Nicolas Corfmat
CSE13S, Winter 2023
ncorfmat@ucsc.edu

Approximating Absolute Precision Using Series

Assignment 2: A Little Slice of π

1 Description of Program

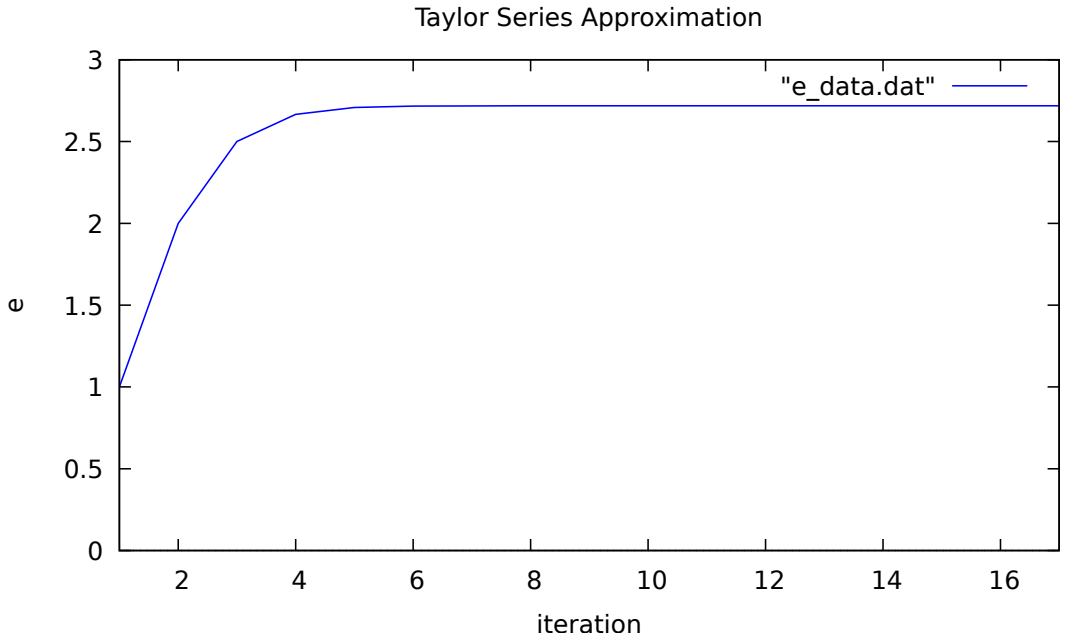
This program approximates the values of e, π, and square roots using the user-defined functions e(), pi_bbp(), pi_madhava(), euler(), pi_viete(), and sqrt_newton(), and compares them to their counterparts found using math.h functions. These results are then printed and further represented in the generated graphs.

2 Estimating Euler's Number Using A Taylor Series

Euler's number, as denoted by e, is a mathematical constant whose digits never end or repeat. As a result, calculating its exact value would be a tedious (and do not forget infinite) task. However, with the help of the following Taylor Series, can we generate a value of e precise enough to be compared with the one assigned in the C math.h library.

$$e = \sum_{k=0}^{\infty} \frac{1}{k!}$$

Furthermore, after graphing the value of e and the number of iterations it took to estimate that value, we get:



