MATLAB Simulation Project

The Final MATLAB project

Probability Engineering

Date: Spring 1403

Master: PhD. A Zaimbashi

Written By:
Alireza Sotoodeh (401412056)

Shahid Bahonar
University of Kerman

CONTENTS

PART 1: MATLAB SCRIPT FOR CONVERGENCE ANALYSIS OF SAMPLE MEAN AND COVARIANCE

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%{
Author: Alireza Sotoodeh (Student ID: 401412056)
Subject: Convergence of Mean and Covariance
Description: Demonstrates convergence of sample mean and covariance to
             expected values as sample size (N) increases. Computes and
             plots mean and covariance for specific N values, and evaluates
             error convergence.
%}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

INITIALIZATION

This section clears the command window, closes all figures, and clears the workspace variables.

```
clc;        % Clear command window
close all;  % Close all figures
clear all;  % Clear workspace variables
```

PARAMETERS

Define the parameters for the simulation.

```
n = 2;                      % Dimension of the vector
number_of_samples = 10000;  % Default number of samples for mean and covariance computation
expected_mean = 0.5;        % Expected mean value
expected_covariance = 5.0;  % Expected covariance value
```

DEFINE N_VALUES

Specific values of N to evaluate for error convergence.

```
N_values = [number_of_samples, 1000, 100, 10, 1];
```

CREATE A FOLDER TO SAVE RESULTS IF IT DOESN'T EXIST

Check if the 'Results' folder exists; if not, create it.

```
result_folder = 'Results_Project_1';
if ~exist(result_folder, 'dir')
    mkdir(result_folder);
end
```

COMPUTE MEAN AND COVARIANCE

Initialize arrays to store mean and covariance values for different N values.

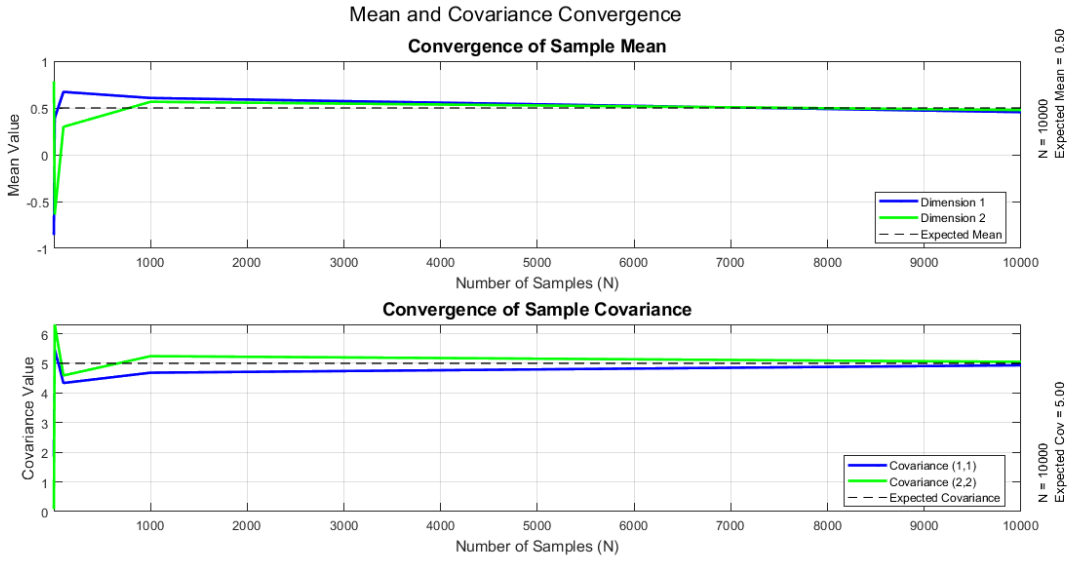
```
mean_values = zeros(n, length(N_values));
cov_values = zeros(n, n, length(N_values));

% Loop over each value of N in N_values to generate random vectors and compute sample mean and
covariance.
for j = 1:length(N_values)
    N = N_values(j);

    % Generate N random vectors with specified mean and covariance.
    random_vectors = expected_mean + sqrt(expected_covariance) * randn(n, N);

    % Compute sample mean.
    sample_mean = mean(random_vectors, 2);

    % Center the vectors around the expected_mean.
    centered_vectors = random_vectors - expected_mean * ones(n, N);

    % Compute sample covariance.
    sample_covariance = (centered_vectors * centered_vectors') / N;

    % Store mean and covariance.
    mean_values(:, j) = sample_mean;
    cov_values(:, :, j) = sample_covariance;
end
```

PLOT MEAN AND COVARIANCE CONVERGENCE

This section plots the convergence of sample mean and covariance values.

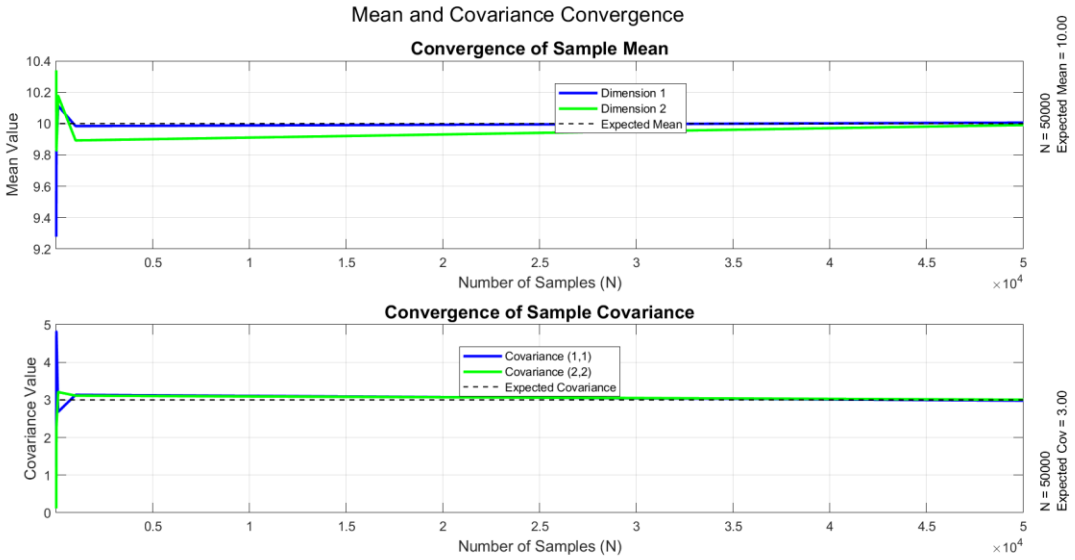
```
figure('WindowState', 'maximized'); % Open figure in fullscreen

% Plot convergence of mean values.
subplot(2, 1, 1);
plot(N_values, mean_values(1, :), 'b-', 'LineWidth', 2); % Blue for first dimension mean
hold on;
plot(N_values, mean_values(2, :), 'g-', 'LineWidth', 2); % Green for second dimension mean
plot(N_values, expected_mean * ones(size(N_values)), 'k--', 'LineWidth', 1); % Expected mean line
hold off;
xlabel('Number of Samples (N)', 'FontSize', 12);
ylabel('Mean Value', 'FontSize', 12);
title('Convergence of Sample Mean', 'FontSize', 14);
legend('Dimension 1', 'Dimension 2', 'Expected Mean', 'Location', 'best');
grid on;
xlim([min(N_values) max(N_values)]);

% Add vertical text annotations for mean.
mean_annotation_x = max(N_values) * 1.03;
```

```
mean_annotation_y = mean(mean_values(:, end));
mean_text = sprintf('N = %d\nExpected Mean = %.2f', number_of_samples(end), expected_mean);
text(mean_annotation_x, mean_annotation_y, mean_text, ...
    'FontSize', 10, 'VerticalAlignment', 'middle', 'HorizontalAlignment', 'left', 'Rotation',
90);

% Plot convergence of covariance values.
subplot(2, 1, 2);
plot(N_values, squeeze(cov_values(1, 1, :)), 'b-', 'LineWidth', 2); % Blue for (1,1) element
hold on;
plot(N_values, squeeze(cov_values(2, 2, :)), 'g-', 'LineWidth', 2); % Green for (2,2) element
plot(N_values, expected_covariance * ones(size(N_values)), 'k--', 'LineWidth', 1); % Expected
covariance line
hold off;
xlabel('Number of Samples (N)', 'FontSize', 12);
ylabel('Covariance Value', 'FontSize', 12);
title('Convergence of Sample Covariance', 'FontSize', 14);
legend('Covariance (1,1)', 'Covariance (2,2)', 'Expected Covariance', 'Location', 'best');
grid on;
xlim([min(N_values) max(N_values)]);

% Add vertical text annotations for covariance.
cov_annotation_x = max(N_values) * 1.03;
cov_annotation_y = mean(squeeze(cov_values(:, :, end)), 'all') ;
cov_text = sprintf('N = %d\nExpected Cov = %.2f', number_of_samples(end), expected_covariance);
text(cov_annotation_x, cov_annotation_y, cov_text, ...
    'FontSize', 10, 'VerticalAlignment', 'middle', 'HorizontalAlignment', 'left', 'Rotation',
90);

% Adjust subplot spacing.
sgtitle('Mean and Covariance Convergence', 'FontSize', 16); % Overall title for the figure

% Save figure in the Results folder as PNG.
saveas(gcf, fullfile(result_folder, 'mean_and_covariance_convergence.png'));
```



Also, in order to run other simulation, I've added a text on each graph to show its parameters!

