Nayeel Imtiaz
Professor Long
CSE 13s
10 October 2021

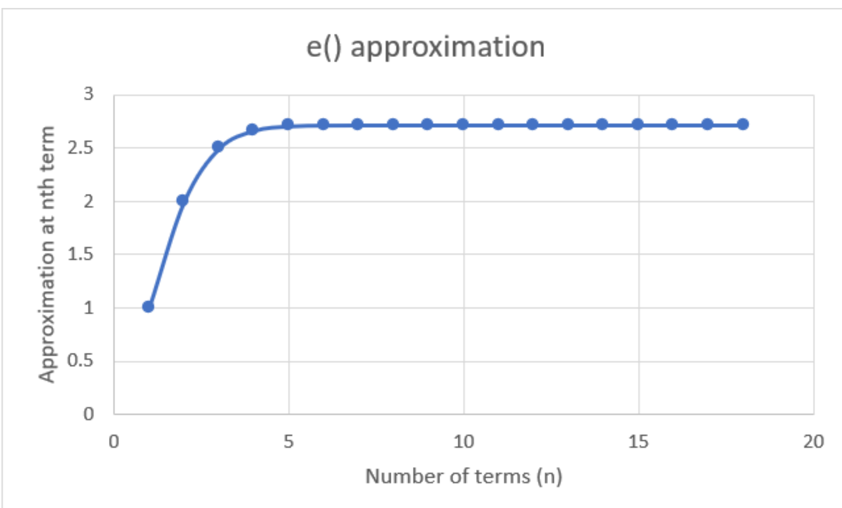
Write Up for Project 2 - “A Little Slice of Pi”

Introduction: The purpose of this assignment is to approximate many different values such as e and pi using various methods. Different methods of approximations include the Madhava series, Euler’s solution, Viete’s Formula, Newton's method, and Bailey-Borwein-Plouffe series. For each series we keep adding terms until the difference between the current term and the previous term is less than the value of epsilon (1 * 10^-14). Other approximation methods include Newton's method and Bailey-Borwein-Plouffe series. There will be a test file that can run all the approximation tests.

Results:

<u>Approximation for e()</u>

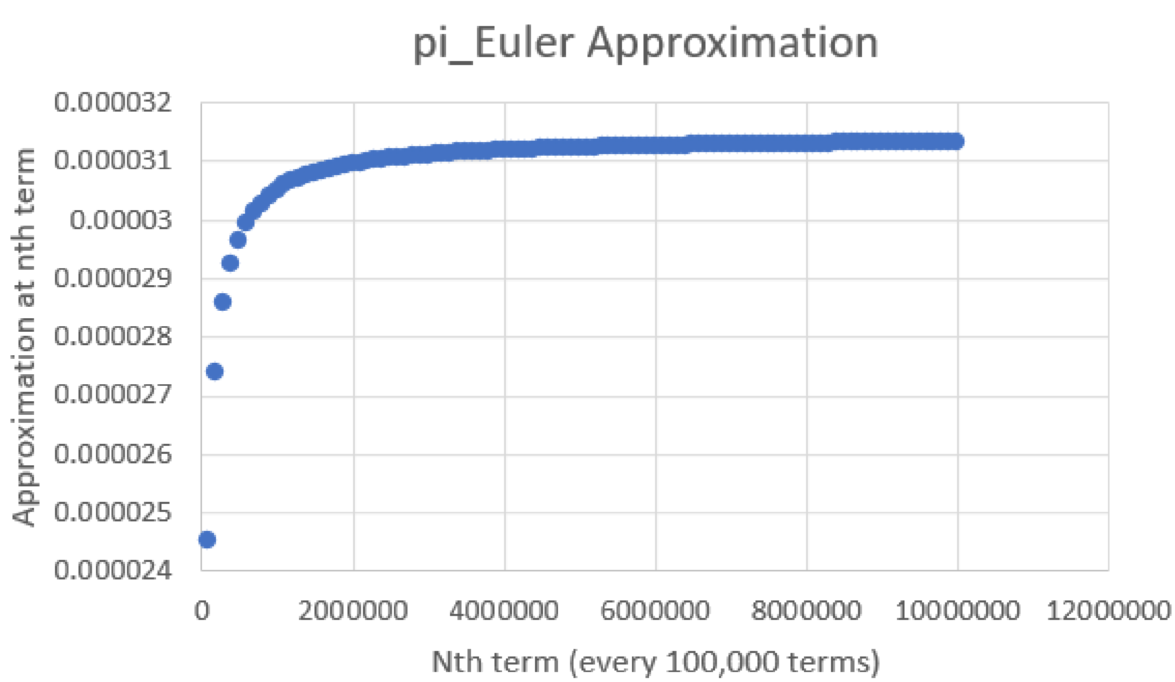
```
e() = 2.718281828459046, M_E = 2.718281828459045, diff = 0.000000000000000
e() terms = 18
```



