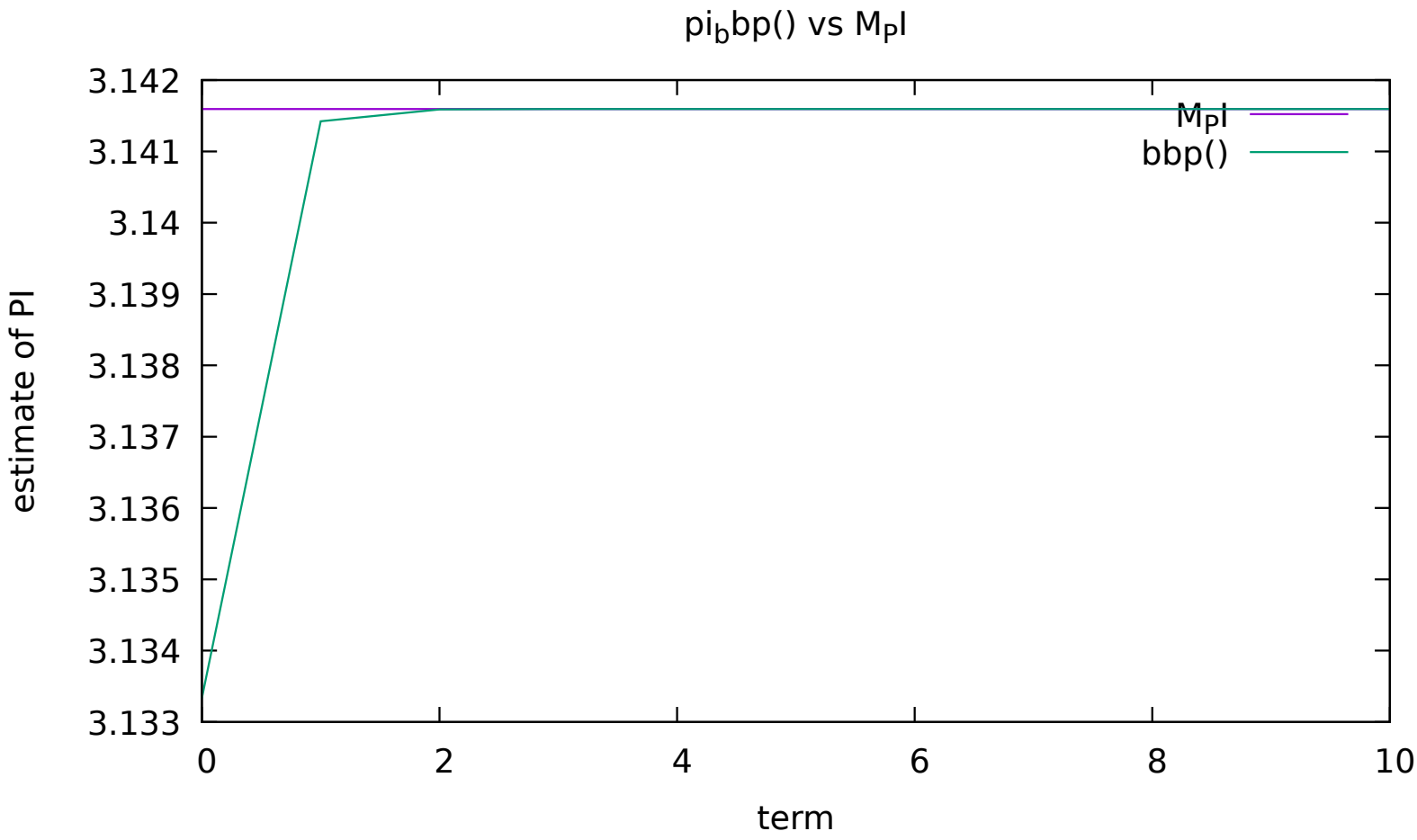
ASSIGNMENT 2 WRITEUP

Chris Moon

January 2023

1 GRAPHS: see below


pi_bbp() vs M_PI
3.142
3.141
3.14
3.139
3.138
3.137
3.136
3.135
3.134
3.133
estimate of PI
0
2
4
6
8
10
term
M_PI
bbp()

