

# Visualization of Complex Mappings

Gustavo M. Goñi and Claudio Delrieux

Departamento de Ing. Eléctrica y Computadoras

Universidad Nacional del Sur, Av. Alem 1253, (8000) Bahía Blanca, ARGENTINA

e-mail: [claudio@acm.org](mailto:claudio@acm.org)

**Abstract**—We present the implementation of a practical system for complex variable mapping visualization (conformal transformations in the complex domain). The system is aimed to help the understanding of formal concepts in complex variable analysis (zeros, poles, singularities, multiplicity). The system is flexible and interactive, and is also useful for understanding the behavior of complex dynamic systems, the arising of fixed points, strange attractors, and other topics in the theory of fractal sets and chaotic dynamic systems. Finally, we show that it can be also useful as an image processing tool.

## I. INTRODUCTION

One of the most difficult issue in teaching and learning complex variable functions is the lack of a good graphical representation. Functions of a real number  $y = f(x)$  are easily represented by Cartesian graphs, and therefore their behavior can be easily visualized. This is true even for two dimensional real functions  $z = f(x, y)$ , which can be represented as a manifold within a 3D space. However, in the complex domain, a function  $w = f(z)$  (which can be represented as  $u + iv = f(x + iy)$  [1], [2], [3]), needs two dimensions both for representing the domain and the range, and therefore we need four spatial dimensions if we wish to represent it with a generalization of a Cartesian map.

Different alternatives have been explored for representing graphically complex functions. For instance, a graph  $\mathbb{R}^2 \rightarrow \mathbb{R}$  can be performed, regarding only a real valued property of  $f$ , for instance  $\Re(w)$ ,  $\Im(w)$ ,  $|w|$ ,  $\arg(w)$  and many other. With these associations, it is possible to obtain a limited 3D representation of the behavior of the function. Another possibility is to visualize the map  $w = f(z)$  as a vector map. The interest of this approach lies in the fact that there exists a considerable literature about visualizing vector fields and dynamical systems, and those results may be applicable also in visualizing complex maps. However, regarding a complex map as a vector field hides the crucial fact that the issues of interest in this two areas are quite different, since in vector fields the most important visualization is of its phase diagram, and the features there arising (critical points, cycles, homoclynic trajectories and so on), while in the visualization of complex maps we are interested in the representation of poles, zeros and singularities in general.

Yet another possible approach is to apply techniques of scientific visualization for the representation of +3D maps by means of color maps. In our situation, we can color-code the complex domain (i.e., associate a 2D color

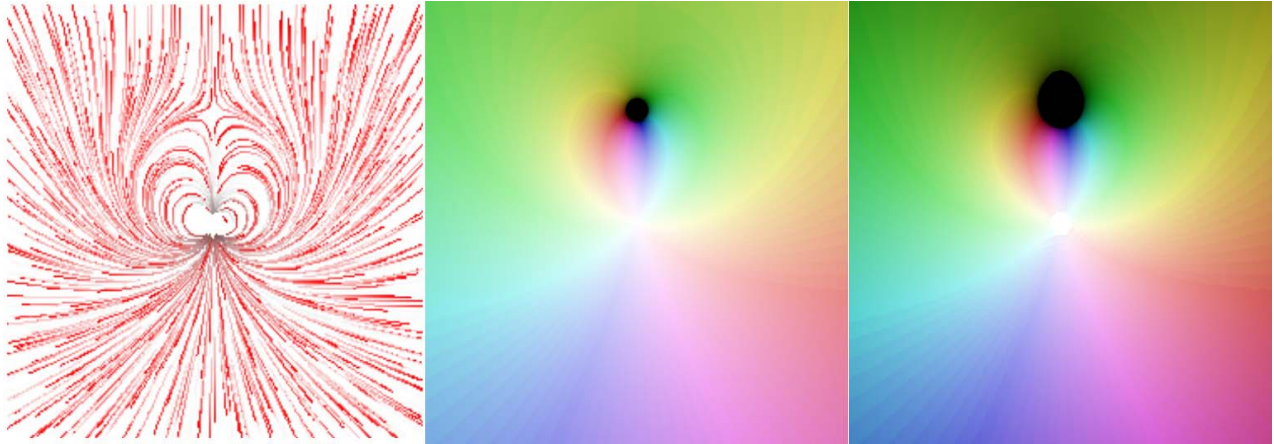


Fig. 1. The function  $w = \frac{z^2}{z - i}$  represented as (a) vector field, (b) color-coded, and (c) distorted image.

gamut that biunivocally represents the complex value of every point in the domain), and then represent the complex map simply mapping the color of every complex number  $z$  in the domain into the position  $w = f(z)$ . In this work we extend this idea in a novel way. Instead of using a color-coded map, we can use any arbitrary input image. This can be easily justified, since a color-coded map is implicitly an image where every pixel representing the complex number  $z$  holds the color that corresponds to the color coding of  $z$ . Our claim is that color-coding can be very representative of many features of complex maps (i.e. zeros and poles), but the overall behavior of the map is most of the times better exemplified using an input image that cleverly shows the features of the complex function.

