AMS595 Project 1

Zhuoyi Chen

September 15, 2024

Part 1

As explained in the description, the Monte Carlo method can be used to estimate the value of π by generating random points within the square $[0, 1] \times [0, 1]$. One can use the distance of each point to the origin to determine with the point is within the quarter of a circle. Afterwards, the value of π can be estimated by

$$\pi \approx 4 \cdot \frac{\text{number of points within the quarter of a circle}}{\text{total number of points}}$$

In part 1, I used the number of points varied from 10^2 to 10^7 with the increase of the multiplication of 10. Note that the experiment of 10 points is omitted because the number of points is too small such that the estimated value is far from the true value of π. To measure the estimated deviation from the true value, I used the absolute error

$$|\text{estimated } \pi - \pi|$$

From the output, we can see that the estimated π value is less deviated from the true value of π as the number of points increases. As expected, the running time increases and the absolute value decreases as the number of points increases. However, from my observations, the estimated π value can be closer to the true value of π in some experiments. This is due to the fact that Monte Carlo is a randomized algorithm, and the result cannot be guaranteed. In any case, the tendency is described as above.

Part 2

In part 2, I design a variable called *graph_ind*. As explained in the comment in the script, If the variable $graph\_ind = true$, the program plots the points as the points are being generated. However, it takes a long time to generate, as the points are plotted within the for loop. To test the effect, please set $graph\_ind = true$. If the variable $graph\_ind = false$, the program plots all the points after all points are generated. In the published pdf, the variable *graph_ind* is set to be *false*.

Using a similar algorithm above, now instead I use a while-loop to estimate the value of π. Without knowing the true value of π, I calculate the error by the absolute value of the subtraction of the current estimated π value and the previous estimated π value. The step of each estimation is done after 100^k loops where k is increased by one after every checking. To illustrate, the errors are the following sequence

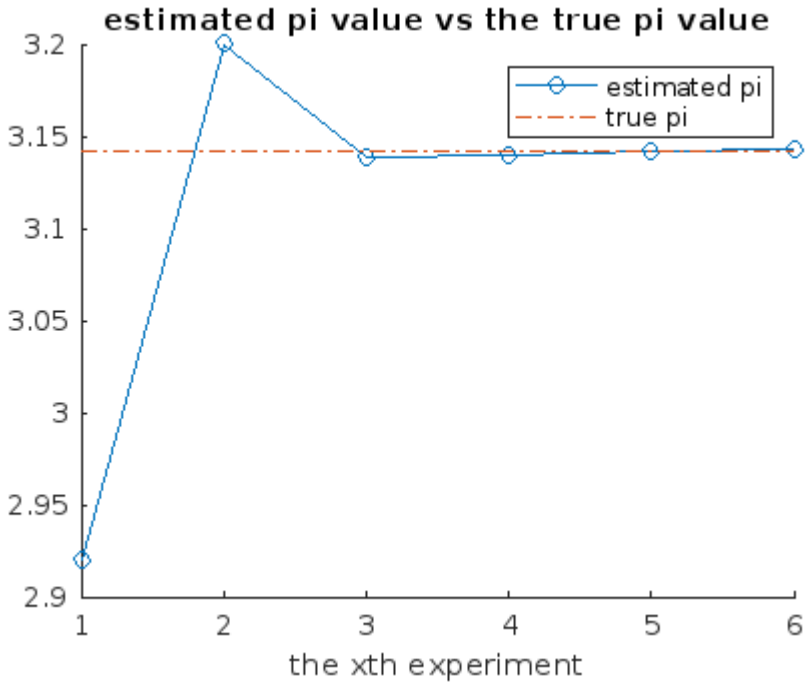
$$\begin{aligned}\text{error} =& |\text{estimated } \pi \text{ for 100 points} - 0|,\\ & |\text{estimated } \pi \text{ for } 100^2 \text{ points} - \text{estimated } \pi \text{ for 100 points}|,\\ & |\text{estimated } \pi \text{ for } 100^3 \text{ points} - \text{estimated } \pi \text{ for } 100^2 \text{ points}|, \ldots\end{aligned}$$

