The Loewner Equation and Weierstrass' Function

Gavin Glenn

SENIOR THESIS

in

Mathematics

Presented to the Faculties of the University of Tennessee in Partial Fulfillment of the Requirements for the Bachelor of Science in Mathematics with Honors Concentration

May 2017

Supervisor of Dissertation

_____

Joan Lind, Professor of Mathematics

Undergraduate Major Advisor

_____

Remus Nicoara, Professor of Mathematics

Acknowledgments

I would like to thank Dr. Joan Lind, for her excellent mentorship and constant positive attitude towards this project. She showed me how interesting the world of research can be and inspired me to continue studying pure math.

I would also like to thank David Horton for getting me up to speed on the project and for his help with generalizing the proofs of the bounds on the Weierstrass function.

The Loewner Equation and Weierstrass' Function

Gavin Glenn

Dr. Joan Lind, Advisor

INTRODUCTION

The Loewner Differential Equation provides a correspondence between continuous, real-valued functions and certain growing families of sets in the upper half of the complex plane. When the real-valued function is Brownian motion, this correspondence is known as Schramm-Loewner evolution, and has been studied extensively in probability theory. Weierstrass functions share many important properties with Brownian motion, and in [5] the authors study an analog of Schramm-Loewner evolution using a particular Weierstrass function. As an extension of the later work, this thesis explores the Loewner Equation numerically and the Weierstrass function analytically, with the aim of advancing the study of each.

A classical result which is foundational for our work is that is that any continuous function corresponds to a connected set in the upper half plane via the Loewner Equation. In [5], the authors state and prove results about the characterization of these sets. In particular, they relate a Hölder continuity modulus of the Weierstrass function to two different phases for connected the sets: simple curves and non-simple curves. Numerical simulations suggest the existence of a third phase,

that of space-filling curves, and the work of this thesis contains some initial steps toward studying that question.

The thesis is roughly divided as follows. In chapter one, we introduce the Loewner equation and describe the correspondence it provides between continuous functions and curves in the upper half plane. We also state some relevant theorems. In chapter 2, we derive algorithms for simulating the curves given by the correspondence. We also demonstrate results from the author's own simulation program ([1]). Finally, in chapter 3, we take a closer look at Weierstrass functions, and establish bounds on them, as generalizations of the work in [5].

Contents

Chapter 1

The Loewner Equation

1.1 Definitions and Examples

We begin by introducing the Loewner differential equation and some basic definitions that will be used throughout.

Definition 1.1.1. Let λ be a continuous mapping from the time interval $[0, T]$ into $\mathbb{R}$ and $z_0 \in \overline{\mathbb{H}} \setminus \{\lambda(0)\}$, where $\mathbb{H} = \{x + iy \mid y > 0\}$ is the upper half of the complex plane. The form of the Loewner Differential Equation we consider is defined to be the initial value problem

$$\frac{d}{dt} z(t) = \frac{2}{z(t) - \lambda(t)} \tag{1.1.1}$$

with initial condition $z(0) = z_0$. The existence and uniqueness theorems from elementary differential equations guarantee that a unique solution $z(t)$ to (1.1.1) will exist so long as the denominator does not vanish. This motivates another definition.

Definition 1.1.2. Let $K_t := \left\{ z_0 \in \overline{\mathbb{H}} \mid z(s) = \lambda(s) \text{ for some } s \in [0, t] \right\}$. That is, K_t is the set of all initial values that eventually lead to a singularity in (1.1.1). Observe that if $t \leq u$ then $K_t \subseteq K_u$. We call this growing set K_t the hull at time t.

The hull may be thought of in the following manner. The curve $z\,(t)$, which depends on $\lambda\,(t)$, begins at the initial value z_0 and extends, for some time, throughout the upper half plane. The function $\lambda\,(t)$ on the other hand, always remains on the real axis. If $z_0 \in \mathbb{H}$ then $z\,(t)$ will remain in $\mathbb{H}$ unless it becomes equal to $\lambda\,(t)$. This is not obvious; it follows from a perspective of conformal maps, which we will introduce later. If instead $z_0 \in \mathbb{R}$ then (1.1.1) implies that it will remain in $\mathbb{R}$, and that it will move away from $\lambda\,(t)$. If $\lambda\,(t)$ then moves quickly enough, it may actually catch up to $z\,(t)$. In either case, whenever $z\,(t)$ and $\lambda\,(t)$ coincide, z_0 is added to the hull. Because of this relationship, we will refer to $\lambda\,(t)$ as the driving function, since it drives $z(t)$ throughout the upper half plane and possibly across the real axis. If $\lambda\,(t)$ does catch up to $z\,(t)$ at some time s, then we say the initial

value z_0 is captured at time s.

A simple example is nonetheless instructive.

Example 1.1.3. Suppose $\lambda(t)$ is identically 0. Writing $z(t) = x(t) + iy(t)$, equation (1.1.1) becomes

$$x'(t) + iy'(t) = \frac{2}{x(t) + iy(t)}$$

Upon which,

$$x'(t) + iy'(t) = \frac{2(x(t) - iy(t))}{(x(t))^2 + (y(t))^2}.$$

So

$$\operatorname{Re} z'(t) = \frac{2 \operatorname{Re} z(t)}{|z(t)|^2} \tag{1.1.2}$$

$$\operatorname{Im} z'(t) = \frac{-2 \operatorname{Im} z(t)}{|z(t)|^2}. \tag{1.1.3}$$

Thus initial values $z_0 \in \mathbb{H} \setminus \{\lambda(0)\}$ move strictly down. They move strictly to the left if their real part is negative and strictly to the right if their real part is positive. If their real part is 0 then they move precisely down the imaginary axis, toward the origin where $\lambda(t)$ sits. In this case (1.1.3) becomes separable, and we can solve it on $[0, t]$ to obtain $y(t) = \sqrt{y^2(0) - 4t}$. Thus if $y(s) = \lambda(s) = 0$ for some $s \in [0, t]$ then $z(0) = y(0) = 2\sqrt{s}$. Hence $K_t = \left\{iy \mid 0 \leq y \leq 2\sqrt{t}\right\}$ is a vertical line segment starting at the origin with length $2\sqrt{t}$. An image of the hull created by our program appears in figure 1.1 below.