You can see further test:



As the graphs show, by increasing number of sample the value of mean and covariance converge to expected value!

CALCULATE ERRORS

Calculate errors for selected N_values by comparing computed values with expected values.

```
mean_errors = vecnorm(mean_values - expected_mean);
cov_errors = zeros(size(N_values));

for j = 1:length(N_values)
    % Calculate Frobenius norm of the difference between sample covariance and expected
covariance.
    cov_errors(j) = norm(squeeze(cov_values(:, :, j)) - expected_covariance * eye(n), 'fro');
end
```

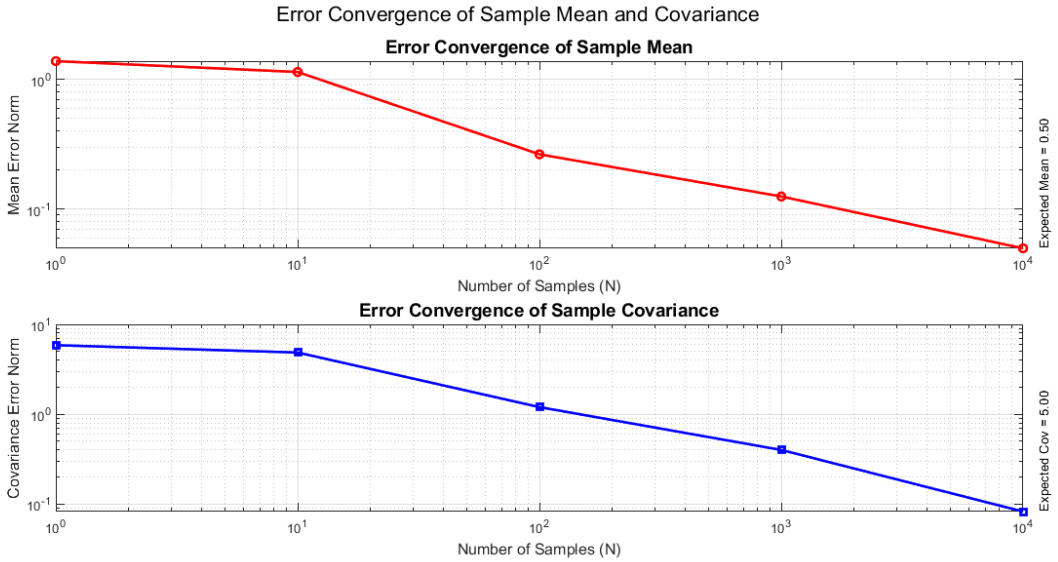
PLOT ERROR CONVERGENCE

This section plots the convergence of errors in sample mean and covariance values.

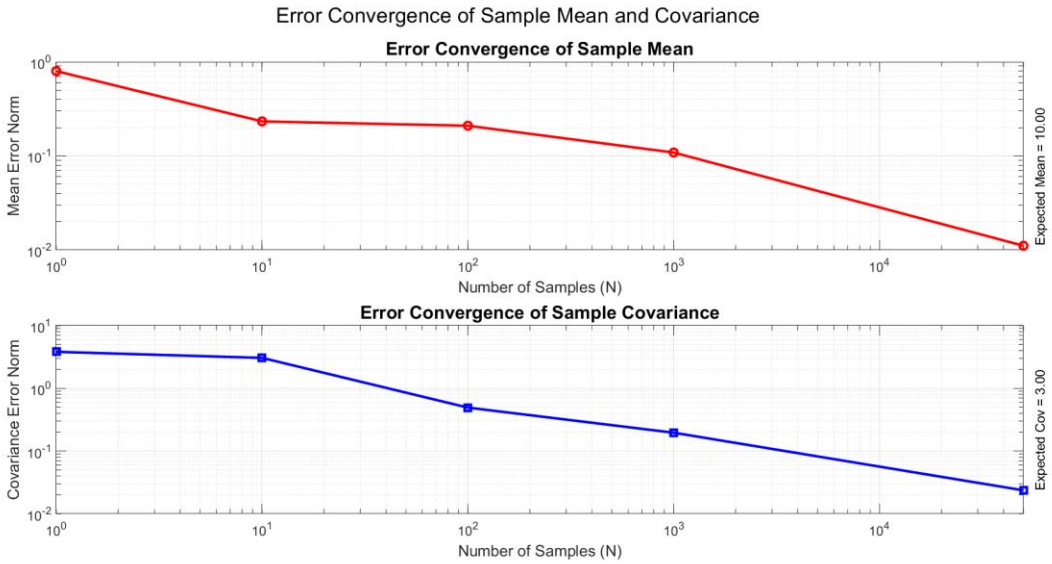
```
figure('WindowState', 'maximized'); % Open figure in fullscreen

% Plot errors for mean convergence.
subplot(2, 1, 1);
loglog(N_values, mean_errors, 'ro-', 'LineWidth', 2); % Red for mean error
xlabel('Number of Samples (N)', 'FontSize', 12);
ylabel('Mean Error Norm', 'FontSize', 12);
title('Error Convergence of Sample Mean', 'FontSize', 14);
```

```
grid on;
xlim([min(N_values) max(N_values)]);

% Add text outside and to the right of the plot (vertical).
text(max(N_values)*1.2, mean(mean_errors) - (mean(mean_errors) - min(mean_errors)), ...
    sprintf('Expected Mean = %.2f', expected_mean), ...
    'FontSize', 10, 'VerticalAlignment', 'middle', 'HorizontalAlignment', 'left', 'Rotation',
90);

% Plot errors for covariance convergence.
subplot(2, 1, 2);
loglog(N_values, cov_errors, 'bs-', 'LineWidth', 2); % Blue for covariance error
xlabel('Number of Samples (N)', 'FontSize', 12);
ylabel('Covariance Error Norm', 'FontSize', 12);
title('Error Convergence of Sample Covariance', 'FontSize', 14);
grid on;
xlim([min(N_values) max(N_values)]);

% Add text outside and to the right of the plot (vertical).
text(max(N_values)*1.2, mean(cov_errors) - (mean(cov_errors) - min(cov_errors)), ...
    sprintf('Expected Cov = %.2f', expected_covariance), ...
    'FontSize', 10, 'VerticalAlignment', 'middle', 'HorizontalAlignment', 'left', 'Rotation',
90);

% Adjust subplot spacing.
sgtitle('Error Convergence of Sample Mean and Covariance', 'FontSize', 16); % Overall title for
the figure

% Save figure in the Results folder as PNG.
saveas(gcf, fullfile(result_folder, 'error_convergence.png'));
```

As you can see by increasing number of sample the value of mean and covariance converge to expected value! So there would be less error value .



