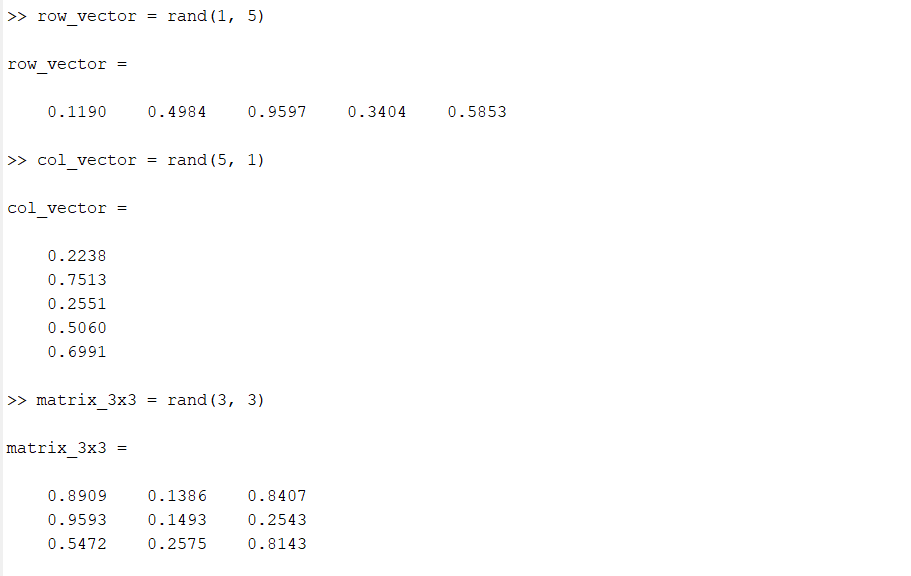
**Experiment 1**

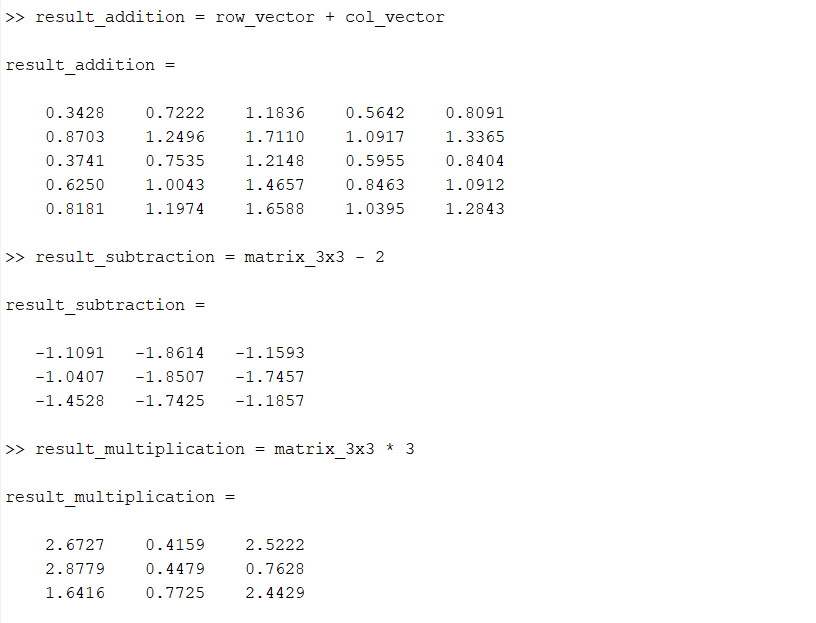
Aim: Creating a One and Two-Dimensional Array (Row / Column Vector) (Matrix of given size) then,

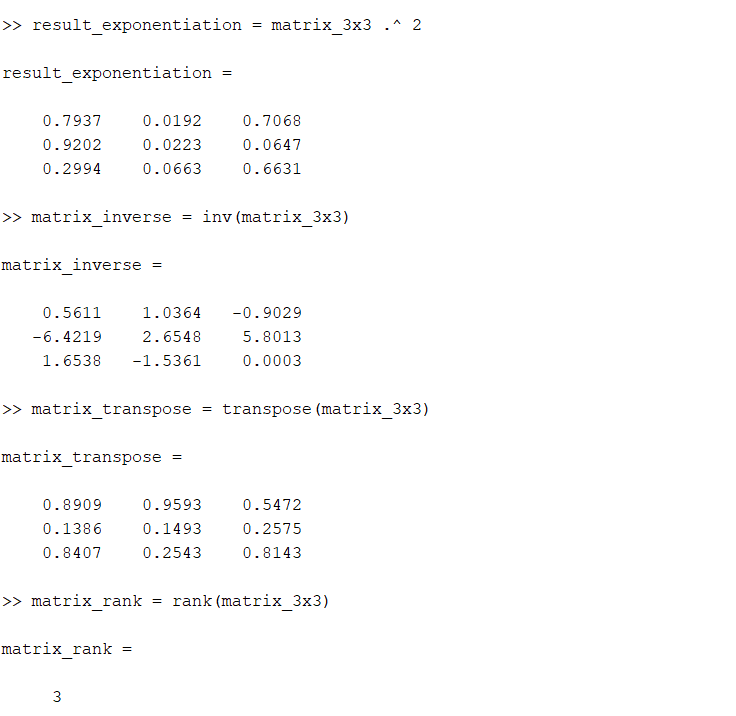
(A). Performing Arithmetic Operations - Addition, Subtraction, multiplication, and Exponentiation.

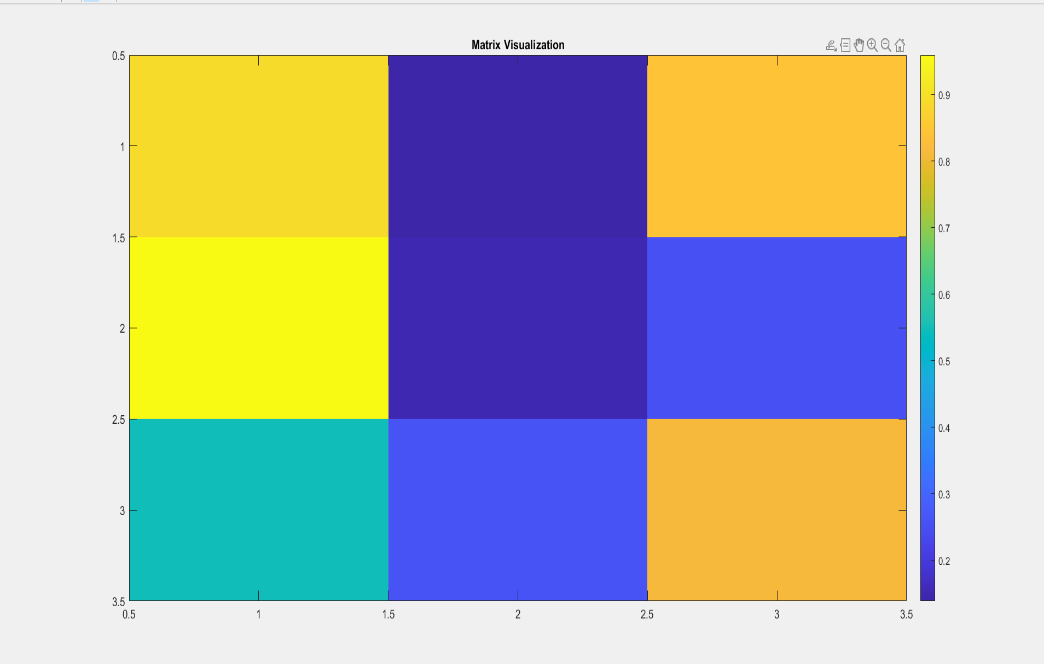
(B). Performing Matrix operations - Inverse, Transpose, Rank with plots.

