Exploring the Intricate Beauty and Mathematical Properties of the Mandelbrot Set using Python: A Visual Journey

Irfan Rahman
Ivan Retamoso
Creating Data Science Pathways for STEM Student Success

Abstract

The Mandelbrot Set, discovered by Benoit Mandelbrot in 1979, is a captivating fractal structure that continues to fascinate mathematicians and enthusiasts. This Presentation explores the Mandelbrot Set using Python programming and visualizations, providing an introduction to fractals and their historical context. We delve into the mathematical foundations, explaining complex numbers and the iterative process defining the set. Python implementations and visualizations showcase the mesmerizing self-similarity of the set. Rendering and visualizing techniques are introduced, utilizing Python libraries such as Matplotlib and NumPy. Efficient computation and plotting strategies are discussed, including escape time and color mapping algorithms. These techniques reveal the intricate boundaries and infinitely complex patterns that emerge when zooming into the set's regions, showcasing the detail achievable through Python-powered visualization methods. Additionally, we explore the broader significance of the Mandelbrot Set, highlighting its connections to chaos theory, complex dynamics, and its profound influence on mathematics. Emphasizing its educational and aesthetic value, we underscore Python's accessibility as a powerful tool for comprehending and visualizing the Mandelbrot Set's intricate nature. This presentation offers a comprehensive and visually engaging journey into the Mandelbrot Set, showcasing its beauty and mathematical significance. Through Python programming and advanced visualizations, we enhance our understanding and appreciation for this mesmerizing fractal structure and its broader implications.

Introduction

The Mandelbrot Set is a famous mathematical fractal that is defined by iterating a simple mathematical formula for complex numbers. It consists of all the complex numbers c for which the iteration of the formula $f_c(z) = z^2 + c$ remains bounded when starting from 0. In simpler terms, it is the set of points in the complex plane that do not escape to infinity when the formula is repeatedly applied to them. The boundary of the Mandelbrot set displays intricate and infinitely complex patterns, making it a visually stunning and fascinating object of study in mathematics and computer graphics.
In 1980, Benoit Mandelbrot personally made an effort to visualize the Mandelbrot set in cooperation with IBM. The result of this endeavor is displayed below:

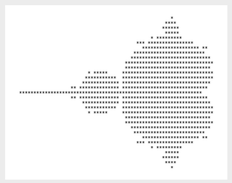

Figure 1. The first Mandelbrot set

By looking at it, he initially thought that this set was disconnected. However, as better versions of the Mandelbrot Set emerged, he changed his mind and concluded that it was connected. (This was formally proven by Douady and Hubbard using concepts from topology and complex analysis).

Methods and Materials

In today's era, with substantial computational capabilities at our disposal, a multitude of Mandelbrot images can be found online. Yet, generating these images demands sophisticated software and substantial computing resources. To clarify, our objective is to demonstrate that by skillfully employing the mathematical concepts and inherent traits of the Mandelbrot set, coupled with elementary Python coding (which is open source), we can effectively showcase the Mandelbrot set's core attributes. This approach allows us to both visualize the Mandelbrot set and unravel its essential characteristics in a straightforward and accessible manner.

Figure 2. Benoit Mandelbrot (1924-2010)

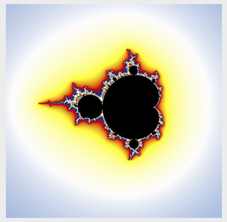

Figure 3. Mandelbrot Set today

Mathematics supporting our code

Mathematically speaking, the following is a necessary and sufficient condition for a point in the complex plane to belong to the Mandelbrot Set. A point c belongs to the Mandelbrot Set if and only if $|z_n| \leq 2$ for all $n \geq 0$. In other words, the absolute value of z_n must remain at or below 2 for c to be in the Mandelbrot Set, and if that absolute value exceeds 2, the sequence will escape to infinity, putting this together with the definition of the Mandelbrot Set, We were able to design the core of our code in python shown in Figure 4.

```
def mandelbrot(c, max_iter):
    z = 0
    for i in range(max_iter):
        z = z**2 + c
        if abs(z) > 2:
            return i
    return max_iter
```

Figure 4. Mandelbrot Set Iteration Function

Display of Mandelbrot Set in Python

To set the boundaries for the display of the Mandelbrot Set in Python, we used the command *np.linspace* applied horizontally and vertically to create a set of points that will either stay inside of the Mandelbrot Set or will escape to infinity.

```
def display_mandelbrot(width, height, x_min, x_max, y_min, y_max, max_iter):
    x = np.linspace(x_min, x_max, width)
    y = np.linspace(y_min, y_max, height)
    img = np.empty((height, width))
    for i in range(height):
        for j in range(width):
            img[i, j] = mandelbrot(x[j] + 1j * y[i], max_iter)
```

