TECHNICAL UNIVERSITY OF MUNICH

DEPARTMENT OF INFORMATICS

Bachelor's Thesis in Informatics

Interactive Tools for Visualising Phase Plots of Complex Functions

Gerhard-Mathias Konnerth

TECHNICAL UNIVERSITY OF MUNICH

DEPARTMENT OF INFORMATICS

Bachelor's Thesis in Informatics

Interactive Tools for Visualising Phase Plots of Complex Functions

Interaktive Tools zur Visualisierung von Phasendiagrammen komplexer Funktionen

Author:	Gerhard-Mathias Konnerth
Supervisor:	Prof. Dr. Dr. Jürgen Richter-Gebert
Advisor:	M. Sc. Aaron Montag
Submission Date:	8 August 2017

I confirm that this bachelor's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 8 August 2017

Gerhard-Mathias Konnerth

Acknowledgments

Firstly, I would like to thank Prof. Dr. Dr. Jürgen Richter-Gebert for proposing this wonderful subject. While working on this thesis, I have had the freedom to explore the domain of complex analysis as depicted by [Wegert 2012] independently and to select those topics which I found most impressive and most suitable for a beautiful visualization.

I am also grateful to M. Sc. Aaron Montag for the support regarding technical difficulties related to CindyJS and for guiding me towards possible improvements of the approximations which I used in the visualization tools. The comprehensive feedback on my work is also very much appreciated.

Lastly, I would like to thank Mr Eduard Pop for pointing out the grammatical and syntactical flaws one is likely to overlook when repeatedly reading and improving one's own work.

Abstract

Visualizations of mathematical concepts are of great educational value. In this thesis, focus lies on the usage of phase plots in complex analysis. The JavaScript framework CindyJS [*CindyJS* 2017] allows the production of mathematical content in an interactive fashion. Using this framework, I have implemented four pieces of software which aim to visualize the argument principle, power series, the Riemann sphere and analytic landscapes, respectively. The content is intended to be intuitive and self-explanatory. Ideally, the user would discover the tools' features as well as the mathematical concepts which they represent autodidactically and by exploration. The following work briefly covers the mathematical background of the aforementioned principles of complex analysis, as well as the usability and the educational value of the respective visualization tool which I have produced. I will also describe essential parts of each implementation.

Contents

1 Introduction

1.1 Problem Statement

This thesis aims to visualize various ideas and theorems from complex analysis. The means of representation shall be the phase plot covered in detail by [Wegert 2012]. Four pieces of software were produced (often referred to as *visualization tools* or simply *tools*), each one centered around one specific concept. The objective is not only the display of information but also the fabrication of an intuitive and interactive visualization environment. Ideally the user would not only comprehend the presented mathematical concept, but also explore its behavior for different input, as well as the respective visualization tool's features and limitations. Besides discussing matters of complex number theory and details regarding the implementation for each subject, I also try to present aspects of usability and educational value of each tool. Selected examples shall emphasize some implications of the mathematical concepts, as well as the utility of the tools' features.

Each piece of code was written using *CindyJS* [*CindyJS* 2017], a framework centered around the interactive visualization of mathematical concepts. It is intended for web programming.

1.2 Contribution

The four mathematical concepts which represent the bases for the visualization tools as well as the theory behind them are inspired by [Wegert 2012]. My contribution lies in the conversion of selected ideas form [Wegert 2012] into interactive visualization mechanisms. I designed, selected and assembled some procedures to approximate the mathematical entities which are necessary for said ideas and principles to work on a variety of inputs. Moreover, analysis of the usability of the produced tools is in the center of interest.

1.3 Overview

Before proceeding to the discussion about the tools, several basic concepts regarding complex number representation will be clarified briefly in Chapter 2. In particular these include phase plots and the properties they reflect, as well as phase plot representation in CindyJS. The reader who is experienced with those aspects might consider skipping this chapter. Afterwards, we proceed to the four concepts of complex analysis and their visualization which this thesis aims to cover.

The Argument Principle [Wegert 2012, p. 102] can be regarded as a fundamental concept behind the visualization approach which is adopted from [Wegert 2012, p. 4]. It offers information about the number of roots of a complex function inside a given curve (with some constraints) only by analyzing numbers on said curve. The visualization tool shall demonstrate this for arbitrary curves in Chapter 3.

Power Series [Wegert 2012, p. 73-] are infinite sums of the form $\sum_{n=0}^{\infty} a_n(z-z_0)^n$. Here, especially the convergence of the series inside a disk is of special interest, since it yields a beautiful visualization. Chapter 4 will include the observation of the (approximate) identity between a partial sum [Wegert 2012, p. 73] $f_k = \sum_{n=0}^{k} a_n(z-z_0)^n$ and a given function g inside a disk on the complex plane for a certain k_0 and all $k \geq k_0$.

The Riemann Sphere is a perception of complex analysis where numbers are projected from the complex plane onto a sphere [Wegert 2012, p. 20-]. It yields a different view on complex function representation. Moreover, [Wegert 2012, p. 20-] argues that it offers the possibility to geometrically interpret the point at infinity. In Chapter 5 a geometrical construction of the Riemann Sphere is described.

Analytic Landscapes [Wegert 2012, p. 27-] involve not the phase of complex numbers, but their modulus, unlike the representations discussed so far. Including color analogously to phase plots yields an attempt to represent complex functions in their entirety. As the name suggests, analytic landscapes are three-dimensional objects, which will be depicted and discussed in an interactive 3D environment in Chapter 6.

2 Fundamentals

Before proceeding to the visualization tools introduced in Chapter 1, it is necessary to cover the basic concepts which are used here to represent complex functions. The term complex functions is used to refer to functions of the form given in Equation (2.1). Arguably, the most intuitive and educational mode to analyze a function is by discussing a graphical representation. This statement shall be supported in the following chapters by various examples of functions and their properties, which are deducible from a visualization.

$$f : D \to \mathbb{C}, D \subseteq \mathbb{C} \tag{2.1}$$

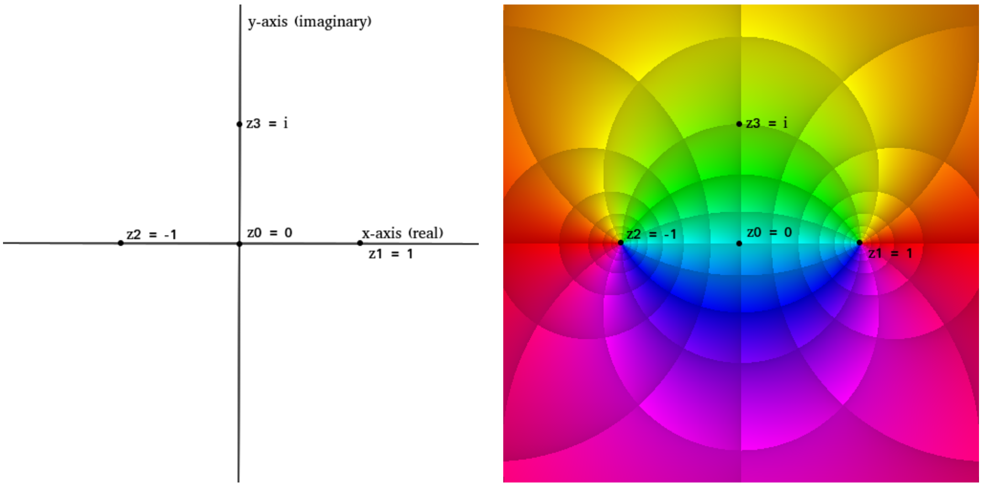
2.1 Complex Number Visualization

Since the graph of a complex function lives in a four-dimensional space [Kranich 2016, p. 90], it is not trivial to visualize. The visualization approach used in this thesis is based on the polar form of a complex number, where the complex number $z = a + bi$ is represented using its modulus $|z| = \sqrt{a^2 + b^2}$ and its argument $\arg(z)$. The latter is obtained using the property $\tan(\arg(z)) = \frac{b}{a}$ while considering in which quadrant z is located. Finally, this produces the representation $z = |z| \exp(i \arg(z))$ [Bornemann 2013, p. 4]. [Wegert 2012, p. 3] argues that $\arg(z)$ is more suitable for visualization than $|z|$, which is why plotting a phase as described in Section 2.2 is adopted as a visualization approach in this thesis.

2.2 Phase Plots

For now, we focus on two-dimensional representations of the above defined f. Since the domain D of f lives in two-dimensional space, colors are used to represent $\arg(f(z))$. One way to convey the intuition behind such a plot, which will be referred to as *phase plot*, following the notation from [Wegert 2012, p. 30], is by discussing an example.

Figure 2.1b shows the phase plot of the function $f(z) = \frac{z-1}{z+1}$. The two dimensional colored region serves as the complex plane, where the positive imaginary axis points upwards and the positive real axis points to the right. The black points resemble example numbers located on the complex plane. Imagine interpreting the coordinate system in Figure 2.1a as the complex plane, computing the corresponding complex number z for each of the points, computing $f(z)$ and then assigning the color corresponding to $\arg(f(z))$ to that point.



(a)

(b)

Figure 2.1: Coloring the image of a function on the complex plane

The depicted section of the complex plane depends on the respective case. In these figures, the bottom left corner represents the point $(-2, -2)$ and the upper right one $(2, 2)$. The locations of z_0, z_1, z_2 and z_3 arise as depicted on the figures. Please note that their location does not change after "applying" f to the complex plane. Merely the color is affected by the function.

2.3 Seeing Properties on Phase Plots

At the beginning of Chapter 2 it was mentioned that representing the phase of a function reveals information about its properties [Wegert 2012, p. 30]. In this color scheme which is adopted from [Wegert 2012, p. 4], red represents $\arg(f(z)) = 0$, which increases in a counterclockwise order towards cyan, which represents $\arg(f(z)) = \pi$. Figure 2.2 shows the phase plot of the identity function in order to visualize the meaning of each color. The gray lines which appear concentric in Figure 2.2 represent the constant modulus $|f(z)|$, while the gray lines which lie on isochromatic lines on the phase plot represent the constant argument $\arg(f(z))$.

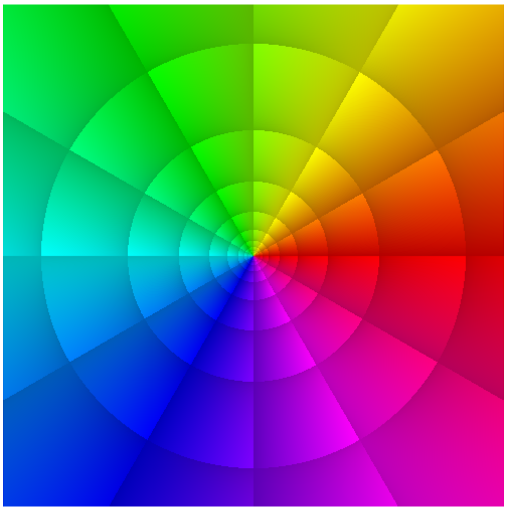


Figure 2.2: Phase plot of $id(z) = z$

Going back to Figure 2.1, the points z_2 and z_1 where all colors meet are of importance. [Wegert 2012, p. 4] offers an explanation on this matter: the fact that all colors meet at z_1 indicates that z_1 is a root of f. It is a simple root, which can be deduced from the phase plot by the fact that every color appears only once around z_1. The order of colors is of importance, since at z_2 we observe a similar pattern, but $\lim_{x \to z_2} f(x) = \infty$. When wandering around such a point in a counterclockwise direction, the order of colors red, yellow, green, etc. implies a root at that point, whereas the inverse order implies a pole.

2.4 Producing Phase Plots in CindyJS

The basis for most of the visualization tools which are going to be covered consists of a rectangular region where it is possible to manipulate points. This region shall be referred to as the canvas, following the notation from [Montag, Gagern, and Richter-Gebert 2017]. It represents an arbitrary rectangular sector of $\mathbb{R}^2$. Thus, points in the implementations can have two-dimensional coordinates, which determine their position on the canvas. We refer to these as coordinates in canvas space.

On this canvas we will draw phase plots, i.e. assign colors to each point in a square region applying the methods from Section 2.2. The algorithm which determines the color of a particular point is described by [Kranich 2016, p. 90]. Figure 2.3 depicts a canvas (with a black border around it) and two phase plots. In this example, the bottom left corner of the canvas has coordinates $(0, 0)$ and the upper right corner $(10.5, 6)$, the red dots indicate the corners of each plot. These points live in canvas space and for two of them the canvas coordinates are depicted (between square brackets) for better understanding.

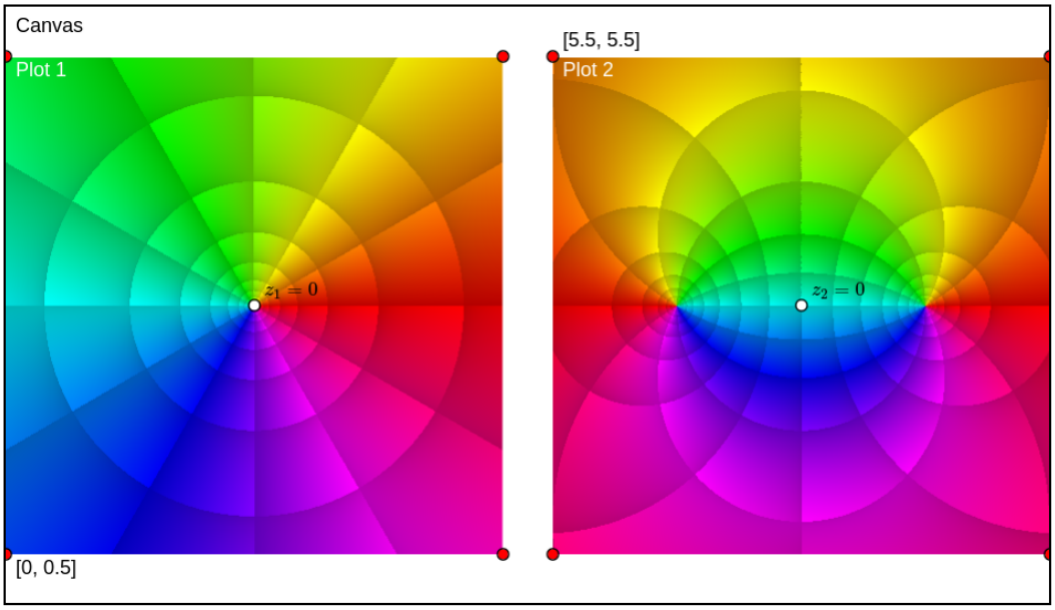


Figure 2.3: Canvas with points and phase plots

Both phase plots depict the same region of the complex plane. In this case it is the square between $(-2, -2)$ and $(2, 2)$. Consequently, the white dots in the center of each phase plot lie on top of the same complex number: 0, regardless of their coordinates in canvas space. Thus, when coloring those parts of a canvas where a phase plot is desired, basic transformations must be applied to determine the correct complex number corresponding to each point. Perceiving the difference between points on the canvas space and their value if interpreted as a complex number on a phase plot is necessary in order to understand the details on the implementations from the following chapters.

3 The Argument Principle

3.1 Problem Statement

Section 2.2 gave an overview on how properties of a function are reflected in its phase plot. Now, an investigation on the backgrounds of the aforementioned observations shall follow. In particular, I will discuss the color arrangement around roots and poles. As mentioned in Section 2.2 (and as mentioned by [Wegert 2012, p. 4]), all colors meet at a root. In the representation which is used in this thesis, the order of colors must be red, yellow, green, cyan, blue and purple when wandering around the root in a counterclockwise direction. From now on, I will refer to this order as the *positive direction* on the color circle. In the case of poles, the order of colors is reversed. The number of times the color circle is run through corresponds to the multiplicity of the root/pole. The reader is referred to [Wegert 2012, p. 4-] for details. As will be argued in the following paragraphs, the background of these observations lies in the argument principle for analytic functions. Before proceeding to its definition, winding and chromatic numbers will be investigated.

Winding. Let γ be a closed path around $z_0 \in \mathbb{C}$. The winding number $n(\gamma, z_0)$ reflects the number of times the curve γ passes around z_0 in a counterclockwise direction [Weisstein 2002b]. Analogously, a negative winding number is associated to a clockwise movement. From now on we shall consider only winding around $z_0 = 0$. Moreover, interest lies in in the mapping of the path γ under the function f, which [Wegert 2012, p. 101-] defines as the path $f \circ \gamma$ in the image set of f. The winding number of the image of the path γ arises, denoted by $\text{wind}_\gamma f$. Intuitively, one might think of the image path $f \circ \gamma$ as follows: consider $f : D \to \mathbb{C}$ where $D \subset \mathbb{C}$. Let $z \in D$ follow γ, then the trajectory of $f(z) \in \mathbb{C}$ corresponds to $f \circ \gamma$. [Wegert 2012, p. 101-] discusses these concepts in a more detailed manner.

Chromatic Number of a Path. The chromatic number $\text{chrom}_f \gamma$ as described by [Wegert 2012, p. 103-] is used to construct a connection between the winding of a path and the roots/poles of a function. Let $\text{chrom}_f \gamma$ be the number of times the path γ traverses the color circle (in positive direction) in the phase plot of the function f. The following holds: $\operatorname{chrom}_f \gamma = \operatorname{wind}_\gamma f$. The reader is referred to [Wegert 2012, p. 101-] for a detailed explanation. Intuitively, this implies that if path γ circles a simple root of f on its phase plot, we expect the path $f \circ \gamma$ to circle the point $z_0 = 0$ exactly once. A similar observation can be made regarding poles.

Figure 3.1 shows an example of this observation. On the left side, the path γ is represented by the black line in a counterclockwise direction. The phase plot represents the function $f(z) = (z - 0.5)(z + 0.5)$. A chromatic number of $\operatorname{chrom}_f \gamma = 2$ arises from the phase plot. Applying f on γ yields the path $f \circ \gamma$ on the right hand side of Figure 3.1, likewise in a counterclockwise direction. It is trivial to observe $\operatorname{wind}_{\gamma f} = 2$. The background, which is the phase plot if the identity function $\operatorname{id}(z) = z$, is chosen to emphasize this property.