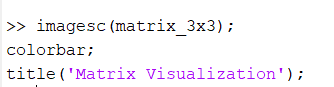
Solution:











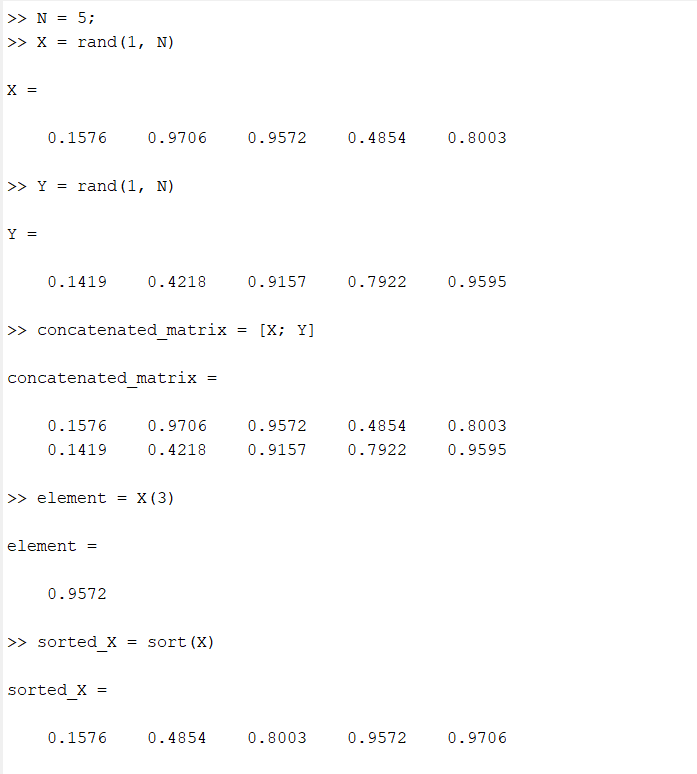
**Experiment 2**

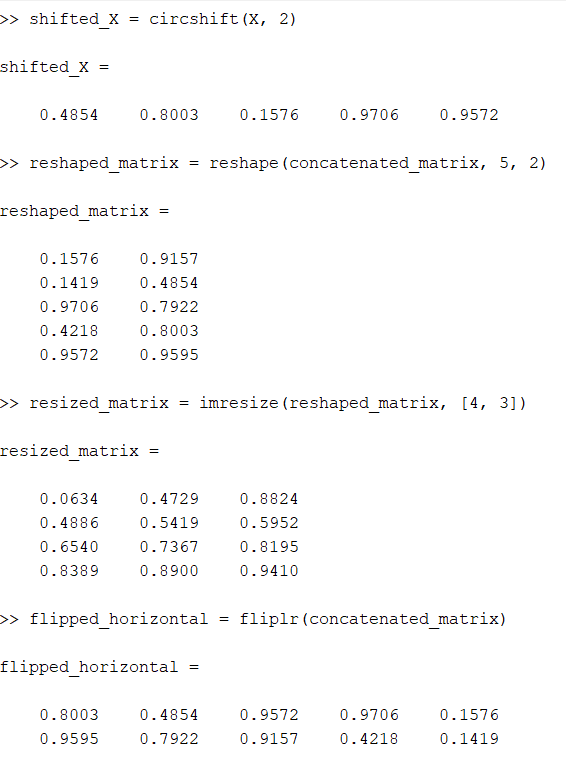
Aim: Performing Matrix Manipulations - Concatenating, Indexing, Sorting, Shifting, Reshaping, Resizing and Flipping about a Vertical Axis / Horizontal Axis; Creating Arrays X & Y of given size (1 x N) and Performing

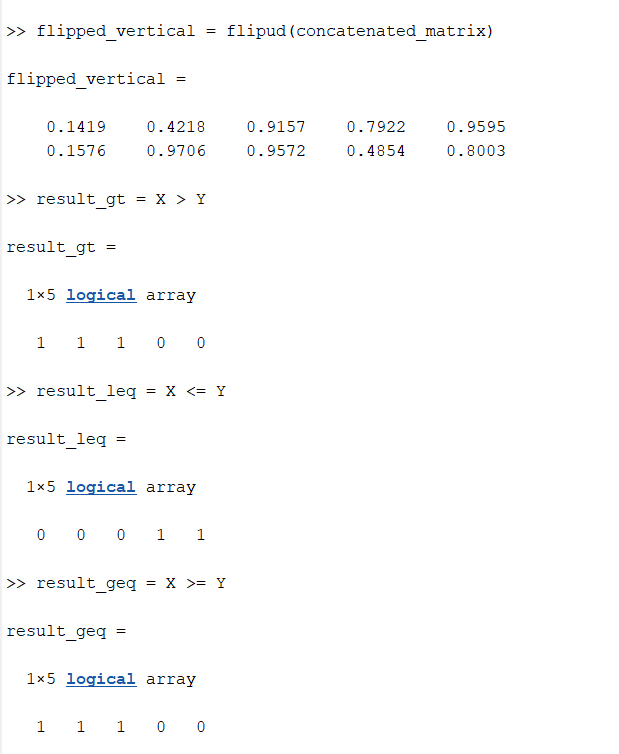
(A). Relational Operations - >, <=, >=, ~=

