The Mathematica® Journal

Domain Coloring on the Riemann Sphere

María de los Ángeles Sandoval-Romero
Antonio Hernández-Garduño

Domain coloring is a technique for constructing a tractable visual object of the graph of a complex function. The package *complexVisualize.m* improves on existing domain coloring techniques by rendering a global picture on the Riemann sphere (the compactification of the complex plane). Additionally, the package allows dynamic visualization of families of Möbius transformations. In this article we discuss the implementation of the package and illustrate its usage with some examples.

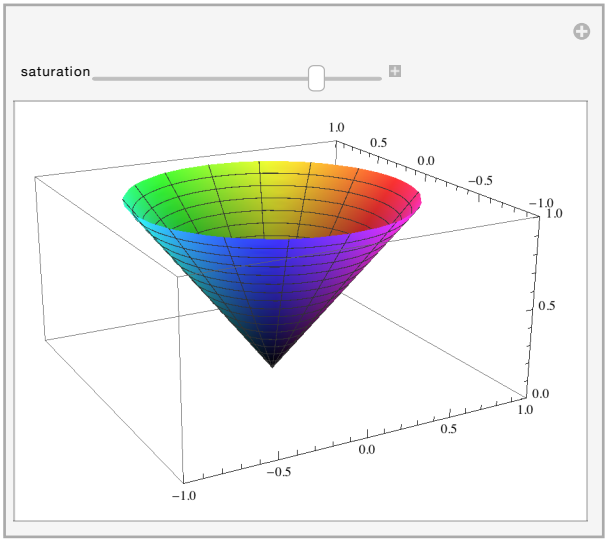
1. Introduction

Domain coloring is a technique that uses a color spectrum to compensate for the missing dimension in the graph of a complex function. It was first introduced in [1], with a detailed description of the complex-plane case in [2]. A general definition is found in [3]. More precisely, consider a function $f: U \to V$ between two sets U, V (for example, two complex manifolds). Choose a "color" function, $\kappa: V \to \text{HSB}$, where HSB denotes the Hue-Saturation-Brightness standard color space. Next, for any $z \in U$, compute $f(z)$ and evaluate the resulting color $\kappa \circ f(z)$, assigning this color to the preimage z. The final colored domain has all the information (through the color) needed to say where the point $z \in U$ gets mapped. Of course, the effectiveness of domain coloring depends on the choice of an adequate color scheme κ.

Geometrically, the HSB color space is identified with an inverted solid cone. In cylindrical coordinates, HSB is parametrized by $(s\,b\cos\theta, s\,b\sin\theta, b)$, where $\theta \in [0, 2\pi]$, $s \in [0, 1]$, and $b \in [0, 1]$. Then θ corresponds to the hue value of the visible spectrum, with $\theta = 0$, $\theta = \pi$, and $\theta = 2\pi - \epsilon$ corresponding to red, cyan, and violet, respectively. The axis of the cone $s = 0$ corresponds to zero saturation (purely a gradient of grays), and the external surface of the cone $s = 1$ corresponds to full saturation. The brightness corresponds to the height b of the cone, with $b = 0$ (the vertex) being black and $b = 1$ (the top) corresponding to full brightness.

This shows the HSB color space.

```
Manipulate[
 ParametricPlot3D[{s b Cos[θ], s b Sin[θ], b}, {θ, 0, 2 π},
  {b, 0, 1},
  ColorFunction → Function[{x, y, z, θ, b}, Hue[θ, s, b]],
  PlotRange → {{-1, 1}, {-1, 1}, {0, 1}},
  Lighting → {{"Ambient", White}},
  ViewPoint → {-1.5, -2.8, 1.1}],
 {{s, 0.8, "saturation"}, 0, 1}
]
```



In other words, HSB $= (S^1 \times [0,1] \times [0,1])/\sim$, with the equivalence given by $(\theta_1, 0, b) \sim (\theta_2, 0, b)$ for all $\theta_1, \theta_2 \in S^1$, and $S^1 \times [0,1] \times \{0\}$ identified as a single point, the vertex.

This article considers $U = V = \hat{\mathbb{C}}$, where $\hat{\mathbb{C}}$ is the Riemann sphere: the complex numbers $\mathbb{C}$ plus a single point at infinity (the north pole), embedded in $\mathbb{R}^3$, and $\kappa\colon \hat{\mathbb{C}} \to$ HSB, the *color scheme*, a usually injective map from the sphere to the solid inverted cone. Given $f\colon U \longrightarrow V$, we call U and V the *domain* and *image* Riemann spheres, respectively. (The terminology *source* and *target* spheres is also used.)

The package *complexVisualize.m* renders the domain coloring of a complex function $f\colon \hat{\mathbb{C}} \longrightarrow \hat{\mathbb{C}}$ on the Riemann sphere. The two main commands defined in the package are `complexVisualize` and `mobiusVisualize`. The former renders the domain coloring of a generic $f(z)$; the latter is optimized for Möbius transformations for the purpose of dynamic rendering. This article describes the package's implementation and illustrates its use with some examples.

In the color schemes that we explicitly consider here, the value of the hue is directly related to the argument of the image of the function, while the saturation and brightness are related to the modulus of the image values. Since a complex-valued function is determined by the argument and modulus of its image, the color scheme gives a complete global picture of the function.

The program also draws the preimage of the grid on the image sphere determined by its longitudes and latitudes. Furthermore, the preimages of three special geodesics are emphasized: the preimages of the equator are red, and the meridians corresponding to the intersections of the X-Z and Y-Z planes with the image sphere are blue and green, respectively. These correspond to the unit circle and the real and imaginary axes on the image sphere.

To use the functions in this article, make sure the `$Path` variable points to the file complexVisualize.m (*Mathematica*'s front end menu command File▸Install does this automatically) and load it.

```
Needs["complexVisualize`"]
```

2. Domain Coloring

In this section we continue to elaborate on the theory of domain coloring and discuss the main ideas behind the implementation of the package *complexVisualize.m*. We also take a close look at the implementation of some color schemes.

2.1. Implementation of Domain Coloring

Let us now formalize the concepts behind domain coloring and develop the code that provides its basic implementation.

The code in the next two subsections is self-contained and should be evaluated sequentially. The code in the third subsection requires loading the package *complexVisualize.m*.

Main Ideas and Basic Implementation

Given a complex function $f: \mathbb{C} \longrightarrow \mathbb{C}$, let

$$\kappa: \mathbb{C} \longrightarrow \text{HSB} \cong \left(S^1 \times [0, 1] \times [0, 1]\right) / \sim$$

satisfy the condition $(h \circ \kappa)(z) = \arg(z)$, where h is the projection $h: \text{HSB} \longrightarrow S^1$. Recall that HSB is interpreted as the Hue-Saturation-Brightness space. Such a function κ is referred to as the *color scheme*. Domain coloring can be understood as the implementation of the composition $\kappa \circ f: \mathbb{C} \longrightarrow \text{HSB}$ to assign color to the domain of f.

