1. **Tossing coin:**

Code 1:

```
clc;
clear all;
close all;
n = input('enter n: ');
results = randi([0, 1], 1, n);
outcome = ' ';
for i = 1:n
    if results(i) == 0
        disp('Toss Result: Heads');
        outcome = strcat(outcome, 'H');
    else
        disp('Toss Result: Tails');
        outcome = strcat(outcome, 'T');
    end
end
tt_outcomes = 2^n;
prob = 1 / tt_outcomes;
fprintf('Outcome: %s\n', outcome);
fprintf('Probability of this outcome: %.2f (1/%d)\n', prob,
tt_outcomes);
```

Code 2:

```
clc;
close all;
clear all;
count_heads=0;
count_tails=0;
n=input("Enter the number of tosses: ");
  for i=1:n
    if rand>0.5
        count_heads=count_heads+1;
        fprintf('Heads\n')
    else
        count_tails=count_tails+1;
        fprintf('Tails\n')
    end
  end
probability_heads=count_heads/n;
probability_tails=count_tails/n;
fprintf("The probability of heads is :%f\n",probability_heads);
fprintf("The probability of tails is :%f\n",probability_tails);
probability_atleast1tail=1-(0.5)^n;
probability_atleast1head=1-(0.5)^n;
```

```
fprintf("The probability of atleast 1 head is
:%f\n",probability_atleast1head);
fprintf("The probability of atleast 1 tail is
:%f\n",probability_atleast1tail);

categories = {'Heads', 'Tails'};
counts = [count_heads, count_tails];
figure;
bar(categories, counts);
title('Number of Heads and Tails');
xlabel('Outcome');
ylabel('Count');
grid on;
```

Rolling dice:

Code 1:

```
clc;
close all;
clear all;
r1 = randi([1,10],1,1);
r2 = randi ([1,6],1,1);
indprob = 1./6;
ttprob = (indprob).^r1;
disp(ttprob);
```

Code 2:

```
% Parameters
num_trials = 10000;  % Number of trials for each experiment

% Coin Toss Experiment
coin_toss = rand(1, num_trials) > 0.5;  % 1 for heads, 0 for tails
heads_count = sum(coin_toss == 1);  % Count of heads
tails_count = num_trials - heads_count;  % Count of tails

% Die Roll Experiment
die_roll = randi([1, 6], 1, num_trials);  % Random die rolls (1-6)
outcome_counts = histcounts(die_roll, 1:7);  % Count of each outcome
(1-6)

% Experimental Probabilities
P_heads = heads_count / num_trials;
P_tails = tails_count / num_trials;

P_die = outcome_counts / num_trials;

% Plot Coin Toss Results
figure;
subplot(1, 2, 1);
```

```
bar([P_heads, P_tails]);
set(gca, 'xticklabel', {'Heads', 'Tails'});
xlabel('Outcome');
ylabel('Experimental Probability');
title('Coin Toss Probabilities');
ylim([0 1]);

% Plot Die Roll Results
subplot(1, 2, 2);
bar(1:6, P_die);
xlabel('Die Outcome');
ylabel('Experimental Probability');
title('Die Roll Probabilities');
ylim([0 1]);

% Display Theoretical Probabilities
disp(['Theoretical Probability for Heads: ', num2str(1/2)]);
disp(['Theoretical Probability for Tails: ', num2str(1/2)]);
disp('Theoretical Probability for each Die outcome: 1/6');
```

2. Gaussian distribution:

Code 1:

```
clc;
clear all;
close all;
x = -10:0.1:10;

u = 0 ; sigma = 1;
f1 = (1/(sqrt(2.*pi.*sigma.^2))).*exp(-0.5*((x-u).^2)/sigma.^2);

u=1; sigma = 1;
f2 = (1/(sqrt(2.*pi.*sigma.^2))).*exp(-0.5*((x-u).^2)/sigma.^2);

figure; plot(x,f1);
figure;plot(x,f2);
```