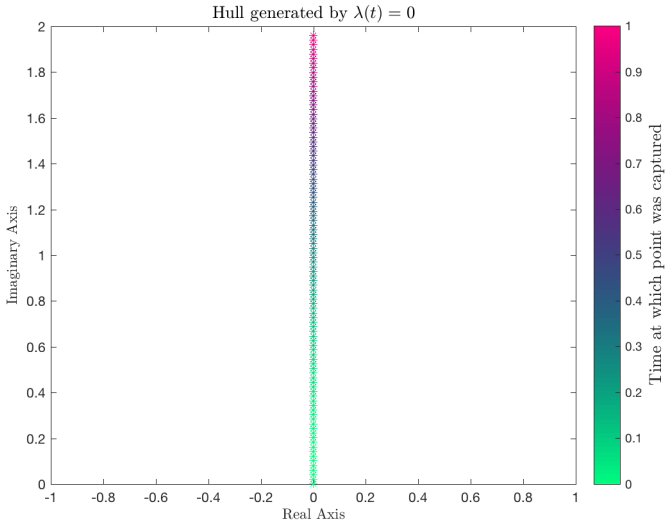


Figure 1.1: Example 1 Hull

1.2 Hull Behavior

The hull is precisely the set we referred to in the introduction, which corresponds to the real valued $\lambda(t)$ via the Loewner equation. In the first section we thought of the hull as a single set which grows in time. Another useful way of conceptualizing the situation is to consider the family of hulls $\{K_t\}_{t\in[0,T]}$ for each fixed t. For t fixed, we can let z vary and define

$$g_t(z) := z_t, \text{ where } z_0 = z.$$

In the example from section 1.1, the hulls K_t were all simple curves. The hulls may be non-simple (curves that hit back once on themselves) or even space filling for certain driving functions. The following is true of the hulls in general.

Theorem 1.2.1. *Let λ be a continuous on an interval $[0,T]$ and fix $t \in [0,T]$. Then the hull K_t generated by $\lambda(t)$ is connected. Furthermore, $g_t(z)$ is a conformal mapping from $\mathbb{H} \setminus K_t$ onto $\mathbb{H}$.*

The term "conformal mapping" will be defined in the next section. At present the important takeaway from this theorem is that, for each fixed time t, the solution g_t to the Loewner equation maps the set of all points which have not been captured onto the upper half plane in a way that preserves the geometry. In particular, boundaries map to boundaries.

This result [1] motivates much of our study of the hull. In some ways, the hull is being encoded (via g_t) into the real-valued function $\lambda(t)$. The interest is in studying how this encoding works. Can we detect properties of the hull from properties of $\lambda(t)$? The current answer to this question is "yes, sometimes." We need some preliminary definitions.

Definition 1.2.2. Let f be a real valued function defined on some set $X \subseteq \mathbb{R}$. If there are real numbers C and $\alpha > 0$ such that

$$|f(x) - f(y)| \leq C\,|x-y|^{\alpha}$$

for all $x, y \in X$, then we say f is Hölder continuous with exponent α on X.

We will be interested in a special case of Hölder continuity with $\alpha = 1/2$. Suppose $f : X \to \mathbb{R}$ is Hölder continuous with $\alpha = 1/2$, and let M be the smallest number such that $|f(x) - f(y)| \leq M\,|x-y|^{\frac{1}{2}}$. This induces a norm on the vector space of all functions on X which are Hölder continuous with $\alpha = 1/2$. We will denote

[1] For a proof of this theorem see G. Lawler's "Conformally Invariant Processes in the Plane", Sec. 4.1

this space as $H^{1/2}(X)$ and the norm as $\|f\|_{1/2}$. We are ready to state some hull characterization theorems.

Theorem 1.2.3. *([5]) If $\lambda \in H^{1/2}([0,T])$ with $\|\lambda\|_{1/2} < 4$ then the hulls generated by λ are all simple curves contained $\mathbb{H} \cup \{\lambda(0)\}$.*

Note that in general the converse is not true. A counter example given in [3] is $\lambda(t) = c\sqrt{t}$, for $c \geq 4$, which does generate simple curves.

Another theorem relates a similar criterion to space-filling hulls.

Theorem 1.2.4. *([6]) There is some constant $c > 4$ such that if $\|\lambda\|_{1/2} < c$ then the hulls generated by λ are not space-filling.*

These results prompt us to turn to numerical simulation of the hull. We know that phase transitions in the hulls do exist, but we do not know exactly where they are. Ideally, numerical approximations will give us a good idea.

Chapter 2

Simulating Loewner Flow

Theoretically, it is easy to describe the criterion for a point z_0 to be captured by a given driving function λ. That is, z_0 is captured precisely when $z(t) = \lambda(t)$ in the Loewner equation (1.1.1). When λ is a constant, as in example 1.1.3, the Loewner equation becomes separable, and an analytic solution is easily derived. However for an arbitrary driving function, including most of the ones which are of interest to us, a general solution to the Loewner equation is not available. For this reason, we turn to numerical simulation. But immediately we face several obstacles. For one, we must find a reliable way to simulate the Loewner flow that does not break down once points are captured, by avoiding divisions by extremely small quantities.

2.1 The Forward Euler Approximation

In the world of numerical analysis, the Loewner equation is not actually all that complicated. There exist several numerical methods we could employ to obtain an approximate solution. But for reasons that will soon become clear, we will use perhaps the most basic one: the Forward Euler method. We derive it as follows.

