Approximating Pi using Archimedes method

Implementation

In my first attempt, I implemented this code as such. I used the formula given and iterated from nought through 50 to attempt to close in on a good approximation for pi.

```
# my starting value for p nought
p = 1/math.sqrt(3)
for i in range(0,51):
    # work out the number of sides of the shape inside the circle
    sides = 6 * pow(2,i)

    # work out the side length using the formula
    p = (math.sqrt(p * p + 1) - 1) / p
    # now times this by the number of sides to get an estimated circumference
    estimated_circumference = p * sides

    pi_estimate = estimated_circumference * 2
    print("Iteration", i, "Estimated value", pi_estimate, "Error", abs(math.pi -
pi_estimate))
```

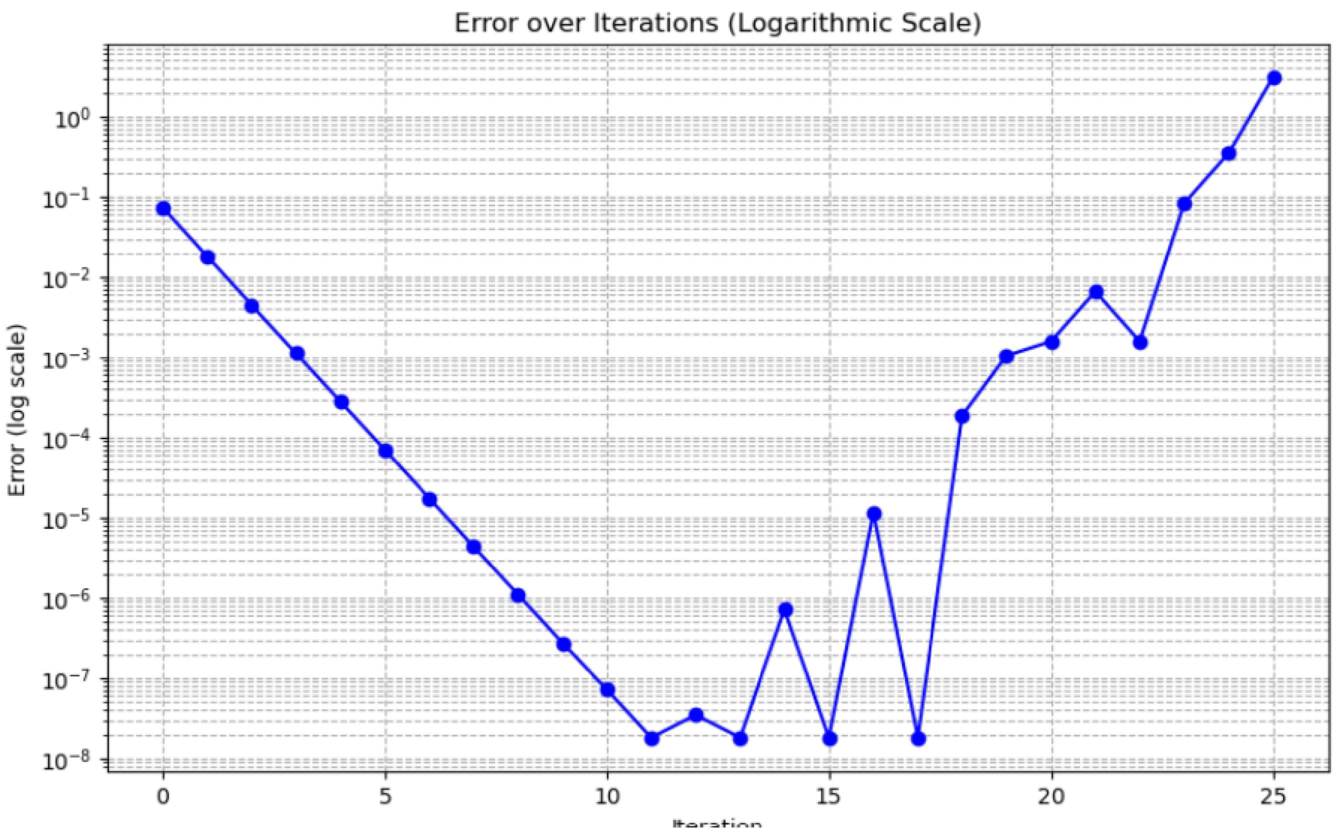
Results

Here are the results I got from this code.

```
Iteration 0 Estimated value 3.215390309173475 Error 0.0737976555836819
Iteration 1 Estimated value 3.1596599420975098 Error 0.01806728850771666
Iteration 2 Estimated value 3.146086215131467 Error 0.004493561541673685
Iteration 3 Estimated value 3.1427145996455734 Error 0.0011219460557803096
Iteration 4 Estimated value 3.141873049979866 Error 0.00028039639007282346
Iteration 5 Estimated value 3.141662747055068 Error 7.009346527508953e-05
Iteration 6 Estimated value 3.1416101765995217 Error 1.752300972857057e-05
Iteration 7 Estimated value 3.141597034323337 Error 4.380733543918325e-06
Iteration 8 Estimated value 3.141593748816856 Error 1.0952270628195038e-06
Iteration 9 Estimated value 3.141592927873633 Error 2.7428384008487683e-07
Iteration 10 Estimated value 3.1415927256225915 Error 7.203279839274046e-08
Iteration 11 Estimated value 3.141592671741545 Error 1.8151752101402963e-08
Iteration 12 Estimated value 3.1415926189008863 Error 3.468890685809356e-08
Iteration 13 Estimated value 3.141592671741545 Error 1.8151752101402963e-08
Iteration 14 Estimated value 3.141591935881973 Error 7.177078202857956e-07
Iteration 15 Estimated value 3.141592671741545 Error 1.8151752101402963e-08
Iteration 16 Estimated value 3.141581007579364 Error 1.164601042891178e-05
Iteration 17 Estimated value 3.141592671741545 Error 1.8151752101402963e-08
Iteration 18 Estimated value 3.1414061547376217 Error 0.00018649885217713684
Iteration 19 Estimated value 3.1405434924010995 Error 0.0010491611886935814
Iteration 20 Estimated value 3.1400068646909682 Error 0.0015857888988248803
Iteration 21 Estimated value 3.1349453756588517 Error 0.006647277930941442
Iteration 22 Estimated value 3.1400068646909682 Error 0.0015857888988248803
Iteration 23 Estimated value 3.224515243534819 Error 0.0829225899450261
Iteration 24 Estimated value 2.791117213058638 Error 0.350475440531155
