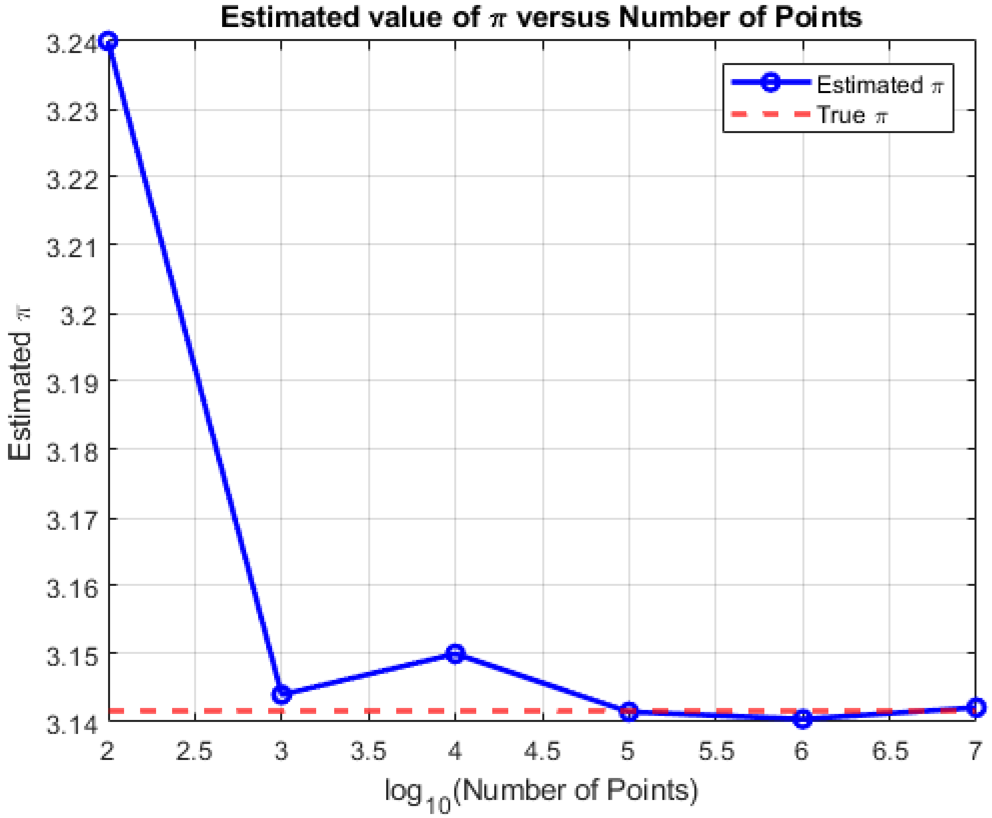
Table of Contents

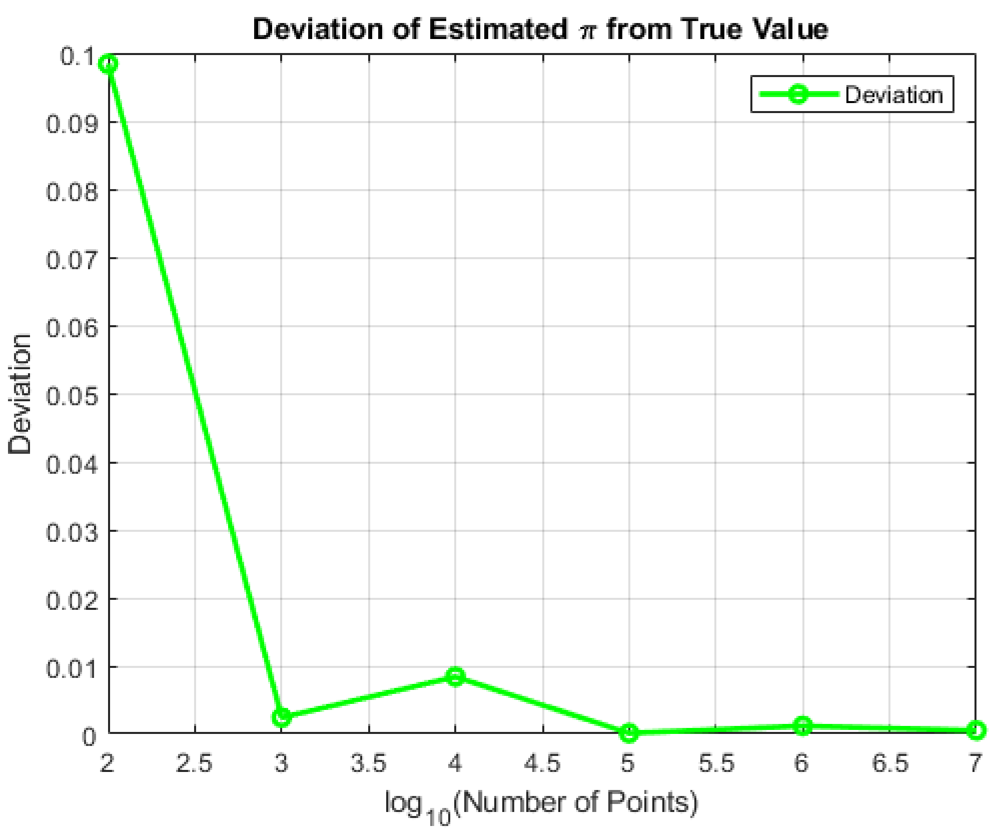

```
% AMS 595 - Assignment 1
% Amol Arora, SBUID: 116491705
```

