Randomized algorithm: Assignment 1

Niebo Zhang Ye

1 Throwing darts

The primary objective of this exercise is to calculate the probability of hitting a specific pre-defined target on a dashboard. To illustrate, we consider a square of length d as the entire space and inscribe a circle with maximum dimensions within it as the dart board.

We approximate the value of π by randomly throwing darts within the square. Ideally, the proportion of darts that land within the circle should approximate the ratio of the circle's area to the total area of the square.

Assuming the square's centre is $O = (0, 0)$, we can simplify the problem to check if the distance from O is less or equal to d.

Assume the number of thrown darts N and the number of darts falling inside the circle C. We have that

$$\lim_{N\to\infty} \frac{C}{N} = \frac{A(D(0, d/2))}{A(S(0, d))}$$

Where $A(D(0, d/2))$ is the area of the disk with center O and radius $d/2$ and $A(S(O, d))$ is the area of the square of center O and length d.

Thus, we have that

$$\lim_{N\to\infty} \frac{C}{N} = \frac{\pi}{4}$$

For instance, if we make the calculations, we obtain that

$$\pi = \lim_{N\to\infty} 4\frac{C}{N}$$

1.1 Implementation

For the implementation, as mentioned in the previous section, we consider the square center of length d to be $(0, 0)$; consequently, it is the center of the circle as well. For instance, the maximum circle inscribed within the square of length d has $r = d/2$. Actually, the problem can be simplified to $d = 1$ and $r = 1/2$. Thus, we take the random variables $P_X, P_Y \sim U[-0.5, 0.5]$, and the point $P = (P_X, P_Y)$, we see that

$$P \in D(0, r) \iff d(P, O)^2 = P_X^2 + P_Y^2 \leq r^2$$

.

In our code implementation, we opted for numpy arrays over lists to eliminate the need for loops. This choice has significantly enhanced the efficiency of operations involving numpy arrays, allowing us to conduct experiments with larger values of n and achieve greater precision in our results.

Algorithm 1 Dart

```
1: function throwDarts(N)
2:     hits ← 0
3:     for i ← 1 to N do
4:         x ← random number from U[0, 1]
5:         y ← random number from U[0, 1]
6:         if d(x, y)^2 < r^2 then
7:             hits ← hits + 1
8:     return 4 × hits/N
```

In the previous pseudocode, we have the main core of the class `Dart`, where we can see how we obtain the value for π.

1.2 Experiment

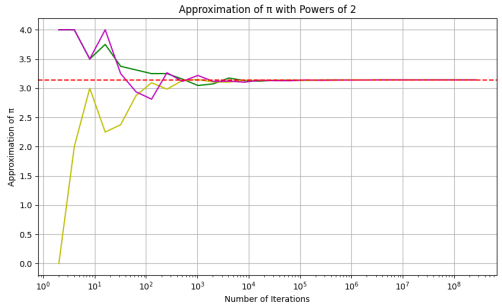

Figure 1.1: X axis is in log scale.

In Figure 1.1, we can see an approximation of the value of π. For the experiment, we fixed three different values of the seed to observe the same behaviour for all the seeds. We set $N = 2^i$ for $i = 1 \dots 29$ and plotted the results. We can observe how the final value tends to π as the values get larger. The red dashed line represents the true value of π.

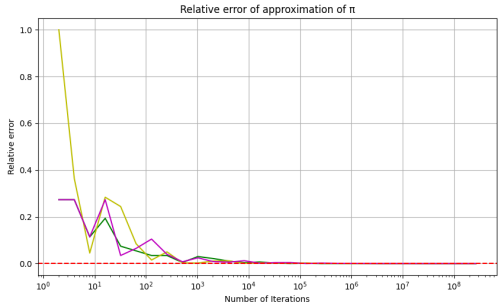

Figure 1.2: X axis is in log scale.

In Figure 1.2, we compute the relative value of the approximation of π using the following equation

$$\epsilon = \left|\frac{\pi - P_\pi}{\pi}\right| \tag{1}$$

In the plot, we can observe that for $N > 100$, the relative error is consistently less than 1. In the table found in Appendix A, 40 different relative errors are computed using different seeds, and we still observe that the error does not exceed 1.

