Assignment 1 - TFY4235
Fractal drum

Thorvald M. Ballestad

March 20, 2020

Abstract

Investigating the eigenmodes of a fractal drum. Is it possible to say something about the shape of a drum from its eigenfrequencies?

1 Introduction

In this project we investigate the eigenmodes of a fractal drum. The infinte fractal shape is approximated by a finite recursion. The wave equation is solved in fourier space using a finite element method. We are interested in finding out what recursion level we are able to achieve and how accurately we can estimate the fractal dimension of the drum based on the numerical solutions to the eigenfrequencies.

2 Theory

The relevant theory is given in the exercise text[1]. The most important parts are repeated here.

The fractal is a Koch fractal, more specifically the Minkowski Sausage variant[2]. The iteration for creating this fractal is transforming each line segment into a square wave, shown in figure 1. Beginning with a square, apply

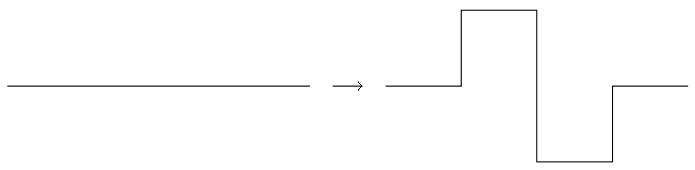

Figure 1: One iteration of the fractal generation.

the iteration for each segment. For each new segment, apply the iteration recursively. The square is denoted as level 0, after the first iteration we are at level 1 and so on.

The fractal is placed on a grid. For this grid we will define a grid constant δ. $\delta = 1$ corresponds to a grid where the distance between grid points is the same as the length of the shortest line segment on the fractal. For $\delta = 2$, there are two grid points for each shortest line segment, and so on.

The wave equation we want to solve, in Fourier space, is

$$-\nabla^2 U(\mathbf{r},\omega) = \frac{\omega^2}{v^2} U(\mathbf{r},\omega), \qquad \text{in } \Omega \tag{1a}$$

$$U(\mathbf{r},\omega) = 0, \qquad \text{on } \partial\Omega. \tag{1b}$$

Discretice $U(\mathbf{r})$ as $U_{x,y}$, where x and y are integers, and $U_{x,y} = U(xh\hat{x} + yh\hat{y})$, where h is the distance between grid points. For the discrete laplacian, we will use five point and nine point stencil. That is, central finite difference in two dimensions at second and fourth order defined as [3][4]:

$$h^2\nabla_5^2 U_{x,y} = U_{x+1,y} + U_{x-1,y} + U_{x,y+1} + U_{x,y-1} - 4U_{x,y} \tag{2a}$$

$$\begin{aligned} h^2\nabla_9^2 U_{x,y} = {} & \frac{4}{3}U_{x+1,y} + \frac{4}{3}U_{x-1,y} + \frac{4}{3}U_{x,y+1} + \frac{4}{3}U_{x,y-1} \\ & - \frac{1}{12}U_{x+2,y} - \frac{1}{12}U_{x-2,y} - \frac{1}{12}U_{x,y+2} - \frac{1}{12}U_{x,y-2} \\ & - 5U_{x,y}, \end{aligned} \tag{2b}$$

where the subscripts on the ∇^2-operator denotes five and nine point stencil respectively.

As described in [1] the Weyl-Berry conjecture predicts that

$$\Delta N(\omega) = \frac{A}{4\pi}\omega^2 - N(\omega) \tag{3}$$

scales as ω^d, where d is the fractal dimension of the drum. The fractal dimension of our fractal, when recursion level $l \to \infty$, is 3/2[1]. This quantity can therefore be used for numerical validation.

3 Method

3.1 Generating the fractal

The generation of the fractal structure is implementet quite naievly. For example, the array of points is not pre-allocated, but appended during each loop-cycle. The number of points for a given level is easy to calculate, so this would be a simple thing to fix, but as it is not time consuming at all, the naive method is kept. The fractal points are stored as integers. See 3.2.

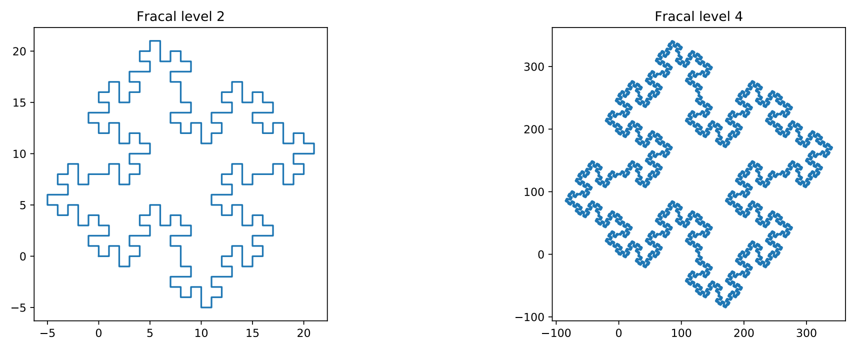

Figure 2: Fractal for $l = 2$ and $l = 4$.

