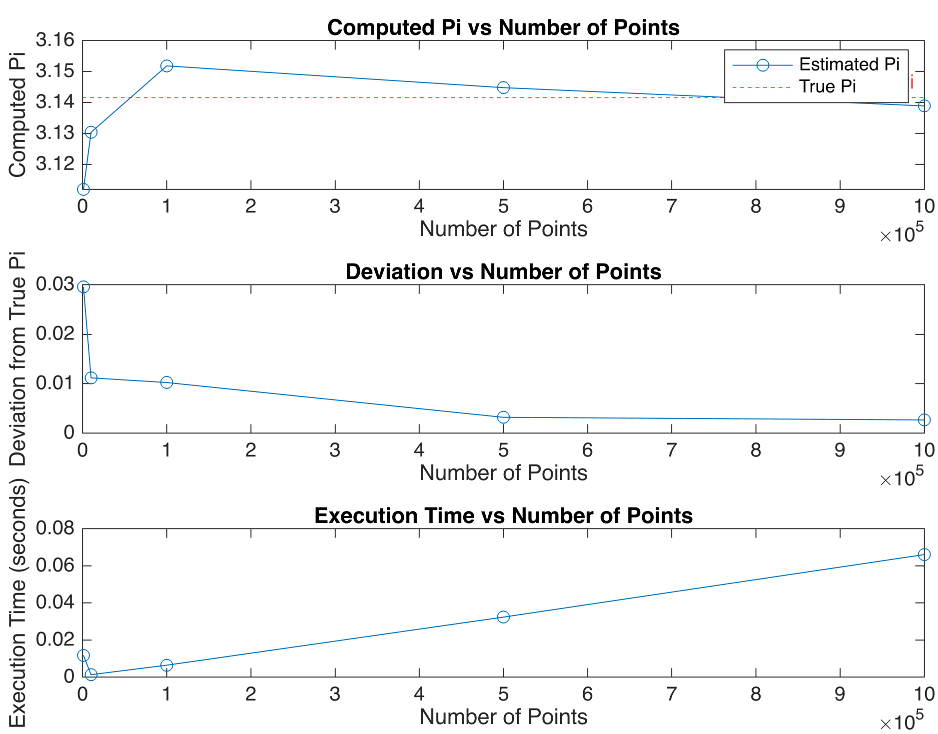
```
% Task 1: For loop use

% define variables
true_pi = pi;
N_values = [1000, 10000, 100000, 500000, 1000000];  % Different numbers of
random points
computed_pi = zeros(size(N_values));  % Array to store computed Pi values
deviation = zeros(size(N_values));  % Array to store deviations from true Pi
execution_times = zeros(size(N_values));  % Array for execution times

% For loop over each number of points in N_values
for j = 1:length(N_values)
    N = N_values(j);
    inside_circle = 0;

    % measuring execution time
    tic;

    % Generate random points and check if they are inside the quarter circle
    for i = 1:N
        x = rand;
        y = rand;
        if x^2 + y^2 <= 1
            inside_circle = inside_circle + 1;
        end
    end

    % Estimate Pi
    computed_pi(j) = 4 * inside_circle / N;

    % deviation from the true Pi value
    deviation(j) = abs(computed_pi(j) - true_pi);

    % Record execution time
    execution_times(j) = toc;

    % Print details for each iteration
    fprintf('N = %d, Computed Pi = %.10f, Deviation = %.10f, Execution Time
= %.5f seconds\n', ...