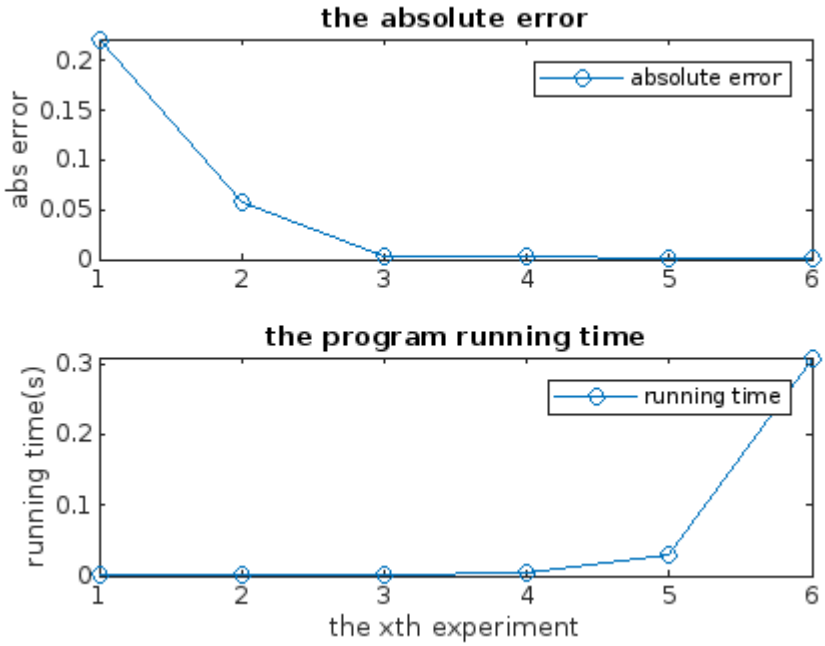
As the Monte Carlo algorithm converges, the above error calculation ensures the required significant figures.

I also construct a matrix of $[x,\ y,\ z]$ where (x, y) is the coordinate of the random generated point, and $z \in \{0, 1\}$ is the indicator value whether the point is within the circle. If the variable $graph\_ind = false$, I will use the sort method respective to the indicator value and plot all the points relatively quickly.

```
%%%%%%%%%%% Question 1
% k is the number of random points, exp is the number of experiences
% conducted, and required_time and estimated_pi all record the desired
% values for further uses.
exp = 6;
num_exp = (1:exp);

% skipping the experiment using 10 points.
k = 10.^(num_exp+ones(1,exp));
required_time = zeros(1,exp);
estimated_pi = zeros(1,exp);

% for loop record the time and the estimated pi for each experiment as the
% number of point is increased by multiplying 10.
for i = 1:exp
    tic
    est_pi = monte_carlo_est_pi(k(i));
    required_time(i) = toc;
    estimated_pi(i) = est_pi;
end

true_pi = pi*ones(1,exp);
error = abs(true_pi - estimated_pi);

figure
hold on
plot(num_exp,estimated_pi,'-o')
plot(num_exp,true_pi,'-.')
title('estimated pi value vs the true pi value')
legend('estimated pi','true pi')
xlabel('the xth experiment')
hold off

figure
tiledlayout(2,1)

hold on
% plot the error value
ax1 = nexttile;
plot(ax1,num_exp,error,'-o')
title(ax1,'the absolute error')
ylabel(ax1,'abs error')
legend(ax1,'absolute error')
hold off

hold on
% plot the program running time
ax2 = nexttile;
plot(ax2,num_exp,required_time,'-o')
title(ax2,'the program running time')
ylabel(ax2,'running time(s)')
xlabel(ax2, 'the xth experiment')
```

```
legend(ax2,'running time')
hold off
```



