A Walk Along A Space-Filling Curve

Mark Winter

April 12, 2024

I certify that this project report has been written by me, is a record of work carried out by me, and is essentially different from work undertaken for any other purpose or assessment.

Abstract

Space-filling curves are a class of curves which map a line onto a region, typically the unit square. They, in their very existence, challenge notions of dimension. Thus they caused a substantial upheaval in the understanding of the field of mathematics around the time of their invention. In this paper, we begin by describing background principles, including cardinality, dimension, Netto's Theorem, and Cantor's transfinites. We then cover five different space-filling curves: Peano's curve, Hilbert's curve, Sierpiński's curve, Lebesgue's curve, and Schoenberg's curve. For each curve, we provide a history of the mathematicians who discovered them and then proceed to describe the curves algebraically. Finally, we plot the resultant approximations to visually demonstrate the differences between the curves. The code used to generate the graphs to further clarify the curves themselves is included for the reader's investigative use at the end of this paper.

Contents

1 Introduction

The subject of space-filling curves is an unintuitive one. At first blush, one is tempted to consider the traditional concept of dimension to be substantial, meaning that morphisms of sets between dimensions should be"difficult" to construct in some way. This intuition proves to be true, to a point, but fails in some respects.

In this project, I investigate the initial discovery of space-filling curves, considering the history of the mathematicians who came up with five of the most well-known ones: Guiseppe Peano, David Hilbert, Wacław Sierpiński, Henri Lebesgue, and Isaac Schoenberg. To do this, I begin by considering the definitions of curves and dimension, as well as Cantor's transfinites and cardinal numbers. Netto's Theorem is another key component of my exploration of the character of space-filling curves. The proofs contained herein were worked out myself.

From here, I continue to plot the space-filling curves in Python, generating iterations of the curves to illustrate their significance. As space-filling curves transform a line to cover a space, graphical understanding of the curves is essential to grasp their individual characters and construction.

Finally, I consider the next steps that might be taken through my project, including graphing further curves and using different methodologies to produce those graphs.

My project is accessible to anyone with a basic grasp of topology and higher-level mathematics, for instance students taking or considering taking MT4526. I chose this project due to my own love of programming and mathematics, and the potential I found in this topic for the combination of both. I also deeply enjoyed exploring the proofs created by the mathematicians and considering the different ways in which they each considered space-filling curves.

2 Curves

In order to consider the possibility of space-filling curves, we must first define them.

The definition of a curve is the simplest place to start. A curve is most easily thought of as the path traveled by a point through a space over time. To break down this notion, let us consider a parameterization of the time it takes for the point to travel from start to finish, calling this time t, with origin start time 0. As such, we can describe the location of the point at a given time x by way of some function f(x), which maps a time in the interval [0,t] onto the point's location in the target space at that time.[1]

In particular, we want to be able to locate the point at any possible time between its start and end, meaning that f(x) must be defined on the densest ordered set we can muster. This set is the real numbers, $\mathbb{R}$.

Using this intuitive definition, a few claims can be made about the function f(x). First, it is taken that the point does not "jump" from place to place, or more particularly that for any small change in time, the point does not travel "too far". But this can, and mathematically should, be made more precise.

Let us consider an arbitrary time x, and call the aforementioned small change in time ϵ. What we require then is that the image of the set [x-ϵ, x+ϵ] not travel "too far" from the center point x. The most straightforward interpretation of this notion is to require that the image over f of this set [x-ϵ, x+ϵ] be bounded, say by some finite number δ.

Now we can clearly see that we have arrived at the classic epsilon-delta definition of continuity. Continuity is a very important consideration in the discussion of space-filling curves. Since they are the transformation of a line (and indeed, are curves) they must be continuous.

If no point on the curve is revisited, then a bijection exists between the time interval and the curve. This curve is known as a *Jordan arc*, or a *simple* curve. If the values of the function mapping the curve are equal at the endpoints of the time interval, then the curve creates a bijection with a circle, and is known as a *Jordan curve*.[1]

2.1 Space-Filling Curves

So what is a space-filling curve? A very simple definition is that a space-filling curve is a curve (which we have defined as a point moving through space) which touches every point in the space at least once. To consider this further, we need to explore the concepts of cardinality and dimension. Although this definition of a space-filling curve is relatively simple, the exploration of these concepts will help to explain their complexity and deep importance in the field of mathematics.

3 Dimension and Cardinality

To begin with, we must discuss Cantor's ideas of cardinality and his transfinites. We may then move into an exploration of dimension, and why space-filling curves so deeply upset previous notions of this topic. This section relies heavily on J. W. Dauben's book, *Georg Cantor: His Mathematics and Philosophy of the Infinite*.[2]

3.1 Georg Cantor

Georg Cantor (1845-1918) was a Russian mathematician who worked in Germany for the majority of his career, where he taught at the University of Halle. Although he initially made great strides in the field of trigonometry, Cantor is best known as the founder of set theory, where his discovery of cardinal numbers also allowed him to demarcate different infinities, as I will discuss in this section. He corresponded with David Hilbert on paradoxes in set theory, showing links between the different mathematicians' space-filling curves discussed herein.[3, 4]

3.1.1 Cantor's Cardinal Numbers

A cardinal number denotes the number of elements of a set. For a finite set, this is a natural number. For an infinite set, this is an infinite cardinal number, denoted as $\aleph_x$, where x denotes the rank of that cardinal number. The sequence of cardinal numbers is denoted as $0, 1, 2, 3, ...; \aleph_0, \aleph_1, \aleph_2, ...$

Cardinality is intimately linked with the notion of one-to-one correspondences, or bijections. Two sets have the same cardinality if and only if there is a bijection between their elements. This is intuitive for finite sets, as it only makes sense for two sets to have the same number of elements if you can create a one-to-one correspondence uniquely mapping each element from the first set to each element of the second set.

3.1.2 Cantor's Transfinites

In 1873, Cantor proved that the set of rational numbers, $\mathbb{Q}$, are countably infinite, by creating a one-to-one correspondance with the set of natural numbers, $\mathbb{N}$. The next year, he proved that the set of real numbers, $\mathbb{R}$, is uncountably infinite. He did this by working out that it was impossible to create a one-to-one correspondence to $\mathbb{N}$. Between this time and 1884, Cantor also worked out his theory of cardinality.

Cantor proved that infinite sets may have different cardinalities. Two main points to note here are that $\mathbb{R}$ has a greater cardinality than $\mathbb{N}$, and that a proper subset, one which does not include all elements of the set, may have the same cardinality as the original set. This second point is obviously impossible for finite sets.

All infinite sets with the same cardinality as $\mathbb{N}$, denoted $\aleph_0$, are called countably infinite. Examples include $\mathbb{Q}$, the set of all rational numbers, and the set of all real algebraic numbers, numbers which are the solutions to polynomials with integer coefficients. Also, any unbounded subset of $\mathbb{N}$ has the same cardinality as $\mathbb{N}$. All other infinite sets are uncountably infinite.

Cantor further proved in 1878 that any two finite-dimensional smooth manifolds, no matter their dimension, have the same cardinality, and thus a bijection can be found between them.[3] This is deeply important for our discussion of space-filling curves.

In my work within this dissertation, although I am considering the transformation from the unit interval to the unit square, I am doing so with finite iterations. The cardinality of the space-filling curve can approach infinite cardinal numbers, but for practical purposes I am not creating infinitely filled graphs.

3.2 Dimension

This section relies heavily on Erin Pearse's article, "An introduction to dimension theory and fractal geometry: Fractal dimensions and measures."[5]

The concept of space that most intuitively clashes with the idea of a space-filling curve is that of dimension. Intuitively, a line should not be able to map onto a square because it lacks the same kind of "width". A line is two-dimensional while a square is three-dimensional. Thus the idea of a line mapping onto a square seems impossible.

The idea of dimension, prior to Peano's construction of the first space-filling curve, can be summarized thusly, in the line of L. E. J. Brouwer: a single point has dimension zero, and if a set has points for which the boundaries of arbitrarily small neighborhoods all have dimension n, then that set has dimension $n + 1$.

This definition of dimension relied on the idea of space as fundamentally composed of a collection of points which can be identified and categorized. As a space-filling curve changes the dimension of an object by a continuous transformation, this definition can no longer work. In combination with Cantor showing the cardinality of the line and plane to be equal - meaning that a bijection can be constructed between them - this could upset the entire idea of dimension. If a continuous bijection can be found between a line and a plane, in other words, between sets of dimension one and two respectively, then dimension as a concept would become useless.

3.2.1 New Definitions of Dimension

Clearly, we still use dimension. It must therefore have a definition which cannot be destroyed by the mere existance of space-filling curves. Various methods of finding set dimension have been proposed, such as compass dimension, self-similarity dimension, box-counting dimension, Minkowski dimension, and Hausdorff dimension.