            N, computed_pi(j), deviation(j), execution_times(j));
end
```

```
N = 1000, Computed Pi = 3.1120000000, Deviation = 0.0295926536, Execution Time = 0.01176 seconds
N = 10000, Computed Pi = 3.1304000000, Deviation = 0.0111926536, Execution Time = 0.00135 seconds
N = 100000, Computed Pi = 3.1518400000, Deviation = 0.0102473464, Execution Time = 0.00647 seconds
N = 500000, Computed Pi = 3.1448080000, Deviation = 0.0032153464, Execution Time = 0.03239 seconds
N = 1000000, Computed Pi = 3.1389120000, Deviation = 0.0026806536, Execution Time = 0.06608 seconds
```

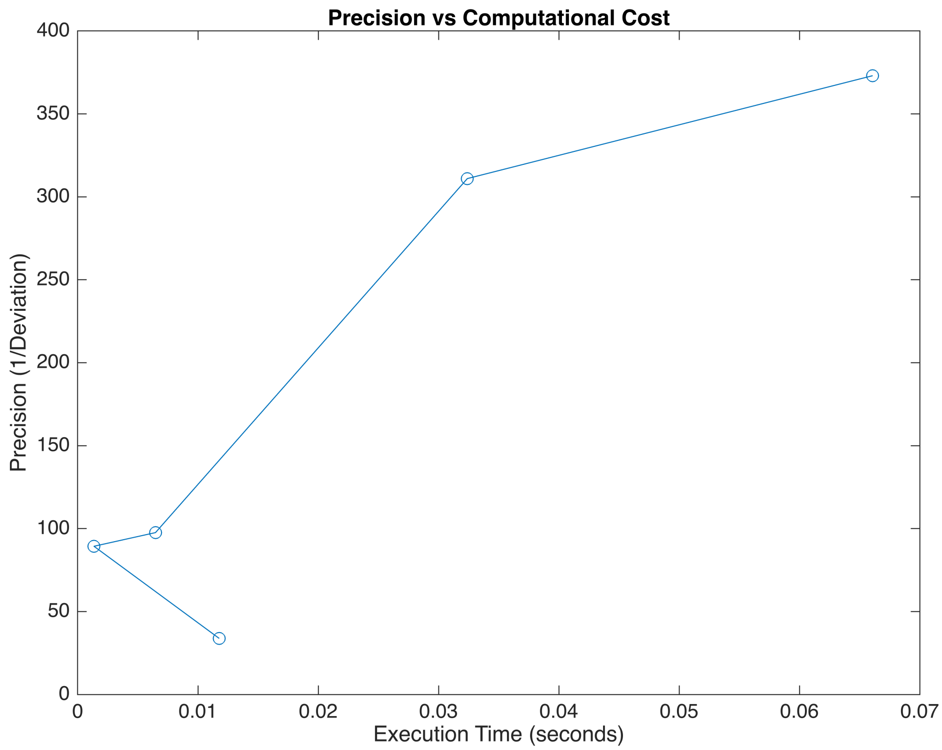
```
% Plot the computed value of Pi as the number of points increases
figure;
```

```
subplot(3,1,1);
plot(N_values, computed_pi, '-o');
hold on;
yline(true_pi, '--r', 'True Pi');
xlabel('Number of Points');
ylabel('Computed Pi');
title('Computed Pi vs Number of Points');
legend('Estimated Pi', 'True Pi');
hold off;

% Plot the deviation from true Pi as the number of points increases
subplot(3,1,2);
plot(N_values, deviation, '-o');
xlabel('Number of Points');
ylabel('Deviation from True Pi');
title('Deviation vs Number of Points');

% Plot the execution time as the number of points increases
subplot(3,1,3);
plot(N_values, execution_times, '-o');
xlabel('Number of Points');
ylabel('Execution Time (seconds)');
title('Execution Time vs Number of Points');
```




```
% Create a plot comparing precision (1/Deviation) to computational cost
(Execution Time)
```

```
figure;
plot(execution_times, 1./deviation, '-o');
xlabel('Execution Time (seconds)');
ylabel('Precision (1/Deviation)');
title('Precision vs Computational Cost');
```



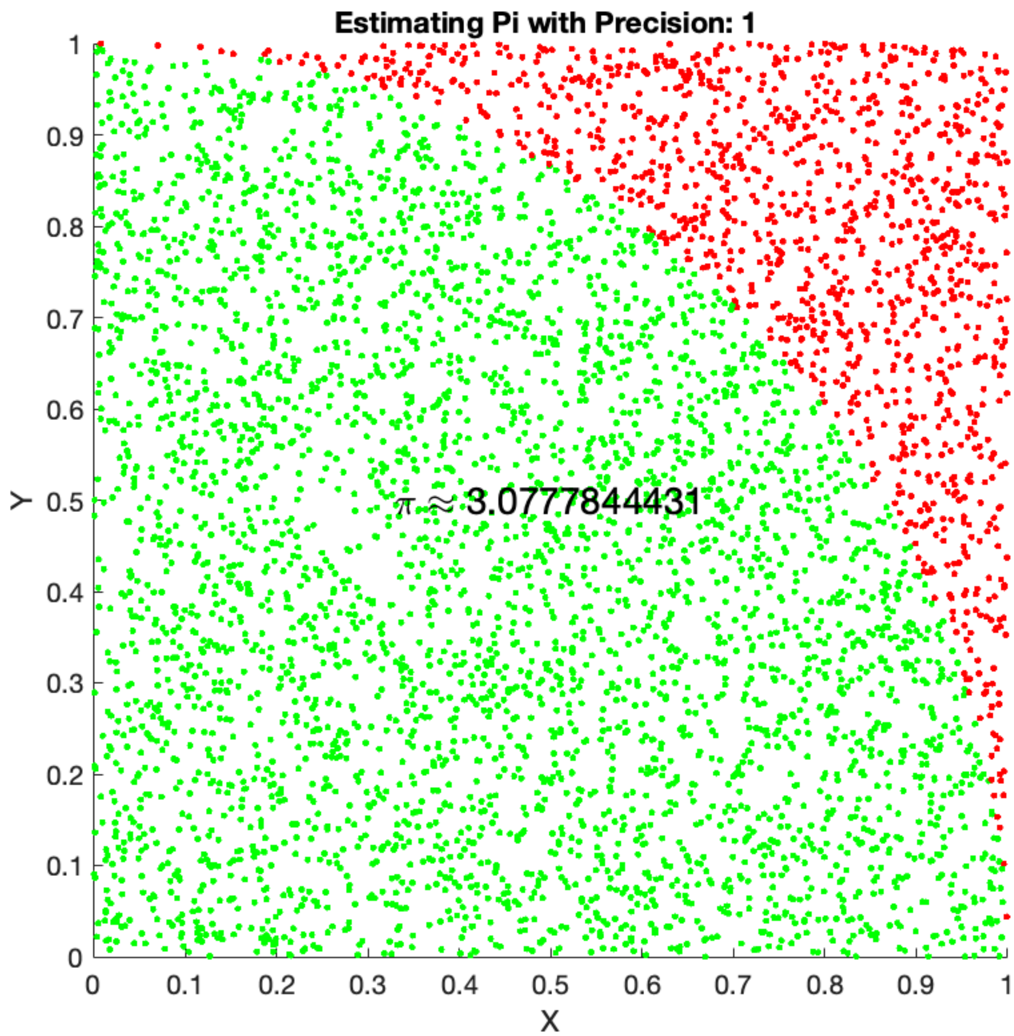
```
% Task 2: Compute Pi using While Loop for a Specified Precision