BBP GRAPH ESTIMATION ERROR

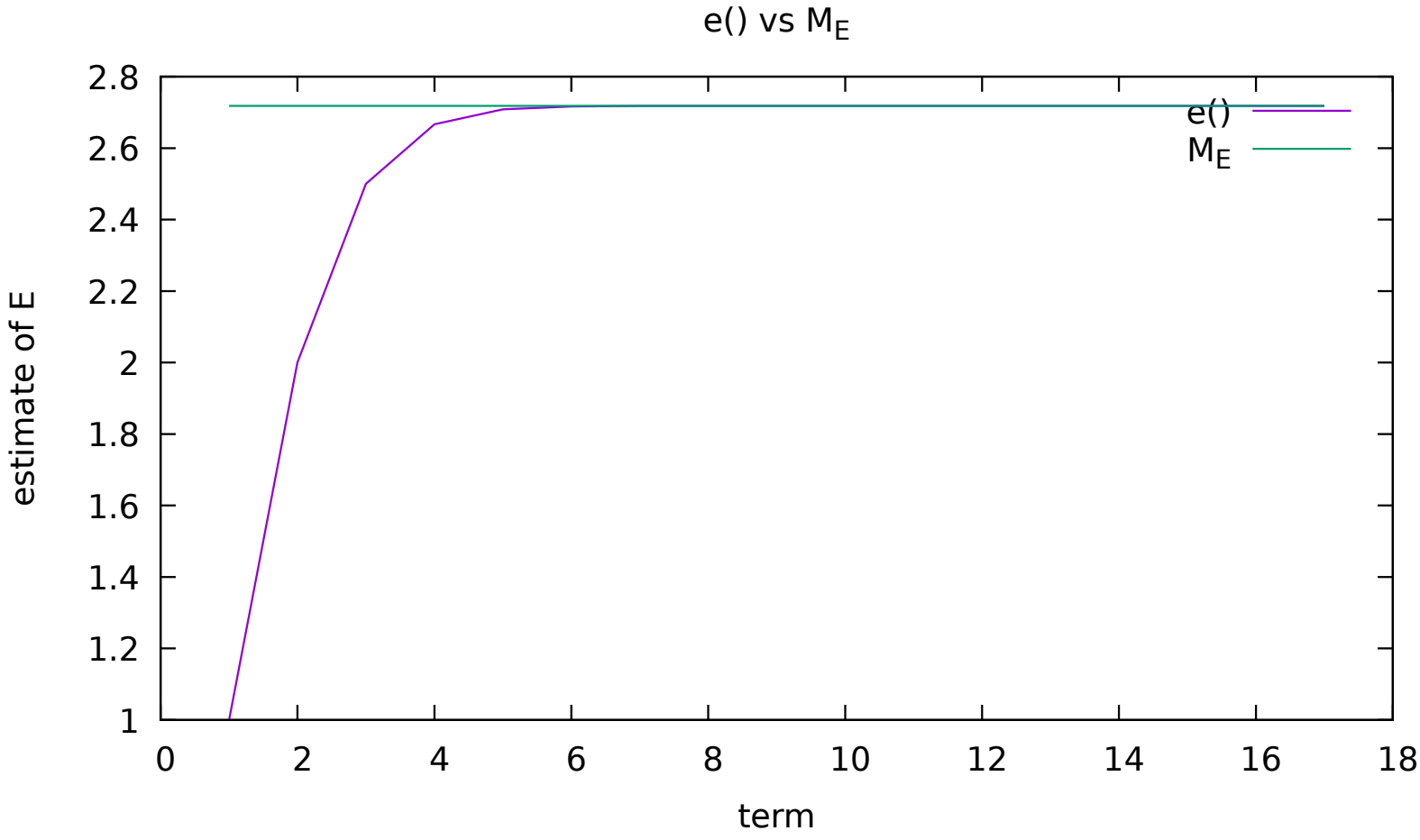
The above graph shows the BBP formula for calculating PI, compared to the math.h (c's standard math library) value of PI.

After around 11 iterations, the BBP function's estimation of PI was very close to math.h's estimation.

In fact, after just one iteration, the BBP estimate was already closer to M PI than my other function's estimations at 1 iteration.

Likely, BBP is quick to reach a relative accuracy due to only requiring simple multiplication.

However, I expect the formula to slow down when calculating larger decimal values for PI, since the function requires exponential calculations, and nested multiplication.