We may also investigate dimension from a topological perspective. The central goal of a dimension function $d(X)$ is to map some topological space X to an integer, understood as that space's dimension, with a dimension of -1 indicating the empty set and the dimension of all other sets being positive or zero. There are three common functions used in topology to accomplish this, namely small inductive dimension, large inductive dimension, and covering dimension.[6]

4 Netto's Theorem

Eugen Netto (1846-1919) was a German mathematician primarily known for his work on abstract group theory, combinatorics, and space-filling curves. He taught at the University of Strasbourg, the University of Berlin, and the University of Gießen.[3, 7] Netto produced one of the first major results in the field of space-filling curves, one which is central to understanding the topic as a whole.

Netto's theorem states that if f represents a bijective map from an m-dimensional smooth manifold μ_m onto an n-dimensional smooth manifold μ_n and $m \neq n$, then f is necessarily discontinuous. In other words, there may exist no continuous bijection between two smooth manifolds of different dimension.[3] This is more than sufficient for our needs, as the space-filling curves discussed here are simple mappings from [0,1] onto $[0,1]^2$, which are certainly both manifolds.

4.1 A proof of Netto's Theorem

This proof relies centrally on the topological notions of connectedness and compactness, and their invariance under continuous functions. Consider a bijective map $f : [0,1] \longrightarrow [0,1]^2$. We know that such a function must exist as the ambient spaces $\mathbb{R}$ and $\mathbb{R}^2$ are of the same cardinality. We also can say that any space-filling curve which maps the unit interval to the unit square is a function of this type. Note also that since f is bijective, its inverse function f^{-1} must also exist. Assume for contradiction that f is continuous.

Consider a closed set $A \subseteq \mathbb{R}$. Then, $A \bigcap [0,1]$ is bounded and closed and thus compact. Since f is assumed to be continuous, then $f[A \bigcap [0,1]]$, the image $\mathcal{A}$ of $A \bigcap [0,1]$ under f, is also compact and thus closed. Since f is bijective, it must also be the case that it maps closed sets to closed sets, and thus is also continuous.

Since f^{-1} is continuous, it maps connected sets onto connected sets. Consider removing a point p_0 from the interior of [0,1], and its image $f(p_0)$ from $[0,1]^2$. This would cause the resulting punctured interval $[0,1] \setminus p_0$ to be disconnected, whereas the punctured plane $[0,1]^2 \setminus p_0$ remains connected. This would imply that f^{-1} maps a connected set, the punctured plane, onto a disconnected one, the punctured interval, which gives us a contradiction. Thus, it follows that f cannot be both continuous and bijective.

5 Peano's Space-Filling Curve

Both this section and the next four (Hilbert, Sierpiński, Lebesgue, and Schoenberg) often refer to Hans Sagan's important text, *Space-Filling Curves*.[3]

5.1 Guiseppe Peano

Guiseppe Peano (1858-1932) was an Italian mathematician, notable as the founder of symbolic logic and for his work on mathematical analysis, the foundations of mathematics, and the creation of a formal universal logical language. He worked as a professor at the University of Turin from 1890 until his death. He is most well known in the field of analysis for his proof of the existence of a solution to a system of first order differential equations. Further, it was Peano who introduced the symbol $\in$ to indicate membership in a set and $\ni$ to indicate "such that".[8]

Peano invented space-filling curves as continuous surjective mappings from [0,1] onto the unit square, in 1890, although Hilbert also described these a year later in 1891. Peano and Hilbert were also linked through the 1900 International Congress of Mathematicians, where Hilbert famously outlined his 23 problems for the new millennium.

5.2 Peano's Curve

Peano defined a map f_p from the interval [0,1] to the unit square in terms of the operator

$$kt_j = 2 - t_j (t_j = 0, 1, 2) \tag{1}$$

as follows:

$$f_p(0_3 t_1 t_2 t_3 t_4 \dots) = \begin{pmatrix} 0_3 t_1 (k^{t_2} t_3)(k^{t_2+t_4} t_5) \dots \\ 0_3 (k^{t_1} t_2)(k^{t_1+t_3} t_4) \dots \end{pmatrix} \tag{2}$$

where k^ν denotes the νth iterate of k. Peano demonstrated that this mapping is both surjective and continuous. Notably, despite being continuous, the Peano curve is nowhere differentiable. [9]

This definition is most easily understood geometrically, using methods developed by Hilbert after Peano's initial construction, as a mapping of nine sections of the unit interval to nine subsquares of the unit square, with each corresponding interval-subsquare pair defined by a pair of t_n values. This notion can then be used to define a "geometric Peano curve" as an iterated function system propagated by self-similarity.

Figure 1, generated by the code in Appendix A, demonstrates this generation of an approximating polygon by the repeated application of linear operators, in this case numbering six iterations. The color of the line describes the position on the original pre-transformation interval as seen in the legend. This graphical overview allows us to see how the self-similarity of the curve is retained as its complexity grows.

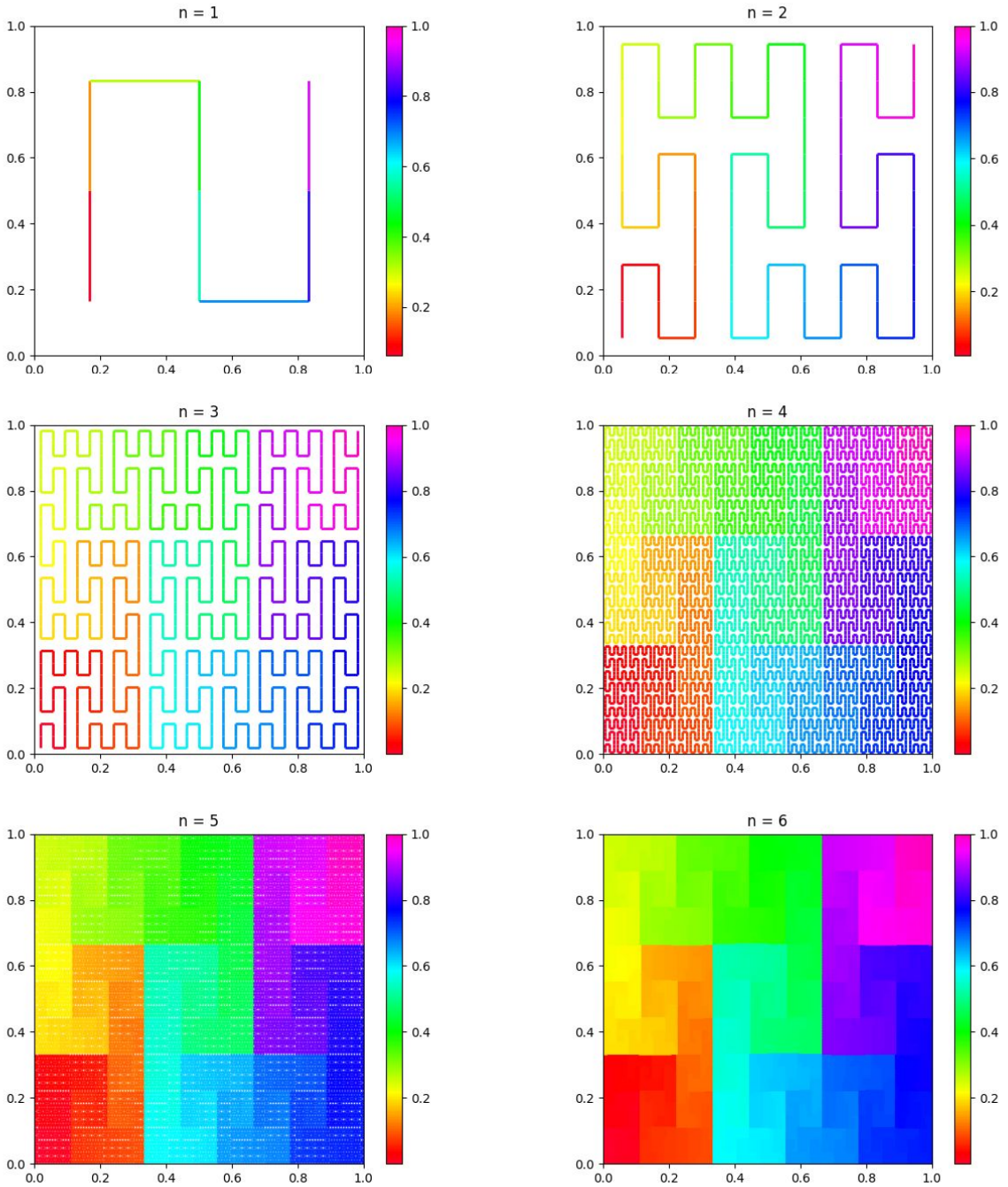


Figure 1: Peano's space-filling curve as generated by my code, found in Appendix A. These depict a mapping from the unit interval to the unit square, where the color of the line describes the position on the original interval as seen in the legend. The labels n = 1 to n = 6 describe the number of iterations of the iterated function system generating the curve.