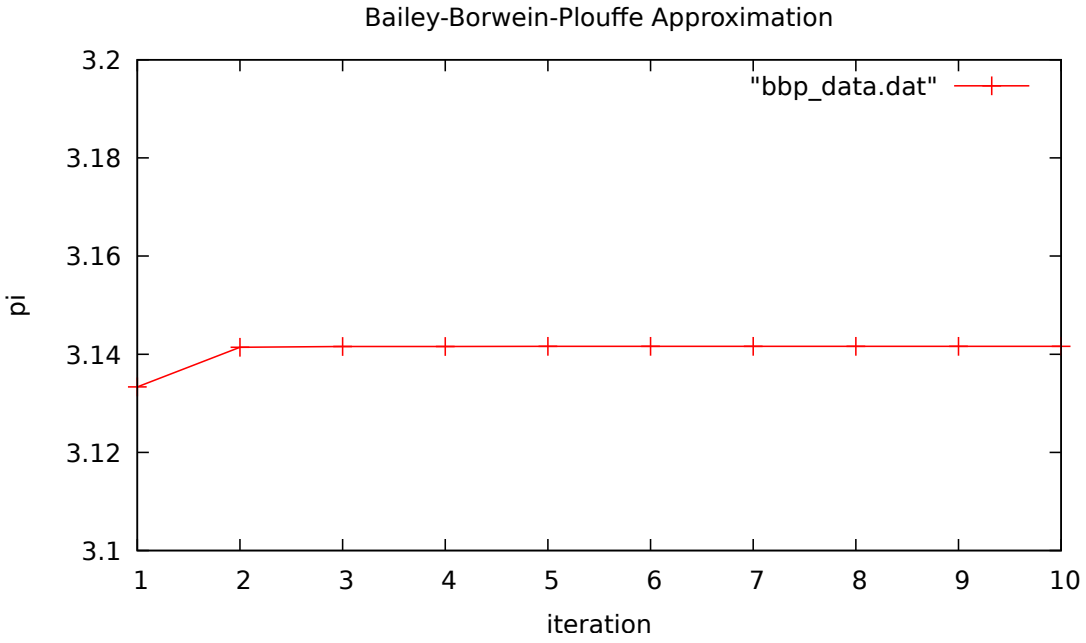
As shown by the graph above, it only takes roughly 17 terms to approximate a value of e that has a difference of 2×10^{-15} (not shown on graph). In fact, we immediately get a precise number within the first 4 to 5 iterations, proving how effective this method is for approximating the value of e. Moreover, using this Taylor series is effective for requiring low computational power, easing the processes within your computer's CPU.

3 The Bailey-Borwein-Plouffe Formula

Pi, as denoted by π, is also a mathematical constant whose digits continue infinitely. The given Bailey-Borwein-Plouffe Formula can be used to estimate the value of π to a precision close enough to the one provided in C's math.h library.

$$p(n) = \sum_{k=0}^{n} 16^{-k} \frac{(k(120k+151)+47)}{k(k(k(512k+1024)+712)+194)+15}$$

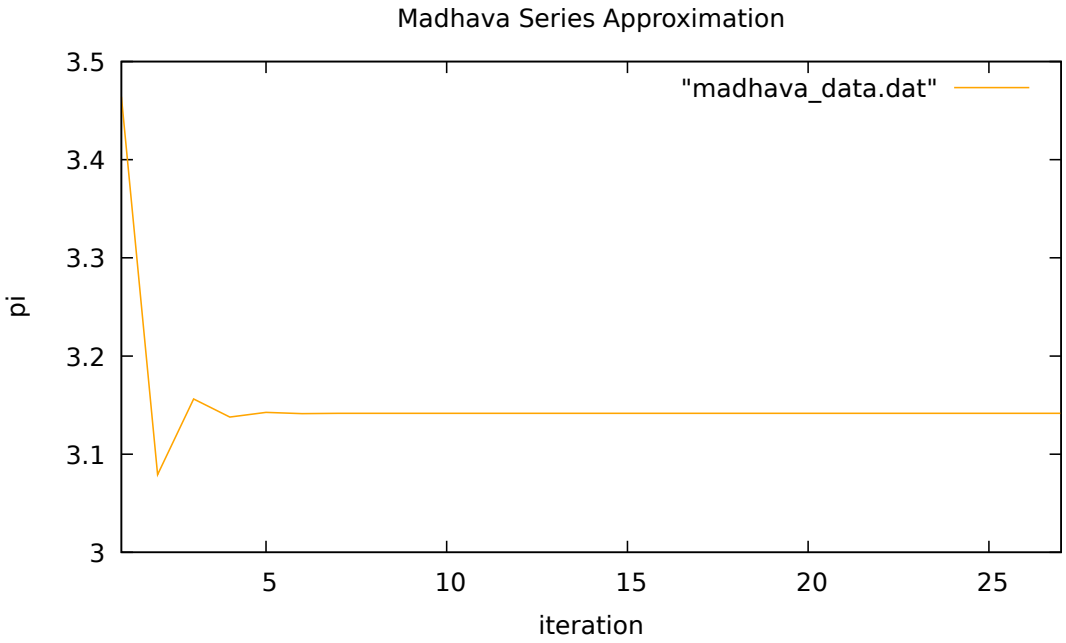
Graphing the value of π and the number of iterations it took to estimate that value gets us:



According to the produced graph, using this formula requires roughly 10 iterations to approximate a π close enough to one in the included library (difference of 2×10^{-15}). Since the series begins at around the value of π, all it takes is several more iterations to decrease the gap between its actual value (explaining the evident 'plateau' between iterations 2 and 10). Although it might appear as if the value of π is unchanging within this period, it is in fact getting more precise, but at an increasingly smaller rate. The low iteration count of this method makes it simple enough for a computer to calculate it practically instantly.

4 The Madhava Series

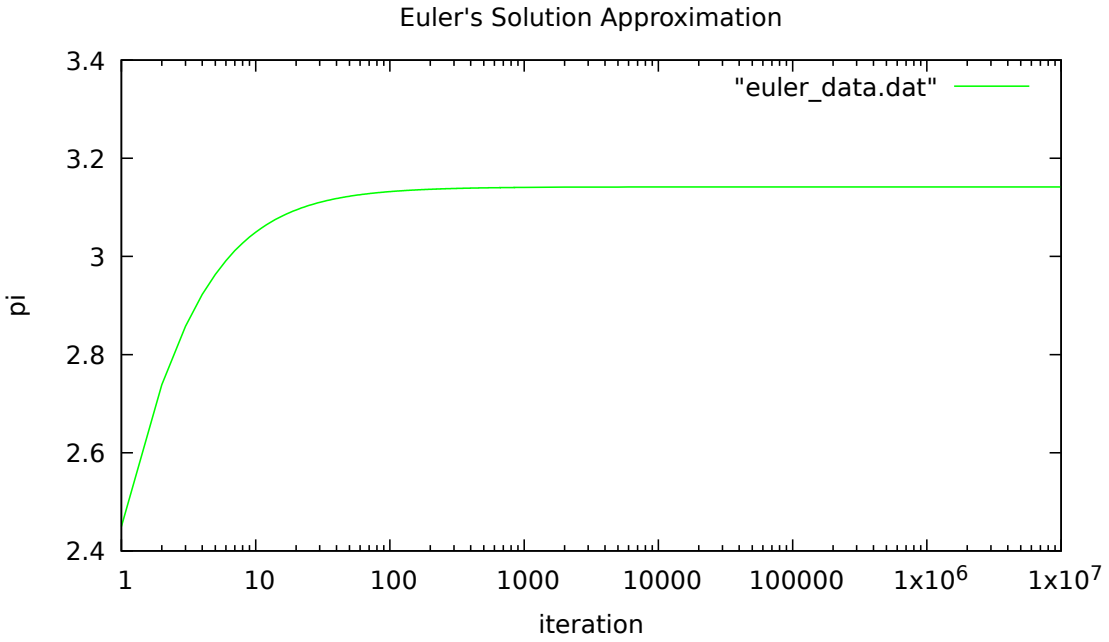
$$Formula: \quad p(n) = \sqrt{12} \sum_{k=0}^{n} \frac{(-3)^{-k}}{2k+1}$$



Observing the plot of the Madhava Series, this formula computes roughly 27 terms before it yields a value of π that is 7×10^{-15} off from the library's π value. In this approach, we add then subtract increasingly smaller fractions as we get closer to π and within the first 4 iterations, our fractions becomes so small that it is hard to notice any changes happening to the approximated value of π. While there exist differences between the efficiency of the Madhava Series and the Bailey-Borwein-Plouffe approach, they are so extremely minute in the greater picture that it would be equally fine to utilize these functions to generate precise values of π.

