```
clc;
clearvars -except STORE INDEX;
close all;
%rng default;
%INDEX=1; %  % COMMENT BEFORE RUNNING
rng (INDEX,'twister');
% Parameters to Vary for several data bits
% 1.) SNR VS BER
% 2.) SIR VS BER
% 3.) SINR VS BER
% 4.) Ns (OR) DATA RATE VS BER
% 5.) Nu VS BER for SIR=0 dB (i.e, under ideal power control)

% For Repeatable Random Number Results, the convention to use is:
% Initialize rng (RUN NUMBER,'twister'). Store the RUN NUMBER in a struct
% Use a seperate struct for each parameter variation.
% Setting Global Parameters
tsample=0.01;
Tf=50; % Total Frame Time
```

```
prompt = {'Number of Bits:','SNR:','SIR:','Number of Monocycles','Number of Active Users'};
dlg_title = 'Input';
num_lines = 1;
defaultans = {'1000','0','-17','1','8'};
answer = inputdlg(prompt,dlg_title,num_lines,defaultans);
data = str2double(answer);
```

```
cca=[prompt;answer'];
```

```
Frame_Duration=Tf;
Pulse_Distort_Flag=0;
Distance=[];
dd={'Frame_Duration','Pulse_Distort_Flag','Distance'};
dd1=num2str(Frame_Duration);
dd2=num2str(Pulse_Distort_Flag);
dd3=num2str(Distance);
ddd=[dd; {dd1 dd2 dd3}];
%INDEX=1;
aa={'RNG(INDEX_GENERATOR)'; [num2str(INDEX),'_twister']};
cc=[cca ddd aa];
```

```
Ns=data(4);   % Number of Monocycles transmitted for one bit
Nu=data(5);   % Number of Active Users ( Desired User + Interferers)
T= 0:tsample:(Tf*Ns)-tsample;
Bit_Duration=Ns*Tf*1e-9;
Data_Rate=1/Bit_Duration;
fprintf('Bit Duration = %1.2f ns', Bit_Duration/1e-9);
```

```
Bit Duration = 50.00 ns
```

```
fprintf('Data Rate = %1.2f Mbps', Data_Rate/1e6);
```

```
Data Rate = 20.00 Mbps
```

```
ff=sprintf('%1.2f Mbps', Data_Rate/1e6);
Nbits=data(1);
bits=round(rand(1,Nbits));
SNR=data(2); % dB
SIR_dB=data(3);
ccc1={'Data Rate'; ff};
c2=[ccc1 cc]
```

```
c2 =
    'Data Rate'     'Number of Bits:'    'SNR:'    'SIR:'    'Number of Monocycles'    'Number of Active Users'    'Frame_Duration'    'Pulse_Distort_Flag'
    '20.00 Mbps'    '1000'               '-16'     '-17'     '1'                       '8'                         '50'                '0'
```

```
% Desired User & Interferers
%bits= [1 1 1 0 0 0 1 0 1 0];
% function [ Data_Frame, Correlator_Frame, PPM, Correlation_Template, Code, No_of_Hops] = Signal_Frame_Generate(Frame_Duration, No_of_Monocycles, Data_Bit
% function [ Normz_Interferer, Code, Distance, Distance_Index ] = Interferer_Frame_Generate( Desired_Signal, No_of_Active_Users, SIR_dB, Frame_Duration, N
% function [ SPulse_Distort, SPulse_Correct, Distance ] = Distance_Pulse_Distort_Loop( Signal_Template, Correlation_Template_Flag, Correlation_Template, N
% function [ SPulse_Distort, SPulse_Correct ] = Distance_Pulse_Distort( Signal_Template, Correlation_Template, Distance )

% Initializing storage vectors for Signal, Signal+Noise, Signal+Intererer & Signal+Noise+Interferer
Signal=zeros(Nbits,Ns);
TSignal=zeros(1,Nbits);
DataBit_Signal=zeros(1,Nbits);
```

```
     Signal_Noise=Signal;
     TSignal_Noise=TSignal;
     DataBit_Signal_Noise=DataBit_Signal;
Signal_Interferer=Signal;
TSignal_Interferer=TSignal;
DataBit_Signal_Interferer=DataBit_Signal;
     Signal_Interferer_Noise=Signal;
     TSignal_Interferer_Noise=TSignal;
     DataBit_Signal_Interferer_Noise=DataBit_Signal;

h=waitbar(0,'1','Name','Fractional Progress');
for b=1:Nbits
[D,C,P,CT]=Signal_Frame_Generate(Tf,Ns,bits(b),Pulse_Distort_Flag,Distance);
K=Interferer_Frame_Generate(D,Nu,SIR_dB,Tf,Ns,0,Pulse_Distort_Flag);
ID=D+K;
% Even very low SNR doesn't corrupt the results as expected because
% the signal power is concentrated over a very small time interval
% whereas the noise power is distributed throughout the entire frame.