Code 2:

```
clc;
clear all;
close all;


x = -10:0.1:10;
t = -10:0.1:10;
x_shifted = x + cos(50*t);
```

```
u = 0; sigma = 1;
f1 = (1 / (sqrt(2 * pi * sigma^2))) .* exp(-0.5 * ((x - u).^2) /
sigma^2);

u = 0; sigma = 1;
f2 = (1 / (sqrt(2 * pi * sigma^2))) .* exp(-0.5 * ((x_shifted -
u).^2) / sigma^2);

u = 1; sigma = 1;
f3 = (1/(sqrt(2.*pi.*sigma.^2))).*exp(-0.5*((x-u).^2)/sigma.^2);

u = 1; sigma = 1;
f4 = (1/(sqrt(2.*pi.*sigma.^2))).*exp(-0.5*((x_shifted-
u).^2)/sigma.^2);

figure;
plot(x, f1, 'r-', 'LineWidth', 2);
hold on;
plot(x, f2, 'g-', 'LineWidth', 2);
title('Original and Shifted Gaussian Distributions');
xlabel('Value');
ylabel('Probability Density');
legend('Original Gaussian', 'Shifted Gaussian');
grid on;

figure;
plot(x, f3, 'r-', 'LineWidth', 2);
hold on;
plot(x, f4, 'g-', 'LineWidth', 2);
title('Original and Shifted Gaussian Distributions');
xlabel('Value');
ylabel('Probability Density');
legend('Original Gaussian', 'Shifted Gaussian');
grid on;
```

Code 3:

```
clc;
clear all;
close all;

n_samples = 100;
t = 0:n_samples-1;

num_signals = input('Enter the number of signals to combine: ');
signal = zeros(size(t));


for i = 1:num_signals
```

```
    A = input(['Enter amplitude A' num2str(i) ': ']);
    w = input(['Enter frequency w' num2str(i) ': ']);
    signal = signal + A * cos(w * t);
end


figure;
stem(t, signal, 'LineWidth', 1.5, 'MarkerFaceColor', 'r');
title('Combined Signal (Discrete)');
xlabel('Discrete Time (n)');
ylabel('Amplitude');
grid on;


unique_values = unique(round(signal, 2));
counts = histc(signal, unique_values);
probabilities = counts / sum(counts);


figure;
bar(unique_values, probabilities, 'FaceColor', [0.2 0.6 0.8]);
title('Probability Distribution of Signal Values');
xlabel('Unique Signal Values');
ylabel('Probability');
grid on;


disp('Unique Values and Their Probabilities:');
disp(table(unique_values', probabilities', 'VariableNames',
{'Value', 'Probability'}));
```

- **audio analysis:**

```
% Define the directory of the audio file
audio_file = 'C:\Users\K.Nihitha\Downloads\ptsp1.m4a';

% Read the audio file
[audio_signal, Fs] = audioread(audio_file); % Fs is the sampling
frequency
audio_signal = audio_signal(:, 1); % If stereo, take the first
channel

% Frame parameters
frame_duration = 0.5; % Frame duration in seconds
samples_per_frame = round(frame_duration * Fs);
num_frames = ceil(length(audio_signal) / samples_per_frame);

% Initialize for entire signal probability calculation
```

```
all_extreme_vals = [];

% Loop through each frame
for i = 1:num_frames
    % Extract the current frame
    start_idx = (i - 1) * samples_per_frame + 1;
    end_idx = min(i * samples_per_frame, length(audio_signal));
    frame = audio_signal(start_idx:end_idx);

    % Create a rectangular signal of amplitude 1
    rect_signal = ones(size(frame));
    fitted_signal = frame .* rect_signal;

    % Find maximas and minimas
    [max_vals, max_locs] = findpeaks(fitted_signal); % Maximas
    [min_vals, min_locs] = findpeaks(-fitted_signal); % Minimas
(inverted)
    min_vals = -min_vals; % Correct inversion
    all_extreme_vals = [all_extreme_vals; max_vals; min_vals]; %
Collect for whole signal

    % Probability calculation for this frame
    unique_vals = unique([max_vals; min_vals]);
    probabilities = zeros(size(unique_vals));
    for j = 1:length(unique_vals)
        probabilities(j) = sum([max_vals; min_vals] ==
unique_vals(j)) / length([max_vals; min_vals]);
    end

    % Create a figure for this frame
    figure;
    sgtitle(['Frame ', num2str(i), ' Analysis']);

    % Subplot 1: Entire audio signal
    subplot(5, 1, 1);
    time = (0:length(audio_signal) - 1) / Fs; % Time for the entire
audio signal
    plot(time, audio_signal);
    hold on;
    % Highlight the current frame range
    frame_time = [(start_idx - 1) / Fs, end_idx / Fs];
    y_limits = ylim;
    fill([frame_time(1), frame_time(2), frame_time(2),
frame_time(1)], ...
         [y_limits(1), y_limits(1), y_limits(2), y_limits(2)], ...
         'yellow', 'FaceAlpha', 0.3, 'EdgeColor', 'none');
    xlabel('Time (s)');
    ylabel('Amplitude');
    title('Entire Audio Signal with Highlighted Frame');
```

```
    grid on;

    % Subplot 2: Framed signal
    subplot(5, 1, 2);
    time_frame = (start_idx:end_idx) / Fs; % Time for the current
frame
    plot(time_frame, frame);
    xlabel('Time (s)');
    ylabel('Amplitude');
    title('Framed Signal');
    grid on;

    % Subplot 3: Rectangular framed signal
    subplot(5, 1, 3);
    plot(time_frame, fitted_signal);
    xlabel('Time (s)');
    ylabel('Amplitude');
    title('Rectangular Framed Signal');
    grid on;

    % Subplot 4: Signal with maximas and minimas
    subplot(5, 1, 4);
    plot(time_frame, fitted_signal);
    hold on;
    plot((max_locs + start_idx - 1) / Fs, max_vals, 'ro',
'MarkerFaceColor', 'r');
    plot((min_locs + start_idx - 1) / Fs, min_vals, 'bo',
'MarkerFaceColor', 'b');
    xlabel('Time (s)');
    ylabel('Amplitude');
    title('Maximas and Minimas');
    legend('Signal', 'Maxima', 'Minima');
    grid on;

    % Subplot 5: Probability curve
    subplot(5, 1, 5);
    stem(unique_vals, probabilities, 'MarkerFaceColor', 'g');
    xlabel('Amplitude Value');
    ylabel('Probability');
    title('Probability Curve');
    grid on;
end

% Final: Probability curve for the entire signal
unique_vals_full = unique(all_extreme_vals);
probabilities_full = zeros(size(unique_vals_full));
for j = 1:length(unique_vals_full)
    probabilities_full(j) = sum(all_extreme_vals ==
unique_vals_full(j)) / length(all_extreme_vals);
```

```
end

% Plot probability curve for the entire signal
figure;
stem(unique_vals_full, probabilities_full, 'MarkerFaceColor', 'g');
xlabel('Amplitude Value');
ylabel('Probability');
title('Overall Probability Curve for Entire Signal');
grid on;
```