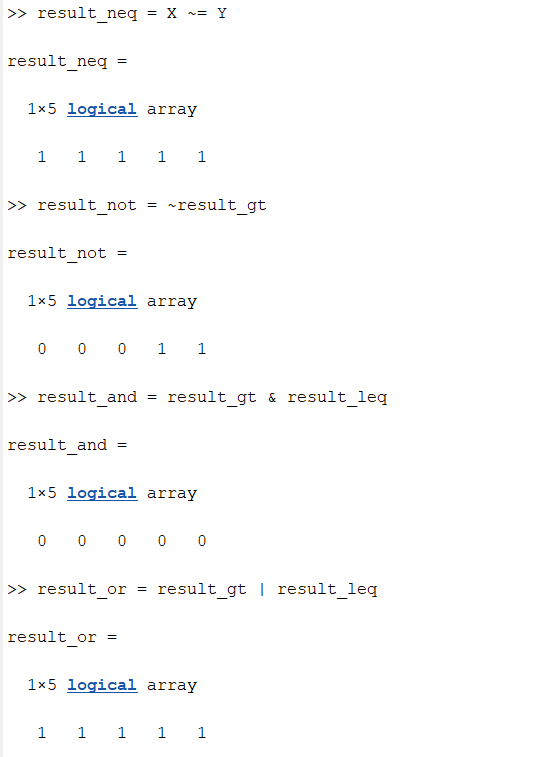
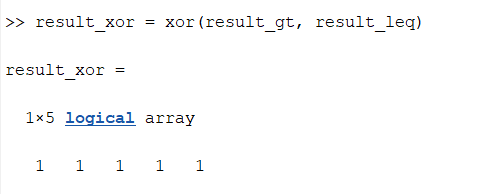
(B). Logical Operations - ~, & |, XOR.

Solution:







**Experiment 3**

Aim: Generating a set of Commands on a given Vector   
(Example: X = [1 8 3 9 0 1]) to

(A). Add up the values of the elements (Check with sum)

(B). Compute the Running Sum (Check with sum), where Running Sum for element j = the sum of the elements from 1 to j, inclusive. (C) Generating a Random Sequence using rand() / randn() functions and plot them.

Solution:

X = [1, 8, 3, 9, 0, 1];

commands = {

'sum(X)',

'cumsum(X)',

'randn(1, length(X))'

};

results = cell(size(commands));

for i = 1:length(commands)

results{i} = eval(commands{i});

end

disp('Results:');

for i = 1:length(results)

disp(['Command: ', commands{i}]);

disp(['Result: ', num2str(results{i})]);

disp(' ');

end

figure;

plot(results{3});

title('Random Sequence Plot');

xlabel('Index');

ylabel('Value');

Results:

Command: sum(X)

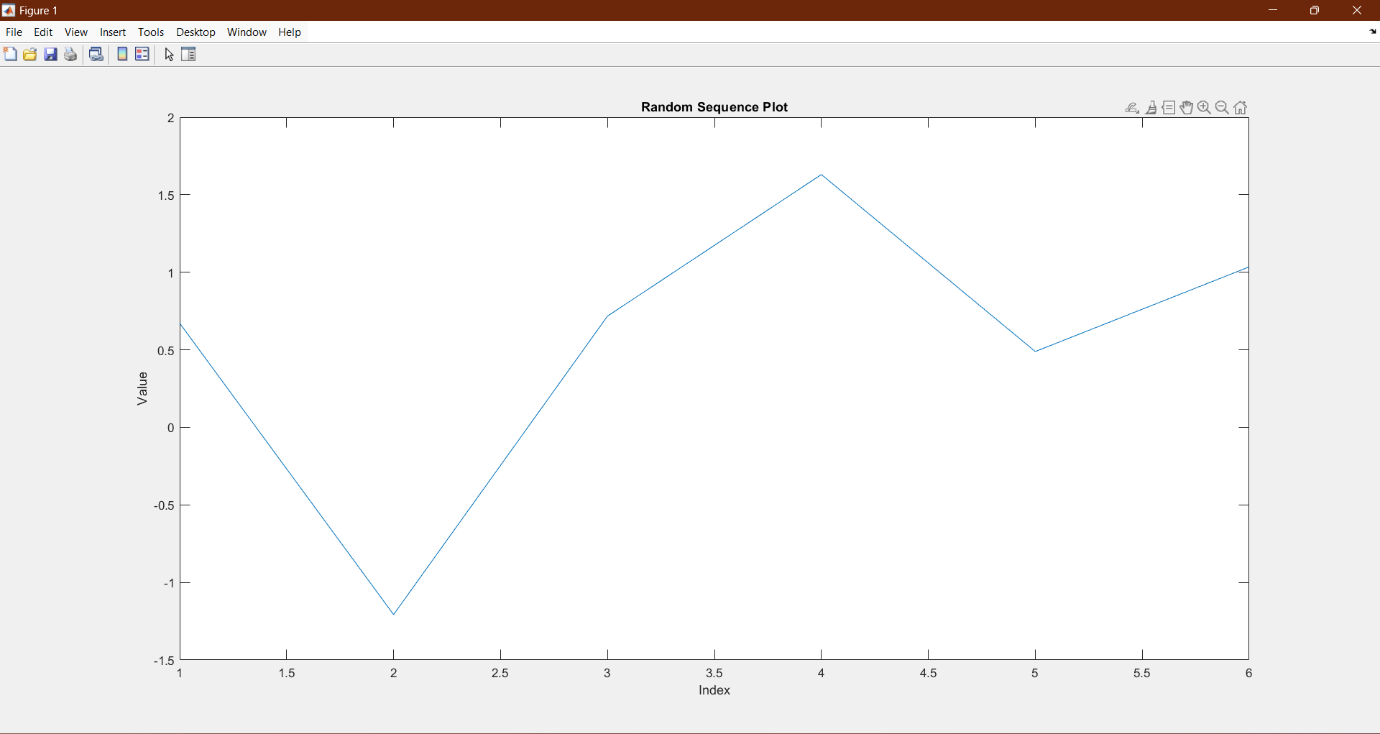
Result: 22

Command: cumsum(X)

Result: 1 9 12 21 21 22

Command: randn(1, length(X))

Result: 0.6715 -1.2075 0.71724 1.6302 0.48889 1.0347



**Experiment 4**

Aim: Evaluating a given expression and rounding it to the nearest integer value using Round, Floor, Ceil and Fix functions;

Also, generating and Plots of (A) Trigonometric Functions - sin(t),cos(t), tan(t), sec(t), cosec(t) and cot(t) for a given

duration, ‘t’. (B) Logarithmic and other Functions – log(A), log10(A), Square root of A, Real nth root of A.

