*DIFFERENTIAL EQUATIONS
AND
CONTROL PROCESSES
N 1, 2005
Electronic Journal,
reg. N P23275 at 07.03.97*

*http://www.neva.ru/journal
e-mail: diff@osipenko.stu.neva.ru*

Ordinary differential equations

ADVANCED GRAPHICS IN COMPLEX ANALYSIS AND DYNAMICS

Alessandro Rosa

Freelance programmer
c/o Locatelli, Via Cappuccini 116/A, I-72100 Brindisi, Italy
e-mail: zandor_zz@yahoo.it

Abstract

The release (3.13) of Complex Mapper features newer local and global visualization methods and functionalities to display the behavior of functions in one complex variable. This program is based upon an intuitive interface. It is intended to be flexible tool for researchers, students and teaching courses. It can display any user-defined map, thanks to a parser handling functions with arithmetic, trigonometric, hyperbolic and inverse operators. An online and step-by-step help is also available. This article intends to extend the concepts in [10] and to present the new features.

1 Introduction

My mathematical interests in iteration theory and my experience as programmer led me, together with the mathematicians Daniel S. Alexander (Drake University, Iowa) and Felice Iavernaro (University of Bari, Italy), to embark on a book-project ([1]) containing the English translations of the most important French and German papers (by Cremer, Denjoy, Fatou, Julia, Lattès, Siegel and others) concerning of the theory of holomorphic dynamics in one complex variable on the whole Riemann sphere, currently in preparation. These works are presented in a modern context. The original works are introduced by a current historical research and supported by contemporary computer graphics. The goal is to set a comfortable historical context and to link the old results to latest ones via the prefaces[1] to the translations themselves. My original idea behind the first coding of Complex Mapper was to provide highly detailed figures to support the concepts throughout this book. As the time passed by, the basic kernel of this program acquired new functionalities and became a stand-alone tool, then independent of the above project, so that I finally coded a user-friendly visual interface, an on-line help and I distribute it on the Web[2]. This article shows the latest analysis tools and can be regarded as a prosecution of the work in [10].

2 Getting started with Complex Mapper

2.1 The visual interface

The graphical architecture of Complex Mapper is windows-based and relies on a Multi-Document Interface (MDI) allowing the user to handle more functions at a time by means of separated child windows, so that a high degree of freedom and flexibility is achieved, as shown in figure 1. A side bar is shown on the right, helping the management of the complex coordinates to display the maps. A given function can be iterated by setting the index inside the upper edit control, in order to study the

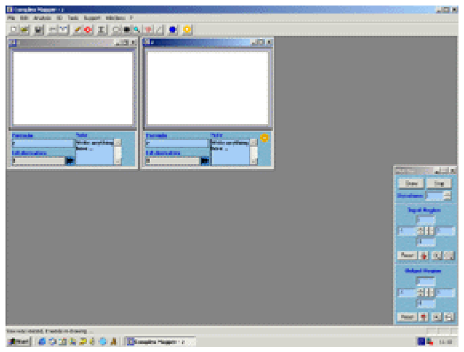
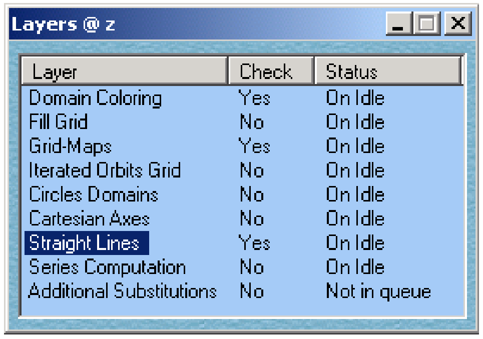


Figure 1: The Multi-Document Interface.

[1]Written by experts in this field.

[2]Freely downloadable at http://www.malilla.supereva.it

dynamics of a complex function. Both the toolbar and the menu on the top allow the user to select the desired visualization method and related features. The next chapters will illustrate them all in details. Their management was structured into a multi-levels system (see figure 2), where each level can be toggled on/off by a simple mouse click on the related row. The performance status (first column on the right) is also monitored, giving the user the acquaintance on the current drawing process. Moreover, the layered structure allows to view (and to toggle off, if not required) multiple graphics methods at the same time, for example in figure 3, where the values distribution (colors) and dynamics (grid) are displayed for the complex function $f(z) : \sin(z)$.


Layers @ z

Layer	Check	Status
Domain Coloring	Yes	On Idle
Fill Grid	No	On Idle
Grid-Maps	Yes	On Idle
Iterated Orbits Grid	No	On Idle
Circles Domains	No	On Idle
Cartesian Axes	No	On Idle
Straight Lines	Yes	On Idle
Series Computation	No	On Idle
Additional Substitutions	No	Not in queue

Figure 2: Multi-Layers structure.

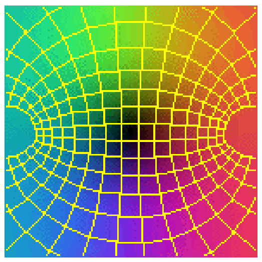


Figure 3: Concurrent visual methods.

All this should yield an easy approach to the basic tools: a concept also strengthened by an on-line help describing all components and what they are intended for. No user programming skills are assumed and thus no programming language is required to draw such figures, but just some essential knowledge in complex analysis and dynamics.

2.2 Formula Parser

Complex Mapper also includes a built-in parser to input any (simple or compound) function in one complex variable z. It supports[3] arithmetic, powers, trigonometric and inverses, hyperbolic and inverses, as well as other operators and it offers an high degree of freedom again in creating user-defined input function. This latest version (3.13) allows the user to input a number of parameters into the formula: a new and relevant feature is especially useful when treating complex series (see section 7).

[3]See on-line help for the full list.

2.3 Locally and globally

I will often refer to '*local*' or '*global*' methods throughout this paper. The former is a method which, due to its technical limits or purposes, can be applied to bounded regions exclusively, i.e. any subset of the complex plane and which can be included inside a disk of finite radius. The latter applies to methods working over the Riemann sphere, i.e. including the point at infinity too.

2.4 The toolbar

This subsection lists the additional features of the button in the upper toolbar. From left to right: 1) New View; 2, 3) Open/Save files; 4) Undo; 5) Copy Views; 6) Draw; 7) Stop; 8) Series Computation; 9) Track a Circle Domain; 10) Zoom a Grid; 11) Zoom a View; 12) Track Orbit; 13) Set a Straight Line; 14) Riemann Sphere Navigator; 15) On-Line Help.

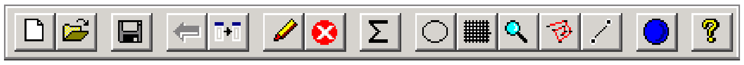


Figure 4: The toolbar.

3 Grid-Maps

3.1 Source grid-maps

This method is *local* and developed to evince the behavior of a complex function inside a user-defined neighborhood of the given point.

The term '*grid*' refers to a source grid: a discrete and finite set of points on a surface and linked together (in the geometrical sense) by a given rule (determining the related topology); '*map*' points to the application of a function on the source grid so that a new set of resulting points[4], which link together according to the same rule as the source grid, is produced[5]. Any grid-map can be handled by setting its coordinates inside a flying plot bar (see fig. 1) and parameters in the window in figure 11. In our graphic environment, points are assumed to have complex coordinates: $x + iy$. Complex Mapper allows the possibility of two types of grid-maps: *orthogonal* (fig. 5) and *radial* (fig. 6). The source grid-maps can be easily viewed by setting the mapping function to

[4]Alternatively said, '*image points*'.

[5]This is achieved by first computing the position of all points of the grid and indexing them.

the identity, i.e. $f(z) : z$, and then by clicking the 'draw' button inside the plot bar. The same terms explain how source points are generated and plotted.

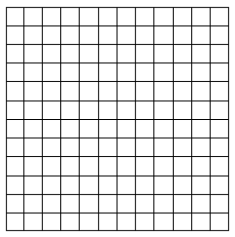


Figure 5: Orthogonal grid-map.

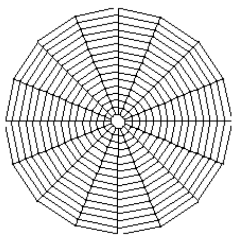


Figure 6: Radial grid-map.