6 Hilbert's Space-Filling Curve

6.1 David Hilbert

David Hilbert (1862-1943) was a Prussian mathematician who, it has been said, 'set the course' for 20th century mathematics. He held positions at the University of Königsberg and the University of Göttingen. He worked in all areas of contemporary mathematics, making contributions to algebraic forms, number theory, geometry, analysis, and linear systems, and has been described as the second greatest influence on the field of geometry after Euclid. His work on Hilbert spaces, complete and separable dot-product spaces, eventually led towards work on the convergence of Fourier series. He is further remembered for the twenty-three unsolved problems he posed to contemporary mathematicians in Paris in 1900. These included the continuum hypothesis, the well ordering of the reals, Goldbach's conjecture, the transcendence of powers of algebraic numbers, the Riemann hypothesis, and the extension of Dirichlet's principle. These questions had a huge influnce on mathematics in the 20th century.[10]

6.2 Hilbert's Curve

6.2.1 Geometrical Construction of Space-Filling Curves

Hilbert was the first to find a general geometric process by which to generate and thus construct an entire class of space-filling curves. This made space-filling curves geometrically sensible, and much easier to understand. He considered the following principle: if an interval $\mathcal{I}$ can be mapped continuously onto a square $\mathcal{Q}$, then after partitioning both into four congruent subintervals and subsquares respectively, each subinterval may be mapped onto each subsquare. These subintervals and subsquares may be taken as the new $\mathcal{I}$ and $\mathcal{Q}$ respectively and the process can repeat infinitely.

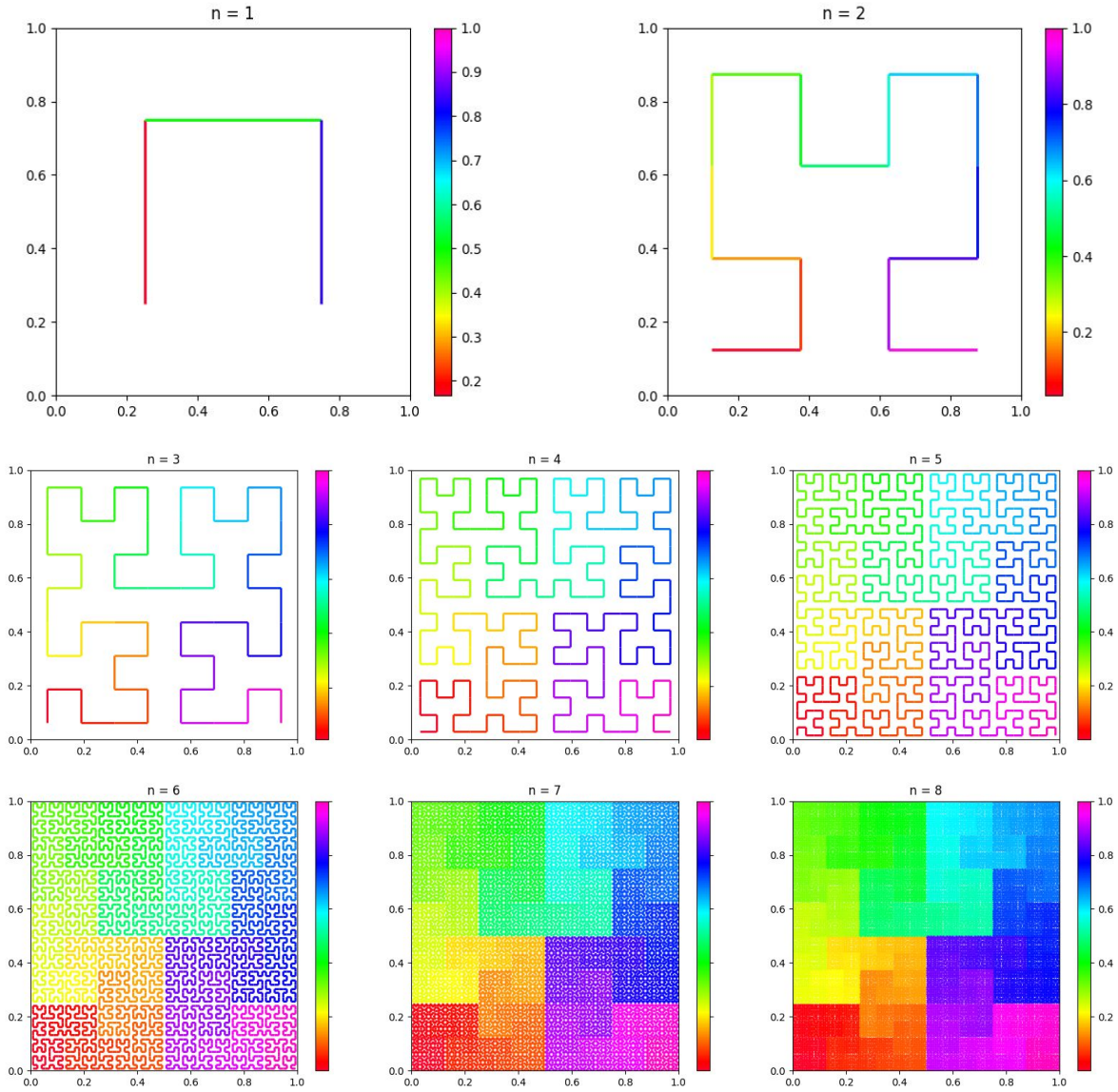
6.2.2 Hilbert's Curve

These subsquares and subintervals may be arranged such that their relationships are preserved through this process, in other words, such that if a square corresponds to an interval, its subsquares correspond to that interval's subintervals. This is the Hilbert's space-filling curve, which is also defined as:

Every point $t \in \mathcal{I} = [0, 1]$ is uniquely determined by a sequence of nested closed intervals generated by successive partitioning, the lengths of which shrink to zero. This sequence corresponds to a unique sequence of nested closed squares, the diagonals of which shrink to a point, and which define a unique point in $\mathcal{Q}$, the image $f_h(t)$ of t. $f_{h*}(\mathcal{I})$ is the Hilbert curve.

Again, as with the Peano curve, the Hilbert curve is continuous, but nowhere differentiable. Its construction is very similar to that of the geometric Peano curve, and is done here again by repeated application of linear operators. Figure 2 shows eight iterations of this process. These graphs, generated by the code in Appendix A, demonstrate how the positions of the points on the original interval are transformed to cover the unit square. This graphical overview again allows us to see how the self-similarity of the curve is retained as its complexity grows.

Figure 2: Hilbert's space-filling curve as generated by my code, found in Appendix A. These depict a mapping from the unit interval to the unit square, where the color of the line describes the position on the original interval as seen in the legend. The labels n = 1 to n = 8 describe the number of iterations of the iterated function system generating the curve.



7 Sierpiński's Space-Filling Curve

7.1 Wacław Sierpiński

Wacław Sierpiński (1882-1969) was a Polish mathematician who worked at the University of Lemberg and the University of Warsaw. He further worked to found the Polish School of Mathematics and the important mathematical journal, Fundamenta Mathematicae. He also revived and reinvigorated the Acta Arithmetica and the Mathematical Institute of the Polish Academy of Science, which desperately needed to be reorganized after the occupation of Poland during World War Two, when all universities, colleges, and other institutes of higher learning were shut down. Sierpiński mainly worked in the fields of set theory, topology, and number theory. He was an incredibly prolific writer, publishing 724 papers and 50 books in total over the course of his career, and is remembered, alongside a named street, prize, and auditorium, with a crater on the moon which bears his last name. He also received numerous honorary degrees and was elected to a large number of influential societies over the course of his lifetime.[11]

7.2 Sierpiński's Curve

Sierpiński showed that there is a bounded and continuous function f of a real variable t such that

$$f(t) + f(t + 1/2) = 0 \text{ for all real } t \tag{3}$$

and

$$2f(t/4) + f(t + 1/8) = 1 \text{ for all } t \in [0, 1] \tag{4}$$

and that

$$(x, y) = (f(t), f(t - 1/4)) \text{ for all } t \in [0, 1] \tag{5}$$

passes through every point of the square $[-1, 1]^2$. In the course of an intensive proof that f has all required properties, he found the following representation:

$$f(t) = \frac{\Theta(t)}{2} - \frac{\Theta(t)\Theta(\tau_1(t))}{4} + \frac{\Theta(t)\Theta(\tau_1(t))\Theta(\tau_2(t))}{8} - \dots \tag{6}$$