This technique can be extended to the Riemann sphere $\hat{\mathbb{C}} \cong S^2 \subset \mathrm{R}^3$. Identifying $\mathbb{C}$ with the x-y plane in R^3, the *stereographic projection*

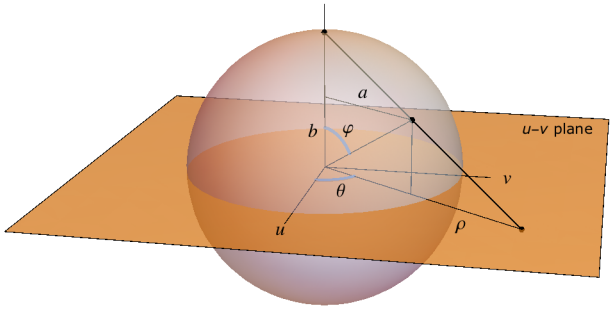
$$\sigma : \hat{\mathbb{C}} \longrightarrow \mathbb{C} \bigcup \{\infty\}$$

is defined by the requirement that $\zeta \in \hat{\mathbb{C}}$, $\sigma(\zeta)$, and $N = (0, 0, 1)$ are collinear, while $\sigma(N) = \infty$. The function f induces

$$\hat{f} : \hat{\mathbb{C}} \longrightarrow \hat{\mathbb{C}},$$

defined by $\hat{f} = (\sigma)^{-1} \circ f \circ \sigma$, with $\hat{f}(N) = \sigma^{-1}(\lim_{z \to \infty} f(z))$, provided that the limit exists in $\mathbb{C} \bigcup \{\infty\}$. (In the package's documentation, the domain and image of $\hat{f}$ are referred to as *source* and *target*, respectively.)

```
drawStereographic[1 + 3 / 2 ⅈ, showLabels → True,
 PlotRegion → {{0, 1}, {-0.4, 1.1}}]
```



▲ **Figure 1**. Stereographic projection.

The coordinate realization of the stereographic projection σ is deduced from Figure 1 as follows. Let $\zeta = (\theta, \varphi) \in \hat{\mathbb{C}}$ and $(u, v) = \sigma(\zeta)$. Let $\rho = \sqrt{u^2 + v^2}$, $a = \sin\varphi$, and $b = z = \cos\varphi$.

By similar triangles,

$$\rho = \frac{a}{1 - b},$$

so

$$\rho^2 = \frac{\sin^2\varphi}{(1 - \cos\varphi)^2} = \frac{1 - \cos^2\varphi}{(1 - \cos\varphi)^2} = \frac{1 + \cos\varphi}{1 - \cos\varphi} = \cot^2(\varphi/2).$$

Note that $\cot^2(\varphi/2) \geq 0$ for $0 \leq \varphi \leq \pi$. Thus,

$$\rho = \cot(\varphi/2),$$

and hence

$$(u, v) = (\rho\cos\theta, \rho\sin\theta) = (\cot(\varphi/2)\cos\theta, \cot(\varphi/2)\sin\theta).$$

So define a function from the plane to the sphere and its inverse.

```
sphericalToProjective[{θ_, φ_}] :=
 {Cos[θ] Cot[φ / 2], Cot[φ / 2] Sin[θ]}

projectiveToSpherical[{u_, v_}] :=
 {ArcTan[u, v], 2 ArcTan[1 / Sqrt[u^2 + v^2]]}
```

A color scheme on the Riemann sphere is a function $\kappa\colon \hat{\mathbb{C}} \longrightarrow \mathrm{HSB}$ such that $(h \circ \kappa)(\theta, \varphi) = \theta$, where (θ, φ) are spherical coordinates on $\hat{\mathbb{C}}$, and again h denotes projection onto S^1. Domain coloring is implemented, essentially, by a parametric plot.

```
ParametricPlot[
 {Cos[θ] Sin[φ], Sin[θ] Sin[φ], Cos[φ]},
 {θ, 0, 2 π}, {φ, 0, π},
 ColorFunction → colorfunction
]
```

Here *colorfunction* is, conceptually, given by $\kappa \circ \hat{f}$.

For example, consider the function f and color scheme κ:

$$f(z) = z^3 + 1 \,/\, z^3,\ \kappa(\theta, \varphi) = \mathrm{hue}(\theta \,/\, (2\pi)), \tag{1}$$

where the function hue(x) corresponds to the command `Hue[x]`. To start, transform $f(z)$ to its vector form.

```
complexToVectorFunction[f_, z_][{x_, y_}] :=
 ComplexExpand[{Re[#], Im[#]} &[f] /. {z → x + i y}]
```

For example, here is the vector form of $z \mapsto z^3 + 1 \,/\, z^3$.

```
theFunction[{x_, y_}] =
 complexToVectorFunction[z^3 + 1 / z^3, z][{x, y}]
```

$$\left\{x^3 - 3\,x\,y^2 + \frac{x^3}{(x^2+y^2)^3} - \frac{3\,x\,y^2}{(x^2+y^2)^3},\ 3\,x^2\,y - y^3 - \frac{3\,x^2\,y}{(x^2+y^2)^3} + \frac{y^3}{(x^2+y^2)^3}\right\}$$

Choose the color scheme to be $(\theta, \varphi) \mapsto \kappa(\theta, \varphi)$.

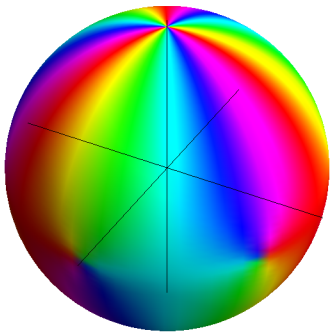
```
colorScheme1[θ_, φ_] := Hue[θ / (2 π)]
```

Therefore this is $\kappa \circ \hat{f}$.

```
theColoring = Composition[colorScheme1 @@ # &,
   projectiveToSpherical, theFunction,
   sphericalToProjective];
```

This is the domain coloring of $f(z)$.

```
ParametricPlot3D[{Cos[θ] Sin[φ], Sin[θ] Sin[φ], Cos[φ]},
 {θ, 0, 2 π}, {φ, 0, π},
 ColorFunction → Function[{x, y, z, θ, φ},
   theColoring[{θ, φ}]], ColorFunctionScaling → False,
 PlotPoints → 70, Mesh → None, Ticks → False, Boxed → False,
 AxesOrigin → {0, 0, 0}]
```



▲ **Figure 2**. Domain coloring associated with `theFunction` using the chosen `colorScheme1`; see equations (1).

Remark. The reader may already experiment at this point with different functions or color schemes by running the previous commands with a different definition of `theFunction` or `colorScheme1`.

■ *The Reference Lines*

The information given by the domain coloring can be complemented by displaying the preimages of the standard *reference lines*.

reference line (preimage)	default color
real axis	blue ●
imaginary axis	green ●
unit circle	red ●

▲ **Table 1**. Default colors for reference lines.

These preimages are given, respectively, by $(\pi_x \circ f \circ \sigma)^{-1}(0)$, $(\pi_y \circ f \circ \sigma)^{-1}(0)$, and $(\pi_R \circ f \circ \sigma)^{-1}(1)$, where $\pi_x(x+i\,y) = x$, $\pi_y(x+i\,y) = y$ and $\pi_R(x+i\,y) = \sqrt{x^2+y^2}$. This defines the projections π_x, π_y, and π_R.

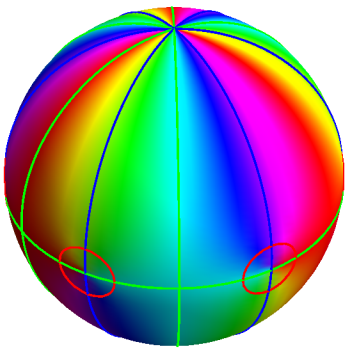
```
polarR[{x_, y_}] := Sqrt[x^2 + y^2];
cartesianX[{x_, y_}] := x;
cartesianY[{x_, y_}] := y;
```

This defines the reference lines.

```
theReferenceLines = {
   Composition[polarR, theFunction, sphericalToProjective][
     {#4, #5}] &,
   Composition[cartesianX, theFunction,
      sphericalToProjective][{#4, #5}] &,
   Composition[cartesianY, theFunction,
      sphericalToProjective][{#4, #5}] &
  };
```

This shows the domain coloring in Figure 2 complemented with the preimages of the reference lines.

```
ParametricPlot3D[
 {Cos[θ] Sin[φ], Sin[θ] Sin[φ], Cos[φ]}, {θ, 0, 2 π},
 {φ, 0, π},
 ColorFunction → Function[{x, y, z, θ, φ},
   theColoring[{θ, φ}]],
 ColorFunctionScaling → False,
 MeshFunctions → theReferenceLines,
 Mesh → {{1}, {0}, {0}},
 MeshStyle → Map[{Thickness[0.0045], #} &,
   {Red, Blue, Green}], PlotPoints → 73, Mesh → None,
 Ticks → False, Boxed → False, Axes → False
]
```



▲ **Figure 3**. Domain coloring of function and color scheme (1), showing reference lines.