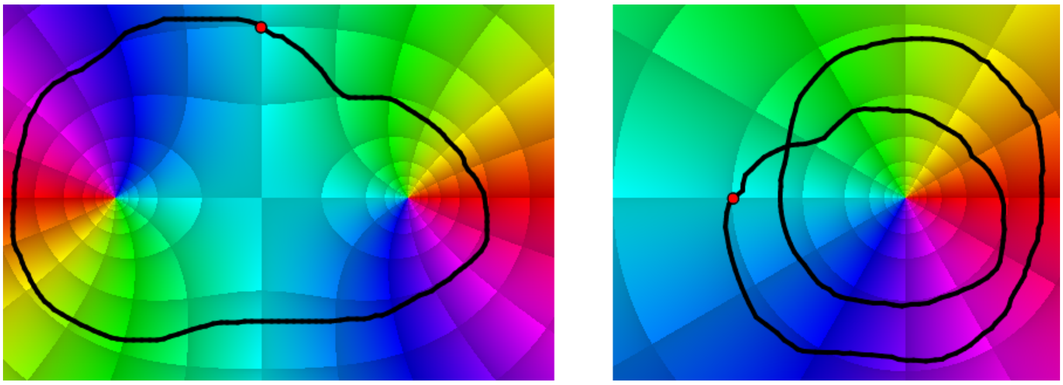


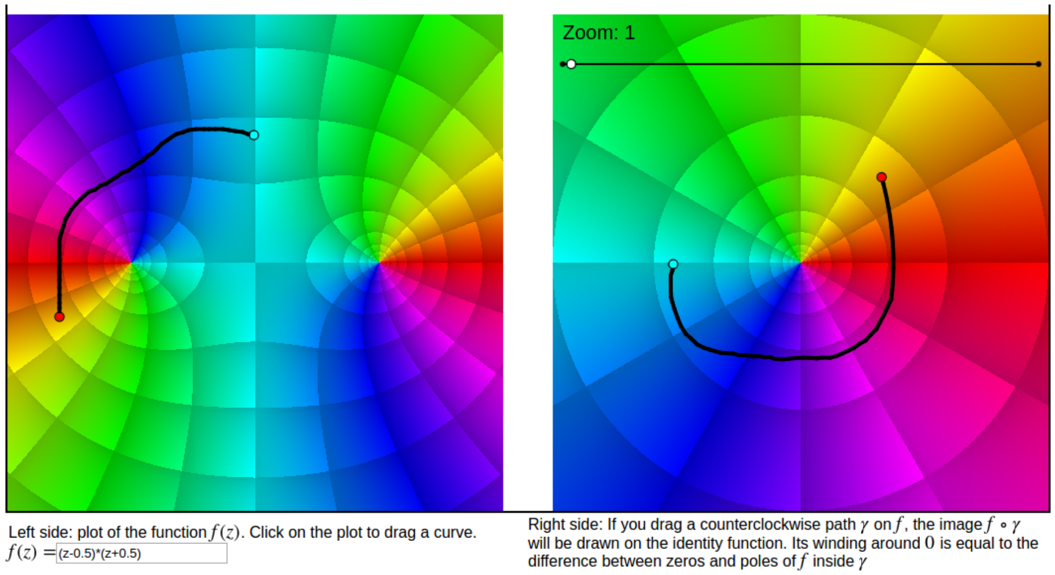
Figure 3.1: $\operatorname{chrom}_f \gamma = \operatorname{wind}_\gamma f$

The chromatic number and the number of roots inside a path. Let J be a positively oriented Jordan curve such that f has no roots on J. [Wegert 2012, p. 102] suggests that the values of f on J determine the number of roots of f in the interior of J. The Argument Principle for analytic functions states that for f analytic in D and $J \cup \operatorname{int} J \subset D$, $\operatorname{wind}_J f$ is equal to the number of roots of f in $\operatorname{int} J$ (where each root must be counted with its multiplicity) [Wegert 2012, p. 102]. From the above explanations, it follows that we can determine the number of roots in the interior of a curve on the phase plot of a function f by counting how many times the path runs through the color circle in a positive direction. Note that [Wegert 2012, p. 102] discusses only Jordan curves in this context. We generalize our visualization approach for arbitrary curves.

All the above information applies not only to roots of f, but also to poles. Every color appears around a pole, but the order of colors is opposite to that in the case of roots: red, purple, blue, cyan, green and yellow (in counterclockwise direction). Along with this observation, [Wegert 2012, p. 4] points out that the number of isochromatic lines emerging from a root or pole represents its multiplicity. Further observations and examples shall be discussed in Section 3.2, where the visualization tool is evaluated.

3.2 Evaluation

The aim of this chapter is to produce a piece of software, where the user may input a path γ on the phase plot of a function f and observe the winding of $f \circ \gamma$. The user is provided with two phase plots, as shown in Figure 3.2. The phase plot on the left hand side (in the following, this plot might also be referred to as "plot one") represents the arbitrary function f, which the user may input in the text field below it. On this plot, a path γ is to be input via mouse dragging, that is, as soon as the user clicks and holds the left mouse button, the points T (green) and P (red) are placed at the mouse position. T will remain fixed, while P follows the mouse position while the user is dragging. A black path will be drawn starting at point T, following P. On mouse button release the path will either disappear if it is not closed (i.e. P and T are further away from each other than a given ϵ), or will remain visible if the path is closed.



Left side: plot of the function $f(z)$. Click on the plot to drag a curve.
$f(z) =$ (z-0.5)*(z+0.5)

Right side: If you drag a counterclockwise path γ on f, the image $f \circ \gamma$ will be drawn on the identity function. Its winding around 0 is equal to the difference between zeros and poles of f inside γ

Figure 3.2: Path winding visualization

The visualization consists of two parts. On the one hand, while the path is drawn on plot one, on the right hand side (plot two) the combination $f \circ \gamma$ will be drawn. The user is able to view the construction of $f \circ \gamma$ in real time, which is useful for understanding its direction (clockwise or counterclockwise). That part of $f \circ \gamma$ which lies outside the plotting range of plot two will be hidden (i.e. cut off). For γ this is not possible, since the user cannot drag the point P outside plot one. In order to visualize the hidden parts of $f \circ \gamma$, the user may manipulate the zoom slider, which increases/decreases the plotting range for plot two.

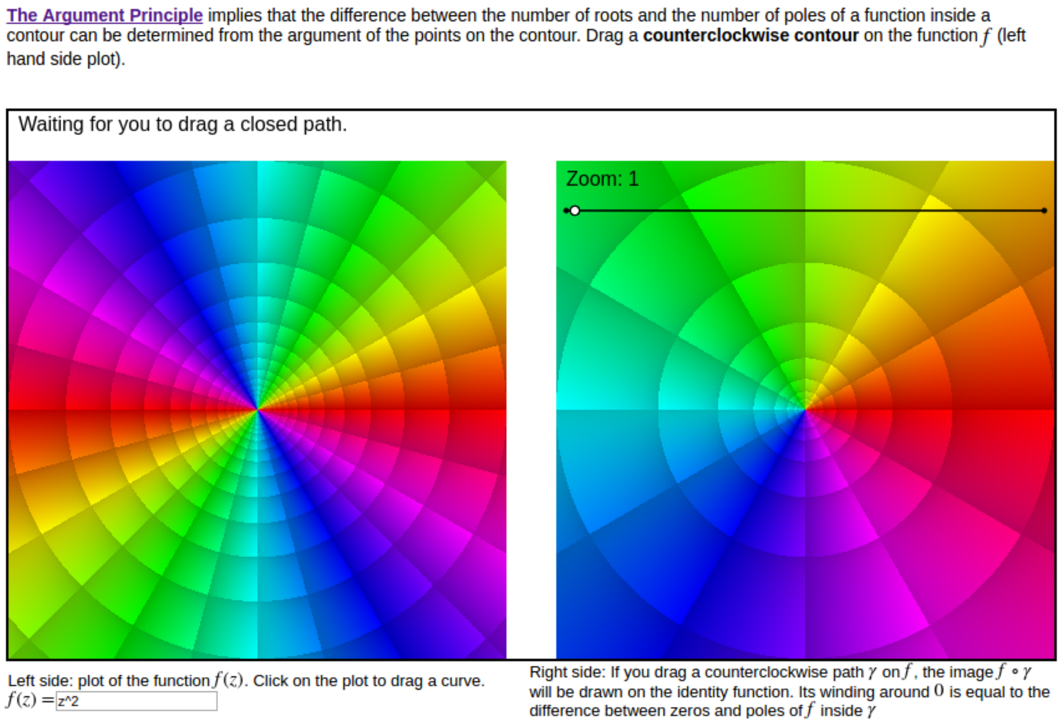
For the second part of the visualization, the tool will compute information regarding the number of roots and poles of f inside the drawn curve. For several simple functions it is possible to determine the exact number of roots or poles inside the drawn curve. In many cases this task becomes more demanding and is beyond the scope of this thesis. What focus lies on here is the difference between the number of roots and poles in the interior of γ. Throughout this chapter this number will be referred to as n_γ. This matter will be discussed in more detail in Section 3.2.1. n_γ is displayed above the plots after the user has finished drawing a closed path on plot one.

Figure 3.3 shows a complete view of the tool. The explanation regarding the mathematical background (i.e. the argument principle) is intended to be as short as possible, in order to capture the user's attention while providing the necessary information to use the tool. The only information which the user is meant to comprehend at first sight is their task to draw a counterclockwise (at first) contour on the left hand side plot. After the user starts drawing (i.e. dragging points), the mapping $f \circ \gamma$ will appear, which might encourage the user to read the short explanation below plot two. Lastly, after the path γ was closed, the text directly above the plots will change into a display of n_γ as defined above. This behavior is intended to encourage the user to focus on one visualization aspect at a time instead of splitting their attention and thus causing confusion. This is merely the reasoning behind the layout of the tool. Testing whether it represents an actual workflow or whether the tool does appear intuitive at first sight is beyond the scope of this thesis.

3.2.1 Selected Examples

Similarity of roots and poles

In Section 3.1 we discussed how roots and poles are reflected on a phase plot. The order of the colors around the point of interest was used in order to determine whether it was a pole or a root. This implies that circling a root in counterclockwise direction is equivalent to circling a pole in clockwise direction. This behavior is reflected in Figure 3.4, where plot one shows the function $f(z) = \frac{1}{z}$. Looking only at the third row of the figure, where the completed path γ is depicted, one might argue that the observations from Section 3.1 about the chromatic number and the number of roots in the interior of γ have been disproved. There are no roots of f in the interior of γ,



The Argument Principle implies that the difference between the number of roots and the number of poles of a function inside a contour can be determined from the argument of the points on the contour. Drag a **counterclockwise contour** on the function f (left hand side plot).

Waiting for you to drag a closed path.

Left side: plot of the function $f(z)$. Click on the plot to drag a curve.
$f(z) =$ z^2

Right side: If you drag a counterclockwise path γ on f, the image $f \circ \gamma$ will be drawn on the identity function. Its winding around 0 is equal to the difference between zeros and poles of f inside γ

Figure 3.3: Full view of the visualization tool

but the point $z_0 = 0$ is in the interior of $f \circ \gamma$, therefore $\text{wind}_\gamma f = 1$ seems to be the case. However, since γ is constructed in counterclockwise direction while $f \circ \gamma$ travels clockwise around 0, the winding number $\text{wind}_\gamma f$ of the latter is not 1 but -1.

In Section 3.1 we concluded that for every time γ travels around a root on plot one, we expect $f \circ \gamma$ to circle the point $z_0 = 0$ in plot two. In this example γ circled a pole and we expect a similar behavior to the previous case, namely for $f \circ \gamma$ to circle $z_\infty = \infty$ in plot two, which in some way happens. This example unfolds a view on complex analysis which shall be processed in more detail in Chapter 5. For now, the reader might imagine that the interior of the path $f \circ \gamma$ on plot two contains (when viewed in a counterclockwise direction) not the interior of the circle from a geometrical point of view, but everything which is around it (a visualization attempt has been made in Figure 3.5). As if the complex plane were the surface of a sphere, where the point z_∞ lies on the opposite pole to z_0.

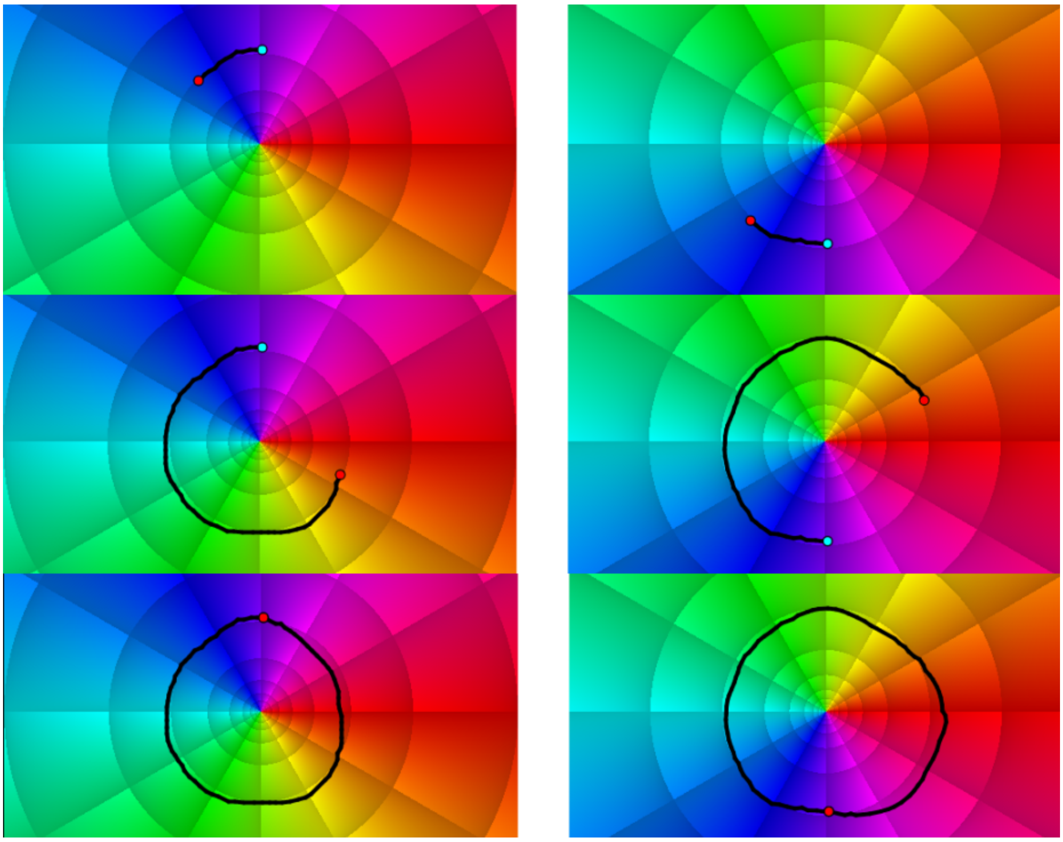


Figure 3.4: Winding directions. Each row is one drawing step

Root Pole Cancellation

Section 3.2 mentioned how this tool does not compute the exact number of roots and poles inside the drawn curve γ, but only the number n_γ, which is the difference between said roots and poles. The reason for this is the hypothesis that in terms of the chromatic number $\mathrm{chrom}_f\,\gamma$, roots and poles of f inside γ cancel each other. This observation can be justified by investigating the example from Figure 3.6. It shows the function $f(z) = \frac{(2z)^2+1}{2z}$ and a path γ which encloses two poles and one root. The mapped path $f \circ \gamma$ has winding number $\mathrm{wind}_\gamma f = 1$, which corresponds to n_γ, the difference between roots and poles.

In this particular example, this concept can be visualized even better. Changing f to $f(z) = \frac{(20z)^2+1}{20z}$, which, intuitively, corresponds to a zoom-out on the function, as shown in Figure 3.7, yields a plot which looks very similar to the identity function on the right hand side of the figure. Thus, in this case the two roots and one pole behave

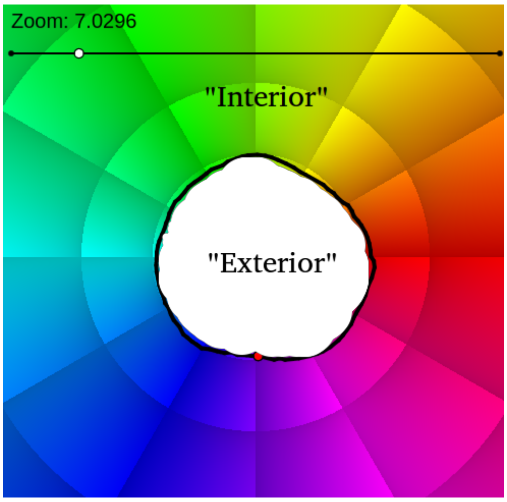


Figure 3.5: Interpretation of the interior of $f \circ \gamma$

just like one root when “viewed from far enough away”. [Wegert 2012, p. 70] mentions this property as well.

3.3 Implementation

This section shall cover the implementation approach to this problem. As mentioned, the implementation focuses on two parts: first, the development of the path $f \circ \gamma$ is visualized in real time while the user drags a path γ on the phase plot of the given function f. Secondly, after γ was closed, information regarding the number of roots and poles inside the path is displayed to the user.

3.3.1 Path Mapping

Point Transformations

Before proceeding to the mapping from path γ on plot one to path $f \circ \gamma$ on plot two, it is necessary to mention that the points P and T live in the coordinate system of the canvas (see Section 2.4 for a definition of the canvas space). Both plots one and two need to show the same plotting range (when the zoom is ignored), which in this case is set to be $[(-1,-1);(1,1)]$. Tests have shown that the implementation is however robust regarding changes of these values and they might be altered by modifying the source code. I use basic affine transformations to map between the positioning of plots one and two on the canvas and their respective plotting ranges.

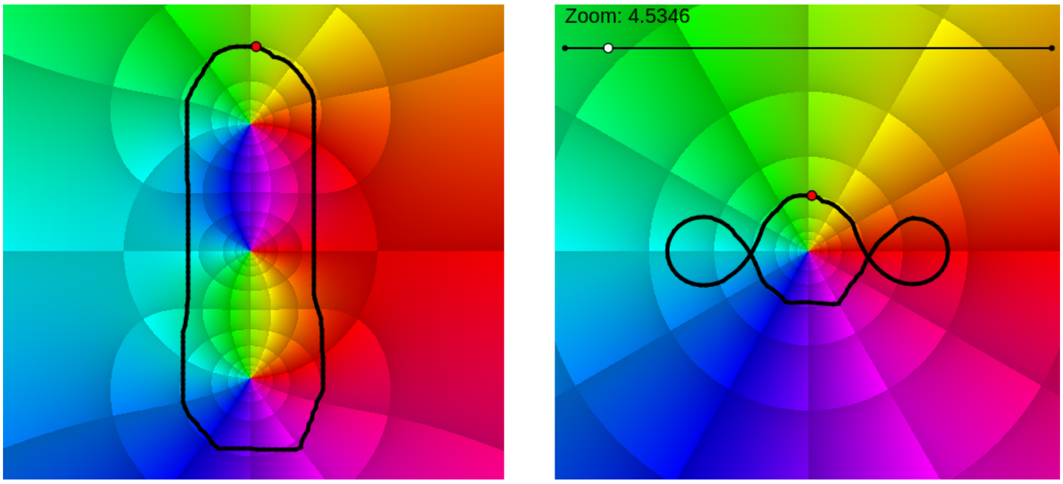


Figure 3.6: In terms of winding, roots and poles cancel each other

In Figure 3.4 the construction of the path $f \circ \gamma$ (right hand side) as a mapping of the path γ on the left hand side of the figure was visualized. The phase plots on both sides represent the same domain (in particular, $[(-1,-1);(1,1)]$). The mapping between paths is defined as applying f to every point $z_0 \in \gamma$. For example, if $f(z) = z^2$ and $z_0 = 0 \in \gamma$, both γ and $f \circ \gamma$ must run through the center of their respective plots. From a programming perspective, this means drawing path segments through the points P_1 and P_2 (which live in canvas space at the position of the center of the respective phase plot). Knowing the position of P_1 (since this point is obtained from the user's mouse position, as they dragged γ on plot one), it is necessary to compute the canvas position of P_2.