ND=awgn(D,SNR,'measured');
% plot(T,ND)
NID=awgn(ID,SNR,'measured');
% plot(T,NID)

% Correlation between Data Frame and Correlation Template for entire data bit
CS=reshape(C,length(D)/Ns,Ns);
DS=reshape(D,length(D)/Ns,Ns);
AS=reshape(ND,length(D)/Ns,Ns);
IS=reshape(ID,length(D)/Ns,Ns);
IAS=reshape(NID,length(D)/Ns,Ns);

    DxC=DS.*CS;
    Signal(b,:)=sum(DxC);
      TSignal(b)=sum(Signal(b,:));
    DataBit_Signal(b)=~(TSignal(b)>0);

    AxC=AS.*CS;
    Signal_Noise(b,:)=sum(AxC);
      TSignal_Noise(b)=sum(Signal_Noise(b,:));
    DataBit_Signal_Noise(b)=~(TSignal_Noise(b)>0);

     IxC=IS.*CS;
    Signal_Interferer(b,:)=sum(IxC);
      TSignal_Interferer(b)=sum(Signal_Interferer(b,:));
    DataBit_Signal_Interferer(b)=~(TSignal_Interferer(b)>0);

     IAxC=IAS.*CS;
    Signal_Interferer_Noise(b,:)=sum(IAxC);
      TSignal_Interferer_Noise(b)=sum(Signal_Interferer_Noise(b,:));
    DataBit_Signal_Interferer_Noise(b)=~(TSignal_Interferer_Noise(b)>0);

    steps=length(bits);

waitbar(b/steps,h,sprintf('Iteration Number = %1.0f\n',b))
end
delete(h)
```

```
% BER Computation
BER_Signal=sum(DataBit_Signal~=bits)/Nbits
```

```
BER_Signal = 0
```

