
u1 = input('Enter Polar Form of User Data 1 : ');
u2 = input('Enter Polar Form of User Data 2 : ');
pn1 = input('Enter Polar Form of PN1 : ');
pn2 = input('Enter Polar Form of PN2: ');
disp('multiplication of user data and pn sequence')
pn11 = u1(1) * pn1;
pn12 = u1(2) * pn1;
pn13 = u1(3) * pn1;
pn14 = u1(4) * pn1;
disp([pn11 pn12 pn13])
pn21 = u2(1) * pn2;
pn22 = u2(2) * pn2;
pn23 = u2(3) * pn2;
pn24 = u2(4) * pn2;
disp([pn21 pn22 pn23])
disp('then add the new user1 and user2 bits')
a= pn11 + pn21 ;
b= pn12 + pn22 ;
c= pn13 + pn23 ;
d= pn14 + pn24 ;
disp('for first user')
R1_1 = a .* pn1 ;
R1_2 = b .* pn1 ;
R1_3 = c .* pn1 ;
R1_4 = d .* pn1 ;
disp([R1_1 R1_2 R1_3 R1_4])
disp('for second user')
R2_1 = a .* pn2 ;
R2_2 = b .* pn2 ;
R2_3 = c .* pn2 ;
R2_4 = d .* pn2 ;
disp([R2_1 R2_2 R2_3 R2_4])
disp('final answer for user1')
F1_1 = sum(R1_1)/4;
F1_2 = sum(R1_2)/4;
F1_3 = sum(R1_3)/4;
F1_4 = sum(R1_4)/4;
disp([F1_1 F1_2 F1_3 F1_4]);
disp('final answer for user2')
F2_1 = sum(R2_1)/4;
F2_2 = sum(R2_2)/4;
F2_3 = sum(R2_3)/4;
F2_4 = sum(R2_4)/4;
disp([F2_1 F2_2 F2_3 F2_4]);
```

code : walsh code using Hadamard

INPUT:

```
clc;
close all;
H1 = input('Enter 1 or 0 :');
disp('1X1 matrix is ')
disp(H1)
H2 = [H1 H1; H1 not(H1)];
disp('2X2 matrix is ')
disp(H2)
H4 = [H2 H2; H2 not(H2)];
disp('4X4 matrix is ')
disp(H4)
H8 = [H4 H4; H4 not(H4)];
disp('8X8 matrix is ')
disp(H8)
H16= [H8 H8; H8 not(H8)];
disp('16X16 matrix is ')
disp(H16)

disp('To check the Orthogonality:')
row1 = input('Enter 1st row no. :');
row2= input('Enter 2nd row no. :');
total_No_bits = numel(H16(row1, :));
agree = 0;
disagree = 0;
for num = 1:total_No_bits
   if H16(row1,num) == H16(row2,num)
      agree=agree + 1;
   else
      disagree = disagree + 1;
   end
end
result = (disagree - agree) / total_No_bits;
if result == 0
   disp('satisfying Orthogonality')
   disp(result)
else
   disp('Not satisfying Orthogonality')
   disp(result)
end
```

code : BER

```
clc;
clear all;
close all;
L=3;
N=256;
SNR = 0:25;
y = db2pow(SNR);
a = nchoosek(5,3);
b = 2*N*y;
BER1= a*((1./b).^2);
L=4;
a = nchoosek(7,4);
b = 2*N*y;
BER2= a*((1./b).^2);
y=0:2:25;
p = plot(SNR,BER1,SNR,BER2,'--');
xlabel('SNR (dB)');
ylabel('Bit Error Rate');
```