- **mean and variance of all distributions:**

```
clc;
clear all;
close all;
N = 10000; % Number of samples

%% Gaussian (Normal) Distribution
m1 = 5; % Mean
var1 = 2; % Variance
gaussian_samples = m1 + sqrt(var1) * randn(1, N);
gass_pdf = (1 / sqrt(2 * pi * var1)) * exp(-((gaussian_samples - m1).^2) / (2 *
var1));

mean_sum = 0;
for i = 1:N
    mean_sum = mean_sum + (gaussian_samples(i) * gass_pdf(i));
end
mean_gaussian = mean_sum / sum(gass_pdf);

variance_sum = 0;
for i = 1:N
    variance_sum = variance_sum + ((gaussian_samples(i).^2) * gass_pdf(i));
end
variance_gaussian = variance_sum / sum(gass_pdf) - mean_gaussian^2;

fprintf('Gaussian Distribution:\n');
fprintf('Estimated Mean: %.4f, Theoretical Mean: %.4f\n', mean_gaussian, m1);
fprintf('Estimated Variance: %.4f, Theoretical Variance: %.4f\n\n',
variance_gaussian, var1);

figure;
histogram(gaussian_samples, 50, 'Normalization', 'pdf');
title('Gaussian Distribution');
xlabel('Value'); ylabel('Probability Density');
grid on;

%% Uniform Distribution
a = 2; b = 8;
uniform_samples = a + (b-a) * rand(1, N);
uni_pdf = ones(1, N) / (b-a);

mean_sum = 0;
for i = 1:N
    mean_sum = mean_sum + (uniform_samples(i) * uni_pdf(i));
```

```matlab
end
mean_uniform = mean_sum / sum(uni_pdf);

variance_sum = 0;
for i = 1:N
    variance_sum = variance_sum + ((uniform_samples(i).^2) * uni_pdf(i));
end
variance_uniform = variance_sum / sum(uni_pdf) - mean_uniform^2;

fprintf('Uniform Distribution:\n');
fprintf('Estimated Mean: %.4f, Theoretical Mean: %.4f\n', mean_uniform, (a + b) /
2);
fprintf('Estimated Variance: %.4f, Theoretical Variance: %.4f\n\n',
variance_uniform, (b - a)^2 / 12);

figure;
histogram(uniform_samples, 50, 'Normalization', 'pdf');
title('Uniform Distribution');
xlabel('Value'); ylabel('Probability Density');
grid on;

%% Exponential Distribution
lambda = 1.5;
exponential_samples = -log(rand(1, N)) / lambda;
exp_pdf = lambda * exp(-lambda * exponential_samples);

mean_sum = 0;
for i = 1:N
    mean_sum = mean_sum + (exponential_samples(i) * exp_pdf(i));
end
mean_exponential = mean_sum / sum(exp_pdf);

variance_sum = 0;
for i = 1:N
    variance_sum = variance_sum + ((exponential_samples(i).^2) * exp_pdf(i));
end
variance_exponential = variance_sum / sum(exp_pdf) - mean_exponential^2;

fprintf('Exponential Distribution:\n');
fprintf('Estimated Mean: %.4f, Theoretical Mean: %.4f\n', mean_exponential, 1 /
lambda);
fprintf('Estimated Variance: %.4f, Theoretical Variance: %.4f\n\n',
variance_exponential, 1 / lambda^2);

figure;
histogram(exponential_samples, 50, 'Normalization', 'pdf');
title('Exponential Distribution');
xlabel('Value'); ylabel('Probability Density');
grid on;
```