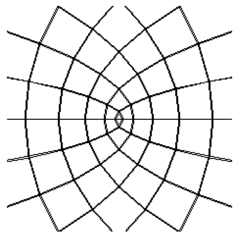


Figure 7: Mapping the orthogonal grid by $f(z) : z^2$.

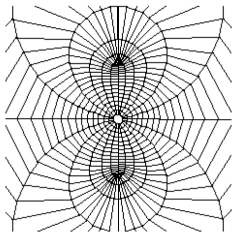


Figure 8: Mapping the radial grid by $f(z) : \dfrac{z}{z^2-1}$.

The examples in the next pages will help the reader to feel comfortable with grid-maps by experiencing that their distinct topologies can fit a better description of complex configurations, depending on the given input function, especially for complex discrete dynamical systems arising from iterations.

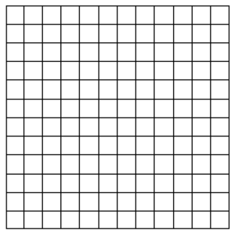


Figure 9: Source grid-map.

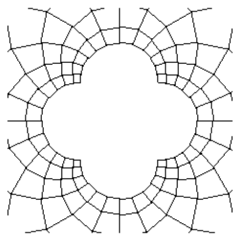


Figure 10: Applying the inversion map $f(z) : \dfrac{1}{\bar{z}}$to an orthogonal grid.

3.2 The algorithm behind grid-maps

Complex Mapper performs a 4-steps algorithm to plot grid-maps. The pseudo-language below, included here to clarify an idea rather than to provide a final version, shows how to generate a grid-map so that it can be coded into the favorite programming language.

- *allocate a memory buffer* for all complex points of a matrix [6] $h \times w$, where h is the number of rows and w is the number of columns.
- *store the complex coordinates* of the grid points into a memory buffer, while taking care of the rule linking them all: by rows and columns for orthogonal model and by sectors and circles for the radial one.

  Example: let the orthogonal grid be $10 \times 10 = 100$ complex points. They are assumed to be linked, per each row, from the first to the tenth point. Once all rows are linked, do the same `for` columns. A pair of nesting `for`-cycles could be used.
- by means of the input function, *compute the (possibly iterated) image of each point* in the source grid and store the new coordinates into the same allocated space of the source point into another (equally dimensioned) memory buffer.

```
for int w = 0; w < row_length_in_points; w++
    for int h = 0; h < column_length_in_points; h++
        store_complex_coordinates_into_memory_buffer_at_[w,h]
        compute_image_of_[w,h]_and_store_it_again_at_[w,h]
    end for
end for
```

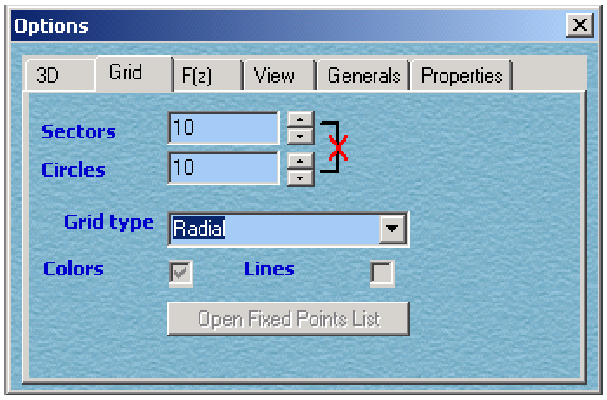
- Another pair of nesting `for`-cycles to plot the resulting before grid:

```
for int w = 0; w < ( row_length_in_points - 1 ) ; w++
    for int h = 0; h < ( column_length_in_points - 1 ); h++
        plot_a_line_from_[w,h-1]_to_[w,h]
    end for
    plot_a_line_from_[w-1,h]_to_[w,h]
end for
```

The management of grid types can be performed into a related property page at figure 11, where the parameters of the grid are set: dimensions (the density of an orthogonal grid is managed by a pair of parameters, defined '*width*' and '*height*' values for orthogonal grids, whereas '*sectors*' and '*circles*' for radial ones), the type and other additional values. The next subsection clarifies

[6]Take care that this matrix is just an artificial tool to have all points inside a set distributed per rows and columns, so that their location is easy via a pair of coordinates. The idea of the matrix does not mean to any topological grid in the space, thus it can easily describe the radial one.

such concepts by examples. Orthogonal and radial are the basic and standard types. Besides them, Complex Mapper features additional types of grids, but they are differently built-up since computed through the concept of streamlines, explained in details in section 5.



Sectors 10
Circles 10
Grid type Radial
Colors ✓ Lines

Figure 11: Source grid parameters.

3.3 A special example on why to apply different grid-maps

In this subsection, both orthogonal and radial grids will be applied to explain why the application of differently shaped grids is required to highlight the local behavior of a function. In general, given a map $g(z)$ with an indifferent fixed point, there might be two subtypes:

- *rationally* indifferent fixed point, i.e. a fixed point c where $g'(c) = e^{2\pi i\theta}$ and θ can be written as the ratio p/q of two coprimes $p, q \in \mathbb{Z}$;
- *irrationally* indifferent fixed point, i.e. a fixed point c where $g'(c) = e^{2\pi i\theta}$, given $\theta \in \mathbb{R}\backslash\mathbb{Q}$ and satisfying a diophantine condition.

The test map here is the complex entire function $f(z) : z + z^5$. The general class of maps $z + z^n$ involves the rational type and thus it often arises in holomorphic dynamics as example of particular local behaviors about *rationally* indifferent fixed points (they were named '*singular dynamics*' at the beginning of the last century, due to the special features they enjoy among all other kinds of dynamics: (super-)attracting and repelling). Investigations by Leau [7], Julia [6] and Fatou [2], showed that such kind of maps generate a vector field, with $(n-1)$ uniformly distributed and alternating radially attracting and repelling directions, arises about this indifferent point. Iterates of points close to

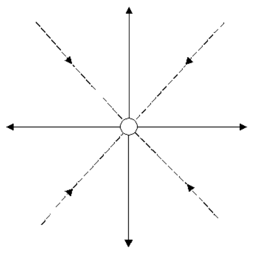


Figure 12: *Even* case: alternating directions for $z + z^5$.

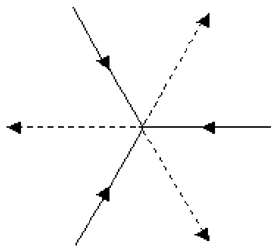


Figure 13: *Odd* case: alternating directions for $z + z^3$.

the fixed point might exhibit a very slow convergence / divergence behavior as indicated in figure 15, depending on their initial position.

Points along a given straight line either converge to the origin or move away from it. In figure 12 the *even* case: $n = 5$ so there are $(n-1) = 4$ attracting and repelling directions which land on the same straight line; if n is *odd* - see figure 13 - the same vector[7] defines both the attracting and the repelling direction, which land on a same line. The invariant neighboring regions of attraction and repulsion are called '*petals*'. This dynamical configuration is said to be a flower (due to its shape, as seen below in 17): it is the 'Fatou-Leau flower' because both mathematicians Fatou and Leau explained these intriguing dynamics in details.

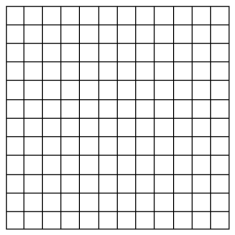


Figure 14: Source orthogonal grid-map.

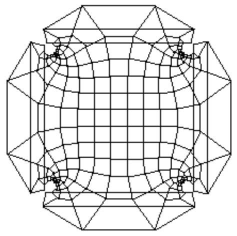


Figure 15: Iterating the orthogonal grid-map.

Figure 14 shows that the application of the orthogonal grid depicts the very slow convergence/divergence rate: the (n-fold iterated) image grid has a very little deformation about the fixed point, the neighboring grid looks like the one produced by the identity map because the iterates converge/diverge to/from the fixed point along the attracting/repelling directions respectively very slowly.

Figure 17 shows the application of mapping to the source radial grid in figure 16. Here flower-shaped dynamics are more clearly visible: in fact, the four invariant petals, intersecting at the origin, are evident. The white hole at the fixed point in the origin is due to the slow rate of convergence / divergence

[7]Thus the attribute of the direction is defined depending on its relation to the fixed point: if this is the first point then the direction is repelling, otherwise if it is the last point, the direction is attracting.

so that the iteration index was not enough large for points to get nearer to it.

