TASK 1

a)

```
function [OutputMatrix] = ComputationMatrix(size)
    OutputMatrix = zeros(size,size);
    arrayA = 0:size-1;
    for i=1:size
        OutputMatrix(i,:) = arrayA;
        arrayA = arrayA+size;
    end
    OutputMatrix = OutputMatrix';
end
```

```
testLoop = LoopMatrix(6)
```

testLoop = 6×6

0	6	12	18	24	30
1	7	13	19	25	31
2	8	14	20	26	32
3	9	15	21	27	33
4	10	16	22	28	34
5	11	17	23	29	35

b)

```
function [outputMatrix] = ReshapeMatrix(size)
    array = 0:size^2 - 1;
    outputMatrix = reshape(array, [size, size]);
end
```

```
testReshape = ReshapeMatrix(6)
```

testReshape = 6×6

0	6	12	18	24	30
1	7	13	19	25	31
2	8	14	20	26	32
3	9	15	21	27	33
4	10	16	22	28	34
5	11	17	23	29	35

c)

```
function [OutputMatrix] = ComputationMatrix(size)
    OutputMatrix = zeros(size,size);
    arrayA = 0:size-1;
    for i=1:size
        OutputMatrix(i,:) = arrayA;
        arrayA = arrayA+size;
    end
    OutputMatrix = OutputMatrix';
end
```

```
testCompute = ComputationMatrix(6)
```

testCompute = 6×6

0	6	12	18	24	30
1	7	13	19	25	31
2	8	14	20	26	32
3	9	15	21	27	33
4	10	16	22	28	34
5	11	17	23	29	35

TASK 2

a)

```
% Generate 1000 random numbers distributed normally with standard deviation
% 2. randn will try to keep mean close to 0 because of normal distribution
e_i = 2 * randn(1000, 1);

mean_e_i = mean(e_i);
std_dev_e_i = std(e_i);

disp(['Mean of e_i: ', num2str(mean_e_i)]);
```

```
Mean of e_i: 0.0082204
```

```
disp(['Standard deviation of e_i: ', num2str(std_dev_e_i)]);
```

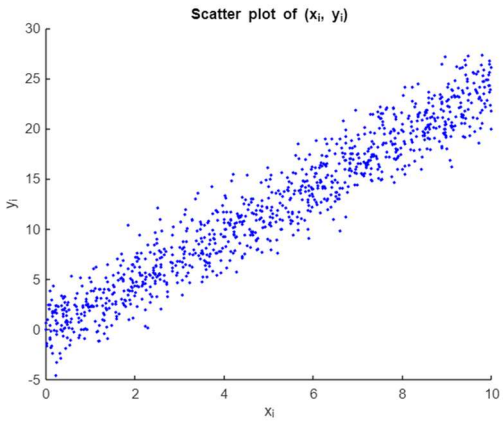
```
Standard deviation of e_i: 1.96
```

b)

```
% Generate 1000 uniformly distributed random numbers in the interval (0, 10)
x_i = 10 * rand(1000, 1);
% Compute yi = 2.4xi + εi
y_i = 2.4*x_i + e_i;
```

c)

```
% Plot the data points (x_i, y_i)
scatter(x_i, y_i, 'b.');
xlabel('x_i');
ylabel('y_i');
title('Scatter plot of (x_i, y_i)');
```




```
% Find the 10th largest number in y_i
sorted_y_i = sort(y_i, 'descend');
tenth_largest_y_i = sorted_y_i(10);

% Find the corresponding x_i for the 10th largest y_i
corresponding_xi_index = find(y_i == tenth_largest_y_i, 1);
corresponding_xi = x_i(corresponding_xi_index);

disp(['10th largest y_i: ', num2str(tenth_largest_y_i)]);
```

```
10th largest y_i: 26.1844
```

```
disp(['Corresponding x_i: ', num2str(corresponding_xi)]);
```

```
Corresponding x_i: 9.2182
```

d)

