Computational Methods for π in Python

Thomas Pasfield, Omar Alhomaidah, Owen Mudgett

December 10, 2022

Abstract

Contents

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 Part I

Put an explanation of the methodology behind the numerical integration method here, such as where pi comes from in the integral and why.

3.2 Part II

Put an explanation for how and why pi emerges from the alternating series here. The whole 'arctan' component and more too.

3.3 Part III

A Monte Carlo estimator is a method of approximating an explicit value using randomness. In the case of approximating pi, a uniform sample of random points will approach the ratio between the Pi occurs in many, many parts of geometry. The most prevalent location being the equation to find the area of a circle, $A = \pi r^2$.

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))
```