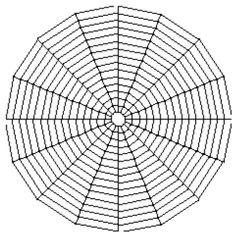


Figure 16: Source radial grid-map.

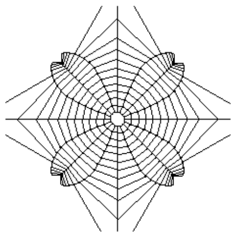


Figure 17: Iterating the radial grid-map via $z + z^5$.

3.4 Colored filled-in grids.

The application of colors increases the potential usefulness of grid-maps. The coloring is achieved by opening the window below and selecting a set of arbitrary colors, as shown in the bottom pictures. The colors at the four corners are user-defined. The gradient is computed per each row of the square. Such arbitrariness allows to know and then manage colors for evincing value distribution and highlight properties such as in figures 19 and 20.

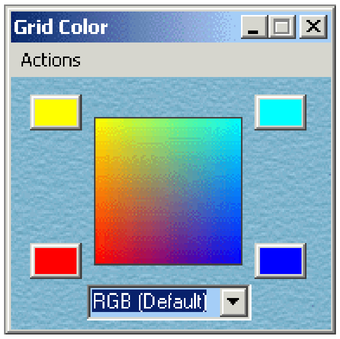


Figure 18: Grid colors editor.

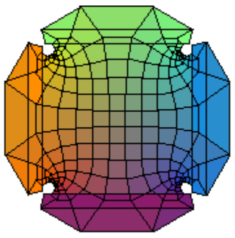


Figure 19: Coloring the figure 15.

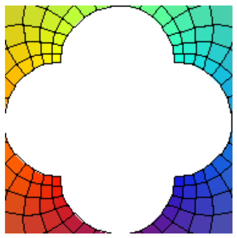
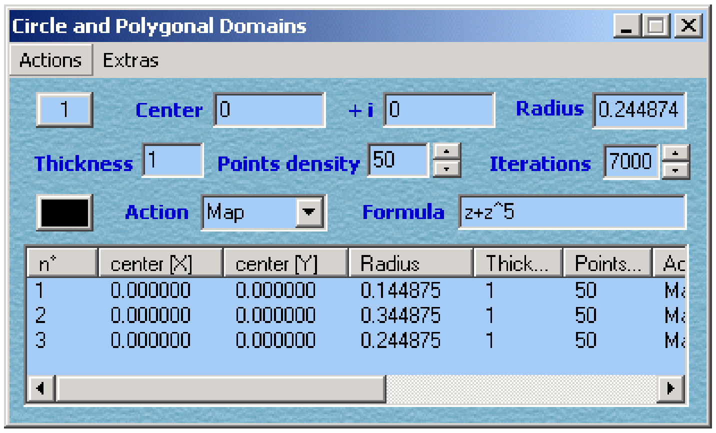


Figure 20: Coloring the figure 17.

Figures 19 and 20 are colored versions of figures 15 and 10 respectively. The same colors distribution, as in figure 20, points to *conformality*: a property holding locally for iterates of $f(z) : z+z^5$, but globally over the Riemann sphere for the function $f(z) : 1/\overline{z}$.

4 Circle Domains : viewing the local dynamics

4.1 Generate the domains



1 Center 0 +i 0 Radius 0.244874

Thickness 1 Points density 50 Iterations 7000

Action Map Formula z+z^5

n°	center [X]	center [Y]	Radius	Thick...	Points...	Ac
1	0.000000	0.000000	0.144875	1	50	Ma
2	0.000000	0.000000	0.344875	1	50	Ma
3	0.000000	0.000000	0.244875	1	50	Ma

Figure 21: Managing the circle domains for the experiment at section 4.3.

The 'circle domains' layer yields the possibility to watch the dynamics of a (possibly iterated) complex function by fixing a finite set of points along a circle with given center and radius, so that the n^{th} images, under that given functions, are plotted. According to the window in figure 21, the behavior of a circle domain is ruled by the 'Action' setting, collecting four types: 1) *None*; 2) *Draw*; 3) *Map*; 4) *Contour*. No action is performed for type 1), whereas 2) just makes the circle be simply plot, that is, as the locus of equidistant points to a center. Type 3) computes, by the associated[8] formula, and draws the orbit of each point along the circle of the given domain. In this case, the 'points density' parameter allows to achieve lower or higher resolutions, depending on the number of points along the boundary of the circle domain itself. The type 4) is a variant of 3) but it plots only an arbitrary subset of images, which can be chosen by selecting the 'Range' entry in the 'Extras' menu at figure 21.

4.2 Example : iterated circle domains about the Julia set

Let $f(z) : 3z^3 - z$ and a sufficiently small circular neighborhood, with 'Contour' action, of a repelling point of the Julia set. Such region is tracked by toggling

[8]The default option is to pick up the same formula as appearing in the plot bar.

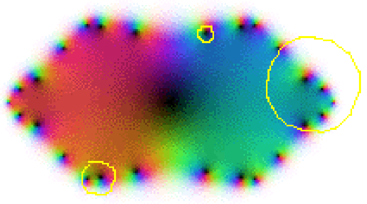


Figure 22: Plotting the iterated circle domains about the Julia set.

on the control 9 (see subsection 2.4) in the upper toolbar of the program. The smallest circle in figure 22 has been assumed as the starting one. Then just two sampling iterates of that circle are computed. The iterated circles are sequenced by larger radii. According to the theory, Julia set points do not converge to the attracting fixed ones and thus the family of iterates is said to fail to be normal there.

4.3 The Fatou-Leau flower via Circle Domains

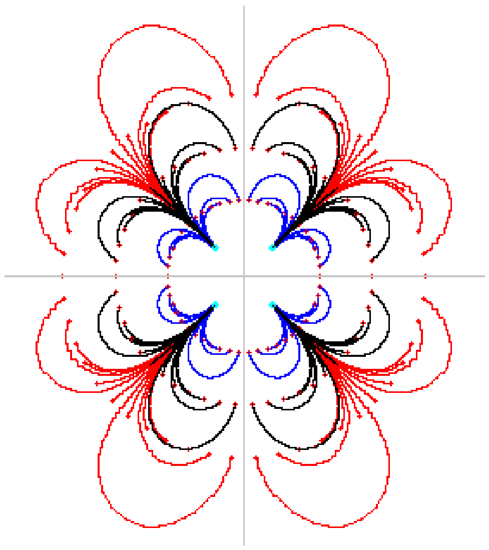


Figure 23: Tracked orbits inside the Fatou-Leau flower for $z + z^5$.

I will consider again the example of a Fatou-Leau's flower (see figure 23), generated by iterating the function $f(z) : z + z^5$, as in subsection 3.3. Let three concentric circles of small radius centered at the origin, in order to highlight the behavior inside the attracting and repelling regions. The starting points are distributed on these circles, whose action was set at 'Map' for them all. By means of the management window in figure 21, the number of points can be arbitrarily set by the user so to have more or less dense grids. In addition, a number of independent parameters relate to each circle domains, formula included. Such last feature allows, for example, to set *infinitesimal perturbations* of a given map and associated it to the circle domain so that the behavior of several maps can be examined about a singularity at a time and simultaneously inside the same window. Once circle points are iterated and their orbits are tracked, figure 23 comes up.

5 Equi-potentials

5.1 The method

Borrowed from the theory of differential equations, this method works locally and can play a very useful role in complex analysis and dynamics too. Potential level curves (also said streamlines, see [8]) measure the work of a function inside a given region about a point, here defined '*stream fixed point*'. This concept can be also applied to iterated maps. Streamlines lie inside the image space and the method related plots the corresponding domains. In general, it is technically possible to apply it arbitrarily to any stream fixed point of the plane, but it is especially useful to investigate on the neighborhoods of the singularities, such as poles, fixed or critical points.

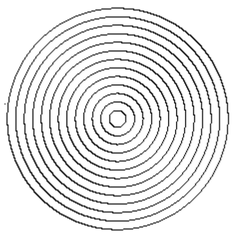


Figure 24: Radial streamlines.

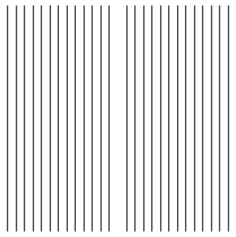


Figure 25: Vertical streamlines.