% Define the desired precision
precision_levels = [2, 3, 4];

% Initialize arrays to store results for each precision level
computed_pis = zeros(1, length(precision_levels));
iterations = zeros(1, length(precision_levels));
execution_times = zeros(1, length(precision_levels));

% for loop over each precision level
for k = 1:length(precision_levels)
    desired_precision = precision_levels(k);
    tolerance = 10^-(desired_precision);  % Tolerance based on desired
precision

    % Initialize variables
    inside_circle = 0;  % points inside the quarter circle
    N = 0;  % Number of points
    computed_pi = 0;  % Initial estimate of Pi
    previous_pi = NaN;  % To store the previous computed value of Pi
    min_iterations = 5000; % Minimum number of iterations to avoid
premature stopping

    % Start timing
    tic;

    % While loop to compute Pi
    while true
        N = N + 1;
        x = rand;
        y = rand;

        % Check if the point is inside the quarter circle
        if x^2 + y^2 <= 1
            inside_circle = inside_circle + 1;
        end

        % Estimate Pi based on the number of points
        previous_pi = computed_pi;  % Store the previous Pi value
        computed_pi = 4 * inside_circle / N;  % Monte Carlo estimation of Pi

        % Ensure a minimum number of iterations to avoid premature stopping
        if N > min_iterations
            % Check if the computed Pi value is stable to the desired
significant figures
            if abs(round(computed_pi, desired_precision) -
round(previous_pi, desired_precision)) < tolerance
                break;  % Exit the loop if the desired precision is achieved
            end
```

```
        end
    end

    % Stop timing
    execution_times(k) = toc;

    % Store the results for the current precision level
