Matrix computation.

```
clc;
clear;

matrix1 = [1 2 3; 4 5 6; 7 8 9];
matrix2 = [9 8 7; 6 5 4; 3 2 1];

matrix1
matrix2

add = matrix1 + matrix2;
add

sub = matrix1 - matrix2;
sub

mul = matrix1*matrix2;
mul

transposeOfMatrix1 = matrix1';
transposeOfMatrix1

inverseOfMatrix1 = inv(matrix1);
inverseOfMatrix1

elemul = matrix1 .*matrix2;
elemul

concat = [matrix1, matrix2];
concat
```

Baaki agr kuch puche toh net se dekh lena

To Plot Sine Wave of frequency 200 Hz.

```
t=[0:0.0001:0.005];
f=200;
x=sin(2*pi*f*t);
plot(t,x);
xlabel("time");
ylabel("sin(x)");
```

To plot a pulse of width 10.

```
clc;
clear;
t = -10:0.005:10;
y = sign(t+5) - sign(t-5);
y = y*0.5;
plot(t, y);
xlabel("time");
ylabel("signal");
```

Plot the spectrum (Amplitude and phase) 0f the pulse generated.

Amplitude:
```
t = -10:0.005:10;
y = sign(t+5) - sign(t-5);
y = y*2;
amp = fft(y);
amp = fftshift(amp);
plot(t, abs(amp)/length(t));
axis([-0.15 0.15 0 2]);
```

phase
```
t = -10:0.005:10;
y = sign(t+5) - sign(t-5);
y = y*2;
phase = fft(y);
phase = fftshift(phase);
phase = angle(phase);
plot(t, phase);
axis([-0.15 0.15 -4 4]);
%change -0.15 0.15 to -4 4 agr sir bole ki thoda slant mein bnna chaiye tha yeh toh
```

Uniform random number and plot its density function. Find its mean and variance.

```
x = rand(1, 10000);
histogram(x);
meanOfx = mean(x);
varianceOfx = var(x);
meanOfx
varianceOfx
```

Generate Gaussian distributed random number and plot its density function. Find its mean and variance.

```
clc;
clear;
x = randn(1, 10000);
histogram(x);
meanOfx = mean(x);
varianceOfx = var(x);
max(x)
min(x)
meanOfx
varianceOfx
```

Compute the Signal to quantization Noise ratio of Uniform Quantization. Plot SNQR versus Quantization levels.

```
clear;
clc;

levels = 1:1000;
amplitude = 5;
frequency = 1;
time = 0:0.001:1;
x = amplitude*sin(2*pi*frequency*time);
mx = max(x);
mn = min(x);
step = (mx-mn)./levels;
for i=levels
    in= round((x-mn)/step(i));
    xq = mn+in*step(i);
    noise = xq-x;
    rmsofnoise = var(noise);
    power = amplitude*amplitude/2;
    sqnr(i) = power/rmsofnoise;
end
sqnrTheoretical= levels.*levels;
sqnrTheoretical = sqnrTheoretical*(3/2);
sqnrPractical = sqnr;

plot(levels, sqnrPractical, levels, sqnrTheoretical);
grid on;
```

Compute the Signal to quantization Noise ratio of Non-Uniform Quantization. Plot SNQR versus Quantization levels.

Using A law

```
clc;
clear;
vin=5;
sampled_signal=2*vin*rand(1,100)-vin;
A=50;
vmax=norm(sampled_signal);
for i=1:100
if sampled_signal(i)/vmax<1/A
companded_signal(i)=(A*vin/vmax)/(1+log(A));
else
companded_signal(i)=(1+log(A*vin/vmax))/(1+log(A));
end
end
signal_pow=(norm(companded_signal)^2)/length(companded_signal);
subplot(2,1,1);
plot(companded_signal);
noise_pow=zeros(1,200);
SNQR=zeros(1,200);
for j=1:200
levels=j;
step_size=2*A/levels;
noise_pow(j)=(step_size^2)/12;
SNQR(j)=signal_pow/noise_pow(j);
end
subplot(2,1,2);
plot(SNQR)
```

Using u law

```
clc;
clear all;
A=5;
Sampled_sgl=2*A*rand(1,100)-A;

%MU-law companding
u=255; %compression constant
for i=1:100
  if Sampled_sgl(i)>0
    companded_sgl(i)=A*log(1+u*(Sampled_sgl(i)/A))/log(1+u);
  else
    companded_sgl(i)=A*log(1-u*(Sampled_sgl(i)/A))/log(1+u);
  end
end
sgl_pow=(norm(companded_sgl)^2)/length(companded_sgl);
subplot(2,1,1);
plot(companded_sgl);
noise_pow=zeros(1,200);
SNQR=zeros(1,200);
for j=1:200 %Number of levels
  levels=j;
  step_size=2*A/levels;
  noise_pow(j)=(step_size^2)/12;
  SNQR(j)=sgl_pow/noise_pow(j);
end

subplot(2,1,2);
plot(SNQR);
```

Yeh mila tumhe matlab chude tum

Bhagwan bachye tumhe/mujhe

Study of passband digital communication technique BPSK. Calculate the BER of BPSK modulated signal.