The results for e() approximation’s value for e and math.h’s value for e are identical. The difference between the two values is 0, so this approximation test was very accurate. 18 terms were needed to converge to the value of e. This must have been very accurate due to the fact that this series is the definition of e.

<u>Approximation for pi_euler()</u>

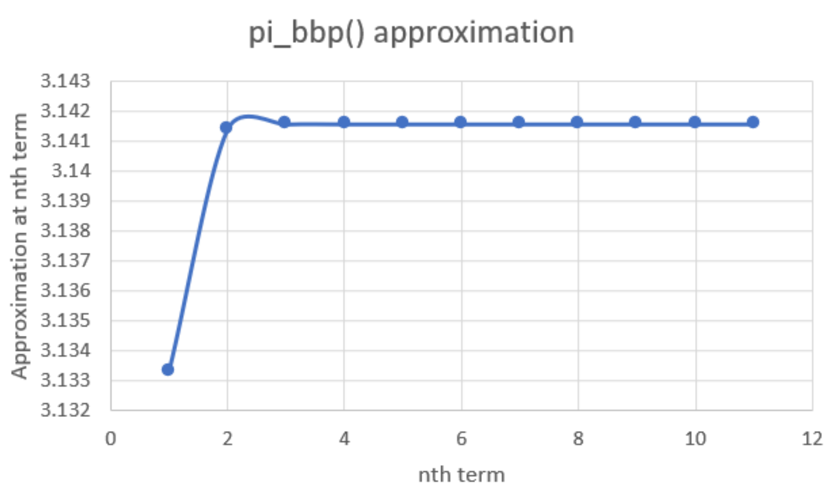
```
pi_euler() = 3.141592558096840, M_PI = 3.141592653589793, diff = 0.000000095492953
pi_euler() terms = 10000001
```



Pi_Euler()'s approximation for π and math.h's value have a noticeable difference. The error margin is about 9.54 * 10^(-9). The reason for this approximation test's inaccuracy is most likely due to C not being able to work with extremely smaller numbers. There are 10 million terms, and each of those terms are inversely squared by the term number.

<u>Approximation for pi_bbp()</u>

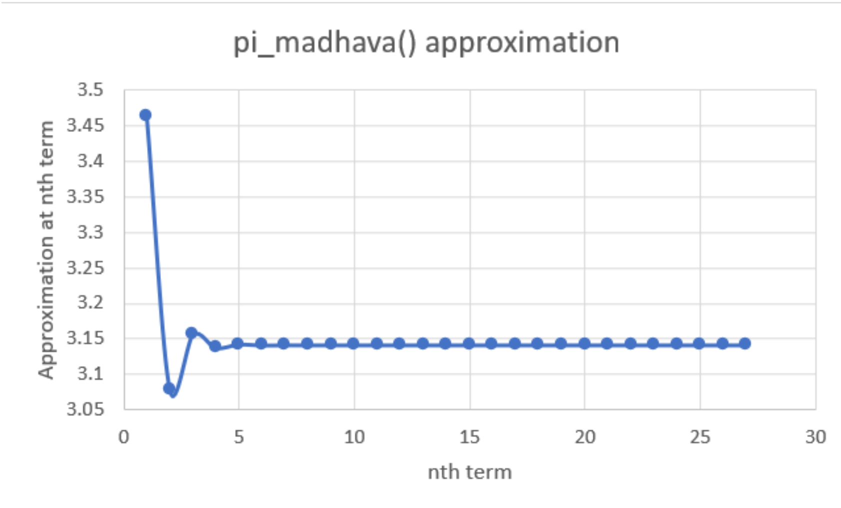
```
pi_bbp() = 3.141592653589793, M_PI = 3.141592653589793, diff = 0.000000000000000
pi_bbp() terms = 11
```