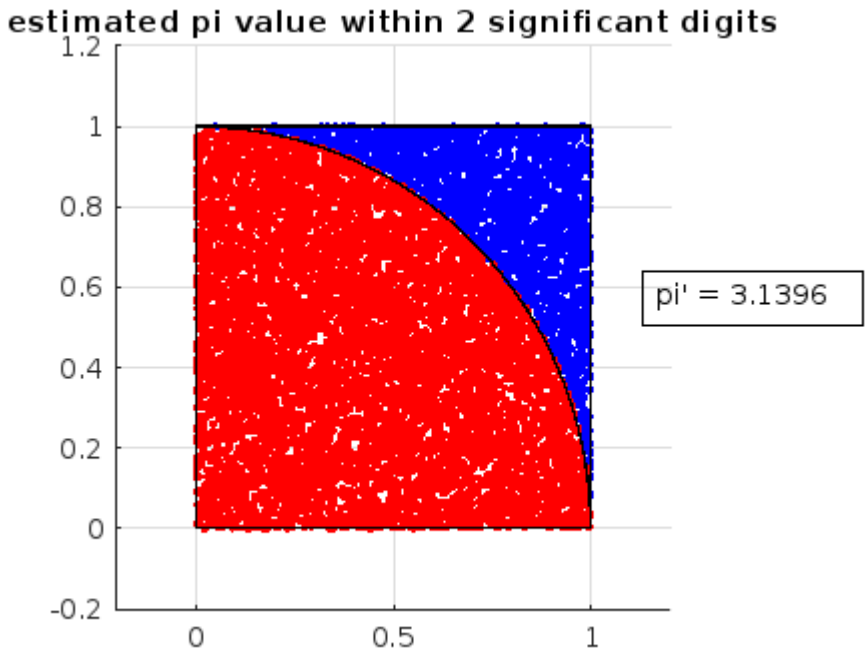


Published with MATLAB® R2024a

```
%%%%%%%%%%% Question 2
% digit: the number of significant to estimate the value of pi

digit = 2;

% If the variable graph_ind = true, the program plots the points as the
% points are being generated. However, it takes a long time to generate. To
% test the effect, please set graph_ind = true. If the variable graph_ind =
% false, the program plots all the points after all points are generated.

graph_ind = false;
[matrix, estimated_pre_pi, count, i] =
monte_carlo_pre_est_pi(digit,graph_ind);

if graph_ind == true
    % the graph has been generated. Output the result
    fprintf(['The estimated pi value with the precision of within %d ' ...
    'significant is %f'], digit, estimated_pre_pi)
else
    % plot the graph manually
    % sort the matrix by the indicator
    matrix_sort = sortrows(matrix,3,'descend');

    % divide the matrix into two, one contains all the points outside the
    % circle, and one contains the points within the circle.
    matrix_out = matrix_sort(1:count,:);
    matrix_in = matrix_sort(count+1:i,:);

    figure
    title(['estimated pi value within ', num2str(digit), ' significant
digits'])

    hold on
    xlim([-0.2, 1.2]);
    ylim([-0.2, 1.2]);

    plot(matrix_in(:,1),matrix_in(:,2),'b.')
    plot(matrix_out(:,1),matrix_out(:,2),'r.')

    % get the current axis;
    a = gca;

    % set the width of the axis (the third value in Position)
    % to be 60% of the Figure's width
    a.Position(3) = 0.6;

    % Draw a square
    sq_x = [0,1,1,0,0];
    sq_y = [0,0,1,1,0];
    plot(sq_x, sq_y, 'k-', 'LineWidth', 1);

    % Draw a quarter of a circle
```

```
    th = 0:pi/50:pi/2;
    xunit = cos(th) ;
    yunit = sin(th) ;
    plot(xunit, yunit,'k', 'LineWidth', 1);

    % put the textbox at 70% of the width and
    % 10% of the height of the figure
    annotation('textbox', [0.7, 0.5, 0.1, 0.1], 'String', ...
        "pi' = " + estimated_pre_pi)
    grid on

    fprintf(['The estimated pi value with the precision of within %d ' ...
    'significant is %f'], digit, estimated_pre_pi)
end
```

```
The estimated pi value with the precision of within 2 significant is 3.139600
```



Published with MATLAB® R2024a