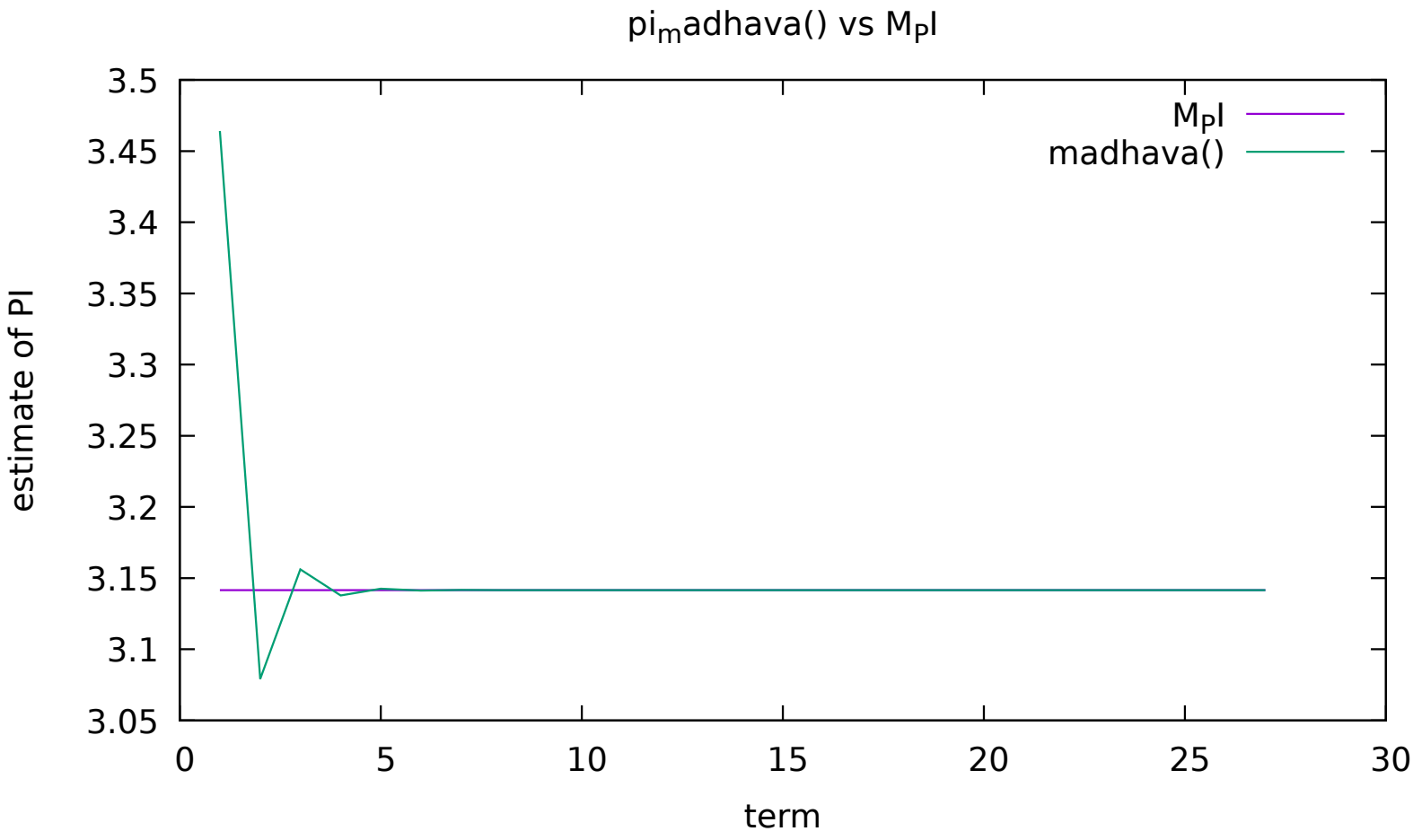

e() vs M_E
estimate of E
term
e()
M_E
2.8
2.6
2.4
2.2
2
1.8
1.6
1.4
1.2
1
0
2
4
6
8
10
12
14
16
18

E GRAPH ESTIMATION ERROR

The above graph represents a series for calculating e, written into a C function, e(), compared to the math.h (c's standard math library) value of e.

The series rapidly approaches M E, and nearly perfectly matches the values for M E. Using 16.15lf to display double values, there is no difference between the values of e() and M E at EPSILON decimals.