Iteration 25 Estimated value 0.0 Error 3.141592653589793
```

After this iteration 25, the code crashed because of division by zero error.

Here is a screenshot of a plot of the prediction's error, over iterations.



This shows how the error started off quite great and gradually improved until the around the 13th iteration. Here the predictions started to fall off quite dramatically until it got worse than the first iteration and eventually crashed the program.

Interpretation

When I first ran the code, I expected my value of p to converge smoothly towards π , with each iteration reducing the error. However, unexpectedly, after a certain point, the value of p began to diverge and eventually approached zero. This was quite surprising because I had anticipated that the error would continue to decrease as the iterations progressed, but after iteration 11, the results worsened dramatically.

Upon investigating the cause, I realized that the issue was related to the accuracy of the computations. Specifically, my values were losing precision due to numerical instability in the original formula. As the iterations progressed, small errors began to accumulate and compound, leading to a significant deviation in the estimated value of p. This instability was caused by a subtraction of nearly equal numbers in the formula, which amplified rounding errors and caused the algorithm to break down.

Improved formulation

To address this issue, I first attempted to increase the precision of the calculations by switching to the np.float64 data type, assuming that this might resolve the accuracy problem. However, this change alone did not solve the issue, as the real source of the instability lay in the mathematical formulation itself.

The original formula was introducing significant numerical errors due to the subtraction of very similar values. This phenomenon, known as catastrophic cancellation, occurs when subtracting nearly equal numbers causes the result to lose significant digits of precision, which then compounds with each iteration.

To fix this, I reformulated the update equation to avoid such subtractions. I replaced the original equation with:

$$p_{n+1} = \frac{p_n}{\sqrt{p_n^2 + 1} + 1}$$

This new formulation avoids subtracting nearly equal terms, thus eliminating the primary source of numerical instability. The new formula allows the algorithm to proceed with much greater precision, leading to more accurate results and stable convergence towards π . This plot was generated using matplotlib and my results from the code above.

Improved Implementation

Here is my new code, which implements the new function.