```
BER_Signal_Noise=sum(DataBit_Signal_Noise~=bits)/Nbits
```

```
BER_Signal_Noise = 0
```

```
BER_Signal_Interferer=sum(DataBit_Signal_Interferer~=bits)/Nbits
```

```
BER_Signal_Interferer = 0.0630
```

```
BER_Signal_Interferer_Noise=sum(DataBit_Signal_Interferer_Noise~=bits)/Nbits
```

```
BER_Signal_Interferer_Noise = 0.1260
```

```

```

```
br1={'BER_Signal','BER_Signal_Noise','BER_Signal_Interferer','BER_Signal_Interferer_Noise'};
br2= num2str(BER_Signal);
br3= num2str(BER_Signal_Noise);
br4= num2str(BER_Signal_Interferer);
br5= num2str(BER_Signal_Interferer_Noise);
br=[br1; {br2, br3, br4, br5}];
```

```
sz_var_1=size(Signal)
```

```
sz_var_1 = 1x2 double

        1000           1
```

```
%sz_var_1=size(DxC)
sz_var=sz_var_1(1)*sz_var_1(2)
```

```
sz_var = 1000
```

```
% Case 1  SQUARE FIRST SUM SECOND
% sum((Signal-Signal_Noise).^2) is like 2^2 + 3^2 = 4 + 9 = 13
Signal_Power=sum(sum(Signal.^2))/(sz_var-1);
Signal_Power_dB=10*log10(Signal_Power)
```

```
Signal_Power_dB = 24.8336
```

```
Noise_Var=sum(sum((Signal_Noise-Signal).^2))/(sz_var-1);
Noise_Var_dB=10*log10(Noise_Var)
```

```
Noise_Var_dB = 4.5882
```

```