■ The Package complexVisualize

The command `complexVisualize`, provided by the package *complexVisualize.m*, automates the domain coloring so far described and adds the following features:

1. The inclusion of the Cartesian x, y, and z axes (colored blue, green, and black).

   These Cartesian axes serve to orient the reader with respect to the location of 1, i, and infinity on the domain Riemann sphere. (We also refer to the Cartesian x and y axes as the *real* and *imaginary* axes, since the inverse stereographic projection sends those axes to the meridians representing the real and imaginary lines on the domain $\hat{\mathbb{C}}$.)

2. The rendering of the pullback of the standard mesh determined by the spherical coordinate parametrization of the target $\hat{\mathbb{C}}$.

   This pulled-back mesh complements the visual cues given by the pullback of the three reference lines. Its implementation follows the same idea used previously to draw `theReferenceLines`, that is to say, using the options `Mesh` and `MeshFunctions` within the sphere rendering by `ParametricPlot3D`.

3. Most importantly, a simple and flexible syntax, with the ability to call various options defined by the package.

   These options (specifying the number or color of mesh lines, for example) let you fine-tune the domain coloring of a given complex function. Standard *Mathematica* options can also be called.

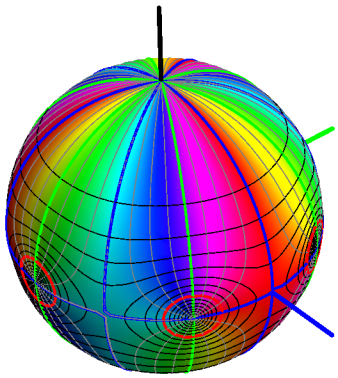
The command `complexVisualize` has the following simple syntax.

```
? complexVisualize
```

```
complexVisualize[f[z], z] provides a visualization of a
    complex function f[z] on the Riemann sphere (compactified
    complex plane) through domain coloring.  It admits the options:
        'colorScheme'  (default is "azimuth")
        'targetMesh'  (default is {15,15})
        'targetMeshColors'  (default is {Gray, Black})
        'referenceMeshColors'  (default is {Red, Blue, Green})
        'referenceMeshThickness'  (default is Thickness[0.0045])
  (Further details available within each option's help.)
```

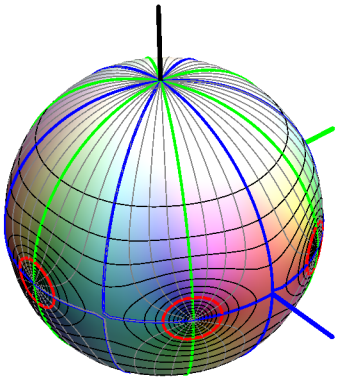
To illustrate it, consider the working example, the domain coloring of $f(z) = z^3 + 1/z^3$ with the same color scheme $\kappa(\theta, \varphi) = \text{hue}(\theta/(2\pi))$.

```
exampleDC1 = complexVisualize[z^3 + 1/z^3, z]
```



Here is an alternate rendering with a different color scheme.

```
exampleDC2 = complexVisualize[z^3 + 1/z^3, z,
  colorScheme → "metallic"]
```



We omit here the discussion of how the various options are implemented. The interested reader may consult the help documentation for individual options or review their implementation in the package.

An exception must however be granted to the option `colorScheme`, which deserves a careful discussion; it is clear that different color schemes will emphasize different features of a given complex function.

2.2. Color Scheme

The "color scheme" $\kappa : \hat{\mathbb{C}} \longrightarrow \text{HSB}$ is specified with the option `colorScheme`.

```
? colorScheme
```

```
colorScheme is an option for complexVisualize that
    determines how the target Riemann sphere is identified
    with the hue−saturation−brightness (HSB) space.

    With the default setting "azimuth", only the hue is
    accounted for and is identified with the azimuthal angle θ.

    With colorScheme −> "azimuthLatitude" the colatitude φ is
    taken into account in the saturation and brightness, so that zero
    (infinity) on the target sphere is totally dark (bright).  Further
    control is achieved with colorScheme −> azimuthLatitude[a,r].

    A general color scheme is specified as
    colorScheme−>cs, with cs[θ, φ] a Hue function of the azimuthal
    and colatitudinal angles in the target Riemann sphere.

    colorScheme −> "azimuth" is equivalent to
        colorScheme −> Function[{θ, φ}, Hue[θ/(2π)]]
    colorScheme −> "azimuthLatitude" is equivalent to
        colorScheme −> azimuthLatitude[0.9,0.3]
    colorScheme −> "metallic" is equivalent to
        colorScheme −> Function[{θ,
    φ}, Hue[θ/(2π), (φ/π)^(1/2), ((π−φ)/π)^(1/2)]]
```

With the setting `colorScheme` → `"azimuth"`, the color scheme function $\kappa_A(\theta, \varphi)$ is implemented with `azimuthColorScheme`.

```
azimuthColorScheme[Θ_] := Hue[Θ/(2 π)]
```

This coloring only keeps track of $\arg(f(z))$ and not the modulus. Although this seems rather restrictive, it is satisfactory in many examples. Indeed, the *argument principle* (see [4]) is usually enough to distinguish between zeros and poles without a visual cue coming from $|f(z)|$.

To keep track of both the argument and the modulus of the function, use the option `colorScheme → azimuthLatitude[a, r]` with $0 < a < 1, 0 < r < 1$.

```
? azimuthLatitude
```

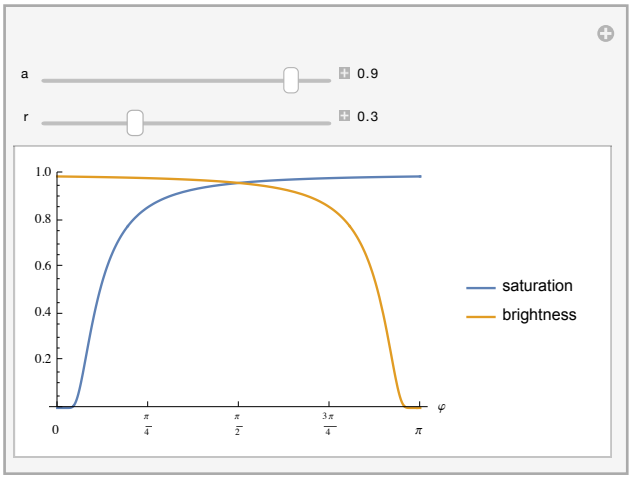
azimuthLatitude[a,r] is a setting for option colorScheme. Parameters must satisfy 0<a<1, 0<r<1. With 'a' near 1, 'r' controls the concentration of black/white around zeros/poles. Setting azimuthLatitude[] or "azimuthLatitude" is equivalent to azimuthLatitude[0.9,0.3].

This implements the corresponding color scheme function $\kappa_{\mathrm{AL}}(\theta, \varphi)$.

```
azimuthLatitude[a_, r_][θ_, φ_] :=
 Hue[θ/(2 π), Exp[Log[a] (r π/φ)^2], Exp[Log[a] (r π/(π - φ))^2]]
```

The *saturation* and *brightness* are given by bump functions.

```
Manipulate[
 Plot[
  Evaluate[Rest[List @@ azimuthLatitude[a, r][θ, φ]]],
  {φ, 0, π},
  PlotRange → {0, 1},
  Ticks → {Table[φ, {φ, 0, π, π / 4}], Automatic},
  PlotLegends → {"saturation", "brightness"},
  AxesLabel → {φ}, ImageSize → 280
 ],
 {{a, 0.9}, 0, 1, Appearance → "Labeled"},
 {{r, 0.3}, 0, 1, Appearance → "Labeled"},
 SaveDefinitions → True
]
```



▲ **Figure 4**. Saturation and brightness for color scheme `azimuthLatitude[a, r]`.

