Assignment 2 Writeup

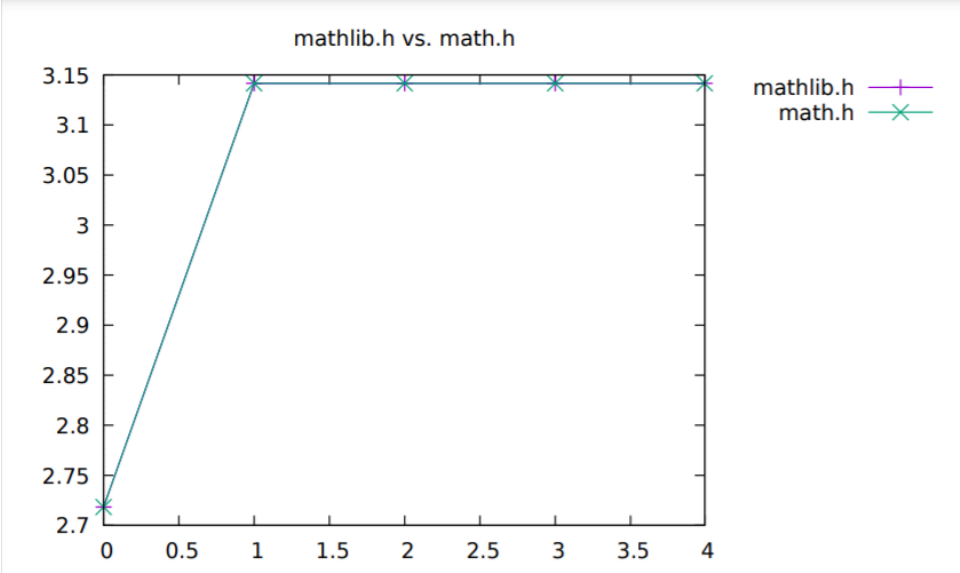
Michael Coe

September 2021

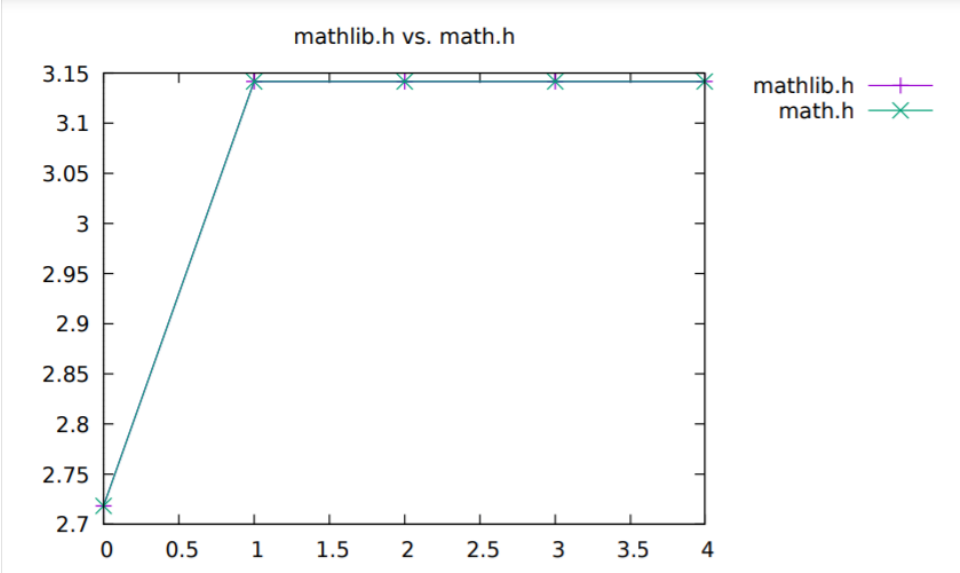
After writing the four different pi approximations, each formula resulting in different accuracy's and terms required to converge. It is also important to look at the supporting functions used for most of these formulas as well: e() and sqrt_newton().

- e() seems to work reasonably well as it converged after 17 terms and had no noticeable difference from M_E.
- sqrt_newton() works as expected with with either 6 or 7 terms required to converge with no noticeable difference except for the square root test of 0.0, which required 47 terms and had a difference of 0.000000000000007, which is likely because the root function will never converge to 0 within an epsilon.
- pi_bbp() won out with the most accurate having no noticeable difference bewtween M_PI as well as only 10 terms to converge, the smallest number of terms for the pi calculation.
- pi_viete() came in second with a 0.000000000000004 difference to M_PI and taking 24 terms to converge.
- pi_madhava() has a difference of 0.000000000000007 to M_PI and used 26 terms to converge.
- pi_euler is in last place with the largest difference of 0.000000095493891 and 10000000 terms used before converging.

Graph: Comparing square root functions from 0.0 to 10.0



Graph: comparing mathlib.h functions (not including sqrt_newton()) to math.h definitions. (points in order e(), pi_bbp(), pi_madhava(), pi_euler(), pi_viete())



Conclusion: pi_bbp() seemed to be the most efficient method of calculating pi as it not only had the least number of terms but also did not require the use of e() or sqrt_newton(). Because it also was the only formula without a noticeable difference to M_PI, a logical conclusion would guess that it is either the most accurate or M_PI uses a similar formula to pi_bpp(). Another possible reason for the other pi methods to have a difference from M_PI is their usage of sqrt_newton() and e(), for although they admittedly had no noticeable differences between sqrt() and M_E (aside from sqrt_newton(0), they still force the approximation of pi to be based on another approximation, which will reduce accuracy.