Let t_k denote the kth time step of the approximation. Assume the change in time at each step is constant and denote this change by Δt. The equation of the line $\phi(t)$ tangent to $z(t)$ at the point $(t_k, z(t_k))$ is given by

$$\phi(t) - z(t_k) = z'(t_k)(t - t_k) = z'(t)\Delta t.$$

But $z'(t_k)$ is given by the Loewner equation. Set $z(t_{k+1}) = \phi(t)$ and make these substitutions to obtain

$$z(t_{k+1}) = z(t_k) + \frac{2\Delta t}{z(t_k) - \lambda(t_k)}.$$

This yields the Forward Euler method, and is our approximation formula for $z(t)$. This is known as a first order method, meaning that the total error at any given iteration is approximately proportional to Δt. This is not ideal, but it is necessary for our approach. So why *do* we choose to approximate the Loewner flow in such a simple way? To answer this question we need to review some basic theory of complex functions.

Definition 2.1.1. Let A be an open subset of $\mathbb{C}$. A function $f : A \to \mathbb{C}$ is said to be conformal if it is analytic and injective on all of A.

If a mapping is conformal then it can be thought of as a function which preserves the "essence" of shapes in its domain, even though it may rotate or stretch them. A useful property of conformal maps is given by the next lemma.

Lemma 2.1.2. *The composition of two conformal maps defined on appropriate domains is again conformal.*

Proof Suppose $f : U \to W$ and $g : W \to Y$ are conformal maps. Thus f and g are analytic and injective. The composition of two injective maps is injective. By the chain rule $g \circ f$ is analytic with $(g \circ f)'(z) = g'(f(z))f'(z)$, which completes the proof. □

Now we are ready to demonstrate the utility of our Loewner approximation. Let t be fixed and set $c := \lambda(t)$. Recall the shorthand for an open ball with radius r and center z_0, $B(r, z_0) = \{z \in \mathbb{C} : |z - z_0| < r\}$.

Theorem 2.1.3. *The Forward Euler approximation given previously is a conformal map from $\mathbb{H} \setminus B(\sqrt{2\Delta t}, c)$ to $\mathbb{H}$. It is continuous on the closure of the domain, and it maps points to the real line if, and only if, they lie on the boundary of the domain.*

Proof
We will recover our approximation as a composition of conformal maps that are easier to work with individually, since their composition will again be conformal (lemma 2.1.2). Set

$f_1(z) := z\sqrt{\frac{1}{2\Delta t}}$
$f_2(z) := z - c\sqrt{\frac{1}{2\Delta t}}$
$f_3(z) := z + \frac{1}{z}$
$f_4(z) = z + c\sqrt{\frac{1}{2\Delta t}}$ and
$f_5(z) := z\sqrt{2\Delta t}$.

For now we leave the domain and codomains of the f_i unspecified. A bit of algebra shows that our approximation is the composition

$$f_5 \circ f_4 \circ f_3 \circ f_2 \circ f_1.$$

It will suffice to examine image of $\overline{\mathbb{H}} \setminus B(\sqrt{2\Delta t}, c)$ under the composition, piece by piece. The claims about conformality and continuity will follow.

Starting from the right, the first map dilates the domain by the factor $\frac{1}{\sqrt{2\Delta t}}$. This maps $\partial(B(\sqrt{2\Delta t}, c) \cap \mathbb{H})$ to $\partial(B(1, \frac{c}{\sqrt{2\Delta t}}) \cap \mathbb{H})$. It leaves points outside the upper half circle in $\overline{\mathbb{H}}$. The map is trivially bijective. Thus, $f_1 : \overline{\mathbb{H}} \setminus B(\sqrt{2\Delta t}, c) \twoheadrightarrow \overline{\mathbb{H}} \setminus B(1, \frac{c}{\sqrt{2\Delta t}})$. Observe that $f_1(z)$ is real if, and only if, z is real.

The next map shifts the domain to the left by $\frac{c}{\sqrt{2\Delta t}}$. This maps $\partial(B(1, \frac{c}{\sqrt{2\Delta t}}) \cap \mathbb{H})$ to $\partial(B(1, 0) \cap \mathbb{H})$. It leaves points outside the upper half circle in $\overline{\mathbb{H}}$. The map is trivially bijective. Thus, $f_2 : \overline{\mathbb{H}} \setminus B(1, \frac{c}{\sqrt{2\Delta t}}) \twoheadrightarrow \overline{\mathbb{H}} \setminus B(0, 1)$. Observe that $f_2(z)$ is real if, and only if, z is real.

For the third map, let $z \in \overline{\mathbb{H}} \setminus B(1, 0)$, and write $z = Re^{i\theta}$. Then $z + \frac{1}{z} = Re^{i\theta} + \frac{1}{R}e^{-i\theta} = (R + \frac{1}{R})\cos\theta + i(R - \frac{1}{R})\sin\theta$. Suppose first that $R > 1$. Since $\theta \in [0, \pi]$, $\sin\theta \geq 0$, which implies that the image of z under the third map is in fact in $\overline{\mathbb{H}}$. Next, if $R = 1$ then the image of z is real because its imaginary part vanishes. Conversely if the image of z is real then either $R = 1$, $\theta = 0$, or $\theta = \pi$. We leave it for the reader to check that f_3 is a bijection from $\overline{\mathbb{H}} \setminus B(1, 0)$ to $\overline{\mathbb{H}}$.

The fourth map and fifth maps are analogous to the second and first. They just reshift and dilate the domain in the opposite ways as f_2 and f_1. The result follows. $\square$

This theorem tell us that the only things mapped to real line, under our approximation, are the real itself and the boundary of the upper half circle $B(\sqrt{2\Delta t}, c)$. What about the points inside of $B(\sqrt{2\Delta t}, c)$? Technically, we have not defined our approximation here, as it would map these points below the real line. This, however, is merely an artifact of the approximation, *not* the actual Loewner flow g_t, which never sends points out of $\overline{\mathbb{H}}$. However, since the approximation is conformal, we know that these points must map down to the real line (or below it)–they will never return to $\mathbb{H}$.