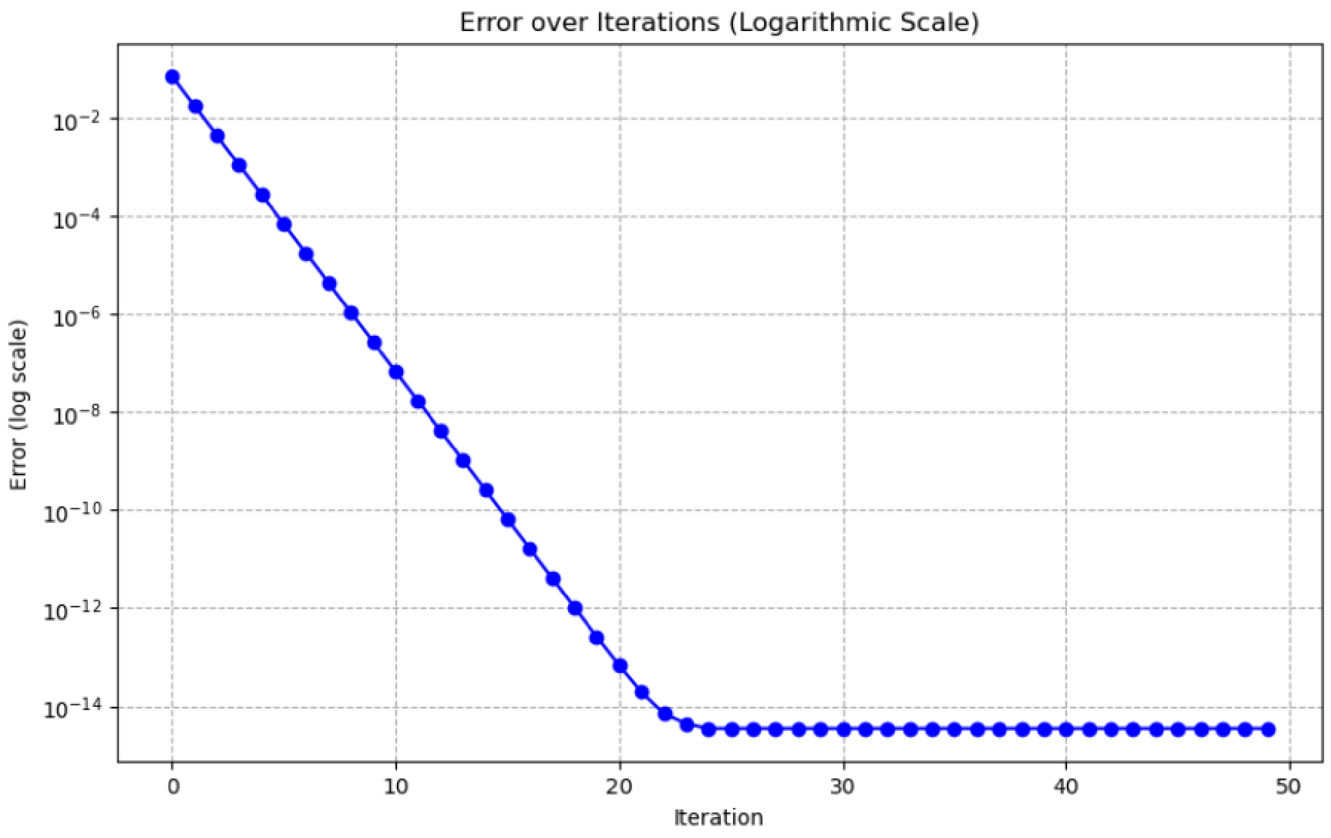
```
import numpy as np

# Initialize the value of p with high precision
p = np.float64(1 / np.sqrt(np.float64(3)))

for i in range(0, 51):
    # Calculate the number of sides of the shape inside the circle
    sides = np.float64(6 * np.float64(2)**i)

    # Use the numerically stable version of the recurrence relation
    p = p / (np.sqrt(p * p + np.float64(1)) + np.float64(1))

    # Multiply this by the number of sides to get an estimated circumference
    estimated_circumference = p * sides

    # Estimate pi by doubling the estimated circumference
    pi_estimate = estimated_circumference * np.float64(2)

    # Calculate the error between the estimated pi and the true value of pi
    error = abs(np.float64(np.pi) - pi_estimate)
    print("Iteration", i, "Estimated value", pi_estimate, "Error", error)
```