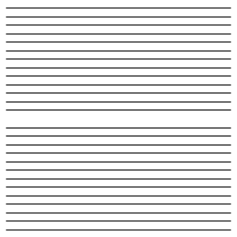


Figure 26: Horizontal streamlines.

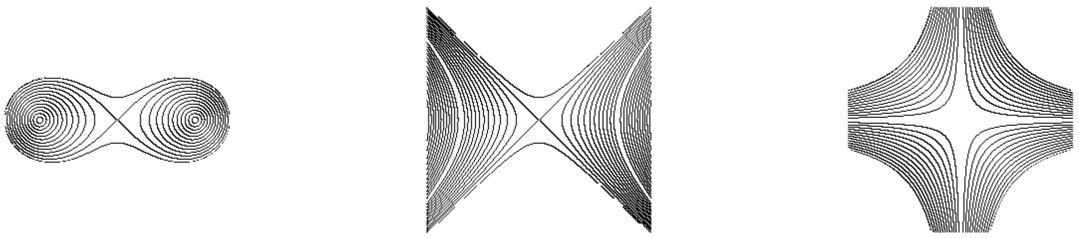
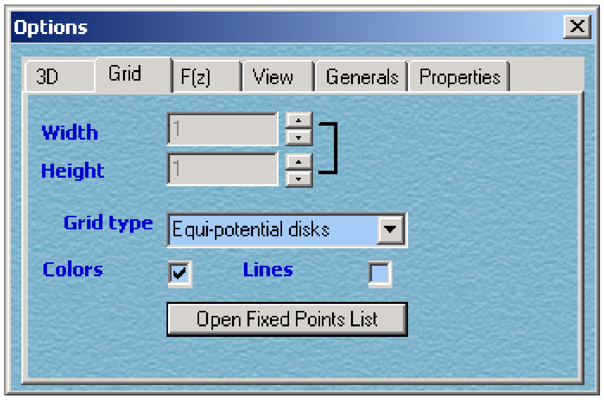


Figure 27: Mapping figures 24, 25, 26 by $f(z) : z^2 - 1$ respectively.

One begins to set a number of streamlines in the neighborhood of the stream fixed point; each streamline separates two levels/regions. Complex Mapper first opens a window for grid management (fig. 28) and then another window (fig.



Width: 1
Height: 1
Grid type: Equi-potential disks
Colors: ✓
Lines: ☐

Figure 28: Managing the rendering by equi-potentials.

30) to set both the point position and the related neighborhood extension. This opens up to a *streamlined domain*. Complex Mapper features three streamlines distributions (figures 24, 25, 26) by: 1) concentric disks ; 2) vertical lines ; 3) horizontal lines. Given two points z and z_1 inside the streamlined region, the images $f(z)$ and $f(z_1)$ are computed and, if the latter ones belong to different levels, then a point (indicating the intersection of an ideal line, linking $f(z)$ and $f(z_1)$, with a level curve) is plot, otherwise no action is performed. The function $f(z) : z^2 - 1$ applies to the three types of streamlined regions, which are located about the origin. $f(z)$ has a super-attracting cycle of period 2 with a repelling point p_1 at $(0, \ -1)$ and a super-attracting p_2 at $(0, \ 0)$, and two repelling fixed points at $\dfrac{1 \pm \sqrt{5}}{2}$.

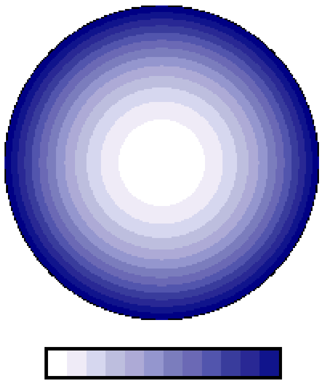


Figure 29: Gradient Color Streamlines about the repelling point $(0, \ -1)$.

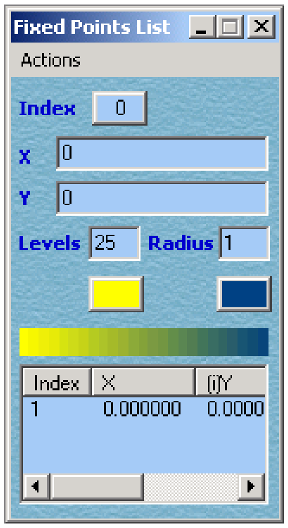
The original lines are deformed under the action of $f(z)$. Figures 27 really to show the preimages of f, that is, the *forward*[9] images of $f_{-1} : \pm\sqrt{w+1}$ about the critical point of the super-attracting cycle of period 2 and assumed that the stream fixed point is at the origin (the critical point). In this case, only figure 27 provides useful information: it is clear that the origin acts as a ramification point having two preimages at 1, which are the repelling periodic points. Figure 29 shows the streamlines about one such periodic point: as expected, the radius of the concentric circles gets larger and larger along the path leading to the stream fixed point itself, because the divergence

[9]The term '*forward*' refers to images with positive iteration index, '*backward*' for negative.

rate increases. In order to improve a clear reading of these figures, Complex Mapper can add color and then match original regions and their images. This helps to know their behavior and find the location. Coloring is achieved by means of a gradient, as one start and one end color tones (see figures 30 and 29) are chosen. Technically, the coloring algorithm checks the level of the n-th image points $z_n = f^{\circ n}(z)$ and sets the related gradient color, as shown in figure 29. For example, with regard to tones distribution in figure 29, white denotes the domains being closer to the fixed point than the blue ones. Hence coloring plays also a useful role to determine the location of streamlined domains, anyway it is not remarkable as lines because it is just applied to distinguish levels. Contextually, I applied the method to complex analysis.

5.2 Other examples : Siegel disks and Herman rings

The following example will show the streamlines method applied to another special map for holomorphic dynamics. Let $f(z) : e^{2\pi i\theta}z + z^2$, where $\theta \in \mathbb{R}\backslash\mathbb{Q}$ and it satisfies a diophantine condition of order κ. According to the theory, such a map has an *irrationally indifferent fixed point.*



Index: 0
X: 0
Y: 0
Levels: 25 Radius: 1

Index	X	(i)Y
1	0.000000	0.0000

Figure 30: Setting up a center for streamlined neighborhoods.

The iteration of a sufficiently close neighborhood of this fixed point yields a *Siegel disk*, that is, a rotation invariant and simply connected domain about the fixed point and which is conformally isomorphic to a disk. Figures 31 and 32 show both the filled-in Julia set and a blow-up of the neighborhood of the Siegel disk. Without regard to the computer approximation (this is why the analytic investigations on formulas yield real evidence. Figures cannot!), the reader can see, especially in the blow-up at figure 32, that the distance between the concentric disks - images of $f^{\circ n}$ - is nearly the same; this suggests that no attraction or repulsion occurs within this bounded region, it is just a rotation.

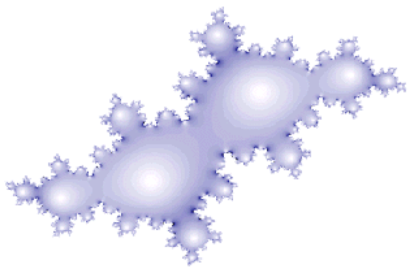


Figure 31: Streamlined filled-in Julia set for a polynomial with a Siegel disk.

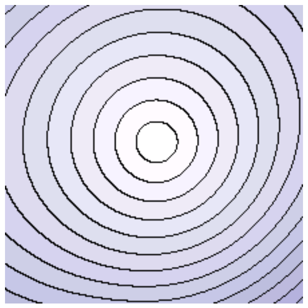


Figure 32: A blow-up about the Siegel disk with streamlines.

Holomorphic dynamics collects another class of rotation domains: they are doubly connected and defined '*Herman ring*'[10]. They are conformally isomorphic to an annulus $\mathcal{A}_r = \{1 < |z| < r\}$. It is proven that there are no Herman rings for maps whose degree is smaller than 3. Complex maps with Herman rings are at least cubic and which can be rewritten as

$$f(z) : e^{2\pi i\theta} z^2 \frac{z-4}{1-4z},$$

where $\theta \in \mathbb{R}\backslash\mathbb{Q}$ satisfies a diophantine condition, like for the simply connected case. Here $\theta = 0.6151732\ldots$ approximates the rotation number $\frac{\sqrt{5}-1}{2}$. As $f(z)$ is iterated, one obtains the figure 33, the blow-up of a neighborhood about the irrationally indifferent fixed point and where the Herman ring is located. In figure 33, an orbit (whose start and end points are painted red and yellow respectively) was tracked in order to evince the rotational character about the ring.