For this purpose, let us define the helper function f_P as in Equation (3.1).

$$\begin{aligned} \text{Let X be a canvas point on } D_C \subset \mathbb{R}^2, \text{ the region on top of which plot one lies} \\ f_P : D_C \to \mathbb{R}^2 \\ \text{Let } z_x := \text{the complex number on plot one on which X lies} \\ f_P(X) := \text{the point on top of } f(z_x) \text{ on plot two} \end{aligned} \tag{3.1}$$

Intuitively, this means determining the canvas coordinates of the complex number $f(z_0)$ on plot two from the canvas coordinates of the complex number z_0 on plot one. In Figure 3.4 for instance, the red and green points on the right half represent $f_P(P)$ and $f_P(T)$, respectively. See Section 3.2 for a explanation on what T and P are.

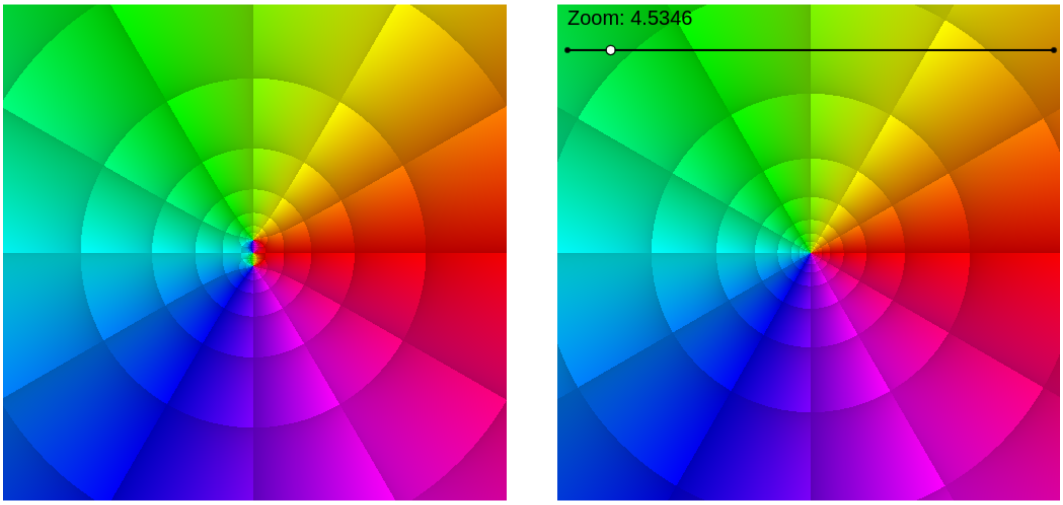


Figure 3.7: Root-pole-cancellation on zoom-out

Example. Let us consider the point X on path γ in the very center of plot one. X might have the coordinates $(2, 2)$ in canvas space. The coordinates on the canvas must not influence the visualization tool. The canvas space might run from $(0, 0)$ to $(100, 100)$, or even from $(-3.14, 2.71)$ to $(42, 42)$. Randomly, we look at an example where the center of plot one corresponds to the canvas point $(2, 2)$. Since the plotting range is $[(-1, -1); (1, 1)]$and X lies in the center of plot one, in order to compute $f_P(X)$ we must not pass the value $z = 2 + 2i$ to f, but $z = 0$. If we were to define f as $f(z) = z^2 + 1$ for instance, we would obtain $z_f := f(z) = 1$. In this example the zoom is ignored, therefore plot two also shows the range $[(-1, -1); (1, 1)]$. Now, considering that plot two is located between $(5, 0)$ (bottom left corner) and $(9, 4)$ (upper right corner) which, as before, is merely a random example, the point $f_P(X)$ must be located at coordinates $(9, 2)$ in order to represent z_f correctly. To solve this problem I defined the corners of each plot as points on the canvas and applied basic linear algebra for the respective transformations. The reader is referred to the source code for the exact implementation.

Path γ on plot one

In the implementation, a path is approximated by a finite number of points which lie on it. Mouse positions are recorded at a given time step until the path has been closed, thus constructing a list of line segments, the first one starting at point T, and the last one ending at point P, which is placed exactly on top of T in case the user has closed the path. On startup, P and T are placed outside the visible area of the canvas. As soon as the user clicks and holds the left mouse button above plot one, a `dragging` flag is set to be true. T is now fixed at the current mouse position, while P is set to follow the cursor. P may not move outside plot one. Therefore, its $x-$ and $y-$coordinates are limited to the plot bounds.

While `dragging` is true, samples of P's current position at discrete time steps (that is at most once every deltaT seconds) are collected. If the user holds the point P still for some time, it is inefficient to record the same position multiple times. Therefore, the parameter deltaD defines the minimum distance (on the canvas) between two successive sampling points. This yields the following procedure:

```
if (dragging,
  // Record the current system time.
  now = seconds();
  // Check time and space conditions as described above.
  if (now >= nextT & |lastPCurve - P| >= deltaD,
    nextT = now + deltaT;
    // Append a tiny segment to the path.
    segments = append(segments, [clone(lastPCurve), clone(P)]);
    // Record P’s position for the next sampling step.
    lastPCurve = clone(P);
  );
  drawall(segments, color->[0,0,0], size->4);
);
```

This procedure, produces a list of line segments which are short enough to appear like a curve (γ) when drawn one after another. The first segment always starts at T's position, while the last segment ends at P's position.

Path $f \circ \gamma$ on plot two

We have defined the set of points which construct path γ on plot one, as well as a correct way to map canvas points from plot one to plot two using function f_P (Section 3.3.1). It is time to introduce the points $P2$ and $T2$, which appear at the same time as P and T. Using f_P, it is easy to approximate path $f \circ \gamma$ on plot two by applying the mapping f_P on every segment of the approximation of path γ on plot one. $T2$ will be fixed at the position of $f_P(T)$, while $P2$ follows $f_P(P)$. While T and P always lie inside plot one, it is possible for $P2$ to travel outside plot two. One of the simplest examples to reflect this behavior is the function $f(z) = 2z$. Figure 3.8 reflects the construction of a simple curve γ for which the mapping $f \circ \gamma$ exceeds the bounds of plot two. As mentioned in Section 3.2, the user has the option to zoom out in order to view the entire curve $f \circ \gamma$. The next section describes the implementation of to the zoom feature.

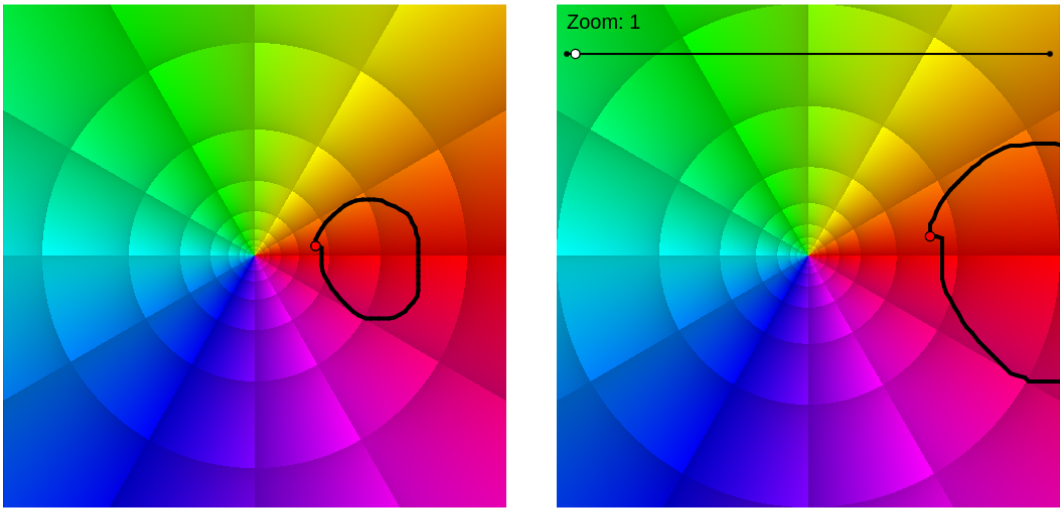


Figure 3.8: $f \circ \gamma$ reaches outside the visible domain of plot two

Zooming on plot two

The approximation of the curve $f \circ \gamma$ consists of the list of line segments. Each of these segments is defined by its two end points, which are expressed in canvas coordinates. The following example describes zooming. Consider the complex number $z = 1$ on plot two. Using the example coordinates from the previous examples, this corresponds to a canvas point at the position $(9, 2)$. A zoom factor of two implies a doubling of the plotting range, which means z must move from the right hand side border of plot two, to the middle of its right half, while maintaining its $y-$coordinate. The position of the points $T2$ (green) and $P2$ (red) before (left) and after a zoom out (right) is depicted in Figure 3.9. The software checks in every frame whether the zoom factor has changed since the past frame, which yields the following computation:

```
// Let X be a point on the curve in plot two.
// Get the complex number corresponding to X from its position on plot 2.
X = trafoFromPlot2ToComplex(X);
// Reverse the old zoom, apply the new zoom;
X = X * zoom / newZoom;
// Transform the newly obtained position on the complex plane
// back to the canvas space above plot 2.
X = trafoFromComplexToPlot2(X);
```

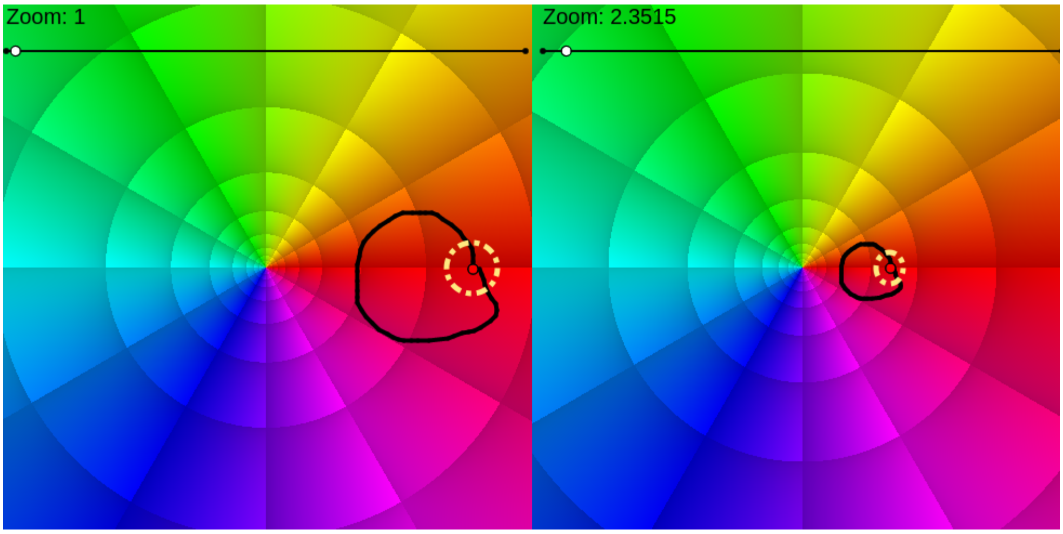


Figure 3.9: Zooming out on plot two

3.3.2 Roots and Poles

As mentioned in Section 3.2, the tool does not display the exact number of roots and poles inside the contour γ, but the difference between the number of roots and the number of poles, which shall be called n_γ. In this implementation n_γ is computed using the chromatic number $\mathrm{chrom}_f\,\gamma$ defined in Section 3.1. This means processing the location of point P in every frame. Color is merely an encoding of the argument of a point on the complex plane. Therefore, the argument of the complex number $f(z_P)$ serves for computing $\mathrm{chrom}_f\,\gamma$, where z_P is the point on the complex plane on plot one on top of which P lies. n_γ is obtained by determining how many times $\arg(f(z_P))$ travels around the origin (i.e. from 0 to 2π or backwards), which is done as follows:

In every frame after a dragging process has started, the tool processes the current position of P, as well as the position of P in the last frame, which we shall call lastP. The difference of argument between $f(z_P)$ and $f(z_{\mathrm{lastP}})$ (the complex numbers on plot one on which P and lastP lie) is the distance which P has traveled around the color circle since the past frame. After applying basic trigonometry to handle the situations where the sign of the argument of P changes from one frame to another, the traveled angle dArg is obtained for each frame and added up until the user closes the path. A distance traveled in the positive direction on the color circle will yield a positive dArg, while the opposite direction a negative one. Thus, roots and poles cancel out each other. After the path has been closed, the totally traveled angle is obtained by adding up all dArg. After dividing by 2π, this yields the desired number n_γ.

3.4 Limitations

Branch Cuts. [Wegert 2012, p. 6] As depicted in Figure 3.10, this implementation does not handle functions with branch cuts in any meaningful way. The function in plot one is $f(z) = \sqrt{z}$. Discussing the meaning of the path γ around zero on the left hand side plot is beyond the scope of this thesis.

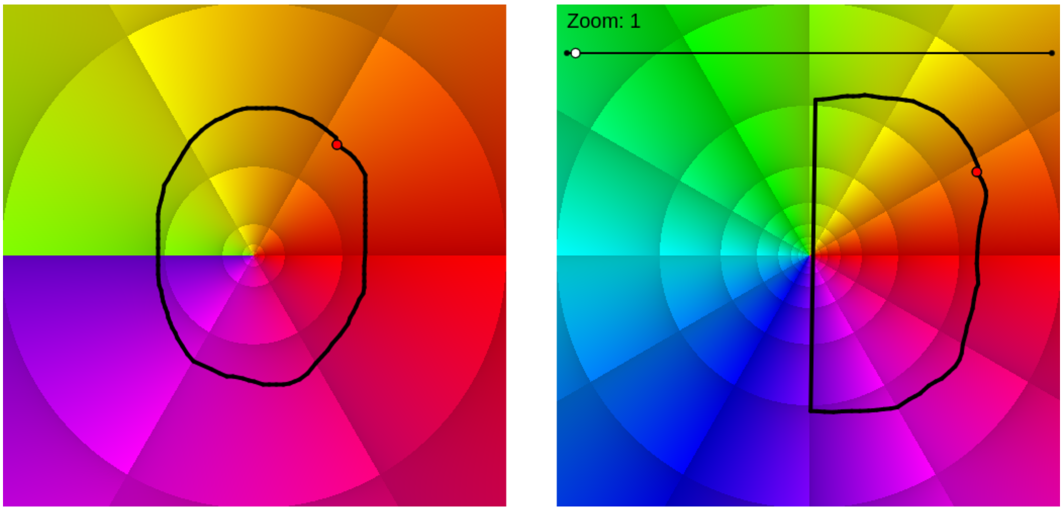


Figure 3.10: Curves passing through branch cuts

Path Direction. After a curve has been completed, there is no way to determine whether it was drawn in a counterclockwise or clockwise direction, which might cause confusion. See Figure 3.10 for an example.

4 Power Series

4.1 Problem Statement

[Wegert 2012, p. 73-] defines complex power series as in Equation (4.1). Let $z_0 \in \mathbb{C}$ be fixed and $(a_n)_{n=0}^{\infty}$ a sequence in $\mathbb{C}$. Partial sums arise as defined in Equation (4.2).

$$\sum_{n=0}^{\infty} a_n (z - z_0)^n \tag{4.1}$$

$$f_k(z) = \sum_{n=0}^{k} a_n (z - z_0)^n \tag{4.2}$$

Let us consider the sequence of $f_k(z)$. If $f_k(z)$ converges for all $z \in D \subseteq \mathbb{C}$, we can use it to define function f from Equation (4.3). In this case, we call D the *disk of convergence*, and it is centered at z_0 on the complex plane. The reader is referred to [Wegert 2012, p. 73-] for a proof.

$$f : D \to \mathbb{C}, f(z) = \sum_{n=0}^{\infty} a_n (z - z_0)^n \tag{4.3}$$

Example. We will now investigate the power series f from Equation (4.4). On the complex unit disk, its values coincide with $\log(z+1)$ [Wegert 2012, p. 77]. Figure 4.1 shows this behavior. The left hand side shows the phase plot of the partial sum $f_{22}(z)$, whereas the right hand side depicts $\log(z+1)$. The unit circle is marked (approximately) by the thick black line on both plots. Until now we were discussing power series of the form in Equation (4.1). This example does not use z_0. Therefore, the disk of convergence of the power series from Equation (4.4) is implicitly centered at 0. This is very limiting and finding a power series to coincide with $\log(z+1)$ in a disk around another point z_1 from Equation (4.4) is not trivial. However, the tool aims to visualize the series expansion of a given function at any point. This problem is covered in the next section.

$$f(z) = \sum_{n=1}^{\infty} (-1)^{n+1} \frac{z^n}{n} \tag{4.4}$$

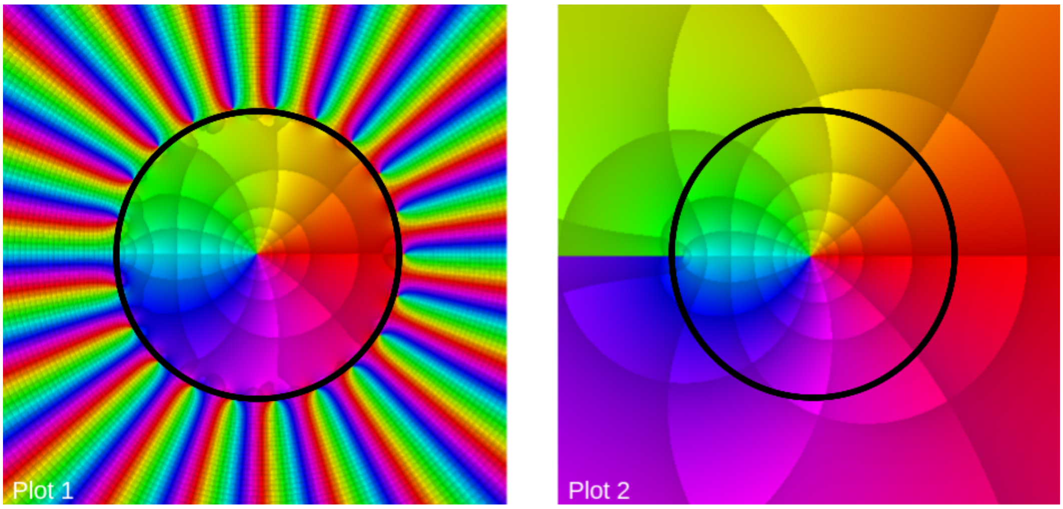


Figure 4.1: Partial sums f_k converge to $\log(z+1)$

4.1.1 Taylor Coefficients

[Wegert 2012, p. 80] discusses how the uniqueness principle of power series implies that the coefficients of a power series which represents a function g around a point z_0 are unique. We refer to said coefficients as Taylor coefficients of g at z_0. In order to overcome the limitation from the example where the visualization was stuck at the disk centered at $z_0 = 0$, there is a procedure to first determine the center z_0 of the disk of convergence, and then compute the Taylor coefficients of the series expansion centered at z_0 using Equation (4.5) [Wegert 2012, p. 140]. Choosing $z_0 = 0$ it can be verified by computation that this yields the same series representation of $\log(z+1)$ as in Equation (4.4).

