Approximations and Representing Real Numbers in C: Writing Process

Isabella Phung

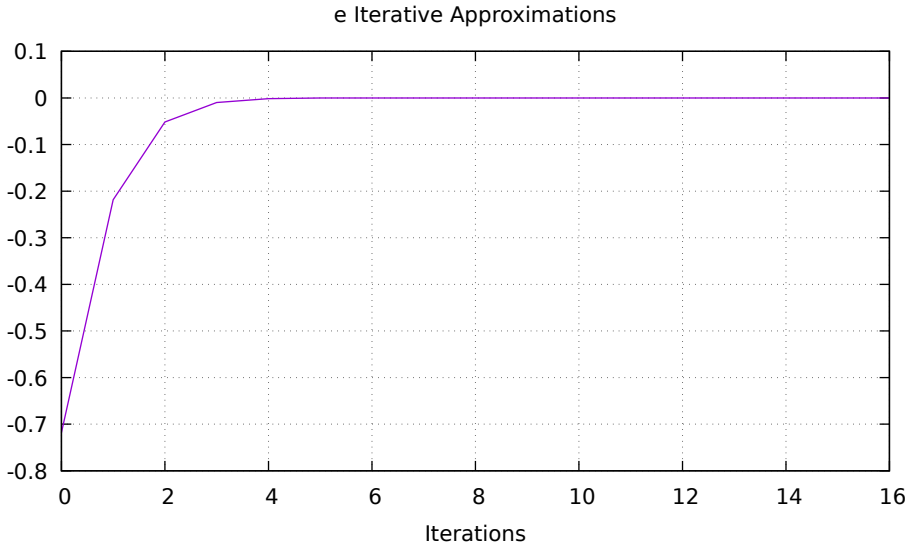
January 2023

1 Introduction

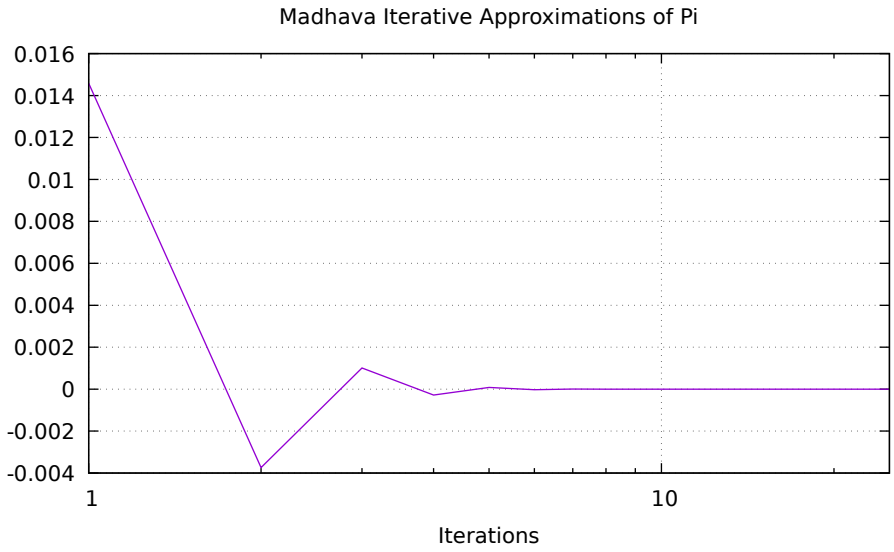
This document details the specific issues, the approximate process of putting together this library, as well as analyzes the efficiency of the different approximation methods. The DESIGN.pdf document elaborates on the steps taken when constructing these files in pseudocode.

2 Graphs

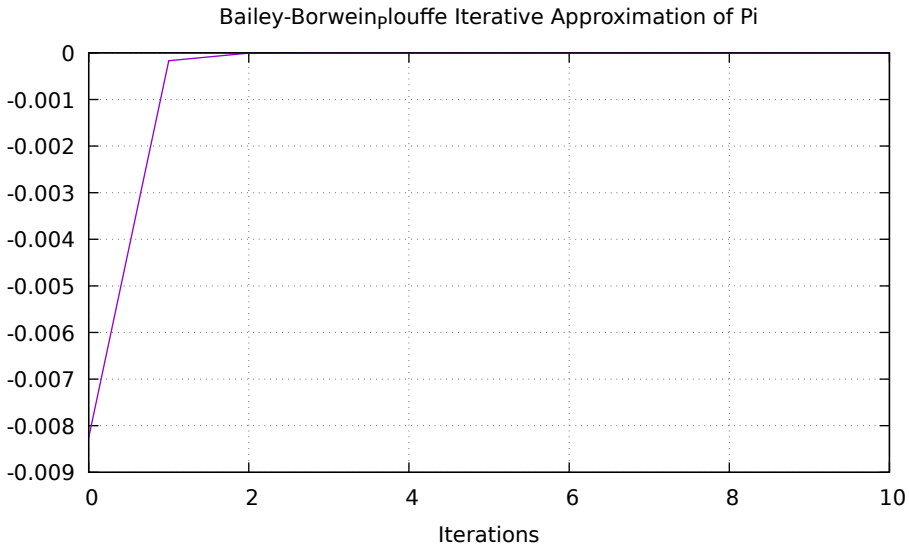
Below is the graph of the error between the estimated value of e from e.c compared to the math.h library's value of e as the program adds more terms to the summation.