Interferer_Var=sum(sum((Signal_Interferer-Signal).^2))/(sz_var-1);
Interferer_Var_dB=10*log10(Interferer_Var)
```

```
Interferer_Var_dB = 24.6038
```

```
Interferer_Noise_Var=sum(sum((Signal_Interferer_Noise-Signal).^2))/(sz_var-1);
Interferer_Noise_Var_dB=10*log10(Interferer_Noise_Var)
```

```
Interferer_Noise_Var_dB = 26.2401
```

```
% Filtered through the template signal
RX_SNR=Signal_Power/Noise_Var;
RX_SNR_dB=10*log10(RX_SNR)
```

```
RX_SNR_dB = 20.2454
```

```
RX_SIR=Signal_Power/Interferer_Var;
RX_SIR_dB=10*log10(RX_SIR)
```

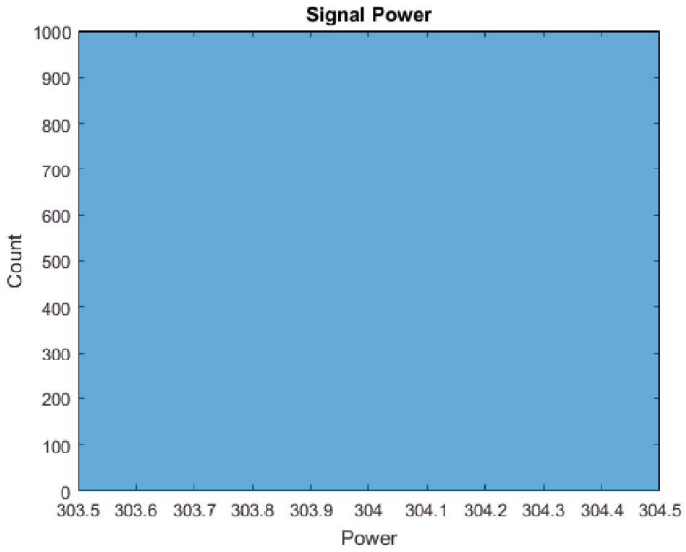
```
RX_SIR_dB = 0.2298
```

```
RX_SINR=Signal_Power/Interferer_Noise_Var;
RX_SINR_dB=10*log10(RX_SINR)
```

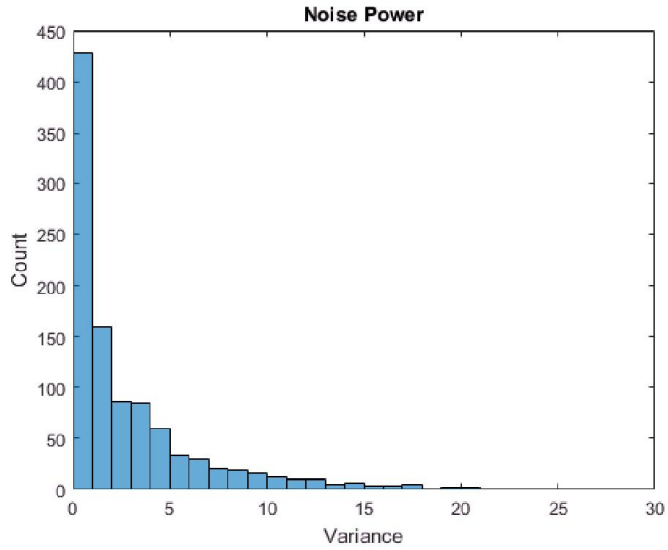
```
RX_SINR_dB = -1.4065
```

```
vs1={'RX_SNR', 'RX_SIR','RX_SINR'};
vs2={'RX_SNR_dB', 'RX_SIR_dB','RX_SINR_dB'};
vt1=num2str(RX_SNR);
vt2=num2str(RX_SNR_dB);
vt3=num2str(RX_SIR);
vt4=num2str(RX_SIR_dB);
vt5=num2str(RX_SINR);
vt6=num2str(RX_SINR_dB);
vvv=[vs1; {vt1, vt3, vt5} ; vs2; {vt2, vt4, vt6}];

vr1={'Signal_Power','Noise_Var', 'Interferer_Var','Interferer_Noise_Var'};
vr2={'Signal_Power_dB','Noise_Var_dB', 'Interferer_Var_dB','Interferer_Noise_Var_dB'};