- **mean and variance of y = 2x+1**:

```
clc;
clear;
close all;

% User Inputs
m1 = 0; % Mean for Case 1
m2 = input('Select any number for mean (Case 2): ');
var1 = input('Select any number for variance (Case 1): ');
var2 = input('Select any number for variance (Case 2): ');

% Number of samples
num_samples = 10000;

%% Generate Gaussian Samples for Both Cases
X1 = m1 + sqrt(var1) * randn(num_samples,1);  % Case 1: Mean = 0, User-defined
Variance
X2 = m2 + sqrt(var2) * randn(num_samples,1);  % Case 2: User-defined Mean &
Variance

Y1 = 2 * X1 + 1; % Transformation Y = 2X + 1
Y2 = 2 * X2 + 1;

% Compute PDFs
fX1 = (1 / sqrt(2 * pi * var1)) * exp(-((X1 - m1).^2) / (2 * var1));
fX2 = (1 / sqrt(2 * pi * var2)) * exp(-((X2 - m2).^2) / (2 * var2));

fY1 = (1 / sqrt(2 * pi * (4 * var1))) * exp(-((Y1 - (2 * m1 + 1)).^2) / (2 * (4 *
var1)));
fY2 = (1 / sqrt(2 * pi * (4 * var2))) * exp(-((Y2 - (2 * m2 + 1)).^2) / (2 * (4 *
var2)));

% Compute CDF
[cdf_Y1, y_bins1] = hist(Y1, 50);
cdf_Y1 = cumsum(cdf_Y1) / num_samples;

[cdf_Y2, y_bins2] = hist(Y2, 50);
cdf_Y2 = cumsum(cdf_Y2) / num_samples;

%% Plot all graphs
figure;

% Case 1: PDF of X
subplot(3,2,1);
histogram(X1, 50, 'Normalization', 'pdf');
xlabel('X values'); ylabel('Density f_X(X)');
title(sprintf('Case 1: PDF of X (Mean=0, Variance=%d)', var1));
grid on;

% Case 1: PDF of Y
subplot(3,2,3);
histogram(Y1, 50, 'Normalization', 'pdf');
xlabel('Y values'); ylabel('Density f_Y(Y)');
title(sprintf('Case 1: PDF of Y (Variance=%d)', var1));
grid on;

% Case 1: CDF of Y
subplot(3,2,5);
```

```
plot(y_bins1, cdf_Y1, 'r', 'LineWidth', 2);
xlabel('Y values'); ylabel('F_Y(Y)');
title('Case 1: CDF of Y');
grid on;

% Case 2: PDF of X
subplot(3,2,2);
histogram(X2, 50, 'Normalization', 'pdf');
xlabel('X values'); ylabel('Density f_X(X)');
title(sprintf('Case 2: PDF of X (Mean=%d, Variance=%d)', m2, var2));
grid on;

% Case 2: PDF of Y
subplot(3,2,4);
histogram(Y2, 50, 'Normalization', 'pdf');
xlabel('Y values'); ylabel('Density f_Y(Y)');
title(sprintf('Case 2: PDF of Y (Mean=%d, Variance=%d)', m2, var2));
grid on;

% Case 2: CDF of Y
subplot(3,2,6);
plot(y_bins2, cdf_Y2, 'r', 'LineWidth', 2);
xlabel('Y values'); ylabel('F_Y(Y)');
title('Case 2: CDF of Y');
grid on;
```