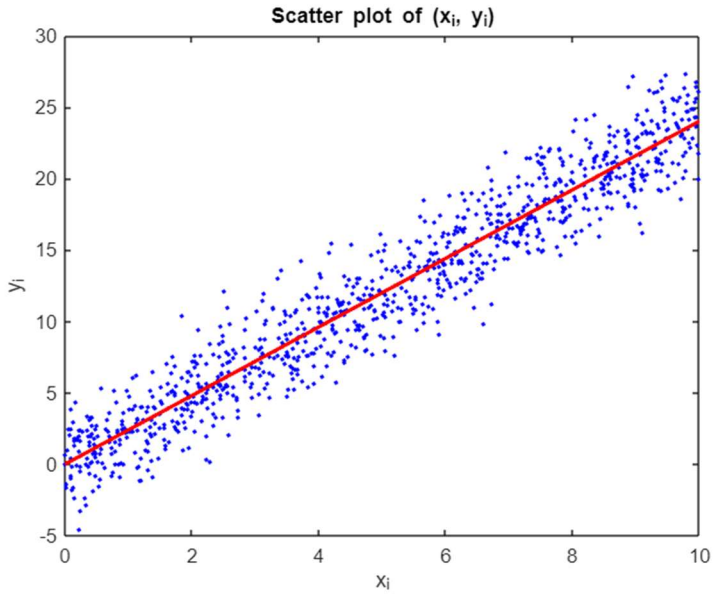
```
% Use least squares to find the optimal Beta
Beta_hat = (x_i' * x_i) \ (x_i' * y_i);

disp('Computed value of Beta:');
```

```
Computed value of Beta:
```

```
disp(Beta_hat);
```

```
    2.4016
```

```
% Plot the fitted curve
x_fit = linspace(0, 10, 1000); % Generate x values for fitting curve
y_fit = Beta_hat * x_fit;

hold on
plot(x_fit, y_fit, 'r-', 'LineWidth', 2);
```



Task 3

```
p1 = [-1, 0, 0, 0, 1];
poly2str(p1, 'x')
```

```
ans = '   -1 x^4 + 1'
```

```
roots_p1 = roots(p1)
```

```
roots_p1 = 4×1 complex
    -1.0000 + 0.0000i
     0.0000 + 1.0000i
     0.0000 - 1.0000i
     1.0000 + 0.0000i
```

```
factored_form_p1 = poly(roots_p1)
```

```
factored_form_p1 = 1×5
      1.0000    0.0000   -0.0000    0.0000   -1.0000
```

```
p2 = [-6, 11, -6, 1];
poly2str(p2, 'x')
```

```
ans = '   -6 x^3 + 11 x^2 - 6 x + 1'
```

```
roots_p2 = roots(p2)
```

```
roots_p2 = 3×1
      1.0000
      0.5000
      0.3333
```

```
factored_form_p2 = poly(roots_p2)
```

```
factored_form_p2 = 1×4
      1.0000   -1.8333    1.0000   -0.1667
```

```
p3 = [1, 2.3, 2.3, 0.9, 1.7, 2.8, 1];
poly2str(p3, 'x')
```

```
ans = '   x^6 + 2.3 x^5 + 2.3 x^4 + 0.9 x^3 + 1.7 x^2 + 2.8 x + 1'
```

```
roots_p3 = roots(p3)
```

```
roots_p3 = 6×1 complex
     0.6000 + 0.8000i
     0.6000 - 0.8000i
    -1.0000 + 1.0000i
    -1.0000 - 1.0000i
    -1.0000 + 0.0000i
    -0.5000 + 0.0000i
```

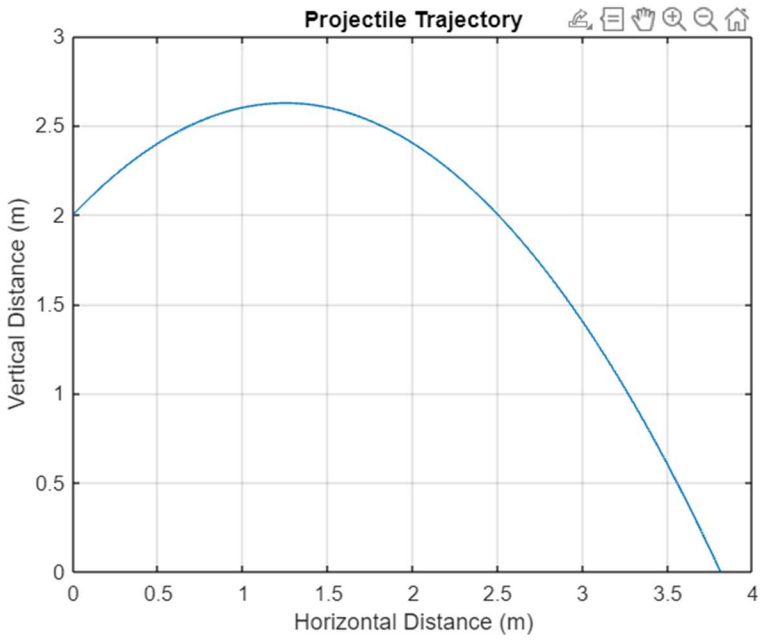
```
factored_form_p3 = poly(roots_p3)
```

```
factored_form_p3 = 1×7
      1.0000    2.3000    2.3000    0.9000    1.7000    2.8000    1.0000
```