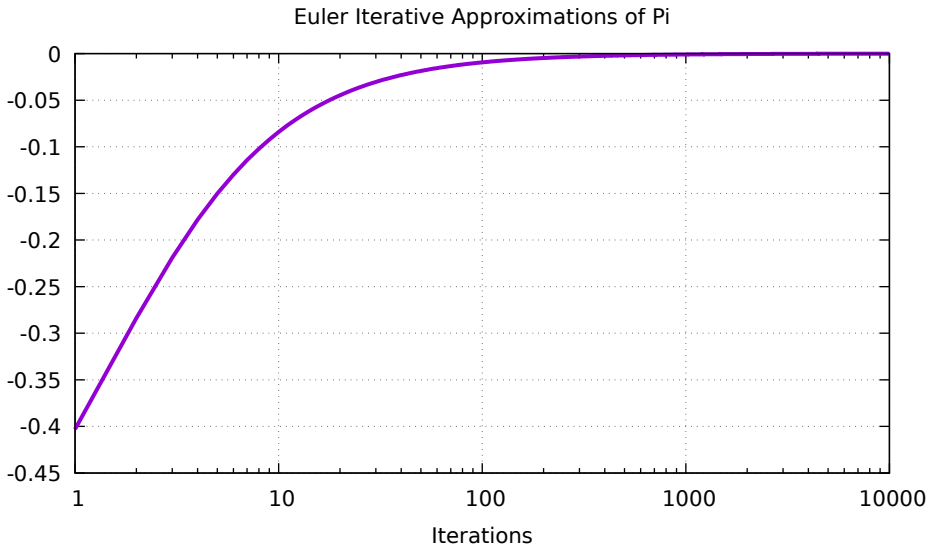
Below is the graph of the error between the estimated value of pi from madhava.c compared to the math.h library's value of pi as the program adds more terms to the summation.



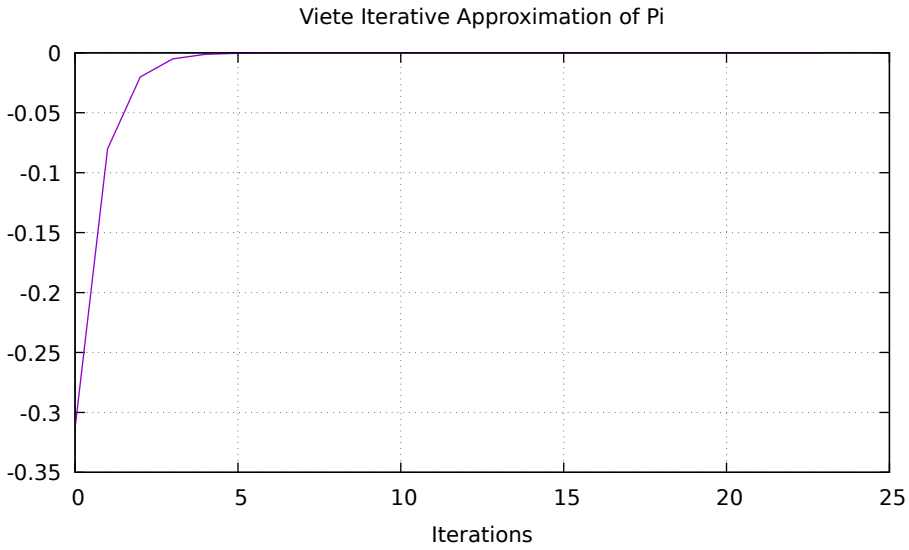
Below is the graph of the error between the estimated value of pi from bbp.c compared to the math.h library's value of pi as the program adds more terms to the summation.



Below is the graph of the error between the estimated value of pi from euler.c compared to the math.h library's value of pi as the program adds more terms to the summation.



Below is the graph of the error between the estimated value of pi from viete.c compared to the math.h library's value of pi as the program adds more terms to the pi product.



3 Conclusion

Out of all of the algorithms, the Bailey-Borwein-Plouffe (BBP) algorithm completed the estimation in the least amount of iterations. The initial error from the first guess is also the smallest out of all of the observed algorithms. What's notable, however, is how different the behavior of Euler's solution is to the other algorithms. As a reminder, Euler's solution is dictated below.

$$\sqrt{6\sum_{k^2}^{n}\frac{1}{k^2}}$$

If this algorithm were to get within 10 to the -14th, it would require 10 to the 14th iterations of the program, which, although theoretically possible, would take a significantly long period of time to compute. Already going to 100,000 iterations caused an integer overflow error when multiplying k*k. This caused the values to become negative before reverting back to positive. I experimented with unsigned 32 bit integers but I found that the program was still very slow when exceeding 10,000 iterations. I limited the algorithm to 10,000 iterations so the program would run instantly. As a result, the final error is around 0.00009.

4 Resources

Makefile Linking Reference
Undefined Reference to sqrt, using -lm in makefiles
Additional Tutorial on makefiles and linking