From this discussion, we can conclude that the only possible points that could be captured by $\lambda(t)$ are those inside $\overline{B}(\sqrt{2\Delta t}, \lambda(t))$. Indeed, if z_t were captured at time t then at some time $s < t$, we must have $z_s \in \overline{B}(\sqrt{2\Delta t}, \lambda(s))$. Though it is conceivable that there are other points inside of $\overline{B}(\sqrt{2\Delta t}, \lambda(s))$ that will never be captured, when Δt is small and we only sample finitely many points, there can not be "too much" excess. Our simulation then, will be as simple as applying the approximation to points in $\overline{\mathbb{H}}$ and counting them as captured whenever they are inside $\overline{B}(\sqrt{2\Delta t}, \lambda(s))$ for some s.

2.2 A First Algorithm

Now we are ready to write down a simple algorithm for hull simulation. We will represent a rectangular section of the upper half plane as a uniform grid of points in $\overline{\mathbb{H}}$. We will then repeatedly apply the Loewner flow approximation over some time interval $[0, T]$ to this grid and test for when points lie within the upper half circles previously described. If, so we will say they have been captured and plot them as part of the hull.

A few observations can improve efficiency a bit. First, since λ is continuous it must attain its extrema on the closed time interval. Thus we can never expect to capture points with real part larger than $\sup_{t\in[0,T]} \lambda(t)$ or with real part smaller than $\inf_{t\in[0,T]} \lambda(t)$. There are algorithms for finding these extrema, given $\lambda(t)$, but a practical realization must always have a failsafe for when they do not converge. We can think of our left and right bounds for the grid as coming from these algorithms or being specified by the user. Finally, we can never expect to capture points with imaginary part larger than $2\sqrt{T}$, giving us a top bound for the grid (see example 1.1.3). Of course, our bottom bound for the grid is by default 0.

Algorithm 1 Let `nTimeSteps` denote the number of time steps the simulation is to run; `nGridRows` and `nGridCols` denote the grid dimensions; and `rightBound`, `leftBound`, and `topBound` denote the bounds on the grid. Let `M` denote a boolean indexing matrix of the same size as the grid, and denote the entry-wise negation of `M` by `M'`. `M` will represent whether the corresponding entry in the grid has been captured.

```
1.)  INPUT: Δt, λ(t), nTimeSteps, nGridRows, nGridCols.

2.)  Create the matrix Grid using  nGridRows, nGridCols, rightBound, leftBound,
     and topBound.

3.)  Set M to have the same dimensions as Grid, with all entries 0.

4.)  for j=1,...,nTimeSteps

          Set tempM = M;

          Test Grid(M') for capture criterion.  Update M(M') accordingly.

          Apply Loewner Approximation to Grid(M'), store result in Grid(M');

          Plot Grid(M != tempM);

     end
```

As one can see, this code makes extensive use of logical indexing. The idea is that at each iteration we test only the uncaptured points in the capture criterion. Next,

we plot those points that were captured just this iteration, so the user can watch the hull evolve, without re-plotting old points. Finally, we apply the flow only to the points that remain uncaptured.

The space complexity of the algorithm is clearly proportional to the grid dimensions. Though the amount of space is increased by the use of the boolean matrices, these are typically easier to store than regular matrices and increase the algorithm's speed.

The time complexity of the algorithm is proportional to the number of time steps multiplied by the grid's dimensions.

An advantage to this algorithm is that it is simple and makes good use of logical indexing. There is, however, one significant disadvantage: the grid's is uniform spacing. This means that no matter how the driving function behaves at a given time, the algorithm samples the same amount of points. Thus, for a specified grid, if the driving function moves too quickly, the algorithm might not detect the capturing of some points. One solution to this problem would be to create a nonuniform grid with densities proportional to the change in the driving function, but we do not employ that technique here.

Another small drawback is that even though our rectangular grid can be sized based on the extrema of the driving function as well as a height bound; it still may be a significant oversampling of points, especially when the hull is thin. The price we pay in memory is still very small though, and a re-fitting technique does not save us enough memory to be worth its increase in time.

2.3 A Second Algorithm

Here we introduce another algorithm for hull simulation, which is based on similar ideas as the first algorithm, but has the advantage of never having to test whether points have been captured. It also ensures that we only store data points that will be part of the (approximate) hull. The price we pay is in performance.

In section 2.1, we learned that our approximation formula was a conformal mapping from $\mathbb{H} \setminus B(\sqrt{2\Delta t}, c)$ to $\mathbb{H}$. In fact, it was bijection from $\overline{\mathbb{H}} \setminus B(\sqrt{2\Delta t}, c)$ to $\overline{\mathbb{H}}$. Thus, we can speak of its inverse. It turns out to be the map

$$z \mapsto \frac{z + c + \left(\sqrt{z - c - 2\sqrt{2\Delta t}}\right)\left(\sqrt{z - c + 2\sqrt{2\Delta t}}\right)}{2}.$$

Recall that c is $\lambda(t)$ at some fixed time t. Note that we use the principal branch cut when defining the square root.

Since our approximation formula mapped upper half circles, which which contained the hulls, down to the real line, the inverse map will do the opposite. The inverse map will also be conformal.

For this algorithm, we will think of time starting at T and running backwards to 0. Our algorithm will involve creating the same upper half circles at each time step, but then inverting them under this inverse map. After sufficient inversion, the result will be a collection of initial values for the approximation, specifically, the ones which would be captured under the forward approximation. This gives us an approximate hull.

Algorithm 2 Let `H` be a matrix whose columns will contain the data points describing inverse images of the upper half circles created at each time step. Define `Circ(rad, c)` to be function which returns, as a column vector, points describing an upper half circle with radius `rad` and center `c`. The size of the upper half circle vectors returned will be unspecified here. Let `nTimeSteps` denote the number of time steps the simulation is to run. Note that our indexing will start at 1, and that H will contain inverse images of one per column. Thus, H will have `nTimeSteps` columns.

1.) INPUT: nTimeSteps, $\lambda(t)$, Δt.

2.) Set the last column of H = Circ($2\sqrt{\Delta t}$, λ((nTimeSteps-1)*Δt))

3.) for j=nTimeSteps, down to 2

Apply inverse map to columns j,..,nTimeSteps of H.