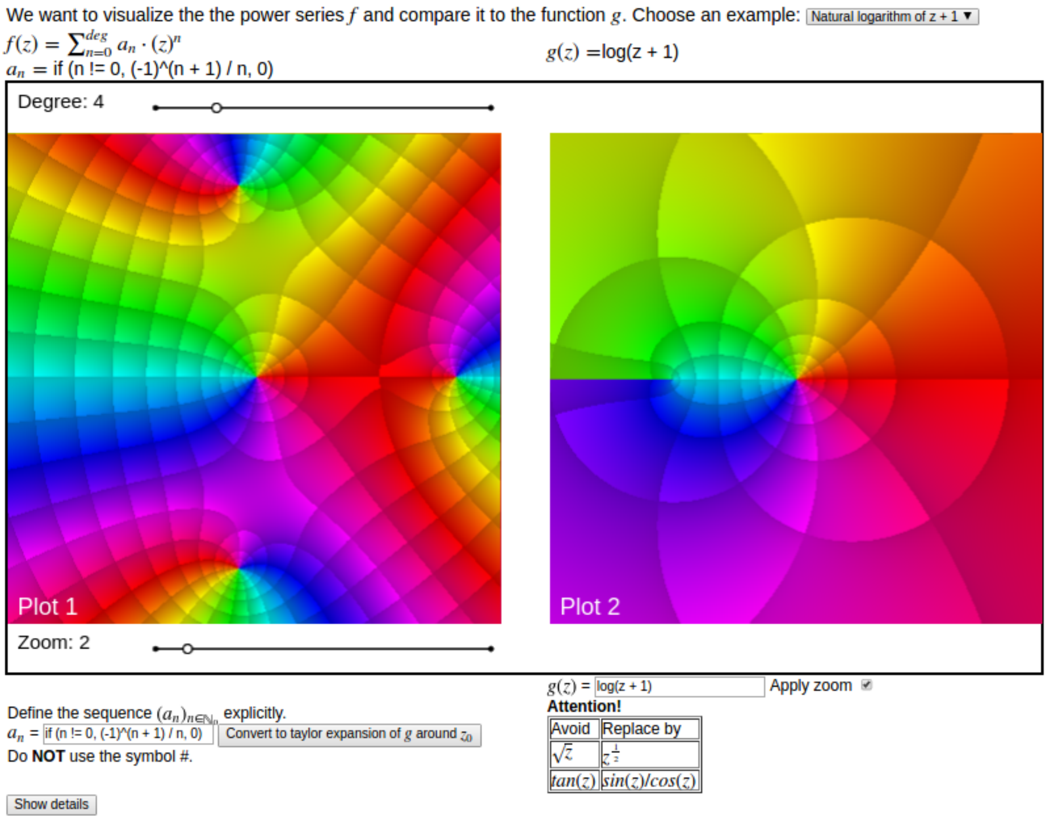
$$a_n = \frac{f^{(n)}(z_0)}{n!} \tag{4.5}$$

The aim of this chapter is to describe a tool which enables the visualization of various partial sums of different degrees alongside arbitrary functions for comparison, analogously to the graphical representation in Figure 4.1. Imagine having to verify whether the identity $\log(z+1) = \sum_{n=0}^{\infty} a_n (z-z_0)^n \forall z : |z| < 1$ holds. Figure 4.1 shows a visualization of this identity.

Having a neat definition of power series for a given function g around a point z_0 as in the example from Equation (4.4) is not always the case. The method from Equation (4.5) is more general and can be applied to arbitrary analytic functions [Wegert 2012, p. 140]. I included this method in the visualization and facilitated the inverse concept: defining a function g and investigating the series expansion given by Equation (4.5) at various points.

Note: In order to avoid confusion, I shall only refer to power series with coefficients computed by Equation (4.5) as Taylor series. I will use the term *explicit power series* for series defined by inputting a different definition of the sequence a_n.

4.2 Evaluation



We want to visualize the the power series f and compare it to the function g. Choose an example: Natural logarithm of z + 1 ▼

$f(z) = \sum_{n=0}^{deg} a_n \cdot (z)^n$

a_n = if (n != 0, (-1)^(n + 1) / n, 0)

$g(z)$ =log(z + 1)

Degree: 4

Zoom: 2

$g(z)$ = log(z + 1) Apply zoom ☑

Attention!

Avoid	Replace by
$\sqrt{z}$	$z^{\frac{1}{2}}$
$tan(z)$	$sin(z)/cos(z)$

Define the sequence $(a_n)_{n\in\mathbb{N}_0}$ explicitly.

a_n = if (n != 0, (-1)^(n + 1) / n, 0)

Do **NOT** use the symbol #.

Figure 4.2: Tool for visualizing power series

Figure 4.2 shows a part of the visualization tool. The user is provided with two phase plots. The one on the left hand side shall be referred to as *plot one* and the other one as *plot two*. In all figures the plots will be marked with their respective name (bottom left corner) in order to avoid confusion, because some figures might show only one of them. This information is not present in the actual tool. Plot one shows the phase of the partial sum f of a power series whereas plot two represents an arbitrary function g.

The partial sum f is defined as described in Section 4.1. Firstly, the user can select various examples from the drop down menu in the upper right corner, i.a. the example from Equation (4.4), which is also depicted in Figure 4.2. This corresponds to the first part of the tool: visualizing the convergence of the sequence of partial sums f for a given power series. The slider above the plots is useful for this purpose, as it increases the degree of the partial sum, thus visualizing the disk of convergence. The partial sum f is depicted for different degrees in Figure 4.3. One interesting observation is that the roots of f seem to cluster along a circle. This is the boundary of the disk of convergence. The reader is referred to [Wegert 2012, p. 78-] for details on this phenomenon.

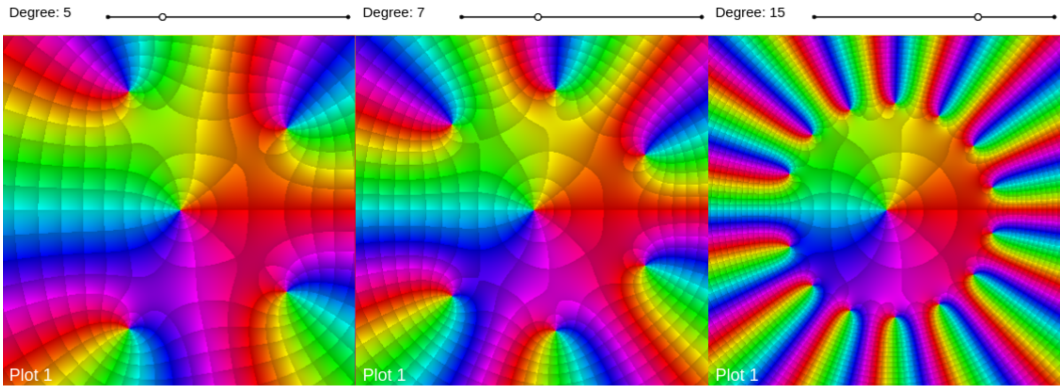
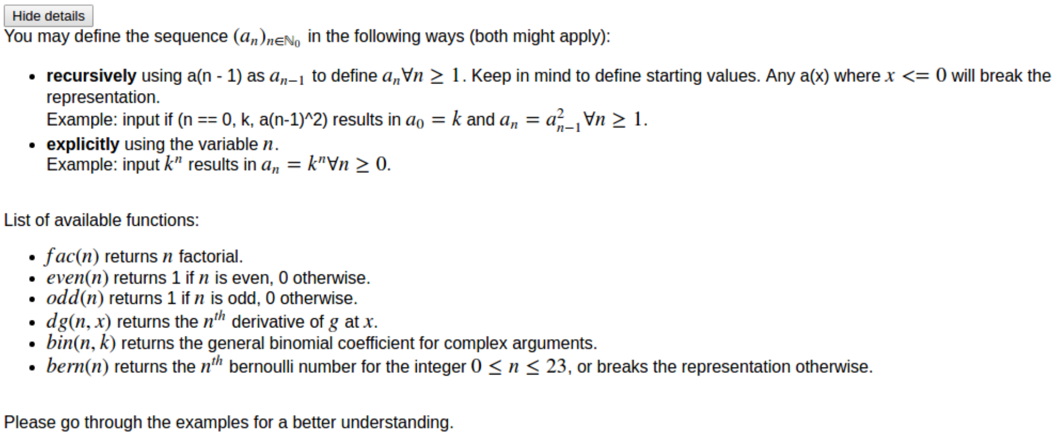


Figure 4.3: Three partial sums with increasing (from left to right) degrees

Since the tool provides only nine examples of selected power series expansions, the user has the option to define the series a_n as well es the function g in an arbitrary way using the input fields below the plots. The tools which are available for the definition of a_n are hidden in Figure 4.2. Figure 4.4 shows explanations of the available functions and variables. These are hidden on load.

For the second part of the visualization, i.e. expanding function g around an arbitrary point z_0, the user can use the button on the right side of the input field for a_n, which replaces the current definition of a_n with the expression `dg(n,z0)/fac(n)`, which is the definition of the Taylor coefficients as given in Equation (4.5). At the same time the point z_0 appears in the center of plot one. It is the expansion point for the series representation. Figure 4.5 shows an example of the series expansion of the function $g(z) = \log(z+1)$ for different values of z_0.



You may define the sequence $(a_n)_{n\in\mathbb{N}_0}$ in the following ways (both might apply):

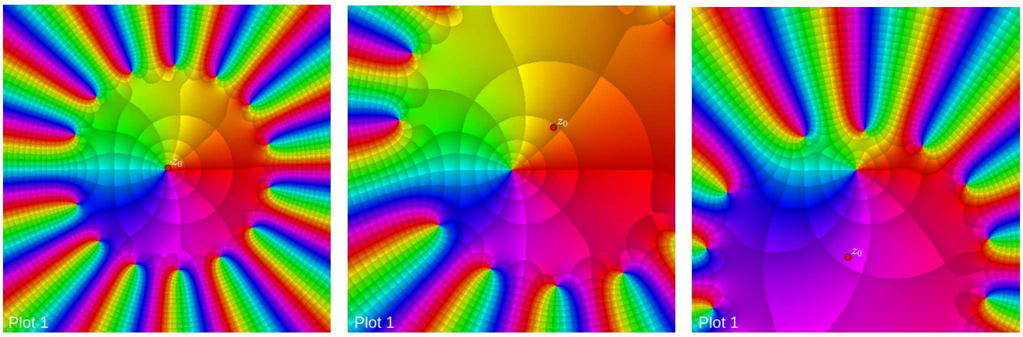
- **recursively** using a(n - 1) as a_{n-1} to define $a_n \forall n \geq 1$. Keep in mind to define starting values. Any a(x) where $x <= 0$ will break the representation.
  Example: input if (n == 0, k, a(n-1)^2) results in $a_0 = k$ and $a_n = a_{n-1}^2 \forall n \geq 1$.
- **explicitly** using the variable n.
  Example: input k^n results in $a_n = k^n \forall n \geq 0$.

List of available functions:

- $fac(n)$ returns n factorial.
- $even(n)$ returns 1 if n is even, 0 otherwise.
- $odd(n)$ returns 1 if n is odd, 0 otherwise.
- $dg(n, x)$ returns the n^{th} derivative of g at x.
- $bin(n, k)$ returns the general binomial coefficient for complex arguments.
- $bern(n)$ returns the n^{th} bernoulli number for the integer $0 \leq n \leq 23$, or breaks the representation otherwise.

Please go through the examples for a better understanding.

Figure 4.4: Details on the usage of the tool



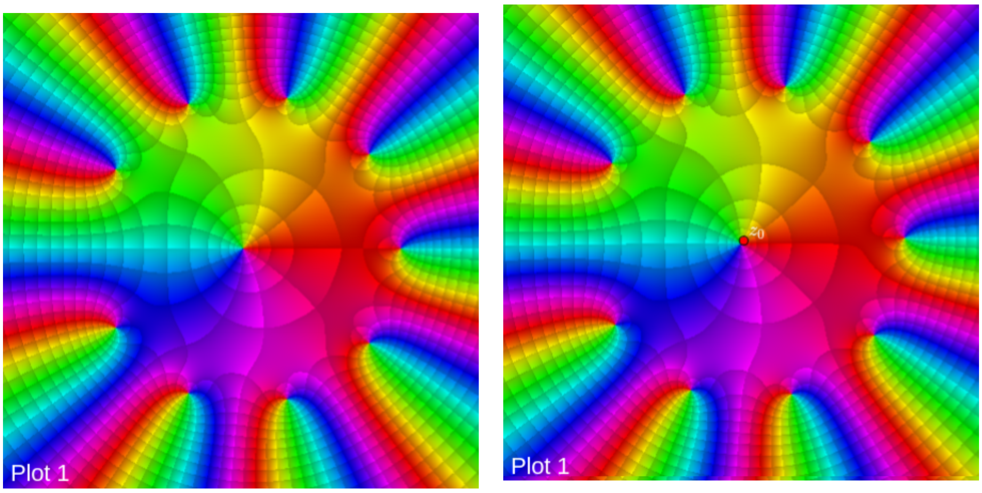
(a) $z_0 \approx 0$ (b) $z_0 \approx 0.5 + 0.5i$ (c) $z_0 \approx -i$

Figure 4.5: Series expansion of $g(z) = \log(z + 1)$ around different points

4.2.1 Selected Examples

Uniqueness Principle

Section 4.1.1 presented the uniqueness principle for complex power series. In order to visualize this concept, we shall go back to the example from Section 4.1, where $f(z) = \sum_{n=1}^{\infty}(-1)^{n+1}\frac{z^n}{n}$ which converges to $\log(z+1)$ for $|z| \leq 1$. We want to compare this series representation of $\log(z+1)$ with the Taylor series obtained from Equation (4.5). Figure 4.6a shows an approximation of f by the partial sum of degree 10 using the first series representation, whereas Figure 4.6b approximates the second representation using the same degree. It is obvious that they are almost identical. This is a visualization of the uniqueness principle.



(a) Partial sum $f(z) = \sum_{n=1}^{10}(-1)^{n+1}\frac{z^n}{n}$ (b) Partial sum $f(z) = \sum_{n=1}^{10}\frac{g^{(n)}(0)}{n!} \cdot z^n$

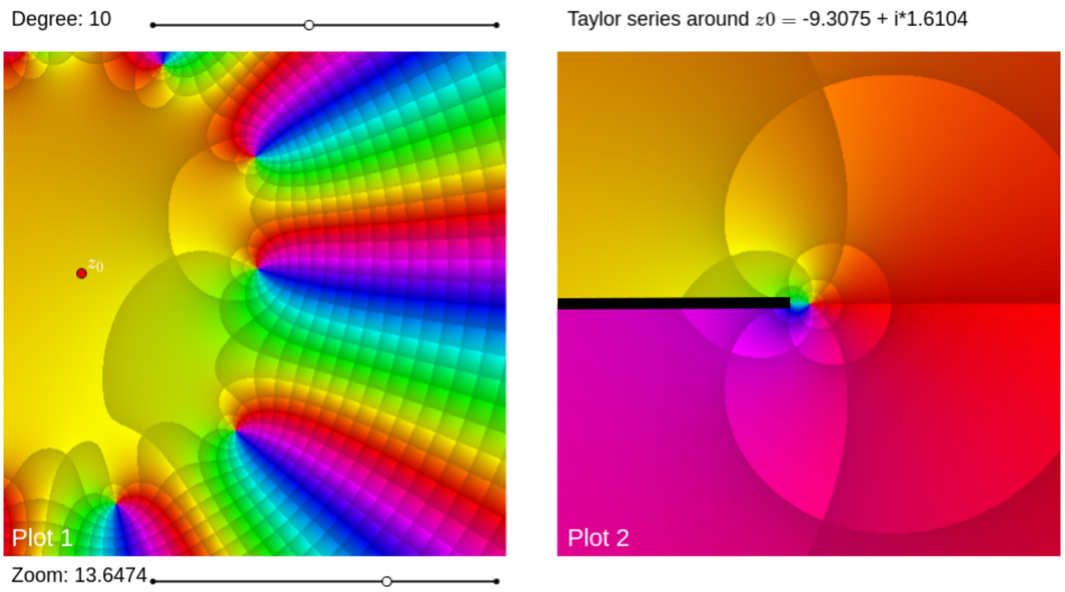
Figure 4.6: Comparison between two series expansions of $g(z) = \log(z+1)$ around $z_0 \approx 0$

Branch cuts

Figure 4.7 emphasizes the branch cut on the function $g(z) = \log(z+1)$ with the thick black line on plot two. Imagine we could look past this branch cut to see what values of g would follow if there were no discontinuity at that position [Wegert 2012, p. 6]. Using the Taylor series representation I make an attempt at this problem. Plot one on Figure 4.7 shows what happens if one expands g at a point close to the branch cut. A part of the plot, which had not been visualized before, was "discovered". This principle is a step towards analytic continuation, which is beyond the scope of this thesis. The reader is referred to [Wegert 2012, p. 117] for a deep illustration of this subject.

4.3 Implementation

In the implementation of this tool the power series $\sum_{n=0}^{\infty} a_n \cdot (z-z_0)^n$ is approximated by partial sums of the form $f(z) = \sum_{n=0}^{\text{deg}} a_n \cdot (z-z_0)^n$. In order to avoid numerical issues when computing the powers z^n, Horner's Method as described in [Newbery 1974] is applied. The coefficients a_n are precomputed on every relevant input change. Thus, it is trivial to render f and g.



Degree: 10

Taylor series around $z0$ = -9.3075 + i*1.6104

Zoom: 13.6474

Figure 4.7: Series representation of $g(z) = \log(z+1)$ around a point near a branch cut

4.3.1 Helper Functions

In order to expand the functionality of the tool, several helper functions are provided to the user in order to define the sequence a_n. The non-trivial functions are `bin(n,k)` and `bern(n)`.