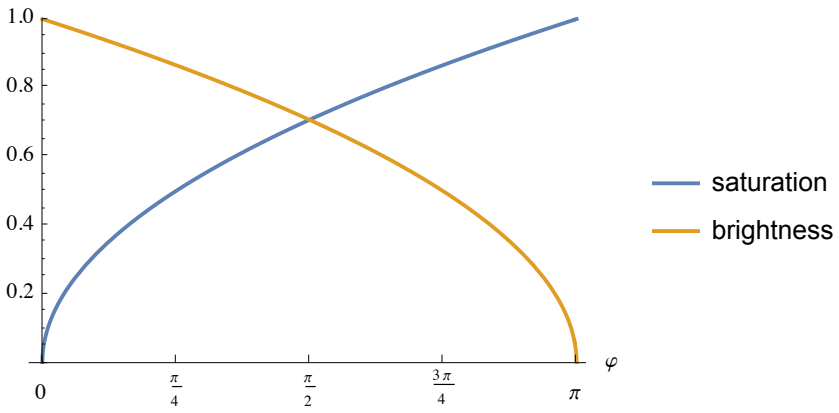
The net effect is that close to zeros of the function, the coloring tends to *darken*, while close to poles the coloring tends to *whiten*. This is because on the target $\hat{\mathbb{C}}$, zero corresponds to $\varphi = \pi$ (the south pole) and infinity corresponds to $\varphi = 0$ (the north pole). The parameters a and r fine-tune how this effect is achieved. The rational function shown in Section 3.2 gives a nice illustration of the `azimuthLatitude` color scheme.

A somewhat less sophisticated but attractive method of getting the "dark zeros and bright poles" effect is to use the color scheme function $\kappa_M(\theta, \varphi)$.

```
metallicColorScheme[θ_, φ_] := Hue[θ/(2 π), Sqrt[φ/π], Sqrt[1 - φ/π]]
```

This is invoked with the setting `colorScheme → "metallic"`. The saturation and brightness curves associated with this color scheme are shown in Figure 5, and its usage is illustrated in Section 3.2.

```
Plot[
 {Sqrt[φ/π], Sqrt[1 - φ/π]}, {φ, 0, π},
 PlotRange → {0, 1},
 Ticks → {Table[φ, {φ, 0, π, π / 4}], Automatic},
 PlotLegends → {"saturation", "brightness"},
 AxesLabel → {φ}, ImageSize → 280
]
```



▲ **Figure 5.** Saturation and brightness for color scheme "metallic".

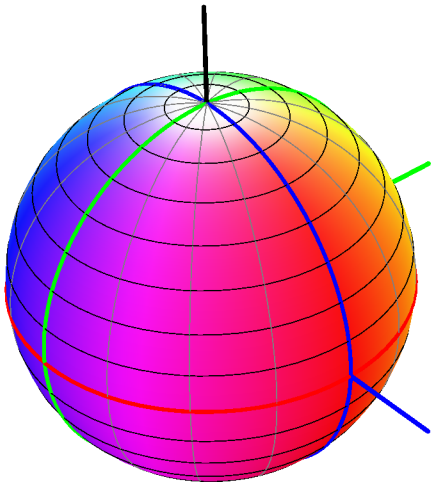
3. Examples

This section lists domain coloring examples, which illustrate the concepts discussed so far.

3.1. The Identity

Here is the domain coloring of the identity function $f(z) = z$.

```
complexVisualize[z, z, colorScheme → "azimuthLatitude",
 Lighting → {{"Ambient", White}}]
```



This seemingly trivial example familiarizes us with the essence of domain coloring: $\arg(f(z))$ is represented by the hue, and as $f(z) \to 0$ ($f(z) \to \infty$) the color darkens (brightens), assuming you use `colorScheme → azimuthLatitude[]`. The real and imaginary axes are clearly shown in blue and green, respectively. The unit circle is the equator, shown in red.

Additionally, the x, y, and z axes are always drawn by `complexVisualize` in blue, green, and black, respectively. Their purpose is to keep track of the Riemann sphere in its role as the domain of the function, with 1, i, and ∞ located, respectively, at the intersection of the x, y, and z axes with the sphere.

3.2. A Rational Function

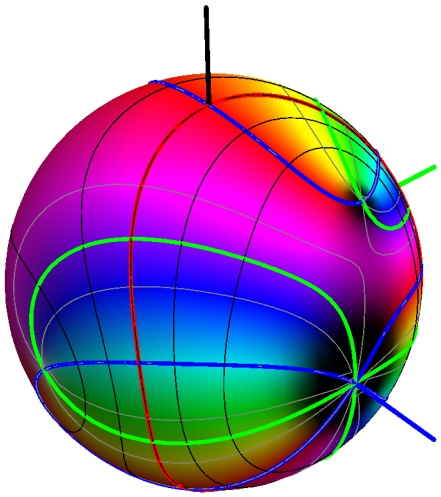
The domain coloring of the rational function

$$f(z) = \frac{(z-1)^2\,(z-2-i)}{(z-i)^2\,(z+i)}$$

serves to illustrate the global information that we get working on the Riemann sphere, as well as the nature of the zeros and poles.

Here is a rational function.

```
complexVisualize[(z - 1)^2 (z - 2 - i)/(z - i)^2 (z + i), z,
 colorScheme → azimuthLatitude[0.9, 0.1], targetMesh → {5, 5}]
```



This example shows an important result in complex analysis, the *argument principle* [4]: the number of total hue variations and orientation show the nature (type and order) of isolated singularities. The brightness and saturation complement this information, with total brightness with zero saturation corresponding to a pole, and zero brightness corresponding to a zero. See Section 2.2 for a description of the color scheme `azimuthLatitude` used in this example.

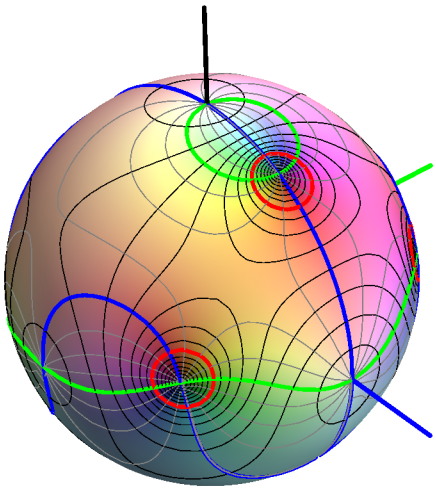
3.3. Another Rational Function

Next we illustrate the domain coloring of the rational function

$$f(z) = -z - \frac{1}{z} + \frac{z-1}{z+1} + \frac{z+1}{z-1} + \frac{z-i}{z+i} + \frac{z+i}{z-i}$$

using an alternate color scheme that we have called `metallic`, whose definition is also discussed in Section 2.2. The subtler variations in saturation and brightness still give a visual cue to the location of the zeros and poles.