Store Circ($2\sqrt{\Delta t}$, λ((j-2)*Δt)) in column j-1 of H.

end

4.) Plot H

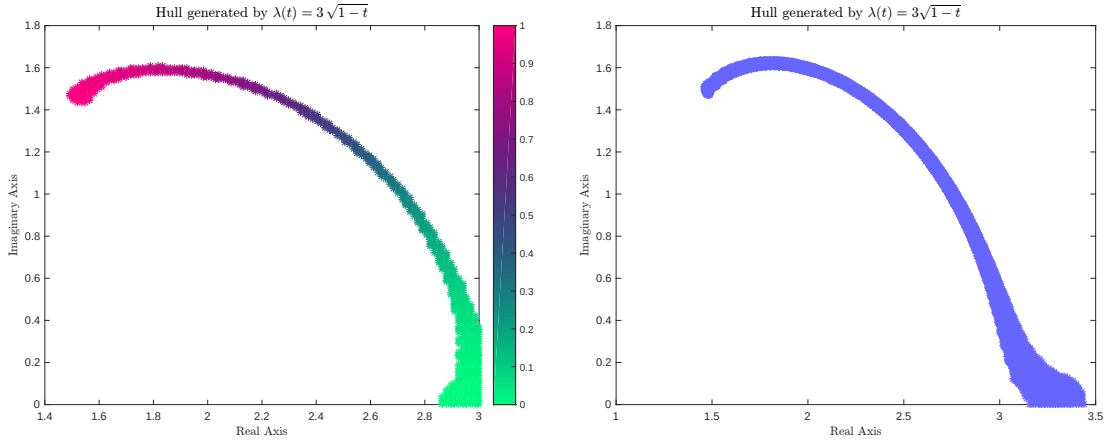
We start with the half circle from the second to last time step, since in the in next *forward* time step (if there were one), this half circle would map down to the real line, and we would color each point in it as captured. In the loop, we need to invert all previous upper half circles, to ultimately map them back to their starting values at time 0. We then create a new upper half circle in column `j-1` of `H`, in anticipation of it being inverted the next iteration (and all subsequent iterations). Note that efficiency can be slightly improved by first storing a constant upper half circle with radius $2\sqrt{\Delta t}$ and center 0. Then, at every iteration of the loop, the new upper half circle we need to create will just be the upper half constant circle shifted by λ((j-2)*Δt)). In doing this, we only need to invoke our circle creation function once.

Note also that we have not specified exactly how the upper half circles are represented. Just like the grid, they can be represented as a matrix of points which trace out their boundary and fill their interior. If there are d points within each half circle then the algorithm's space complexity is proportional to d times the number of time steps.

Let N be the number of time steps. We can sum up the number of inverse map function calls to be $1 + 2 + \cdots + (N-2) + (N-1) = \frac{N(N-1)}{2}$. Since the number of arithmetical operations (in the inverse map) is constant, the number of total arithmetical operations performed in the algorithm is $\mathcal{O}(d \cdot \frac{N(N-1)}{2})$.

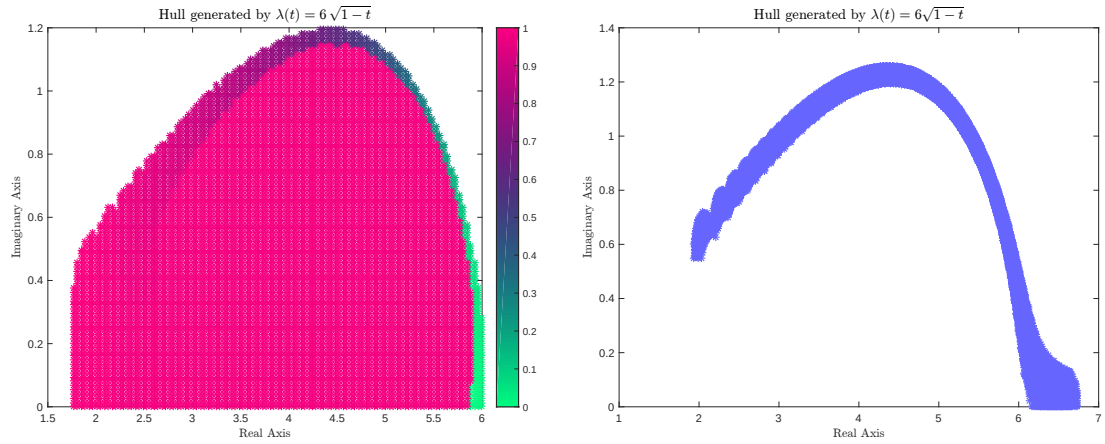
2.4 Some Simulation Results

Below are some images of hulls generated by certain driving functions, obtained from our MatLab program with both algorithms. The hulls represented in figure 2.1 are generated by $\lambda(t) = 3\sqrt{1-t}$, which has $\|\lambda\|_{1/2} < 4$, so we expect these hulls to be simple. The simulations seem to agree. The hulls represented in figure 3.1 are generated by $\lambda(t) = 6\sqrt{1-t}$, which has $\|\lambda\|_{1/2} > 4$, so we expect these hulls to be non-simple. In this case, the second algorithm fails to detect a significant portion of the hull–the part underneath the curve that is added at time 1. The hulls represented in figure 2.3 are generated by $\lambda(t) = \sin(t)$. Notice the periodicity in the hull.