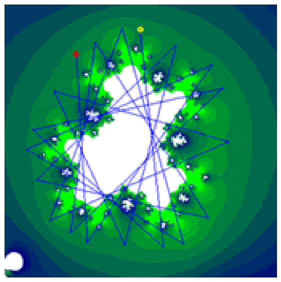


Figure 33: Stream domains about an Herman ring with tracked orbit.

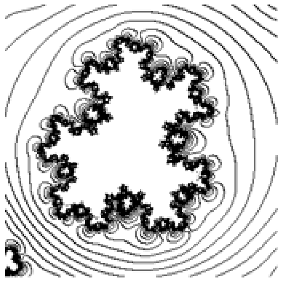


Figure 34: Black and white version of figure 33 with streamlines.

[10]In honor of Michel Herman (1942-2000), who developed a related and rigorous theory during 70's and 80's.

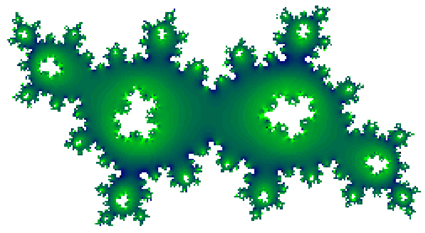


Figure 35: Filled-in Julia set with Herman ring via *colored* streamlines.

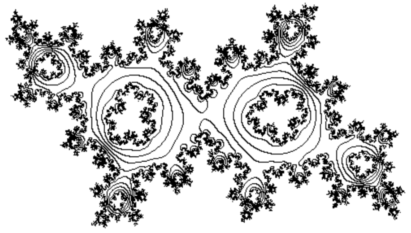


Figure 36: Filled-in Julia set with Herman ring via *blank* streamlines.

5.3 Several stream fixed points

Thanks to the multi-layered structure of its graphic tools, Complex Mapper allows to plot different visualization methods, piled up as in figures 37 and 38, where the methods of '*domain coloring*' and the '*streamlined domains*' have been simultaneously applied. Let $f(z) : z^4 - 1$. In figure 37, the fixed points of $g(z) : f(z) - z$ are computed and plotted. The distribution of the successive streamlines about such points means that they are all attracting. The streamlines show the attracting vector fields for each point and the transition zones where there is an equilibrium of the attracting forces of the fixed points. Here the trick. Figures as below can be obtained by setting a point between all those ones to be checked. For example one point, with domain radius 2, is set at the origin. Since this domain includes all other fixed points, all of them will be involved in the investigation.

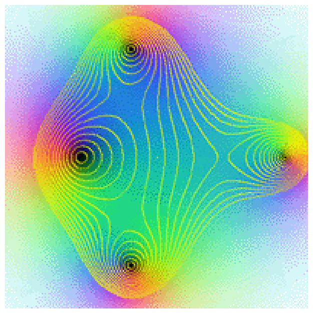


Figure 37: Streamlines about the fixed points of $f(z) : z^4 - 1 = 0$.

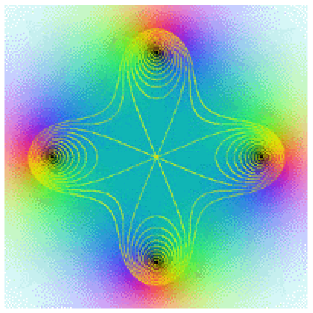


Figure 38: Streamlines about the roots of $f(z) : z^4 - 1 = 0$.

5.4 More than one streamlined domain

Complex Mapper gives the chance to input more than one stream fixed point and to compute more than one streamlined domain per figure. This is useful in several cases, when there is more than one singularity and then the simultaneous look at each point is required. Figures 39/A, 39/B and 39/C depict the streamlined domains computed for the fixed points of

$$z_{n+1} = g(z_n) : z_n - \frac{f(z_n)}{f'(z_n)},$$

where z_n belongs to a sequence of iterates and the formula is the transformed function arising from application of Newton-Raphson's method to a given $f(z)$. The fixed points of $g(z)$ are the roots of $f(z)$. Here let $f(z) : z^2 - 1$. The white '*holes*' indicate that there are domains of points which does not seem to be attracted to the fixed points, with regard to the localization parameters of the streamlined domains. But, for larger values, the holes shrink to points. Here two streamlined domains, whose stream fixed points are those ones of $g(z)$, i.e. $F_1(-1, 0)$ and $F_2(1, 0)$, are defined.

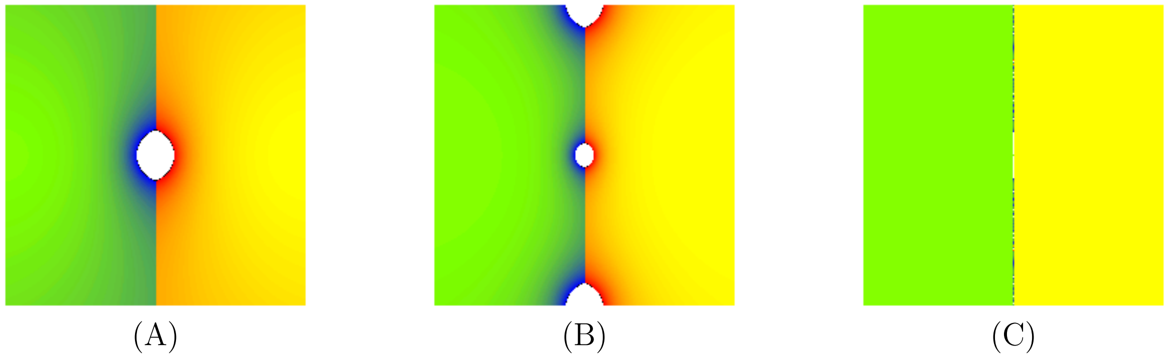


Figure 39: Streamlines for $g(z)$ at 1, 2 and 20 steps respectively.

Figure 39/A shows the first iterate of $g(z)$: colors of streamlines are still distinct. Figure 39/B shows the second iterate: as closer as points get to the attracting fixed point of the related basins, domains (and thus colors, due to the same limit value) tend to merge into one, which is the color of the potential level 0 at last. Note that white holes are increasing in number, thus there are more and more points escaping the basins of attractions: they belong to the Julia set, located on the imaginary axis in this example. Finally figure 39/C shows the plot associated to a large iteration index: the points inside the investigated region converge to the fixed points of the basins they belong to, but the iterated images of the Julia set points are trapped inside the set itself. Both loci are then said to be completely invariant.

6 Domain Coloring

6.1 Some introductive colorimetry

The term '*domain coloring*' was coined by professor Frank Farris, according to several web-references. For deeper technical information, refer to [9]. We start with the *RGB cube*: a 3-D geometrical model of the color *additive* system, i.e. a three-vectors space where each component is a color among Red, Green, Blue. Lack of light (and then of colors) yields the black. Full light intensity means that all primary colors are summed up together: Red + Green + Blue = White. If only one component is *partly* added to black, a shade comes out; shades of the three components, added together, yield color sets whose range depends on the discretization order of the given graphics system. For example, most computers *true color* system is based upon the 24-bits codification: each additive primary component is associated to one 8-bits value. Thus $2^8 = 256$ shades of components can be painted and $256^3 = 16.777.216$ colors could be generated. In addition, I mention a relation between the additive and the *subtractive* system, said CMY (Cyan, Magenta, Yellow): if two components of the RGB system are mixed, then yield one component of the CMY system. For example, Red + Green = Yellow, or Blue + Green = Cyan, Red + Blue = Magenta. This all is summed up in figure 40.

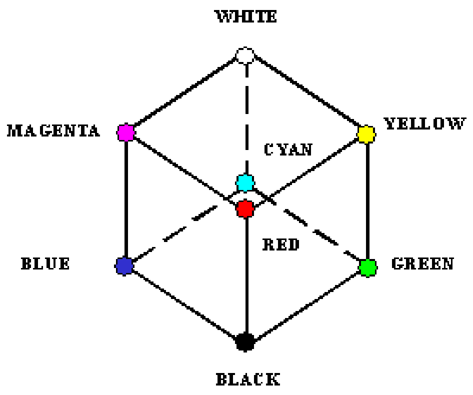


Figure 40: The RGB color model.