Where Θ and τ_k are both 1-periodic functions defined as:

$$\Theta(t) = \left\{ \begin{array}{lll} -1 & \text{if} & t \in [1/4, 3/4) \\ 1 & \text{if} & t \in [0, 1/4) \text{ or } t \in [3/4, 1) \end{array} \right\} \tag{7}$$

and

$$\tau_1(t) = \left\{ \begin{array}{lll} 1/8 + 4t & \text{if} & t \in [0, 1/4) \text{ or } t \in [1/2, 3/4) \\ 1/8 - 4t & \text{if} & t \in [1/4, 1/2) \text{ or } t \in [3/4, 1) \end{array} \right\} \tag{8}$$

where

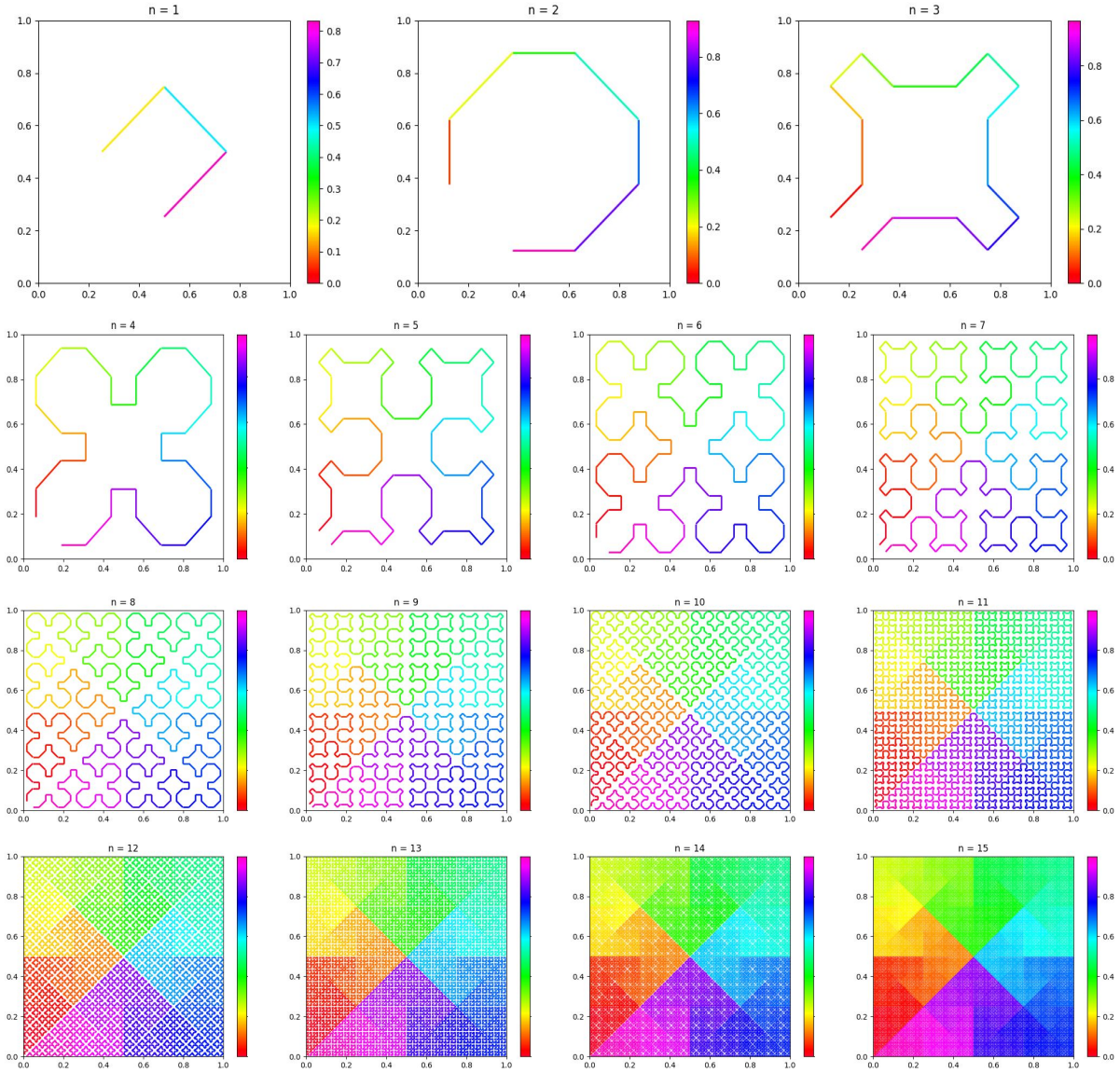
$$\tau_k(t) = \tau_1^k(t) \tag{9}$$

Sierpiński demonstrated that the curve described by the formula above is the limit of a uniformly convergent series of polygons, fifteen of which are depicted in Figure 3. We can see that the self-similarity of the curve is retained through ever more complex iterations.

Surprisingly, the Sierpiński curve can be generated in much the same way as the previous two curves by way of repeated linear operators. It does, however, require a good deal more work and consideration than Peano and Hilbert's curves required in order to do so.

The Sierpiński polygon has a number of useful symmetries, notably it is symmetric across the main $x = y$ diagonal. Subdividing the shape and looking at only the top section one can see that each iteration is similar to two copies of the previous, under a one eighth rotation in each direction connected by the endpoints. Once again, this then allows the use of linear operators to do the heavy lifting of the iterated function system, as is shown in Figure 3.

Figure 3: Sierpiński's space-filling curve as generated by my code, found in Appendix A. These depict a mapping from the unit interval to the unit square, where the color of the line describes the position on the original interval as seen in the legend. The labels n = 1 to n = 15 describe the number of iterations of the iterated function system generating the curve.



8 Lebesgue's Space-Filling Curve

8.1 Henri Lebesgue

Henri Leon Lebesgue (1875-1941) was a French mathematician who taught at the Collége de France and the Academy of Sciences in Paris. He is credited with 165 different papers and books. Lebesgue is particularly well known for his theory of measure and his introduction of the Lebesgue integral, greatly expanding the possibilities of Fourier analysis. He also made contributions to the fields of topology, potential theory, the Dirichlet problem, the calculus of variations, set theory, the theory of surface area and dimension theory. Lebesgue received a number of accolades, honorary degrees, prizes, and academy elections over the course of his lifetime.[12]

8.2 The Cantor function

Consider the following mapping of the Cantor set onto the plane:

$$f(0_3(2t_1)(2t_2)(2t_3)\dots) = \begin{pmatrix} 0_2 t_1 t_3 t_5 \dots \\ 0_2 t_2 t_4 t_6 \dots \end{pmatrix} \text{ for } t_i \in 0, 1 \tag{10}$$

Consider then the interpolation between these points

$$f_l(t) = \frac{1}{b_n - a_n}[f(a_n)(b_n - t) + f(b_n)(t - a_n)], a_n \leq t \leq b_n \tag{11}$$

This constructs the Cantor function, also known as the Devil's Staircase. This function is continuous, with derivative zero at almost all points, specifically, with nonzero derivative on a set of measure zero. Yet, it still grows from zero to one as its range progresses from zero to one. Lebesgue's space-filling curve uses a similar construction in order to create a space-filling curve that is also differentiable almost everywhere.

8.3 Lebesgue's Curve

Lebesgue extended the aforementioned linear mapping

$$f(0_3(2t_1)(2t_2)(2t_3)\dots) = \begin{pmatrix} 0_2 t_1 t_3 t_5 \dots \\ 0_2 t_2 t_4 t_6 \dots \end{pmatrix} \tag{12}$$

continuously into the unit square by linear interpolation between set points.

Allow (a_n, b_n) to be an interval removed by the construction of the middle third Cantor set Γ at the nth step. Then we may construct f_l, Levesgue's space-filling curve, in terms of the above f as:

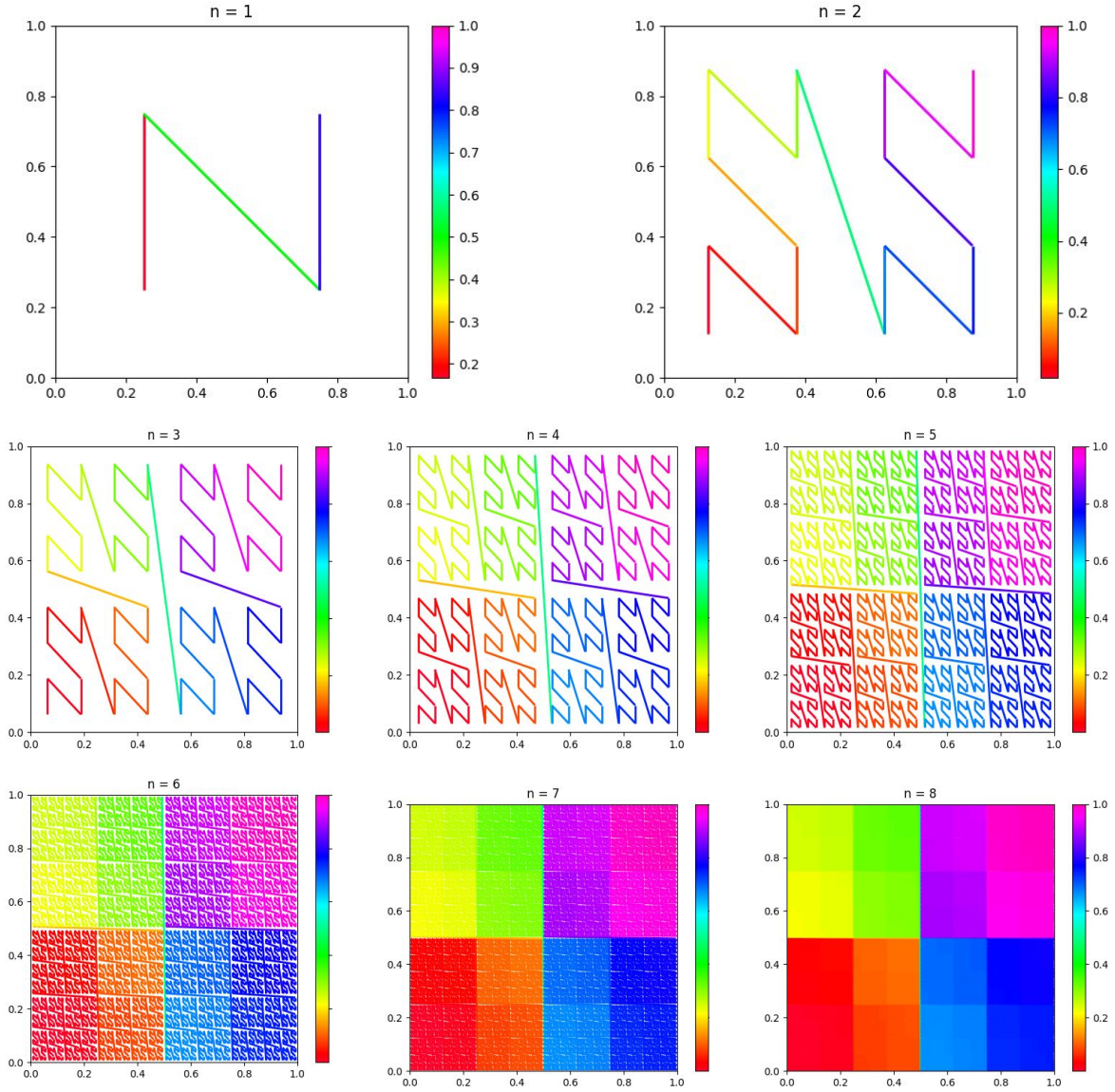
$$f_l(t) = \frac{1}{b_n - a_n}[f(a_n)(b_n - t) + f(b_n)(t - a_n)], a_n \leq t \leq b_n \tag{13}$$

By construction, f_l is continuous on Γ^c, the complement of the Cantor set, and maps the unit interval onto the unit square.

Note that Lebesgue's space-filling curve is differentiable almost everywhere: it is differentiable on Γ^c and not differentiable on Γ. Unlike the other space-filling curves discussed so far, Lebesgue's space-filling curve does not have the property that each part of itself is again a space-filling curve. Due to its careful parameterization on the Cantor set, the derivative of the curve is defined on almost all of the unit interval, so all of these sections must map onto finite lengths of curve in the image. For example, the middle third of the unit interval maps directly onto the vertical line through the middle of the unit square. This can be seen in the figures below by how the colours shift more dramatically across these sections.

Generating Lebesgue's space-filling curve proved to be somewhat complicated. The first step of the process was actually to generate an appropriately sampled Cantor set by which to index the curve. After that, linear operators again take the stage to produce the regions of self-similarity, but the interpolator and line shader do the work of fitting the sections together appropriately. Figure 4 contains the first eight iterations, generated by code found in Appendix A.

Figure 4: Lebesgue's space-filling curve as generated by my code, found in Appendix A. These depict a mapping from the unit interval to the unit square, where the color of the line describes the position on the original interval as seen in the legend. The labels n = 1 to n = 8 describe the number of iterations of the iterated function system generating the curve.



9 Schoenberg's Space-Filling Curve

9.1 Isaac Schoenberg

Isaac J. Schoenberg (1903-1990) was a Romanian mathematician who emigrated to the United States of America under a Rockefeller scholarship in 1930 after completing his studies. He worked at the Institute for Advanced Studies at Princeton University, Swarthmore College, Colby College, the University of Pennsylvania, and the University of Wisconsin. This illustrious list does not even encompass his work with the *Annals of Mathematics*, the Aberdeen Ballistics Research Laboratory and the Madison Army Research Center. He worked on Hilbert spaces but is especially well known for his work on the theory of splines and their use in a variety of areas.[13]

9.2 Schoenberg's Curve

Beginning again with the continuous mapping from the Cantor set Γ onto the unit square, Schoenberg extended this mapping into the interval by way of a simple projection function onto a trapezoidal wave function. By using only the composition of very simple continuous functions, Schoenberg was able to create a very compelling and

somewhat well-behaved space-filling curve.

Schoenberg defined ϕ_{sc}, ψ_{sc} as components of the map $f_{sc}\colon \mathcal{I} \longrightarrow \mathcal{Q}$ using the following trapezoidal wave:

$$p(t) = \left\{ \begin{array}{lll} 0 & \text{for} & 0 \leq t \leq \frac{1}{3} \\ 3t-1 & \text{for} & \frac{1}{3} \leq t \leq \frac{2}{3} \\ 1 & \text{for} & \frac{2}{3} \leq t \leq 1 \end{array} \right\}, p(-t) = p(t), p(t+2) = p(t) \tag{14}$$

This lets us define ϕ_{sc} and ψ_{sc} in terms of p:

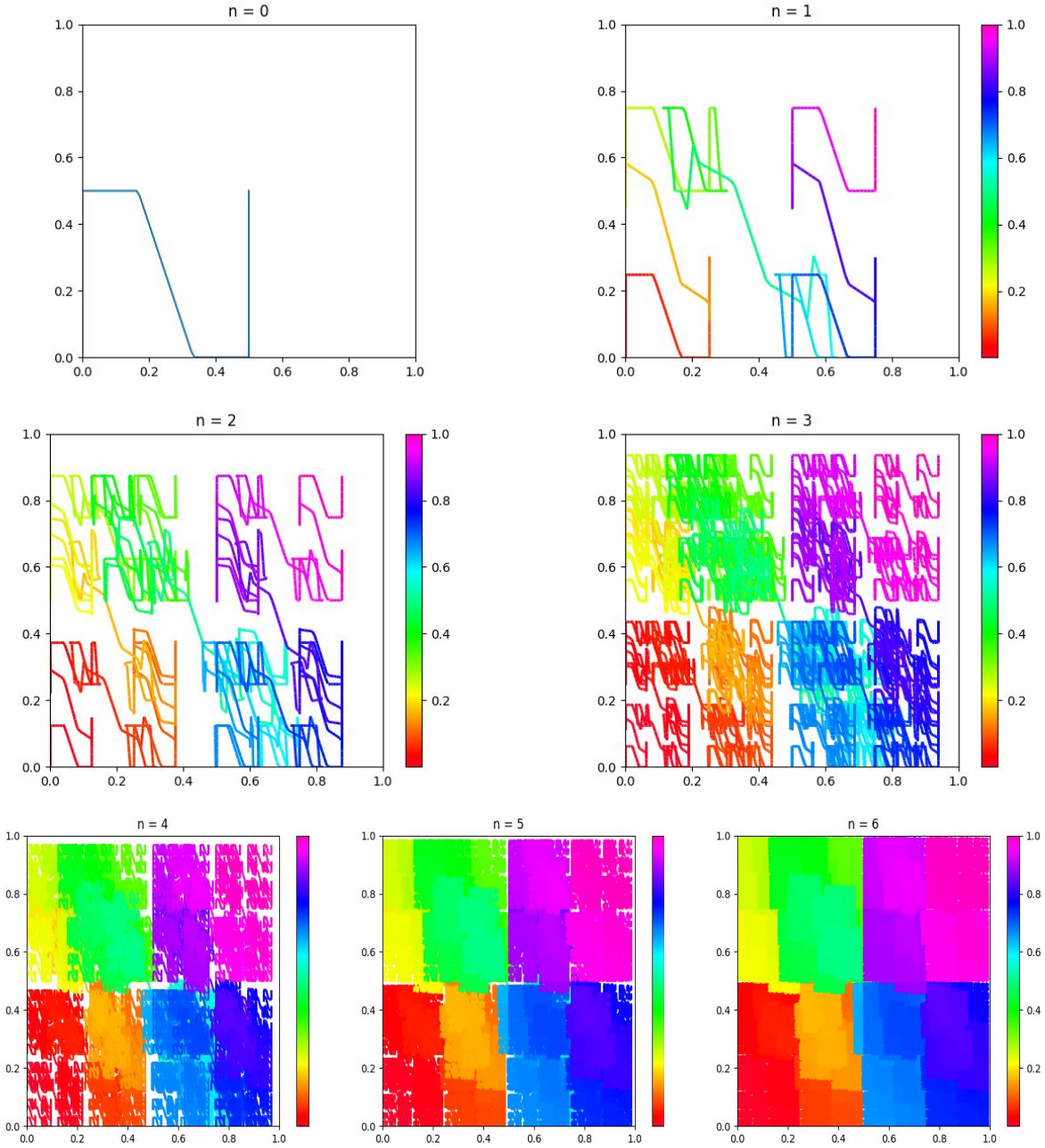
$$\phi_{sc}(t) = \frac{1}{2} \sum_{k=0}^{\infty} \frac{p(3^{2k}t)}{2^k}, \psi_{sc}(t) = \frac{1}{2} \sum_{k=0}^{\infty} \frac{p(3^{2k+1}t)}{2^k} \tag{15}$$

The resulting function is continuous as it is simply the composition of many simple continuous functions, and the geometric series $\sum_{n=0}^{\infty} \frac{1}{2}^n$ dominates the sum, allowing it to easily remain uniformly convergent. Since a uniformly convergent series of continuous functions is a continuous function, ϕ_{sc} and ψ_{sc} are indeed continuous.