5 Euler's Solution

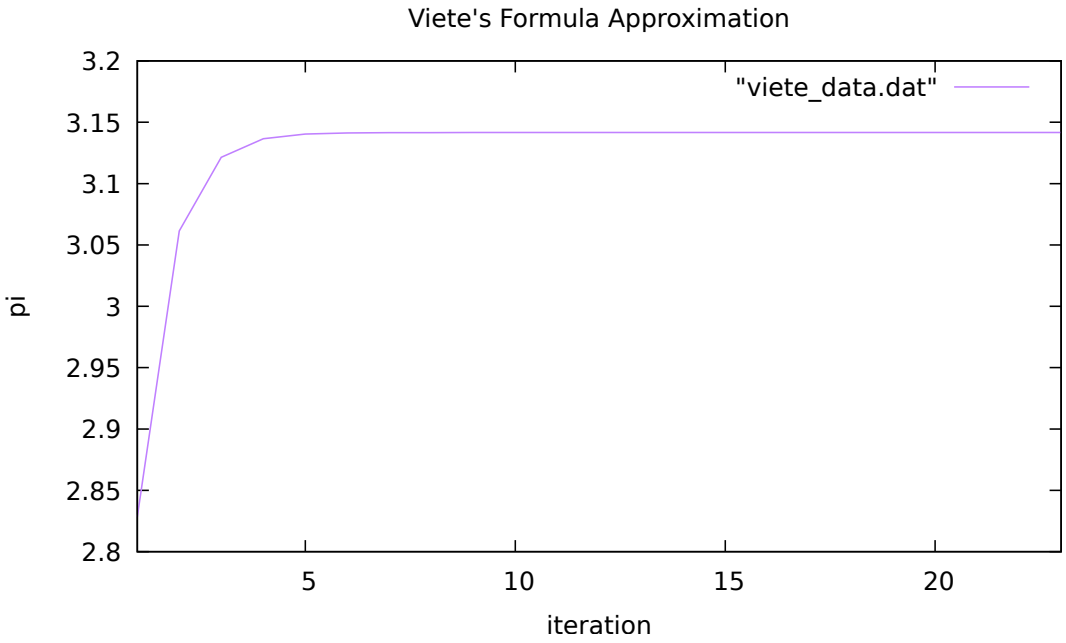
$$Formula: \quad p(n) = \sqrt{6 \sum_{k=1}^{n} \frac{1}{k^2}}$$



I want to take the time to commemorate all computers that approximate π using this solution because, wow! According to the graph, it takes roughly 10 million loops to yield a value of π that is roughly 9.55×10^{-8} off from the library's π value. Absorbing that information, can we immediately discern that this method is not only less precise than the previous two mentioned, it is also unimaginably less efficient at calculating it. The plateau observed on the graph represents the values of k^2 that get exponentially smaller as the term count increases. In order to store such a large integer, I had to use the *long* variable type, making it possible to calculate the formula's denominator. Ultimately, I would categorize Euler's Solution as an "avoid when possible," as a result of its intensive cost on your computer's CPU. In my case, calculating all 10 million data points would have been taxing for my computer, leading me to record every thousandth point after the 1000th iteration.