Solution:

expr = 3.78 \* (5.62 / 2.45) + 7.91;

rounded\_round = round(expr);

rounded\_floor = floor(expr);

rounded\_ceil = ceil(expr);

rounded\_fix = fix(expr);

disp('Rounded Values:');

disp(['Round: ', num2str(rounded\_round)]);

disp(['Floor: ', num2str(rounded\_floor)]);

disp(['Ceil: ', num2str(rounded\_ceil)]);

disp(['Fix: ', num2str(rounded\_fix)]);

Rounded Values:

Round: 17

Floor: 16

Ceil: 17

Fix: 16

t = linspace(0, 2\*pi, 100);

sin\_t = sin(t);

cos\_t = cos(t);

tan\_t = tan(t);

sec\_t = 1 ./ cos(t);

cosec\_t = 1 ./ sin(t);

cot\_t = 1 ./ tan(t);

A = linspace(0.1, 10, 100);

log\_A = log(A);

log10\_A = log10(A);

sqrt\_A = sqrt(A);

nth\_root\_A = A.^(1/3);

figure;

subplot(2, 2, 1);

plot(t, sin\_t, t, cos\_t, t, tan\_t);

title('Trigonometric Functions');

legend('sin(t)', 'cos(t)', 'tan(t)');

xlabel('t');

ylabel('Value');

subplot(2, 2, 2);

plot(t, sec\_t, t, cosec\_t, t, cot\_t);

legend('sec(t)', 'cosec(t)', 'cot(t)');

xlabel('t');

ylabel('Value');

subplot(2, 2, 3);

plot(A, log\_A, A, log10\_A);

title('Logarithmic Functions');

legend('log(A)', 'log10(A)');

xlabel('A');

ylabel('Value');

subplot(2, 2, 4);

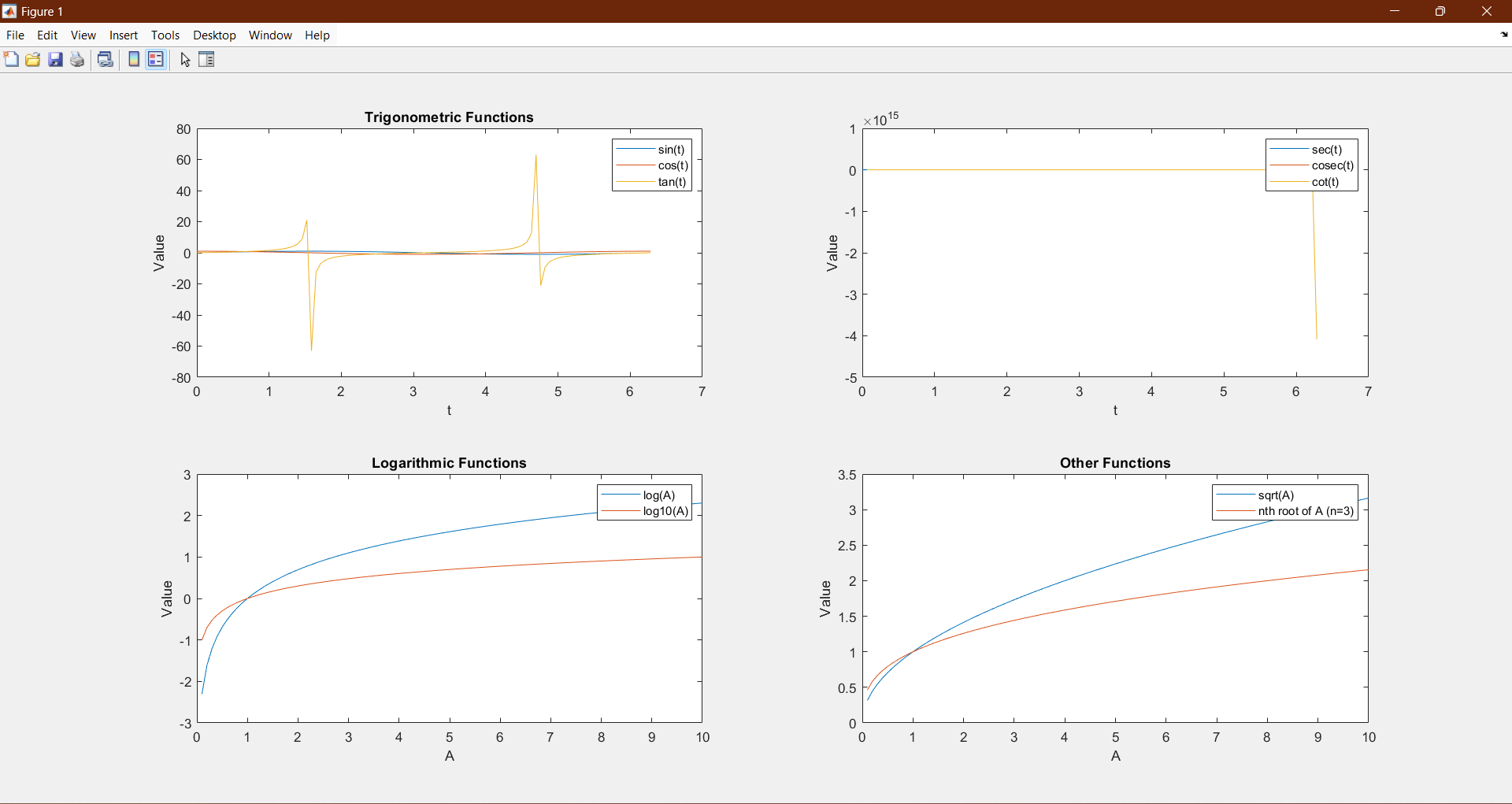
plot(A, sqrt\_A, A, nth\_root\_A);

title('Other Functions');

legend('sqrt(A)', 'nth root of A (n=3)');

xlabel('A');

ylabel('Value');



**Experiment 5**

**Aim:** Creating a vector X with elements, Xn = (-1)n+1/(2n-1) and Adding up 100 elements of the vector, X; And, plotting the functions, x, x3, ex , exp(x2 ) over the interval 0 < x < 4 (by choosing appropriate mesh values for x to obtain smooth curves), on A Rectangular Plot.

Solution :

n = 1:100;

X = (-1).^(n+1) ./ (2\*n - 1);

sum\_X = sum(X);

x\_values = linspace(0, 4, 1000);

x = x\_values;

x\_cube = x\_values.^3;

exp\_x = exp(x\_values);

exp\_x\_squared = exp(x\_values.^2);

figure;

subplot(2, 2, 1);

plot(x\_values, x);

title('x');

xlabel('x');

ylabel('Value');

subplot(2, 2, 2);

plot(x\_values, x\_cube);

title('x^3');

xlabel('x');

ylabel('Value');

subplot(2, 2, 3);

plot(x\_values, exp\_x);

title('e^x');

xlabel('x');

ylabel('Value');

subplot(2, 2, 4);