`bin(n,k)` returns the generalized binomial coefficient for complex arguments given by Equation (4.6) [Weisstein 2003a]. Here, the Lanczos Approximation yields an approximation of Γ [Weisstein 2002c]. Please note that this is merely an attempt of approximating the generalized binomial coefficient. Testing the accuracy of the implementation is beyond the scope of this thesis.

$$\binom{n}{k} = \frac{\Gamma(n+1)}{\Gamma(k+1)\Gamma(n-k+1)} \tag{4.6}$$

`bern(n)` [Wegert 2012, p. 89] discusses the function $g(z) = \frac{z}{e^z-1}$ which is the generating function for the sequence B_n from Equation (4.7) [Weisstein 2002a]. These numbers are referred to as the Bernoulli numbers [Wegert 2012, p. 90]. g is included into the set of example functions with their respective series representation. Therefore, the first 23 Bernoulli numbers are hard-coded for the sake of the implementation's simplicity. An algorithm to approximate them is beyond the scope of this thesis.

$$\frac{z}{e^z - 1} = \sum_{n=0}^{\infty} \frac{B_n}{n!} z^n \tag{4.7}$$

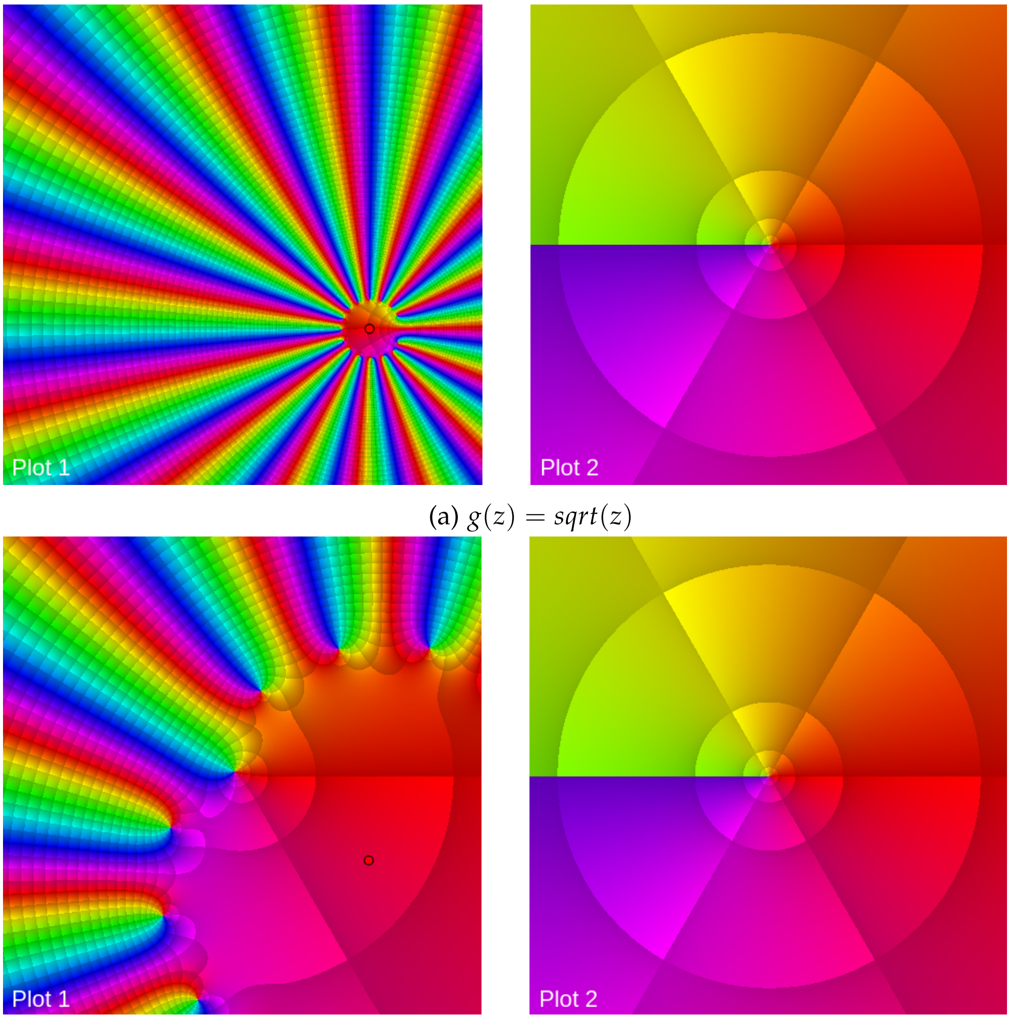
4.4 Limitations

User Interface. One major downside of this implementation is the complicated usage. Ideally, the user would comprehend the tool at first sight and be able to use parts of it without being distracted by the more complex features. I try to accomplish this through the examples provided in the drop down menu in the upper right corner of the interface (see Figure 4.2). The desired workflow is to first select several examples, manipulate the sliders to watch the convergence, and only then try to define more complex partial sums. This is also the reason why the information regarding available functions from Figure 4.4 is hidden on load.

Problematic Functions. As seen in Figure 4.2 in the bottom right corner, some functions are to be avoided in order to ensure a correct visualization. I use automatic differentiation [Griewank and Walther 2003] to realize $g^{(n)}(z)$, as used in the Taylor coefficients (Equation (4.5)). In the current version of CindyJS, the functions `sqrt` and `tan` produce complications. Figure 4.8 shows the behavior if one tries to represent $g(z) = \sqrt{(z)}$ as a Taylor series in comparison to $g(z) = z^{\frac{1}{2}}$. It is quite possible that other functions which are not listed in the warning are problematic as well. It is beyond the scope of this thesis to analyze which functions are affected and to work around them.

Degree Limitation. At the time being the degree of the partial sum f is limited to 23. The reason for this are the hard-coded Bernoulli numbers as described in Section 4.3.1. An algorithm to approximate Bernoulli numbers up to a higher degree would eliminate the problem. It is uncertain whether in this case the Bernoulli numbers are useful enough to outweigh the degree limitation. However, a maximum degree of 23 is reasonable to visualize power series convergence and an even higher degree might not produce noticeably better results.

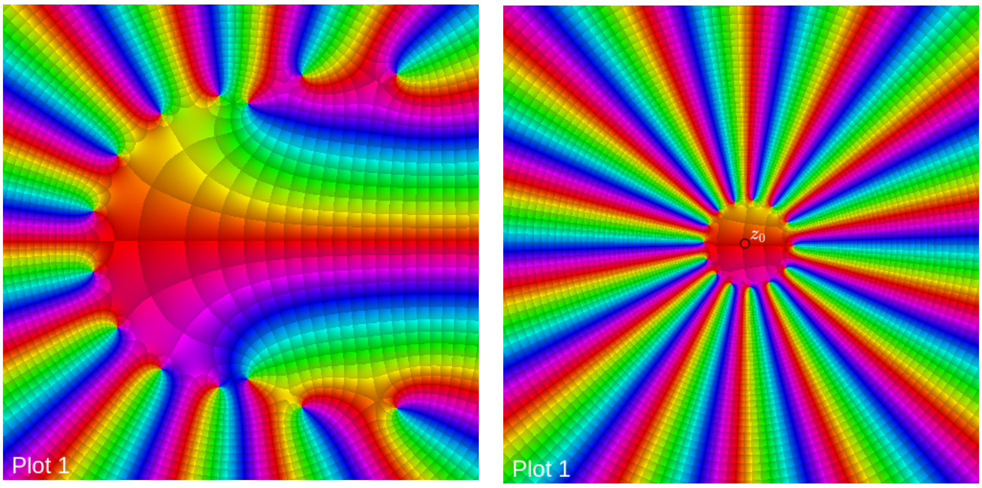
Taylor Series Expansion Point. When switching to a Taylor series representation for the function g, the point z_0 appears. This is described in more detail in Section 4.2. When this happens, z_0 should theoretically be set to 0. However, this value might break the visualization on certain functions, for example $g(z) = \frac{e^z-1}{z}$. Therefore, a tiny offset is added to z_0 which, ideally, should not have any visual effects. That is, after the switch from an explicit power series representation to a Taylor series, there should not be any noticeable changes. However, some functions are problematic from this point of view. Figure 4.9 shows the issues which are present for $g(z) = \frac{e^z-1}{z}$.



(a) $g(z) = sqrt(z)$

(b) $g(z) = z^{\frac{1}{2}}$

Figure 4.8: Example on why to avoid using `sqrt`



(a) Explicit partial sum $f(z) = \sum_{n=0}^{14} \frac{1}{(n+1)!} z^n$

(b) Taylor expansion of $g(z) = \frac{e^z - 1}{z}$ around $z_0 \approx 0.1 + 0.1i$

Figure 4.9: Issues when switching to the Taylor series of $g(z) = \frac{e^z - 1}{z}$

5 The Riemann Sphere

5.1 Problem Statement

[Anderson 1999, p. 7-] defines stereographic projection as a the bijection ϕ between a circle $\mathbb{S}^1$ in the complex plane and the real line $\mathbb{R}$ (see Equation (5.1) and Equation (5.2)). $\mathrm{re}(z)$ and $\mathrm{im}(z)$ denote the real and imaginary parts of z, respectively). Note that stereographic projection makes sense for various definitions of $\mathbb{S}^1$. In this example, $\mathbb{S}^1$ is considered to be the circle around i with radius 1.

$$\mathbb{S}^1 := \{\, z \in \mathbb{C} \mid (\mathrm{re}(z))^2 + (\mathrm{im}(z) - 1)^2 = 1 \,\} \tag{5.1}$$

$$\phi : \mathbb{S}^1 \setminus \{2i\} \to \mathbb{R} \tag{5.2}$$

Let K_z be the line between the points $2i$ and z. The function ϕ is defined as $\phi(z) = \mathbb{R} \cap K_z$, the intersection between K_z and the real line. Here it becomes obvious why ϕ is defined on $\mathbb{S}^1 \setminus \{2i\}$. Following the terminology form [Anderson 1999, p. 7-], the point where the bijection is not defined shall be referred to as the “north pole”.

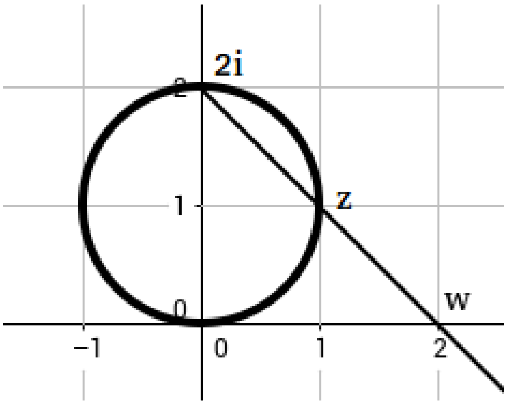


Figure 5.1: Example of stereographic projection. Realized with [Hohenwarter, Borcherds, Ancsin, et al. 2017]

Figure 5.1 shows an example, where $\phi(z) = w$. One way of interpreting this is that the circle $\mathbb{S}^1$ has been constructed from the real line $\mathbb{R}$ by adding a single point which is not contained in $\mathbb{R}$ ($2i$ in this case)[Anderson 1999, p. 8-]. Moreover, [Anderson 1999, p. 8-] applies the same procedure to the complex plane $\mathbb{C}$, thus producing the Riemann sphere $\overline{\mathbb{C}}$.

$$\overline{\mathbb{C}} = \mathbb{C} \cup \{\infty\} \tag{5.3}$$

The generalization of Figure 5.1 to the two dimensional case can be realized by associating the complex plane $\mathbb{C}$ with the $xz-$plane in $\mathbb{R}^3$ and the complex number $a = x + zi$ with the point $(x, 0, z)$. Now, following the notation by [Anderson 1999, p. 11-], we define $\mathbb{S}^2$ as the sphere of radius 1 around the point $(0, 1, 0)$ by Equation (5.4) [Glassner 1989, p. 36].

$$\mathbb{S}^2 := \{\, (x, y, z) \in R^3 \mid x^2 + (y-1)^2 + z^2 = 1 \,\} \tag{5.4}$$

In order to represent $\overline{\mathbb{C}}$, the bijection Φ must be defined for the two dimensional case. First, let the point $N = (0, 2, 0)$ be the north pole. Define L_P as the line passing through the point $P \in \mathbb{S}^2$ and the north pole N.

$$\Phi : \mathbb{S}^2 \setminus \{N\} \to \mathbb{C} \tag{5.5}$$

$$\Phi(P) = L_P \cap \mathbb{C} \tag{5.6}$$

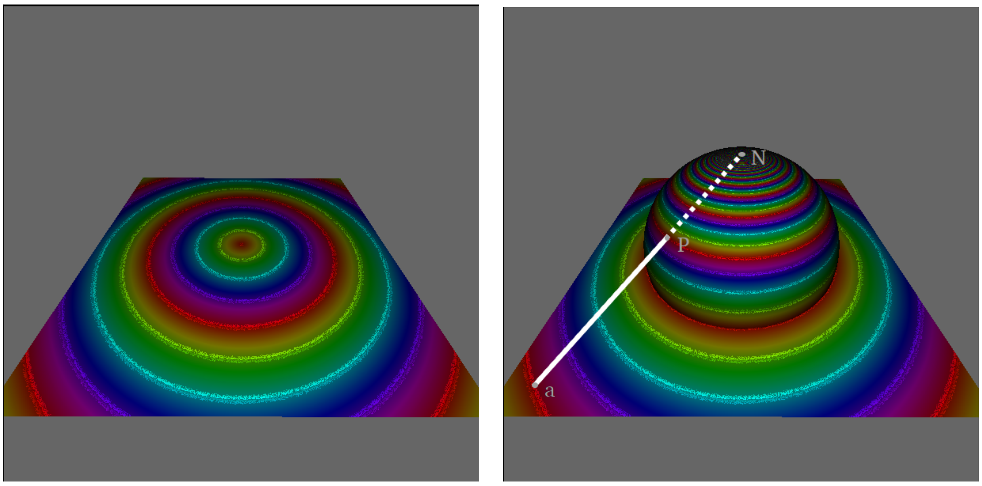
Having defined the sphere $\mathbb{S}^2$ and a bijection Φ which associates every point $P \in \mathbb{S}^2$ with a complex number $a \in \mathbb{C}$. In order to represent $\overline{\mathbb{C}}$, the point ∞ is still required. According to [Wegert 2012, p. 20-], the point N corresponds to ∞ in $\mathbb{C}$, because if a point $P \in \mathbb{S}^2$ approaches N, the absolute value of $a = \Phi(P)$ (i.e. the distance between a and the origin) grows arbitrarily large (see Section 5.2.1 for a visualization). Thus, the stereographic projection defined by $\mathbb{S}^2$ and Φ is a representation of the Riemann sphere $\overline{\mathbb{C}}$.

Figure 5.2 shows a visualization of this process. Figure 5.2a displays a phase plot of the function $f(z) = e^{i|z|5}$ on the complex plane (viewed from above in a 3D space). In Figure 5.2b a sphere is centered at the origin. The white line is an example of the projection of the point P on the sphere onto the complex number a.

In the following sections, the visualization tool which produced Figure 5.2 shall be described.

5.2 Evaluation

The visualization tool is depicted in Figure 5.3. The layout is intended to be self-explanatory and to incentivize the user to explore the functionality from coarse to fine. The sliders on the right hand side concern the most simple functions of the



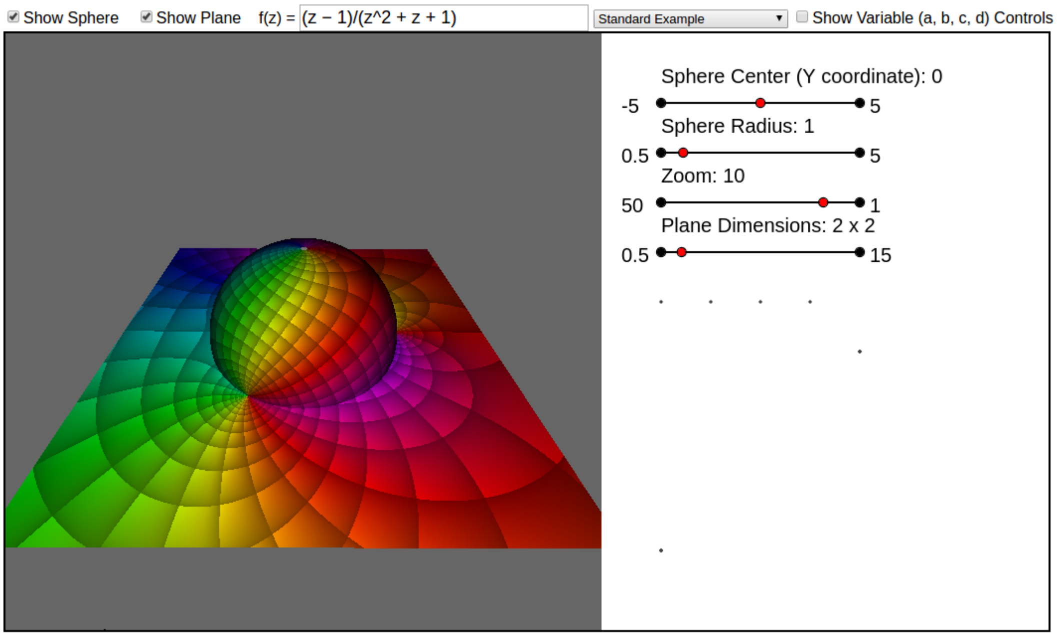
(a) Complex plane

(b) Complex plane with Riemann sphere

Figure 5.2: Stereographic projection: the Riemann sphere

tool. Regarding the $Y-$coordinate of the sphere center, please note that stereographic projection is defined even if the sphere is below the complex plane. Intersections between the line L_P and the complex plane (see the definition of stereographic projection in Section 5.1) is still defined and unique. The north pole is on top of the sphere (where the word "top" refers to the positive y-axis), depicted by a tiny white dot.

The function to be visualized can be altered using the input field on the top. However, it makes more sense to explore the tool using the drop down menu on its right side. While doing so, the user will notice the sudden appearance of the variables a and b. This is intended to attract the attention to the rightmost checkbox, where the user can opt to view a more complicated piece of the UI. As shown in Figure 5.4, the newly appeared variable box allows the altering of the variables a, b, c and d, which the user might use in the expression for $f(z)$. The value of each variable is determined by the positions of the points A, B, C and D on the square area, which resembles a section of the complex plane. Another option would be to allow inputting explicit values for the four variables in text fields, which would make the tool more powerful. However, this mechanism is intended to be more dynamic, because I believe it is more enjoyable to drag a variable and watch the plot reflect the changes in real time. This element is hidden at first, since it might overcomplicate the visualization tool at first sight.



☑ Show Sphere ☑ Show Plane f(z) = (z − 1)/(z^2 + z + 1) Standard Example □ Show Variable (a, b, c, d) Controls

Sphere Center (Y coordinate): 0
-5 5
Sphere Radius: 1
0.5 5
Zoom: 10
50 1
Plane Dimensions: 2 x 2
0.5 15

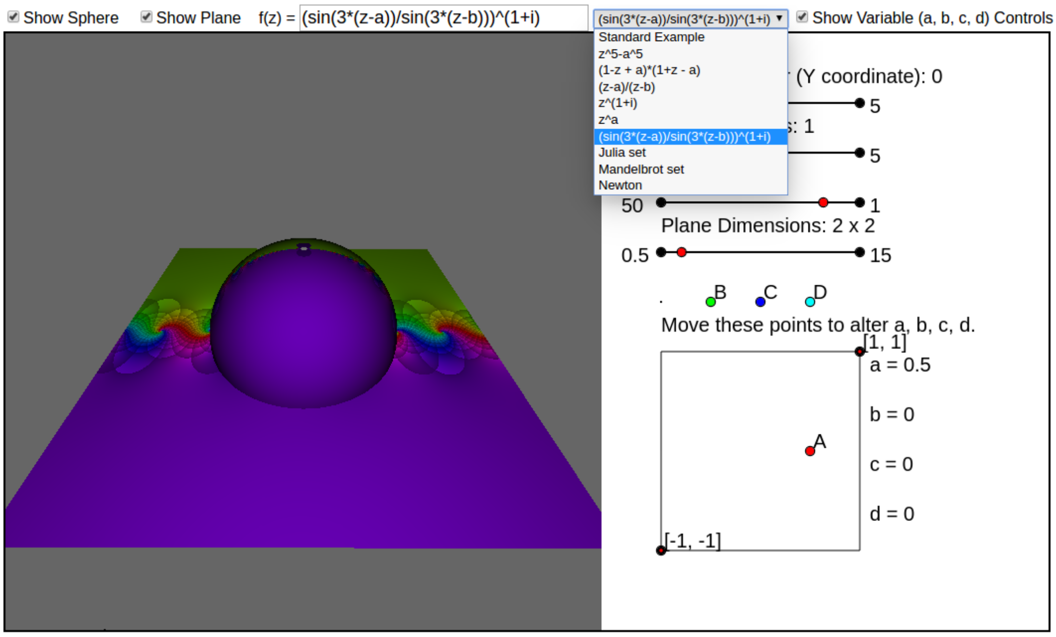
Figure 5.3: Riemann sphere visualization tool

5.2.1 Selected examples

One advantage of this representation of the Riemann sphere is the option to "look towards infinity". The following two examples make use of this property.

