Estimating π - Report

Adrian Hjert

23/Feb/2023

Introduction

The aim of this report is to explore how we can estimate the value of π primarily using the Monte Carlo method which involves randomly throwing darts at a square surface with an arch drawn in. Furthermore, a short exploration of the Liebniz formula, though the code for this is entirely given in the instructions.

Methods

First, we need to create our **dart/1** function. The following code shows how we compare x^2 and y^2 to r^2.

```
def dart(r) do
  x = :rand.uniform()
  y = :rand.uniform()
  :math.pow(r, 2) > (:math.pow(x, 2) + :math.pow(y, 2))
end
```

This bit of code will generate a random x and y value and return *true* or *false* based on if r^2 is larger or smaller than the sum of the squares of the x and y values. A *true* result indicates that we have hit within the arch and a *false* result indicates the opposite, namely that we have missed. Now, we wish to set up a system of rounds so we can accumulate the number of hits and thus make an estimate of the value of π. To do this we must think of some different cases. Of course, the first one that comes to mind is that we have no rounds. This has the following code.

```
def round(0, _r, hits) do hits end
```

After this, we can simply use the **dart/1** function to count the number of of hits each individual round. This can be seen in the following **round/3** function.

```
 def round(rds, r, hits) do
      if dart(r) do
  round(rds-1, r, hits+1)
else
  round(rds-1, r, hits)
  end
  end
```

Now, we wish to set up a function where we can run multiple rounds. This function, **rounds/5** can be partially seen in the following code. This is to make it easier to follow along in the explanation.

```
 def rounds(rds, ttr, r) do
    rounds(rds, ttr, 0.0, r, 0)
    end
    def rounds(0, _ttr, total, _r, hits) do
    IO.puts(" Estimate: #{4.0 * hits/total}") end

    def rounds(rds, ttr, total, r, hits) do
    hits = round(ttr, r, hits)
    total = total + ttr
    pi = 4.0 * hits/total
...
end
```

This part, which takes 5 parameters, namely *rds* which is the number of rounds to be completed, *ttr* which is the nmber of throws this round, *total* which is the total number of darts thrown so far, r which is the radius of the arch, and finally, *hits* which is the number of successful hits. The function works by calling the **round/3** function described previously to throw a specified number of darts at the arch and count how many darts land within it. This part then calls itself recursively until the specified number of rounds have been completed and further accumulates the number of darts that has hit inside the arch while simultaneously doubling *ttr* (seen later in this section) and uses the results to estimate the value of π. The next part of this code is mostly responsible for formatting and comparing the results. This can be seen in the following code.

```
     ...
    :io.format("  total = ~12w, pi = ~14.10f,
     dp = ~14.10f, da = ~8.4f,
     dz = ~12.8f\n", [total, pi, (pi - :math.pi()),
    (pi - 22/7), (pi- 355/113)])

    rounds(rds-1, ttr*2, total, r, hits)
    end
```

This part of the code presents the results where the total number of throws are displayed along with that round's estimation of π along with the absolute, percentage, and rounded differences. This code also shows the call where the throws per round is doubled.

The final part of the implementation to estimate π is to use the Liebniz function. This code is entirely provided in the instructions so only its results will be discussed in the **Results and Conclusions** section.

Results and Conclusions

Some different values for number of rounds, throws per round, and radius were tested to find a reasonable estimate for π. The call

```
MCarlo.rounds(5, 10_000, 1)
```

gave an estimation correct to 3 decimal places. A call that was correct to 3-4 decimal places was

```
MCarlo.rounds(10, 100_000, 1)
```

This took some time however, in the order of minutes but still less than 10 minutes. A call that took over 10 minutes but gave a correct estimation to 4-5 decimal places was

```
MCarlo.rounds(10, 1_000_000, 1)
```

This gave an estimate of 3.141573. Note, there are more decimals but they were omitted as it the estimation is not correct past the 5.

This is obviously not quite sustainable if we want to get a better estimate as we could potentially be waiting for hours to get a good value past perhaps 6 decimal places. Instead, we can use the Liebniz function which calculates π accurate to 8 decimal places in well under a second. Much better.