However, this is expected, since the series used in e()'s logic is the actual mathematical definition of the constant e. It should be the most accurate representation of E.


pi_madhava() vs M_PI
estimate of PI
term
M_PI
madhava()
3.5
3.45
3.4
3.35
3.3
3.25
3.2
3.15
3.1
3.05
0
5
10
15
20
25
30

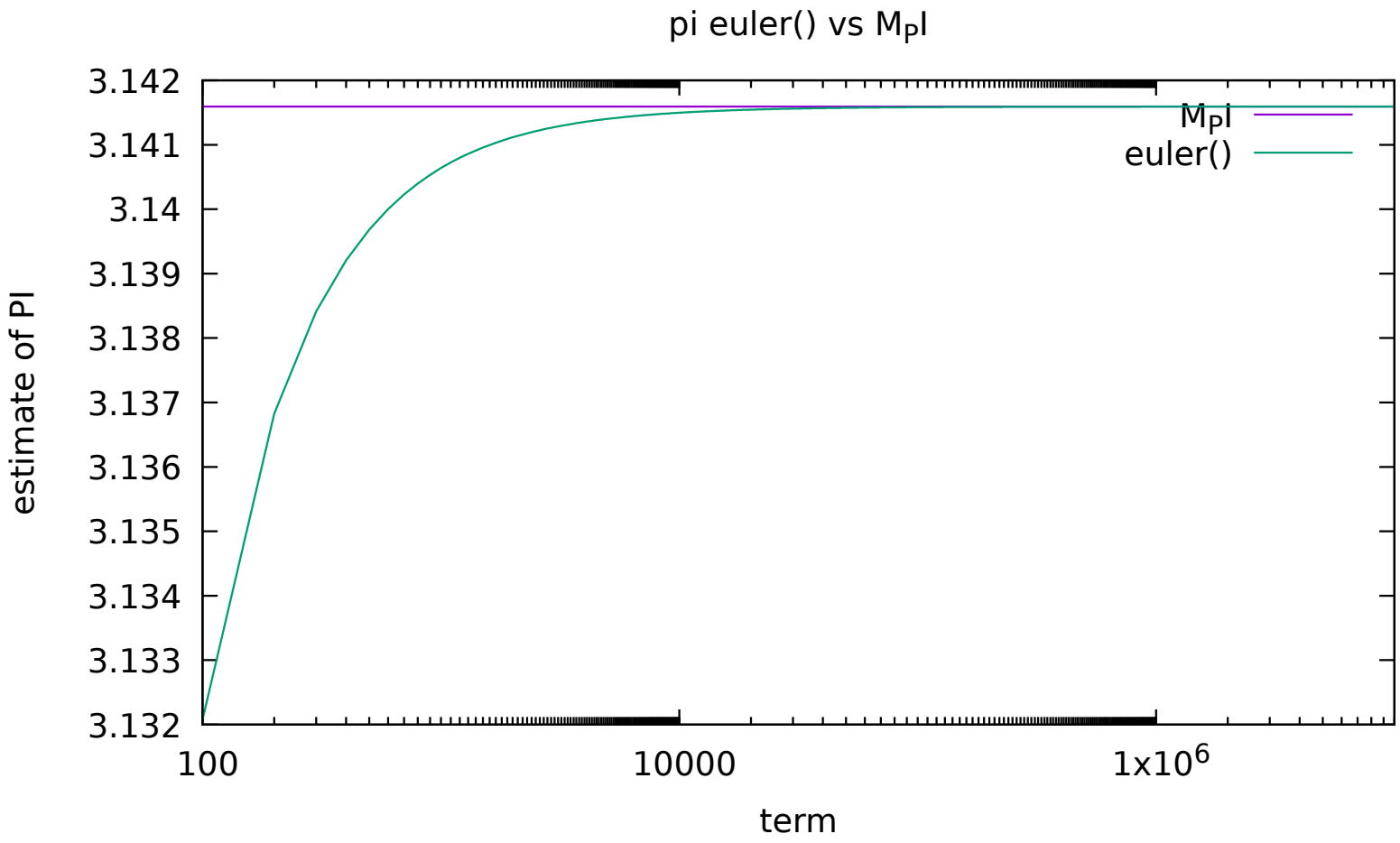
MADHAVA GRAPH ESTIMATION ERROR

The above graph shows the Madhava series for calculating PI, compared to the math.h (c's standard math library) value of PI.

There is fairly decent inaccuracy at a low number of terms. This is probably due to Madhava utilizing a sqrt function, which also includes error in values.

The error also fluctuates from too high, compared to M PI, to too low.