vdd1=num2str(Signal_Power);
vdd2=num2str(Signal_Power_dB);
vd1=num2str(Noise_Var);
vd2=num2str(Noise_Var_dB);
vd3=num2str(Interferer_Var);
vd4=num2str(Interferer_Var_dB);
vd5=num2str(Interferer_Noise_Var);
vd6=num2str(Interferer_Noise_Var_dB);
vv=[vr1; {vdd1, vd1, vd3, vd5} ; vr2; {vdd2, vd2, vd4, vd6}];
```

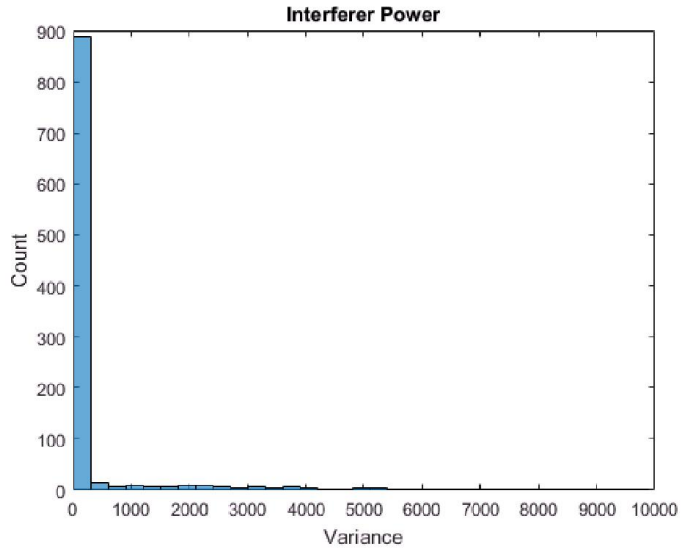
```
histogram(((Signal).^2));
xlabel('Power'); ylabel('Count'); title('Signal Power');
```



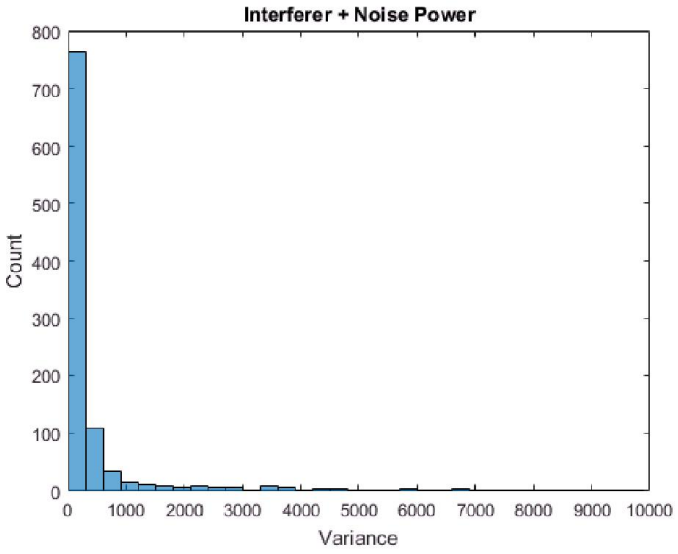

```
histogram(((Signal-Signal_Noise).^2))
xlabel('Variance'); ylabel('Count'); title('Noise Power');
```



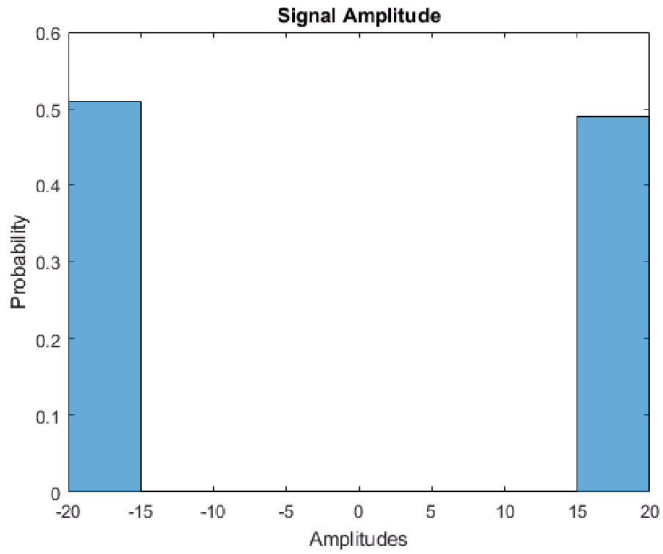

```
histogram(((Signal-Signal_Interferer).^2))
xlabel('Variance'); ylabel('Count'); title('Interferer Power');
```



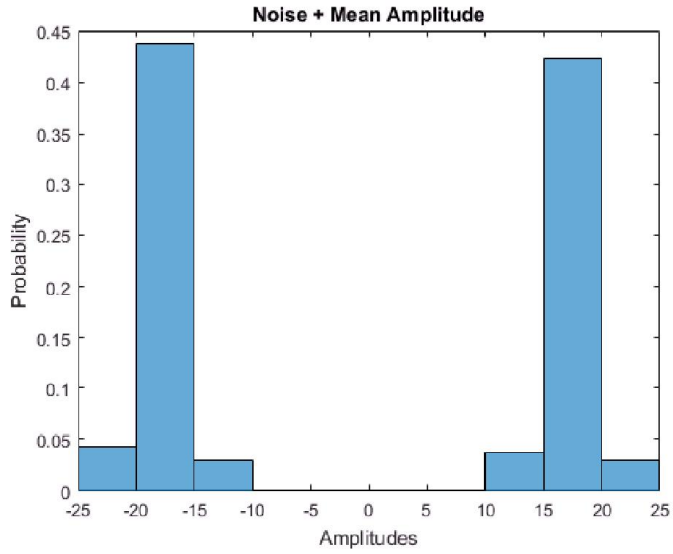

```
histogram(((Signal-Signal_Interferer_Noise).^2))
xlabel('Variance'); ylabel('Count'); title('Interferer + Noise Power');
```



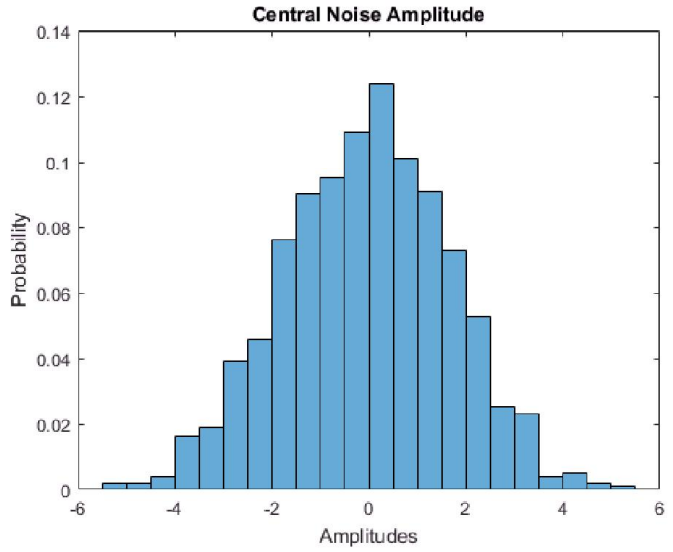

```
histogram(Signal,'Normalization','Probability')
xlabel('Amplitudes'); ylabel('Probability'); title('Signal Amplitude');
```



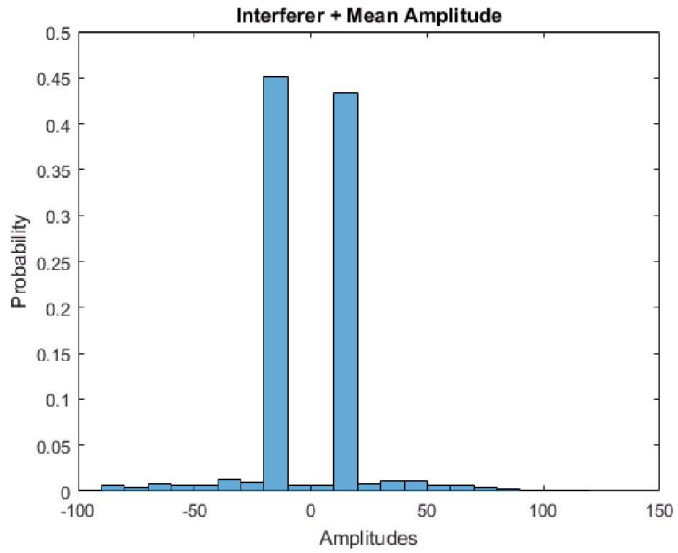