What is less clear in Schoenberg's construction than in other space-filling curves is whether the mapping fully covers the unit square. This, however, can be resolved in an interesting way. The restriction of the Schoenberg curve to the Cantor set Γ is actually identical to the mapping of the nodes of Lebesgue's space-filling curve, that is, the two curves agree entirely on the Cantor set, and as such Schoenberg's curve must certainly cover the unit square.

Plotting the Schoenberg curve proved to be a different type of challenge. Due to the (relative) simplicity of the curve's expression, it is a relatively simple task to plot the value of some point in the interval, but what proved difficult was choosing these points in the interval such that the very intricate and complex character of the curve is captured without requiring excessive processing time. This challenge had been avoided in the previous curves, as the computations necessarily found the corners of the approximating polygons, but calculating these same points for the Schoenberg curve was a non-trivial task. Figure 5 shows seven iterations of the curve calculated by code from Appendix A.

Figure 5: Schoenberg's space-filling curve as generated by my code, found in Appendix A. These depict a mapping from the unit interval to the unit square, where the color of the line describes the position on the original interval as seen in the legend. The labels n = 0 to n = 6 describe the number of iterations of the iterated function system generating the curve.



10 Plotting Space-Filling Curves

Computers are miraculous machines that within the past few decades have enabled a huge breadth of new exploration and understanding in the field of mathematics. I decided early on in this project that a computational element would be central to my work.

10.1 Approximating Polygons and Iterated Function Systems

Plotting a space-filling curve is a somewhat contradictory pursuit. Any technically correct plotting would simply show a fully filled in square with no curve-like character whatsoever. Indeed, many articles about space-filling curves make this precise joke to introduce the concept. Thus the primary method of recreating this aspect of character is through the use of approximating polygons. In addition, I thought that it would be useful to have the distance along a curve indicated by way of a colour gradient, which helps to keep the curve legible even at higher iterations.

The method of plotting the approximating polygon of a space-filling curve that I struck upon is very similar to how many self-similar fractals are generated. At each step, the figure would be extended by linearly transformed copies of itself and re-scaled and transformed into a self-similar copy. This could then be repeated for a finite number of steps, and, with careful accounting, the correct parameterization could be maintained in order to accurately describe the curve as a parameterized path through space.

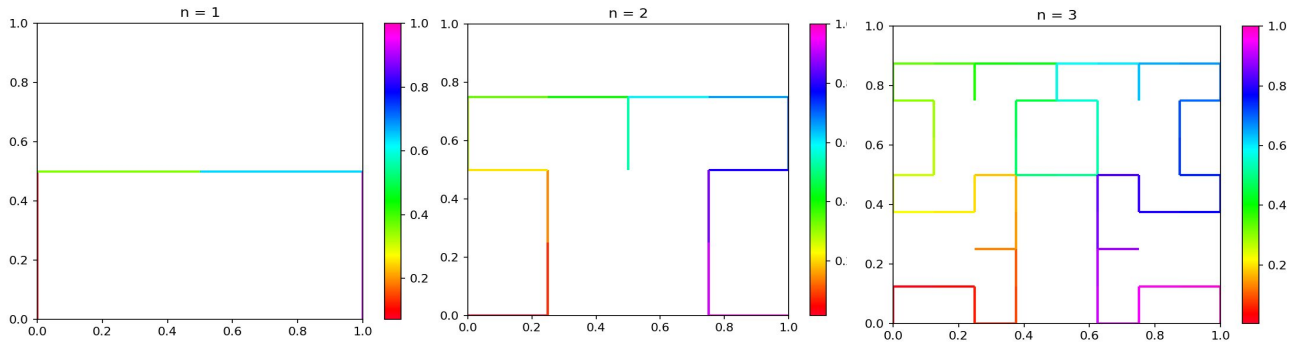
This, however, gave rise to a very interesting situation. As a recursive process, this method required a sort of base case in order for calculation to actually begin. As the most obvious solution, you could consider the base case to be the filled in unit square. This fails for two reasons. Firstly, it fails to give the curve any kind of character, as mapping whole regions onto whole regions with no empty space simply gives rise to the filled in square. Secondly, it pushes the entire question posed by a space-filling curve into the base case, as the base case would require a base parameterization, which is simply circular. This question of a base case required more investigation.

10.2 Leitmotifs

The leitmotif of an approximation to a space-filling curve is the name for exactly this base case. Most notably, by construction as the attractor of an iterated function system, the limit of a space-filling curve is the same, regardless of what leitmotif is used to seed it. This is intuitively reasonable, as the self-similar subsections of the curve can be thought of as shrinking to a point, and as such what exactly is in each of those shrinking sections ceases to have any impact on the shape of the curve. The typical leitmotif used in almost all depictions of approximating polygons is simply the interpolated line connecting the centre points of two connected subsections. This is what was used in all examples above, but other leitmotifs are possible. I have written my code in such a way that swapping these generating sets is simple, and I encourage the reader to test this out for themselves.

Consider, for example, the leitmotif of a Hilbert curve as a straight line interpolation between (0,0) and (0,1). This gives rise to the phenomenon shown in Figure 6.

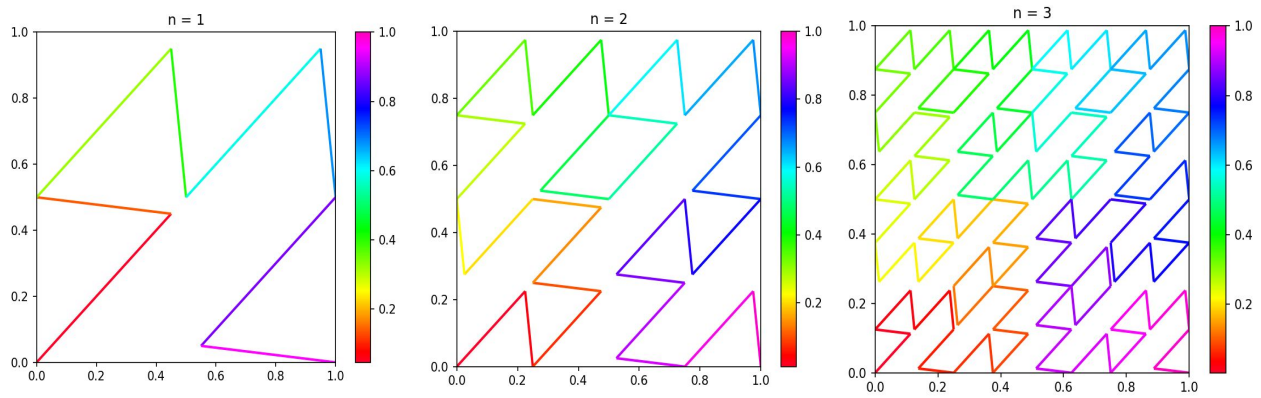
Figure 6: Three iterations of a construction of Hilbert's curve using a bottom line leitmotif as generated by my code, found in Appendix A. These depict a mapping from the unit interval to the unit square, where the color of the line describes the position on the original interval as seen in the legend. The labels n = 1 to n = 3 describe the number of iterations of the iterated function system generating the curve.



This bottom line leitmotif curve notably has many self-intersections. This is important, as the typically shown centre point to centre point interpolations appear surjective, suggesting an exception to Netto's theorem. Curves of this type, specifically curves generated by leitmotifs which touch the corners of the square, are not surjective even in the finite case, providing a soothing counter-counterexample.

