# Modelling Acoustic Scattering via Fractal Mesh Decomposition

Yuvan Raja

October 1, 2025

## 1 Introduction

In this project I develop a variety of strategies for code that produces fractal meshes for Koch Snowflake and Heighway Dragon curves. These meshes are created so that the mesh components can represent infinitesimal elements when approximating numerical integrals on fractal domains. The task of performing numerical integrals on fractal domains is an area of interest, as evaluating these integrals is a key challenge in finding the solutions of Integral Equations on fractal domains, such as those governing acoustic wave scattering.

## 2 Background

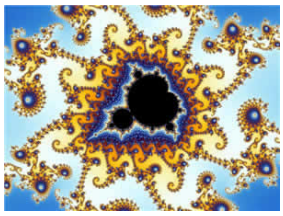
### 2.1 Fractals

To pin down a definition on what exactly a fractal is, is a hard task. Many mathematicians have tried to define the term but the resulting definitions all have differing degrees of strictness on what properties a fractal should have. As such the term fractal can describe a whole host of phenomena, but the essence of the concept is that a fractal is an object which contains detail on every length-scale.

One common way to generate objects with this fractal property is to exploit self-similarity, the idea that an object can contain smaller copies of itself. This motivates two methods of generating fractals, prefractal approximation and IFS attraction.

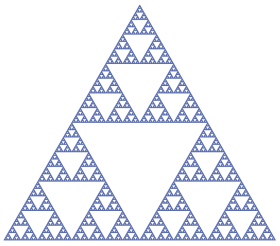
#### 2.1.1 IFS Attractors

One method of producing a fractal is via collecting together contraction mappings to form iterated function systems (IFSs). A contraction mapping is a transformation $f\colon X \longrightarrow X$, satisfying $d_X(f(\mathbf{x}), f(\mathbf{y})) < d_X(\mathbf{x}, \mathbf{y})$, where $X$ is a metric space with metric $d_X$, (Pollicott, 2022). Common examples of contraction mappings include geometric transformations on $X = \mathbb{R}^2$ that are composed of translation, rotation and dilation with scale fator $-1 < s < 1$. We can then



(a) Mandelbrot set (www.mathsisfun.com, n.d.).

(b) Coastline of Britain (rspatial.org, n.d.).



(c) Sierpinski Triangle (Martínez-Cruz et al., 2024).

Figure 1: Examples of fractals.

consider an IFS, which is a set of contraction mappings $\{f_i\}_{i=1}^{N}$. The Hutchinson operator F, associated with this IFS is defined as the topological closure of the union of contractions, so for $A \subset X$

$$F(A) = \overline{\bigcup_{i=1}^{N} f_i(A)}$$

Conceptually, we can think of the IFS as representing multiple contractions all applied simultaneously on a set.

The fractal set $S$ is then a non-empty compact set which is fixed under the Hutchinson operator $F$. This is because we can think of this fractal $S$, as being composed of $N$ smaller copies of itself, each given by one of the contractions $f_i$. So $S$ satisfies

$$F(S) = S$$

### 2.1.2 Prefractal Approximation

To produce a fractal via prefractal approximation, we can again consider an IFS with associated Hutchinson operator F. However, instead of directly seeking a solution of $F(S) = S$, we can instead build up a solution by repeatedly applying $F$, starting from an initial non-empty compact set $S_0$. Then,

$$S = \lim_{n \to \infty} F^n(S_0)$$

The limit is taken with respect to the Hausdorff metric, which describes how close two sets are to each other according to

$$d_{\mathrm{H}}(X, Y) := \max \left\{ \sup_{x \in X} d(x, Y),\ \sup_{y \in Y} d(X, y) \right\}$$

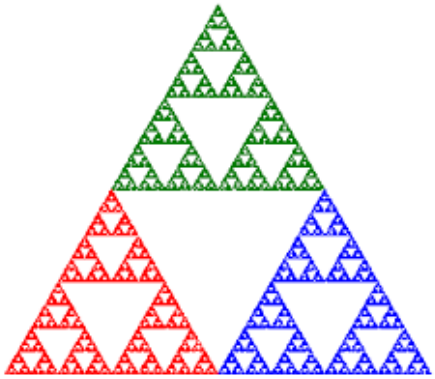


Figure 2: Sierpinski triangle showing 3 contraction mappings (larryriddle.agnesscott.org, n.d.)