Improved Results

These results I got from this code were much improved, as the values close in on the real value for pi as I iterate, without the program ever crashing.

```
Iteration 0 Estimated value 3.215390309173473 Error 0.07379765558367968
Iteration 1 Estimated value 3.1596599420975013 Error 0.018067288507708223
Iteration 2 Estimated value 3.1460862151314357 Error 0.004493561541642599
Iteration 3 Estimated value 3.1427145996453696 Error 0.0011219460555764726
Iteration 4 Estimated value 3.141873049979825 Error 0.00028039639003196726
Iteration 5 Estimated value 3.1416627470568503 Error 7.009346705721953e-05
Iteration 6 Estimated value 3.1416101766046913 Error 1.7523014898213063e-05
Iteration 7 Estimated value 3.1415970343215283 Error 4.380731735142973e-06
Iteration 8 Estimated value 3.1415937487713546 Error 1.0951815614390625e-06
Iteration 9 Estimated value 3.1415929273850987 Error 2.7379530553872655e-07
Iteration 10 Estimated value 3.1415927220386157 Error 6.844882260992335e-08
Iteration 11 Estimated value 3.141592670702 Error 1.7112206762703863e-08
Iteration 12 Estimated value 3.141592657867846 Error 4.278053022943595e-09
Iteration 13 Estimated value 3.1415926546593083 Error 1.0695151431150407e-09
Iteration 14 Estimated value 3.141592653857174 Error 2.673807841802045e-10
Iteration 15 Estimated value 3.14159265365664 Error 6.684697240189053e-11
Iteration 16 Estimated value 3.141592653606507 Error 1.6713741501916957e-11
Iteration 17 Estimated value 3.1415926535939738 Error 4.1806558215284895e-12
Iteration 18 Estimated value 3.1415926535908403 Error 1.0471623568264476e-12
Iteration 19 Estimated value 3.141592653590057 Error 2.637889906509372e-13
Iteration 20 Estimated value 3.1415926535898615 Error 6.838973831690964e-14
Iteration 21 Estimated value 3.1415926535898127 Error 1.9539925233402755e-14
Iteration 22 Estimated value 3.1415926535898 Error 7.105427357601002e-15
Iteration 23 Estimated value 3.1415926535897976 Error 4.440892098500626e-15
Iteration 24 Estimated value 3.1415926535897967 Error 3.552713678800501e-15
Iteration 25 Estimated value 3.1415926535897967 Error 3.552713678800501e-15
Iteration 26 Estimated value 3.1415926535897967 Error 3.552713678800501e-15
Iteration 27 Estimated value 3.1415926535897967 Error 3.552713678800501e-15
Iteration 28 Estimated value 3.1415926535897967 Error 3.552713678800501e-15
Iteration 29 Estimated value 3.1415926535897967 Error 3.552713678800501e-15
Iteration 30 Estimated value 3.1415926535897967 Error 3.552713678800501e-15
Iteration 31 Estimated value 3.1415926535897967 Error 3.552713678800501e-15
Iteration 32 Estimated value 3.1415926535897967 Error 3.552713678800501e-15
Iteration 33 Estimated value 3.1415926535897967 Error 3.552713678800501e-15
Iteration 34 Estimated value 3.1415926535897967 Error 3.552713678800501e-15
Iteration 35 Estimated value 3.1415926535897967 Error 3.552713678800501e-15
Iteration 36 Estimated value 3.1415926535897967 Error 3.552713678800501e-15
Iteration 37 Estimated value 3.1415926535897967 Error 3.552713678800501e-15
Iteration 38 Estimated value 3.1415926535897967 Error 3.552713678800501e-15
Iteration 39 Estimated value 3.1415926535897967 Error 3.552713678800501e-15
Iteration 40 Estimated value 3.1415926535897967 Error 3.552713678800501e-15
Iteration 41 Estimated value 3.1415926535897967 Error 3.552713678800501e-15
Iteration 42 Estimated value 3.1415926535897967 Error 3.552713678800501e-15
Iteration 43 Estimated value 3.1415926535897967 Error 3.552713678800501e-15
Iteration 44 Estimated value 3.1415926535897967 Error 3.552713678800501e-15
Iteration 45 Estimated value 3.1415926535897967 Error 3.552713678800501e-15
Iteration 46 Estimated value 3.1415926535897967 Error 3.552713678800501e-15
Iteration 47 Estimated value 3.1415926535897967 Error 3.552713678800501e-15
Iteration 48 Estimated value 3.1415926535897967 Error 3.552713678800501e-15
Iteration 49 Estimated value 3.1415926535897967 Error 3.552713678800501e-15
Iteration 50 Estimated value 3.1415926535897967 Error 3.552713678800501e-15
```

Here is a screenshot of a plot of the prediction's error, over iterations.



This shows the errors decreases logarithmically over iterations. And with more iterations seems to close in on the real value of pi. This plot was generated using matplotlib and my results from the code above.

Interpretation

The improved algorithm gave much better results, providing a very accurate estimate of π. As shown in the graph, the error gets smaller with each iteration, and the estimate steadily gets closer to the true value of π. With more iterations, the estimate becomes even more accurate, showing that the new formula is stable.

This big improvement shows that the new formula avoids the problems that the original method had. By not subtracting numbers that are almost the same, the new method keeps the accuracy and prevents errors from building up over time. Because of this, the algorithm now converges more quickly and provides a reliable way to estimate π.