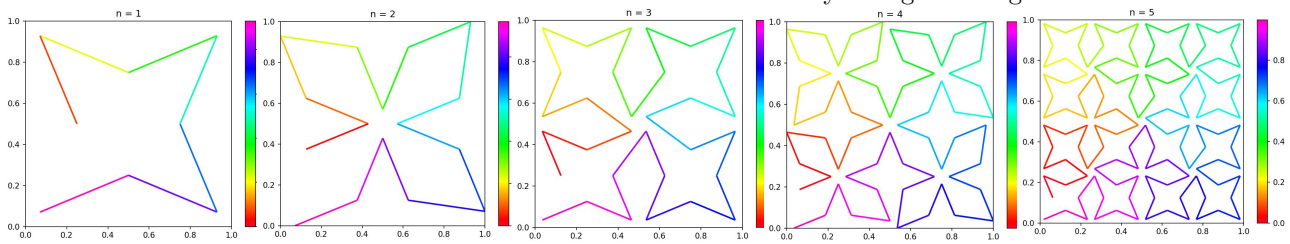
Another interesting leitmotif on the Hilbert curve is the main diagonal line from (0,0) to (0.9,0.9) and back to (1,0), which meets up with the start of the next square, as depicted in Figure 7.

Figure 7: Three iterations of a construction of Hilbert's curve using a top line triangle leitmotif as generated by my code, found in Appendix A. These depict a mapping from the unit interval to the unit square, where the color of the line describes the position on the original interval as seen in the legend. The labels n = 1 to n = 3 describe the number of iterations of the iterated function system generating the curve.



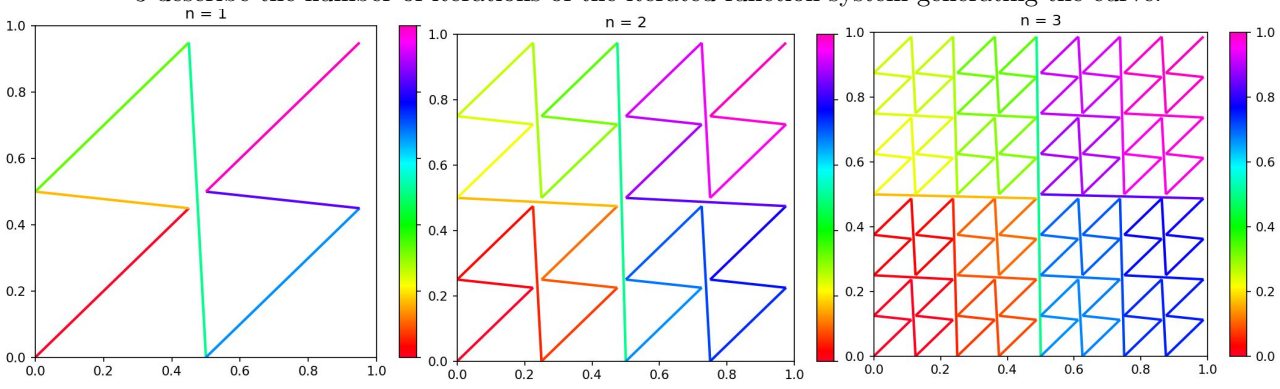
The same can be done with other similarly leitmotif-seeded curves, for example, shown are some approximating polygons to the Sierpiński curve, using as a seed the line from (0.1,0.1) to the normally used centre of the triangle at $(\sqrt{2}/2, \sqrt{2}/4)$. This gives rise to an interesting tilted snowflake-like pattern, as shown in Figure 8.

Figure 8: Five iterations of a construction of Sierpiński's curve using a different line leitmotif as generated by my code, found in Appendix A. These depict a mapping from the unit interval to the unit square, where the color of the line describes the position on the original interval as seen in the legend. The labels n = 1 to n = 5 describe the number of iterations of the iterated function system generating the curve.



One more example is that of Lebesgue's space-filling curve, which is sometimes approximated by the half open main diagonal line instead of connecting centre points like the others. In Figure 9 I have approximated this, reducing the diagonal to the line from (0,0) to (0.9,0.9) for a clearer image without self-intersections. This depiction also helps to better display the uneven colour gradient, as Lebesgue's curve is not traversed at "constant speed," unlike the previous curves mentioned, instead having subsquares indexed by the Cantor set.

Figure 9: Three iterations of a construction of Lebesgue's curve using a main diagonal leitmotif as generated by my code, found in Appendix A. These depict a mapping from the unit interval to the unit square, where the color of the line describes the position on the original interval as seen in the legend. The labels n = 1 to n = 3 describe the number of iterations of the iterated function system generating the curve.



This understanding of the conveniences and resultant limitations imposed by these approximating polygons is very important to the nature of space-filling curves. All leitmotifs operated on by a particular curve's iterated function system generate the very same space-filling curve. In this way, these approximating polygons are very specifically not themselves space-filling curves, only finite descriptions of a kind. This, of course, is sensible and easy to accept, as on any finite approximation, there will still remain literal uncovered points.

11 Conclusion

A distinction to consider after reading this project is that it is important when studying to separate the reality of the space-filling curve as the limit of a sequence of functions from the tangible and plottable polygonal approximations which are the central method of communicating structure and character in this field of study. The later iterations, in which the low level character of the curves becomes unimportant, but where the shifting color of the square denotes the general effect of the curve, may in fact be a more useful insight into these functions. It is interesting, for example, how Lebesgue's space-filling curve spends nearly all of its time on a small subset of the plane, or how Schoenberg's curve's lack of self similarity causes the colours to overlap and blur together.

Space-filling curves also do have applications to other fields, particularly in data science and organization in computer science. They are often used to find or order points within a data space, for instance to organize hues within a commercial color dictionary.[14, 15]

In a subsequent project, I would hope to further explore the relationship between topological ideas of dimension and the construction of space-filling curves, specifically in topological spaces that are not homeomorphic to $\mathbb{R}^n$ as I think that being able to generalize the theory on space-filling curves would be extremely interesting in that direction, and could have some interesting applications to mappings between more exotic spaces. Another direction that I think there is more value in pursuing is in the more general theory of generating space-filling curves based on arbitrary stochastic functions, as I think an even more general and less strictly structured approach than the perscriptivism of codified and named curves could be very interesting.

References

[1] C. Tricot, *Curves and Fractal Dimension*. Springer-Verlag, 1995.

[2] J. W. Dauben, *Georg Cantor: His Mathematics and Philosiphy of the Infinite*. Princeton University Press, 1990.

[3] H. Sagan, *Space-Filling Curves*. Springer-Verlag, 1994.

[4] J. J. O'Connor and E. F. Robertson, "Georg Ferdinand Ludwig Philipp Cantor." https://mathshistory.st-andrews.ac.uk/Biographies/Cantor/, 1998.

[5] E. Pearse, "An introduction to dimension theory and fractal geometry: Fractal dimensions and measures," *Cornell*, 2005.

[6] M. G. Charalambous, *Dimension Theory: A Selection of Theorems and Counterexamples*. Springer Cham, 2019.

[7] J. J. O'Connor and E. F. Robertson, "Eugen Otto Erwin Johannes Netto." https://mathshistory.st-andrews.ac.uk/Biographies/Netto/, 2017.

[8] J. J. O'Connor and E. F. Robertson, "Giuseppe Peano." https://mathshistory.st-andrews.ac.uk/Biographies/Peano/, 1997.

[9] P. D. Humke and K. V. Huynh, "Finding keys to the peano curve," *Acta Mathematica Hungarica*, vol. 167, pp. 255–277, 2022.

[10] J. J. O'Connor and E. F. Robertson, "David Hilbert." https://mathshistory.st-andrews.ac.uk/Biographies/Hilbert/, 2014.

[11] J. J. O'Connor and E. F. Robertson, "Wacław Sierpiński." https://mathshistory.st-andrews.ac.uk/Biographies/Sierpinski/, 1997.

[12] J. J. O'Connor and E. F. Robertson, "Henri Lebesgue." https://mathshistory.st-andrews.ac.uk/Biographies/Lebesgue/, 2004.

[13] J. J. O'Connor and E. F. Robertson, "Isaac Jacob Schoenberg." https://mathshistory.st-andrews.ac.uk/Biographies/Schoenberg/, 2016.

[14] A. Jaffer, "Color-space dimension reduction." https://people.csail.mit.edu/jaffer/Color/CSDR, 2005.

[15] B. V. D. Broucke, "Space-filling curves." https://bertvandenbroucke.netlify.app/2019/01/18/space-filling-curves/, 2019.