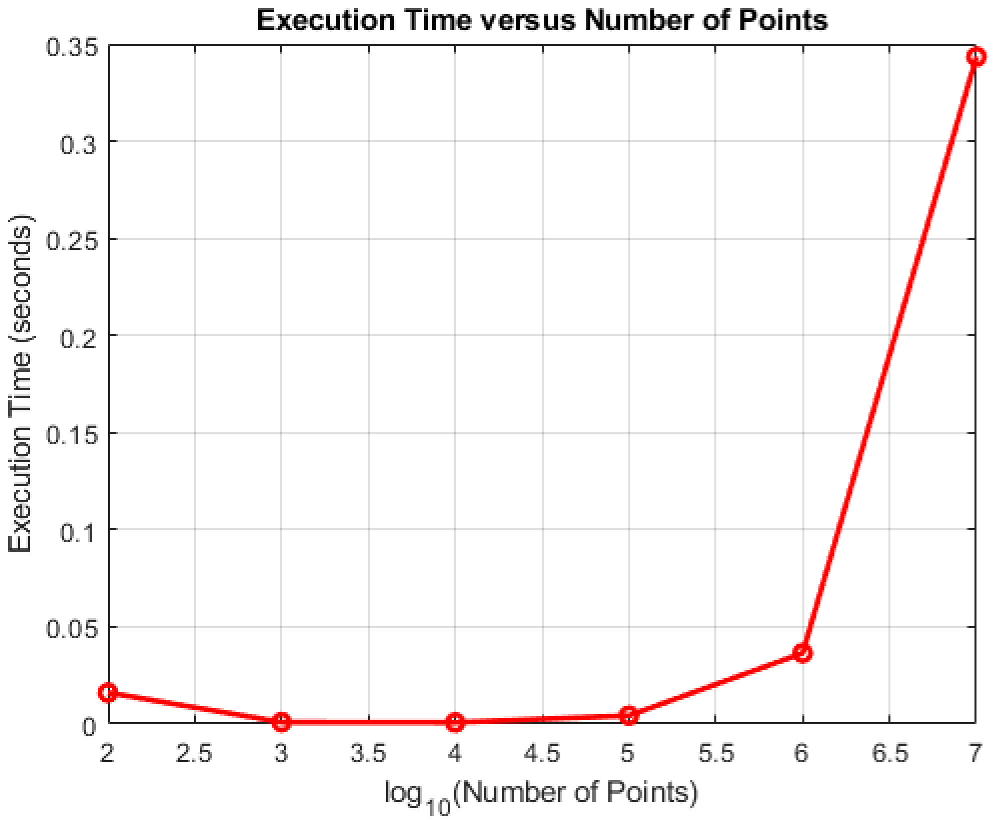
Part 1: Monte Carlo Estimation of π with for loop

