Task 6 Estimate π Using Monte Carlo

Adrian Jonsson Sjödin

Spring Term 2023

Introduction

The task in this exercise is to implement a Monte Carlo simulation to estimate the value of π, and to improve the accuracy of the estimated value of π by throwing more and more darts and continuously estimate a value for π. The task description provides skeleton code for a function `dart/1` that throws a random dart and returns true or false depending on whether the dart hits inside an arch with a given radius. The task description also provides the skeleton code for a function `round/3` that throws a number of darts on a target with a given radius and counts how many darts land inside the arch. The task is to implement a function called `rounds/3` that runs a number of rounds, and displays the estimated value of pi after each round, comparing the estimate to `:math.pi()` to see how far off the the real (or elixirs) value of π the estimate is. The challenge is to beat the estimates found by Archimedes (two decimals) and Zu Chongzhi (six decimals).

Method

The goal of this task was to estimate the value of π using the Monte Carlo method. This meant simulating the throwing of darts at a target with a given radius and then count how many landed inside an arch with that radius. By repeating this process with more and more darts we can get an estimate for the value of π.

The Monte Carlo method itself consist of randomly generate points inside of a square and then see how many of them land inside an arch with in the square. The ratio of the number of points that fall inside the circle to the total number of points generated is an estimate of the ratio of the area of the circle to the area of the square. Using the formulas for the areas of the square and the circle, we can derive the following expression:

$$\pi = 4 \cdot \frac{\text{number of points inside circle}}{\text{total number of points}}$$

As stated in the introduction the task included code skeleton for the three functions `dart/1`, `round/3`, and `rounds/3`, and these where the ones we implemented to solve the problem.

The `dart/1` function simulates the throwing of a single dart at a target with a given radius, and returns true if the dart lands inside the arch, and false otherwise. This is done using Pythagoras Theorem.

The `round/3` function simulates a round of dart-throwing by generating a given number of darts and counting how many land inside the arch. This is done with a simple `if else` statement in which we call the `dart/1` function to see if we should increment the accumulator or not.

The `rounds/3` function is used to run multiple rounds of dart-throwing, and displays the estimated value of π after each round, along with the difference from the real value of π. The function uses different values for the number of rounds, and number of darts per round. The code can be seen in code overview 1, and the full code with explaining documentation and comments can be seen on my GitHub.

Code Overview 1: `rounds/5`

```
def rounds(k, j, r) do
    rounds(k, j, 0, r, 0)
end
defp rounds(0, _, t, _, a) do
    IO.puts("Estimate: #{4.0 * a / t}")
end
defp rounds(k, j, t, r, a) do
    a = round(j, r, a)
    t = t + j
    pi = 4.0 * a / t
    IO.puts("Estimate: #{pi} (error: #{pi - :math.pi()})")
    rounds(k-1, j*2, t, r, a)
end
```

Result

We ran the `rounds/3` function with various parameter values to generate estimates for the value of π. In table 1 you can see the result of the closest estimate we achieved.

Rounds	**Darts per Round**	**Target Radius**	**Estimate of π**
10	100,000	10,000	3.142112
10	1,000,000	100,000	3.141577

Table 1: Estimated values of π using different parameter values.

As shown in table 1, our estimates for the value of π were accurate to 4 decimal places in both cases. This is better than the estimate obtained by Archimedes (accurate to 2 decimal places), but worse than Zu Chongzhi (accurate to 6 decimal places).

Discussion

Our result shows that the Monte Carlo method can be utilized to estimate π to at least four decimals. The limitation of this approach is however that it is computationally intensive. Running multiple rounds of dart-throwing with a large number of darts per round was very time consuming and is why I choose to keep the rounds low, and instead have more darts and a larger radius.

Furthermore, the accuracy of our estimates depends all three parameters, so finding the optimal parameter values is a trial-and-error process. You could most likely use this method to estimate π to six, or even more decimal places, but because of the above stated reasons I decided to stop when I achieved four decimal precision. It would simply take to long to be worth it to use this method to achieve a higher precision, and that time would be better spent trying to implement another solution. However if a accuracy of just two decimal is enough then this could be a worthwhile solution, and if nothing else it proves that the theory behind the Monte Carlo method is correct.