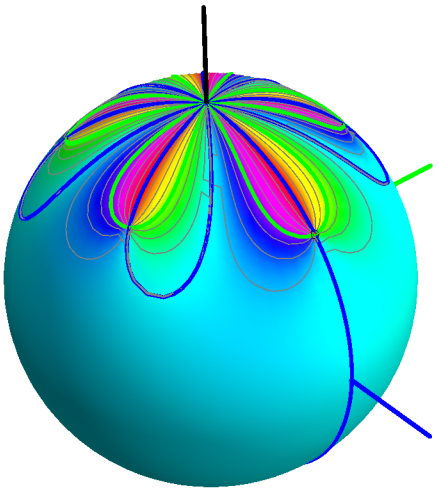
```
complexVisualize[-z - 1/z + (z - 1)/(z + 1) + (z + 1)/(z - 1) + (z - i)/(z + i) + (z + i)/(z - i), z,
 colorScheme → "metallic"]
```



3.4. A Polynomial Function

Next, let us look at a polynomial like $f(z) = z^5 + z^3 + z^2 + 10\,z - 100$.

```
complexVisualize[z^5 + z^3 + z^2 + 10 z - 100, z,
 colorScheme → "azimuth"]
```



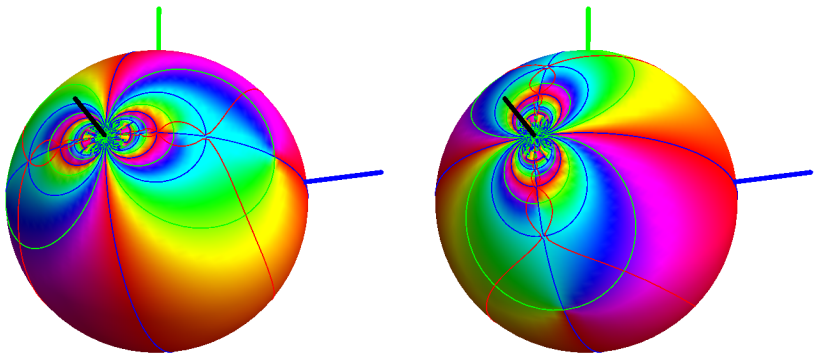
This example illustrates the relevance of having a compact domain, the Riemann sphere, to deal with isolated singularities. Indeed, whenever a function has no essential singularities, the number of zeros and poles, counted with multiplicity, coincide. This is of course a consequence of the fact that the Euler characteristic of the sphere is $\chi = 2$, so its genus is zero.

3.5. Transcendental Functions

Essential singularities have a visual interpretation closely related with the great Picard theorem [4]. Indeed, at an essential singularity we expect to observe an infinite variation of color.

Here are the circular and hyperbolic cosines.

```
GraphicsRow[
 complexVisualize[#, z,
    referenceMeshThickness → Thickness[0.002],
    targetMesh → None, colorScheme → "azimuth",
    ViewPoint → {0.9, -1.5, 2.9},
    ViewVertical → {0, 1, 0}] & /@
  {Cos[z], Cosh[z]},
 ImageSize → 500, Spacings → Scaled[-0.4]
]
```

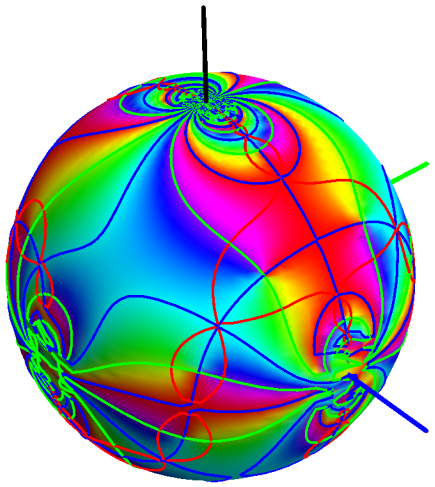


Apart from observing the essential singularities at infinity, the relation $\cos(i\,z) = \cosh(z)$ becomes evident as a $\pi/2$ rotation of the coloring around the z axis. Other identities between circular and hyperbolic trigonometric functions can be visualized in a similar fashion.

To give one more example of a transcendental function, let us look at the "baroque" coloring that we get with the function

$$f(z) = \sin\left(z + \frac{1}{z} + \frac{z-1}{z+1} + \frac{z+1}{z-1} + \frac{z-i}{z+i} + \frac{z+i}{z-i}\right).$$

```
complexVisualize[
 Sin[z + 1/z + (z - 1)/(z + 1) + (z + 1)/(z - 1) + (z - i)/(z + i) + (z + i)/(z - i)], z,
 referenceMeshThickness → Thickness[0.003],
 targetMesh → None
]
```



4. Dynamic Visualization

Given a family of complex functions, an attractive possibility is the use of *Mathematica*'s dynamic graphical capabilities provided by the `Manipulate` command to observe how the domain coloring changes as we move around the given function family.

One class of complex functions that lends itself to dynamic visualization is the class of Möbius transformations. There have been important efforts to construct optimal computational models that characterize the geometry of the Möbius group, one of the most popular being [5]. Our approach here is to construct a `Manipulate` in which the *reference curves* (i.e., the preimages of the real and imaginary axes, as well as the preimage of the unit circle) are drawn while parameters are dynamically varied. Once variation of the parameters stops, the rest of the elements of domain coloring are rendered.

The Möbius transformations constitute the set of all linear rational functions on the extended complex plane, which is identified with the set of all conformal automorphisms on the Riemann sphere [6], also called the *Möbius group*:

$$\text{Möb}\left(2, \hat{\mathbb{C}}\right) := \left\{ f(z) = \frac{a\,z + b}{c\,z + d} \,\middle|\, z \in \hat{\mathbb{C}}, a, b, c, d \in \mathbb{C}, a\,d - b\,c = 1 \right\}.$$

It is well known [7] that

$$\frac{\text{GL}(2, \mathbb{C})}{Z_{\text{GL}}} \cong \text{Möb}\left(2, \hat{\mathbb{C}}\right) \cong \frac{\text{SL}(2, \mathbb{C})}{Z_{\text{SL}}},$$

where $\text{GL}(2, \mathbb{C})$ ($\text{SL}(2, \mathbb{C})$) is the general (special) linear group with complex entries and $Z_{\{\cdot\}}$ denotes its center.

4.1. A Useful Characterization

A Möbius transformation $f(z)$ is completely characterized by the preimages of 0, 1, and ∞, which we will refer to as z_0, z_1, and z_∞, respectively. Indeed, it is easy to see that in terms of these parameters

$$f(z) = \frac{(z - z_0)\,(z_1 - z_\infty)}{(z - z_\infty)\,(z_1 - z_0)}.$$

Thus, dynamic visualization of Möbius transformations can be implemented by assigning a 2D controller to each of the parameters z_0, z_1, z_∞. Note that a point on the rectangle $[0, 2\,\pi] \times [-1, 1]$ gets mapped to $\hat{\mathbb{C}}$ through the use of cylindrical coordinates:

$$(\theta, z) \mapsto \left(\sqrt{1 - z^2}\, \cos\theta,\, \sqrt{1 - z^2}\, \sin\theta, z \right) \in S^2 \subset \mathrm{R}^3.$$

Since a Möbius transformation, viewed as conformal automorphisms in $\hat{\mathbb{C}}$, sends circles to circles, the preimages of the reference curves (the real axis, the imaginary axis, and the unit circle) are circles on $\hat{\mathbb{C}}$. We call them *reference circles*. Using the `Manipulate` option `ControlActive`, a fast-rendering version of `complexVisualize` can be implemented, in which only the reference circles are drawn. To this effect, it suffices to characterize each reference circle by three points.

reference circle (preimage of)	three points
real axis	z_0, z_1, z_∞
imaginary axis	z_0, z_i, z_∞
unit circle	z_1, z_i, z_{-1}

▲ **Table 2**. Reference curves determined by three preimage points.

Here $z_\zeta \stackrel{\text{def}}{=} f^{-1}(\zeta)$. Each reference circle is drawn by the elementary geometric construction described in the following. The simplicity of the construction translates in an instantaneous rendering, which allows a dynamic visualization with the command `Manipulate`.