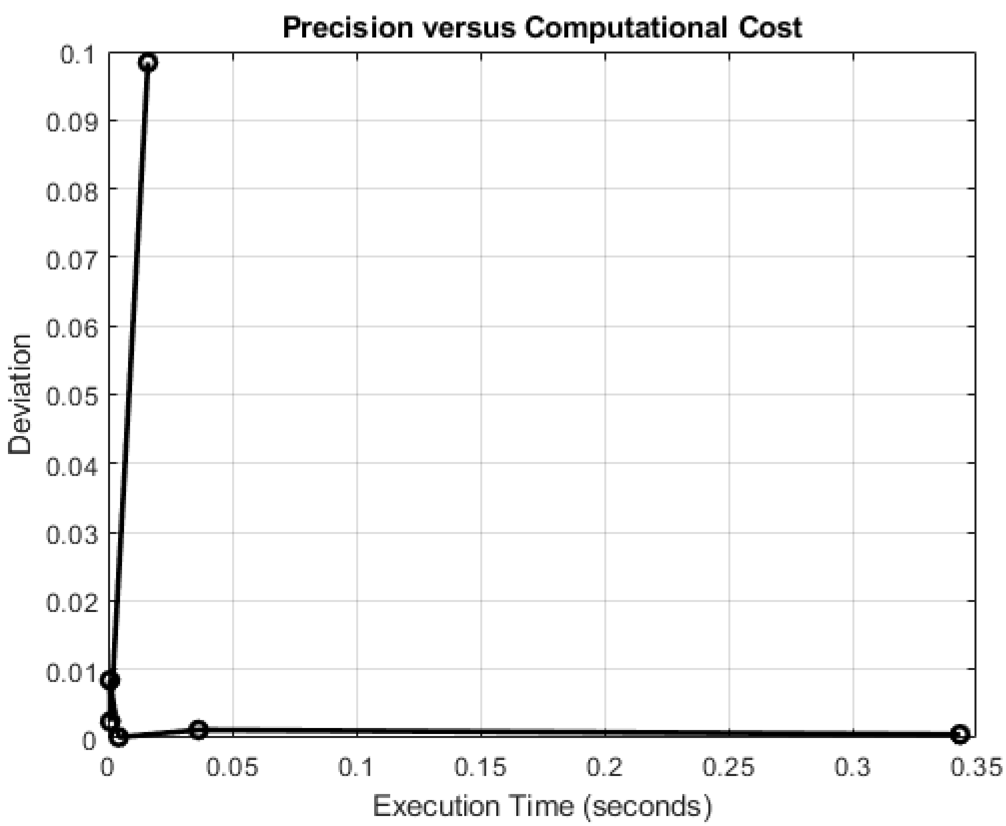
```
% I have taken points on the log-scale
% (More number of points will help to get closer to PI)
num_points = [10^2, 10^3, 10^4, 10^5, 10^6, 10^7];

% Initializing arrays
pi_estimates = zeros(size(num_points));
deviation = zeros(size(num_points));
time_taken = zeros(size(num_points));

true_pi = pi; % The true value of π for comparison.

% Loop through each number of points.
for i = 1:length(num_points)
    N = num_points(i);

    % Start timing
    tic;

    % Using the rand function to generate random points (x, y) between [0, 1]
    x = rand(1, N);
    y = rand(1, N);

    % This will check if points fall within the quarter circle
    inside_circle = (x.^2 + y.^2) <= 1; % Similar to conditinoal selection

    % Estimating π based on the ratio of points inside the circle
    pi_estimates(i) = 4 * sum(inside_circle) / N;

    % Stop timing
    time_taken(i) = toc;

    % Computing deviation from the true value of π
    deviation(i) = abs(pi_estimates(i) - true_pi);
end

% I will be saving (using 'saveas' command) the figures,
```

```
% so that it can be included in the report and github.
figure;
plot(log10(num_points), pi_estimates, 'b-o', 'LineWidth', 2);
hold on;
yline(true_pi, '--r', 'LineWidth', 2);
title('Estimated value of \pi versus Number of Points');
xlabel('log_{10}(Number of Points)');
ylabel('Estimated \pi');
legend('Estimated \pi', 'True \pi');
grid on;
saveas(gcf, 'Result_Files/pi_estimation_plot.png');

figure;
plot(log10(num_points), deviation, 'g-o', 'LineWidth', 2);
title('Deviation of Estimated \pi from True Value');
xlabel('log_{10}(Number of Points)');
ylabel('Deviation');
legend('Deviation');
grid on;
saveas(gcf, 'Result_Files/pi_deviation_plot.png');

figure;
plot(log10(num_points), time_taken, 'r-o', 'LineWidth', 2);
xlabel('log_{10}(Number of Points)');
ylabel('Execution Time (seconds)');
title('Execution Time versus Number of Points');
grid on;
saveas(gcf, 'Result_Files/execution_time_plot.png');

figure;
plot(time_taken, deviation, 'k-o', 'LineWidth', 2);
xlabel('Execution Time (seconds)');
ylabel('Deviation');
title('Precision versus Computational Cost');
grid on;
saveas(gcf, 'Result_Files/precision_vs_cost_plot.png');
```