Pi_bbp's approximation for π is identical to math.h's value for π. As it can be seen from the snippet from the terminal, the difference is 0 with a precision of 15 decimal digits. The high accuracy can be attributed to the low number of terms, 11 terms, needed to reach close to π. This is opposite to pi_euler's approximation as that test used around 10 million terms, but it was far less accurate than pi_bbp.

<u>Approximation for pi_madhava()</u>

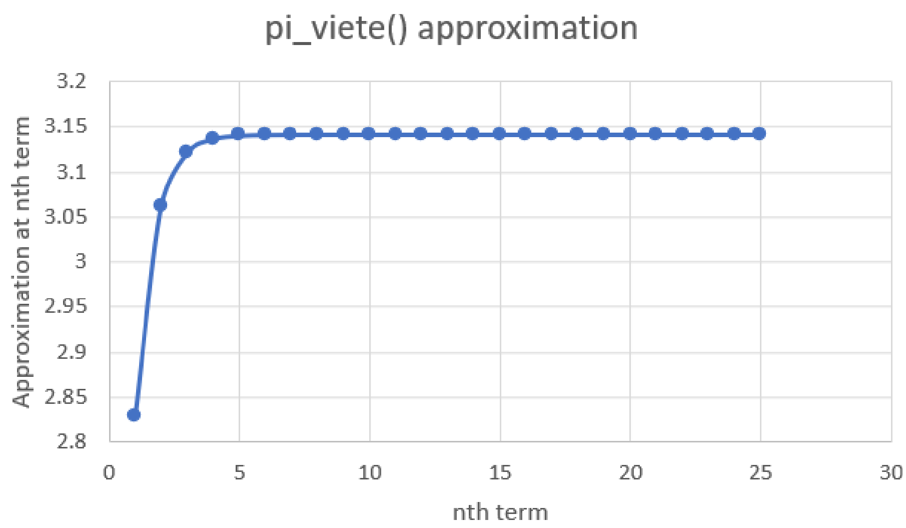
```
pi_madhava() = 3.141592653589800, M_PI = 3.141592653589793, diff = 0.000000000000007
pi_madhava() terms = 27
```



The approximation test for pi_madhava has a slightly different value for π compared to math.h's value. 27 terms were needed for my pi_madhava function, and the difference is around 7 * 10^(-15). Again, this small difference may be attributed to using more terms than pi_bbp(). Also, multiplying by square root of 12 could have caused the slight difference. The square root of 12 is calculated using a function from newton.c.

<u>Approximation for pi_viete()</u>

```
pi_viete() = 3.141592653589793, M_PI = 3.141592653589793, diff = 0.000000000000000
pi_viete() terms = 25
```



Pi_viete()'s approximation for π was very accurate compared to math.h's value for π. 25 terms were needed for my function, and the difference is also 0 with a precision of up to 10^(-15). Viete's formula is known to produce very accurate results for π, so it is no surprise that there is almost no difference between pi_viete()'s approximation from math.h's π value. This series is different from others as each term/factor is multiplied to the previous value.