- **pdf of x and y:**

```
clc;
clear all;
```

- `close all;`

```
N = 1000;
%% Gaussian Distribution
X = randn(N);

minmax_values = minmax(X);
X_min = minmax_values(1);
X_max = minmax_values(2);

m1 = (X_min + X_max) / 2;
var1 = ((X_max - X_min)^2) / 12;

disp(['Mean: ', num2str(m1)]);
disp(['Variance: ', num2str(var1)]);

X1 = m1 + sqrt(var1) * randn(N,1);
Y1 = 2 * X1 + 1;
x_range = linspace(min(X1), max(X1), 1000);
pdf_X1 = (1 / sqrt(2 * pi * var1)) * exp(-((x_range - m1).^2) / (2 *
var1));

y_range = linspace(min(Y1), max(Y1), 1000);
```

```
pdf_Y1 = (1 / sqrt(2 * pi * (4 * var1))) * exp(-((y_range - (2 * m1 +
1)).^2) / (2 * (4 * var1)));

subplot(3,2,1)
histogram(X1, 50, 'Normalization', 'pdf');
hold on;
plot(x_range, pdf_X1, 'r', 'LineWidth', 2);
xlabel('x values'); ylabel('f(x)');title('PDF of X');
grid on;

subplot(3,2,2)
histogram(Y1, 50, 'Normalization', 'pdf');
hold on;
plot(y_range, pdf_Y1, 'r', 'LineWidth', 2);
xlabel('y values'); ylabel('f(y)');title('PDF of Y');
grid on;

%% Uniform Distribution
a = -1; b = 1;
X_unif = a + (b-a) * rand(N,1);
m_unif = mean(X_unif);
var_unif = var(X_unif);

Y_unif = 2 * X_unif + 1;

x_range = linspace(min(X_unif), max(X_unif), 1000);
pdf_X_unif = ones(size(x_range)) / (b-a);

y_range = linspace(min(Y_unif), max(Y_unif), 1000);
pdf_Y_unif = ones(size(y_range)) / (2*(b-a));

subplot(3,2,3)
histogram(X_unif, 50, 'Normalization', 'pdf');
hold on;
plot(x_range, pdf_X_unif, 'r', 'LineWidth', 2);
xlabel('x values'); ylabel('f(x)'); title('PDF of X (Uniform)');
grid on;

subplot(3,2,4)
histogram(Y_unif, 50, 'Normalization', 'pdf');
hold on;
plot(y_range, pdf_Y_unif, 'r', 'LineWidth', 2);
xlabel('y values'); ylabel('f(y)'); title('PDF of Y (Uniform)');
grid on;

%% Exponential Distribution
lambda = 1;
X_exp = exprnd(1/lambda, N, 1);
m_exp = mean(X_exp);
var_exp = var(X_exp);

Y_exp = 2 * X_exp + 1;

x_range = linspace(min(X_exp), max(X_exp), 1000);
```

```
pdf_X_exp = lambda * exp(-lambda * x_range) .* (x_range >= 0);

y_range = linspace(min(Y_exp), max(Y_exp), 1000);
pdf_Y_exp = (lambda / 2) * exp(-lambda * (y_range - 1) / 2) .* (y_range >=
1);

subplot(3,2,5)
histogram(X_exp, 50, 'Normalization', 'pdf');
hold on;
plot(x_range, pdf_X_exp, 'r', 'LineWidth', 2);
xlabel('x values'); ylabel('f(x)'); title('PDF of X (Exponential)');
grid on;

subplot(3,2,6)
histogram(Y_exp, 50, 'Normalization', 'pdf');
hold on;
plot(y_range, pdf_Y_exp, 'r', 'LineWidth', 2);
xlabel('y values'); ylabel('f(y)'); title('PDF of Y (Exponential)');
grid on;
```