Estimated value of π versus Number of Points
Estimated π
True π
Estimated π
3.24
3.23
3.22
3.21
3.2
3.19
3.18
3.17
3.16
3.15
3.14
2
2.5
3
3.5
4
4.5
5
5.5
6
6.5
7
$\log_{10}$(Number of Points)


Deviation of Estimated π from True Value
Deviation
Deviation
0.1
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
2
2.5
3
3.5
4
4.5
5
5.5
6
6.5
7
$\log_{10}$(Number of Points)

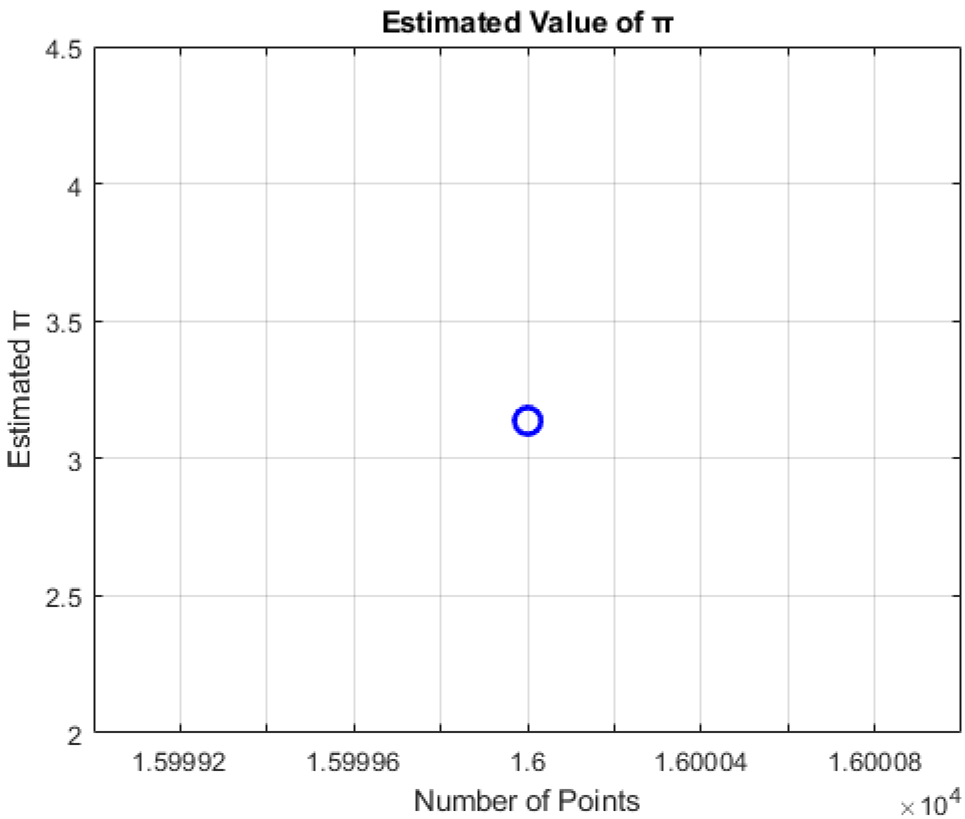

Execution Time versus Number of Points
Execution Time (seconds)
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
2
2.5
3
3.5
4
4.5
5
5.5
6
6.5
7
log10(Number of Points)


Precision versus Computational Cost
Deviation
0.1
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
Execution Time (seconds)