```
histogram(Signal_Noise,'Normalization','Probability')
xlabel('Amplitudes'); ylabel('Probability'); title('Noise + Mean Amplitude');
```



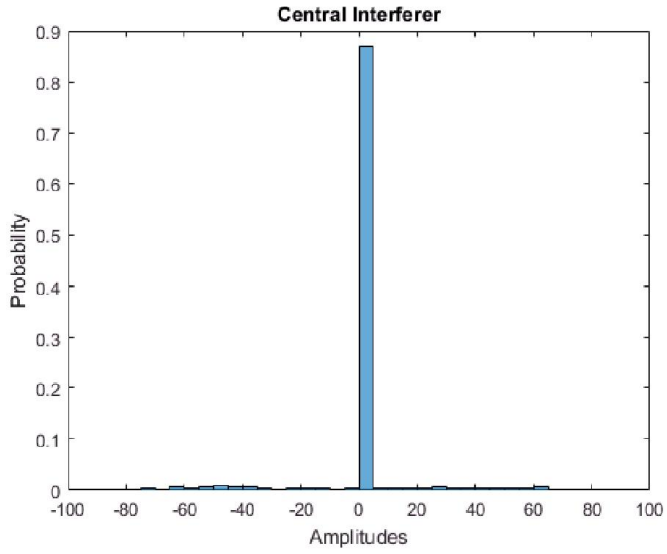

```
histogram(Signal_Noise-Signal,'Normalization','Probability')
xlabel('Amplitudes'); ylabel('Probability'); title('Central Noise Amplitude');
```



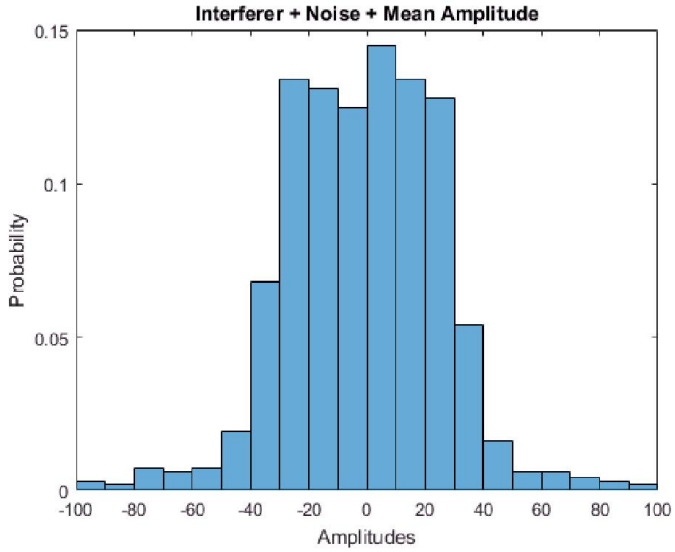

```
histogram(Signal_Interferer,'Normalization','Probability')
xlabel('Amplitudes'); ylabel('Probability'); title('Interferer + Mean Amplitude');
```



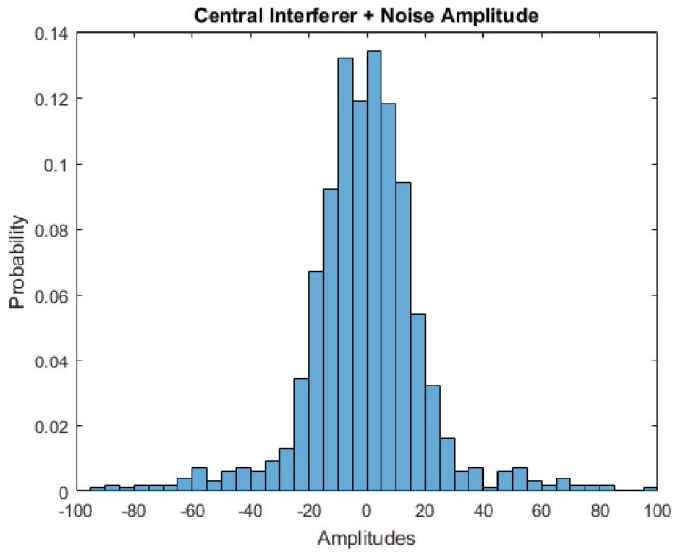

```
histogram(Signal_Interferer-Signal,'Normalization','Probability')
xlabel('Amplitudes'); ylabel('Probability'); title('Central Interferer');
```




```
histogram(Signal_Interferer_Noise,'Normalization','Probability')
xlabel('Amplitudes'); ylabel('Probability'); title('Interferer + Noise + Mean Amplitude');
```




```
histogram(Signal_Interferer_Noise-Signal,'Normalization','Probability')
xlabel('Amplitudes'); ylabel('Probability'); title('Central Interferer + Noise Amplitude')
```



```
STORE(INDEX).Specs = c2;
STORE(INDEX).Parameters=vvv;
STORE(INDEX).Signal=Signal;