TASK 4

```
function draw_trajectory(v, theta, x0, y0)
    g = 10; % Acceleration due to gravity (m/s^2)

    % Compute time of flight (T) using the vertical motion equation
    T = max(roots([-0.5*g, v*sind(theta), y0])); % maximum positive root

    % Time intervals for plotting
    t_intervals = linspace(0, T, 1000);

    % Compute x and y positions at each time interval
    x_positions = x0 + v * cosd(theta) * t_intervals;
    y_positions = y0 + v * sind(theta) * t_intervals - 0.5 * g * t_intervals.^2;

    % Plot the trajectory
    plot(x_positions, y_positions);
    xlabel('Horizontal Distance (m)');
    ylabel('Vertical Distance (m)');
    title('Projectile Trajectory');
    grid on;
end
```

```
v = 5; % Initial velocity (m/s)
theta = 45; % Launch angle (degrees)
x0 = 0; % Initial horizontal position (m)
y0 = 2; % Initial vertical position (m)

draw_trajectory(v, theta, x0, y0);
```



```
function v_required = find_required_speed(theta, xt, yt, y0)
    g = 10; % Acceleration due to gravity (m/s^2)

    % Convert launch angle from degrees to radians
    theta_rad = deg2rad(theta);

    % Compute the horizontal distance to the target (range)
    R = xt;

    % Compute the required initial velocity using the modified range formula
    v_required = sqrt(((R * g - (yt - y0)^2) / sin(2 * theta_rad)));
end
```

```
% Given values
theta = 45; % Launch angle (degrees)
xt = 10; % Target's x-coordinate (m)
yt = 3.5; % Target's y-coordinate (m)
y0 = 2; % Initial vertical position (m)

% Calculate the required speed
v_required = find_required_speed(theta, xt, yt, y0);
disp(['The required speed to hit the target: ', num2str(v_required), ' m/s']);
```

```
The required speed to hit the target: 9.8869 m/s
```

TASK 5

```
function x = logistic_map(r, x0, num_iterations)
    % Initialize array to store iterates
    x = zeros(1, num_iterations + 1);

    % Set initial condition
    x(1) = x0;

    % Iterate to compute x[k]
    for k = 1:num_iterations
        x(k + 1) = r * x(k) * (1 - x(k));
    end
end
```

```
% Given parameters
rs = [0.3, 1.8, 2.2, 2.5, 2.7];  % Growth rate parameters
x0 = 0.1; % Initial condition
num_iterations = 50; % Number of iterations

% Compute iterates for each value of r
for i = 1:length(rs)
    r = rs(i);
    x = logistic_map(r, x0, num_iterations);

    % Display the iterates
    disp(['r = ', num2str(r)]);
    disp(x);
end
```

```
r = 0.3
    0.1000    0.0270    0.0079    0.0023    0.0007    0.0002    0.0001    0.0000    0.0000    0.0000    0.0000    0.0000    0.00

r = 1.8
    0.1000    0.1620    0.2444    0.3324    0.3994    0.4318    0.4416    0.4439    0.4443    0.4444    0.4444    0.4444    0.44

r = 2.2
    0.1000    0.1980    0.3494    0.5001    0.5500    0.5445    0.5456    0.5454    0.5455    0.5455    0.5455    0.5455    0.54

r = 2.5
    0.1000    0.2250    0.4359    0.6147    0.5921    0.6038    0.5981    0.6010    0.5995    0.6002    0.5999    0.6001    0.60

r = 2.7
    0.1000    0.2430    0.4967    0.6750    0.5923    0.6520    0.6126    0.6407    0.6215    0.6351    0.6257    0.6323    0.61
```