The north pole has the role of ∞. [Wegert 2012, p. 20-] points out that since the distance between the complex numbers z_1 and z_2 on the complex plane grows arbitrarily large as the equidistant points P_1 and P_2 (where $z_i = \Phi(P_i)$, i.e. z_i is the projection of P_i) on the sphere approach the north pole N, the latter can be considered to represent ∞. This principle is visualized in Figure 5.5. The construct depicts the function $f(z) = \exp(5z)$, which is characterized by parallel isochromatic lines on the complex plane. Looking at the red lines on the sphere, where Figure 5.5 is marked with the dotted line, the distance between two red lines gets smaller and smaller as the dotted line approaches the north pole N. Since the distance between the projection of these lines onto the complex plane is constant, this observation substantiates the first statement from this paragraph.

Fundamental Theorem of Algebra. Figure 5.6 depicts the visualization of the function $f(z) = z^2$. Following [Wegert 2012, p. 67], it is trivial to observe that every



☑ Show Sphere ☑ Show Plane f(z) = (sin(3*(z-a))/sin(3*(z-b)))^(1+i)

☑ Show Variable (a, b, c, d) Controls

(Y coordinate): 0

Plane Dimensions: 2 x 2

Move these points to alter a, b, c, d.

a = 0.5

b = 0

c = 0

d = 0

Figure 5.4: Riemann sphere visualization tool. Advanced

isochromatic line emerging from a root of the polynomial ends up at infinity. An example is marked in Figure 5.6 by the dotted black line on the cyan colored line. On the left side of the figure, the sphere is hidden in order to visualize the phase plot of the function f inside the unit circle (which would otherwise be covered by the sphere). There, the cyan line starts from a root of f (which is zero in this case) and continues southwards. The dotted black line on the right-hand side of Figure 5.6 marks the projection of the continuation of the cyan line outside the unit circle onto the Riemann sphere, where it approaches the only pole of f: ∞. From the reversal of this principle (every line emerging from a pole ends up at a root), [Wegert 2012, p. 67] points out that in cases like this one, the number of isochromatic lines, which emerge from a pole is equal to the number of roots (and thus the degree of this polynomial by the fundamental theorem of algebra [Weisstein 2003b]). A generalization of this statement is beyond the scope of this thesis.

5.3 Implementation

The tool was implemented using ray tracing. [Glassner 1989, p. 5-] describe ray tracing as a rendering technique which follows the principle that the images humans see

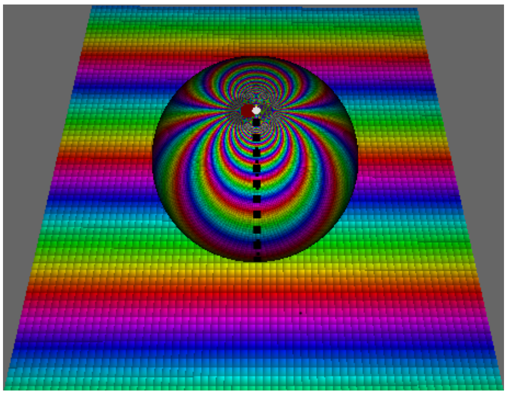


Figure 5.5: Visualizing ∞ on $f(z) = \exp(5z)$

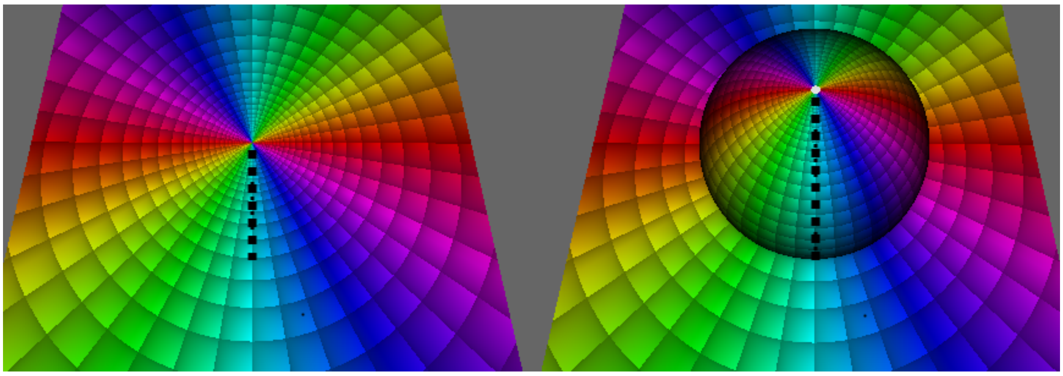


Figure 5.6: Visualizing the fundamental theorem of algebra

are composed of light which is emitted from a light source and reflected by various surfaces which form the scene. From a computational perspective, the inverse situation is modeled. A point in the 3D scene is chosen as a point of view (from now on referred to as the camera center C) [Glassner 1989, p. 35]. Starting from this point, rays are shot into the 3D scene in an attempt to compute a color for each one [Glassner 1989, p. 4-]. Since only a discrete 2D screen consisting of pixels needs to be colored, the situation is modeled by placing a discrete 2D grid G inside the 3D scene. One ray is shot from C through each point on the grid.

Figure 5.7 shows an example of ray tracing on a 3D scene. The Ray which originates from C and travels through the point M on G hits the 3D scene (consisting of 3 spheres in this case) in the four intersection points I_1, I_2, I_3 and I_4. Assuming that the spheres are not transparent, the pixel corresponding to M has to reflect the intersection I_1. Thus, the ray-scene intersection which is closest to M has to be found, as described by

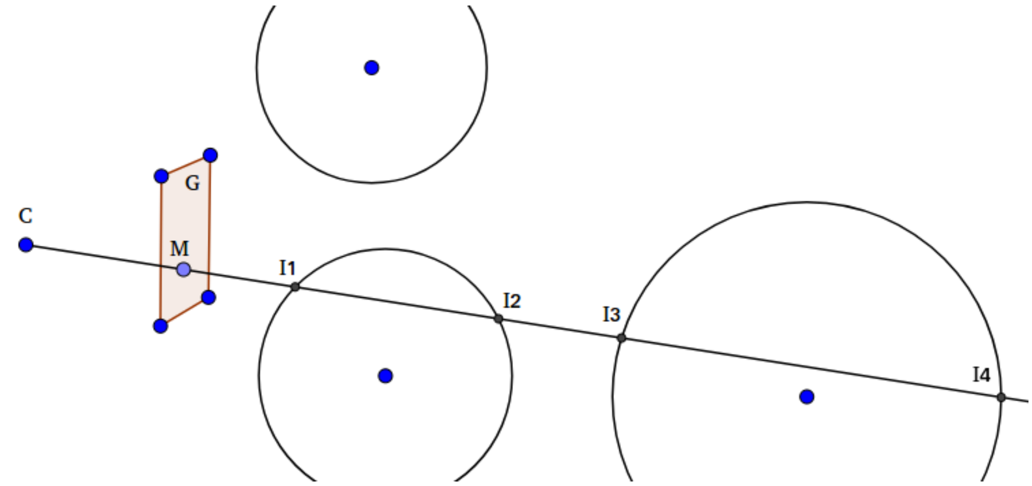


Figure 5.7: Ray Tracing Example. Realized with [Hohenwarter, Borcherds, Ancsin, et al. 2017]

[Glassner 1989, p. 35-], who use the following representation for this purpose:

$$D := (M - C)/|M - C| \tag{5.7}$$

$$r(t) := C + t \cdot D \tag{5.8}$$

In Equation (5.7), D represents the direction of the ray. Each point I on the ray r can be expressed using a scalar t and Equation (5.8). Intuitively, t describes how far to travel from C along direction D to hit point I [Glassner 1989, p. 35]. Since Equation (5.9) holds for the example from Figure 5.7, the goal is to find the smallest (positive) t_i [Glassner 1989, p. 35-].

$$\forall i \in \{1, 2, 3, 4\} : \exists t_i : I_i = C + t_i \cdot D \tag{5.9}$$

5.3.1 Camera parameters

The goal of the implementation is to render a 2D image of the Riemann sphere embedded in a 3D scene. The reader may imagine this process as coloring each pixel of a 2D grid, which functions as a window to the 3D scene. The coordinate system on this grid shall be referred to as the canvas space, whereas the 3D scene lives in $\mathbb{R}^3$ which shall also be referred to as the world.

This implementation defines the principal point P to be at the center of G following [Hartley and Zisserman 2003]. Specifically, P is fixed and each point M on G is merely $P + \delta$ where δ is some offset. The camera appears at position C. The direction vectors dirRight and dirUp define the current orientation of the camera. They are used to translate the canvas coordinates to world coordinates. While coloring the canvas, the algorithm traverses each pixel with canvas coordinates (x, y) and computes a world point M on the grid G corresponding to the canvas coordinates (x, y). The following pseudocode illustrates this procedure.

```
// Let CEN be the canvas point corresponding to the center of the colored region.
for each point P which needs to be colored {
  coords = (#.x, #.y) - CEN;
  // Translate from canvas space to world space.
  // Subtract C to obtain the ray direction.
  D = P + coords.x * dirRight + coords.y * dirUp - C;
  // Normalize D.
  D = D/|D|;
  // Compute the color at the corresponding pixel.
  getColor(D);
}
```

Ray-scene intersections

The scene from Figure 5.2 consists of a plane and a sphere. For every ray, all intersections with these two objects need to be computed in order to find the one which is closest to C.

The plane Equation (5.10) [Glassner 1989, p. 51] is used to define the complex plane. As described in Section 5.1, it corresponds to the $(x, z)-$plane in $\mathbb{R}^3$. Solving Equation (5.10) for t after inputting the $x-$, $y-$ and $z-$coordinates of $I = C + t \cdot D$, yields the distance between C and the complex plane on that specific direction D. After the ray-plane intersection has been computed, the respective pixel needs to be colored using the same procedure as in Section 2.2. The plane is merely a 2D phase plot in a 3D environment.

$$ax + by + cz + d = 0 \tag{5.10}$$

$$(x - x_K)^2 + (y - y_K)^2 + (z - z_K)^2 = r \tag{5.11}$$

An analogous procedure is applied to compute the intersection between the ray and the sphere. Let $K = (x_K, y_K, z_K)$ be the sphere center. The sphere Equation (5.11) [Glassner 1989, p. 36] yields either one or two values for t, or none at all. If there are real values for t which satisfy the equation (i.e. if there is a ray-sphere intersection for that particular D), a color for the intersection point $I = C + t \cdot D$ (for the smallest positive value of t) on the sphere needs to be computed. Applying the principles of stereographic projection described in Section 5.1, it is obtained by finding the intersection point between the line which goes through the north pole N of the sphere and I, and the complex plane.

The final color of a pixel is obtained by choosing the smaller positive of the values t_1 and t_2 which are obtained by the ray-plane, and the ray-sphere intersection, respectively. Thus, the 2D image of the 3D scene is produced by applying basic geometry.

Lighting

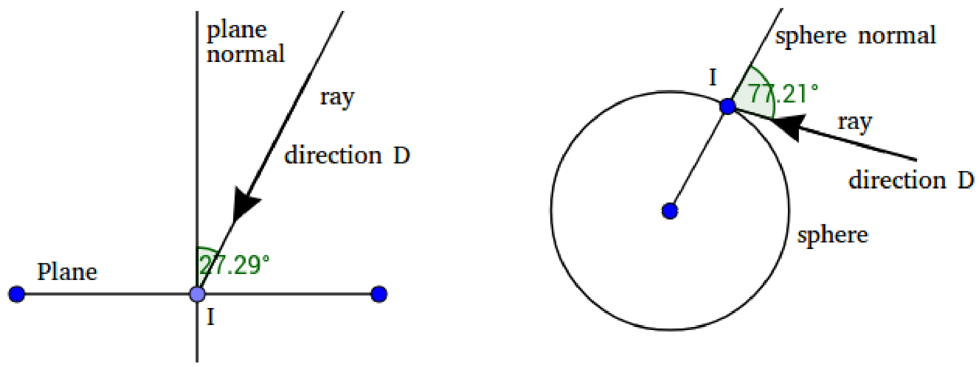


Figure 5.8: Lighting of the 3D Scene. Realized with [Hohenwarter, Borcherds, Ancsin, et al. 2017]

The very simple and intuitive approach to the scene lighting used in this implementation is illustrated in Figure 5.8: the smaller the angle α between ray and surface normal, the lighter the color. Thus, the light source is considered to be the camera center. The cosine function reflects this behavior on the interval $[0, \pi]$. [Blinn 1977] gives an overview of this procedure.

Rotation

The rotation of the world space is realized through mouse dragging. The camera is always oriented towards the world origin. Let P_1 be a world point at which the mouse is pointed in a current frame. Note that infinitely many points lie on that specific ray. In my implementation the choice of P_1 is optimized to produce a rotation which is as natural as possible. Analogously, P_2 is a world point at which the mouse was pointed in the previous frame. The aim of this rotation procedure is to move P_2 to the location of P_1 while the origin is maintained, therefore the axis of rotation is the cross product

$P1 \times P2$. [Murray 2013] offers details about how to produce the respective rotation matrix.

5.4 Limitations

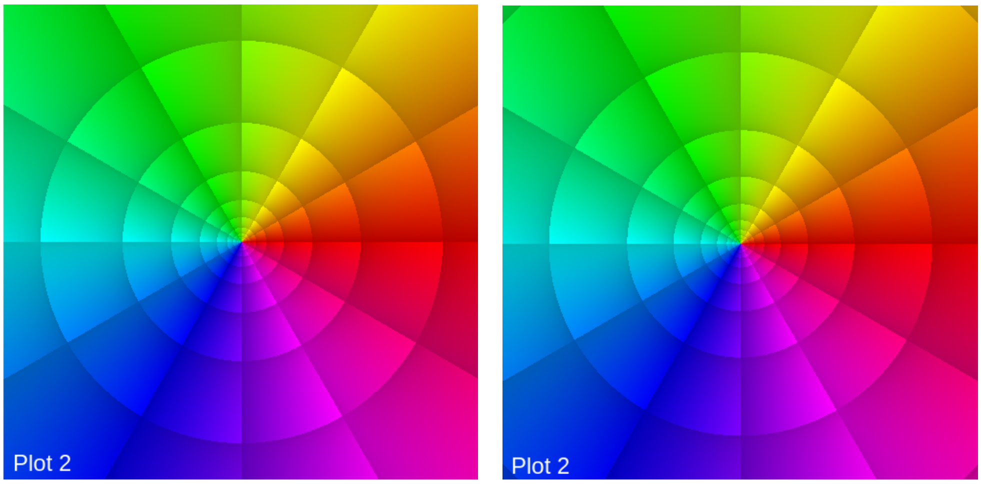
Lighting. As described in Section 5.3.1 the lighting procedure in this implementation is very simple. Experimenting with different lighting techniques as well as introducing different light sources other than the camera center might lead to a more natural or more appealing visualization. For some detail on this matter the reader is referred to [Glassner 1989, p. 8-].

Antialiasing. In Figure 5.5 it is noticeable that antialiasing has been omitted in this implementation, especially when looking at the margins of the sphere. One approach to this problem is described in [Glassner 1989, p. 12-]. However, it is worth considering the trade-off between visualization enhancement and the additional computational effort.

6 Analytic Landscapes

6.1 Problem Statement

Chapter 2 gave an overview of phase portraits as an instrument of visualizing complex functions. As [Wegert 2012, p. 3] points out, by using color as an encoding of the complex argument, only one part of the function is involved. An example where this is critical is depicted in Figure 6.1. Figure 6.1a shows the phase of the function $f_1(z) = z$, whereas Figure 6.1b shows $f_2(z) = 3z$. Not only is the difference barely noticeable at first sight, it is also unlikely for the viewer to determine which is which. One approach could be using the (in this case) concentric contour lines around the point 0. The reader is referred to [Wegert 2012, p. 6] for details on this matter. However, considering that the aim of this thesis is to produce intuitive representations, it is reasonable to claim that the phase plots from Figure 6.1 are not useful to visualize the difference between f_1 and f_2.