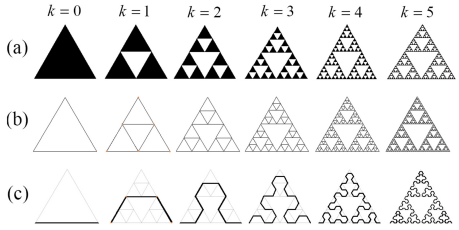



Figure 3: Three sequences of prefractal approximation to the Sierpinski triangle, with different initial sets $S_0$ (larryriddle.agnesscott.org, n.d.)

The contraction mapping theorem guarantees equivalence between the IFS Attractor and Prefractal Approximation approaches (Pollicott, 2022).

## 2.2 The Koch Snowflake

The Koch Snowflake is the unique fractal associated with IFS given by

$$f_1(\mathbf{x}) = \begin{pmatrix} \frac{1}{2} & \frac{-\sqrt{3}}{6} \\ \frac{\sqrt{3}}{6} & \frac{1}{2} \end{pmatrix} \mathbf{x}$$

$$f_2(\mathbf{x}) = \begin{pmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{3} \end{pmatrix} \mathbf{x} + \begin{pmatrix} \frac{1}{\sqrt{3}} \\ \frac{1}{3} \end{pmatrix}$$

$$f_3(\mathbf{x}) = \begin{pmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{3} \end{pmatrix} \mathbf{x} + \begin{pmatrix} \frac{1}{\sqrt{3}} \\ \frac{-1}{3} \end{pmatrix}$$

$$f_4(\mathbf{x}) = \begin{pmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{3} \end{pmatrix} \mathbf{x} + \begin{pmatrix} \frac{-1}{\sqrt{3}} \\ \frac{1}{3} \end{pmatrix}$$

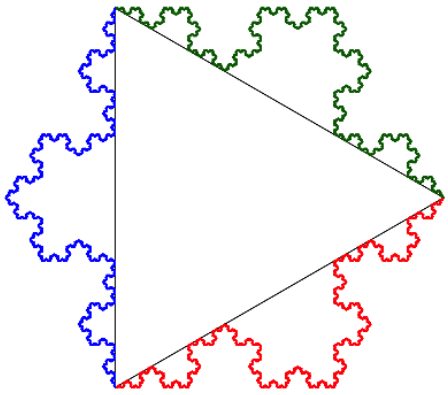


Figure 4: Koch Snowflake bordered by 3 Koch curves (larryriddle.agnesscott.org, n.d.)

$$f_5(\mathbf{x}) = \begin{pmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{3} \end{pmatrix} \mathbf{x} + \begin{pmatrix} \frac{-1}{\sqrt{3}} \\ \frac{-1}{3} \end{pmatrix}$$

$$f_6(\mathbf{x}) = \begin{pmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{3} \end{pmatrix} \mathbf{x} + \begin{pmatrix} 0 \\ \frac{2}{3} \end{pmatrix}$$

$$f_7(\mathbf{x}) = \begin{pmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{3} \end{pmatrix} \mathbf{x} + \begin{pmatrix} 0 \\ \frac{-2}{3} \end{pmatrix}$$

(larryriddle.agnesscott.org, n.d.)

The Koch snowflake is the primary fractal which I shall be investigating throughout my ORIS project, as an example of a self-similar fractal. It is interesting to remark that the Koch snowflake is 2 dimensional, but it's boundary is composed of three Koch curves which are $\frac{\log(4)}{\log(3)} \approx 1.26-$dimensional curves. The other example fractal that I use to illustrate ideas during this project is the Heighway dragon, which is the attractor of a two-function IFS.