In Fig. 1 we show the complex function  $w = \frac{z^2}{z - i}$ , represented using the last three mentioned techniques (vector field representation, color-coding, and image mapping). The vector field representation was generated using streamlines, but any other technique can serve to this purpose [7], [8]. The color-coding

technique in the example associates luminance to the modulus of  $z$  and chrominance to its argument. The input image in the third technique is simply a color-coded complex plane with the same associations as in the second technique. As was expected, the second and third examples produce a very similar result.

## II. COMPLEX FUNCTION PARSING

One of the indispensable elements of our interactive application is that the user should be able to introduce and investigate their own complex functions. This goal poses some difficulties, since the usual mathematical software that allows symbolic evaluation (v. g., Matlab, Mathematica, Maple) is usually too slow for generating the needed amount of data for real time visualizations. Then, we had to design an implement a parser and evaluator for complex functions.

This parser follows several stages. When the user introduces a complex equation in the user interface, for instance  $w = \frac{\text{sqr}(\text{mod}(z + 2))}{(z - i)}$ , the first stage corresponds to the syntactical analysis. If the result of this stage is that the

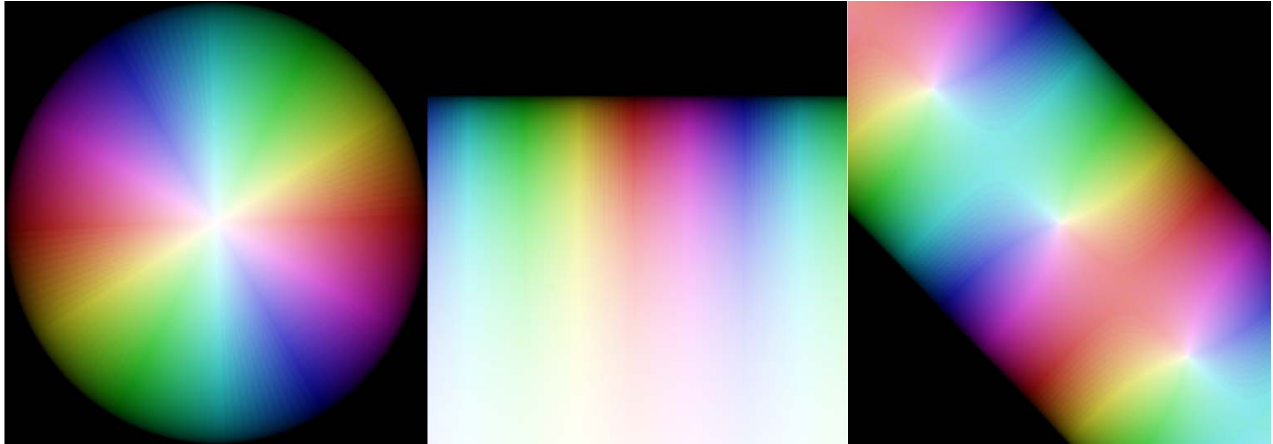


Fig. 2. Functions (a)  $w = z^2$ , (b)  $w = \exp(-iz)$ , y (c)  $w = \text{sen}((1 + i)z)$ , represented as a distortion of a bivariate map.

equation is syntactically correct, then the second stage converts the equation into a normal form that facilitates the execution performed by the evaluator. Then, the normalized equation is converted into a binary tree form, which is in fact the one recognized by the evaluator. The evaluator performs in a post-order recursive evaluation (i.e., first evaluates left and right subtrees, and after this the operator in the root is applied to the two results taken as operands).

Most of the times this evaluation is performed over the *inverse* of the complex function, at least, whenever we are using an input image and rendering the complex function as a distortion of this image. For doing this, the procedure is very simple. Take the complex value  $w$  associated to every pixel in the output image. Compute the preimage  $z = f^{-1}(w)$ . Then convert the complex value  $z$  into the pixel coordinates in the input image, fetch the color in that pixel, and finally paint the initial pixel in the output image with this color. In the next section we show how this procedure produces impressive results even using the standard color-coding schema explained in the

previous section.