You can see further test:



DISPLAY THE ACCURACY RESULTS

Display the expected mean and covariance values, and the errors for selected N values.

```
disp('Expected Mean:');
disp(expected_mean);
disp('Expected Covariance:');
disp(expected_covariance);
disp('Errors with selected N values:');
disp(table(N_values', mean_errors', cov_errors', 'VariableNames', {'N', 'MeanError',
'CovError'}));
```

1. **First run command window:**

Expected Mean:
0.5000

Expected Covariance:
5

Errors with selected N values:

N	MeanError	CovError
10000	0.049646	0.082593
1000	0.12494	0.40055
100	0.26453	1.2002
10	1.1444	4.8616
1	1.3881	5.8689

2. **Second run command window:**

Expected Mean:
10

Expected Covariance:
3

Errors with selected N values:

N	MeanError	CovError
50000	0.011093	0.023622
1000	0.1086	0.19599
100	0.20935	0.49269
10	0.23285	3.058
1	0.79742	3.8196

PART 2: CORRELATION ANALYSIS OF RANDOM VARIABLES

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%{
Author: Alireza Sotoodeh (Student ID: 401412056)
Subject: Correlation Analysis of Random Variables
Description: Demonstrates the correlation between random variables Y and Z
             under different conditions and generates scatter plots.
%}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

INITIALIZATION

This section clears the command window, closes all figures, and clears the workspace variables.

```
clc;        % Clear command window
close all;  % Close all figures
clear all;  % Clear workspace variables
```

PARAMETERS

Define the number of samples for the simulation.

```
N = 10000;  % Number of samples
```

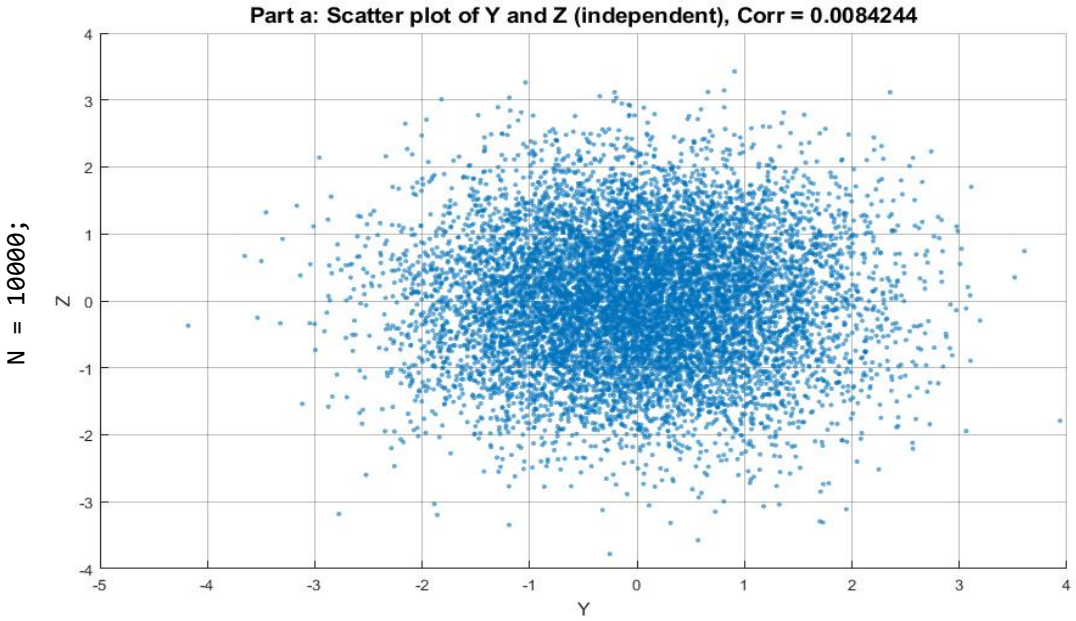
PART A: Y AND Z ARE INDEPENDENT AND NORMALLY DISTRIBUTED

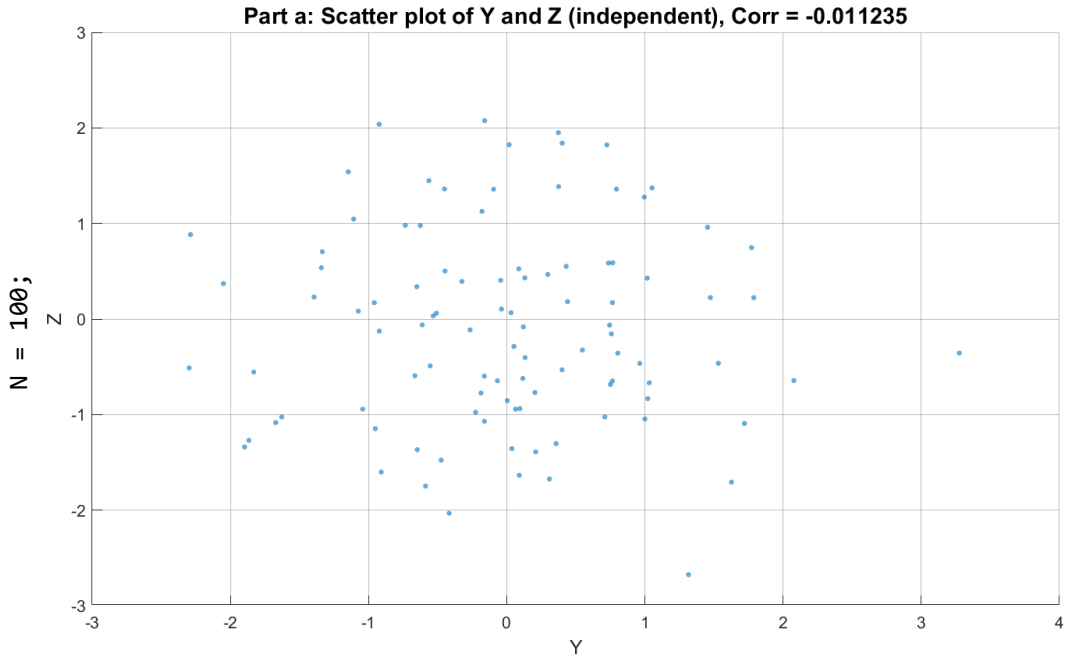
Generate independent random variables Y and Z from a standard normal distribution.

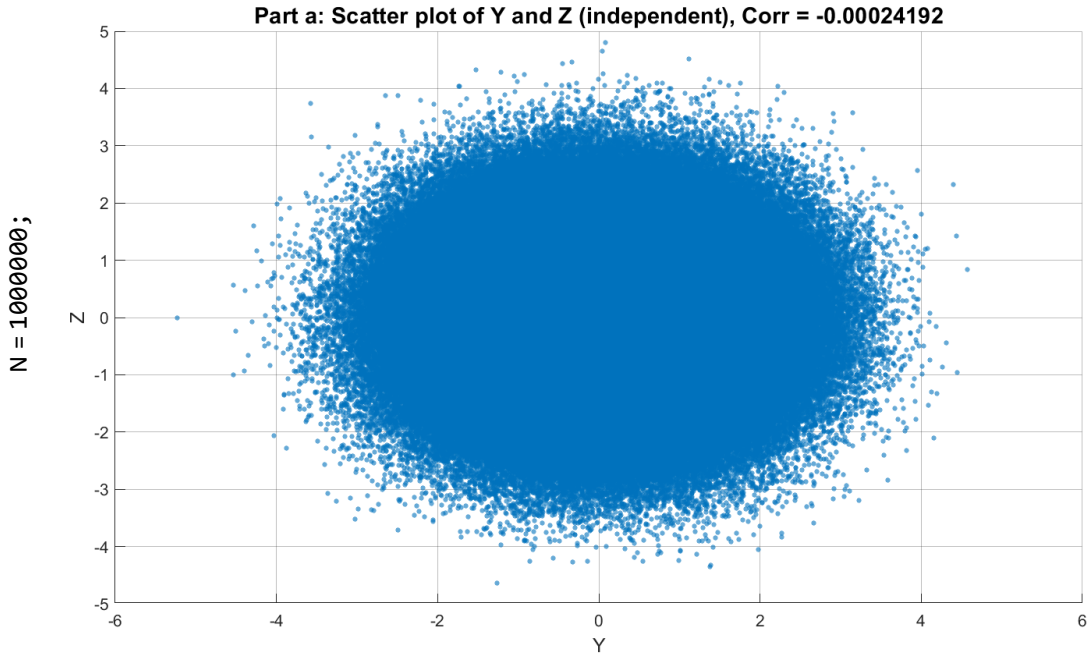
```
Y = randn(N, 1); % Y ~ N(0,1)
Z = randn(N, 1); % Z ~ N(0,1)

% Compute the covariance and correlation between Y and Z.
cov_YZ = cov(Y, Z);
corr_YZ = cov_YZ(1, 2) / sqrt(cov_YZ(1, 1) * cov_YZ(2, 2));
disp(['Part a: Correlation between independent Y and Z: ', num2str(corr_YZ)])

% Scatter plot of Y and Z.
figure('Units', 'normalized', 'Position', [0.1, 0.1, 0.8, 0.8]);
scatter(Y, Z, 10, 'filled', 'MarkerFaceAlpha', 0.6); % Use filled circles with transparency
title(['Part a: Scatter plot of Y and Z (independent), Corr = ', num2str(corr_YZ)], 'FontSize',
14, 'Interpreter', 'none');
```

```
xlabel('Y', 'FontSize', 12);
ylabel('Z', 'FontSize', 12);
grid on;
ax = gca; % Get current axis
ax.GridAlpha = 0.3; % Adjust grid transparency
```