Approximation Test for sqrt_newton()

```
sqrt_newton(0.000000) = 0.000000000000007, sqrt(0.000000) = 0.000000000000000, diff = 0.000000000000007
sqrt_newton() terms = 47
sqrt_newton(0.100000) = 0.316227766016838, sqrt(0.100000) = 0.316227766016838, diff = 0.000000000000000
sqrt_newton() terms = 7
sqrt_newton(0.200000) = 0.447213595499958, sqrt(0.200000) = 0.447213595499958, diff = 0.000000000000000
sqrt_newton() terms = 7
sqrt_newton(0.300000) = 0.547722557505166, sqrt(0.300000) = 0.547722557505166, diff = 0.000000000000000
sqrt_newton() terms = 6
sqrt_newton(0.400000) = 0.632455532033676, sqrt(0.400000) = 0.632455532033676, diff = 0.000000000000000
sqrt_newton() terms = 6
sqrt_newton(0.500000) = 0.707106781186547, sqrt(0.500000) = 0.707106781186548, diff = 0.000000000000000
sqrt_newton() terms = 6
sqrt_newton(0.600000) = 0.774596669241483, sqrt(0.600000) = 0.774596669241483, diff = 0.000000000000000
sqrt_newton() terms = 5
sqrt_newton(0.700000) = 0.836660026534076, sqrt(0.700000) = 0.836660026534076, diff = 0.000000000000000
sqrt_newton() terms = 5
sqrt_newton(0.800000) = 0.894427190999916, sqrt(0.800000) = 0.894427190999916, diff = 0.000000000000000
sqrt_newton() terms = 5
sqrt_newton(0.900000) = 0.948683298050514, sqrt(0.900000) = 0.948683298050514, diff = 0.000000000000000
sqrt_newton() terms = 5
sqrt_newton(1.000000) = 1.000000000000000, sqrt(1.000000) = 1.000000000000000, diff = 0.000000000000000
sqrt_newton() terms = 1
sqrt_newton(1.100000) = 1.048808848170152, sqrt(1.100000) = 1.048808848170151, diff = 0.000000000000000
sqrt_newton() terms = 5
sqrt_newton(1.200000) = 1.095445115010332, sqrt(1.200000) = 1.095445115010332, diff = 0.000000000000000
sqrt_newton() terms = 5
sqrt_newton(1.300000) = 1.140175425099138, sqrt(1.300000) = 1.140175425099138, diff = 0.000000000000000
sqrt_newton() terms = 5
sqrt_newton(1.400000) = 1.183215956619923, sqrt(1.400000) = 1.183215956619923, diff = 0.000000000000000
sqrt_newton() terms = 5
sqrt_newton(1.500000) = 1.224744871391589, sqrt(1.500000) = 1.224744871391589, diff = 0.000000000000000
sqrt_newton() terms = 5
sqrt_newton(1.600000) = 1.264911064067352, sqrt(1.600000) = 1.264911064067352, diff = 0.000000000000000
sqrt_newton() terms = 5
sqrt_newton(1.700000) = 1.303840481040530, sqrt(1.700000) = 1.303840481040530, diff = 0.000000000000000
sqrt_newton() terms = 6
sqrt_newton(1.800000) = 1.341640786499874, sqrt(1.800000) = 1.341640786499874, diff = 0.000000000000000
sqrt_newton() terms = 6
sqrt_newton(1.900000) = 1.378404875209022, sqrt(1.900000) = 1.378404875209022, diff = 0.000000000000000
sqrt_newton() terms = 6
sqrt_newton(2.000000) = 1.414213562373095, sqrt(2.000000) = 1.414213562373095, diff = 0.000000000000000
sqrt_newton() terms = 6
sqrt_newton(2.100000) = 1.449137674618944, sqrt(2.100000) = 1.449137674618944, diff = 0.000000000000000
sqrt_newton() terms = 6
sqrt_newton(2.200000) = 1.483239697419133, sqrt(2.200000) = 1.483239697419133, diff = 0.000000000000000
```

```
sqrt_newton(7.300000) = 2.701851217221257, sqrt(7.300000) = 2.701851217221257, diff = 0.000000000000000
sqrt_newton() terms = 7
sqrt_newton(7.400000) = 2.720294101747087, sqrt(7.400000) = 2.720294101747087, diff = 0.000000000000000
sqrt_newton() terms = 7
sqrt_newton(7.500000) = 2.738612787525828, sqrt(7.500000) = 2.738612787525829, diff = 0.000000000000000
