Assignment 2 - Pi

Angela Shen

January 2023

1 Abstract

While it is impossible to calculate the exact value of pi, there are many methods of calculating and approximating its value. This assignment implements multiple methods of approximating pi (and other constants like square roots and e).

2 Introduction

Assignment 2 contains the following deliverables:

Makefile - Compiles an executable of mathlib-test.c and creates multiple object files (mathlib-test's dependencies). For a list of these c files, .c below.

README.md - Details the process of building any necessary files, the command line options for any executables, and any errors or bugs.

DESIGN.pdf - Contains the pseudo code and descriptions of each c file.

WRITEUP.pdf - This document. Describes the assignment in its entirety and discusses the results.

mathlib-test.c - Tests the implemented formulas approximating pi/square root/e against mathematical constants in math.h.

mathlib.h - Links all of the following c files with mathlib-test.c using include "mathlib.h" so that functions can be used across c files.

bbp.c - Estimates pi using the Bailey-Borwein-Plouffe formula.

e.c - Estimates e using the Taylor series.

euler.c - Estimates pi using Euler's formula.

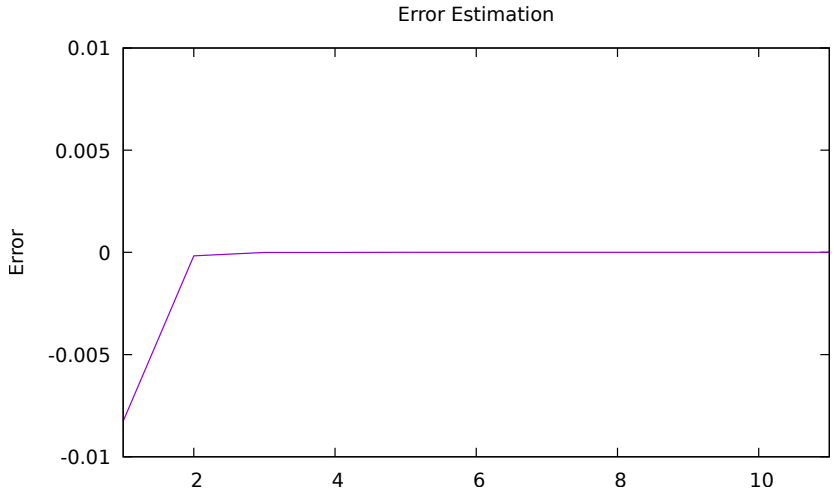
madhava.c - Estimates pi using the Madhava series.

newton.c - Estimates square roots using Newton's method.

viete.c - Estimates pi using Viète's formula.

The following will discuss the results of the c programs described above.

3 Bailey-Borwein-Plouffe Formula



The Bailey-Borwein-Plouffe Formula (henceforth abbreviated as BBP) is used to approximate the value of pi. As shown above, BBP converges very quickly, taking only 11 terms until its changes are less than epsilon. This matches the provided resources binary. The recorded difference between BBP and M_PI is 0 (with 14 trailing 0s), which shows that it is also very accurate.

4 Euler's Number

Euler's number, or e, is approximated using. My output also matches the binary for this formula, which also converges quickly in 18 terms and has a difference of 0 (with 14 trailing 0s), showing that the approximation is very accurate.

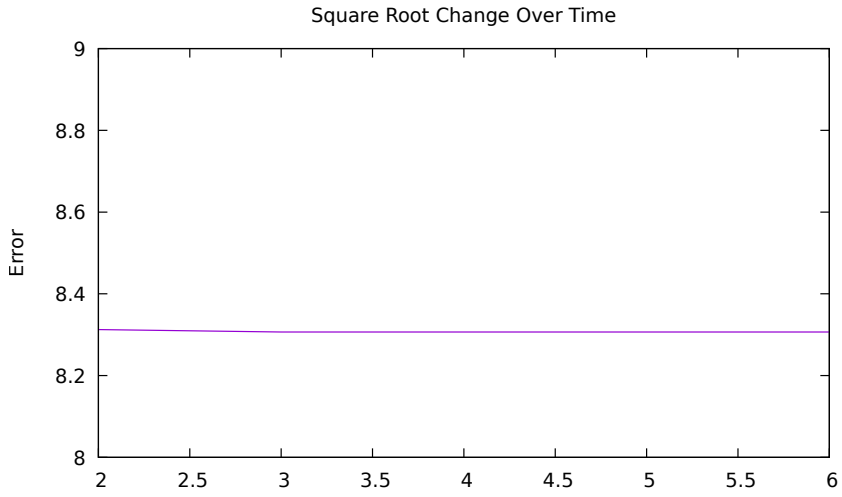
5 Euler

This section refers to Euler's method for calculating pi. It takes a long time to converge and is less accurate than other methods. Because of how many steps are necessary for this method and the inaccuracies of floating point, there are some differences between the resource binary and my own mathlib-test. The difference between my estimate and M_PI is 0.000000095981660, which is similar to but slightly different from the resource binary. The number of iterations is also different. My mathlib-test ran 9948880 iterations whereas the resource binary ran exactly 10 million.

6 Madhava

Another method for approximating pi is Madhava's formula. This formula diverges a bit more slowly than BBP (27 terms for Madhava, 11 for BBP) and has a very slight difference with the standard M_PI (0.000000000000007). This matches the resource binary.

7 Newton



Newton's method is used to approximate square roots. In this graph, the square root of 69 is approximated. It quickly converges to the point where the difference is barely noticeable on the graph in only a few terms (taking only 6 steps until the changes are less than epsilon).

8 Viète

Viete's formula, or Viète if you are a connoisseur of sorts, also approximates pi. It is about as accurate and efficient as BBP or Madhava. Because of floating point, there are more differences between the provided resource and my own test. While my test takes one more iteration (24 terms versus 23, I checked how I was counting my terms and I am fairly certain it is accurate), my test is actually a bit more accurate than the resource binary (0.000000000000004 vs 0.000000000000018).

9 Conclusion

While my results were mostly accurate to the provided resources, floating point is not quite exact and therefore, some minor differences were present. This taught me that since floating point is never completely accurate and can change depending on the compiler, system, or even the order of operations used, minor differences should be expected to occur. In addition, all of the numbers calculated in these formula are just approximations, since they are irrational numbers and no one can truly give the complete value of pi, e, or even some square roots.

I also learned a lot about compilation and creating Makefiles, since this is the first assignment where those were not directly provided to us.

10 Credit and Notes

My sqrt_newton() function was not based directly on any c code, but rather portable.py in the resources repository. Notably, most of the formulas are missing graphs. I unfortunately could not produce suitable graphs before the due date. Epsilon is defined as 1e-14.