Command window: `Part a: Correlation between independent Y and Z: 0.0084244`







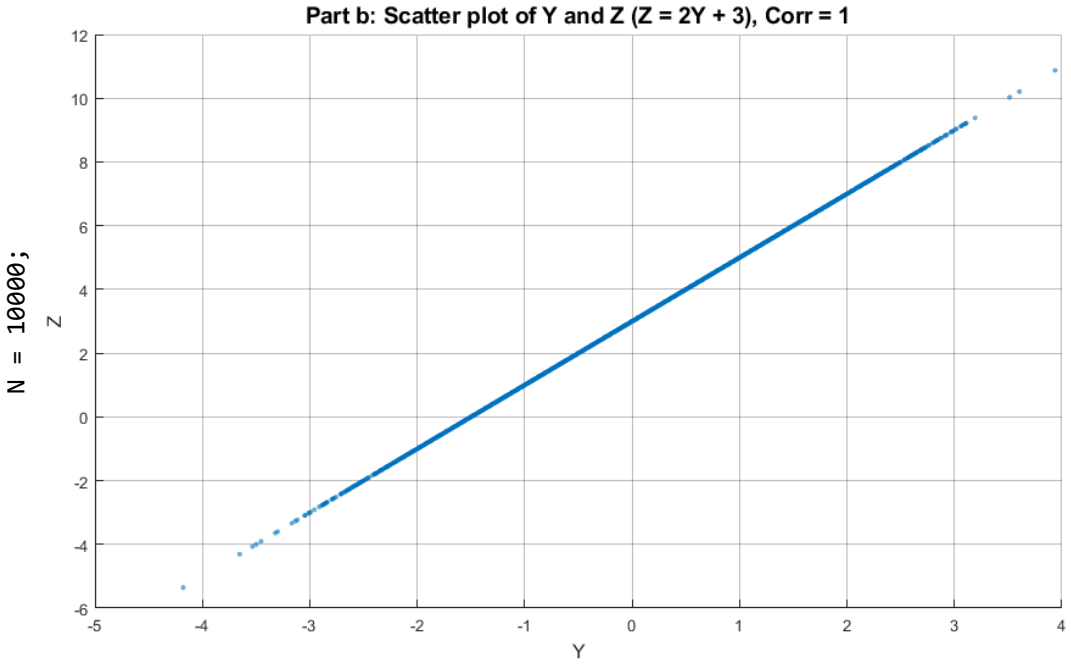
By increasing N (number of samples) we get closer to zero !

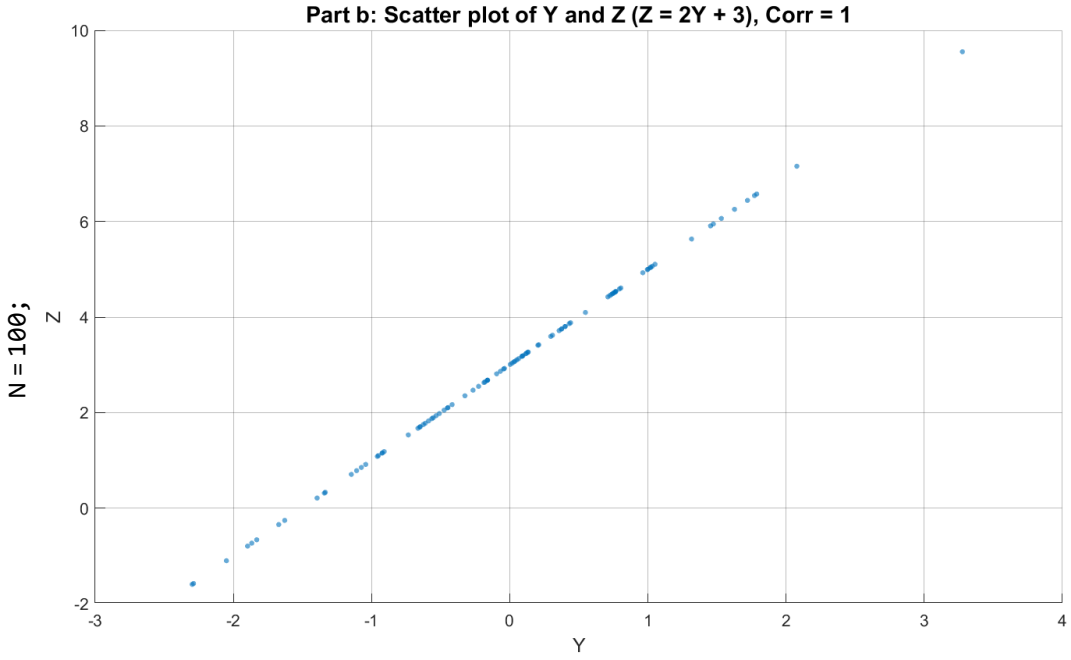
PART B: Z = 2Y + 3

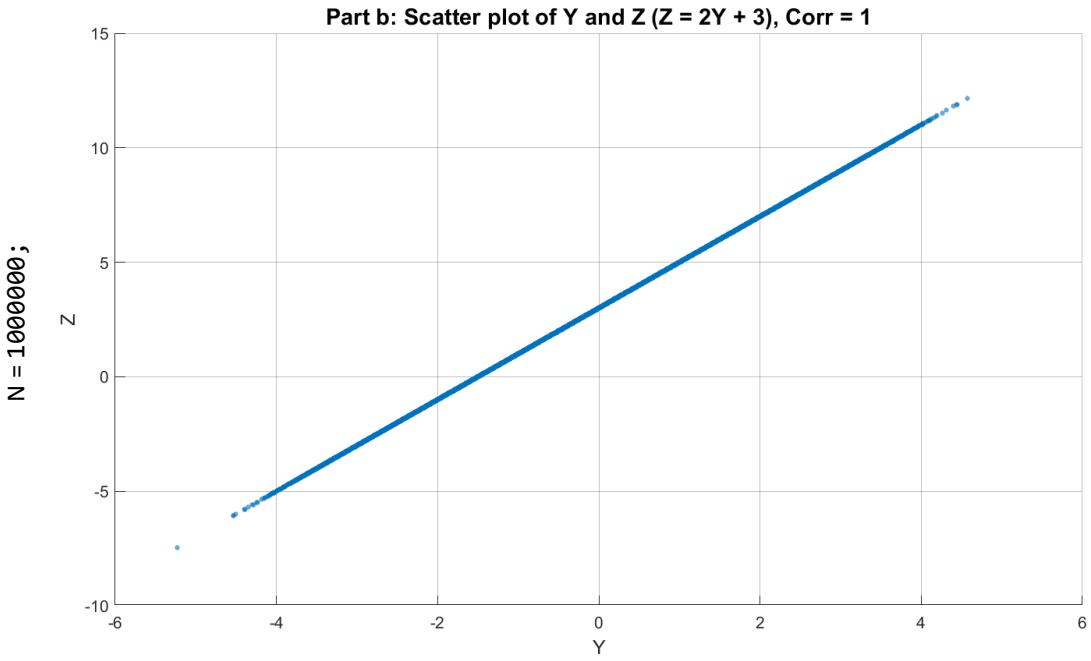
Define Z as a linear transformation of Y.

```
Z = 2*Y + 3;

% Compute the covariance and correlation between Y and Z.
cov_YZ = cov(Y, Z);
corr_YZ = cov_YZ(1, 2) / sqrt(cov_YZ(1, 1) * cov_YZ(2, 2));
disp(['Part b: Correlation between Y and Z = 2Y + 3: ', num2str(corr_YZ)])

% Scatter plot of Y and Z.
figure('Units', 'normalized', 'Position', [0.1, 0.1, 0.8, 0.8]);
scatter(Y, Z, 10, 'filled', 'MarkerFaceAlpha', 0.6); % Use filled circles with transparency
title(['Part b: Scatter plot of Y and Z (Z = 2Y + 3), Corr = ', num2str(corr_YZ)], 'FontSize',
14, 'Interpreter', 'none');
xlabel('Y', 'FontSize', 12);
ylabel('Z', 'FontSize', 12);
grid on;
ax = gca; % Get current axis
ax.GridAlpha = 0.3; % Adjust grid transparency
```