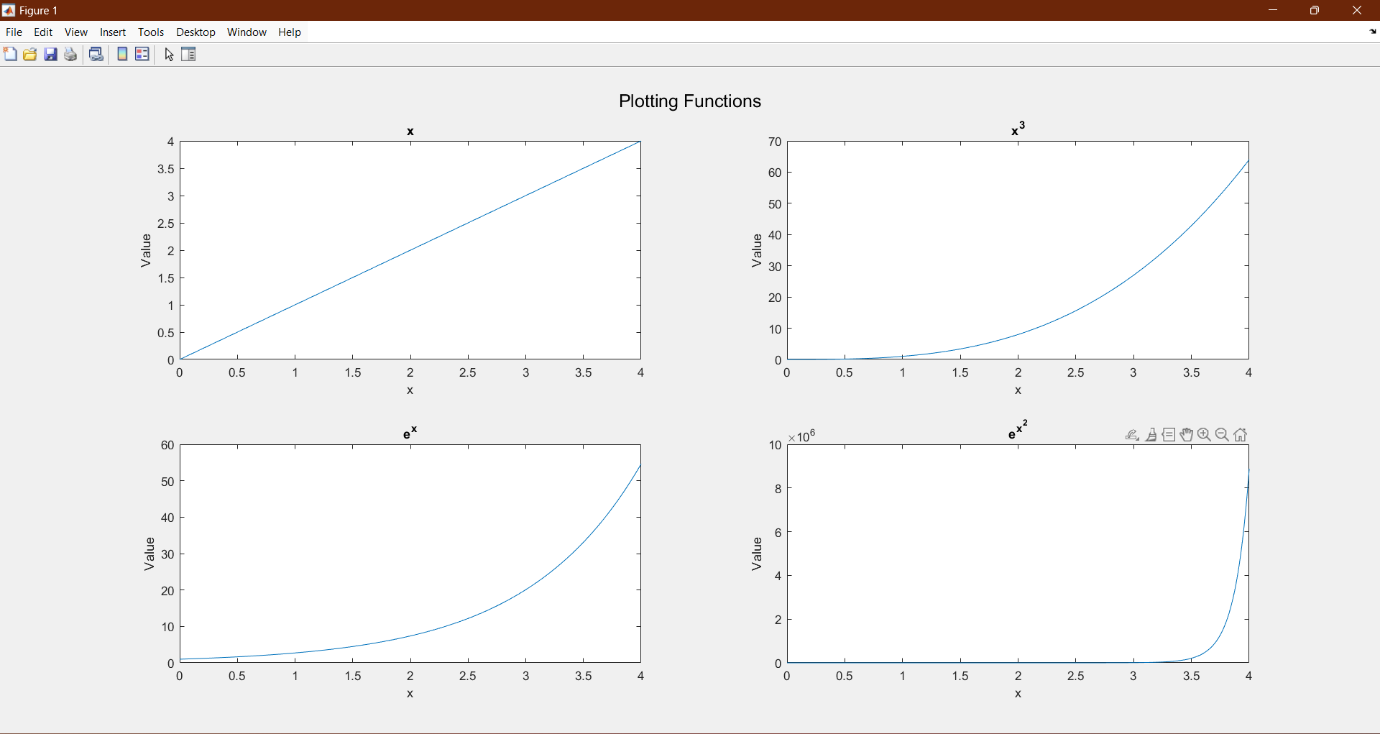
plot(x\_values, exp\_x\_squared);

title('e^{x^2}');

xlabel('x');

ylabel('Value');

sgtitle('Plotting Functions');



**Experiment 6**

**Aim:** Generating a Sinusoidal Signal of a given frequency with Titling, Labeling, Adding Text, Adding Legends, Printing Text in Greek Letters, Plotting as Multiple and Subplot. Time scale the generated signal for different values. E.g. 2X, 4X, 0.25X, 0.0625X.

Solution:

f = 1;

t = linspace(0, 2\*pi, 1000);

y = sin(2\*pi\*f\*t);

scaling\_factors = [2, 4, 0.25, 0.0625];

figure;

subplot(length(scaling\_factors) + 1, 1, 1);

plot(t, y, 'b', 'LineWidth', 1.5);

title('Original Sinusoidal Signal');

xlabel('Time (s)');

ylabel('Amplitude');

grid on;

text(1, 0.8, 'Original Signal', 'Color', 'blue');

legend('Signal');

hold on;

for i = 1:length(scaling\_factors)

scaled\_t = linspace(0, 2\*pi\*scaling\_factors(i), 1000);

scaled\_y = sin(2\*pi\*f\*scaled\_t);

subplot(length(scaling\_factors) + 1, 1, i + 1);

plot(scaled\_t, scaled\_y, 'r', 'LineWidth', 1.5);

title(['Scaled Sinusoidal Signal (', num2str(scaling\_factors(i)), 'X)']);

xlabel('Time (s)');

ylabel('Amplitude');

grid on;

text(1, 0.8, ['Scaled Signal (', num2str(scaling\_factors(i)), 'X)'], 'Color', 'red');

legend('Signal');

hold on;

