Some naive and computationally polished techniques to approximate π.

Shubhajit Dey

Indian Institute of Science Education and Research - Bhopal
Madhya Pradesh, India

Abstract

Abstract. We primarily study Monte-Carlo Methods, the theory of Riemann Integrals and some infinite series', namely, the Madhava-Gregory-Leibniz series and Ramanujan-Gregory-David series and apply these to conduct experiments aiming to approximate π. The target is to minimize the relative error, associated with approximation, well within the order of 10^{-10} or lesser.

1 Introduction

π has been one of the centres of fascination in applied mathematics, scientific computing, or, to be honest, in mathematics entirely for ages. On 14th March 2019, when the mathematics community and mathematics enthusiasts around the globe were celebrating the **International Pi Day** for that year, Emma Haruka Iwao and her team consisting of employees from Google Research created a record of approximating π to the highest number of significant digits. Iwao calculated pi to 31 trillion digits, dwarfing the previous record of 22 trillion digits, made sometime back in 2016.

It is very interesting that using only the first 39 significant digits of π, one can actually encompass measurements starting from the diameter of a hydrogen atom to the width of the measurable universe. This intriguing concept not only puts a scratch of doubts as to whether we need 31 trillion digits of π, but also motivates one to find out how far one can probably go on to approximate π, using generic mathematical principles and computational tools. Using this as the primary motivation, through this elementary project, we intend to approximate π. Our primary intention is to chase down the target of reaching the relative error, associated with approximation well within the order of 10^{-10} or lesser.

We would be exploring and explaining working principles of Monte-Carlo methods[5], Riemann Integration[3], and some very intimidating infinite series[7][2]. Further, interesting corners in mathematics revolving around the constant π can be found in [6]. The platform/language used to implement the codes is MATLAB. The style used to explain and introduce experiments in this article is beginner-friendly and can be easily understood by entry-level programmers and mathematics students. The algorithms presented and used in this article are backed up by robust mathematical concepts and theorems, mostly belonging to the section on real analysis. Such theorems and their respective proofs are discussed in detail in this article, but one can find them in [1]. Such preliminary mathematical concepts from undergraduate studies are prerequisites for comprehending the scope of this article.

2 Method 1: A Monte - Carlo Method

The Monte-Carlo Method[5] basically relies on random sampling to generate numeric results. In our context, we will use the probability theory to approximate π. In order to get a proper picture of the theory, let us set up a simple experiment. Suppose we have a square and a circle inscribed inside it. Now, **the theory** suggests that, if we 'randomly' throw dots in the construction we have, the probability that a dot will land up inside the circle is given by:

$$\approx \frac{\text{Area of the Circle}}{\text{Area of the Square}}.$$

Now, in order to solve the problem in our hands, we need to set up a similar construction as above, smart enough to get the desired outcome. The plan is to have such a set-up so that the mentioned probability has a limiting value of π.

Set up : Let us select a unit square in the first coordinate and the quarter of the unit circle centred at the origin. As depicted in the figure, the quarter would belong completely inside the unit square. We can now, apply the theory under consideration to solve our purpose. Here, the limiting value of the probability of a dot being inside the quarter, for high enough dots thrown would be:

$$\approx \frac{\text{Area of the Circle}}{\text{Area of the Square}} = \frac{\frac{1}{4} \cdot \pi \cdot (1)^2}{(1)^2} = \frac{1}{4} \cdot \pi.$$

Methodology : Thus, if we pick a bunch of points and inspect the percentage of points that land inside the circle, we can approximate π.

Code : GitHub.

Results :

Iterations	Approximated π	Relative error	Time Elapsed (in sec.)
10,000	3.1444	8.9361e-04	112.480338
100,000	3.1394	6.8521e-04	1037.527324
1,000,000	3.1406	3.2234e-04	8760.730752

3 Method 2: A Riemann Integral Approach

This method actually centres around the area of a circle. A circle of radius 1 will have area π. Thus, by extending the idea, if we manage to accurately compute the unit circle area, we would be approximating π in extension.

Now, when it comes to finding areas, Integrals are pretty popular in the task. Let us consider the curve Γ given by:

$$\Gamma := \{(x, y) \mid y = \sqrt{1 - x^2};\ x, y \in \mathbb{R}\}.$$

The area captured by Γ, and the x-axis in the first quadrant is the quarter of the unit circle centred at the origin. Now, in order to calculate the area under the curve, i.e., the area of the quarter unit circle, we use the theory of Riemann Integrals[3]. This concept was developed by the famous mathematician Bernhard Riemann. The theory first defines something popular by the name of Riemann Sum of a continuous and bounded function f on a closed interval, defined as:

$$\sum_{k=0}^{n} (x_{i+1} - x_i) \cdot f(x_j), \qquad \text{where } x_j \in (x_i, x_{i+1}).$$

By theory, the Riemann Integral of the function f is $\mathcal{S}$, if it exists, is such that:

$$|\mathcal{S} - \sum_{k=0}^{n}(x_{i+1} - x_i) \cdot f(x_j)| < \epsilon,$$

for any given $0 < \epsilon$. Note that, if the closed interval under consideration is $[a, b]$, then we consider a proper partition of the interval as $\mathcal{P} := \{a = x_0, x_1, x_2, \cdots, x_k = b\}$.

Set up : By continuation from the concept discussed, this Riemann Integral, $\mathcal{S} \approx \frac{\pi}{4}$ for the curve Γ.

Methodology : Intuitively, the product $(x_{i+1} - x_i) \cdot f(x_j)$ gives us the area of rectangles under the curve Γ. Thus, conceptually, if we sum the areas of these rectangles, then we would get the required $\mathcal{S}$. The finer the partitions, that is, the more appropriate rectangles, the better the approximation.