Figure 5. Mandelbrot Set Display Function

Results

Our visual journey through the Mandelbrot set using Python yields remarkable results. We showcase intricate fractal patterns, such as the famous "seahorse valley," "elephant valley," and "Mandelbrot's thumbprint." These visualizations provide a stunning representation of the self-similar structures and infinite complexity found within the Mandelbrot Set. Furthermore, we analyze the boundary of the set and observe fascinating phenomena such as infinitely intricate filaments and "mini Mandelbrot" sets, revealing the infinite richness of its mathematical properties.

In the Mandelbrot set, the number of iterations (max_iter) determines the level of detail in the generated image. The more iterations we use, the more accurate the representation of the set will be, and we'll see finer details and intricate patterns. The Mandelbrot set is shown in three pictures below, each of which highlights different iteration sizes to reveal its entrancing beauty.

The Mandelbrot Set for different numbers of iterations

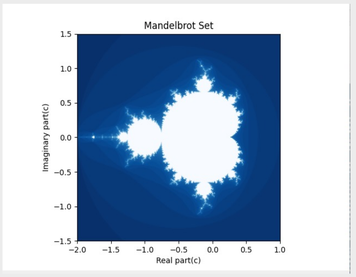

Figure 6. Using 50 iterations

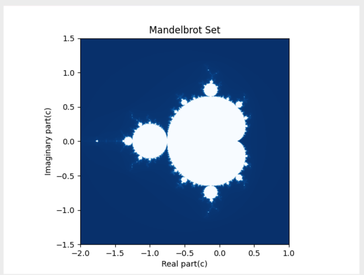

Figure 7. Using 300 iterations

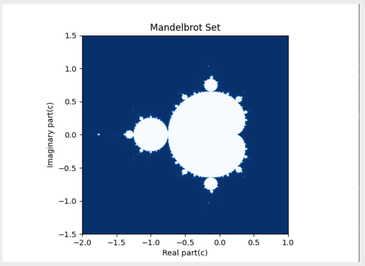

Figure 8. Using 1000 iterations

Discussion

Using Python programming we were able to explore The Mandelbrot Set, an infinitely complex and visually stunning fractal in mathematics. The iterative process behind this mesmerizing mathematical object generates its infinite complexity. Through Python, the set is visualized with its intricate patterns, such as the "seahorse valley" and "elephant valley," unfolding at various magnification levels. Key parameters like escape time and convergence rate are calculated to distinguish the set's interior and exterior regions. The significance of critical points in shaping the Mandelbrot Set and generating satellite fractals within it is investigated. The presentation emphasizes the broader implications of fractals in art, aesthetics, natural phenomena, and chaos theory, highlighting the beauty arising from seemingly chaotic processes. In our exploration of fractals, we observed their presence in the tangible world, manifesting in natural phenomena like clouds, lightning, the branching patterns of trees, the intricate structure of human lungs, tiny snowflakes, and even the complex form of broccoli. Can you come up with additional instances of real-world fractals? If you can, please share them with me.

Figure 9. Seahorse Valley

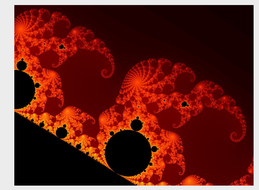

Figure 10. Elephant Valley

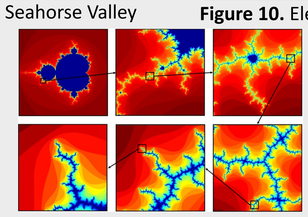

Figure 11. Self Similarity

References

1.The Mandelbrot Set
Numberphile https://youtu.be/NGMRB4O922I

2.Draw the Mandelbrot Set in Python
https://realpython.com/mandelbrot-set-python/

3.Iterations https://math.stackexchange.com/questions/16970/a-way-to-determine-the-ideal-number-of-maximum-iterations-for-an-arbitrary-zoom

4.MERLOT: The Multimedia Educational Resource for Learning
http://www.math.utah.edu/~alfeld/math/mandelbrot/mandelbrot.html

Acknowledgement

We are grateful to the NSF for the opportunity to work on this Data Science project.
Special thanks to Professor Oleg Muzician for his Python video tutorials, which greatly aided our project.
BMCC Math Jupyter Tutorial 1 https://youtu.be/dgZfvYo99Tk

BMCC Math Jupyter Tutorial 2 https://youtu.be/YwZILTNWWHY