## 2.3 Acoustic Equation

The theory of acoustic propagation and scattering is well-studied, and the fundamental equations that govern the dynamics of a sound wave are no different, even when we consider scattering off of a fractal surface (Caetano et al., 2024). In the same way in which Netwon's second law $F = ma$ might allow someone to setup the equations governing any particular mechanical system, we can use the following general equations to construct a particular equation which can be solved to determine the scattered waveform. These equations are the Helmholtz equation, which describes the oscillatory nature of a wave:

$$\Delta u + k^2 u = 0, \tag{1}$$

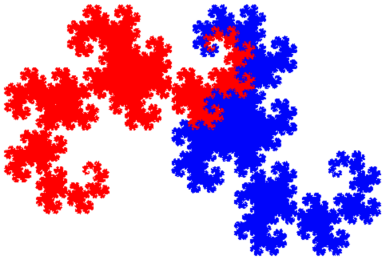


Figure 5: Heighway dragon showing 2 smaller copies (larryriddle.agnesscott.org, n.d.)

The Sommerfeld condition, which asserts that the direction of energy flow is from source to sink, rather than the opposite:

$$\frac{\partial u(x)}{\partial r} - \mathrm{i}ku(x) = o(r^{-n/2}), \qquad r := |x| \to \infty, \text{ uniformly in } \hat{x} := x/|x|. \quad (2)$$

Finally, there are scattering conditions which describe how waves scatter off a surface. The exact form of the scattering condition depends on the shape of the scattering surface, but it takes the form of an equation involving the acoustic potential $\mathcal{S}$ defined via a certain integral across the fractal surface $\Gamma$

$$\mathcal{S}\psi(x) = \int_\Gamma \Phi(x,y)\psi(y)\,\mathrm{d}s(y). \quad (3)$$

Crucially, in order to approximate solutions to these acoustic equations, it will be necessary to approximate the value of integrals defined across fractal domains.

## 2.4 Numerical Integration

As we have seen, the challenging step in approximating solutions of acoustic equations on fractal domains is the necessity to perform numerical integration across non-standard fractal domains. To approximate typical Riemann integrals on real domains, a method such as the midpoint rule is used, where a continuous function is replaced which a function that it is constant along small intervals of length $\delta x$. This method works well, because

$$\lim_{n\to\infty} \sum_{x_n=a}^{x_n=b} f(x_n)\delta x = \int_a^b f(x)dx$$

This idea can be reused even with a fractal domain, but now it is not possible to split the domain up into small rectangles, due to the irrational dimension of the boundary. Instead, we can split the domain into smaller copies of the fractal itself. Thus the primary aim of my project is to develop a program to create

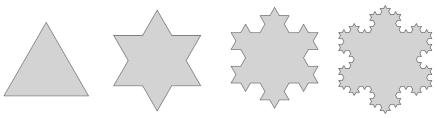


Figure 6: First 4 Prefractal Approximations of the Koch Curve, adapted from (larryriddle.agnesscott.org, n.d.)

meshes of Koch snowflakes to represent infinitesimal elements partitioning a larger snowflake. They should be meshed together in ways that will simplify the process of numerically solving differential and integral equations.

# 3 Solution Development

## 3.1 Storing and Drawing Snowflakes

In order to work with snowflakes using code, we must consider the necessary parameters that define a snowflake. To determine the size and position, we can use the coordinates of the centre, and the length of the diameter. There are two relevant orientations that a snowflake might be in, for a given mesh. They differ by a rotation of $\pi/6$ radians, or, equivalently reflection about $y = x$.

When we then come to draw the snowflakes that comprise a mesh, it is impossible to draw out the fractals exactly, as the boundary of the fractal is 1.26... dimensional. Therefore we instead choose to draw the 4th prefractal approximant, as seen in Figure 6.