STORE(INDEX).Signal_Noise=Signal_Noise;
STORE(INDEX).Signal_Interferer=Signal_Interferer;
STORE(INDEX).Signal_Interferer_Noise=Signal_Interferer_Noise;
STORE(INDEX).Variances=vv;
STORE(INDEX).BER=br;
STORE(INDEX).Specs
```

```
ans =
    'Data Rate'     'Number of Bits:'    'SNR:'    'SIR:'    'Number of Monocycles'    'Number of Active Users'    'Frame_Duration'    'Pulse_Distort_Flag'
    '20.00 Mbps'    '1000'               '-16'     '-17'     '1'                       '8'                         '50'                '0'
```

```
INDEX=INDEX+1
```

```
INDEX = 9
```

```
%STORE=rmfield(STORE, 'TSignal_Interferer_Noise')
STORE
```

```
STORE =
1x8 struct array with fields:

    Specs
    Parameters
    Signal
    Signal_Noise
    Signal_Interferer
    Signal_Interferer_Noise
    Variances
    BER
```

```
% STORE.Variances
% STORE.BER
% STORE.Parameters
% u=STORE.Variances;
% uu=str2double(u(2,:))
% uuu=mean(uu)
```

```
% Accessing data from a struct
[a b, c d, e f g h]=STORE.Parameters; % Store all the columns of the struct for a particular field in seperate cells.
x=[a b c d e f g h] % Concatenate all the seperate cells into a single cell.