Part 2: Estimating π Until Required Precision is Met

```
% Set parameters for the precision-based estimation.
required_precision = 1e-3;
initial_points = 10^3;
points_increment = 10^3;

% To prevent an infinite loop, I have set a maximum number of iterations.
max_iterations = 1000;
num_points = initial_points;
prev_pi_estimate = 0;
deviation = Inf; % Good Starting piont.
iteration_count = 0;

% Continue estimating π until the deviation is below the required precision
or the max iterations are reached.
while deviation > required_precision && iteration_count < max_iterations
    % Generating random points (x, y) between [0, 1]
    x = rand(1, num_points);
    y = rand(1, num_points);

    % Checking if points fall within the quarter circle
    inside_circle = (x.^2 + y.^2) <= 1;

    % Estimate π and calculate deviation from previous estimate.
    pi_estimate = 4 * sum(inside_circle) / num_points;
    if iteration_count > 0
        deviation = abs(pi_estimate - prev_pi_estimate);
    end

    % Update parameters for the next iteration.
    prev_pi_estimate = pi_estimate;
    num_points = num_points + points_increment;
    iteration_count = iteration_count + 1;
end

% Print the final results.
fprintf('Estimated value of π: %.6f\n', pi_estimate);
fprintf('Number of points used: %d\n', num_points);
fprintf('Number of iterations: %d\n', iteration_count);
fprintf('Final deviation: %.6f\n', deviation);

% Save the final estimation plot.
figure;
plot(num_points - points_increment, pi_estimate, 'bo', 'MarkerSize', 10,
'LineWidth', 2);
title('Estimated Value of π');
xlabel('Number of Points');
ylabel('Estimated π');
grid on;
saveas(gcf, 'Result_Files/pi_estimation_precision_plot.png');
```

```
Estimated value of π: 3.137000
Number of points used: 17000
Number of iterations: 16
Final deviation: 0.000733
```



Part 3: Monte Carlo Estimation Visualization with Simulation

I have published only the function for this, since it requires user input to run.

```
% Define a function for visualizing the estimation process with a GIF.
function pi_estimate = pi_estimate_visualize(target_precision, gif_filename)
    if target_precision <= 0
        error('Target precision must be a positive value.');
    end

    num_points = 0;
    previous_pi_estimate = 0;
    iteration_count = 0;
    deviation = Inf;

    % Create a new figure for plotting the simulation.
    figure;
    hold on;
    title('Monte Carlo Estimation of π');
```

```
    xlabel('x');
    ylabel('y');
    axis([0 1 0 1]);
    grid on;

    if nargin < 2
        gif_filename = 'monte_carlo_simulation.gif';
    end

    % Run the simulation until the desired precision is achieved.
    while deviation > target_precision
        num_points = num_points + 10^3;
        x = rand(1, num_points);
        y = rand(1, num_points);
        inside_circle = (x.^2 + y.^2) <= 1;
        plot(x(inside_circle), y(inside_circle), 'g.', 'MarkerSize', 5);
        plot(x(~inside_circle), y(~inside_circle), 'r.', 'MarkerSize', 5);
        pi_estimate = 4 * sum(inside_circle) / num_points;
        deviation = abs(pi_estimate - previous_pi_estimate);
        previous_pi_estimate = pi_estimate;
        iteration_count = iteration_count + 1;

        % Annotate the plot with the current π estimate.
        text(0.5, 0.9, sprintf('Estimated \\pi: %.6f', pi_estimate),
'HorizontalAlignment', 'center', 'FontSize', 12, 'BackgroundColor', 'white');

        % Capture the plot as an image for the GIF.
        frame = getframe(gcf);
        img = frame2im(frame);
        [imind, cm] = rgb2ind(img, 256);
        if iteration_count == 1
            imwrite(imind, cm, gif_filename, 'gif', 'LoopCount', inf,
'DelayTime', 0.1);
        else
            imwrite(imind, cm, gif_filename, 'gif', 'WriteMode', 'append',
'DelayTime', 0.1);
        end

        % Pause briefly to visualize the plot update.
        pause(0.01);
    end

    % Print final results of the visualization.
    fprintf('Estimated value of π: %.6f\n', pi_estimate);
    fprintf('Number of points used: %d\n', num_points);
    fprintf('Number of iterations: %d\n', iteration_count);
    fprintf('Final deviation: %.6f\n', deviation);

    % Round the final estimate to the desired precision.
    pi_estimate = round(pi_estimate, -log10(target_precision));
end

% Example call to the visualization function (uncomment to run):
% pi_estimate_visualize(1e-3, 'Result_Files/monte_carlo_visualization.gif');
```

Published with MATLAB® R2024a