- **pdf and cdf of 2 gaussians:**

```
clc;
clear;
close all;

n = 1000;
X = randn(n);
Y = randn(n);

minmax_values = minmax(X);
X_min = minmax_values(1);
X_max = minmax_values(2);

mx = (X_min + X_max) / 2;
varx = ((X_max - X_min)^2) / 12;

minmax_values = minmax(Y);
Y_min = minmax_values(1);
Y_max = minmax_values(2);

my = (Y_min + Y_max) / 2;
vary = ((Y_max - Y_min)^2) / 12;

X = mx + sqrt(varx) * randn(n,1);
x_range = linspace(min(X), max(X), 1000);
fX = (1 / sqrt(2 * pi * varx)) * exp(-((x_range - mx).^2) / (2 * varx));

Y = my + sqrt(vary) * randn(n,1);
y_range = linspace(min(Y), max(Y), 1000);
fY = (1 / sqrt(2 * pi * vary)) * exp(-((y_range - my).^2) / (2 * vary));

[cdf_X, x_bins] = hist(X, 50);
```

```
cdf_X = cumsum(cdf_X) / n;

[cdf_Y, y_bins] = hist(Y, 50);
cdf_Y = cumsum(cdf_Y) / n;

% case 1: x and y are statistically independent

mz = mx + my;
varz1 = varx + vary;
Z = mz + sqrt(varz1) * randn(n,1);
z_range = linspace(min(Z), max(Z), 1000);
fZ = conv(fX , fY,'same');

fZ = fZ / trapz(z_range, fZ); % Normalize fZ to preserve probability
density

[cdf_Z, z_bins] = hist(Z, 50);
cdf_Z = cumsum(cdf_Z) / n;

subplot(2,4,1)
histogram(X, 50, 'Normalization', 'pdf');
hold on;
plot(x_range, fX, 'r', 'LineWidth', 2);
xlabel('x values'); ylabel('f(x)');title('PDF of X');
grid on;

subplot(2,4,2)
histogram(Y, 50, 'Normalization', 'pdf');
hold on;
plot(y_range, fY, 'r', 'LineWidth', 2);
xlabel('y values'); ylabel('f(y)');title('PDF of Y');
grid on;

subplot(2,4,3)
histogram(Z, 50, 'Normalization', 'pdf');
hold on;
plot(z_range ,fZ, 'r', 'LineWidth', 2);
xlabel('z values'); ylabel('f(z)');title('PDF of Z(independent case)');
grid on;

subplot(2,4,5);
histogram(X, 50, 'Normalization', 'cdf');
hold on;
plot(x_bins, cdf_X, 'r', 'LineWidth', 2);
xlabel('X values'); ylabel('F_X(X)');
title('CDF of X');
grid on;

subplot(2,4,6);
histogram(Y, 50, 'Normalization', 'cdf');
hold on;
plot(y_bins, cdf_Y, 'r', 'LineWidth', 2);
xlabel('Y values'); ylabel('F_Y(Y)');
title('CDF of Y');
```

```
grid on;

subplot(2,4,7);
histogram(Z, 50, 'Normalization', 'cdf');
hold on;
plot(z_bins, cdf_Z, 'r', 'LineWidth', 2);
xlabel('Z values'); ylabel('F_Z(Z)');
title('CDF of Z(independent case)');
grid on;

%case 2: x and y are dependent

R = corrcoef(X, Y);
corr_XY = R(1,2);

mz = mx + my;
varz2 = varx + vary + 2*corr_XY*sqrt(varx * vary);
Z2 = mz + sqrt(varz2) * randn(n,1);
z_range2 = linspace(min(Z2), max(Z2), 1000);
fZ2 = (1 / sqrt(2 * pi * varz2)) * exp(-((z_range2 - mz).^2) / (2 *
varz2));

[cdf_Z2, z_bins2] = hist(Z2, 50);
cdf_Z2 = cumsum(cdf_Z2) / n;

subplot(2,4,4)
histogram(Z2, 50, 'Normalization', 'pdf');
hold on;
plot(z_range2 ,fZ2, 'r', 'LineWidth', 2);
xlabel('z values'); ylabel('f(z)');title('PDF of Z (dependent case)');
grid on;

subplot(2,4,8);
histogram(Z2, 50, 'Normalization', 'cdf');
hold on;
plot(z_bins2, cdf_Z2, 'r', 'LineWidth', 2);
xlabel('Z values'); ylabel('F_Z(Z)');
title('CDF of Z(dependent case)');
grid on;
```

- **mean and variance of z = x+y:**

```
clear;
close all;

n = 1000;
X = randn(n);
Y = randn(n);

minmax_values = minmax(X);
X_min = minmax_values(1);
```

```
X_max = minmax_values(2);

mx = (X_min + X_max) / 2;
varx = ((X_max - X_min)^2) / 12;

minmax_values = minmax(Y);
Y_min = minmax_values(1);
Y_max = minmax_values(2);

my = (Y_min + Y_max) / 2;
vary = ((Y_max - Y_min)^2) / 12;

X = mx + sqrt(varx) * randn(n,1);
x_range = linspace(min(X), max(X), 1000);
fX = (1 / sqrt(2 * pi * varx)) * exp(-((x_range - mx).^2) / (2 * varx));

Y = my + sqrt(vary) * randn(n,1);
y_range = linspace(min(Y), max(Y), 1000);
fY = (1 / sqrt(2 * pi * vary)) * exp(-((y_range - my).^2) / (2 * vary));

% case 1: x and y are statistically independent

mz = mx + my;
varz1 = varx + vary;
Z = mz + sqrt(varz1) * randn(n,1);
z_range = linspace(min(Z), max(Z), 1000);
fZ = conv(fX , fY,'same');

fZ = fZ / trapz(z_range, fZ);

%case 2: x and y are dependent

R = corrcoef(X, Y);
corr_XY = R(1,2);

mz = mx + my;
varz2 = varx + vary + 2*corr_XY*sqrt(varx * vary);
Z2 = mz + sqrt(varz2) * randn(n,1);
z_range2 = linspace(min(Z2), max(Z2), 1000);
fZ2 = (1 / sqrt(2 * pi * varz2)) * exp(-((z_range2 - mz).^2) / (2 *
varz2));

disp(['mz: ', num2str(mz)]);
disp(['varz1(independent case): ', num2str(varz1)]);
disp(['varz2(dependent case): ', num2str(varz2)]);

subplot(2,1,1)
histogram(Z, 50, 'Normalization', 'pdf');
hold on;
plot(z_range ,fZ, 'r', 'LineWidth', 2);
xlabel('z values'); ylabel('f(z)');title('PDF of Z(independent case)');
grid on;

subplot(2,1,2)
```

```
histogram(Z2, 50, 'Normalization', 'pdf');
hold on;
plot(z_range2 ,fZ2, 'r', 'LineWidth', 2);
xlabel('z values'); ylabel('f(z)');title('PDF of Z (dependent case)');
grid on;
```