The fluctuation, from negative to positive, is likely due to the Madhava series taking the power of a negative number (-3).


pi euler() vs MPI
3.142
3.141
3.14
3.139
3.138
3.137
3.136
3.135
3.134
3.133
3.132
estimate of PI
100
10000
1x10^6
term
MPI
euler()

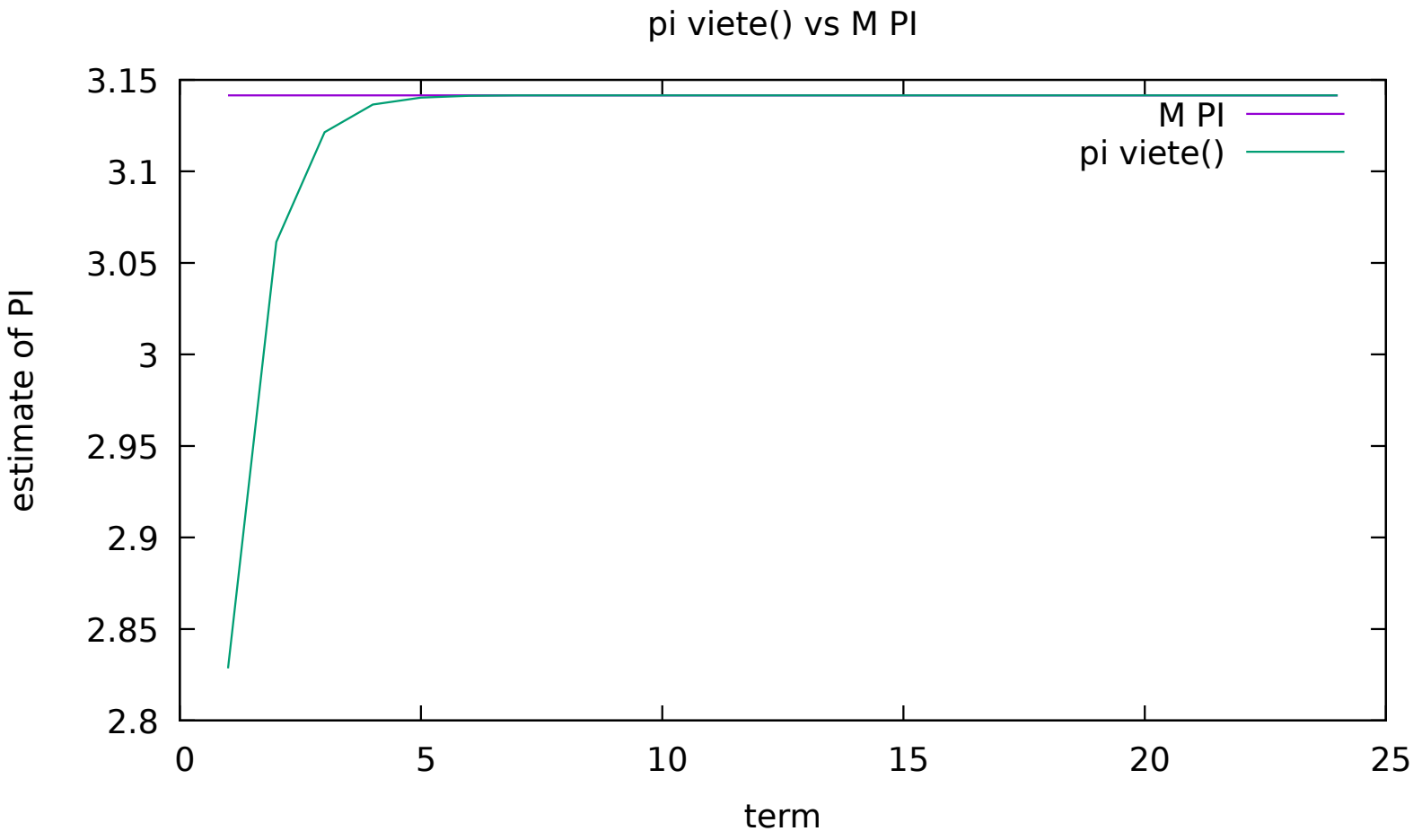
EULER GRAPH ESTIMATION ERROR

The above graph shows the Euler series for calculating PI, compared to the math.h (c's standard math library) value of PI.

Clearly, the Euler series approaches M PI the slowest of all the series used before.

It also has the greatest error compared to M PI.

