# Monte Carlo and $\pi$

Sareh Jalalizad

Spring Term 2023

## Introduction

For this assignment, we were supposed to calculate an approximation of $\pi$ using the Monte Carlo method. The Monte Carlo method is a popular approach to estimating pi, where random points (darts) are generated within a square and the number of points within a circle inscribed in that square are counted. The ratio of the number of points inside the circle to the total number of points can be used to estimate the value of pi and the formula is $\pi = 4$ * (the number of points (darts) inside the circle/total number of points within the square).

## Method

To estimate the value of $\pi$ using the Monte Carlo method in Elixir, we first need to define a function that can throw a random dart and tell us whether it hit inside the circle or not. This function, called dart/1, takes a radius r as input and returns true if the dart lands within the circle, and false otherwise and we can use the comparison $r^2 > x^2 + y^2$ for that.

```
def dart(r) do
  x = Enum.random(0..r)
  y = Enum.random(0..r)
  :math.pow(r, 2) > :math.pow(x, 2) + :math.pow(y, 2)
end
```

Next, we define a function called round/3 that throws a number of darts, k, on a target with radius r and adds the number of hits to an accumulated value a. It uses the dart/1 function to determine whether each dart lands inside the circle, and adds the hits to the accumulated value a. We use a recursive algorithm to iterate over until we reach the base case where there are no darts left to throw. At this point, we return the accumulated number of darts that have landed inside the circle.

```
def round(0, _, a) do a end
 def round(k, r, a) do
  if dart(r) do
    round(k - 1, r, a + 1)
  else
    round(k - 1, r, a)
  end
end
```

Finally, we set up a test that runs a number of rounds and displays the estimated value of $\pi$ after each round. This function takes three inputs: the number of rounds k, the number of darts per round j, and the radius of the circle r.

```
def rounds(k, j, r) do
  rounds(k, j, 0, r, 0)
end
def rounds(0, _, t, _, a) do 4 * a / t end
def rounds(k, j, t, r, a) do
  a = round(j, r, a)
  t = t + j
  pi = 4 * a / t
  :io.format("Estimate value for \pi: #{pi},
               Difference: #{pi - :math.pi()}")
  rounds(k-1, j, t, r, a)
end
```

## Conslusion

Overall, this assignment was interesting. We tested our solution by running multiple rounds with different numbers of darts and different radii. The results showed that the estimated value of $\pi$ converged towards the actual value of $\pi$ as the number of darts increased. In conclusion, the Monte Carlo method provides an effective way to estimate the value of $\pi$ and our solution demonstrated how to implement this method in Elixir using simple functions.