Code : GitHub.

Results :

Iterations	Approximated π	Relative error	Time Elapsed (in sec.)
1,000,000	3.1416	6.3624e-07	2.892718
10,000,000	3.1416	6.3842e-08	5.194456
100,000,000	3.1416	4.6696e-09	6.156474

4 Method 3: Approach using expansion of the arctan function

Since the last approach we used, It appears that it is much more efficient to find the limiting value of an infinite or a finite series. Thus, continuing on this principle, we use an idea from trigonometry that $\tan(\frac{\pi}{4}) = 1$. Thus, we can use the inverse tangent function, especially arctan(1), to get the value of π.

Set up : For implementing the concept discussed above, we use an infinite series expansion of the arctan function, popularly known as the Gregory-Leibniz Series (also the Madhava–Gregory Series)[7] developed by the famous mathematicians Gottfried Leibniz, James Gregory, and Madhava of Sangamagrama.

Methodology : The series goes as follows:

$$arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^{2n+1}}{2n+1}.$$

Clearly, to serve our purpose, we can put $x = 1$ in the above equation.

Code : GitHub.

Results :

Iterations	Approximated π	Relative error	Time Elapsed (in sec.)
100,000,000	3.1416	3.1832e-09	9.802578
1,000,000,000	3.1416	3.1886e-10	67.670268
10,000,000,000	3.1416	3.2292e-11	709.5580

5 Method 4: An improved approach using an infinite series

As the name suggests, in this final trial, we would use an advanced infinite series which directly approximates for the constant $\frac{1}{\pi}$ or simple variations of this constant. The series under consideration was developed by a very famous mathematician Srinivasa Ramanujan. His idea was then taken forward and developed further by Gregory Chudnovsky and David Chudnovsky[2][4].

Methodology : The series discussed are as follows (only one of them is implemented in the code to approximate pi):

$$\frac{1}{\pi} = \frac{\sqrt{8}}{99^2} \sum_{n=0}^{\infty} \frac{(4n)!}{(4^n n!)^4} \cdot \frac{1103 + 26390n}{99^{4n}} \quad \cdots Ramanujan,$$

$$\frac{4}{\pi} = \frac{1}{882} \sum_{n=0}^{\infty} \frac{(-1)^n (4n)!}{(4^n n!)^4} \cdot \frac{1123 + 21460n}{882^{2n}} \quad \cdots Ramanujan,$$

$$\frac{1}{\pi} = 12 \cdot \sum_{n=0}^{\infty} \frac{(-1)^n (6n)!}{(3n!)(n!)^3} \cdot \frac{13591409 + 545140134n}{640320^{3n+\frac{3}{2}}} \quad \cdots Chudnovsky.$$

Code : GitHub.

Results :

Iterations	Approximated π	Relative error	Time Elapsed (in sec.)
3	3.1416	3.7460e-14	2.9814e-04
5	3.1416	3.7460e-14	1.1349e-04
10	3.1416	3.7460e-14	9.0822e-05

Conclusions

The results obtained from each experiment are interesting. Starting with the first method, the time complexity of the technique, keeping in mind the relative error of the approximations made, is disappointing. One can notice that the time elapsed for running 1M iterations took ~ 3 hours. This factor makes this method the least preferred one to approximate π. Moving on to the following method, the Riemann Integral approach thrashed down on the relative error made in the calculation and was lighter on the time complexity factor. To have a head-on comparison, Method 2 took only 2.892718 seconds to complete 1M iterations, giving a much lower relative error. In contrast, Method 1 under the same metrics took ~ 3 hours to complete the task, keeping aside the fact that the relative error was way higher than anticipated. Between Methods 1 and 2, method two wins over 1. When method 3 comes into the picture, the scenario and the comparison procedure slightly becomes tricky. It is because method 3 gave promising results from the relative error perspective but at the cost of increased time complexity compared to 2. For the time being, let us keep method three at a better hierarchy than method 2. Finally, when we look at method 4, it becomes crystal clear that it is difficult to do the job more efficiently than this, given the circumstances of the experiment. The benchmark result that none of the other methods could produce even with high iterations like 10M, 100M, 1B or even 10B, method 4 delivered it with just three iterations. To keep things clean, one form must compare the time complexity involved. Also, the critical fact to be noted is that, with the usage of method 4, we have achieved our primary goal of having the relative error of approximation in order of 10^{-10} or lesser. Here, we have performed a relative approximation error in the order of 10^{-14}.

Outcome : We have the most efficient method amongst the ones we tried.

Model	Iterations	Approximated π	Relative error	Time Elapsed (in sec.)
Ramanujan-Chudnovsky Infinite Series method	10	3.1416	3.7460e-14	9.0822e-05

References

[1] A. Burden, R. Burden, and J. Faires. *Numerical Analysis, 10th ed.* Cengage Pvt. Ltd., 01 2016.

[2] J. Guillera. Proof of the chudnovsky's series for $1/\pi$. *Nature*, 03 2020.

[3] P. Heywood. W. rudin, principles of mathematical analysis (mcgraw-hill publishing company ltd., 1964), ix 270 pp., 62s. *Proceedings of the Edinburgh Mathematical Society*, 14(3):247–247, 1965.

[4] L. Milla. An efficient determination of the coefficients in the chudnovskys' series for $1/\pi$. *The Ramanujan Journal*, 57(2):803–809, Feb 2022.

[5] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods.* Springer New York, 2004.

[6] K. A. Rousan. Pi and it's hidden circles. *Researchgate*, 05 2019.

[7] E. W. Weisstein. *Gregory Series.* Wolfram MathWorld, 2016.

* * *

- TOOLS USED -

1. `overleaf.com` ····· LaTeX platform.
2. `MATLAB R2022b 9.13` ····· to implement codes.