## 3.2 IFS Decomposition

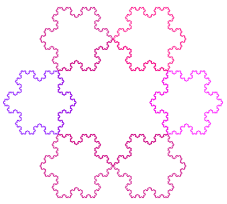
The key idea that allows us to generate meshes, is that the Iterated Function System (see Section 2.1) encodes the relevant information to partition a fractal into smaller copies of itself. For example, the IFS of a Koch Snowflake describes the partition of a Koch Snowflake into 7 smaller copies, as can be seen in Figure 7.
Then, in order to create a mesh composed of many smaller Koch snowflake tiles, we can repeatedly apply this IFS decomposition to the new smaller snowflakes, until we are satisfied with the resulting mesh.

## 3.3 Refinement Strategies

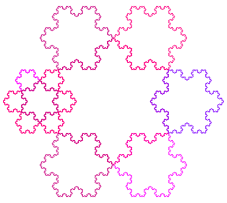
In the above process, there are many choices to be made as to which snowflakes should be decomposed. This means that it is up to us to decide upon a refinement strategy - an algorithm determining which snowflakes to decompose.
There are a few overarching ideas that guide which choices we should make. Recall that the purpose of the mesh is to allow functions to be approximated as

```
def IFSDecompose(self):
    if self.theta == 0:
        s0 = [self.x, self.y, self.d /sqrt(3), 1 - self.theta]
        s5 = [self.x + self.d/(2 * sqrt(3)), self.y + self.d/6, self.d / 3, self.theta]
        s4 = [self.x, self.y + 2*self.d/6, self.d / 3, self.theta]
        s1 = [self.x, self.y - 2*self.d/6, self.d / 3, self.theta]
        s3 = [self.x - self.d/(2*sqrt(3)), self.y + self.d/6, self.d / 3, self.theta]
        s6 = [self.x + self.d/(2*sqrt(3)), self.y - self.d/6, self.d / 3, self.theta]
        s2 = [self.x - self.d/(2*sqrt(3)), self.y - self.d/6, self.d / 3, self.theta]
    else:
        s0 = [self.x, self.y, self.d /sqrt(3), 1 - self.theta]
        s6 = [self.x + self.d/6 , self.y + self.d/(2 * sqrt(3)), self.d / 3, self.theta]
        s1 = [self.x + 2*self.d/6, self.y , self.d / 3, self.theta]
        s4 = [self.x - 2*self.d/6, self.y , self.d / 3, self.theta]
        s2 = [self.x + self.d/6, self.y - self.d/(2*sqrt(3)), self.d / 3, self.theta]
        s5 = [self.x - self.d/6, self.y + self.d/(2*sqrt(3)), self.d / 3, self.theta]
        s3 = [self.x - self.d/6, self.y - self.d/(2*sqrt(3)), self.d / 3, self.theta]
```

(a) Subprogram



(b) Snowflake after 1 decomposition.



(c) Snowflake after 2 decompositions.

Figure 7: IFS Decomposed snowflakes.

taking a constant value across subsections of the snowflake, rather than continuously varying across the snowflake. The smaller the snowflakes used in the mesh are, the more accurate this approximation will become, and the function can vary more smoothly. Of course, there is a trade off as the smaller the snowflakes are, the more constituents there will be, and it will be more computationally intensive to numerically solve the acoustic equation across these constituents. Therefore, we must try and prioritise certain regions of the snowflake.

For the approximation to remain accurate, it is more important to have a higher density of snowflakes in areas in which we expect the function to be varying steeply, rather than in areas in which we expect the function to be varying slowly. In the acoustic case, this area of interest is the boundary of the curve. Additionally, the symmetry of the snowflake is highly relevant. Exploiting symmetry allows computations to be sped up, as certain identical parts of the computation can be skipped. Thus, where possible we should preserve the inherent symmetry of the Koch snowflake - namely the sixfold rotational symmetry.