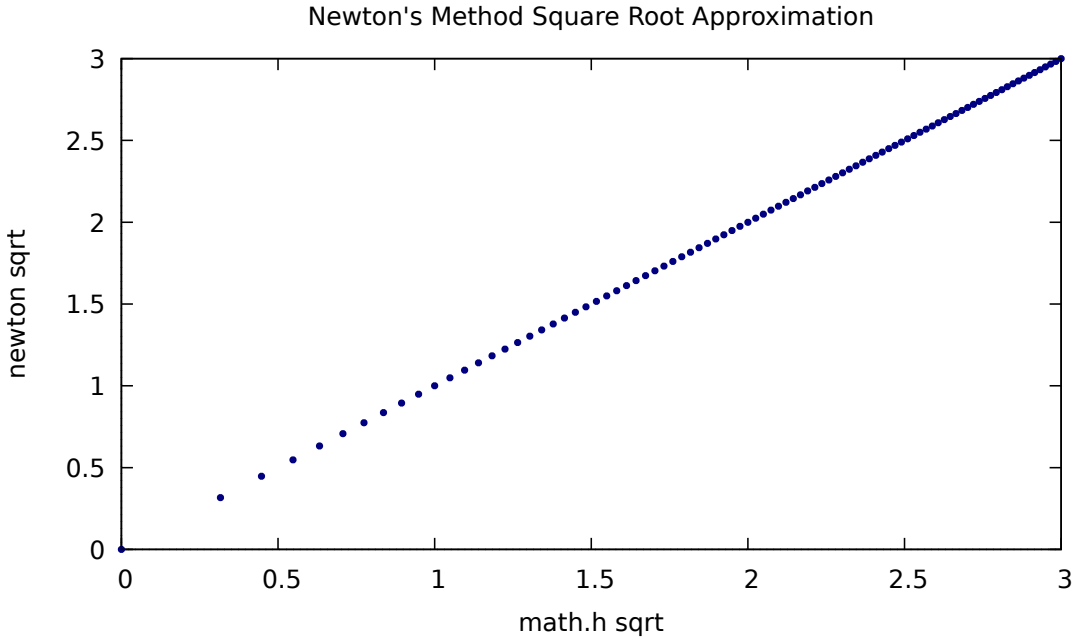
6 Viète's Formula

Formula: $\frac{2}{\pi} = \Pi_{k=0}^{\infty} \frac{a_i}{2}$ $, where \;\; a_1 = \sqrt{2} \;\; and \;\; a_k = \sqrt{2 + a_{k-1}} \;\; for \;\; all \;\; k > 1.$



Much like the Bailey-Borwein-Plouffe and Madhava Series, Viète's Formula requires minimal computational power to precisely approximate a π close enough to the given library's π (difference of 1.8×10^{-14}). Similarly, it reaches a close level of precision within the first 5 iterations, making this method both efficient and near-perfect if one were to ever need to estimate the value of π.

7 Newton's Method For Approximating Square Roots



The graph above compares the square root values calculated using the built-in math.h function and our version, which is an implementation of Newton's Method. Firstly, it becomes visibly apparent that this relationship is linear, yet not completely. We could draw a precise linear regression but never a line that goes through each point, given that there are differences between the math.h and newton square root values. Moreover, between 0 and roughly 0.3, there appears to be missing data, however what really occurred was when creating data values, I performed both square root functions on values beginning from 0 and ending at 10, incrementing by 0.1 each time. So in reality, the smallest the root could have been (other than 0) was $\sqrt{0.1}$. Overall, Newton's Method is extremely well equipped at approximating square roots.

8 Conclusion

From this assingnment, I learned that the computations performed by a computer ultimately come down to the basic four operators: addition, subtraction, multiplication, and division. With these in hand, can one develop mathematical series that can approximate values such as e and π, for example. While there are numerous approaches to estimating these numbers, there exists pros and cons to each approach that must be considered before applying them. Precision and efficiency are

effectively the two factors one must think of before utilizing these formulas, as it proves essential in improving your program's performance and reducing the amount of computational power required.

9 Credits

- The method and approach used to approximate the square root of a given value was provided by Professor Long. In the final program, this is found solely in the sqrt_newton() function.