3. Mandelbrot Set

> *"Clouds are not spheres, mountains are not cones, coastlines are not circles, and bark is not smooth, nor does lighening travel in a straight line."* – Benoit Mandelbrot

3.1. Aim

A fractal is a geometric shape that contains self-similar images within itself - you can zoom in on a section and it will have just as much detail as the whole fractal. Many objects in the natural world exhibit fractal behaviour. For example, the human circulatory system is a fractal. If you look at the blood vessels in your hand, they resemble the overall shape that the complete system takes on. The Mandelbrot set is also an example of a fractal - it is recursively defined and infinitely detailed.

The Mandelbrot set is a set of complex numbers C result from repeated iterations of the following function:

$$z_{n+1} = z_n^2 + C$$

with the initial condition $z_0 = 0$.

A given complex number C belongs to the Mandelbrot Set if :math`|z_n|`, the magnitude of z_n, remains bounded, i.e. does not diverge. If $|z_n|$ diverges, then C does not belong to the Mandelbrot set.

In fact, it can be shown that if $|z_n| > 2$ for some value of n, it will subsequently rapdily tend to infinity, i.e. diverge, meaning that C is not in the Mandelbrot set.

You can also assume that if $|z_n|$ has not diverged after 255 iterations, it will not diverge at larger values of n, meaning that C is in the Mandelbrot set.

3.2. Checkpoint task

Write an object-oriented Python program that will give a visual representation of the Mandelbrot set.

To obtain a visual plot of the Mandelbrot set, the complex plane should be represented as a 2D grid and the values of N (the number of iterations needed to reach the threshold $|z_n| > 2$) calculated for complex numbers C corresponding to points on the grid. As explained above, you should set an upper iteration limit of $N = 255$.

The value of N should then be converted to a colour and plotted on the grid.

You should explore values of C in the range $-2.025 \leq \mathrm{Re}(C) \leq 0.6$ and $-1.125 \leq \mathrm{Im}(C) \leq 1.125$.

The marker may ask you to display the Mandelbrot set for a different range of values. To receive full marks, your code must be capable of displaying a different region with minimal changes to the code. Ideally, this should be doable by changing the test code only.

All the plotting must be done using `matplotlib`.

▼ How many points to plot?

As you increase the number of points on the grid, you will not only increase the resolution of the display but will also significantly increase the computation time. You should experiment with your code to determine a suitable number of points to plot, but a grid of 512×512 points will probably give a reasonable balance between the resolution of the display and computation time.

3.3. Marking Scheme

- Marking Scheme for Mandelbrot Set Checkpoint

Note!

The code you submit for assessment **must be your own work**, not the output of Generative AI or the work of another person. The course team will use their best judgement to determine whether this is true. See the dis-

cussion in the Checkpoint Marking Scheme and the Generative AI Policy on the course Learn page for more information.

3.4. Optional extras

Optional extras are not marked and don't count towards the assessment for the checkpoint.

Write a program to display a Julia set. Julia sets are produced from the same formula as the Mandelbrot set but used in a different way. When making a picture of a Julia set, C remains fixed during the whole generation process, while the values of z_0 varies. The value of C determines the shape of the Julia set: in other words, each point of the complex plane is associated with a particular Julia set.

▼ Note

In order to draw a picture of the Mandelbrot set, we iterated the formula for each point C of the complex plane and always started with $z_0 = 0$.

Some of the most famous Julia sets are:

- $C = -1$
- $C = -i$: the Dendrite
- $C = 0.5$
- $C = -0.1 + 0.8i$: the Rabbit
- $C = 0.36 + 0.1i$: the Dragon

3.5. Relevant course sections

Studying the following course sections will help you complete this checkpoint:

- NumPy arrays
- Plotting using matplotlib

Additional material that you may find useful:

- Magic methods