(a) Algorithm 1 (b) Algorithm 2

Figure 2.1: Results of each algorithm applied to $\lambda(t) = 3\sqrt{1-t}$



(a) Algorithm 1 (b) Algorithm 2

Figure 2.2: Results of each algorithm applied to $\lambda(t) = 6\sqrt{1-t}$

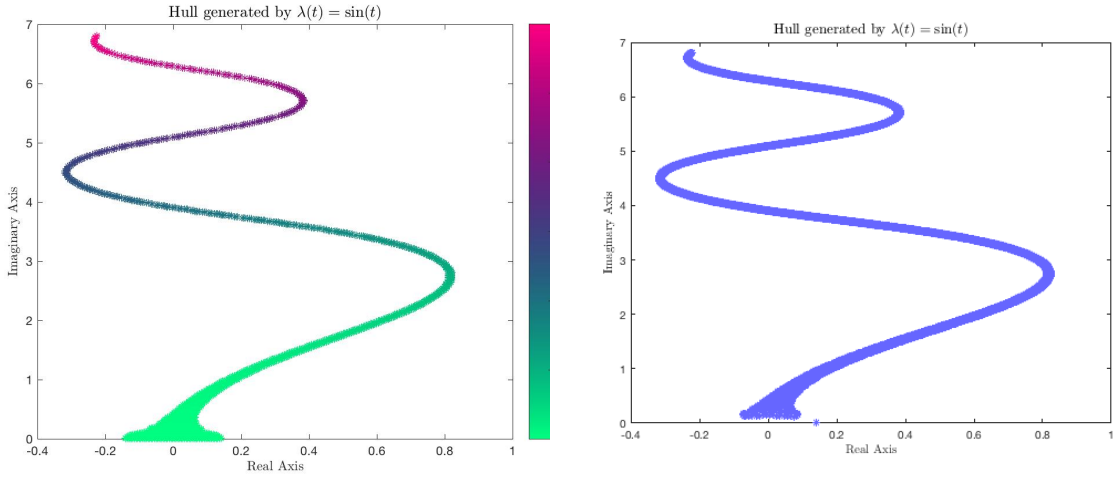


Figure 2.3: Results of each algorithm applied to $\lambda(t) = \sin(t)$
Algorithm 1 on the left. Algorithm 2 on the right.

Chapter 3

Weierstrass Functions

In 1872, Karl Weierstrass introduced the first published example of a function which is everywhere continuous yet nowhere differentiable ([4]). Technically, Weierstrass functions are a family of such functions subject to certain parameters a and b. The most general parameterization, introduced by G.H. Hardy ([2]), is as follows

Let $b > 1$ and $a \geq \frac{1}{\sqrt{b}}$. Then the Weierstrass function is defined

$$W_{a,b}(t) := \sum_{n=0}^{\infty} a^n \cos(b^n t).$$

The parameterization we are interested in is the edge case where $a = \frac{1}{\sqrt{b}}$. From now on, we will write just $W(t)$ for the the function $W_b(t) = \sum_{n=0}^{\infty} b^{-n/2} \cos(b^n t)$.

The purpose of this chapter is to generalize two results about bounds on $W(t)$ given in [5] as well as relate these bounds to the hulls generated by $W(t)$, which is also done [5]. The proofs on the bounds are thus very similar to those in [5], and the results on the hulls follow directly from the work done in [5]. At the end of the chapter we will also present some results of running $W(t)$ through our hull generation program.

3.1 Bounds on the Weierstrass Function

Recall from chapter one the norm $\|.\|_{1/2}$ we defined of $H^{1/2}(X)$, the vector space of all functions which are Hölder continuous with exponent $1/2$ on some set X. It is well known that the Weierstrass function is in $H^{1/2}(\mathbb{R})$. Our first theorem gives a bound on $\|W\|_{1/2}$.

Theorem 3.1.1. *Let $b > 1$. Then the norm $\|.\|_{1/2}$ of $W(t) = \sum_{n=0}^{\infty} b^{-n/2}\cos(b^n t)$ satisfies*

$$||W||_{1/2} \leq \frac{b}{\sqrt{b}-1} + \frac{2}{1-\frac{1}{\sqrt{b}}}.$$

Proof Set $a := \frac{1}{\sqrt{b}}$ and observe that

$$|W(t+h) - W(t)| \leq 2\max_{s\in\mathbb{R}}|W(s)| = 2\sum_{n=0}^{\infty} a^n = \frac{2}{1-a} = \frac{2}{1-\frac{1}{\sqrt{b}}}$$

.
Therefore, when $|h| \geq 1$, the claim holds. Henceforth assume $0 < |h| < 1$. Using the identity $\cos(x) - \cos(y) = -2\sin\left(\frac{x+y}{2}\right)\sin\left(\frac{x-y}{2}\right)$ implies

$$|W(t+h)-W(t)| \leq 2\sum_{n=0}^{\infty} a^n \left|\sin\left(\frac{b^n(2t+h)}{2}\right)\right| \left|\sin\left(\frac{b^n h}{2}\right)\right| \leq 2\sum_{n=0}^{\infty} a^n \left|\sin\left(\frac{b^n h}{2}\right)\right|.$$

Now, we can find a $p \in \mathbb{N}$ such that $b^{-p} \leq |h| \leq b^{-p+1}$. Using this p, we split the summation and get

$$\begin{aligned}
|W(t+h) - W(t)| &\leq 2\sum_{n=0}^{p-1} a^n \left|\sin\left(\frac{b^n h}{2}\right)\right| + 2\sum_{n=p}^{\infty} a^n \left|\sin\left(\frac{b^n h}{2}\right)\right| \\
&\leq |h|\sum_{n=0}^{p-1}(ab)^n + 2\sum_{n=p}^{\infty} a^n \\
&= |h|\frac{a^p b^p - 1}{ab-1} + 2\frac{a^p}{1-a} \\
&\leq a^p\left(|h|\frac{b^p}{ab-1} + \frac{2}{1-a}\right),
\end{aligned}$$

where we have used that $|\sin x| \leq |x|$ and $\sum_{n=0}^{p-1} r^n = \frac{r^p-1}{r-1}$. But $b^{-p} \leq |h| \leq b^{-p+1}$

so $b^p|h| \leq b$ and $a^p = \sqrt{b^{-p}} \leq \sqrt{|h|}$. Hence,

$$|W(t+h) - W(t)| \leq \sqrt{|h|}\left(\frac{b}{ab-1} + \frac{2}{1-a}\right) = \sqrt{|h|}\left(\frac{b}{\sqrt{b}-1} + \frac{2}{1-\frac{1}{\sqrt{b}}}\right).$$

□

Our next result gives a dense set of local maxima for $W(t)$. Its drawback is our assumption on b.