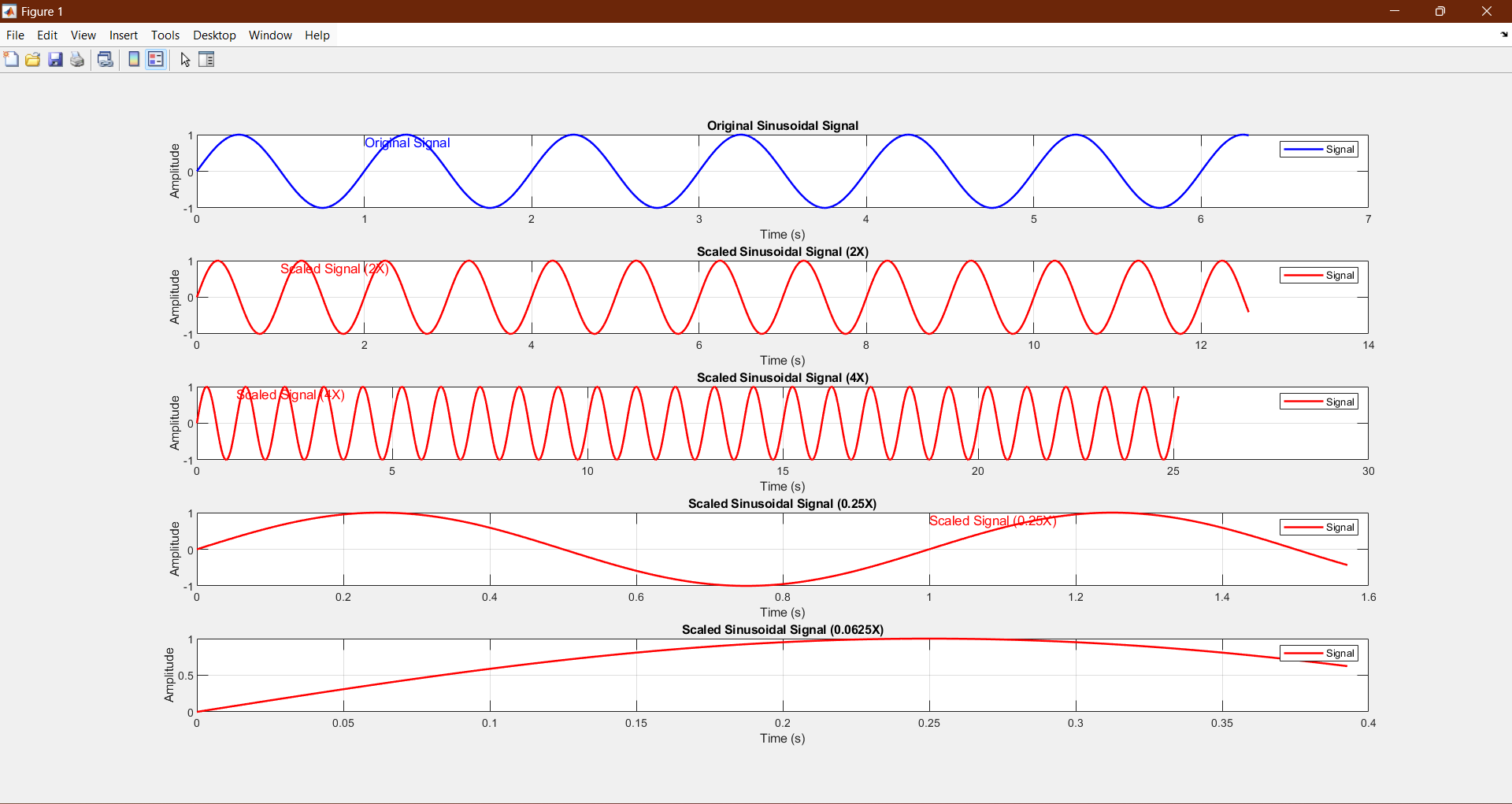
end

text(2, 0.5, '\alpha \beta \gamma \delta \epsilon \mu \nu \omega', 'Interpreter', 'latex');

set(gcf, 'Position', [100, 100, 800, 800]);

tightfig;

print('sinusoidal\_signals.pdf', '-dpdf', '-r300');

****

**Experiment 7**

Aim: Solving First, Second and third Order Ordinary Differential Equation using Built-in Functions and plot.

Solution:

First-Order ODE:

k = 0.1;

ode\_func = @(t, y) -k \* y;

tspan = [0 10];

y0 = 1;

[t, y] = ode45(ode\_func, tspan, y0);

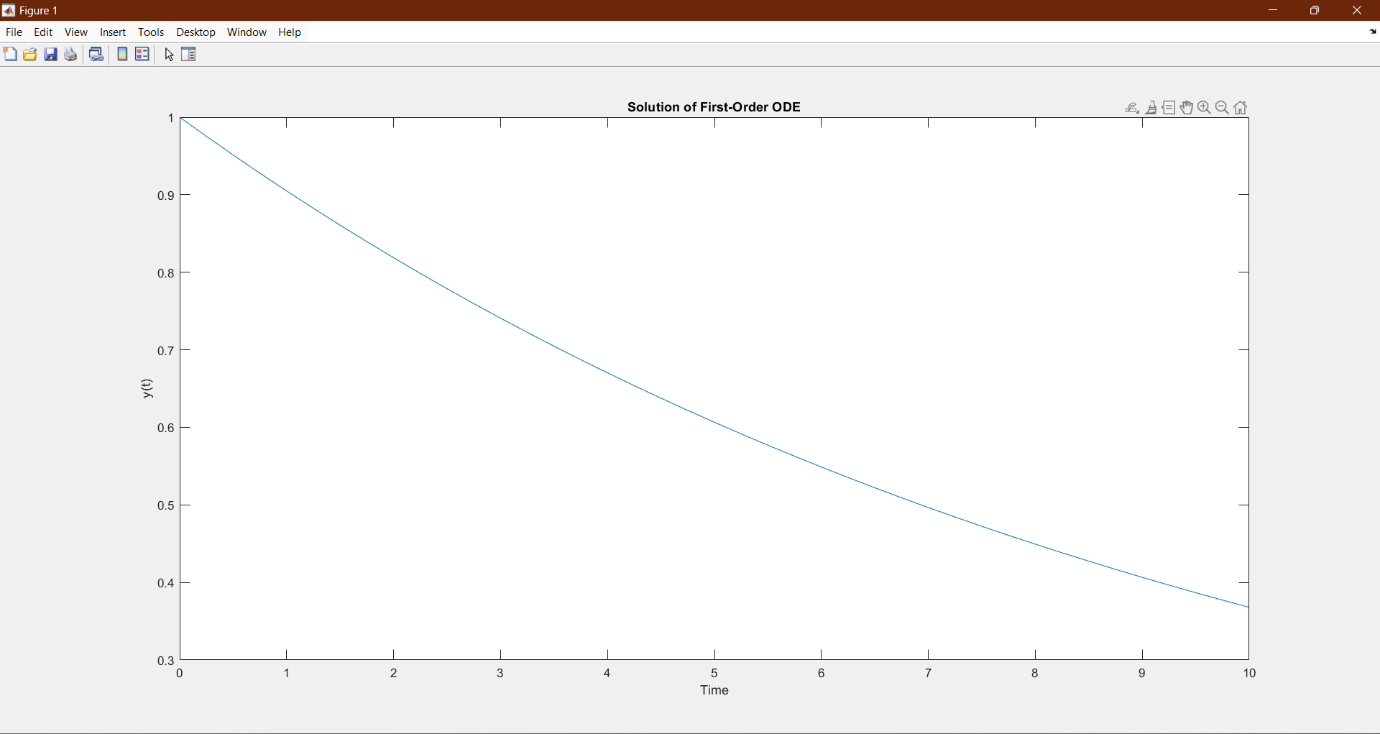
figure;

plot(t, y);

title('Solution of First-Order ODE');

xlabel('Time');

ylabel('y(t)');



Second-Order ODE :

m = 1;

c = 0.2;

k = 1;

ode\_func = @(t, y) [y(2); (1/m) \* (- c \* y(2) - k \* y(1))];

tspan = [0 10];

y0 = [0; 0];

[t, y] = ode45(ode\_func, tspan, y0);

