Computational Methods for π in Python

Thomas Pasfield, Omar Alhomaidah, Owen Mudgett

December 10, 2022

Abstract

Contents

1 Introduction **3**

2 Problem Statement **3**

3 Method/Analysis **3**
3.1 Numerical Integration 3
3.2 Sum of Alternating Series 6
3.3 Monte Carlo Integration 7

4 Solutions/Results **7**
4.1 A subsection 7
4.1.1 A subsubsection 7
4.1.2 A further subdivision 7

5 Discussion/Conclusions **7**

A Python Codes **9**

1 Introduction

Pi is an extremely useful number, being fundamental to mathematics for so much more than just geometry, and having extensive applications in physics. It is simply the ratio of the circumference of a circle and the diameter of the circle, yet applies to so much more. Being such a useful number, it is important that we have an accurate value for it, and can calculate it as needed. This paper explores different methods of these calculations, using three categories of methods. These categories are numerical integration, sum of alternating series, and Monte Carlo integration.

2 Problem Statement

The goal is to approximate π with Python using different methods, and to provide an analysis of the methods relative to each other. The code must accept a number of iterations, the "N-value", and run each method using that value.

3 Method/Analysis

3.1 Numerical Integration

For some problems we can't integrate the function. Instead, we need to approximate the integration. So, we use what we call numerical integration. When we are provided a set of data rather than an explicit function or when it is challenging to locate the antiderivative of the integrand, we apply numerical integration. When it is difficult to identify a closed form of an integral or when we simply need an approximation of the integral value, we typically employ numerical integration to estimate its values.

The midpoint rule, trapezoidal rule, and Simpson's rule are the methods for numerical integration that are most frequently used.

Two functions are provided for this task.

$$\int_0^1 4\sqrt{1-x^2}dx$$

$$\int_{-1}^1 \frac{1}{\sqrt{1-x^2}}dx$$

Trapezoid Rule

The Trapezoidal Rule divides the whole area into smaller trapezoids rather than utilizing rectangles to calculate the area under the curves. This integration determines the area by approximating the region underlying a function's graph as a trapezoid. The left and right sums are averaged out according to this rule.

ADD FIGURES

As we see from the figure above, we pick points where Δx intersect and use these values to solve for the trapezoid function.

Trapezoid Function:

$$\int_a^b f(x)dx \approx \Delta\left(\frac{1}{2}f(x_0) + \Sigma_{i=1}^{N-1} f(x_i) + \frac{1}{2}f(x_n)\right)$$

where $\Delta x = \frac{b-a}{n}$ and $x_i = a + i\Delta x$

Ex.

$$T_n = \frac{\Delta x}{2}\left(f(x_0) + 2f(x_1) + 2f(x_{n-1} + f(x_n)\right)$$

$$T_{10} = \frac{\Delta x}{2}\left(f(0) + 2f(0.1) + 2f(0.2) + \cdots + f(1)\right)$$

$$T_{10} = \frac{0.1}{2}\left(4\sqrt{1-0}4\sqrt{1-0.1^2} + 4\sqrt{1-0.2^2} + \cdots + 4\sqrt{1-1^2}\right)$$

$$T_{10} = 3.1045$$

Simpson's $\frac{1}{3}$ Rule

The errors for the midpoint and trapezoid rules behave in a highly predictable manner if the function is not linear on a subinterval; they have the opposite sign. For instance, the trapezoid will be too high and the midpoint will be too low if the function is concave up. Thus, it makes reasonable that averaging the trapezoid and midpoint would give a more accurate estimate. However, we can perform better in this instance than a simple average. Using a weighted average will reduce the error. We employ Simpson's, a quadratic polynomial function, to get the appropriate weight.

ADD FIGURES

As we see from the figure above, we will have the same point as trapezoid where we pick points that Δx intersect and use these values to evaluate the Simpson's function.

Simpson's Function:

$$\int_a^b f(x)dx \approx \frac{\Delta x}{3}\left(f(x_0) + 4\Sigma_{i=1,3,5}^{N-1} f(x_i) + 2\Sigma_{i=2,4,6}^{N-2} f(x_i) + f(x_n)\right)$$

where $\Delta x = \frac{b-a}{n}$ and $x_i = a + i\Delta x$

Midpoint Rule

We run the risk of the values at the endpoints not accurately representing the average value of the function on the subinterval if we use the endpoints of the subintervals to approximate the integral. The midpoint of each subinterval is a location that is significantly more likely to be close to the average. The midpoint of every subinterval is used here to calculate the sum.