    computed_pis(k) = round(computed_pi, desired_precision);
    iterations(k) = N;

    % Display the results for the current precision level
    fprintf('Precision level: %d significant figures\n', desired_precision);
    fprintf('Computed Pi = %.10f\n', computed_pis(k));
    fprintf('Number of iterations = %d\n', N);
    fprintf('Execution time = %.5f seconds\n\n', execution_times(k));
end
```

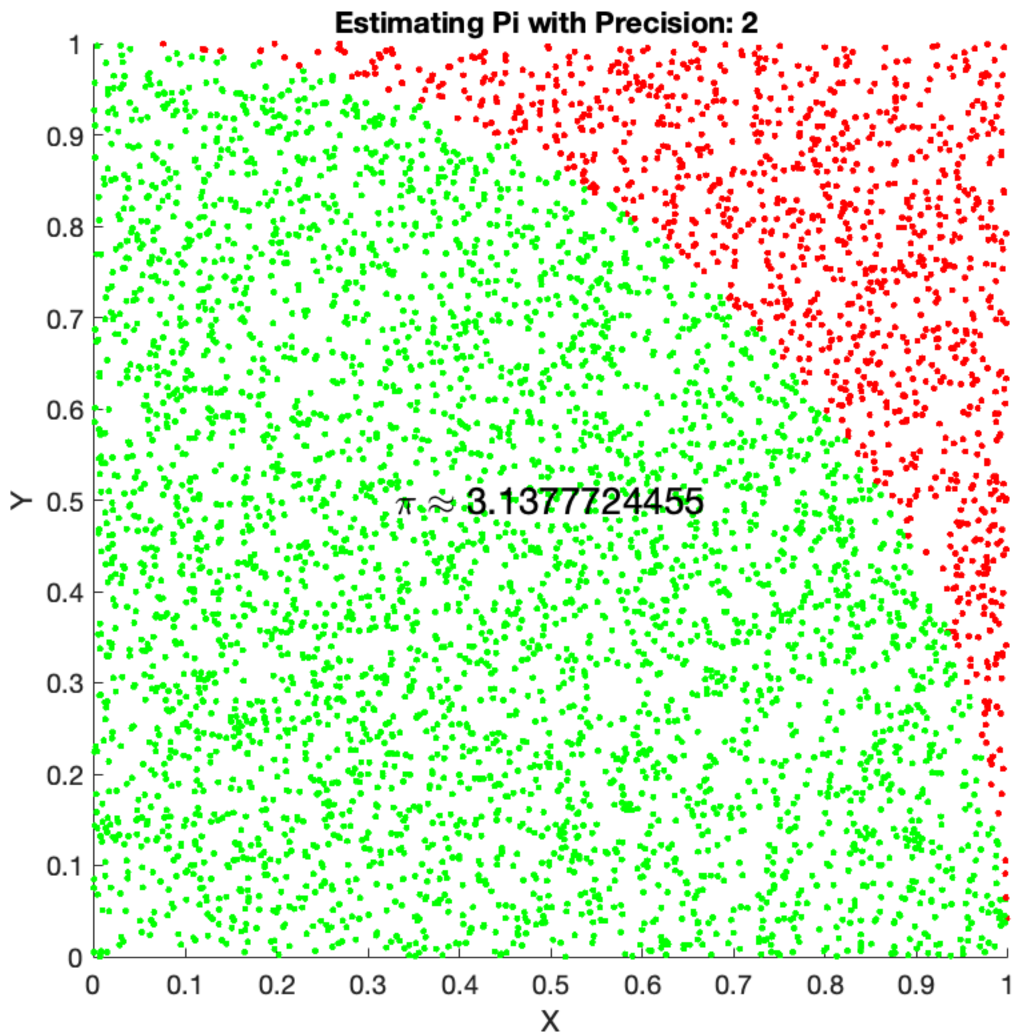
```
Precision level: 2 significant figures
Computed Pi = 3.1200000000
Number of iterations = 5001
Execution time = 0.00279 seconds
Precision level: 3 significant figures
Computed Pi = 3.1430000000
Number of iterations = 5001
Execution time = 0.00206 seconds
Precision level: 4 significant figures
Computed Pi = 3.1466000000
Number of iterations = 5006
Execution time = 0.00235 seconds
```

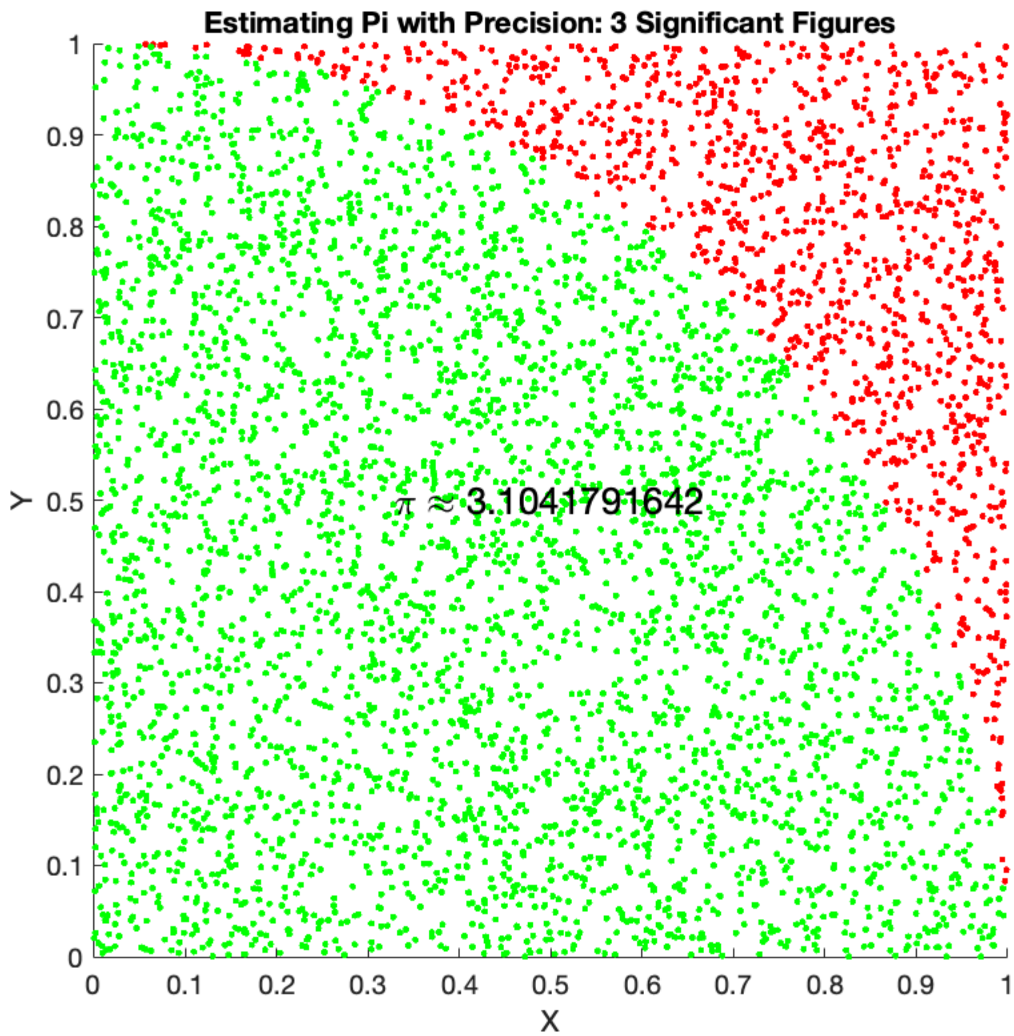
```
% Task 3: Compute Pi with User-Defined Precision
function computed_pi = compute_pi()

    % Ask user for the desired precision level
    desired_precision = input('Enter the desired precision level (for
example: 1 or 2 or 3): ');

    % Initialize variables
    tolerance = 10^-(desired_precision);  % Tolerance based on user-
specified precision
    inside_circle = 0;
    N = 0;
    computed_pi = 0;  % Initial estimate of Pi
    previous_pi = NaN;  % To store the previous computed value of Pi
    min_iterations = 5000;  % Minimum iterations to prevent early stopping

    % figure for graphical display
    figure;
    hold on;
    axis equal;
    axis([0 1 0 1]);
    title(sprintf('Estimating Pi with Precision: %d ', desired_precision));
    xlabel('X');
    ylabel('Y');

    % Start the Monte Carlo simulation
    tic;  % Start timer

    while true
        N = N + 1;