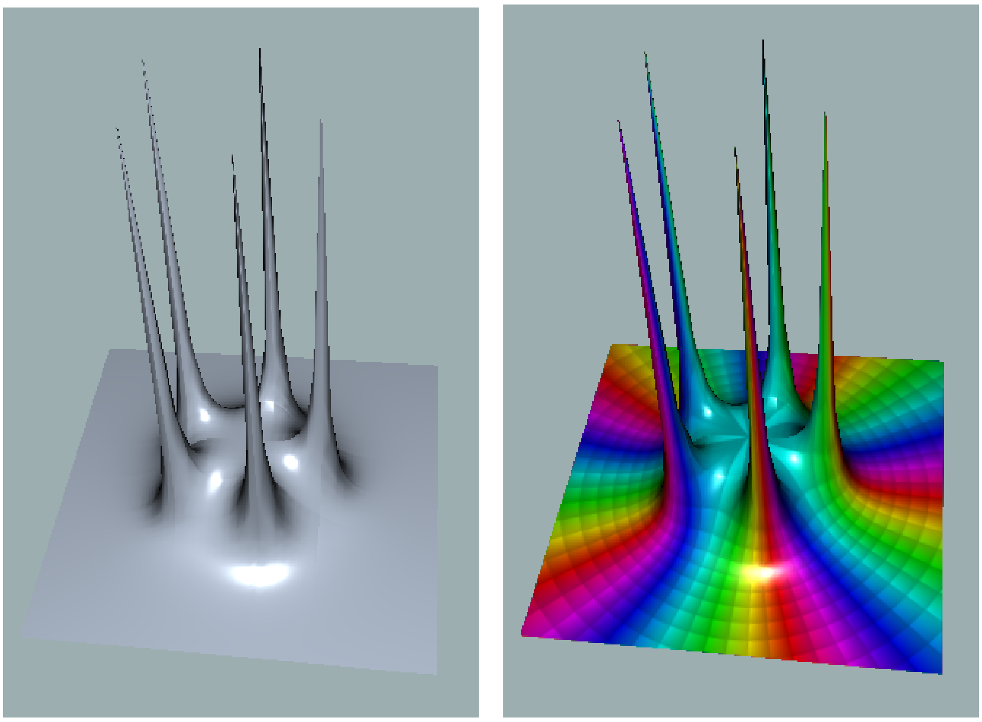
(a) $f_1(z) = z$

(b) $f_2(z) = 3z$

Figure 6.1: Comparison between the phase plots of z and $3z$

In order to overcome this issue, not only the argument, but also the modulus of a given complex function f must be included into the visualization. One possible approach is the analytic landscape as described by [Wegert 2012, p. 1-]. In a three-dimensional space, the complex plane is represented on the horizontal (x, z) plane. Each complex number $a = x + zi$ yields a complex modulus as defined in Equation (6.1) [Wegert 2012, p. 17]. This is the y-value for the point (x, z) which corresponds to a. The y-axis is considered to point upwards. Figure 6.2a shows this representation on the function $f(z) = \frac{1}{(z^5-1)}$. [Wegert 2012, p. 3] argues that phase is a very important aspect of complex function visualization, even more important than the modulus. Therefore, it is included in the visualization tool, producing the colored analytic landscape [Wegert 2012, p. 3] from Figure 6.2b. For simplicity, I will use the term "analytic landscape" to refer to the colored version from now on.

$$|x + iz| = \sqrt{x^2 + z^2} \tag{6.1}$$



(a) Without argument information

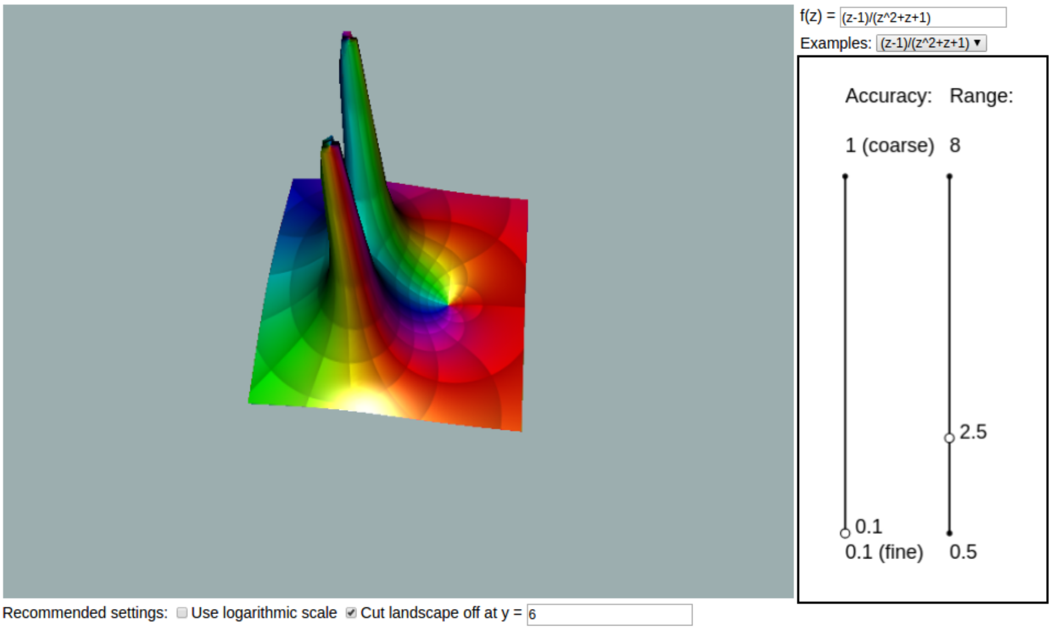
(b) With argument information

Figure 6.2: Analytic landscape representation of $f(z) = \frac{1}{(z^5-1)}$

The aim of this chapter is to produce a piece of software which computes the analytic landscape for an arbitrary input function f. Especially, focus lies on the correct representation of roots and poles, since these arise from the landscapes in an obvious manner (e.g. the five poles from Figure 6.2).

6.2 Evaluation

Figure 6.3 shows the tool for visualizing analytic landscapes. The user is provided with a large view of the landscape in a 3D environment. Zoom, rotation and similar controls are intuitive by mouse wheel and mouse dragging. In the upper left corner, the user can input an arbitrary function or choose an example. The landscape which is produced by the software is merely an approximation. Its accuracy can be modified using the left one of the two sliders on the right hand side. The exact way it works shall be described in Section 6.3.3. For now it is sufficient to consider that a very accurate approximation is assumed to require significantly more computational power than a less accurate one. The rightmost slider changes the size of the depicted domain (i.e. the section of the complex plane above which the landscape is produced. It is centered around 0). For a more pleasant visualization of poles, one might choose to cut off the landscape at a certain height.



f(z) = (z-1)/(z^2+z+1)

Examples: (z-1)/(z^2+z+1) ▾

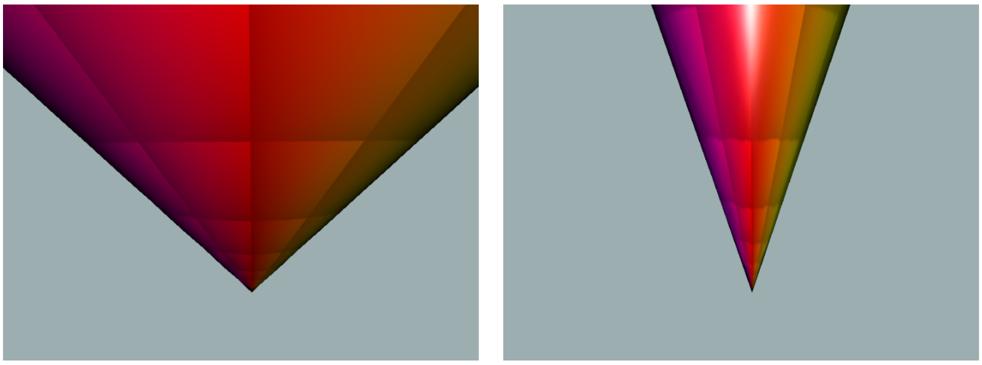
Recommended settings: □ Use logarithmic scale ☑ Cut landscape off at y = 6

Figure 6.3: Analytic landscape visualization tool

The tool is intended to provide a simple and self-explanatory user interface. The container where the analytic landscapes are rendered occupies most of the visible area, which should attract the user's attention. The two sliders on the right hand side are understood best by trial, whereas a drop down menu above them provides several examples, from simple (e.g. $f(z) = \frac{1}{(z^5-1)}$) to more complicated (e.g. $\Gamma(z)$). The number of UI elements is reduced to a bare minimum in order to provide a quick-to-learn usage.

6.2.1 Selected Examples

Different modulus on same phase. We now return to the example in Section 6.1 from which the discussion about analytic landscapes began. The phase plot did not offer any intuition about the difference between the functions $f_1(z) = z$ and $f_2(z) = 3z$. Figure 6.4 shows the same functions represented using analytic landscapes. It is trivial to detect which represents f_1 and f_2, respectively. Moreover, it is possible to determine the represented function from the landscape alone, by noticing the approximate slope of 1 on Figure 6.4a and thus conclude that the respective landscape represents $f_1(z) = z$. This observation can be applied to Figure 6.4b analogously.



(a) $f_1(z) = z$

(b) $f_2(z) = 3z$

Figure 6.4: Comparison between analytic landscapes of z and $3z$

The Gamma Function. For a verification of the implementation, I used the gamma function $\Gamma(z)$ as described by [Weisstein 2002c]. Figure 6.5 shows the result produced by the tool. The reader is referred to [Weisstein 2002c] for comparison.

6.3 Implementation

From a geometrical point of view, the analytic landscape which the tool tries to produce can be described as a surface in a three dimensional environment. This surface needs

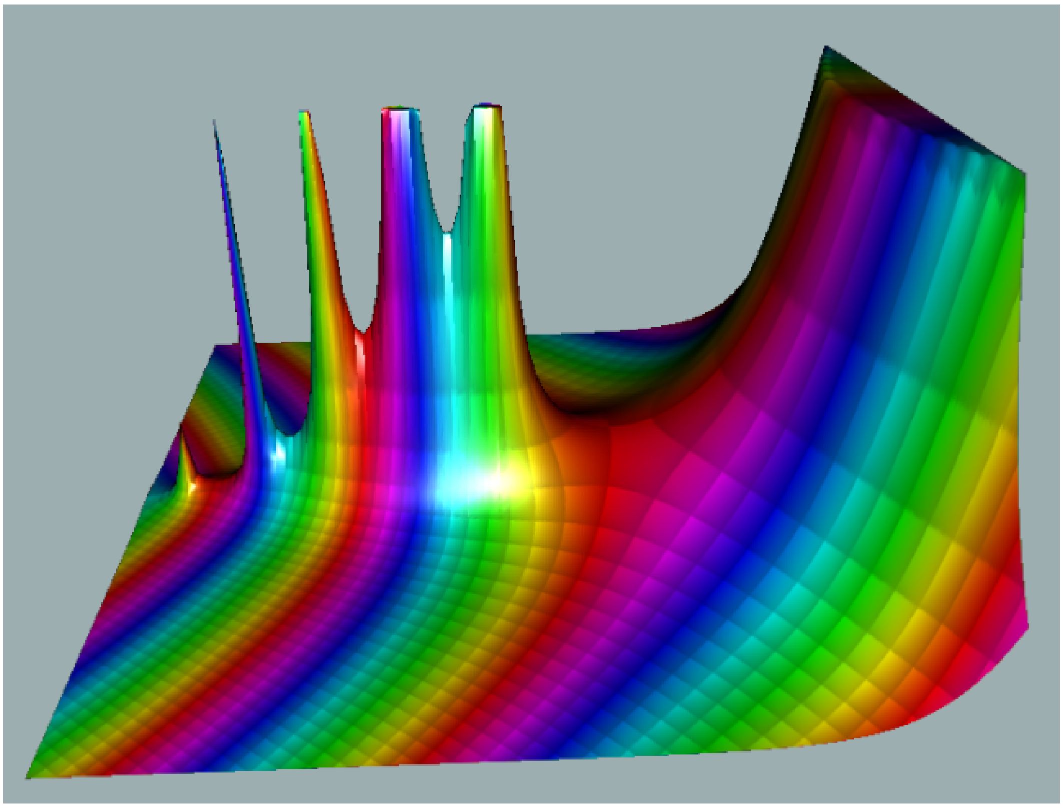


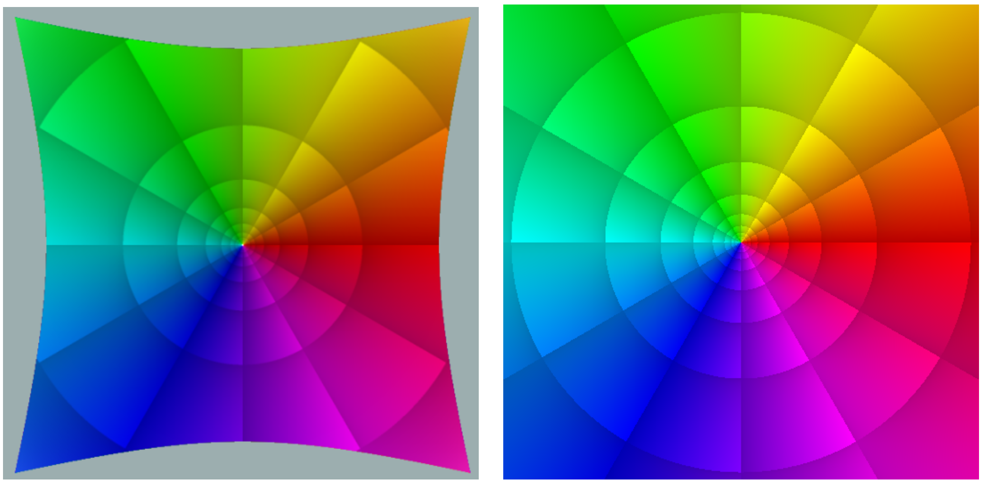
Figure 6.5: Analytic landscape of the Gamma function

to be discretized in order to produce a regular grid which can then be rendered using Cindy3D [Gagern 2017].

6.3.1 Coordinate System

Before proceeding to the actual implementation steps, clarification about the setup of the scene from a geometrical point of view is required. For the following examples, the function $f(a) = a$ is plotted, as it is relatively simple to comprehend. Deliberately, the parameter is named a and not z in order to avoid confusion with the $z-$coordinate. We will discuss positioning of points in the 3D scene and therefore use the coordinates (x_P, y_P, z_P) for any point P in the scene.

Please note that the coordinates of points, lines and other geometric objects which will be discussed in this chapter might differ form their actual coordinates in the Cindy3D environment. Some parameters needed to be adapted for consistency with the previous chapters. Especially, the landscapes should resemble regular (flat) phase plots, as depicted in Figure 6.6 if viewed from above (positive y-direction). For further details, the reader is referred to the source code.



(a) Analytic landscape from above

(b) Phase plot

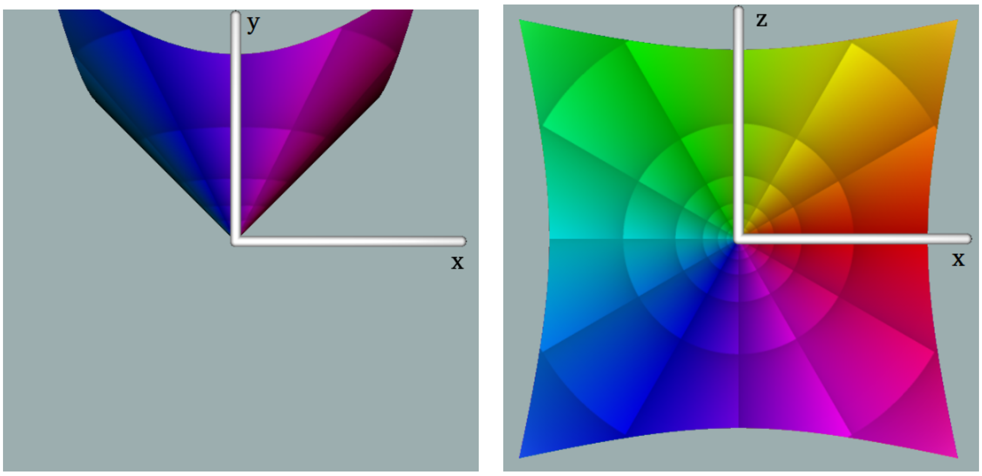
Figure 6.6: Comparison between analytic landscape and phase plot for the function $f(a) = a$

Figure 6.7 clarifies the placement of the analytic landscape in the 3D coordinate system. The white line segments show the respective axes in a positive direction. In Figure 6.7a, the camera is placed at $z = -10$, whereas Figure 6.7b shows the landscape from above at $y = 10$.

Thus, the complex plane lives on the (x, z)-plane. I defined a positive real valued boundary for the region which will be plotted. Figure 6.8 shows the part of the complex plane (i.e. the white square) which serves as a domain for the plots from the examples above. We shall refer to this area as the plotting range. In order to obtain a complex number for each point on the plane, the $x-$axis represents the real part and the $z-$axis the imaginary part. For a better understanding, some points and their respective complex value are marked on Figure 6.8.

6.3.2 Constructing the Landscape

Figure 6.9 shows an example of how the computation is executed for the complex number $a_0 = 1$. The point X has 3D coordinates $(1, 0, 0)$ and therefore represents a_0. The complex modulus of $f(1)$ is $|1| = 1$ by Equation (6.1). Consequently, the point Y with coordinates $(1, 1, 0)$ represents a point with a $y-$coordinate corresponding to the modulus of a_0. Points on the analytic landscape (e.g. Y) shall be referred to as *landscape*



(a) From a side (b) From above

Figure 6.7: Analytic landscape of $f(a) = a$ from different points of view

points throughout this section. This defines the helper function f_h from Equation (6.2), which constructs the entire landscape for a given domain on the (x, z)-plane.

$$f_h : \mathbb{R}^3 \to \mathbb{R}^3; f_h(P) = (x_P, |f(x_P + z_P \cdot i)|, z_P) \tag{6.2}$$

6.3.3 Sampling from the Complex Plane

As pointed out at the beginning of Section 6.3, the plotting range on the complex plane needs to be discretized into a regular grid [Gagern 2017]. Therefore, the software evaluates f_h at equidistant points on the plotting range. We refer to this process as *sampling*. The distance between two sampling points on the $(x, z)-$plane (on both the $-x$ and the $z-$direction) shall be the sampling width `plotW`.

A grid of sampling points is depicted in Figure 6.10a. Computing a $y-$coordinate for every point in the grid yields a set of landscape points. Each square formed by four landscape points is split into two triangles [Gagern 2017], thus producing the landscape from Figure 6.10b. The orientation in this figure is different from the former one in order to emphasize the inexact shape of the result due to the unrealistically large sampling width. $f(a) = a$ is a fairly simple function and thus this coarse approximation does not vary much from an enhanced version.

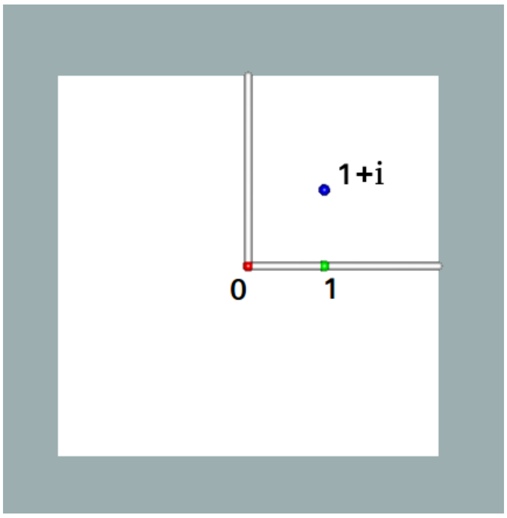


Figure 6.8: Plotting range on the complex plane

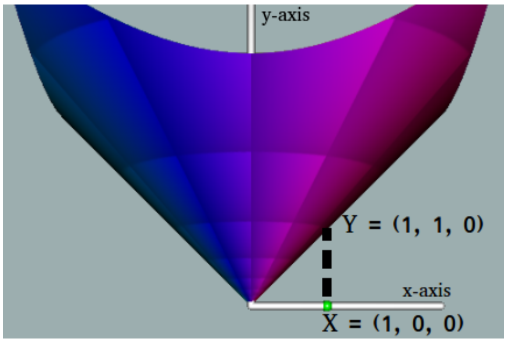
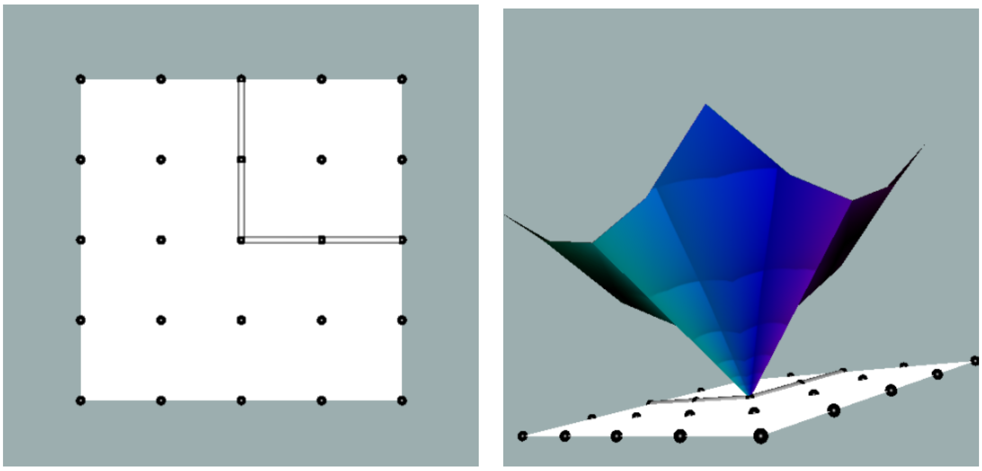


Figure 6.9: Mapping a complex number to a point on the landscape

6.3.4 Optimizing at Points of Interest

In the example from Figure 6.10, the point $(0, 0, 0)$ which corresponds to the complex number $a_0 = 0$ is included in the sampling. a_0 is a root of the function f. In order to visualize the importance of roots in the sampling process, I shall analyze Figure 6.11. Figure 6.10 displays an example with an unrealistically large `plotW`. However, the result is not unacceptably bad.