Command window: `Part b: Correlation between Y and Z = 2Y + 3: 1`


Part b: Scatter plot of Y and Z (Z = 2Y + 3), Corr = 1
N = 10000;
Z
Y


Part b: Scatter plot of Y and Z (Z = 2Y + 3), Corr = 1
N = 100;
Z
Y



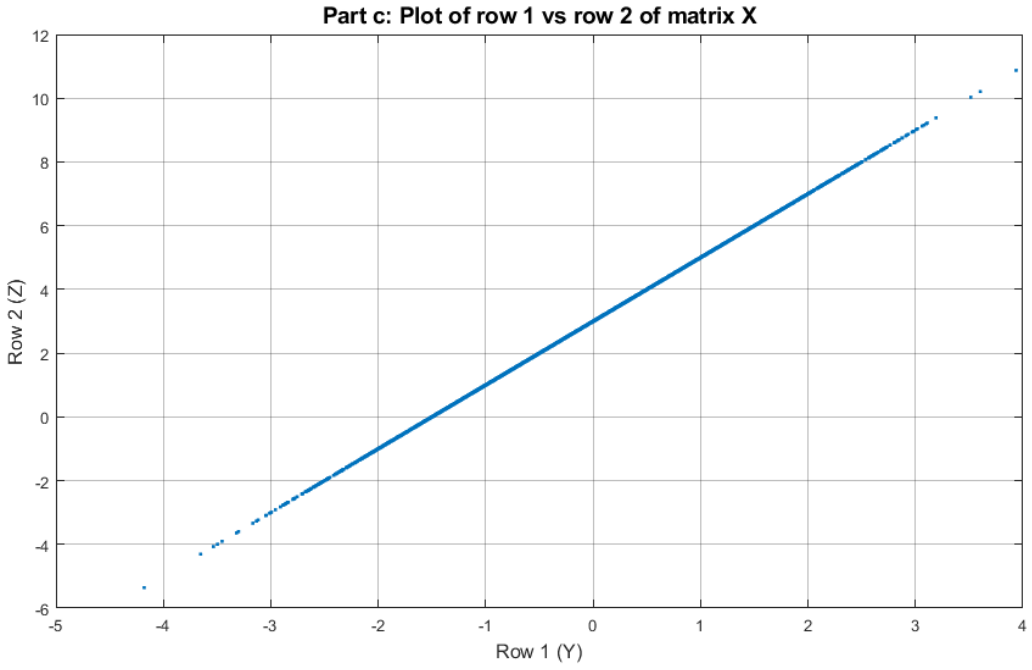
Doesn't matter how much we change the value of N (number of samples) the correlation value still would be 1!

PART C: PLOT ROWS OF MATRIX X

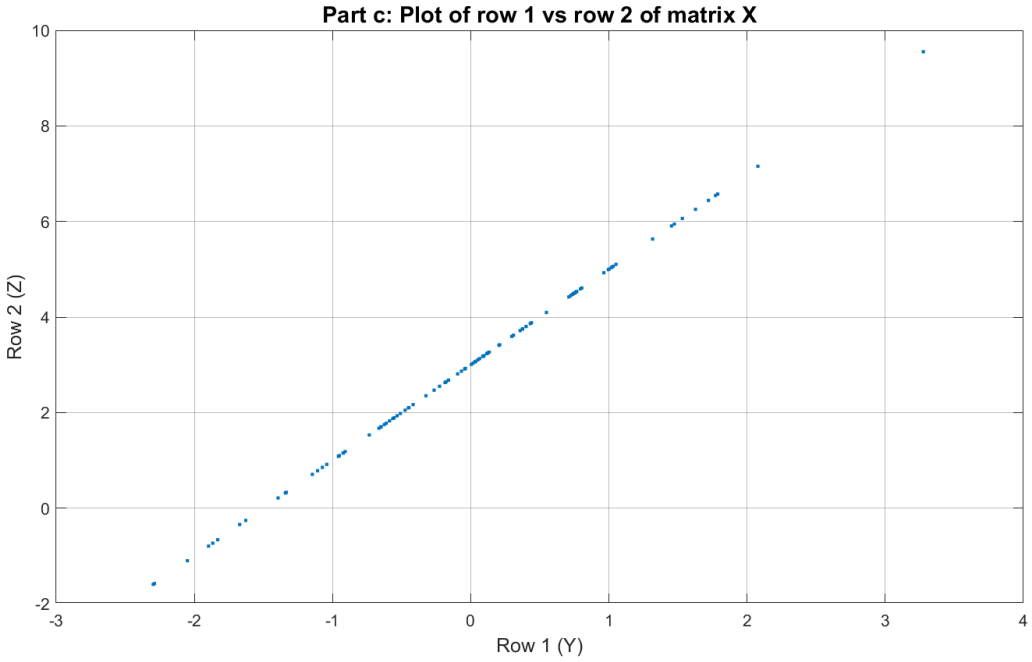
Define matrix X with Y and Z as its rows.

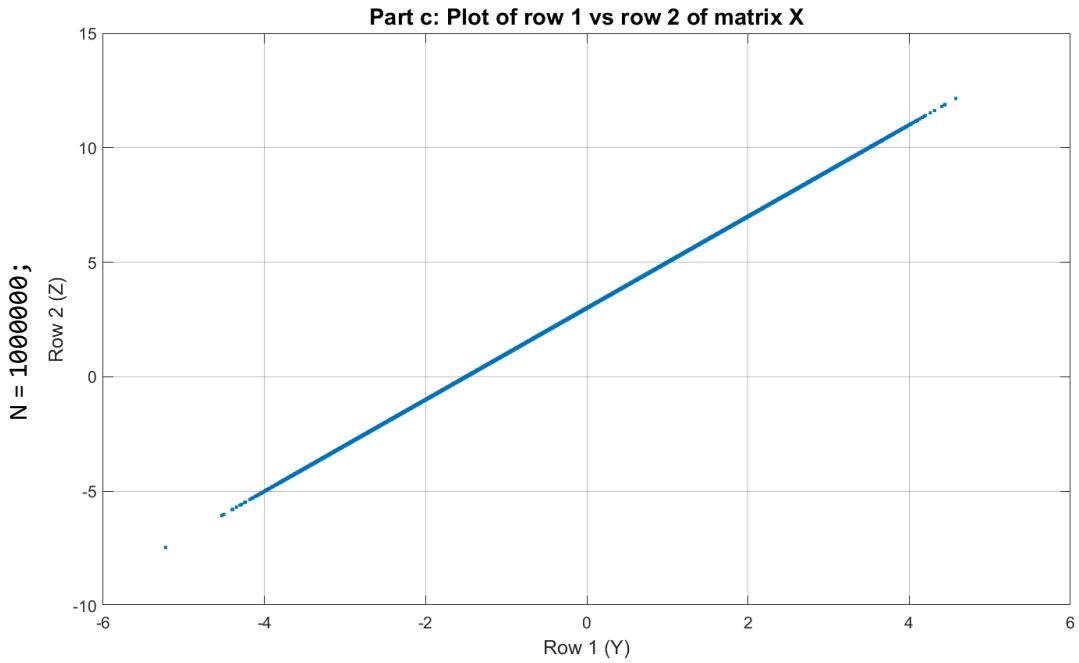
```
X = [Y'; Z'];

% Plot row 1 vs row 2 of matrix X.
figure('Units', 'normalized', 'Position', [0.1, 0.1, 0.8, 0.8]);
plot(X(1, :), X(2, :), '.');
title('Part c: Plot of row 1 vs row 2 of matrix X', 'FontSize', 14, 'Interpreter', 'none');
xlabel('Row 1 (Y)', 'FontSize', 12);
ylabel('Row 2 (Z)', 'FontSize', 12);
grid on;
ax = gca; % Get current axis
ax.GridAlpha = 0.3; % Adjust grid transparency
```