sqrt_newton() terms = 7
sqrt_newton(7.600000) = 2.756809750418042, sqrt(7.600000) = 2.756809750418042, diff = 0.000000000000000
sqrt_newton() terms = 7
sqrt_newton(7.700000) = 2.774887385102319, sqrt(7.700000) = 2.774887385102319, diff = 0.000000000000000
sqrt_newton() terms = 7
sqrt_newton(7.800000) = 2.792848008753786, sqrt(7.800000) = 2.792848008753786, diff = 0.000000000000000
sqrt_newton() terms = 7
sqrt_newton(7.900000) = 2.810693864511037, sqrt(7.900000) = 2.810693864511037, diff = 0.000000000000000
sqrt_newton() terms = 7
sqrt_newton(8.000000) = 2.828427124746188, sqrt(8.000000) = 2.828427124746188, diff = 0.000000000000000
sqrt_newton() terms = 7
sqrt_newton(8.100000) = 2.846049894151539, sqrt(8.100000) = 2.846049894151539, diff = 0.000000000000000
sqrt_newton() terms = 7
sqrt_newton(8.200000) = 2.863564212655268, sqrt(8.200000) = 2.863564212655268, diff = 0.000000000000000
sqrt_newton() terms = 7
sqrt_newton(8.300000) = 2.880972058177584, sqrt(8.300000) = 2.880972058177584, diff = 0.000000000000000
sqrt_newton() terms = 7
sqrt_newton(8.400000) = 2.898275349237886, sqrt(8.400000) = 2.898275349237885, diff = 0.000000000000000
sqrt_newton() terms = 7
sqrt_newton(8.500000) = 2.915475947422648, sqrt(8.500000) = 2.915475947422648, diff = 0.000000000000000
sqrt_newton() terms = 7
sqrt_newton(8.600000) = 2.932575659723033, sqrt(8.600000) = 2.932575659723033, diff = 0.000000000000000
sqrt_newton() terms = 7
sqrt_newton(8.700000) = 2.949576240750523, sqrt(8.700000) = 2.949576240750523, diff = 0.000000000000000
sqrt_newton() terms = 7
sqrt_newton(8.800000) = 2.966479394838263, sqrt(8.800000) = 2.966479394838263, diff = 0.000000000000000
sqrt_newton() terms = 7
sqrt_newton(8.900000) = 2.983286778035257, sqrt(8.900000) = 2.983286778035257, diff = 0.000000000000000
sqrt_newton() terms = 7
sqrt_newton(9.000000) = 2.999999999999997, sqrt(9.000000) = 2.999999999999997, diff = 0.000000000000000
sqrt_newton() terms = 7
sqrt_newton(9.100000) = 3.016620625799669, sqrt(9.100000) = 3.016620625799669, diff = 0.000000000000000
sqrt_newton() terms = 7
sqrt_newton(9.200000) = 3.033150177620618, sqrt(9.200000) = 3.033150177620617, diff = 0.000000000000000
sqrt_newton() terms = 7
sqrt_newton(9.300000) = 3.049590136395378, sqrt(9.300000) = 3.049590136395378, diff = 0.000000000000000
sqrt_newton() terms = 7
sqrt_newton(9.400000) = 3.065941943351175, sqrt(9.400000) = 3.065941943351175, diff = 0.000000000000000
sqrt_newton() terms = 7
sqrt_newton(9.500000) = 3.082207001484485, sqrt(9.500000) = 3.082207001484485, diff = 0.000000000000000
sqrt_newton() terms = 7
sqrt_newton(9.600000) = 3.098386676965931, sqrt(9.600000) = 3.098386676965931, diff = 0.000000000000000
sqrt_newton() terms = 7
```

The approximation test for sqrt_newton() was pretty accurate compared to the square root function from <math.h>. For many of the test values, the difference is about 0 to a precision of 10^(-15), and the number of iterations needed mainly ranged from 5 to 7. An abnormal case would be for the square root of 0. It required 43 iterations and had a large difference compared to the others. This test may be very accurate due to the fact that Newton's method is a very accurate method to approximate values for many different functions.