Theorem 3.1.2. *Suppose b is an integer greater than 1. Let $t_{m,k} = \frac{2m\pi}{b^k}$ for fixed $m \in \mathbb{Z}$ and $k \in \mathbb{N}$. Then there is an $\eta \in \mathbb{N}$ such that if $0 < |h| \leq b^{-(k+\eta)}$ then $W(t_{m,k}) - W(t_{m,k} + h) \geq \kappa\sqrt{|h|}$ for some $\kappa > 0$.*

Note: the numbers η and κ are given explicitly in the proof.

Proof Again we use the same trigonometric identity and obtain

$$\begin{aligned} W(t_{m,k}) - W(t_{m,k}+h) &= 2\sum_{n=0}^{\infty} a^n \sin\left(b^n t_{m,k} + \frac{b^n h}{2}\right)\sin\left(\frac{b^n h}{2}\right) \\ &= 2\sum_{n=0}^{\infty} a^n \sin\left(b^{n-k}(2m\pi) + \frac{b^n h}{2}\right)\sin\left(\frac{b^n h}{2}\right) \\ &= 2\sum_{n=0}^{k-1} a^n \sin\left(b^{n-k}(2m\pi) + \frac{b^n h}{2}\right)\sin\left(\frac{b^n h}{2}\right) + 2\sum_{n=k}^{\infty} a^n \sin^2\left(\frac{b^n h}{2}\right). \end{aligned}$$

where the last step uses the fact that if $n \geq k$ then $\sin\left(b^{n-k}(2m\pi) + \frac{b^n h}{2}\right) = \sin\left(\frac{b^n h}{2}\right)$. Now let B and T represent the beginning and the tail of the sum, represented by the first and second terms in last equation, respectively. We will establish lower bounds on B and T and show that the sum of these bounds is a positive multiple of $\sqrt{|h|}$. First we bound B. By a similar argument as in the previous proof,

$$\begin{aligned} B &= 2\sum_{n=0}^{k-1} a^n \sin\left(b^{n-k}(2m\pi) + \frac{b^n h}{2}\right)\sin\left(\frac{b^n h}{2}\right) \\ &\geq -2\sum_{n=0}^{k-1} a^n \left|\sin\left(\frac{b^n h}{2}\right)\right| \geq -|h|\sum_{n=0}^{k-1}(ab)^n \\ &\geq -|h|\left(\frac{(ab)^k}{ab-1}\right). \end{aligned}$$

Recall that $|h| \leq b^{-(k+\eta)}$ and $ab = \sqrt{b}$. Thus,

$$B \geq -|h| \left(\frac{b^{k/2}}{\sqrt{b}-1} \right) = -\sqrt{|h|} \left(\frac{\sqrt{|h|} b^{k/2}}{\sqrt{b}-1} \right) \geq -\sqrt{|h|} \left(\frac{b^{-\eta/2}}{\sqrt{b}-1} \right).$$

We must now bound T. Without loss we may assume that $h > 0$ because $\sin^2(-x) = \sin^2(x)$. Since all terms in T are positive, we establish a lower bound using a partial sum. Namely,

$$T = 2\sum_{n=k}^{\infty} a^n \sin^2\left(\frac{b^n h}{2}\right) \geq 2\sum_{n=k}^{p} a^n \sin^2\left(\frac{b^n h}{2}\right) \geq 2\sum_{n=k}^{p-1} a^n \sin^2\left(\frac{b^n h}{2}\right),$$

where $p \in \mathbb{N}$ satisfies $b^{-p} \leq h \leq b^{-p+1}$. This expression is well defined whenever $p-1 \geq k$. We know $h \leq b^{-(k+\eta)}$ so $b^{-p} \leq b^{-(k+\eta)} \implies -p \leq -(k+\eta) \implies p-1 \geq p-\eta \geq k$. Now observe that if $x \in [0, \frac{1}{2}]$ then $\sin x \geq 2\sin(\frac{1}{2})x$. We apply this property to our bound by noting that if $n \leq p-1$ then $0 \leq \frac{b^n h}{2} \leq \frac{b^{p-1}h}{2} \leq \frac{1}{2}$. Thus,

$$T \geq 2\sum_{n=k}^{p-1} a^n \sin^2\left(\frac{b^n h}{2}\right) \geq 2\sum_{n=k}^{p-1} a^n \left(\frac{2\sin(\frac{1}{2})}{2} b^n h\right)^2 = 2\sin^2\left(\tfrac{1}{2}\right) h^2 \sum_{n=k}^{p-1} (ab^2)^n.$$

Define $r := ab^2 = b^{3/2}$ and recall that $\sum_{n=k}^{p-1} r^n = \frac{r^p - r^k}{r-1}$. Also since $b^{-p} \leq h$ we have $h^2 = \sqrt{h}\, h^{3/2} \geq \sqrt{h}\, (b^{-p})^{3/2} = \sqrt{h}\, r^{-p}$. Putting this together we obtain

$$\begin{aligned} T &\geq 2\sin^2(\tfrac{1}{2}) \cdot h^2 \sum_{n=k}^{p-1} r^n \geq 2\sin^2\left(\tfrac{1}{2}\right) \cdot \sqrt{h} \cdot r^{-p} \left(\frac{r^p - r^k}{r-1}\right) \\ &= 2\sin^2\left(\tfrac{1}{2}\right) \left(\frac{1-r^{k-p}}{r-1}\right) \sqrt{h} \geq 2\sin^2\left(\tfrac{1}{2}\right) \left(\frac{1-r^{-\eta}}{r-1}\right) \sqrt{h} \\ &= 2\sin^2\left(\tfrac{1}{2}\right) \left(\frac{1-b^{\frac{-3\eta}{2}}}{b^{\frac{3}{2}}-1}\right) \sqrt{h}. \end{aligned}$$