        x = rand;
        y = rand;

        % Check if the point is inside the quarter circle
        if x^2 + y^2 <= 1
            inside_circle = inside_circle + 1;
            plot(x, y, 'g.');  % Plot points inside the circle in green
        else
            plot(x, y, 'r.');  % Plot points outside the circle in red
        end

        % Estimate Pi based on the number of points
        previous_pi = computed_pi;  % Store the previous Pi value
        computed_pi = 4 * inside_circle / N;  % Monte Carlo estimation of Pi

        % Ensure a minimum number of iterations
        if N > min_iterations
            % Check if the computed Pi value is stable to the desired
significant figures
```

```
            if abs(round(computed_pi, desired_precision) -
round(previous_pi, desired_precision)) < tolerance
                break;  % Exit the loop if the desired precision is achieved
            end
        end


        if mod(N, 100) == 0
            drawnow;
        end
    end

    % Stop timer and calculate execution time
    execution_time = toc;

    % print the results
    fprintf('Computed Pi = %.10f\n', computed_pi);
    fprintf('Number of iterations = %d\n', N);
    fprintf('Execution time = %.5f seconds\n', execution_time);

    % final result on the plot
    text(0.5, 0.5, sprintf('\\pi \\approx %.10f', computed_pi), 'FontSize',
14, 'HorizontalAlignment', 'center');

end
```



Estimating Pi with Precision: 1
π ≈ 3.0777844431
Y
X
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Estimating Pi with Precision: 2
π ≈ 3.1377724455
Y
X
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1



Estimating Pi with Precision: 3 Significant Figures
$\pi \approx 3.1041791642$
X
Y
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1