N = 10000;


Part c: Plot of row 1 vs row 2 of matrix X
Row 2 (Z)
Row 1 (Y)
12
10
8
6
4
2
0
-2
-4
-6
-5
-4
-3
-2
-1
0
1
2
3
4

N = 100;


Part c: Plot of row 1 vs row 2 of matrix X
Row 2 (Z)
Row 1 (Y)
10
8
6
4
2
0
-2
-3
-2
-1
0
1
2
3
4



Interpreting the Relationship between Y and Z

In Part c, by plotting Y (row 1 of matrix X) against Z (row 2 of matrix X). Here's how we can interpret the correlation visually:

1. **Direction of Relationship:**
   - Positive slope: Indicates a positive relationship.
   - Negative slope: Suggests a negative relationship.
   - Flat trend: No linear relationship.
2. **Strength of Relationship:**
   - Tight clustering around a line: Strong linear relationship.
   - Scattered points: Weaker correlation.
3. **Correlation Coefficient (ρ):**
   - ρ = 1: Perfect positive correlation.
   - ρ = -1: Perfect negative correlation.
   - ρ = 0: No linear correlation.

PART D: DEFINE Z TO ACHIEVE R_YZ = 2/3

Pearson Correlation Coefficient :

The Pearson correlation coefficient measures the linear relationship between two variables. It ranges from -1 (perfect negative correlation) to 1 (perfect positive correlation).

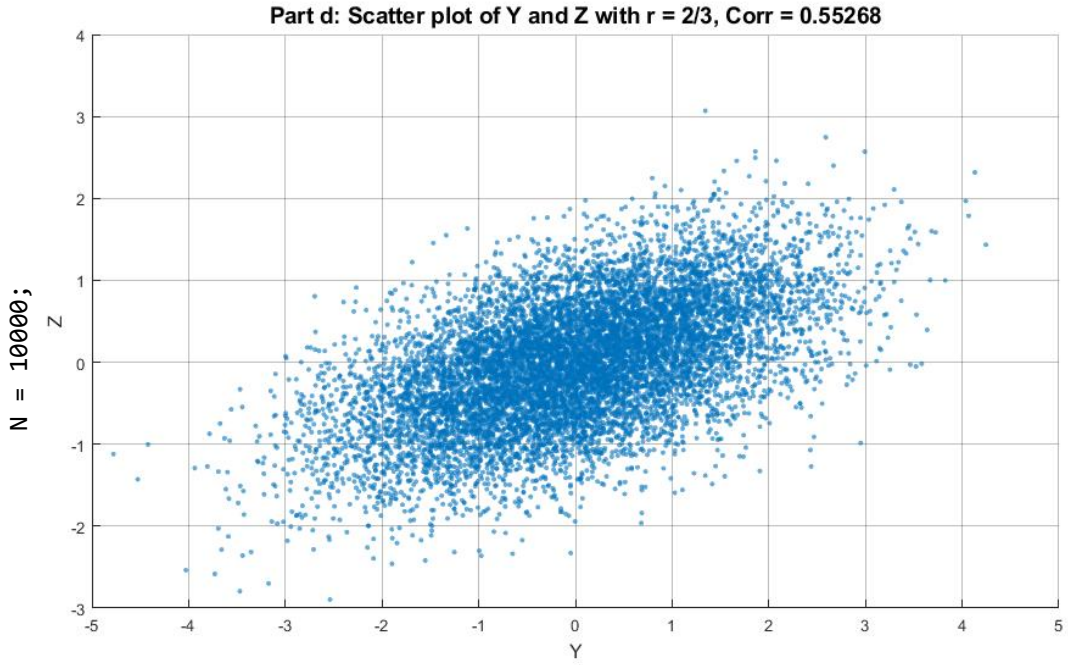
The formula for Pearson correlation is:

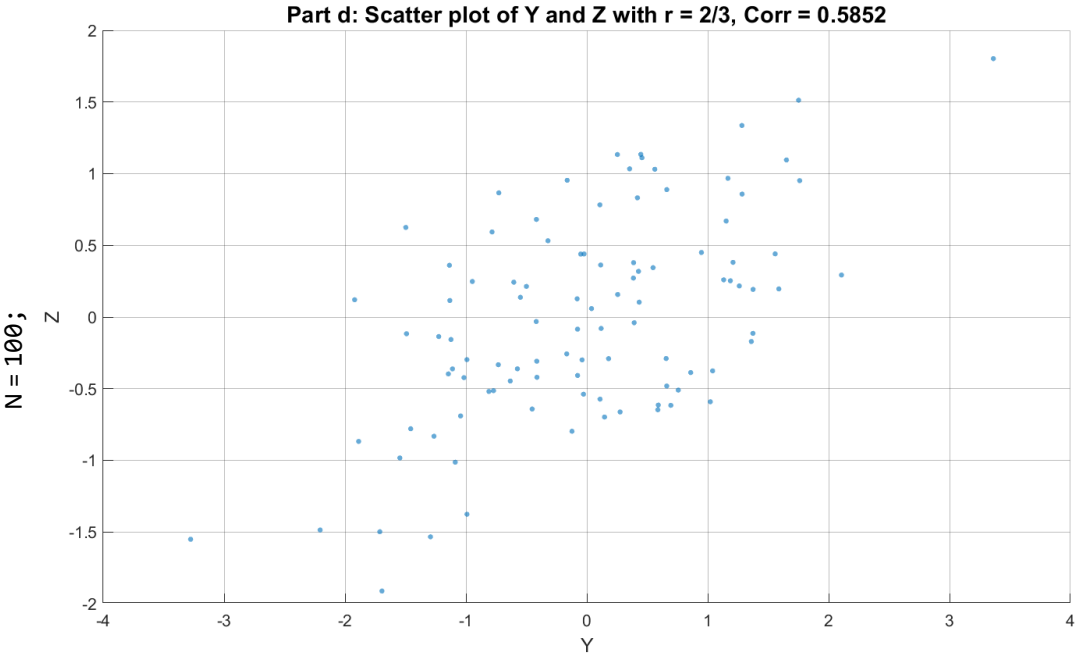
$$r = \frac{N\sum XY - (\sum X)(\sum Y)}{\sqrt{[N\sum X^2 - (\sum X)^2][N\sum Y^2 - (\sum Y)^2]}}$$

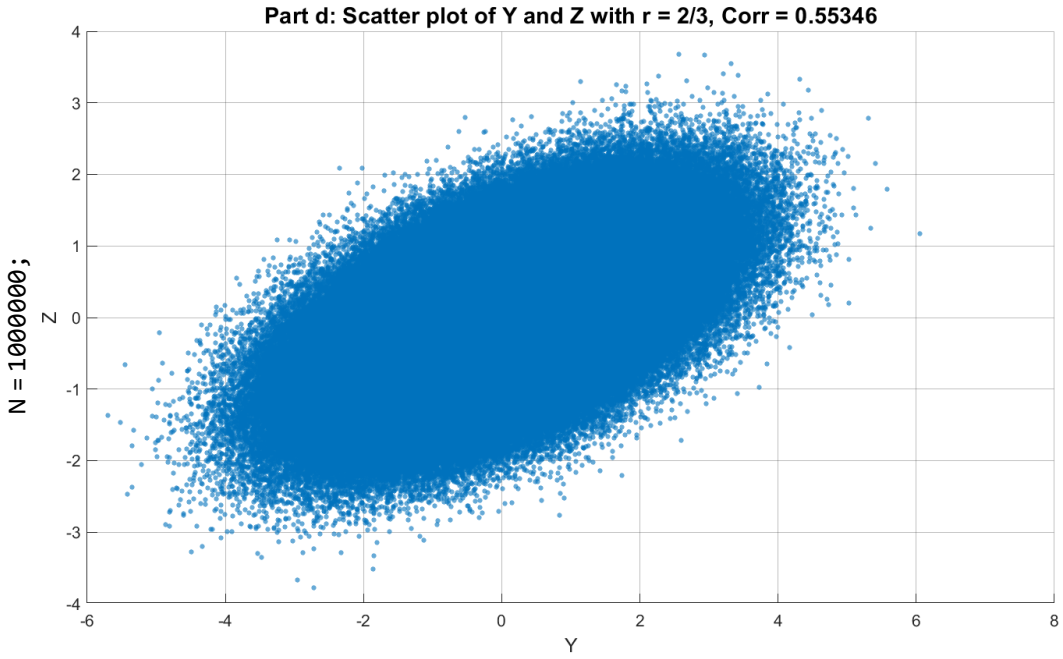
Define parameters for the desired correlation.

```
sigma_Y = 1;
sigma_Z = 1;
r = 2/3;

% Define the mean vector and covariance matrix.
mu = [0; 0];
Sigma = [sigma_Y^2, r*sigma_Y*sigma_Z; r*sigma_Y*sigma_Z, sigma_Z^2];

% Cholesky decomposition to create correlated random variables.
R = chol(Sigma);
Y = randn(N, 1);
Z = randn(N, 1);
Z_combined = R * [Y, Z]';

Y_new = Z_combined(1, :)';
Z_new = Z_combined(2, :)';

% Compute the covariance and correlation between Y_new and Z_new.
cov_YZ = cov(Y_new, Z_new);
corr_YZ = cov_YZ(1, 2) / sqrt(cov_YZ(1, 1) * cov_YZ(2, 2));
disp(['Part d: Correlation between Y and Z with r = 2/3: ', num2str(corr_YZ)])

% Scatter plot of Y_new and Z_new.
figure('Units', 'normalized', 'Position', [0.1, 0.1, 0.8, 0.8]);
scatter(Y_new, Z_new, 10, 'filled', 'MarkerFaceAlpha', 0.6); % Use filled circles with
transparency
title(['Part d: Scatter plot of Y and Z with r = 2/3, Corr = ', num2str(corr_YZ)], 'FontSize',
14, 'Interpreter', 'none');
xlabel('Y', 'FontSize', 12);
ylabel('Z', 'FontSize', 12);
grid on;
ax = gca; % Get current axis
ax.GridAlpha = 0.3; % Adjust grid transparency
```