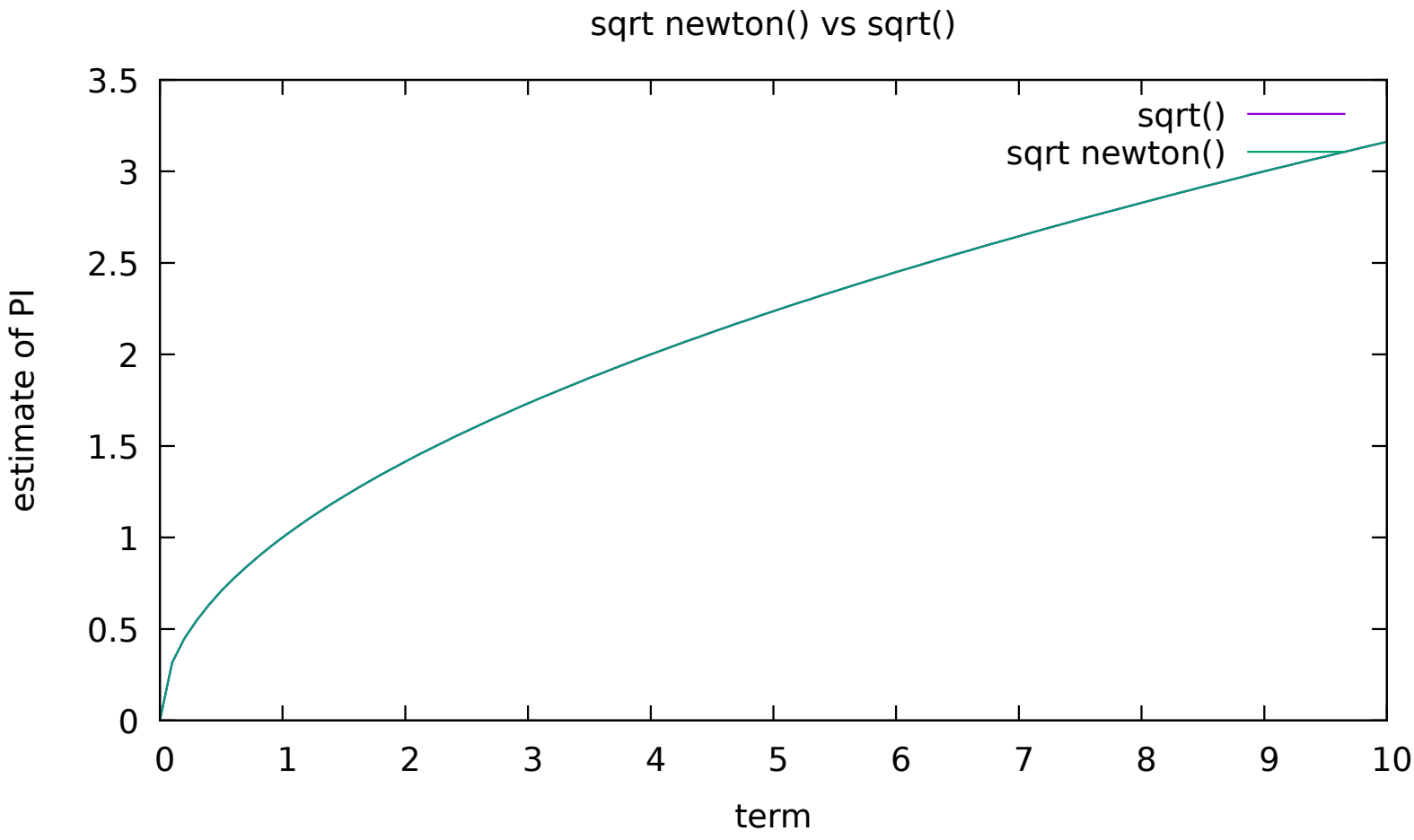
This greater error could be due to a greater number of calculations, with large decimal values. Since C sometimes errors when dealing with large floating point values.


pi viete() vs M PI
estimate of PI
term
M PI
pi viete()
3.15
3.1
3.05
3
2.95
2.9
2.85
2.8
0
5
10
15
20
25

VIETE GRAPH ESTIMATION ERROR

The above graph shows the Viete method for calculating PI, compared to the math.h (c's standard math library) value of PI.

The viete method has a small error, likely due to its use of a square root function, which has some error. After all, not all square root values can be perfectly represented with a limited number of data.


sqrt newton() vs sqrt()
estimate of PI
term
sqrt()
sqrt newton()
0
0.5
1
1.5
2
2.5
3
3.5
0
1
2
3
4
5
6
7
8
9
10

SQRT NEWTON GRAPH ESTIMATION ERROR

The above graph represents a method for calculating the square root of a value, written into a C function, sqrt newton(), compared to the math.h (c's standard math library) sqrt() function.

Sqrt newton() is incredibly accurate to sqrt() with a notable exception of sqrt newton(0), which actually does not return 0.

The small resulting value is probably due to rounding long values produced by division, due to limited data space to represent long decimals.