```

```
x =
  Columns 1 through 23

    'RX_SNR'       'RX_SIR'       'RX_SINR'       'RX_SNR'       'RX_SIR'       'RX_SINR'       'RX_SNR'       'RX_SIR'       'RX_SINR'       'RX_SNR'
    '1.0833'       '1.044'        '0.021046'      '2.0297'       '1.0886'       '0.037068'      '3.7959'       '1.2471'       '0.071554'      '8.2534'
    'RX_SNR_dB'    'RX_SIR_dB'    'RX_SINR_dB'    'RX_SNR_dB'    'RX_SIR_dB'    'RX_SINR_dB'    'RX_SNR_dB'    'RX_SIR_dB'    'RX_SINR_dB'    'RX_SNR_dB'
    '0.34729'      '0.18681'      '-16.7683'      '3.0743'       '0.36857'      '-14.31'        '5.7932'       '0.95886'      '-11.4537'      '9.1663'

  Column 24

    'RX_SINR'
    '0.72336'
    'RX_SINR_dB'
    '-1.4065'
```

```
[a b c d e f g h]=x{4,3:3:end}; % Index Access the required value (which are in the from of strings) and store them in individual variables.
xx=str2double({a b c d e f g h})
```

```
xx = 1x8 double

  -16.7683  -14.3100  -11.4537   -8.7048   -6.0040   -4.1309   -2.8974   -1.4065
```

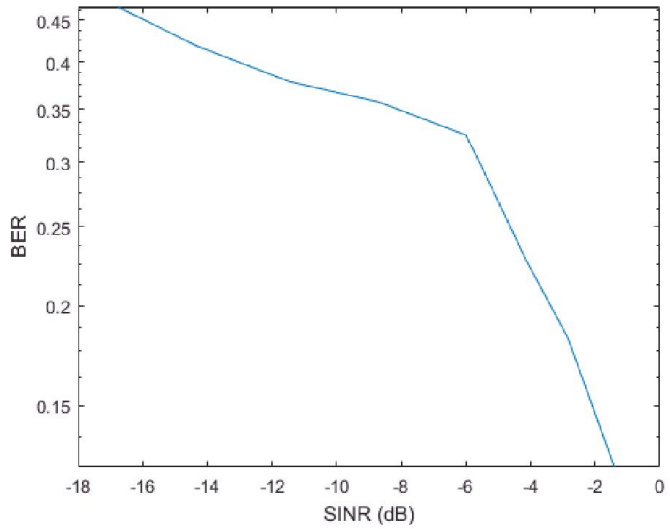
```
[a b c d e f g h]=STORE.BER;
y=[a b c d e f g h]
```

```
y =
  Columns 1 through 16

    'BER_Signal'    'BER_Signal_Noise'    'BER_Signal_Interf…'    'BER_Signal_Interf…'    'BER_Signal'    'BER_Signal_Noise'    'BER_Signal_Interf…'    'BER_
    '0'             '0.151'               '0.045'                 '0.467'                 '0'             '0.071'               '0.05'                  '(

  Columns 17 through 32

    'BER_Signal'    'BER_Signal_Noise'    'BER_Signal_Interf…'    'BER_Signal_Interf…'    'BER_Signal'    'BER_Signal_Noise'    'BER_Signal_Interf…'    'BER_
    '0'             '0'                   '0.048'                 '0.325'                 '0'             '0'                   '0.037'                 '(
```

```
yy=str2double({y{2,4:4:end}})
```

```
yy = 1x8 double

    0.4670    0.4180    0.3780    0.3570    0.3250    0.2270    0.1840    0.1260
```

```
semilogy(xx,yy);
xlabel('SINR (dB)'); ylabel('BER');
```




```
% for j=1:8
%     xx(j)=x{4,((j-2)+j*2)}; % To access x{4,1},x{4,4},x{4,7},x{4,10},x{4,13}, . . .
% yy(j)=y{2,j*4}; % To access x{2,4},x{2,8},x{2,12},x{2,14},x{2,18}, . . .
% end
```

```
Subscripted assignment dimension mismatch.
```

```
%     %[a b c d e f g h]=STORE.Parameters;
% plot(xx,semilog(yy));
% xlabel('SNR (dB)'); ylabel('BER');
```