3.2 The lattice

The lattice is a 2D array of integers. Each array element has a value that corresponds to either inside, outside, or on the border of the fractal. The lattice constant δ is implemented in such a way that the shortest segment on the fractal, correspond to δ steps on the lattice. Ie. if $\delta = 1$, there are no points on the lattice between the start and end point of the line segment on the fractal.

3.3 Inside or outside – determination of inner points

There are two methods implemented in the program for determining if a point is inside or outside the fractal. The only assumption about the structure that the methods makes, is that every line segment is either vertical or horizonal, and not diagonal.

The first method is a breadth first-type search. Given a point that we know is inside the structure, add its four cardinal points – above, below, right, and left – into a list. Visit each of the points in the list. For each of these points that are not border points, append their cardinal points to the list. Continue this process until the list is empty. Notice that the list is processed FILO. A FIFO process would give a depth first search. One could argue that this method is not complete, as it requires one starting point inside the structure. However, if one hits the "edge" of ones region, without hitting a border point, one knows that the starting point was not inside the structure, and thus the method is complete. Also, in the problem at hand we know trivially that the center point of our area is inside the curve. This method will be called middle out.

The second method is based on counting border crossings. For each point in our area, move in one direction, here left, and count the number of border crossings before reaching the end of the area. If it is odd, the point is inside, if it is even the point is outside. The complication is how to count border crossings for a discretized curve. Here, it is done as follows: The grid constant must be at least two. Without this, the problem is not solvable – one can easily construct cases where it would be ambiguous where the curve goes. When scanning, in this case left, check for each point if it is a border point. If it is, check if the point below it is also a border point. If that is the case, count it as a border crossing. This method will be called scan.

On a level 4 fractal with grid constant 2, the performance of the two methods is shown in table 1.

Method	Time[s]	Allocations	Memory[MiB]
Middle out	0.069	788K	32
Scan	3.33	54M	835

Table 1: Performance of two methods for determining if points are inside or outside. Found for level 4 with grid constant 2. The methods are described in 3.3

3.4 Calculation of eigenmodes

We convert (1) into an eigenproblem on matrix form. This is done by converting $U_{x,y}$ into a 1D array U' containing all the innner points of the fractal. A corresponding matrix, representing ∇^2 also made. The programatic challenge here is constructing the matrix, or more precisely mapping our structure onto U' and then find the entries in our matrix that match the operator defined in (2).

There are two aspects of the solution worth mentioning. When discussing the lattice, it was mentioned that each point is an integer representing either inside, outside or border. More specifically -1 represents outside and zero represents border. Any positive integer is inside. Each inner point is given an unique positive integer, representing that points index in U'. This way, we have a direct lookup from the grid to U'. One also need a lookup from U' to the grid. An array of the same dimension as U', with tuples with the grid coordinates does this.

The second aspect worth mentioning is the datastructure of the eigenmatrix. This matrix is extremely sparse. Therefore it is stored in compressed sparse column format, which is the standard sparse format in Julia.

Arpack is used to solve the system.

4 Results and discussion

4.1 Eigenmodes and eigenfrequencies

Level five is the highest recursion level for which I am able to solve the system. The eigenmatrix is then a 983041×983041 matrix, which takes 85MB of memory. The limitations then is obviously not the size of the matrix, but the memory Arpack requires to solve the system.

Table 2 shows the first ten eigenvalues, calculated at level five with grid constant 2. The first ten eigenmodes are shown in figure 3. The eigenmodes were calculated at level four with grid constant 2, using the five point stencil.

Five point	Nine point
5.752	5.753
8.625	8.628
8.625	8.628
8.791	8.795
8.839	8.842
9.200	9.202
9.200	9.202
10.768	10.771
11.533	11.537
11.866	11.869

Table 2: The 10 lowest eigenfrequencies, ω/v, calculated at level five with grid constant two. Five point and nine point refers to whether five point or nine point stencil was used.

