Python Programming in Math
Duke University
Fall 2020
Professor Jeffrey Wong

Fractals and Dimension Finding

Scott Burstein, Sarah Northover

In this project, we write algorithms to create Newton fractals - the fractals related to Newton's method - with function:

```
Newton_Fractal.py
```

containing many parameters that allow the user to change the colour schemes and specify various levels of detail in generating a fractal.

Numerous example functions along with their calculated roots are provided in:

```
functions.py
```

The dimension of each fractal is calculated with the Minkowski–Bouligand dimension method, or the box-counting method in:

```
Box_Count.py
```

Due to the logarithmic nature of the calculation, high accuracy beyond 1.0e-2 was difficult to achieve in the time-frame of the project. This is because the detail (pixel count) required in the fractals needed to increase exponentially in order to add additional data points (beyond n=9) to the dimension solver's linear regression plot. Several assumptions were made for the code:

1. The image loaded is a jpeg with greyscale colours and minimal compression issues.
2. The image size is $(2^k \times 2^k)$ in pixels
3. The fractal is a different colour than the background

The following calculation was necessary to properly scale the Newton Fractals to an ideal detail.

$$f(x) = log_2\left(2.56 \cdot 200 \cdot x\right)$$

where:

$$\begin{aligned} 2.56 &= \text{interval size} \\ 200 &= \text{dpi} \\ x &= \text{scalar} \\ f(x) &= \text{exponent to range found in n (located in Box\_Dim())} \end{aligned}$$

The major disadvantage we found was that to increase the detail for a more accurate dimension calculation, the scalar value x was required to be a power of 2 (for integer values to be found for f(x)). The computation time for Newton_fractal.py to create a fractal thus increased exponentially in response to this.