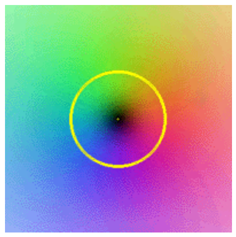


Figure 41: Domain coloring on $\mathbb{C}$. Unit circle is painted yellow.

Figure 41 shows a neighborhood of the origin. According to the discussion before, colors get closer to black as points approach to the origin $(0,0)$. Whereas red, green and blue regions depart along three distinct directions; they tend to turn into white since both real and imaginary components of the associated complex points increase to infinity. The color bands in figure 41 result from a mix of effects from the coloring power of the video-card (which is not sufficiently accurate) and the discretization of the grid to be plotted, where distinct points merge into the same one and thus they are painted the same.

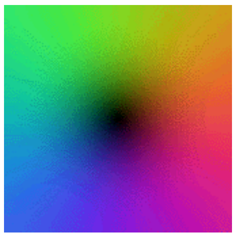


Figure 42: Shadings about the origin.

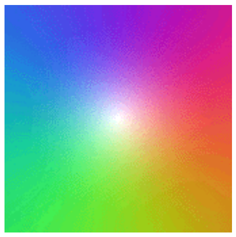


Figure 43: Shadings about ∞.

The 'domain coloring' method sets a *one-to-one* map between the RGB cube and the complex plane, so that each complex point matches only one color. Thus the algorithm paints all complex plane according to the values distribution. The original algorithm was slightly modified by the author so that now smaller complex values are painted by shades of black, whereas largest values are painted white: it fixes an intuitive correspondence between, for example, the modulus of a complex number and light intensity. For sake of simplicity, I gave a related example by displaying together the identity $f(z) : z$ and the inverse map $f(z) : 1/z$, both computed about the origin in figures 42 and 43 respectively. The neighborhoods of the lowest (the origin) value and the largest one (the point at infinity) are visible and clearly explained.

6.2 Applications

This method was originally developed for Complex Analysis, where has been widely applied. The author extended it to Complex Dynamics too. The concept and application remains the same because $f^{\circ n}(z)$ is again a complex map such as $f(z)$. In this theoretical context, the method is helpful to show the nature and structure of Julia sets as well as other local configurations. Below some related figures are shown.

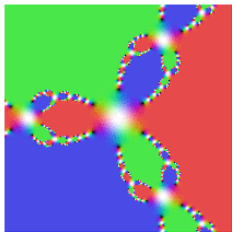


Figure 44: Newton's method for $z^3 - 1$.

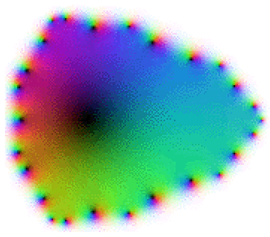


Figure 45: Fixed points of $f^{\circ 3}(z)|f : z^3 - z^2$.

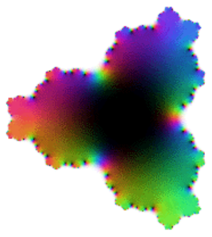


Figure 46: Fixed points of $f^{\circ 4}(z)|f : z + z^4$.

Figure 44 shows the iterations of $g(z)$, which a rational map deduced from Newton-Raphson's method applied to $f(z) : z^3 - 1 = 0$. The iteration process has been deliberately stopped after 5 steps so Julia set points are sufficiently visible like the basins of convergence to each root of $f(z)$. Indeed, according to colors, this Julia set consists of poles (white means full intensity = very large values tending to ∞). The Julia set is here the union of all poles generated by all the successive iterates of $g(z)$.

Figure 45 was obtained by plotting the formula $g(z) : f^{\circ 3}(z) - z = 0$, where $f(z) = z^3 - z^2$. The black points are obviously the roots of $g(z)$, that is, the fixed points of $f^{\circ 3}(z)$. The largest black region surrounds the neighborhood of the super-attracting fixed point, while other black regions denote the repelling fixed points of the Julia set. Here again, iterates were arrested at a very little iterative index, 3, for evincing the generation of the related Julia set. The formula $g(z)$ was computed by a new feature in Complex Mapper, allowing to generate *compound* functions, namely the 'Additional Substitutions'. As shown here, it first computes the iterates and then attaches a map to the resulting values. For fixed points, attach '$-z$' to a given $\zeta(z)$, so to have $\zeta(z) - z$: zeros are precisely the fixed points. Additional maps can be set or attached and then treated by the current visualization method as a new (compound) function. Figure 46 is similar to figure 45, but here $g(z) : f^{\circ 4}(z) - z = 0$, where $f(z) = z + z^4$.

Another application of domain coloring and additional substitutions is shown in figures 47, 48 and 49. In figure 47, the 'domain coloring' method displays the values distribution for the dynamical system generated by $f^{\circ 4}(z) = 0$, where $f(z) = z^2$. The black disk is the basin of points attracted to the origin. In the white region, iterates diverge to the origin while Julia set is differently multi-colored and localized on the unit circle.

Figure 48 shows the computation of $g(z) : f^{\circ 4}(z) - z$, whose zeros are fixed points of f and its iterates. Notice the repelling fixed points (contained in the Julia set) inflating along the unit circle and the super-attracting point at the origin.

Figure 49 shows the modulus of the first derivative of $f^{\circ 4}(z)$ whose values are 1 on the unit disk as expected. If compared to figure 41 (the identity map), the red shadings are close to the real value 1. Again, final conclusions should not be drawn from figures but from analytic computation.

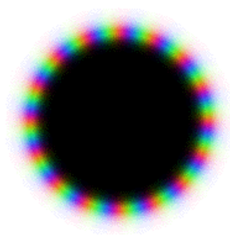


Figure 47: Displaying $f^{\circ 4}(z)$.

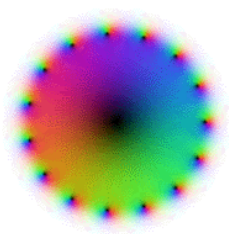


Figure 48: The fixed points of $f^{\circ 4}(z)$.

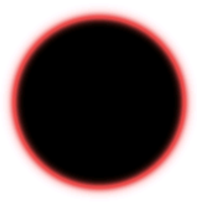


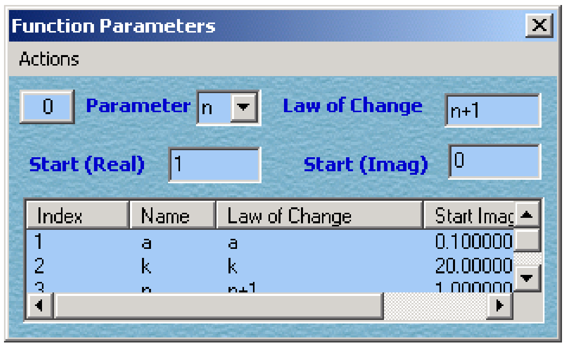
Figure 49: The first derivative of $f^{\circ 4}(z)$.

7 Series

7.1 Extending the original formula parser

This latest release implements an extended parser with parameters. This is very useful to compute series, where often each term includes at least one parameter. In the layers window, the user shall toggle on the 'series computation' layer and open the 'series computation' window (see figure 50), which displays the controls for setting the parameter values for each summand of the given series.

The initial value, the related formula (here defined 'law of change') are required to generate the succession of values (each one is submitted to the summand with the same index) for one parameter at each step. The input syntax is very easy: every letter, pointing to a parameter, shall be preceded by the 'p', so that, for example, the parameter 'a' shall be written as 'pa' for the given formula.



Function Parameters

Actions

0 Parameter: n

Law of Change: n+1

Start (Real): 1

Start (Imag): 0

Index	Name	Law of Change	Start Imag
1	a	a	0.100000
2	k	k	20.00000
3	n	n+1	1.000000

Figure 50: Parameters input window.

7.2 Examples of series.

In the next examples, I will set some parameters:

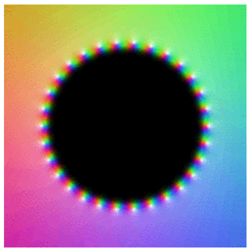


Figure 51: Series (Julia, 1913).

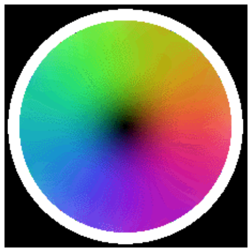


Figure 52: Series (Fredholm, 1890).