With these aims in mind, we now turn to implementing a range of refinement strategies, and evaluating their merits.

### 3.3.1 Fixed Number Strategy

The most basic strategy is to simply fix the number of subsnowflakes $N$, that we would like to have in our mesh, as determined by the computational limits due to the time needed to solve the acoustic equation. We can then refine snowflakes arbitrarily until we reach $N$

There are major issues with this approach. Firstly, it is not necessary possible to generate a mesh with a given number of snowflakes. This is because each times we perform a decomposition, we turn one snowflake into seven, increasing the number of snowflakes by exactly six. As we start with exactly one snowflake, only $N$ of the form $6k + 1$ can every be reached. This issue can be resolved easily, by stopping once we reach the closest number of the form $6k+1$ to $N$.

However, the more critical issues with this approach are that there is no guaranteed symmetry, and the refinement is not targeted towards the boundary.

### 3.3.2 Fixed Width Strategy

A second strategy is to produce a quasi-uniform mesh, in which snowflakes are decomposed until their diameter lies in a fixed interval $[d, \sqrt{3}d]$. In effect there are only two possible sizes of snowflake within this mesh.

This mesh is symmetrical, and has the property that every edge between two snowflakes connects snowflakes that differ in size by a factor of exactly $\sqrt{3}$. This is a highly desirable property because it means that there is only one way in which neighbouring snowflakes border each other. When propagating information during the solving of numerical differential equations, this means that we know that each neighbour lies in one of only six directions from a given

Figure 8: Fixed Width Strategy

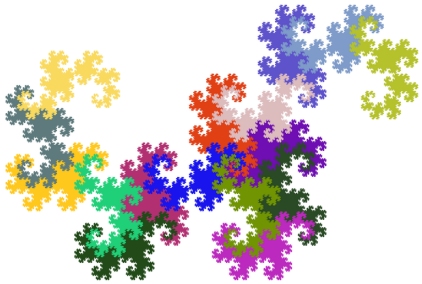
snowflake. This is much nicer than having to deal with a mesh in which a small snowflake is adjacent to a much larger snowflake, which would require considering many more cases. However, a disadvantage of this mesh is that the number of snowflakes required to reach a certain level of refinement grows steeply at an exponential rate. This is because no attempt is made to prioritise the boundary, and instead much unnecessary refinement is made at the centre.

#### 3.3.3 Radial Step Strategy

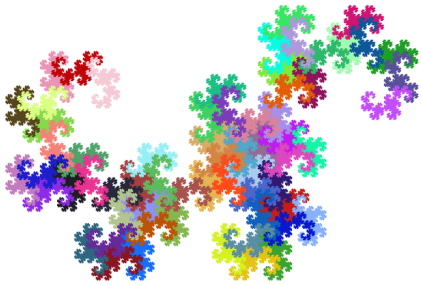
A third strategy is to focus entirely on the boundary, refining only those snowflakes which are within a fixed distance from the perimeter of the curve. This leads to a plot of density against radial distance from the perimeter resembling a step-function. Whilst this does prioritise the boundary effectively, an issue is the discontinuity in density, resulting in edges which occupy only a small fraction of the large central snowflake. As noted with the previous strategy, it is easier to numerically solve differential equations when neighbouring snowflakes are of similar sizes.

#### 3.3.4 Boundary-Heavy Strategy

A fourth strategy is to only decompose snowflakes which touch the boundary. Then once the new snowflakes have been created, we again pass through and decompose all the snowflakes which still touch the boundary. This process can keep on going for a fixed number of iterations. This will create a more gradual density profile, compared to the Radial Step Strategy.



(a)



(b)

Figure 9: IFS Decomposed Heighway Dragons.

