Numerical Methods Project 1: Heat diffusion on Fractal Materials

Quentin Changeat

Kapteyn Institute, University of Groningen
q.changeat@rug.nl

These project uses knowledge and codes that you have developed in the course and its tutorials, but now applied to specific scientific problems. In groups of 2/3, you will have to write a report for the project and provide an executable python code (this should be a ".py", not a notebook, but you are encouraged to first work with notebooks before building your .py code) to produce all your checks and plots. For your report, this should contain relevant background materials, answers to the questions, and information specifying what each member of your group has done[1] You are strongly recommended to use Latex (Overleaf) for beautiful scientific reports. It is important that you describe well the procedures you have followed, as well as the answers to the questions. Please submit your report to Brightspace at latest on Friday June 13, 23.59h. **Don't forget to regularly save your work (at least in two places)!**

The background to the projects is given and discussed in the lecture of **March 20**. You need to create your project groups through Brightspace in the following week. After tutorial 5-6, the remaining tutorials will be dedicated to the projects, although also questions can be asked about the regular lectures and other materials from the course. Sometime in **the first week of June**, we like to have a meeting with your group during the tutorial to check whether you are on track. Please indicate in which tutorial this should be. Outside these times we are also available for any further questions you have.

[1] Submissions will be checked for excessive similarities between groups. Where there is circumstantial evidence of unfair means (including for example, sharing or distributing; copying and pasting from work that is not your own; passing other's work as your own, etc.), we reserve the right to lower the mark for this assignment significantly.

1 Project background

Fractals are infinitely complex, self-similar geometric structures that look similar at different spatial scales. They often appear in nature and are highly relevant in astronomy: fractals can describe large-scale cosmic structures in the universe, dust grains in the interstellar medium and planetary atmospheres, snow flakes. They also appear in engineering design (for radiators and other cooling elements). More generally, they are simply some of the most beautiful mathematical concepts. Fractals are often generated using iterative methods, which is what we are going to use in this project.

2 Simplest fractal: Mandelbrot Set (mark: 2.0)

The simplest and most popular fractal set is the Mandelbrot set. This set is formed by considering the following iterating formula:

$$z_0 = c \tag{1}$$

$$z_{n+1} = z_n^2 + c \tag{2}$$

where c is a defined complex number of the form $c = a + ib$.

Q1) Code this iteration formula in a well-designed python code. Test it for $n = 20$ iterations for the numbers $c = -0.5$, $c = 0.1 + 0.3i$, and $c = -2$. What happens?

Q2) By observing how this iteration formula behave, we can define the Mandelbrot set. A number c is a member of this set if the iteration formula converges. It is not a member of this set if the formula diverges. Plot in a complex graph (i.e., where the x axis is the real part, and the y axis is the imaginary part of the number) a few values of c belonging to the Mandelbrot set.

Q3) Update your code in Q1 to obtain a "safe" version of the iteration formula. Here, safe means that the code returns z_n if the formula converges, and **None** if the formula diverges. You can consider that if at any point $|z_n| > 2$, the formula will diverge.

Q4) Test this formula for at least 10000 well-choosen random values of c and plot the Mandelbrot Set.

3 Sierpinsky Triangle (mark: 2.5)

The Sierpinsky Triangle is another interesting fractal that can be built using an iterative formula similar to the one of the Madelbrot set. The triangle pattern is constructed as follow:

a) First, define the corners of a triangle using points ABC. Let's say A = (-1, 0), B = (1, 0), and C = (0, 1).

b) Chose a random starting point for the algorithm, let's say D = (-0.5, 0.5)

c) Now, at each iteration, construct new points at equal distance between the current points and the corners of the triangle. For instance, in the first iteration, the algorithm creates E between D and A. E is located at E = (E_x, E_y) with $E_x = (D_x+A_x)/2$ and $E_y = (D_y+A_y)/2$. In this examples, we have E = (-0.75, 0.25). Similarly, F is created between D and B, and G is created between D and C.

d) Repeat the algorithm for each new point.

Q5) Code this algorithm in a well-designed python code. Test it for $N = 7$ iterations. What pattern is being made?

Q6) At each iterations, how many new points are created?

Q7) Try adding a fourth corner point to create a quadrilateral structure (try other quadrilaterals than squares).

4 Heat transport on a Sierpinsky surface (mark: 2.5)

Ok, this is now becoming fun. Spacecraft cooling systems (i.e., radiators) often use hollow shapes to try maximizing the cooling area. One such possible shape is a Sierpinsky surface. Let's try modeling how heat propagates on such complex surface by using the Sierpinsky triangle created in the previous section as a graph. A direct approach using finite methods is indeed difficult here due to the complex pattern and instead, we will try a graph approach using nearest neighbours.

Q8) Using the Sierpinsky algorithm created in the previous section, obtain the list of points for a pattern after 7 iterations. For each point, construct a corresponding list of nearest neighbours located at distance $r = 0.1$. For this question, you can use the following snippet:

```
from scipy.spatial import KDTree
tree = KDTree(list_of_points)
list_neighbours = [tree.query_ball_point(p, r=0.1) for p in
                                list_of_points]
```

Note that in this code, list_of_points refers to the list of points from the Sierpinsky pattern, and list_neighbours corresponds to the list of relevant nearest neighbours at distance $r = 0.1$.

Q9) Now, consider the heat equation :

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u \tag{3}$$

where u is the temperature, and $\alpha = 0.1$ is the thermal coefficient (in m^2/s). In discrete irregular structure (like fractals), the graph Laplacian of each point i can be approximated by:

$$(\nabla^2 u)_i \sim \frac{1}{|N_i|} \sum_{j \in N_i} (u_j) - u_i \tag{4}$$

where u_i is the temperature of point i in the graph, N_i is the list of nearest neighbours to point i (from the previous question), and $|N_i|$ is the number of nearest neighbours to point i.

Write down the heat equation for a graph problem.

Q10) Design a code to plot the time evolution of the temperature on a Sierpinsky plate if a heat source of $T = 500\,\mathrm{K}$ and radius $0.05\,\mathrm{m}$ is placed at $(x_\mathrm{H}, y_\mathrm{H}) = (0, 0.5)\,\mathrm{m}$. Evolve the plate temperature starting from ambient temperature and up to $t = 5000\,\mathrm{s}$ using a timestep of $dt = 5\,\mathrm{s}$.

5 Open Section (mark: 3.0)

We are now trying to design a structural piece for a spacecraft that thermally protects our instruments while connecting them to the main bus. This connection is a trapezoid that can be constructed using the following four corners: A = (0, 0), B = (1, 0.2), C = (0, 1), and D = (1, 0.8). Study the benefits of using a fractal-like surface for the connection.

Note: For fun, you can also (but don't have to) implement radiative cooling as:

$$\frac{\partial u}{\partial t} = -K_\mathrm{rad}(u^4 - u_\mathrm{ref}^4) \tag{5}$$

where K_rad is the Radiative rate of change (in $s^{-1}K^{-3}$) and u_ref the reference temperature.