Name	Law of Change	Initial value
a	a (constant value)	0.1
k	$k+1$	1
n	n (constant value)	20

Table 1: Parameters Table for figure 51.

Inspired by [5], the following series $\zeta(z) = \sum_{n=1}^{20} \dfrac{a}{e^{2\pi i \frac{k}{n}} - z}$ is assumed and plotted in figure 51, using the following input : $\sum_{n=1}^{20} \dfrac{pa}{e^{2\pi i \frac{pk}{pn}} - z}$. As proven by Julia himself, the behavior of such series is defined the 'condensation of singularities', a term coined in [4]. The poles of $\zeta(z)$ are progressively 'condensing' along the unit circle as $k, n \to \infty$: they generate a Jordan curve (the unit circle) which splits the whole complex plane into two disjoint regions: one inside and the other outside the unit circle. Such Julia's study is historically interesting since it shows an analogy (but not proven if assumed by that author) between the natural boundary of the domain of analyticity for a series and the Julia set arising from the iterations of a complex function. In this sense, we might guess that the existence of similar situations presenting inside works of Analysis and Dynamics showed that both branches were convering to a same problem: *the study of the behavior for complex functions inside domains enclosed by boundaries of singularities.* Julia will deepen the role of the latter some years later (1918).

Figure 52 shows the plot of the series $\lambda(z) = \sum_{n=1}^{\infty} \dfrac{z^{n^2}}{2^n}$ through the method of 'domain coloring'. $\lambda(z)$ was first investigated, in 1890, by Fredholm (see

Name	Law of Change	Initial value
n	$n+1$	1

Table 2: Parameters Table for figure 52.

[3]) who proved that it converges throughout the open unit disk (shaded tones of red, green, blue) and it is infinitely differentiable on the unit circle (white region), which is also the *natural boundary* for this series.

8 3-D maps

8.1 Technicals

A good display of complex functions can be also achieved through a 3-D system. The related graphic algorithm is easy to code.

For each complex point, one associates a triplet u, v, w where u and v are the real (latitude) and the imaginary (longitude) component respectively and w is a real number which could be deduced from $f(u + iv)$ in a variety of ways: for example, it could be the real or imaginary component, the angle (argument), distance from a given point, the modulus, the complex potential or finally a value computed from the coordinates of the point inside the RGB cube. Basically, such 3-D plot is an histogram of values where every column, which piles up as higher as larger the real value of an image value is, is based at the point on the plane. Hence the 3-D feature allows to have a visual analogy between the numerical distribution of complex values and the space, easier to understand and more intuitive than simple 2D outputs.

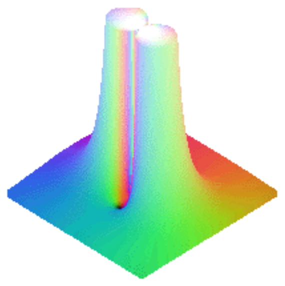


Figure 53: $w = \sin(1/z)$.

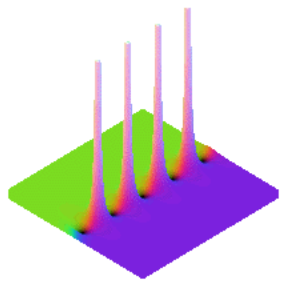


Figure 54: $w = \tan(z)$.

In figure 53, the behavior of $f(z) : \sin\dfrac{1}{z}$ about the origin is shown. Note that values quickly increase due to the essential singularity at the origin itself, although the wild behavior is not clear in this version.

Figure 54 shows the periodic behavior of $f(z) : \tan(z)$, with zeros and poles alternating along the real axis.

$f(z) : \ln(z)$ about the origin appears in figure 55. This point is a logarithm branch, where neighboring image values never get back to the original one, no

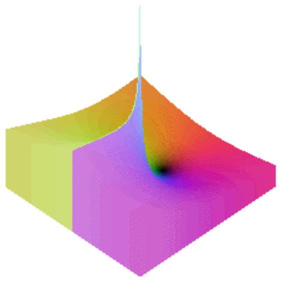


Figure 55: $w = \ln(z)$.

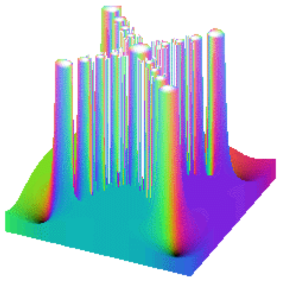


Figure 56: Iterates of $w = z^4 - 1$.

matter how many loops are made around the origin. The same figure shows a rough change of colors right at the branch cut departing from the origin itself.

Figure 56 shows the Newton's iterations of $f(z) : z^4 - 1 = 0$; the iteration index was arrested[11] at 5 in order to show the generation of the Julia set, here consisting again of poles.

All 3-D figures, from 55 to 58, were plotted by using the modulus of w.

9 On the Riemann sphere: a global method

9.1 Technicals

The Riemann sphere $\hat{\mathbb{C}}$, onto which the complex plane $\mathbb{C}$ is *compact*[12]-ified (i.e. $\hat{\mathbb{C}} = \mathbb{C} \cup \infty$), is assumed to have unit radius and the South pole is at the origin: therefore the sphere can be nested into a cube of unit side, as shown here below.

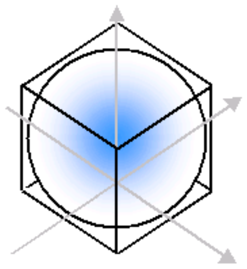


Figure 57: The sphere nested into the unit cube.

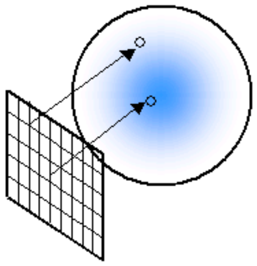


Figure 58: The screen onto the sphere.

This algorithm travels from the sphere to the plane. As proven by some initial tests, a reverse approach causes an incomplete drawing because sphere

[11]Higher indexes relate to very higher inflation of Julia set points in the same region: thus the 3-D map would be confusing in this case.

[12]A set is said to be *compact* when it is closed and bounded.

points getting closer and closer to infinity are thus associated to points on the plane at a longer and longer distance. Therefore they are very hard to reach and compute. But the sphere it is a bounded model and then the localization of these latter points can be very easily achieved. The algorithm is again easy here and very similar to the method of 'domain coloring'.

Imagine the computer screen as a pixelized view of part of the sphere. First associate each pixel of the view to a sphere point, as in figure 61. For more simplicity, let the pixels of screen view be associated to a face of the unit cube above. Then let that point run to the sphere along an orthogonal trajectory (as indicated by the black arrows). As the associated point comes onto the sphere surface, it will be defined '*hit point*'. The computed distance to the sphere center will be about 1: an error tolerance value shall be applied because of

$$d = \sqrt{\texttt{(square-of-the-distance)}} = 1$$

only if the radicand is also 1; it is rather improbable to find surface points due to the decimal approximation involved during finite digits computing, but one could come very close to it by such tolerance value. So the hit point is mapped onto the complex plane by *inverse stereographic projections.* Given a sphere with arbitrary radius r and a (u, v, w) point on its surface, then the (x, y) coordinates of the projection point on the plane are:

$$x = \frac{u}{2r - w}, \qquad y = \frac{v}{2r - w}.$$

The coordinates x, y can be interpreted as the real and imaginary components of the complex number $z = x + iy$. So the related image will be computed by means of the given (possibly iterated) complex function. Finally, the associated color will be yielded by the 'domain coloring' algorithm and then plot at the starting sphere point.

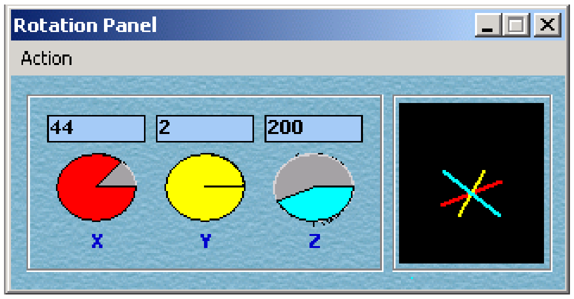


Figure 59: Rotation angles for 3-D models.

Hit point are easily rotated by given input angles, related to the X-Y-Z axes and displayed in figure 59. Such window also relates to figures in section 8.

