Numerical Methods - A Root Finding Comparison + Newton Fractals

Numerical Computation
Nicholas Pritchyk, Drew McFaul, Dylan Ginsberg

Software Package and Environment

- Package is called numerical-methods-python by cfgnunes
- And Newton Fractals by lettergram
- Environment is Jupyter Notebook

Stakeholder

- Mainly data scientists but can be used for security, fault analysis, and by electrical engineers for computing load flow of electrical distribution systems
- Also for Fractals! (Newton Method)

What we did

- Compare various methods of solving equations and look at performance (Bisection, Newton, Secant) and iterations
- Explore the differences and reasons for each methods performance
- Analyze process of Newton's method for creating fractals and experiment with various functions

Methods

Newton's Method

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

Compute derivative and give initial guess x_0, use process below to successively compute a more accurate approximation of the functions root.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Bisection Method

Prerequisites: Continuous function with two opposite signal values.

Method: Continuously divides the interval in two and therefore the ‘root’ is within the range. Done until the interval is adequately small.

A ‘binary search’ method

Secant Method

Prerequisites: Two initial guesses of x_0 and x_1 close to root

Uses secant lines of initial and succeeding values to get increasingly closer to the root approximation.

$$x_n = x_{n-1} - f(x_{n-1})\frac{x_{n-1} - x_{n-2}}{f(x_{n-1}) - f(x_{n-2})}$$

Findings

- Root Finding methods of bisection, Newton, and secant each ran 5 different standard and common polynomials to compute the root
- Package computes iterations taken to converge on root
- Newton method output the lowest amount of iterations

Polynomials used:

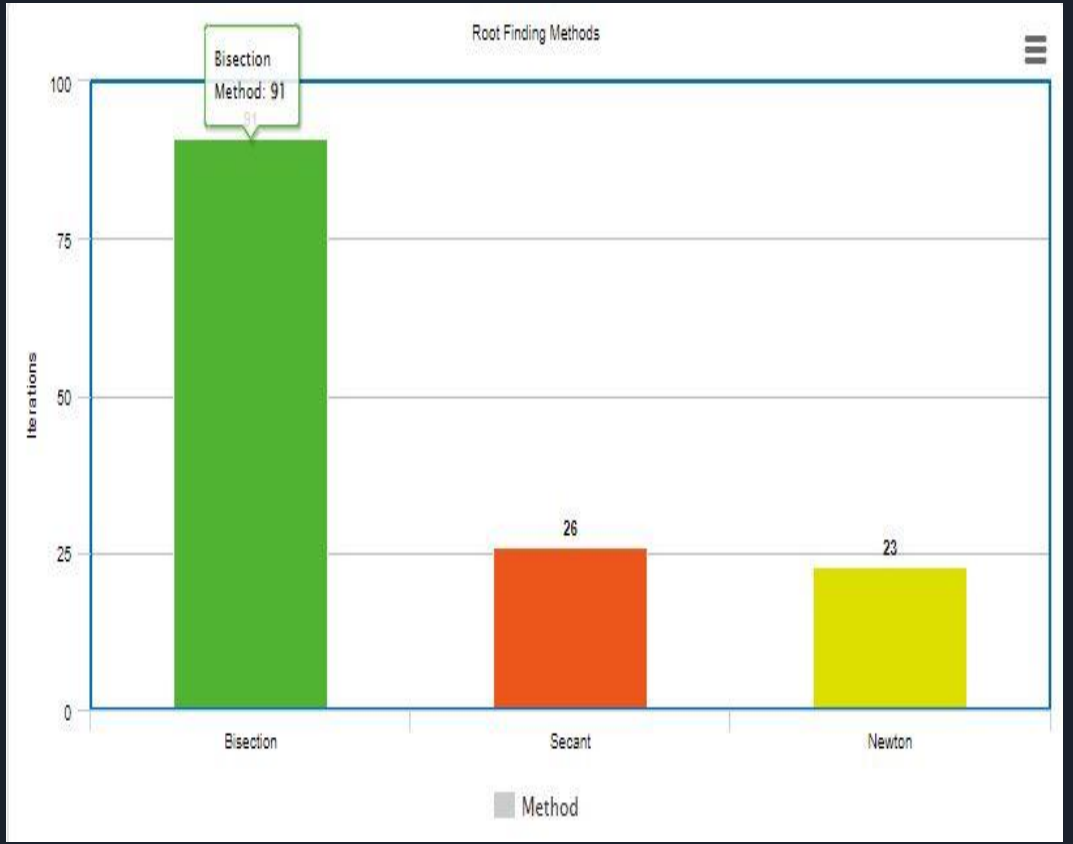
x^2 - cos(x)