4.1.1 Approximation of the Laplacian operator

4.2 Estimation of fractal dimension

As described calculation of fractal dimension can be used for numerical validation. The first one thousand calculated values for $\Delta N(\omega)$ is shown in

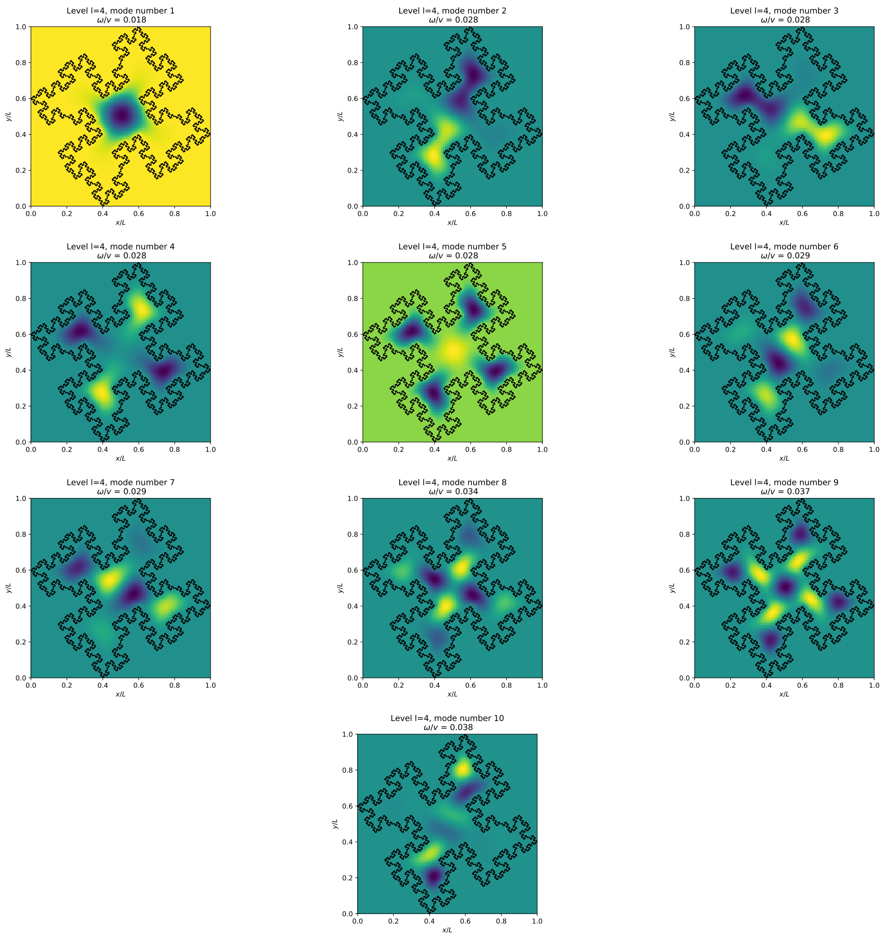

Figure 3: The first 10 eigenmodes

figure 4. The values were found for recursion level four of the fractal and with grid constant three. The values have distinct “drops” in an otherwise smooth curve, at $\omega \approx 0.07$ and $\omega \approx 0.15$. Because of this, the fitted curve is calculated using two different intervals. The first uses all the calculated values, while the second uses only the values up to the first “drop”. They are denoted “Fitted curve” and “Fitted curve for first section” respectively, and the calculated values d and d_2.

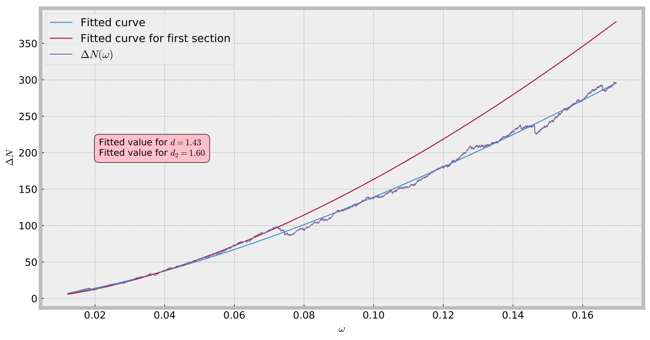

Figure 4: The first one thousand values of $\Delta N(\omega) = \frac{A}{4\pi}\omega^2 - N(\omega)$, where $N(\omega)$ is the integrated density of states. Calculated using recursion level four and grid constant three. The two fitted curves were found using all the points and using only the points until the first “drop”. Theoretically the value of d should be 1.5. The calculated values for d is 1.43 and 1.60 respectively.

4.3 Future extensions of the simulation

References

[1] TFY4235/FY8904: Computational Physics (spring 2020) Assignment1: Vibrations of Fractal Drums. Ingve Simonsen, *Department of Physics, NTNU.* `http://web.phys.ntnu.no/~ingves/Teaching/TFY4235/Assignments/TFY4235_Assignment_01.pdf`. Retrieved Mar 13. 2020.

[2] Wikipedia, Koch Snowflake, `https://en.wikipedia.org/wiki/Koch_snowflake`. Retrieved Mar 6. 2020.

[3] Wikipedia, Finite difference coefficient, `https://en.wikipedia.org/wiki/Finite_difference_coefficient`. Retrieved Mar 11. 2020.

[4] Wikipedia, Five-point stencil, https://en.wikipedia.org/wiki/Five-point_stencil. Retrieved Mar 11. 2020.