Command window: `Part d: Correlation between Y and Z with r = 2/3: 0.55268`


Part d: Scatter plot of Y and Z with r = 2/3, Corr = 0.55268
N = 10000;
Z
Y


Part d: Scatter plot of Y and Z with r = 2/3, Corr = 0.5852
N = 100;
Z
Y



SAVE RESULTS

Create a folder to save results (if it doesn't exist create it!)

```
result_folder = 'Result_Project_2';
if ~exist(result_folder, 'dir')
    mkdir(result_folder);
end

% Save figures as PNG in the Result_Project_2 folder.
saveas(figure(1), fullfile(result_folder, 'part_a_scatter_plot.png'));
saveas(figure(2), fullfile(result_folder, 'part_b_scatter_plot.png'));
saveas(figure(3), fullfile(result_folder, 'part_c_plot.png'));
saveas(figure(4), fullfile(result_folder, 'part_d_scatter_plot.png'));
```

PART 3: GAUSSIAN VARIABLE TRANSFORMATION ANALYSIS'

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%{
Author: Alireza Sotoodeh (Student ID: 401412056)
Subject: Transformation and Distribution Analysis of Gaussian Random Variable
Description: Generates a Gaussian random variable X, applies the transformation Z = e^{-X},
             and analyzes the distribution of Z. It demonstrates the correctness of the
             transformation by comparing the histogram of Z with its theoretical PDF.
%}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

INITIALIZATION

This section clears the command window, closes all figures, and clears the workspace variables.

```
clc;        % Clear command window
close all;  % Close all figures
clear all;  % Clear workspace variables
```

PARAMETERS

Define parameters for the Gaussian random variable.

```
mu = 0; % Mean of the normal distribution
sigma = 1; % Standard deviation of the normal distribution
n = 1000; % Number of samples
```

GENERATE X ~ N(MU, SIGMA^2)

Generate a Gaussian random variable X with mean mu and standard deviation sigma.

```
X = mu + sigma * randn(1, n);
```

APPLY THE TRANSFORMATION Z = E^(-X)

Transform X to Z using the given transformation.

```
Z = exp(-X);
```

DEFINE THE PDF OF Z

Define the probability density function of Z.

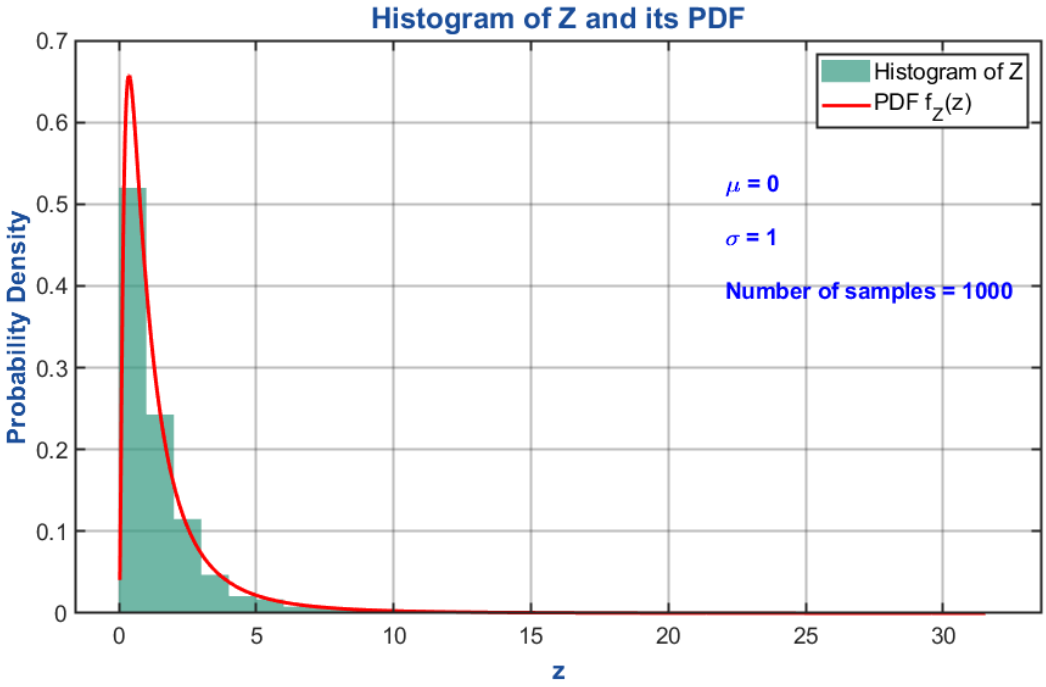
```
f_Z = @(z) (1./z) .* (1 / (sqrt(2 * pi) * sigma)) .* exp(-((log(z) - mu).^2) / (2 * sigma^2));
```

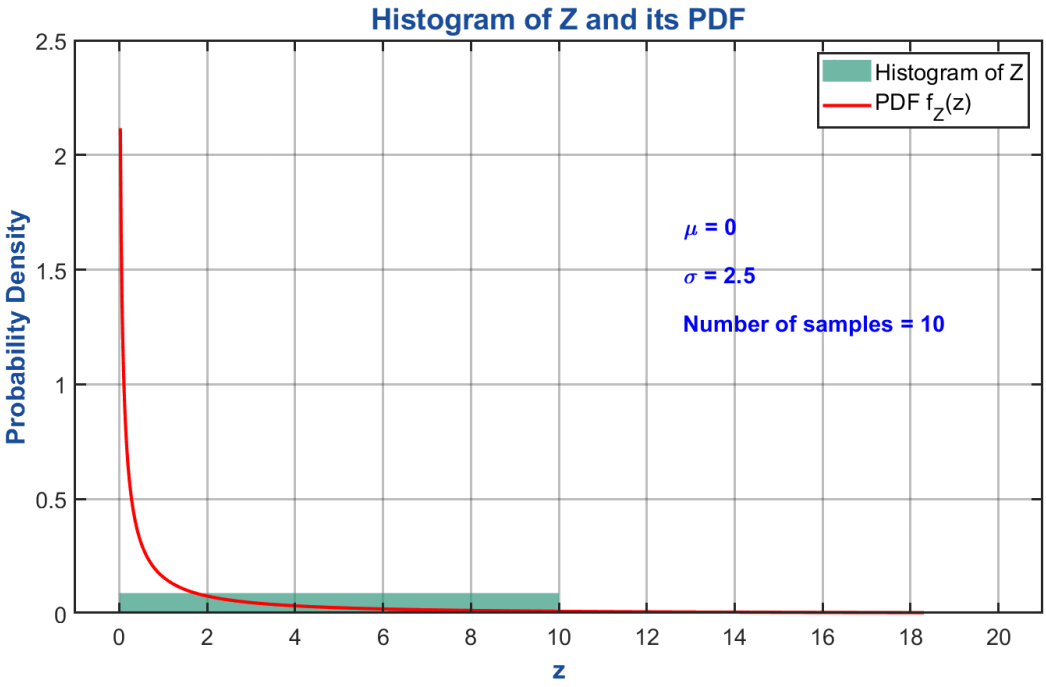
PLOT HISTOGRAM OF Z AND COMPARE WITH ITS PDF