20x^2 - 15xx - 10

6x^2 + 11x - 35

-x^2 + 6x + 18

4x^3 + x + cos(x) - 10



Cons of Each Method

Newton's Method

- Must be able to calculate derivative of function
- Possible instability

Secant Method

- May not converge
- Slower than Newtons

Bisection

- Converges slow
- Requires interval

Pros of Each Method

Newton's Method

- Fast

Secant Method

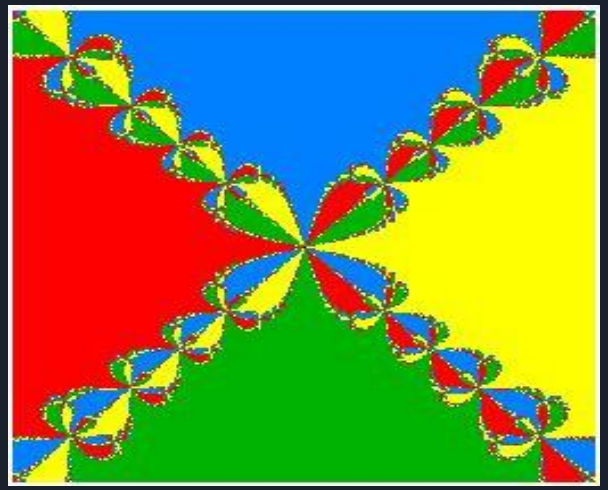
- Fast and does not require derivative like Newton’s

Bisection

- Reliable with guaranteed convergence
- No need to calculate derivative

Newton Fractals - ‘Best method’

- Newton's method root finding is dependent upon the initial guess of the close point of a possible root
- A slight change in the initial guess can provide significant differences in the outcome of iterations or the different roots
- Process involves choosing a function, preferably a complex function, for example a function such as f(z) = z^4 - 1, with four separate roots: +-1, +- i, results in:



Newton Fractals

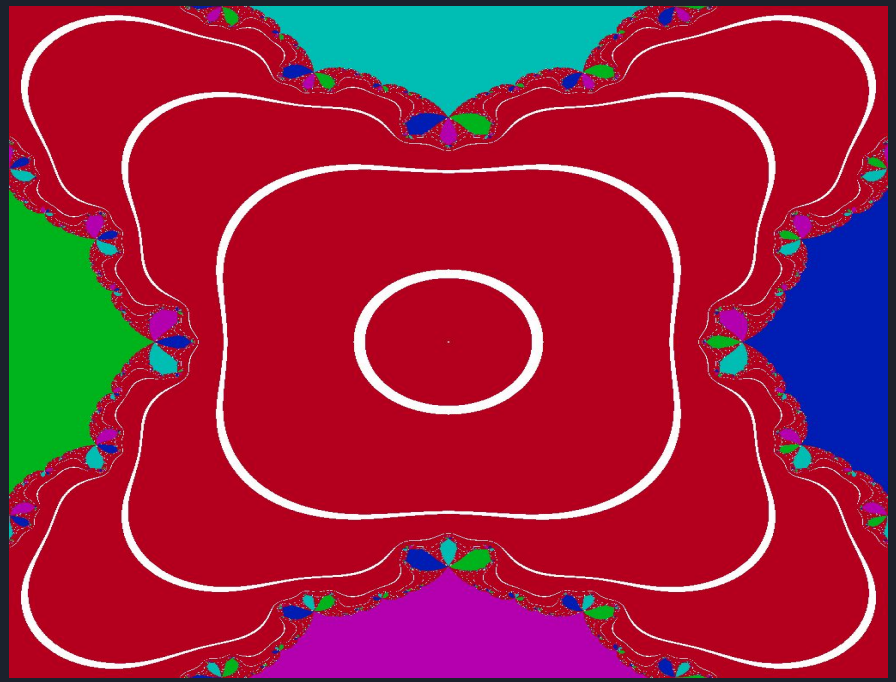
- To create a fractal, use a function with complex roots and run it through Newton's iteration
- Define a 2-dimensional grid in complex number plane
- Compute using Newton's method from each point in the grid and assign the correct converging root
- Plot the various values over each iteration an assigned color associated with the corresponding root

```
roots = []
for y in range(size):
    z_y = y * (y_max - y_min)/(size - 1) + y_min
    for x in range(size):
        z_x = x * (x_max - x_min)/(size - 1) + x_min
        root = newtons_method(f, f_prime, complex(z_x, z_y))
        if root:
            flag = False
            for test_root in roots:
                if abs(test_root - root) < 10e-3:
                    root = test_root
                    flag = True
                    break
            if not flag:
                roots.append(root)
        if root:
            img.putpixel((x, y), colors[roots.index(root)])
print(roots)
```

Experimental Fractal Output

Fractal drawing function output of:

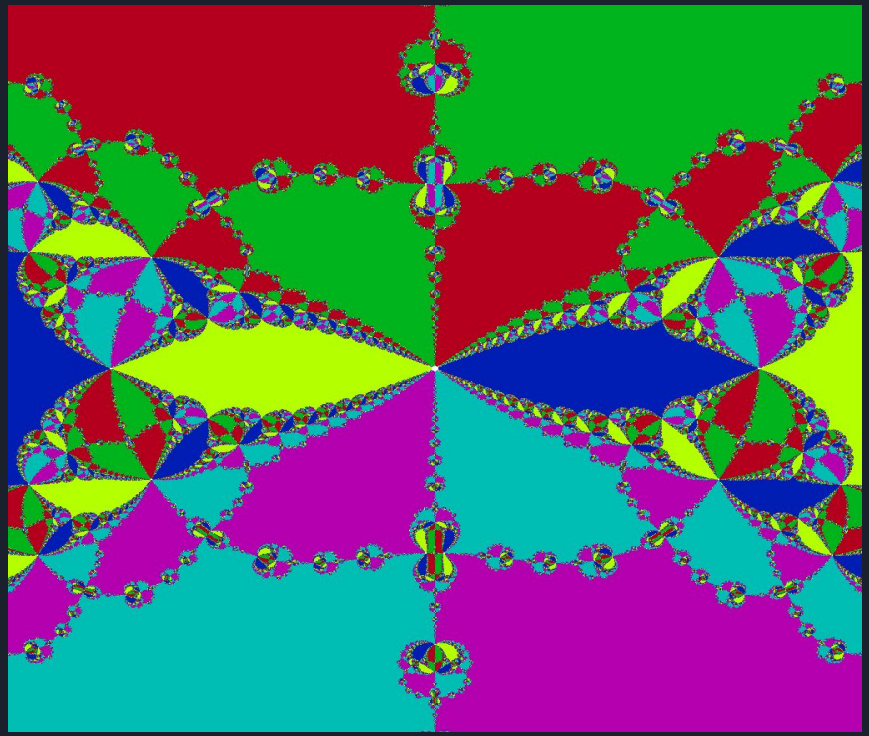
f(z) = z^5 - z



Experimental Fractal Output

Fractal drawing function output of:

f(z) = z^6 - z^2 - 1



Sources - Packages Used

Root-finding methods:

https://github.com/cfgnunes/numerical-methods-python

Fractal Design code:

https://github.com/lettergram/newtonfractals