ADD FIGURES

In the midpoint method, we use the points that occur between Δx values and use these in the Midpoint function to find a solution.

Midpoint Function:

$$\int_a^b f(x)dx \approx \Delta x \Sigma_{i=1}^{N} f(x_i)$$

where $\Delta x = \frac{b-a}{n}$ and $x_i = a + i\Delta x$

Comparison

ADD FIGURES

While the trapezoidal rule uses trapezoidal approximations to estimate the definite integral, the midpoint rule uses rectangular regions to do so. Simpson's rule approximates the original function using quadratic polynomials, then it approximates the definite integral. When the underlying function is smooth, the trapezoidal rule does not provide an accurate number like Simpson's or midpoint rules do. This is due to the fact that Simpson's rule employs quadratic approximation rather than linear approximation. All three methods produce a good approximation for the integrals. Midpoint

was the best approximation for our problem, then Simpson's was very close to it and lastly trapezoid was the least accurate method.

3.2 Sum of Alternating Series

Alternating Series

$$\tan^{-1} x = x - \frac{x^3}{3} + \frac{x^5}{5} - \cdots \approx \Sigma_{n=1}^{N} (-1)^{n+1} \frac{x^{2n-1}}{2n-1}$$

The first alternating sum method uses the alternating series of arctan. Multiplying the series by 4 with an x value of 1 over N iterations approximates π. This method is fairly accurate, but has a strange interaction with N values that are exponentiations of 10. Whichever place N occupies - for example, 1000 in the thousands - the digit in the thousandths place in π will be undershot. N = 1000 gives a π value of 3.140592653840, which is accurate up to 3.141592653 except for the thousandths place.

Machin's Formula

$$4\left(4\tan^{-1}(\frac{1}{5}) - \tan^{-1}(\frac{1}{239})\right) = \pi$$

The second alternating sum method uses arctan again, but twice each with different x values. John Machin created the formula in 1706, and has been widely used since then due to how fast the series converges to π (Nishiyama). The first arctan function is multiplied by 16 and uses x = $\frac{1}{5}$, and the second arctan (subtracted from the first) is multiplied by 4 and uses x = $\frac{1}{239}$. Overall, this method is more accurate than normal 4 * arctan(1,N), but takes slightly longer.

Madhava's Series

$$\sqrt{12}\left(1 - \frac{1}{3 \cdot 3} + \frac{1}{5 \cdot 3^2} - \frac{1}{7 \cdot 3^3} + \cdots\right) = \pi$$

he third alternating sum method uses what's called Madhava's Series, with is the square root of 12 multiplied by the the series:

$$\Sigma(-1)^{n+1}/(2n-1)(3^{n-1})$$

starting at n = 1.

This one is faster than the other two, but takes many more iterations before it reaches accurate digits of π.

3.3 Monte Carlo Integration

A Monte Carlo estimator is a method of approximating an explicit value using randomness. They are useful for getting a rough idea of an outcome, rather than getting an exact or accurate one.

4 Solutions/Results

This section contains the presentation of your solution and results. Describe your implementation of the method(s) for this specific problem, any special features, numerical methods implementation strategy, choices of any parameters, stopping criteria, etc. Present the results in words and plots (annotate by hand if necessary), explain what they mean. Include your code in an Appendix.

4.1 A subsection

Text introducing this subsection.

4.1.1 A subsubsection

Text introducing this subsubsection.

4.1.2 A further subdivision

Text introducing this subsubsection.

5 Discussion/Conclusions

Interpret your solution physically, what we learn from it, comment on strengths and weaknesses of the solution method, any nice features you want to brag about, possible ways to improve it (e.g. how to make it more accurate, more efficient), as appropriate.

References

[1] Heath, Michael T., Scientific Computing: An Introductory Survey, McGraw Hill, 2002.

A Python Codes

Text introducing this appendix. Subsections and further divisions can also be used in appendices.

```
#!/usr/bin/env python3
import math

"""
================================================


MA305 - cw #: your name - date
Purpose: Find the number of trailing zeros in factorial(n).
================================================


"""

n=input('Enter a positive integer:')
n=int(n)

count=0
for i in range(1,n+1):
    count += n//5**i #pow(5,i)

print(' Number of trailing zeros in factorial(',n,'):', count)
print(' Factorial (',n,')=',math.factorial(n))
```