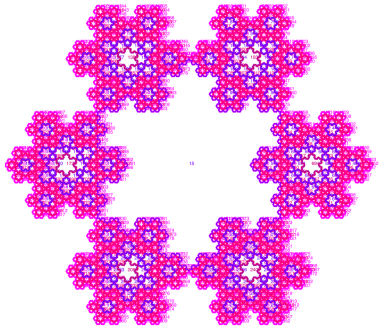


Figure 10: Radial Step Strategy

#### 3.3.5 Smoothed Boundary-Heavy Strategy

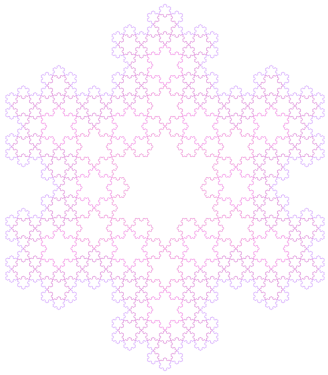
The final strategy I consider is a variation on the Boundary-Heavy Strategy. Instead of only decomposing snowflakes which touch the boundary, we instead only decompose snowflakes whose distance to the boundary is less than their diameter. The Boundary-Heavy Strategy is essentially the same as this strategy, but with the diameter threshold replaced with a tighter radius threshold. There is a tradeoff being made here, in that this strategy will produce more refinement further from the boundary and an increased number of snowflakes, however the mesh will be much smoother. This is because with this strategy, even though the size of snowflakes spans many orders of magnitude, any adjacent snowflakes differ in size only by a factor of $\sqrt{3}$, the ideal proportion for performing calculations. Equivalently, this mesh has the same smoothness as the quasi-uniform mesh, whilst still allowing for a great deal of the refinement to be focussed away from the centre onto the boundary.

**3.3.5.1 Distance to Perimeter** To implement the above strategy, we must determine the distance between the centre of subsnowflakes and the boundary. This is not a simple task, as the boundary itself is a 1.26... dimensional fractal. Thus we cannot simply check each vertex of the boundary and identify the closest vertex, as there are infinitely many vertices. Therefore it was necessary to develop an algorithm to approximate this calculation. The crux of the algorithm is that it is possible to narrow down the range of perimeter in which the closest vertex lies, in a manner akin to a binary search. To start, we can identify which Koch curve is closest to a snowflake by using the threefold symmetry of the snowflake. At each stage we can then halve the range in which the closest vertex lies by checking whether the snowflake centre is closest to the leftmost or rightmost portion of a Koch curve. Then, taking advantage of the self-similarity

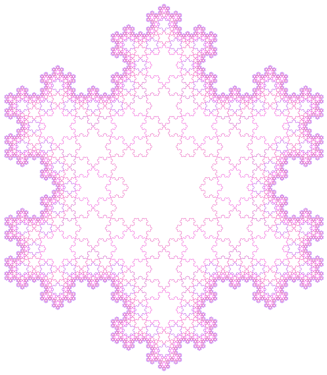
```
def MeshGeneratorBoundaryHeavy_1(s: Snowflake, numberIterations):
    snowflakes = deque()
    snowflakes.append(s)

    for i in range(numberIterations):
        nextIter = []
        for snowflake in snowflakes:
            if snowflake.touchesBoundary():
                nextIter.extend(snowflake.IFSDecompose())
            else:
                nextIter.append(snowflake)
        snowflakes = nextIter
    return snowflakes
```

(a) Subprogram

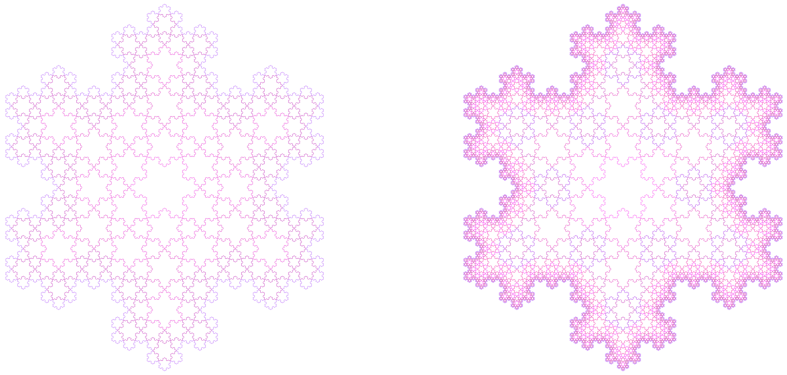


(b) after 3 Iterations.