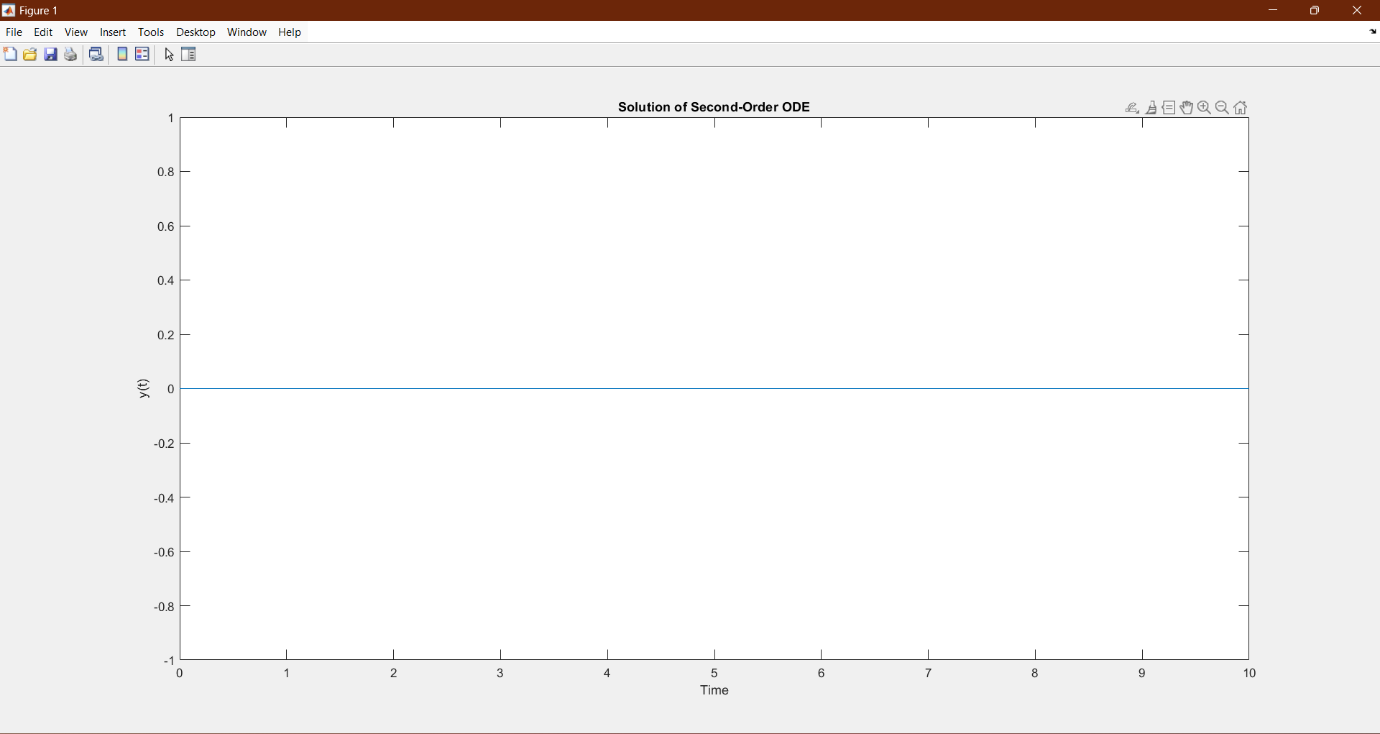
figure;

plot(t, y(:, 1));

title('Solution of Second-Order ODE');

xlabel('Time');

ylabel('y(t)');



Third-Order ODE :

a = 0.1;

b = 0.2;

c = 1;

ode\_func = @(t, y) [y(2); y(3); -a \* y(3) - b \* y(2) - c \* y(1)];

tspan = [0 10];

y0 = [0; 0; 1];

[t, y] = ode45(ode\_func, tspan, y0);

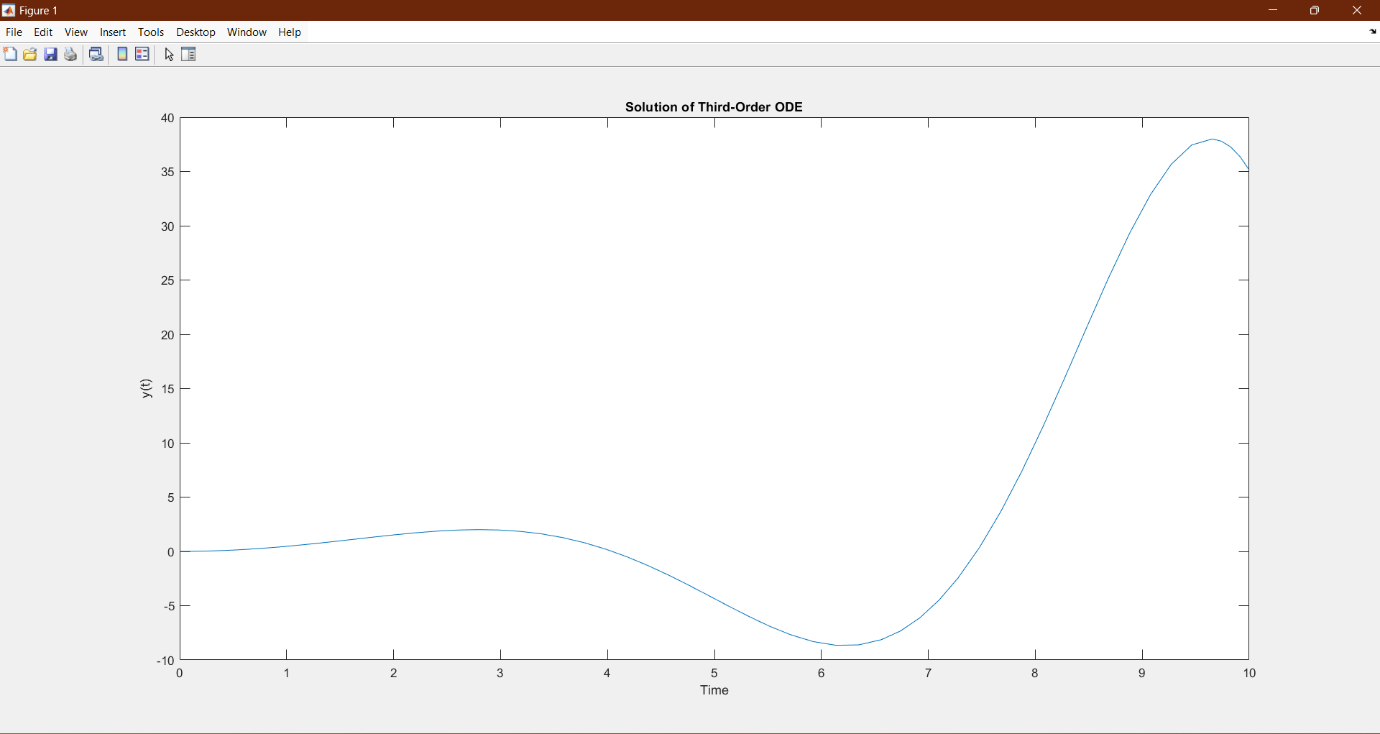
figure;

plot(t, y(:, 1));

title('Solution of Third-Order ODE');

xlabel('Time');

ylabel('y(t)');



**Experiment 8**

Aim: Writing brief Scripts starting each Script with a request for input (using input) to Evaluate the function h(T) using if-else statement, where o h(T) = (T – 10) for 0 < T <100 = (0.45 T + 900) for T > 100. Exercise: Testing the Scripts written using A). T = 5, h = -5 and B). T = 110, h =949.5.