We now have only to show that the sum of the lower bounds on B and T is a positive multiple of $\sqrt{h}$. That is, we must show that

$$2\sin^2\left(\tfrac{1}{2}\right) \left(\frac{1-b^{\frac{-3\eta}{2}}}{b^{\frac{3}{2}}-1}\right) - \left(\frac{b^{\frac{-\eta}{2}}}{\sqrt{b}-1}\right) > 0.$$

Set $\mu := 2\sin^2\left(\frac{1}{2}\right)$. Then the above becomes

$$\frac{\mu\left(1-b^{\frac{-3\eta}{2}}\right)\left(\sqrt{b}-1\right)-\left(b^{\frac{-\eta}{2}}\right)\left(b^{\frac{3}{2}}-1\right)}{\left(b^{\frac{3}{2}}-1\right)\left(\sqrt{b}-1\right)} \geq \frac{\mu\left(1-b^{\frac{-3}{2}}\right)\left(\sqrt{b}-1\right)-\left(b^{\frac{-\eta}{2}}\right)\left(b^{\frac{3}{2}}-1\right)}{\left(b^{\frac{3}{2}}-1\right)\left(\sqrt{b}-1\right)}.$$

The denominator is a product of positive terms, so it is also positive. To ensure that the numerator is also positive, we pick an appropriate η. Namely, for $\varepsilon := \mu\left(1-b^{\frac{-3}{2}}\right)\left(\sqrt{b}-1\right) > 0$ we may pick an η large enough so that

$$\left(b^{\frac{-N}{2}}\right)\left(b^{\frac{3}{2}}-1\right) < \varepsilon$$

for all $N \geq \eta$. Explicitly, any

$$\eta > 2\log\left(\frac{b^{\frac{3}{2}}-1}{\mu\left(1-b^{\frac{-3}{2}}\right)\left(\sqrt{b}-1\right)}\right) \Big/ \log(b)$$

will do the job. With such an η, the numerator is positive for all $N \geq \eta$. □

We also obtain a corresponding set of local minima for $W(t)$. If b is an odd integer, n is a positive integer, and m is an odd integer, then the numbers $t_{m,k} = \frac{m\pi}{b^k}$ form such a set. The proof is directly analogous to the proof of the previous theorem.

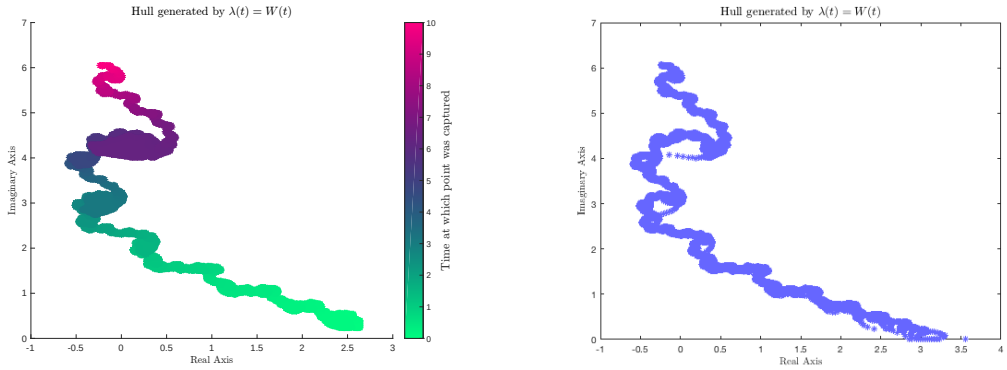
3.2 The Weierstrass Function as a Driving Function

In [5] the authors prove, using their established bounds, that the hulls generated by any multiple c of their Weierstrass function $\sum_{n=0}^{\infty} 2^{-n/2}\cos(2^n t)$ exhibit a phase transitions somewhere between $c = 1/3$ and $c = 20$. We harvest an analogous result for our slightly more general Weierstrass function $W(t) = \sum_{n=0}^{\infty} b^{-n/2}\cos(b^n t)$, via their methods.

Namely, theorem 1.2.3 says that if $c < \frac{4}{\|W\|_{1/2}}$ then the hulls generated by $W(t)$ are all simple curves. We gave an explicit bound on the norm $\|.\|_{1/2}$ of our $W(t)$ in theorem 3.1.1. So it is sufficient that c be less than 4 divided by this bound to guarantee these hulls are simple curves.

On the other hand, suppose κ is the constant corresponding to the smallest η from theorem 3.1.2 (given explicitly in the proof). Then theorem 1.2.4 as well as the proof of theorem (1.1) in [5] imply the hulls generated by $W(t)$ are non-simple if $\kappa \cdot c > 4$.

Here are the results of running the ($b = 2$) Weiestrass function through our program, using both algorithms. As we saw before, it looks like algorithm one is doing a better job of detecting what could be space-filling behavior.



(a) Algorithm 1 (b) Algorithm 2

Figure 3.1: Results of each algorithm applied to $W(t)$

Bibliography

[1] G. Glenn, Loewner Flow Software, https://github.com/gglenn2/Loewner_Flow_Project

[2] G.H. Hardy, "Weierstrass's nondifferentiable function," *Trans. Amer. Math Soc.* **17** (1916), 301-325.

[3] W. Kager, B. Nienhuis, and L. Kadanoff, "Exact solutions for Loewner Evolutions," *J. Statist. Phys.* **115** (2004), 805–822.

[4] K. Weierstrass, "Über continuirliche Functionen eines reellen Arguments, die für keinen Werth des letzeren einen bestimmten Differentialquotienten besitzen, Königlich Preussichen Akademie der Wissenschaften," *Mathematische Werke von Karl Weierstrass,* **2**, Mayer & Mueller, Berlin, Germany, 1895.

[5] J. Lind and J. Robins, "Loewner deformations driven by the Weierstrass function," *Involve* **10** (2017), no 1, 151–164.

[6] J. Lind and S. Rohde, "Spacefilling curves and phases of the Loewner equation," *Indiana Univ. Math. J.* **61** (2012), no 6, 2231–2249.