(c) after 5 Iterations.

Figure 11: Boundary-Heavy Strategy.



(a) after 3 Iterations. (b) after 5 Iterations.

Figure 12: Smoothed Boundary-Heavy Strategy.

of the Koch curve, we can select the appropriate half of the Koch curve, which in itself is a smaller Koch curve. This allows the procedure to be repeated indefinitely.

# 4 Conclusion and Further Conjectures

Following my investigation, the conclusion I have arrived upon is that the Smoothed Boundary-Heavy Strategy produces meshes that are the most promising for solving the acoustic equation, for reasons layed out in the previous section. Therefore the academics I worked with will be using this mesh in their future simulations of acoustic propagation.

Following detailed inspection of the meshes that I produced, a few new claims have been conjectured by the academics with whom I worked. Firstly, it seems that the Smoothed Boundary-Heavy Strategy is guaranteed to produces meshes in which neighbouring elements differ in size by a factor of $\sqrt{3}$. If this property can be proved, it will greatly simplify later computations, and furthers confirms that this mesh is the optimal mesh for use in numerical approximation.

Finally, from inspection of the dragon mesh I produced, it has been recently conjectured that the uniform dragon mesh is composed of a finite number of disjoint lattices. This again could lead to a proof of a strong translational symmetry which would drastically speed up numerical computations.