4.2. Drawing the Three Reference Circles

We now describe the implementation of the geometric construction that quickly renders the three reference circles associated with a given Möbius transformation.

Drawing a Circle through Three Points on $\hat{\mathbb{C}}$

Given three points $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ on the unit sphere, the circle passing through them is given by

$$\mathbf{u} + \mathbf{v}_1 \cos\theta + \mathbf{v}_2 \sin\theta, 0 \le \theta \le 2\pi,$$

where $\mathbf{u}$ is the position vector of the center of the circle

$$\mathbf{u} = \frac{\mathbf{p}_2 \cdot \mathbf{n}}{\| \mathbf{n} \|^2} \mathbf{n}, \text{ with } \mathbf{n} = (\mathbf{p}_1 - \mathbf{p}_2) \times (\mathbf{p}_3 - \mathbf{p}_2),$$

and

$$\mathbf{v}_1 = r(\mathbf{p}_2 - \mathbf{u})^\wedge, \ \mathbf{v}_2 = r(\mathbf{n} \times \mathbf{v}_1)^\wedge,$$

with $r = \sqrt{1 - \| \mathbf{u} \|^2}$, and where $\wedge$ denotes the normalization operator $w^\wedge := w / \| w \|$.

Here is the corresponding code.

```
circleThroughThreePoints[{pt1_, pt2_, pt3_},
  colour_: Black] := Block[{center, v1, v2, v3, r},
  v3 = Cross[pt1 - pt2, pt3 - pt2];
  center = pt2.v3 / v3.v3 v3;
  r = Sqrt[1 - center.center];
  v1 = r Normalize[pt2 - center];
  v2 = r Normalize[v3 × v1];
  ParametricPlot3D[center + Cos[θ] v1 + Sin[θ] v2,
   {θ, 0, 2 π}, PlotStyle → {AbsoluteThickness[2], colour}]
 ]
```

The function `circleThroughThreePoints` fails when $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$ are collinear. This, however, is guaranteed not to occur, because the command is only applied to three distinct points on the sphere, which are thus never collinear.

■ *Drawing the Three Reference Circles on $\hat{\mathbb{C}}$: Implementation*

Given a Möbius transformation $f(z) = (a\,z + b)/(c\,z + d)$, this recovers its coefficients.

```
coefficientsMobius[expr_, z_] := Map[
   Reverse[Table[Coefficient[#, z, i], {i, 0, 1}]] &,
   {Numerator[#], Denominator[#]} &[Together[expr]]
  ] // Flatten

coefficientsMobius[(a z + b) / (c z + d), z] (*check*)
```

```
{a, b, c, d}
```

This gives its five preimages $z_0, z_1, z_\infty, z_i, z_{-1}$.

```
fivePreImages[{a_, b_, c_, d_}] := Block[{fpi},
  Off[Power::infy];
  fpi = {-(b / a), (d - b) / (a - c), -(d / c), (I d - b) / (a - I c),
    -((b + d) / (a + c))};
  On[Power::infy];
  fpi
 ]

fivePreImages[{a, b, c, d}] (*check*)
```

$$\left\{-\frac{b}{a}, \frac{-b+d}{a-c}, -\frac{d}{c}, \frac{-b+i\,d}{a-i\,c}, -\frac{b+d}{a+c}\right\}$$

Since `ComplexInfinity` is an acceptable point in $\hat{\mathbb{C}}$, we switch off the warning `Power::infty` in the declaration of `fivePreImages`.

These five preimages need to be located on the Riemann sphere $\hat{\mathbb{C}}$, and this is achieved by `stereoInv`, the inverse of the stereographic projection.

```
stereo[{x_, y_, z_}] = {x , y} / (1 - z);

stereoInv[ζ_] = {2 u, 2 v, ρsq - 1} / (ρsq + 1) /. {ρsq → u^2 + v^2} /.
   {u → Re[ζ], v → Im[ζ]};

stereoInv[ComplexInfinity] = {0, 0, 1};
```

According to Table 2, given $\{z_0, z_1, z_\infty, z_i, z_{-1}\}$, each reference circle is in correspondence with each triple in the list.

```
referenceTriads[{z0_, z1_, zInf_, zI_, zm1_}] :=
 {{z1, zI, zm1}, {z0, z1, zInf}, {z0, zI, zInf}}
```

```
referenceTriads[{z0, z1, z∞, zi, z-1}](*check*)
```

```
{{z1, zi, z-1}, {z0, z1, z∞}, {z0, zi, z∞}}
```

This finishes all the necessary ingredients needed to draw the reference circles associated with a given Möbius transformation. Let us implement the procedure with an example.

■ *Drawing the Three Reference Circles on $\hat{\mathbb{C}}$: Example*

Let us write the code that renders the reference circles for the Möbius transformation

$$f(z) = \frac{(0.04 - 0.93\,i)\,z + 0.2 + 0.74\,i}{z - 1.1 + 2.41\,i}. \tag{2}$$

This defines the function.

```
theMobiusT = ((0.04 - 0.93 i) z + 0.2 + 0.74 i)/(z - 1.1 + 2.41 i);
```

Here are the preimages under f of 0, 1, ∞, i, and -1, regarded as points on the Riemann sphere $\hat{\mathbb{C}}$.

```
theFivePreimages =
 stereoInv /@
  fivePreImages[coefficientsMobius[theMobiusT, z]]
```

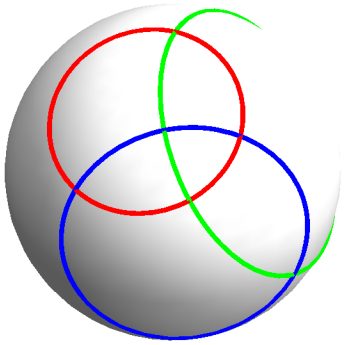
```
{{0.935562, -0.296541, -0.191802},
 {-0.097392, -0.897692, 0.429725},
 {0.274379, -0.60114, 0.750564},
 {0.495081, -0.734103, 0.464745},
 {0.609748, -0.384731, 0.692957}}
```

Hence, using the color code in Table 1, here is how the three reference circles are rendered.

```
theReferenceCircles = MapThread[
   circleThroughThreePoints[#1, #2] &,
   {referenceTriads[theFivePreimages], {Red, Blue, Green}}
  ];
```

Finally, here are the three reference circles rendered on the Riemann sphere $\hat{\mathbb{C}}$.

```
Show[{
  Graphics3D[Sphere[]], theReferenceCircles
 }, Lighting → "Neutral", Boxed → False]
```



4.3. The Command `mobiusVisualize`

The procedure discussed is implemented with the command `mobiusVisualize`, which is also part of the package *complexVisualize.m*.

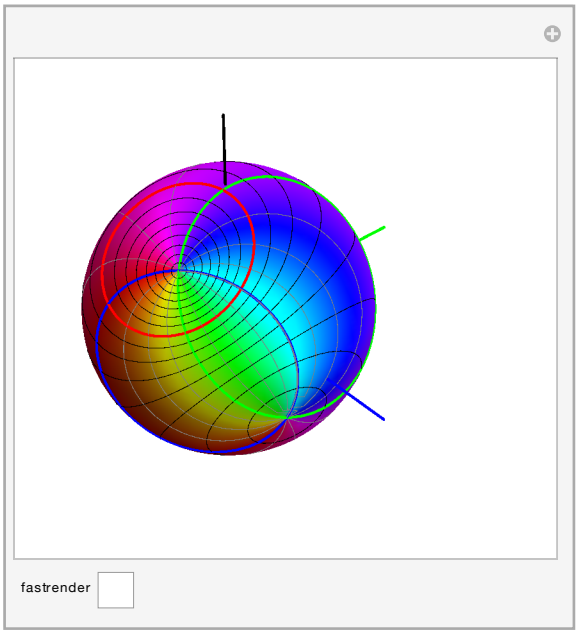
```
? mobiusVisualize
```

mobiusVisualize[f[z], z] provides a visualization of the Möbius transformation f[z] on the Riemann sphere. Its option fastRender (True/False) controls whether only the preimages of the reference curves (the real and imaginary axes, and the unit circle) are drawn.