- **pdf of z = x+y:**

- **(uniform + uniform) and (gaussian + uniform):**

```
clc;
clear all;
close all;

n = 1000;

X = rand(n,1);

sum_X = 0;
for i = 1:n
    sum_X = sum_X + X(i);
en
mx = sum_X / n;

sum_squared_diff = 0;
for i = 1:n
    sum_squared_diff = sum_squared_diff + (X(i) – mx)^2;
end
varx = sum_squared_diff / n;

disp([‘Estimated Mean: ‘, num2str(mx)]);
disp([‘Estimated Variance: ‘, num2str(varx)]);

Y = rand(n,1);

sum_Y = 0;
for i = 1:n
    sum_Y = sum_Y + Y(i);
end
my = sum_Y / n;

sum_squared_diff = 0;
for i = 1:n
    sum_squared_diff = sum_squared_diff + (Y(i) – my)^2;
end
vary = sum_squared_diff / n;
```

```
disp(['Estimated Mean: ', num2str(my)]);
disp(['Estimated Variance: ', num2str(vary)]);

% case 1: x and y are statistically independent

mz = mx + my;
varz1 = varx + vary;

disp(['mz: ', num2str(mz)]);
disp(['varz1(independent case): ', num2str(varz1)]);
Z = X + Y;
figure;
histogram(Z, 200, 'Normalization', 'pdf');  % 30 bins, normalized
title('Histogram of Z = X + Y (Both Uniform)');
xlabel('Z values');
ylabel('Probability Density');

%------------------- X1 is guassian -------------------%

X1 = randn(n,1);

mx1 = sum(X1) / n;
varx1 = sum((X1 - mx1).^2) / n;

% case 1: x and y are statistically independent

mz = mx1 + my;
varz1 = varx1 + vary;

disp(['mz: ', num2str(mz)]);
disp(['varz1(independent case): ', num2str(varz1)]);


Z1 = X1 + Y;
figure;
histogram(Z1, 200, 'Normalization', 'pdf');  % 30 bins, normalized
title('Histogram of Z = X(guassian) + Y(uniform) ');
xlabel('Z values');
ylabel('Probability Density');
```

- **X and y are circular symmetry:**

```
clc;
clear;
close all;

sigma2 = input('Enter the variance of X and Y: ');
sigma = sqrt(sigma2);

N = 10^6;

X = sigma * randn(N, 1);
Y = sigma * randn(N, 1);

Z = sqrt(X.^2 + Y.^2);


%   g(r) = (1 / (2*pi*simga^2)) * exp(-r^2 / 2)
%   f(z) = 2 * pi * g(z) * z
%   f(z) = 2 * pi * (1 / (2 * pi*simga^2)) * exp(-z^2 / 2) * z
%   f(z) = (z/sigma^2) * exp(-z^2 / 2)

z_range = linspace(0, max(Z), 1000);
f_Z = (z_range / sigma2) .* exp(-z_range.^2 / (2 * sigma2));

figure;
histogram(Z, 'Normalization', 'pdf', 'BinWidth', 0.1, 'FaceAlpha',
0.5);
hold on;
plot(z_range, f_Z, 'r', 'LineWidth', 2);
xlabel('Z = sqrt(X^2 + Y^2)');
ylabel('Density f(Z)');
title('Density Function and Histogram of Z');
legend('Histogram of Z', 'Theoretical Density f(Z)');
grid on;
```