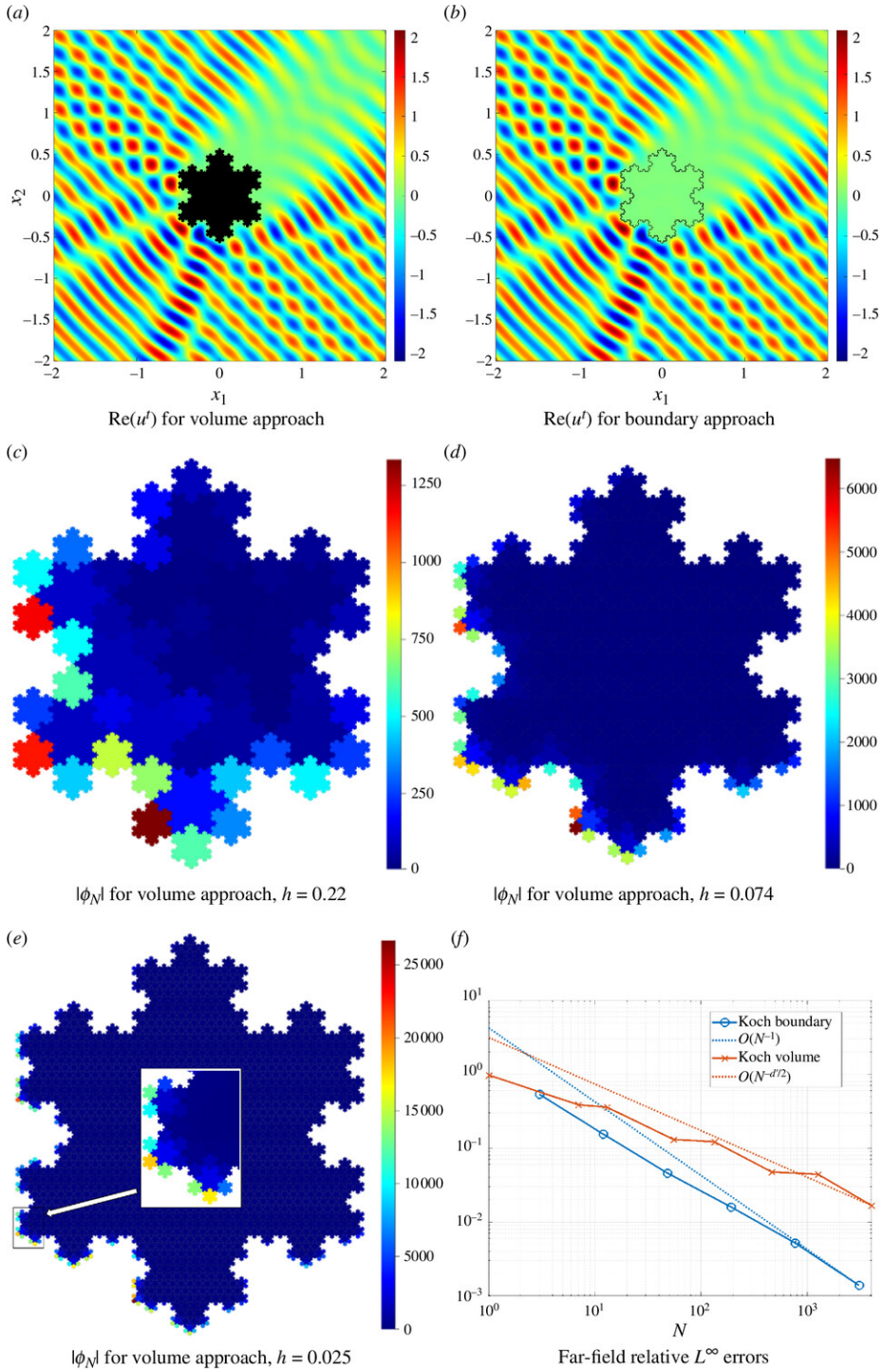



Figure 13: Acoustic Propagation from Koch Snowflakes, (Caetano et al., 2025)

# 5 Bibliography

1. Martínez-Cruz, M.-Á., Patiño-Ortiz, J., Patiño-Ortiz, M. and Balankin, A.S. (2024). Some Insights into the Sierpiński Triangle Paradox. Fractal and Fractional, [online] 8(11), p.655. doi:https://doi.org/10.3390/fractalfract8110655.
2. Caetano, A.M., Chandler-Wilde, S.N., Claeys, X., Gibbs, A., Hewett, D.P. and Moiola, A. (2025). Integral equation methods for acoustic scattering by fractals. Proceedings of the Royal Society A Mathematical Physical and Engineering Sciences, 481(2306). doi:https://doi.org/10.1098/rspa.2023.0650.
3. Caetano, A.M., Chandler-Wilde, S.N., Gibbs, A., Hewett, D.P. and A. Moiola (2024). A Hausdorff-measure boundary element method for acoustic scattering by fractal screens. Numerische Mathematik, 156(2), pp.463–532. doi:https://doi.org/10.1007/s00211-024-01399-7.
4. larryriddle.agnesscott.org. (n.d.). Koch Snowflake. [online] Available at: https://larryriddle.agnesscott.org/ifs/ksnow/ksnow.htm.
5. larryriddle.agnesscott.org. (n.d.). Heighway Dragon. [online] Available at: https://larryriddle.agnesscott.org/ifs/heighway/heighway.htm.
6. www.mathsisfun.com. (n.d.). Mandelbrot Set. [online] Available at: https://www.mathsisfun.com/numbers/
7. rspatial.org. (n.d.). 2. The length of a coastline — R Spatial. [online] Available at: https://rspatial.org/cases/2-coastline.html.
8. Pollicott, M. (2022). Topics in Fractal Geometry. Warwick Unviersity. [online] Available at: https://warwick.ac.uk/fac/sci/maths/people/staff/mark_pollicott/p3/main22.pdf.