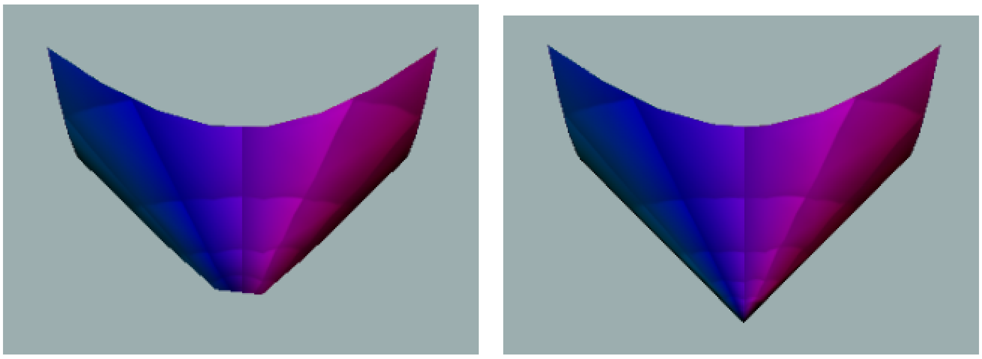
One decisive factor was that coincidentally $a_0 = 0$ was among the sampling points. In Figure 6.11a, this is not the case and the resulting landscape clearly differs from the enhanced one in Figure 6.11b. *Enhanced* in this case refers not only to a significantly smaller `plotW`, but also to some additional optimization steps which will be discussed



(a) Grid on the complex plane (b) Landscape produced from the samples

Figure 6.10: Building the analytic landscape from samples

in this section.

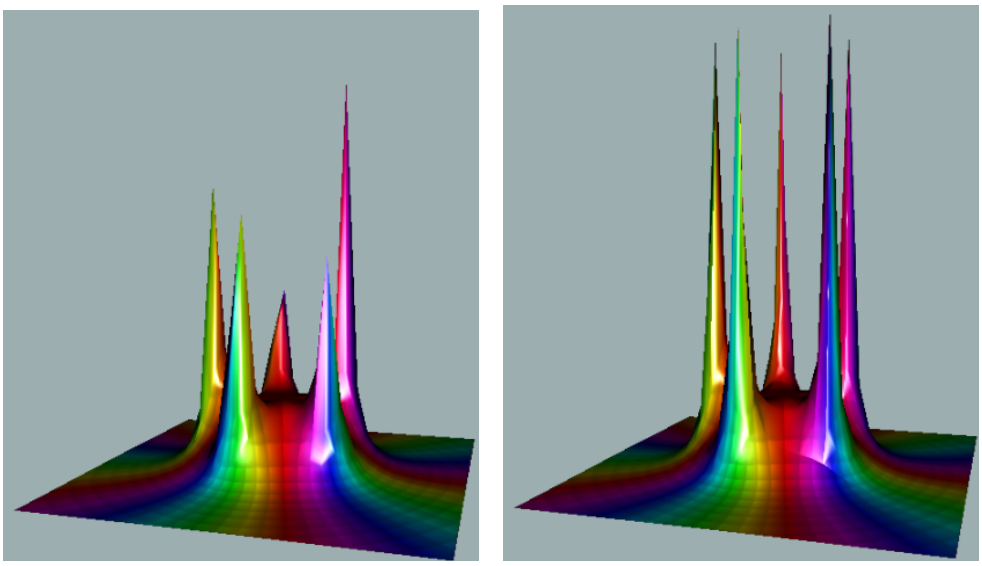


(a) Sampling without $a_0 = 0$ (b) Enhanced sampling

Figure 6.11: Comparison between regular and enhanced sampling

Obviously the sampling width `plotW` is unrealistically large in Figure 6.11a. Figure 6.12 shows a real world example. In this case, it is not the roots of the function, but the poles which influence the appearance of the results significantly. We now leave the example $f(a) = a$ behind and advance to the more complicated $g(a) = \frac{1}{a^5+1}$ Both Figure 6.12a and Figure 6.12b show the function g using the same `plotW`. In Figure 6.12b the software attempts to include the poles of g (or points which are sufficiently close to poles) into the sampling.

Note: It is important to understand that sampling from the complex plane produces a grid where the distance between neighboring points is `plotW` on both the $x-$ and the $z-$direction. This is usually not the final grid which will be redered as an analytic landscape. The optimization steps which I am going to describe add more points to this grid. In most cases the grid points will not be evenly spaced after optimization. The sampling width `plotW` only refers to the initial grid spacing, without optimization.



(a) Sampling without poles

(b) Enhanced sampling (with poles)

Figure 6.12: Comparison between sampling with and without adding poles explicitly

Consequently, the following improvement procedure is obtained (we will refer to it as *preprocessing*). First of all, I define the minimum sampling width `plotWMin`. This is not only the minimum distance between sampling points which the user can choose using the slider, but it is also used in the preprocessing step, where the tool tries to find all roots and poles of the function g, which lie inside the plotting range.

For this purpose, g is evaluated once at steps of `plotWMin` (on the $x-$ and $z-$ direction), but instead of forming the grid which is described in Section 6.3.3, only the points $P = (x_P, 0, z_P)$ where $|g(x_P + z_P i)| < \epsilon$ are of interest, as they may be very close to a root a_0 of g. Ten Newton iterations [Weisstein 2002d] are applied to those points, in hopes of converging to a_0. If the point P_{a_0} which corresponds to a_0 is found, it is included into the grid of sampling points.

An analogous procedure is applied in order to approximate poles of g by looking for points $P = (x_P, 0, z_P)$ where $\frac{1}{|g(x_P+z_P i)|} < \epsilon$ and then applying 10 Newton iterations

[Weisstein 2002d] to $\frac{1}{g}$. The result for $g(a) = \frac{1}{a^5+1}$ is shown in Figure 6.13. Although it is more accurate than the landscape from Figure 6.12a, it is not yet as satisfying as Figure 6.12b.

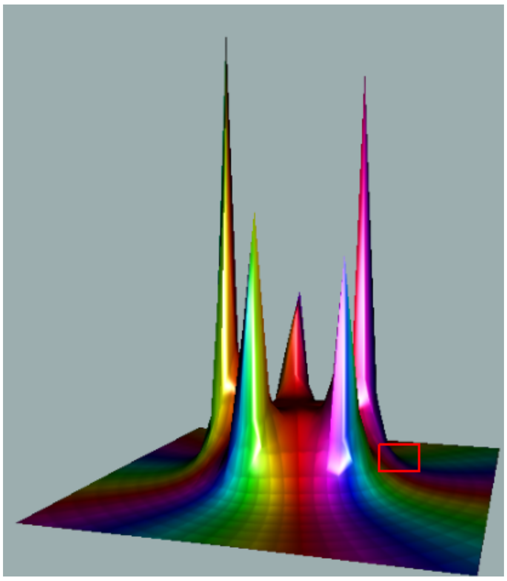


Figure 6.13: Analytic Landscape with sampling around roots and poles

An attempt of improvement was made by revising the conditions $|g(x_P + z_P i)| < \epsilon$ and $\frac{1}{|g(x_P+z_P i)|} < \epsilon$. Figure 6.14 is a zoom-in on the landscape from Figure 6.13. The red square in Figure 6.13 shows the approximate region depicted in Figure 6.14. In terms of $y-$coordinates, the points P_1, P_2 and P_3 on the analytic landscape (marked by the white dots) are very close to each other. However, the slope at P_1 suggests that a pole is fairly close, even if P_1 is not particularly "high", i.e. $\frac{1}{|g(x_{P_1}+z_{P_1} i)|} < \epsilon$ might not necessarily hold. It is still reasonable to perform the mentioned Newton iterations [Weisstein 2002d] at the location of P_1.

Looking back at the preprocessing step where we tried to find roots and poles of the function to-plot (in this case $g(a) = \frac{1}{a^5+1}$), we no longer consider the conditions $|g(x_P + z_P i)| < \epsilon$ and $\frac{1}{|g(x_P+z_P i)|} < \epsilon$ for each preprocessing point $P = (x_P, 0, z_P)$ on the plotting range, but take the slope into consideration. The tool attempts to approximate d, the modulus of the derivative of g at $x_P + z_P i$, and the two conditions above are replaced by $|g(x_P + z_P i)| < \frac{\epsilon}{d}$ for a root and $\frac{1}{|g(x_P+z_P i)|} < \epsilon \cdot d$ for a pole, respectively. Adding this improvement yields the desired result from Figure 6.12b.

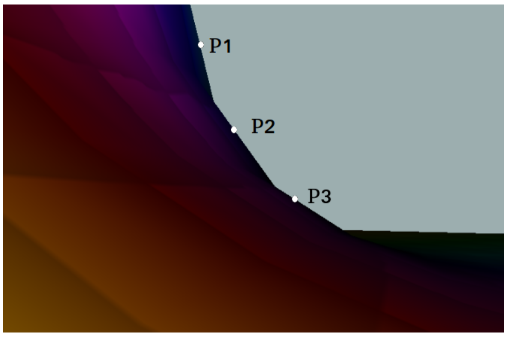


Figure 6.14: Slope at points close to poles

6.3.5 Putting Things Together

Section 6.3.1 describes the parameters of the analytic landscapes which the tool produces from a geometrical point of view. The plotting range is introduced as a region on the $(x, z)-$plane, which is interpreted as the complex plane. Section 6.3.2 defines the mapping f_h which computes a $y-$coordinate for every point P on the plotting range. By applying f_h to all points on the plotting range, the 3D object which represents the desired analytic landscape is obtained. Discretizing the plotting range through the *sampling* described in Section 6.3.3 yields an approximation of the analytic landscape. This process is optimized by the *preprocessing* step from Section 6.3.4, where f_h is applied (ideally) to all roots as well as points very close to all poles of the function for which the landscape is to be produced. These points are included in the regular grid which is then rendered [Gagern 2017].

6.4 Limitations

Steep Poles. Regardless of the optimized preprocessing from Section 6.3.4, some points of interest might still be skipped. One such case occurs if the landscape is almost flat around the very steep pole a_∞. Please note that this reasoning is merely an assumption. A proof of such behavior is beyond the scope of this thesis. Figure 6.15 shows an example where the two poles which are marked by the white circle have probably been missed by the preprocessing.

Limiting the Preprocessing. The preprocessing step described in Section 6.3.4 can be classified as very slow. Therefore, it is reasonable to avoid performing the optimization through Newton iterations [Weisstein 2002d] on too many points. A very simple example is the function $f(a) = 0$. For every point on the plotting range $|f(a)| < \epsilon$

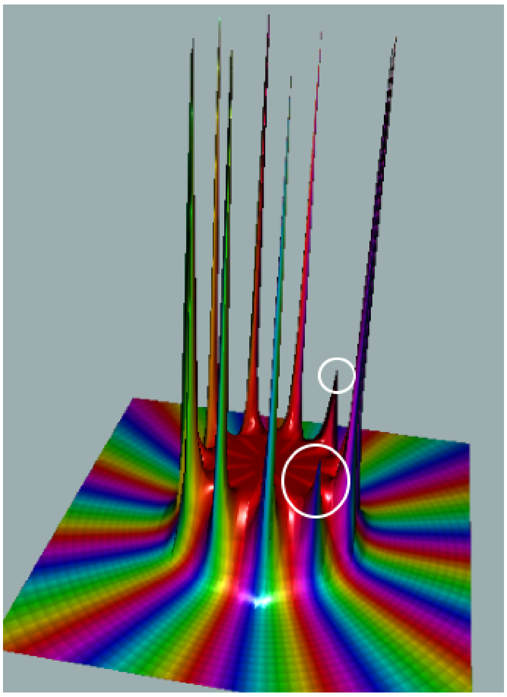
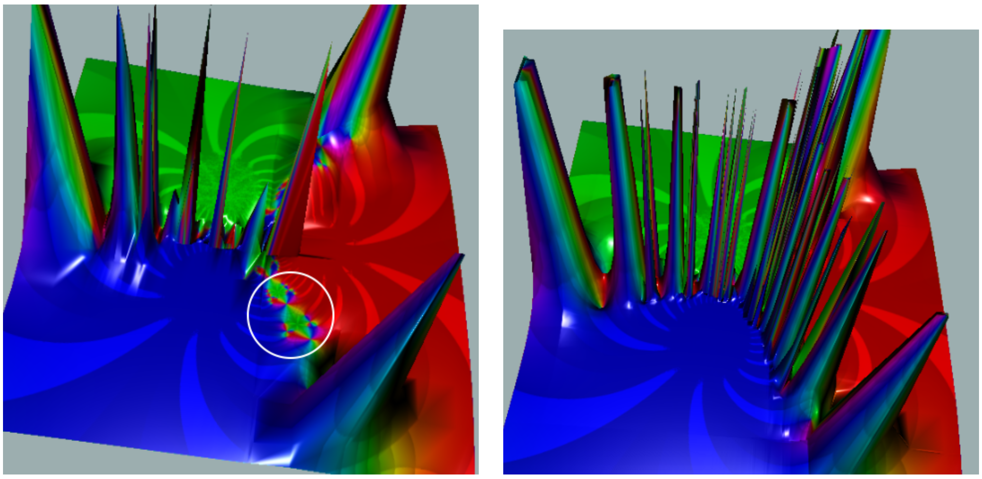


Figure 6.15: Steep poles might be missed by the procedure

holds. Not only is performing the optimization on those points of no benefit for the visualization, it also slows down the execution significantly. Therefore, the number of points on which the optimization in the preprocessing is performed must be limited. As a result, poles could be identified correctly on only a part of the plotting range. Figure 6.16a shows a landscape where the plotting width `plotW` is unrealistically high, and the maximum number of computed poles is very low. Many poles in the region marked by the white circle are not displayed correctly. Figure 6.16b shows a plot of the same function (i.e. the Newton fractal [Burton 2009] for $z^3 - 1$) with the same very high `plotW`, but with a much higher maximum number of poles. There, the resulting landscape is much more accurate.



(a) Small number of poles

(b) High number of poles

Figure 6.16: Comparison between preprocessing with a different maximum number of poles

7 Conclusions

7.1 Accuracy of Visualization

As proposed in Chapter 1, I developed four pieces of software to visualize the argument principle [Wegert 2012, p. 102], power series convergence [Wegert 2012, p. 73-], the Riemann sphere [Wegert 2012, p. 20-] and analytic landscapes [Wegert 2012, p. 27-]. All four implemented tools consist merely of approximations of their respective topic. Curves which demonstrate winding in order to visualize the argument principle [Wegert 2012, p. 102] are visualized by connecting tiny line segments, see Chapter 3. In Chapter 4, power series are approximated by their partial sums [Wegert 2012, p. 73]. Chapter 5 describes the ray tracing [Glassner 1989] approach to producing the geometrical shapes necessary for stereographic projection [Wegert 2012, p. 20-]. In order to produce analytic landscapes [Wegert 2012, p. 27-], a three dimensional plot is approximated by a *sampling* process which is described in Chapter 6. Several examples which are presented in each chapter demonstrate the pleasing degree of accuracy obtained by these approximation approaches. It is worth mentioning that constructing complex entities for an accurate visualization is not always trivial. [Wegert 2012, p. 2] points out that "*beautiful* analytic landscapes are not easy to generate, even with contemporary software".

7.2 Educational Value

In Chapter 1, I declared one of the objectives on this thesis to be a simple, interactive, intuitive and self-explanatory user interface. The tools were designed to be suitable for autodidactic investigation of the respective topic. I attempted to provide a minimalistic user interface which can be explored by trial. Every chapter contains an effort to describe a possible workflow which leads to a straightforward and independent discovery of the tool's features.

It is out of this scope to carry out the necessary testing and user studies to verify whether the proposed scenarios are realistic and which parts of the user interface need to be altered in order to indisputably offer a self-explanatory visualization experience.

7.3 Further Development

The accuracy of the representation tools has been tested on examples and a large number of those examples are covered in this thesis. However, errors can be considered to take part in the development of a piece of software. Therefore, I expect future users to encounter flaws, since the tools were designed to encourage experimentation with the input. With every discovered defect comes an opportunity for these implementations to advance in quality.

List of Figures

Bibliography

Anderson, J. W. (1999). *Hyperbolic Geometry*. London: Springer London.

Blinn, J. F. (1977). "Models of light reflection for computer synthesized pictures." In: *ACM SIGGRAPH Computer Graphics*. Vol. 11. 2. ACM, pp. 192–198.

Bornemann, F. (2013). *Funktionentheorie*. Springer.

Burton, A. (2009). "Newton's method and fractals." In: *Technical manuscript, Whitman College*.

CindyJS (2017). URL: `https://cindyjs.org/` (visited on 06/24/2017).

Gagern, M. von (2017). *Cindy 3D. Bringing 3D to Cinderella. Command reference*. URL: `http://gagern.github.io/Cindy3D/Reference/CommandReference.html#mesh3d-3` (visited on 06/06/2017).

Glassner, A. S. (1989). *An introduction to ray tracing*. Elsevier.

Griewank, A. and A. Walther (2003). "Introduction to automatic differentiation." In: *PAMM* 2.1, pp. 45–49.

Hartley, R. and A. Zisserman (2003). *Multiple view geometry in computer vision*. Cambridge university press.

Hohenwarter, M., M. Borcherds, G. Ancsin, B. Bencze, M. Blossier, A. Delobelle, C. Denizet, J. Éliás, Á. Fekete, L. Gál, Z. Konečný, Z. Kovács, S. Lizelfelner, B. Parisse, and G. Sturr (2017). *GeoGebra*. URL: `http://www.geogebra.org` (visited on 06/24/2017).

Kranich, S. (2016). "Continuity in Dynamic Geometry: An Algorithmic Approach." PhD thesis. München, Technische Universität München, Diss., 2016.

Montag, A., M. von Gagern, and J. Richter-Gebert (2017). *Function Explorer*. URL: `https://github.com/CindyJS/CindyJS/blob/master/examples/cindygl/22_explorer.html` (visited on 06/16/2017).

Murray, G. (2013). "Rotation about an arbitrary axis in 3 dimensions." In: *Online] http://inside. mines. edu*.

Newbery, A. (1974). "Error analysis for polynomial evaluation." In: *Mathematics of Computation* 28.127, pp. 789–793.

Wegert, E. (2012). *Visual Complex Functions. An Introduction with Phase Portraits*. Birkhäuser Basel.

Weisstein, E. W. (2002a). *Bernoulli Number*.

– (2002b). *Contour winding number*.

– (2002c). *Gamma function*.

Weisstein, E. W. (2002d). *Newton's method*.
– (2003a). *Binomial coefficient*.
– (2003b). *Fundamental theorem of algebra*.