2 Buffon's needle

In this section, we present a different experiment that offers a simple method for estimating π by dropping needles onto a floor marked with parallel lines. Given a surface with several parallel stripes, the objective is to calculate the probability that a needle of length l falls in such a way that it intersects one of the stripes. For instance, the distance between the stripes is denoted as t.
Eventually, the probability that this event happens is $\frac{2l}{t\pi}$. We skip the explanation of how to derive this value because it is highly theoretical; thus, one can find the explanation here.
For instance we need to generate two random variables. Thus, we take $X \sim U[0, t/2]$ and $\theta \sim U[0, 90]$. The first value is the distance from the center of needle to the nearest stripe. In this case, the maximum distance is $t/2$. And θ is the angle we need to consider when the needle falls. We can observe that it can be from 0 to 180. But for instance, if we consider the symmetry we can omit the values from 90 to 180 since it does not change the condition to intersect the stripe. And the condition for the thrown needle to cross the line is

$$X \leq \frac{l}{2}\sin(\theta) \tag{2}$$

2.1 Implementation

We follow the same strategy for the previous problem. But in this case, we can directly compute the desired values. The implementation is done such that the values l and t are input parameters and can be modified to observe different behaviours.
The pseudocode for this section is the following one:

Algorithm 2 BuffonNeedle

```
1: function ThrowNeedles(N)
2:     hits ← 0
3:     for i ← 1 to N do
4:         x ← random number from U[0, t/2]
5:         θ ← random number from U[0, 90]
6:         if x ≤ (l/2) sin(θ) then
7:             hits ← hits + 1
8:     return N/hits
```

2.2 Experiment

The experiment follows a similar approach to that of the previous section, utilizing 2^i as the number of needles thrown. For example, we take $t = 1$ and $l = 1/2$ to estimate the value of π. However, different parameter values can also be employed. Nevertheless, the final result must be scaled to yield the approximation of π.

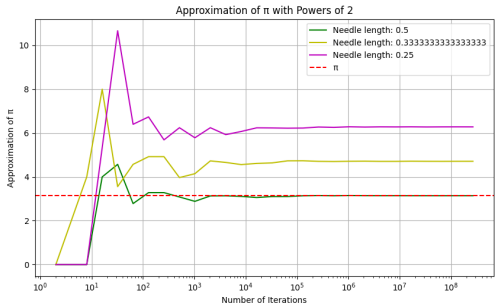

Figure 2.1: Caption

In figure 2.1 we can see the approximation of π when we consider different lengths of needs fixing the distance between stripes to be 1. As we can see for $l = 1/2$, the value converges to exactly π and when we have other values, the approximation needs to be scaled by a value. For instance, when we have $l = 1/3$, the approximation of π is not N/C but $2N/3C$.

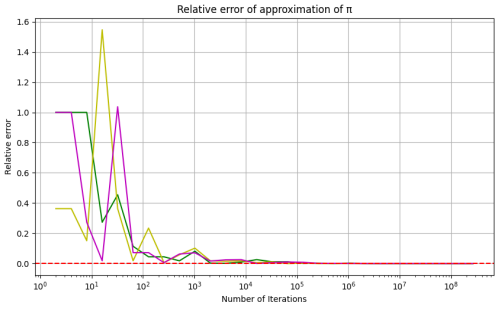

Figure 2.2: Caption

Finally using the equation 2, we compute again the relative error when using this approach to compute the approximation of π. We can see that when we use this method, we required more number of iteration (for instance, number of needles) to obtain a more accurate approximation of π.

A Appendix A

Table A.1: Relative error values for $n = 100$ in throw dart experiment

Seed	Relative error
0	0.0185916
1	0.0323379
2	0.0185916
3	0.0567888
4	0.0450703
5	0.0450703
6	0.0196056
7	0.0185916
8	0.0440564
9	0.031324
10	0.0185916
11	0.0185916
12	0.0705351
13	0.0323379
14	0.094986
15	0.00687316
16	0.00585924
17	0.031324
18	0.0705351
19	0.0196056
20	0.0695212
21	0.031324
22	0.0323379
23	0.0196056
24	0.0185916
25	0.00585924
26	0.0578027
27	0.0959999
28	0.0196056
29	0.00585924
30	0.0196056
31	0.00687316
32	0.0822536
33	0.031324
34	0.185127
35	0.0185916
36	0.107718
37	0.0185916
38	0.0822536
39	0.031324
40	0.00585924
⋮	⋮