9.2 Some global examples

The Riemann sphere method is labelled '*global*' because it allows the user to look at the behavior of a complex function for any input value all over the sphere, that is, including the point at infinity and, mostly, the related neighborhoods (unlike local methods whose approach to them is limited).

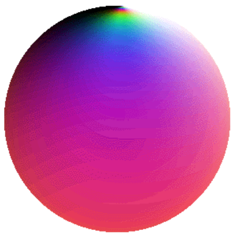


Figure 60: A first view of $f(z) : e^z$ on the Riemann sphere.

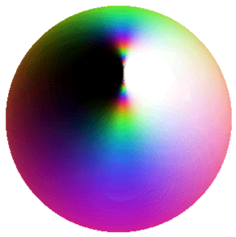


Figure 61: A rotated view of figure 60, pointing to a region about ∞.

Figures 60 and 61 show two plots for the simplest exponential map e^z. Figure 61 shows a vicinity of the point at infinity, where a strange behavior is observed: the north pole is an essential singularity for that exponential map. Because Complex Mapper can also work with iterations, it is natural to want to observe their behavior all over the Riemann sphere, something which has long sought since the first analytical researches of Koenigs (1883-85) and only achieved analytically by Fatou and Julia in the beginning of XXth century.

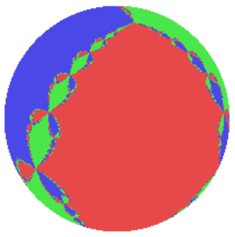


Figure 62: Newton's method for $f(z) : z^3 - 1$.

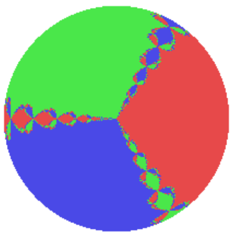


Figure 63: A second view at the origin.

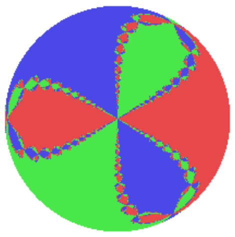


Figure 64: About the point at ∞.

Figure 62 shows the 'domain coloring' plots of Newton's method applied to the cubic equation $f(z) : z^3 - 1 = 0$, which depicts the roots of $f(z)$ as well as the related basins of convergence. Every basin is associated to one color because all of its points converge to the same fixed point - a fixed point for the Newton method, but a root for $f(z)$.

Figure 63 shows the behavior of the method about the point at infinity, while a neighborhood of the origin appears in figure 64.

9.3 Two applications to transcendental dynamics

Under the term '*transcendental dynamics*', one intends the iteration of non-rational maps, i.e. the ratio between two polynomials, where at least one is non-algebraic: loosely speaking, any polynomial including an exponential, or logarithmic, or trigonometric function in the given variable of the polynomial itself. Let $\mathcal{F}$ be the 'Fatou set', i.e. the union of all basins of convergence. Such dynamics show cases which cannot occur for rational maps: the '*Baker domains*' and the '*wandering domains*' (see [11]). Here the support of the sphere much more useful than elsewhere. The former ones are bounded invariant components $U = f(U) \subset \mathcal{F}$ such that all orbits inside U converge to the point at infinity: see figure[13] 65, the Baker domain is the white one on the top.

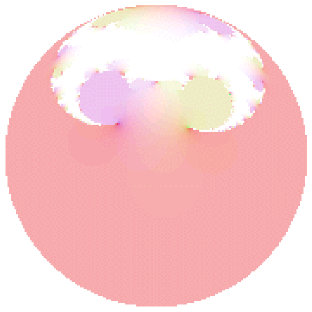


Figure 65: A Baker domain for $f(z) : e^{-z} + z + 0.0001$.

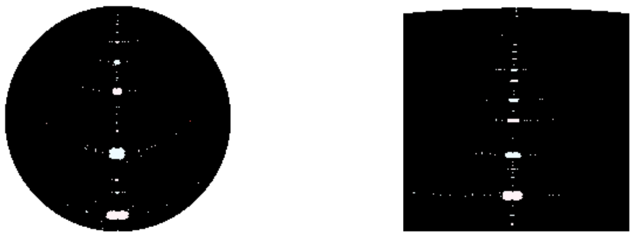
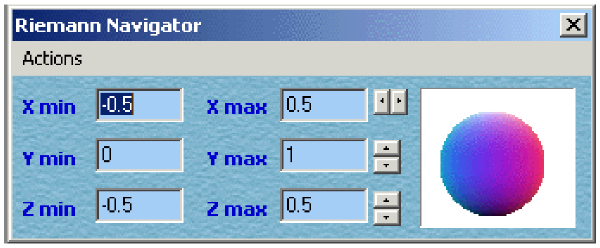


Figure 66: The holes are the wandering domains for the iterates of the function $f(z) : z + \sin(2\pi z)$. A neighborhood of ∞ is displayed on the right. Sphere is almost painted black because most orbits converge to the origin.

Wandering domains are components $U \subset \mathcal{F}$ such that $f_m(U) \cap f_n(U) = \emptyset$ for any integer $m \neq n$, with $m, n > 0$. By means of the Riemann sphere visualization, Complex Mapper allows to look at the *complete* behavior of such functions. Right due to their no-intersection property, endlessly enjoyed by the wandering domains, the Riemann sphere shows fits very well. But it does also for Baker's domains, because of such kind of domain is often localized in the neighborhood of ∞. Thus it can be viewed here in its entirety.

For example, figure 66 is a blow-up of a region about ∞. Complex Mapper supplies an easy navigator to walk along the sphere, by inserting the 3-D coordinates of the region to be zoomed (see figure 67).

[13]Sphere was rotated by $60°|x$, $90°|y$ for better visibility.



X min -0.5 X max 0.5
Y min 0 Y max 1
Z min -0.5 Z max 0.5

Figure 67: The dialog window to blow up on the sphere.

10 Discussion and conclusion

The program Complex Mapper intends to provide a visual help to researchers, educators and students in complex analysis and dynamics, that is, its use is exclusively mathematical. All figures but 12, 40, 57 and 58 have been computed and plot using Complex Mapper. They are explicitly offered as probative of mathematical facts: in fact the best Mathematics requires that the evidence of its statements shall be proven rigorously via the analytical approach. The development of this program will continue along the next versions.

11 Acknowledgments

I am indebted to Daniel S. Alexander and Felice Iavernaro for editing a first draft of this article and for their continuous support as friends and mathematicians.

References

[1] Alexander D. S., Iavernaro F., Rosa A., *Early Days in Complex Dynamics.* In preparation.

[2] Fatou P., *Sur les équations fonctionnelles*, Bulletin Soc. Mat. Fr., 47, 1918, pp. 161-271; 48, 1920, pp. 33-94, pp. 208-314.

[3] Fredholm J., *Om en speciell klass af singulära linier*, Stockh. Öfv., 1890, pp. 131-134.

[4] Hankel H., *Untersuchungen ber die unendlich oft oszillierenden und unstetigen Funktionen*, Math. Annalen, 20, 1882, pp. 63-112.

[5] Julia G., *Sur les lignes singulières de certaines fonctions analytiques*, Bulletin de la Société Mathématique, 41, 1913, pp. 351-366.

[6] Julia G., *Mémoire sur l'itération des fonctions rationnelles*, Journal de Mathématiques Pures et Appliquées, (8), 1, 1918, pp. 47-245.

[7] Leau L., *Étude sur les équations fonctionnelles à une ou à plusieurs variables*, Gauthiers-Villars, 1897.

[8] Needham T., *Visual complex analysis*, Oxford University Press, 2000, p. 453.

[9] Richardson J. L., *Visualizing quantum scattering on the CM-2 supercomputer*, Computer Physics Communications, 63 , 1991, pp. 84-94.

[10] Rosa A., *Inwards to Chaos and Complex Mapper : two graphical interfaces for simulations in complex analysis and dynamics*, Electronic Journal of Differential Equations and Control Processes, St. Petersburg, Russia, 3, 2004.

[11] Sullivan D., *Quasiconformal homeomorphisms and dynamics: I. Solution of the Fatou-Julia problem on wandering domains*, Annals of Mathematics, 122, 1985, pp. 401-418.

Alessandro Rosa
c/o Locatelli,
Via Cappuccini, 116
I-72100 Brindisi (Italy)
e-mail: `zandor_zz@yahoo.it`