The main option for this command is `fastRender`. With `fastRender` → `False`, `mobiusVisualize` simply invokes `complexVisualize` to do the rendering. With the (default) setting `fastRender` → `True`, the procedure described in the previous two subsections is invoked in order to draw the reference circles.

For example, here is the Möbius transformation defined in (2).

```
Manipulate[
 mobiusVisualize[ ((0.04 - 0.93 i) z + 0.2 + 0.74 i)/(z - 1.1 + 2.41 i), z,
  fastRender → fR],
 {{fR, False, "fast render"}, {False, True}},
 ControlPlacement → Bottom, SaveDefinitions → True
]
```



The command `mobiusVisualize` can be customized like `complexVisualize` through options, both the ones defined in the package and most of the standard `ParametricPlot3D` options. The interested reader can consult all the details in the package.

The point of having the ability to switch off the coloring (through the option `fastRender`) is that we can now implement a fast dynamic `Manipulate` to visualize families of Möbius transformations.

4.4. Dynamic Visualization of Möbius Transformations

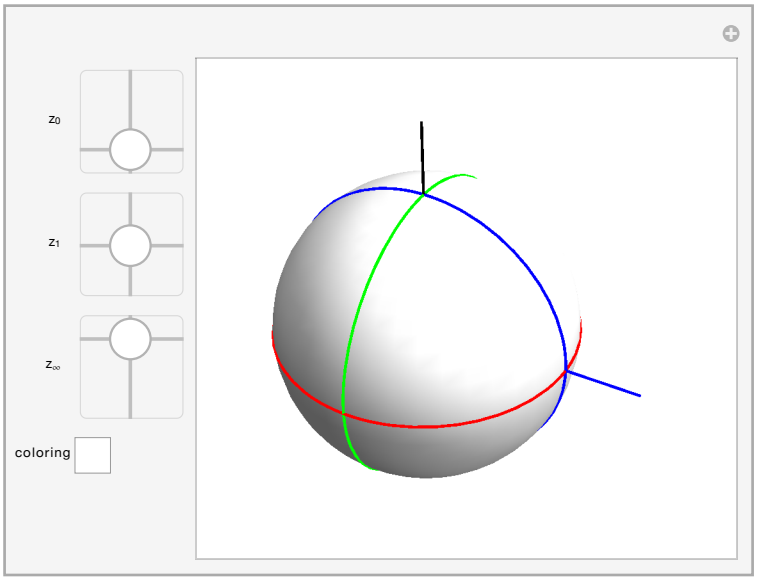
The command `cylC` handles the 2D controllers representing complex parameters on $\hat{\mathbb{C}}$.

```
? cylC
```

cylC[{θ,z}] gives the complex number represented by the
point on the Riemann sphere with cylindrical coordinates {θ,z}.

Here is a `Manipulate` allowing dynamic visualization of the group $\text{Möb}\left(2, \hat{\mathbb{C}}\right)$.

```
Manipulate[
 ϵ = 1 × 10^-6;
 Block[{z0, z1, zinf},
  {z0, z1, zinf} = Map[cylC, {cz0, cz1, czinf}];
  mobiusVisualize[(z - z0) (z1 - zinf)/((z - zinf) (z1 - z0)), z,
   colorScheme → "azimuth",
   fastRender → ControlActive[True, ! coloring]]
 ],
 {{cz0, {0, -1}, "z0"}, {-π, -1}, {π, 1 - ϵ}},
 {{cz1, {0, 0}, "z1"}, {-π, -1}, {π, 1 - ϵ}},
 {{czinf, {0, 1 - ϵ}, "z∞"}, {-π, -1}, {π, 1 - ϵ}},
 {coloring, {False, True}}, ControlPlacement → Left,
 SaveDefinitions → True,
 TrackedSymbols :→ {cz0, cz1, czinf, coloring}
]
```



The wrapper `ControlActive` in the option assignment to `fastRender` enables an instantaneous redrawing of the reference circles as the parameters are varied.

This example can be regarded as a first step in an exploration of how `Manipulate` and `mobiusVisualize` can be used together to dynamically visualize the generators and subgroups of Möb$\left(2, \hat{\mathbb{C}}\right)$.

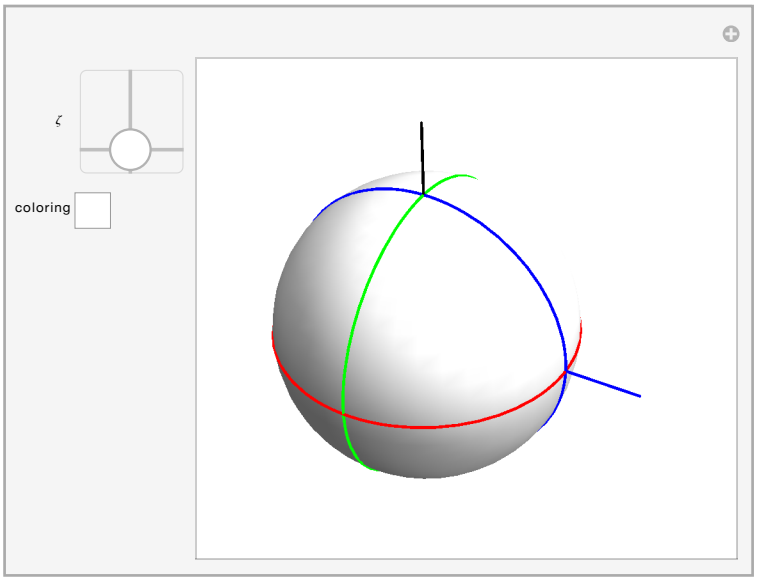
4.5. Generators of Möb$\left(2, \hat{\mathbb{C}}\right)$

To further illustrate the dynamic visualization of Möbius transformations using a `Manipulate`, consider the generators of the group of Möb$\left(2, \hat{\mathbb{C}}\right)$: translations, complex multiplication, and complex inversion.

■ *Translations*

This shows all pure translations $z \mapsto z + \zeta$, with $\zeta \in \mathbb{C}$. Experiment by varying the value of the parameter ζ.

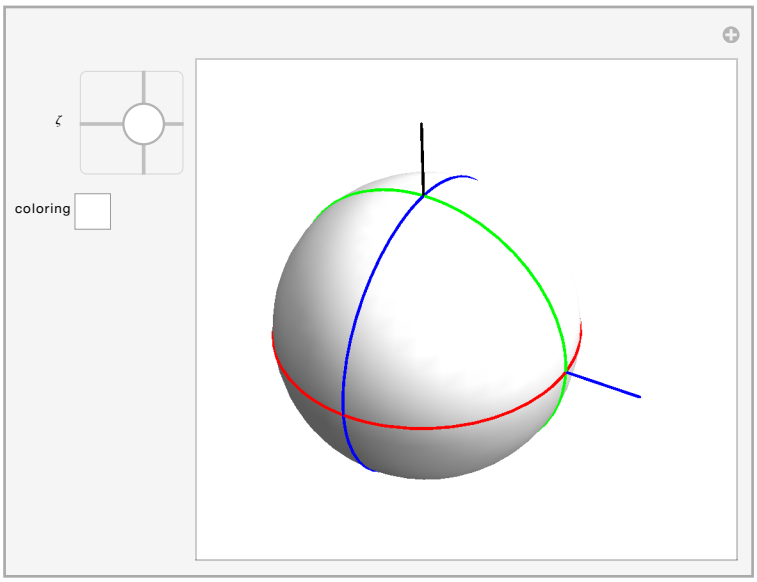
```
Manipulate[
 mobiusVisualize[
  cylC[czeta] + z, z,
  colorScheme → "azimuthLatitude", targetMesh → {14, 14},
  fastRender → ControlActive[True, ! coloring]],
 {{czeta, {0, -1}, "ζ"}, {-π, -1}, {π, 1}},
 {coloring, {False, True}},
 ControlPlacement → Left, SaveDefinitions → True,
 TrackedSymbols :→ {czeta, coloring}
]
```