Method1: preferable

```
clc;
clear;

N=100000;
m = randi([0,1], 1, N);

%mapping
for i = 1:N
    if m(i)==0
        x(i)=-1;
    else
        x(i)=1;
    end
end

ber_sim=[];
ber_theory =[];
%adding noise
for EbN0dB = 0:1:15
EbN0 = db2pow(EbN0dB);
sigma = sqrt(1/(2*EbN0));
r=x + sigma.*randn(1,N);
m_cap = (r>0);

%number of errors
noe = sum(m~=m_cap);
ber_sim1 = noe/N;
ber_sim = [ber_sim ber_sim1];
ber_theory1 = 0.5*erfc(sqrt(EbN0));
ber_theory = [ber_theory ber_theory1];
end

EbN0dB = 0:1:15;
plot(EbN0dB, ber_sim , EbN0dB, ber_theory);
```

Method2:

```
x = randi(2, 1, 10000);
x(x == 2) = -1;
snr =[0:15];
ber = zeros(1, length(snr));
for i = 1:length(snr)
  y = awgn(x, snr(i));
  for j = 1:length(x)
    if x(j) > 0 && y(j) <= 0
      ber(i) = ber(i) + 1;
    end
    if x(j) <= 0 && y(j) > 0
      ber(i) = ber(i) + 1;
    end
  end
end
figure;
plot(snr, ber);
xlabel('SNR');
ylabel('Bit error rate');
title('BER Simulation for BPSK');
```

Run at: Add white Gaussian noise to signal - MATLAB awgn - MathWorks India

Calculate the number of valid code words N and the code rate RC. Specify the complete Code set C.

```
% Generator Matrix
G = [
     1 1 0 0 1 0 1;
     0 1 1 1 1 0 0;
     1 1 1 0 0 1 1;
     ];

N = 7; % total bits
K = 3; % no of message bits

fprintf("Code rate is %d\n", K/N);

possibleCodes = [
            0 0 0;
            0 0 1;
            0 1 0;
            0 1 1;
            1 0 0;
            1 0 1;
            1 1 0;
            1 1 1;
            ];
codeSet = [];
for i=1:8
  code = possibleCodes(i,:);
  codeSet = [codeSet;mod(code*G,2)];
end
fprintf("\nCode set : \n");
disp(codeSet)
```

Determine the generator matrix G’ of the appropriate systematic (separable) code C’.

```
clc;
clear;
G = [ 1 1 0 0 1 0 1;
      0 1 1 1 1 0 0;
      1 1 1 0 0 1 1;
      ];
K=3;
N=7;

% Generating appropriate systematic code

G(3,:) = mod(G(3,:) + G(1,:),2);
G(2,:) = mod(G(3,:) + G(2,:),2);
G(1,:) = mod(G(2,:) + G(1,:),2);
fprintf("Generator matrix :\n");
disp(G)
possibleCodes = [
            0 0 0;
            0 0 1;
            0 1 0;
            0 1 1;
            1 0 0;
            1 0 1;
            1 1 0;
            1 1 1;
            ];
codeSet = [];
for i=1:8
  code = possibleCodes(i,:);
  codeSet = [codeSet;mod(code*G,2)];
end
fprintf("\nCode set :\n");
disp(codeSet)
```

. Determine the syndrome table for single error.

```
clc;
clear;
G = [1 1 0 0 1 0 1; 0 1 1 1 1 0 0; 1 1 1 0 0 1 1];
G(3,:) = mod(G(3,:) + G(1,:),2);
G(2,:) = mod(G(3,:) + G(2,:),2);
G(1,:) = mod(G(2,:) + G(1,:),2);
G;
P = G(1:3, 4:7);
Ht = P';

Ht;
Ht = [Ht; [1 0 0]];
Ht = [Ht;[0 1 0]];
Ht = [Ht; [0 0 1]];
disp("H transpose obtained");
Ht
error = zeros(1, 7);
iden = eye(7, 7);
error = [error; iden];

syndrome = [0 0 0];
syndrome = [syndrome; Ht];
disp("syndrome");
syndrome
disp("error pattern");
error
```