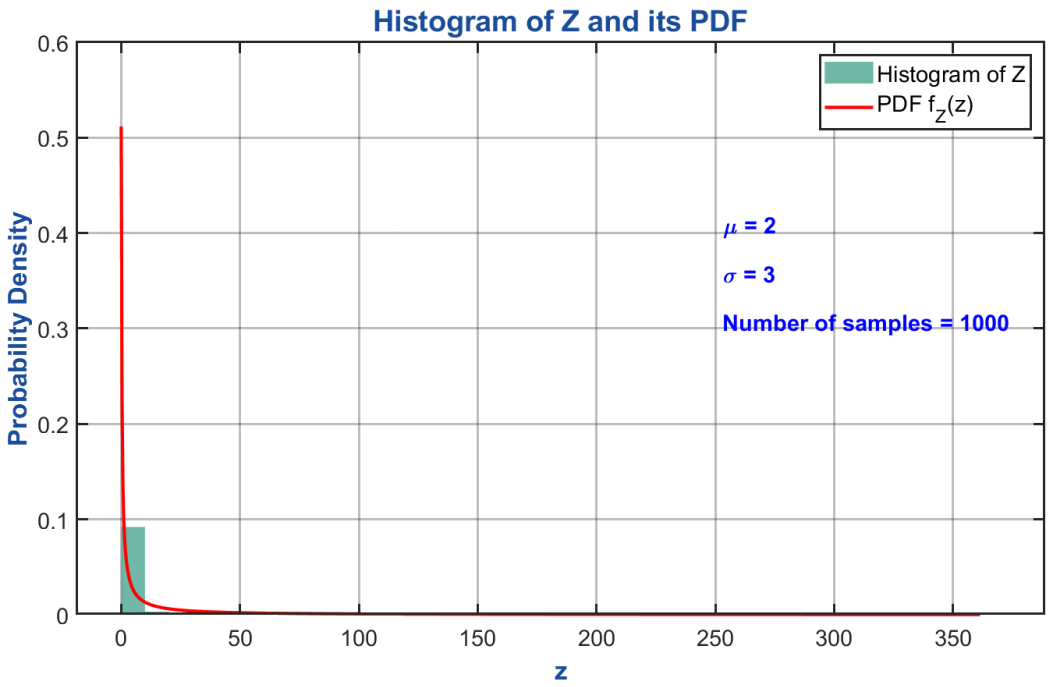
Plot the histogram of Z.

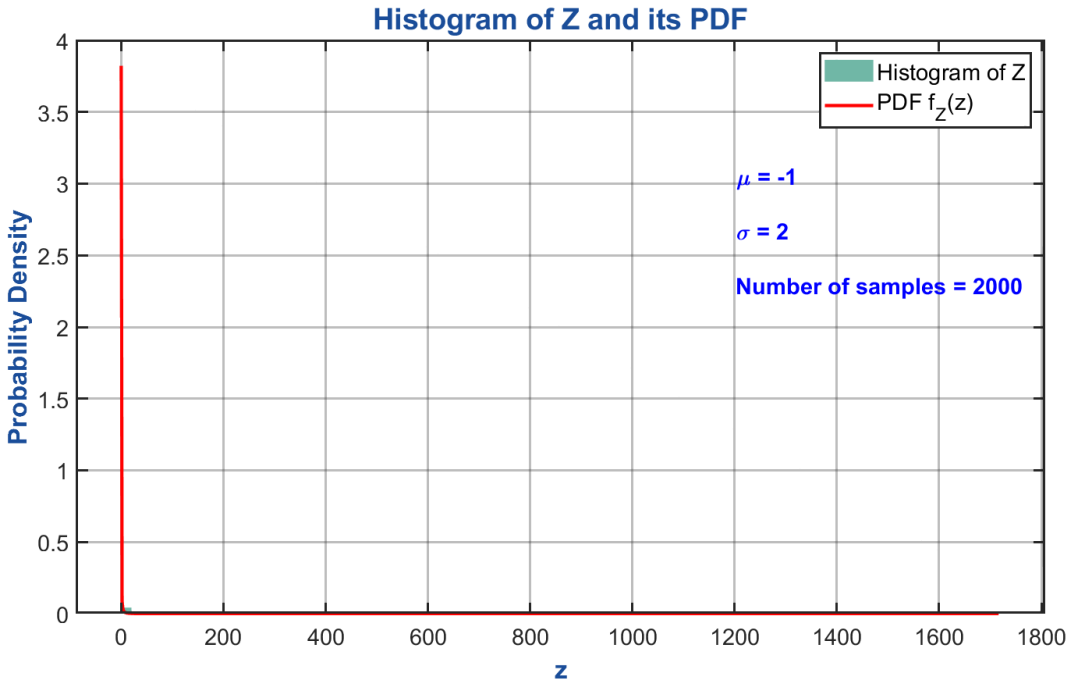
```
figure('Units', 'normalized', 'Position', [0.1, 0.1, 0.8, 0.8]); % Create a figure window with
normalized units
h = histogram(Z, 'Normalization', 'pdf', 'FaceColor', [0.2, 0.6, 0.5], 'EdgeColor', 'none',
'FaceAlpha', 0.7);
hold on;

% Generate values for z to plot f_Z(z)
z_values = linspace(min(Z), max(Z), 1000);

% Plot the PDF f_Z(z)
p = plot(z_values, f_Z(z_values), 'r', 'LineWidth', 2);
hold off;

% Customize the plot appearance
ax = gca; % Get current axis
ax.FontSize = 14; % Increase font size
ax.GridAlpha = 0.3; % Adjust grid transparency
ax.LineWidth = 1.5; % Thicker axis lines
ax.Box = 'on'; % Box around the plot
grid on;

% Labels and title
xlabel('z', 'FontSize', 16, 'FontWeight', 'bold', 'Color', [0.1, 0.3, 0.6]); % Label for x-axis
ylabel('Probability Density', 'FontSize', 16, 'FontWeight', 'bold', 'Color', [0.1, 0.3, 0.6]); %
Label for y-axis
title('Histogram of Z and its PDF', 'FontSize', 18, 'FontWeight', 'bold', 'Color', [0.1, 0.3,
0.6]); % Title of the plot
legend('Histogram of Z', 'PDF f_Z(z)', 'FontSize', 14, 'Location', 'northeast'); % Legend for the
plot

% Add text annotations to the plot
text(max(z_values) * 0.7, max(f_Z(z_values)) * 0.8, ['\mu = ', num2str(mu)], 'FontSize', 14,
'Color', 'b', 'FontWeight', 'bold'); % Mean
text(max(z_values) * 0.7, max(f_Z(z_values)) * 0.7, ['\sigma = ', num2str(sigma)], 'FontSize',
14, 'Color', 'b', 'FontWeight', 'bold'); % Standard deviation
text(max(z_values) * 0.7, max(f_Z(z_values)) * 0.6, ['Number of samples = ', num2str(n)],
'FontSize', 14, 'Color', 'b', 'FontWeight', 'bold'); % Number of samples

% Display parameters in command window
disp(['mu = ', num2str(mu)]);
```

```
disp(['sigma = ', num2str(sigma)]);
disp(['Number of samples = ', num2str(n)]);
```









SAVE RESULTS

Create a folder to save results if it doesn't exist.

```
result_folder = 'Result_Project_3';
if ~exist(result_folder, 'dir')
    mkdir(result_folder);
```

```
end

% Save the figure as PNG in the Result_Project_Transformation folder.
saveas(gcf, fullfile(result_folder, 'transformation_histogram_pdf.png'));
```

Thanks for reading 😊

Hope the best

Alireza Sotoodeh

Spring 1403