■ Complex Multiplication

This shows complex multiplication $z \mapsto \zeta z$, with $\zeta \in \mathbb{C}$.

```
Manipulate[
 ϵ = 1 × 10^-6;
 mobiusVisualize[
  cylC[czeta] z, z,
  colorScheme → "azimuthLatitude",
  fastRender → ControlActive[True, ! coloring]],
 {{czeta, {π / 2, 0}, "ζ"}, {-π, -1 + ϵ}, {π, 1 - ϵ}},
 {coloring, {False, True}},
 ControlPlacement → Left, SaveDefinitions → True,
 TrackedSymbols :→ {czeta, coloring}
]
```



On the complex plane, complex multiplication is a homothety followed by a rotation, both with respect to the origin. The preceding `Manipulate` shows how rotations with respect to the origin become rotations around the z axis. The effect of a homothety also becomes visually evident as a transformation in $\hat{\mathbb{C}}$.

■ *Inversion*

The complex inverse function is $z \mapsto 1/z$. A homotopy continuously relating the identity with inversion is given by

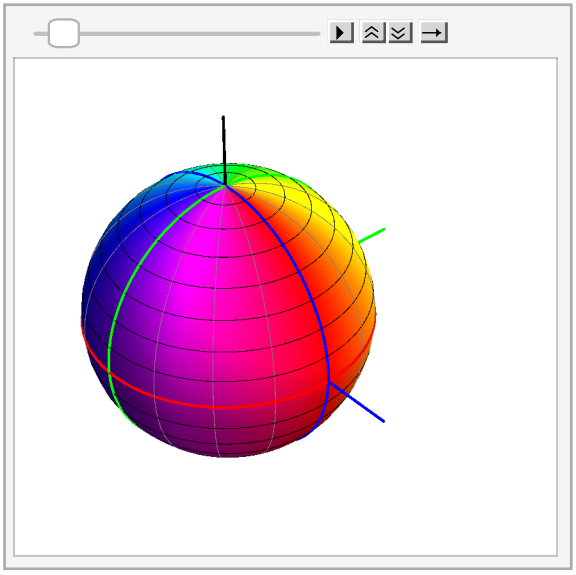
$$\varphi_a(z) = \frac{(1-a)\,z + i\,a}{i\,a\,z + (1-a)},\ a \in [0, 1],$$

so that $\varphi_o(z) = z$ and $\varphi_1(z) = 1/z$.

The following animation shows this homotopy.

```
homotopyInverse =
  Table[mobiusVisualize[((1 - a) z + i a)/(i a z + (1 - a)), z, fastRender → False],
   {a, 0, 1, 1 / 12}];

ListAnimate[homotopyInverse, AnimationRepetitions → 1,
 AnimationRunning → False, SaveDefinitions → True]
```



It becomes evident that on $\hat{\mathbb{C}}$, the geometry of a complex inversion is simply a rotation around the real axis. Note that the simplicity of this homotopy is lost when we try to visualize it purely in the complex plane.

Conclusion

The package *complexVisualize.m* is a robust and flexible tool for generating global domain coloring for holomorphic, meromorphic, and conformal functions of one complex variable, which makes it a valuable exploratory and didactic tool. We have described the main ideas behind its implementation and examples of its use.

You can experiment with further applications. Some ideas for classroom demonstrations are: (a) a sequential rendering of the Taylor expansion of a transcendental function in increasing degree; (b) exploration of the subgroups of Möbius transformations; and (c) exploration of different color schemes.

With the `colorScheme` option it is possible to implement any color rule with an appropriate function between the HSB space and the Riemann sphere. This allows the program to generalize color schemes that have appeared in the literature in the planar case.

Being on the Riemann sphere, the global character of domain coloring gives a visual interpretation of various fundamental complex analysis results. Here we have shown only a few.

When working with automorphisms on the Riemann sphere, it is possible to construct `Manipulate` animations of families of domain colorings. This dynamic visualization allows geometrical interpretations of the fundamental properties and results of these transformations to be given in an accessible and comprehensive manner.

There are many possibilities for extending the reach of the visualizations considered here. Probably the most obvious line of exploration is a deeper analysis of multifunctions on general Riemann surfaces, for which [3] is a good starting point.

Acknowledgments

The authors wish to thank Prof. Ricardo Berlanga Z. for encouragement and useful insight on the subject matter of this paper. The first author acknowledges the support of Ixtli-UNAM under project IX100310.

References

[1] F. A. Farris. “Visualizing Complex-Valued Functions in the Plane.” (Jun 15, 2015) www.maa.org/visualizing-complex-valued-functions-in-the-plane.

[2] H. Lundmark. “Visualizing Complex Analytic Functions Using Domain Coloring.” (Jun 15, 2015) users.mai.liu.se/hanlu09/complex/domain_coloring.html.

[3] K. Poelke and K. Polthier, “Lifted Domain Coloring,” *EuroVis’09, Proceedings of the 11th Eurographics / IEEE-VGTC Conference on Visualization*, Berlin, Germany, June 10–12, 2009, *Computer Graphics Forum*, **28**(3), 2009 pp. 735–742. doi:10.1111/j.1467-8659.2009.01479.x.

[4] L. V. Ahlfors, *Complex Analysis: An Introduction to the Theory of Analytic Functions of One Complex Variable* (International Series in Pure and Applied Mathematics), New York: McGraw-Hill, 1953.

[5] D. Arnold and J. Rogness. "Möbius Transformations Revealed." (Jun 15, 2015) www.ima.umn.edu/~arnold/moebius.

[6] J. Milnor, *Dynamics in One Complex Variable*, 3rd ed., Princeton: Princeton University Press, 2006.

[7] G. Toth, *Finite Möbius Groups, Minimal Immersions of Spheres, and Moduli*, New York: Springer-Verlag, 2002.

A. Sandoval-Romero and A. Hernández-Garduño, "Domain Coloring on the Riemann Sphere," *The Mathematica Journal*, 2015. dx.doi.org/doi:10.3888/tmj.17-9.

List of Additional Material

Additional electronic files:

1. complexVisualize.m

Available at: www.mathematica-journal.com/data/uploads/2015/11/complexVisualize.m

About the Authors

Ángeles Sandoval-Romero is a full professor at Universidad Nacional Autónoma de Mexico, Facultad de Ciencias, Department of Mathematics. She obtained her Ph.D. in mathematics from UNAM in 2006 with a dissertation on scattering theory applied to a system of nonlinear partial differential equations. Her research interests include geometric complex analysis, differential equations, and their applications.

Antonio Hernández-Garduño is a professor in the Mathematics Department of Universidad Autónoma Metropolitana (campus Iztapalapa) in Mexico City. He obtained his Ph.D. from Caltech in 2002 with a dissertation on the bifurcation of relative equilibria in mechanical systems. His research interests include geometric mechanics, vortex dynamics, celestial mechanics, and computer-based math education.

Ángeles Sandoval-Romero
Departamento de Matemáticas
Facultad de Ciencias - UNAM
Circuito Exterior, Ciudad Universitaria
México D.F. 04510
selegna@ciencias.unam.mx

Antonio Hernández-Garduño
Departamento de Matemáticas
Universidad Autónoma Metropolitana - Iztapalapa
Av. San Rafael Atlixco # 186
México D.F. 09340
ahg@xanum.uam.mx