A Code

What follows is a color-enhanced transcription of the code that I wrote in my investigation of this subject. As mentioned, this is also the code used to generate all of the plots shown throughout this project. It can also be found as a .py file attached to my submission under "Additional Materials."

```
import sys
import matplotlib.pyplot as plt
from matplotlib import colormaps
import numpy as np
from matplotlib.collections import LineCollection
from matplotlib.colors import BoundaryNorm, ListedColormap

##########################################################################################
##########################################################################################
############################    How To Use This Program    ###############################
##########################################################################################
##########################################################################################

# Run using python3. Running the file with no arguments will display a default curve.

# The first argument given must be one of { P, H, S, L, Sch } for Peano, Hilbert, Sierpinski,
# Lebesgue, and Schoenberg respectively. This defaults to the Hilbert curve if no arguments
    are
# given.

# The second argument given is the number of iterations of the curve to perform. This defaults
# to 4 if no value is given.

# The third argument is for turning off colouring. If you wish to have the curve without a
# gradient, write nocolor as the third argument. Colouring defaults to on.

# For example, in the correct directory, the command below:
# python SpaceFillingCurves.py P 2
# Would display the second iteration of a geomentric Peano curve

# Additionally, the base set for curves that use the IFS function can be changed below.
# Each row represents a point in the unit square that is then transformed under the IFS.
# As it is now, only the central point of the square is selected, so the curves will be
# the connections between subsquare centres. Note that any number of points may be used.
# Feel free to try different points!

startsett = [
            #[1   , 0.5],
            [0   , 0  ],
            #[0.1, 0.1],
            #[0.4, 0.6],
            #[0.6, 0.4],
            [0.9, 0.9],
            #[1   , 1  ],
            #[1   , 0  ]
]

# Finally, if you wish to save the generated plots, there is a commented out line at the end
    of  this
# file that can be uncommented in order to do so. Alternatively, write save as the third
    argument when
# calling the program.



##############################################
# input handling #############################
##############################################

if(len(sys.argv) == 1):
    curvetype = "H"
else:
    curvetype = sys.argv[1]

if(len(sys.argv) < 3):
    iterations = 3
else:
    iterations = int(sys.argv[2])

color = True
```

```
saveplot = False
if(len(sys.argv) > 3):
    if( sys.argv[3] == "nocolor" ):
        color = False
    if( sys.argv[3] == "save" ):
        saveplot = True

#reform the list of starting points into a pair of coordinate lists
#This makes coordinate transformations and plotting easier later
startset = np.array( [ [a[i] for a in startsett] for i in range(len(startsett[0])) ] )
print("using starting set " + str(startset))


#########################
######### Body #########
#########################

def IFS(i, curveIFS, workingset):
    #This function is the backbone of many of the others in this file
    #deceptively simple and surprisingly common
    #the humble while loop
    while i > 0:
        #print(pointset)
        workingset = curveIFS(workingset)
        i -= 1
    return workingset

def CantorIFS(F):
    #realy makes it look simple doesn't it
    return np.concatenate([F,F+2])/3

def GeometricPeanoIFS(F):
    return [np.concatenate([F[0],   #X
                            -F[0]+1,
                            F[0],
                            F[0]+1,
                            -F[0]+2,
                            F[0]+1,
                            F[0]+2,
                            -F[0]+3,
                            F[0]+2]
        )/3,
            np.concatenate([F[1],   #Y
                            F[1]+1,
                            F[1]+2,
                            -F[1]+3,
                            -F[1]+2,
                            -F[1]+1,
                            F[1],
                            F[1]+1,
                            F[1]+2]
        )/3]
def GeometricPeano(n):
    colors = np.linspace(0,1,int((9**n)*((len(startset[0])))))
    return IFS(n,GeometricPeanoIFS,startset) + [colors]

def HilbertIFS(F):
    return  [np.concatenate([F[1],      #X
                             F[0],
                             (F[0]+1),
                             (-F[1]+2)]
            )/2,
            np.concatenate([F[0],       #Y
                            (F[1]+1),
                            (F[1]+1),
                            -F[0]+1]
            )/2]
def Hilbert(n):
    colors = np.linspace(0,1,int((4**(n))*(len(startset[0]))))
    return IFS(n,HilbertIFS,startset) + [colors]

def SierpinskiIFS(F):
    return  [np.concatenate([
                             +F[0]+F[1],              #X
                             +F[0]-F[1]+np.sqrt(2)
                             ]
```

```
            )/2,
            np.concatenate([
                            +F[0]-F[1],              #Y
                            -F[0]-F[1]+np.sqrt(2)
                            ]
            )/2]
def Sierpinski(n):
    #This one needs one final correction to mirror the image and turn it upright into the unit
     square
    #since the actual IFS as writtenonly operates on a "unit triangle", that is the triangle
    with corners
    #at (0,0), (sqrt2/2,sqrt2/2), (sqrt(2), 0)
    #That also means that it needs a custom startset, although this one too can be changed,
    like the others

    startset = [np.array([
                           #0,
                           #0.1,
                           np.sqrt(2)/2]),
                np.array([
                           #0,
                           #0.1,
                           np.sqrt(2)/4])]

    F = IFS(n,SierpinskiIFS,startset)  # call the IFS

    F = [np.concatenate([
                         +F[0]+F[1],              #X
                         -F[0]-F[1]+np.sqrt(2)
                         ]
        )/np.sqrt(2),
        np.concatenate([
                         +F[0]-F[1],              #Y
                         -F[0]+F[1]+np.sqrt(2)
                             ]
        )/np.sqrt(2)]
    return F + [np.linspace(1,0,(2**(n+1))*len(startset[0]))]

def LebesgueIFS(F):
    return  [np.concatenate([F[0],       #X
                             F[0],
                             F[0]+1,
                             F[0]+1]
            )/2,
            np.concatenate([F[1],        #Y
                            F[1]+1,
                            F[1],
                            F[1]+1]
            )/2]
def Lebesgue(n):
    #very importabtly, the Lebesgue space-filling curve is indexed not by simple linear
    distribution,
    #but instead by a cantor set. A cantor set is constructed using the previously defined
    function and
    #used as the colour index.
    colors = IFS(2*n-1, CantorIFS, np.concatenate([np.zeros(len(startset[0])),
                                                   np.array([1]*len(startset[0]))]))
    return IFS(n,LebesgueIFS,startset) + [colors]

def SchoenbergPieces(x):
    #sawtooth mapping x -> [0,2]
    x = (x - ( 2*(np.floor(x/2))))

    if(x <= (1/3)):
        b = 0
    elif(x <= 2/3):
        b = (3*x) - 1
    elif(x <= 4/3):
        b = 1
    elif(x <= 5/3):
        b = (-3*x) + 5
    else:
        b = 0

    return b
def Schoenberg(n):
```

```
    #The Schoenberg curve is done substantially differently, not through the same IFS loop as the others.
    #It instead uses the formula below ( cited from Hans Sagan's Space-Filling Curves )
    pointcount = 8**(n+2) + 100
    DomainInterval = np.linspace(0, 1, pointcount)
    outx = np.zeros(pointcount)
    outy = np.zeros(pointcount)
    for i in range(pointcount):
        outx[i] = np.sum( [ SchoenbergPieces( DomainInterval[i] * (3**( 2*k ))   ) / (2**(k+1) )  for k in range(n+1) ] )
        outy[i] = np.sum( [ SchoenbergPieces( DomainInterval[i] * (3**((2*k)+1)) ) / (2**(k+1) )  for k in range(n+1) ] )
    return [outx,outy, DomainInterval]



#call the right function based on input
if(curvetype == "H"):
    pointset = Hilbert(iterations)
if(curvetype == "P"):
    pointset = GeometricPeano(iterations)
if(curvetype == "S"):
    pointset = Sierpinski(iterations)
if(curvetype == "L"):
    pointset = Lebesgue(iterations)
if(curvetype == "Sch"):
    pointset = Schoenberg(iterations)



##########################
###### Plotting #########
##########################

#set up the plot
fig, axs = plt.subplots()
axs.set_aspect('equal')
axs.set_xlim([0,1])
axs.set_ylim([0,1])
axs.set_title("n = "+str(iterations))
#note, this fixes the viewing area to the unit square



if(color and iterations != 0):
    # Create a set of line segments to color individually
    # This creates the points as an N x 1 x 2 array so that we can stack points together easily to get the segments.
    points = np.array([pointset[0], pointset[1]]).T.reshape(-1, 1, 2)
    segments = np.concatenate([points[:-1], points[1:]], axis=1)
    dydx = pointset[2]

    #the line's colour value is taken as the average of the value at the next and previous points
    #With more time and work I would put in better interpolation so that longer lines
    #( in lebesgue's and schoenberg's curves ) get additional points added for smoother colouring
    pointset[2] = (pointset[2] + np.concatenate([pointset[2][1:], [pointset[2][-1]]]))/2

    # Create a continuous norm to map from data points to colors
    # The norm class seemed a bit odd to me but it seemed like I can
    norm = plt.Normalize(pointset[2].min(), pointset[2].max())
    lc = LineCollection(segments, cmap='gist_rainbow', norm=norm)
    # Set the values used for colormapping
    lc.set_array(pointset[2])
    lc.set_linewidth(2)
    line = axs.add_collection(lc)
    fig.colorbar(line, ax=axs)

else:
    #if not, just plot with no colours
    axs.plot(pointset[0], pointset[1])



if saveplot:
```

```
285     plt.savefig("SFC_"+curvetype+"_"+str(iterations)+"_"+str(startsett)+".png")
286 
287 #Uncomment the line below to save the generated plots as png files.
288 #plt.savefig("SFC_"+curvetype+"_"+str(iterations)+"_"+str(startsett)+".png")
289 
290 #show plot:
291 plt.show()
```