Solution:

T = input('Enter the temperature T: ');

if T > 0 && T < 100

h = T - 10;

elseif T >= 100

h = 0.45 \* T + 900;

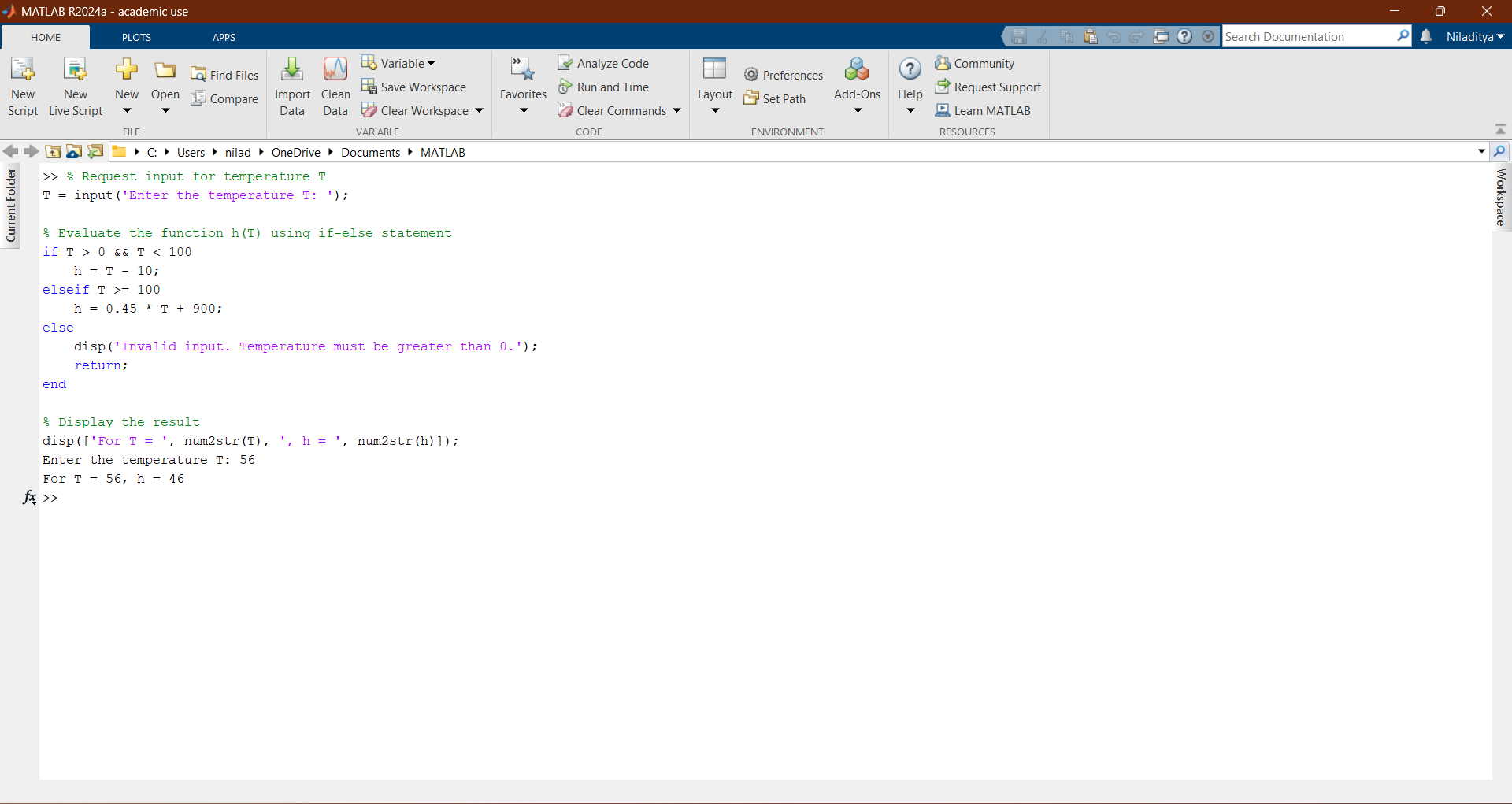
else

disp('Invalid input. Temperature must be greater than 0.');

return;

end

disp(['For T = ', num2str(T), ', h = ', num2str(h)]);



**Experiment 9**

Aim: Generating a Square Wave from sum of Sine Waves of certain Amplitude and Frequencies.

Solution:

duration = 2;

sampling\_frequency = 1000;

fundamental\_frequency = 10;

num\_harmonics = 10;

t = linspace(0, duration, duration \* sampling\_frequency);

square\_wave = zeros(size(t));

for n = 1:2:num\_harmonics\*2

harmonic\_frequency = n \* fundamental\_frequency;

amplitude = 4 / (n \* pi);

square\_wave = square\_wave + amplitude \* sin(2 \* pi \* harmonic\_frequency \* t);

end

figure;

plot(t, square\_wave);

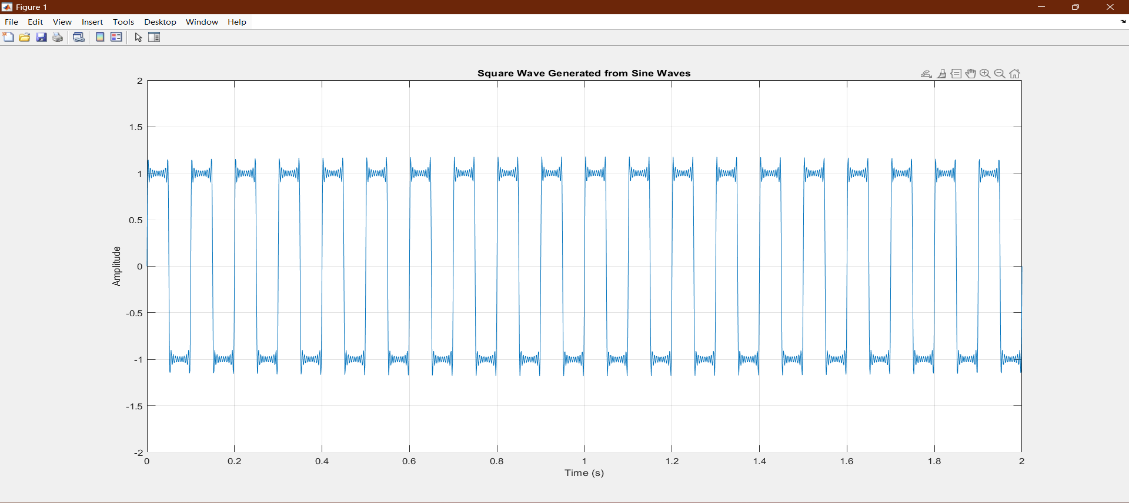
title('Square Wave Generated from Sine Waves');

xlabel('Time (s)');

ylabel('Amplitude');

ylim([-2, 2]); % Adjust ylim as needed

grid on;



**Experiment 10**

Aim: Basic 2D and 3D plots: parametric space curve. polygons with vertices. 3D contour lines, pie and bar charts.

Solution:

t = linspace(0, 2\*pi, 100);

x = cos(t);

y = sin(t);

figure;

plot(x, y);

title('Parametric Space Curve');

xlabel('x');

ylabel('y');

axis equal;

grid on;