- **Speech signal:**

```
clc;
clear;
close all;

% Load Speech Signal
[speech, fs] = audioread("C:\Users\K.Nihitha\Downloads\ptsp1.opus");
speech = speech(:,1); % Use first channel if stereo
```

```matlab
% Time Axis
t = (0:length(speech)-1) / fs;

% Plot the Speech Signal (Blue)
figure;
plot(t, speech, 'b', 'LineWidth', 1.2); % Blue color plot
xlabel('Time (s)'); ylabel('Amplitude');
title('Speech Signal in Time Domain');
grid on;
% Parameters
num_frames = 5; % Number of frames
total_samples = length(speech);
frame_size = floor(total_samples / num_frames);
frames = zeros(frame_size, num_frames);

% Split the signal into frames
for i = 1:num_frames
    start_idx = (i-1) * frame_size + 1;
    end_idx = i * frame_size;
    frames(:, i) = speech(start_idx:end_idx);
end

% Maximum lag for autocorrelation and covariance
max_lag = 50;

% Initialize matrices for results
autocorr_frames = zeros(max_lag + 1, num_frames);
covariance_frames = zeros(max_lag + 1, num_frames);

% Compute Autocorrelation and Covariance for each frame
for f = 1:num_frames
    frame = frames(:, f);
    N = length(frame);

    % Compute mean
    frame_mean = sum(frame) / N;

    % Compute Autocorrelation
    for k = 0:max_lag
        if k < N
            autocorr_frames(k+1, f) = sum(frame(1:end-k) .*
frame(k+1:end)) / (N - k);
        end
    end

    % Compute Covariance
    for k = 0:max_lag
        if k < N
```

```
            covariance_frames(k+1, f) = sum((frame(1:end-k) - 
frame_mean) .* (frame(k+1:end) - frame_mean)) / (N - k);
        end
    end
end

% ----- PLOT 1: Speech Signal Frames -----
figure;
for i = 1:num_frames
    subplot(ceil(num_frames/2), 2, i);
    t = (0:frame_size-1) / fs;
    plot(t, frames(:, i), 'LineWidth', 1.5);
    xlabel('Time (s)'); ylabel('Amplitude');
    title(['Frame ' num2str(i)]);
    grid on;
end

% ----- PLOT 2: Autocorrelation -----
figure;
for i = 1:num_frames
    subplot(ceil(num_frames/2), 2, i);
    lags = 0:max_lag;
    stem(lags, autocorr_frames(:, i), 'filled', 'MarkerFaceColor', 
'b');
    xlabel('Lag'); ylabel('Autocorrelation');
    title(['Autocorrelation - Frame ' num2str(i)]);
    grid on;
end

% ----- PLOT 3: Covariance -----
figure;
for i = 1:num_frames
    subplot(ceil(num_frames/2), 2, i);
    lags = 0:max_lag;
    stem(lags, covariance_frames(:, i), 'filled', 'MarkerFaceColor', 
'r');
    xlabel('Lag');  ylabel('Covariance');
    title(['Covariance - Frame ' num2str(i)]);
    grid on;
end
```

- **Mixed sine signal:**

```
clc;
clear;
close all;

fs = 8000; % Sampling Frequency
T = 2; % Duration (seconds)
t = 0:1/fs:T-1/fs; % Time axis

f1 = 300;  f2 = 600; f3 = 1200;

sine_signal = sin(2 * pi * f1 * t) + 0.5 * sin(2 * pi * f2 * t) +
0.3 * sin(2 * pi * f3 * t);

% Plot the Mixed Sine Signal
figure;
plot(t, sine_signal, 'b', 'LineWidth', 1.5);
xlabel('Time (s)'); ylabel('Amplitude');
title('Mixed Sine Signal');
grid on;

% Compute Autocorrelation
max_lag = 200; % Maximum lag for computation
N = length(sine_signal);
autocorr_signal = zeros(max_lag+1, 1);

for k = 0:max_lag
    if k < N
        autocorr_signal(k+1) = sum(sine_signal(1:end-k) .*
sine_signal(k+1:end)) / (N - k);
    end
end

% Compute Covariance
sine_mean = mean(sine_signal);
covariance_signal = zeros(max_lag+1, 1);

for k = 0:max_lag
    if k < N
        covariance_signal(k+1) = sum((sine_signal(1:end-k) -
sine_mean) .* (sine_signal(k+1:end) - sine_mean)) / (N - k);
    end
end

% Plot Autocorrelation
figure;
lags = 0:max_lag;
stem(lags, autocorr_signal, 'filled', 'MarkerFaceColor', 'b');
```

```
xlabel('Lag'); ylabel('Autocorrelation');
title('Autocorrelation of Mixed Sine Signal');
grid on;

% Plot Covariance
figure;
stem(lags, covariance_signal, 'filled', 'MarkerFaceColor', 'r');
xlabel('Lag'); ylabel('Covariance');
title('Covariance of Mixed Sine Signal');
grid on;
```