### III. IMAGE DISTORTION

As stated in the introduction, there is an equivalence in the color-coded representation of complex functions and in our input image distortion schema, provided that the input image is the standard identity code. This input image is trivially generated assigning to every pixel the color resulting of the color-code of the complex number associated to the pixel. In Fig. 2 we show some examples of complex functions using this standard input image. It is remarkable how in this schema it is very easy to visualize singularities in the function.

However, we can use any *arbitrary* input image and apply the same distortion. If the input image is recognizable, therefore the visualization of the result of the mapping can be much more eloquent that using a color palette. In Fig. 3, we apply the functions  $w = z^2$ ,  $w = \exp(-iz)$  y  $w = \text{sen}((1 + i)z)$  to a recognizable image (a watch), and the resulting distortions shows very clearly the behavior of the function.

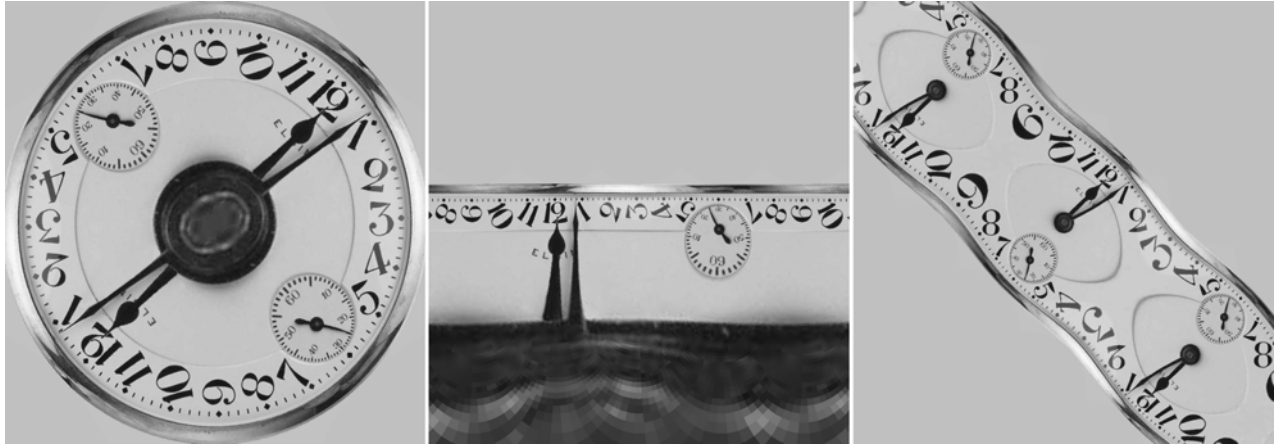


Fig. 3. The same functions of Fig. 2 applied to an input image.

The user interface also provides facilities for trimming the input figure. For instance, in Fig. 3 it is critical to the final result that the pixel at the center of the watch in the input image coincides with the complex number  $(0, 0)$ . This can be achieved by means of translations. It is equivalent, however, to translate the original image and to modify the function  $f$  by a translation term. For instance, if we displace the input image by  $(1, 1)$ , the result will be equivalent to visualize the function  $w = (z - 1 - i)^2$ . The same applies to scaling and rotation. This concept in particular is very important to convey while teaching complex analysis, and therefore a graphical interactive tool for visualizing this behavior can be very valuable.

#### IV. ADVANCED FEATURES

One of the most amazing set of mathematical objects arises from the iteration of complex maps. In effect, the Julia sets, thoroughly investigated since B. Mandelbrot's seminal research, arise as the fixpoint of the iteration of the map  $w = z^2 + C$ , for any complex constant  $C$ . In other words, the set of complex numbers

$Z$  such that if  $z$  is in  $Z$  then  $z^2 + C$  is also in  $Z$  is the *Julia set* associated with the constant  $C$ .

Usual computations of Julia sets are based on indirect properties (stabilities of orbits) that allow for faster rendering. However, iterating the mapping over and over until a practical fixpoint is obtained is much more compelling for understanding that the actual Julia set is a fixed point, and that the dynamic of the iteration is *independent* of the original image. In Fig. 4 we show some of the series of iterations of the complex mapping  $z^2 + C$  for  $C = -1$  over an arbitrary input image. The distorted image at step  $i$  of the iteration is taken as input image for step  $i + 1$ . Needless to say, our application program allows for an easy iteration in this fashion, either direct or inverse.

Another feature included in our application program is the ability for building animations using most popular codecs. One of the most useful purpose for animations is in understanding the effect of smoothly changing a parameter in the mapping (f.e., the translation term, or an exponent). For facilitating this,

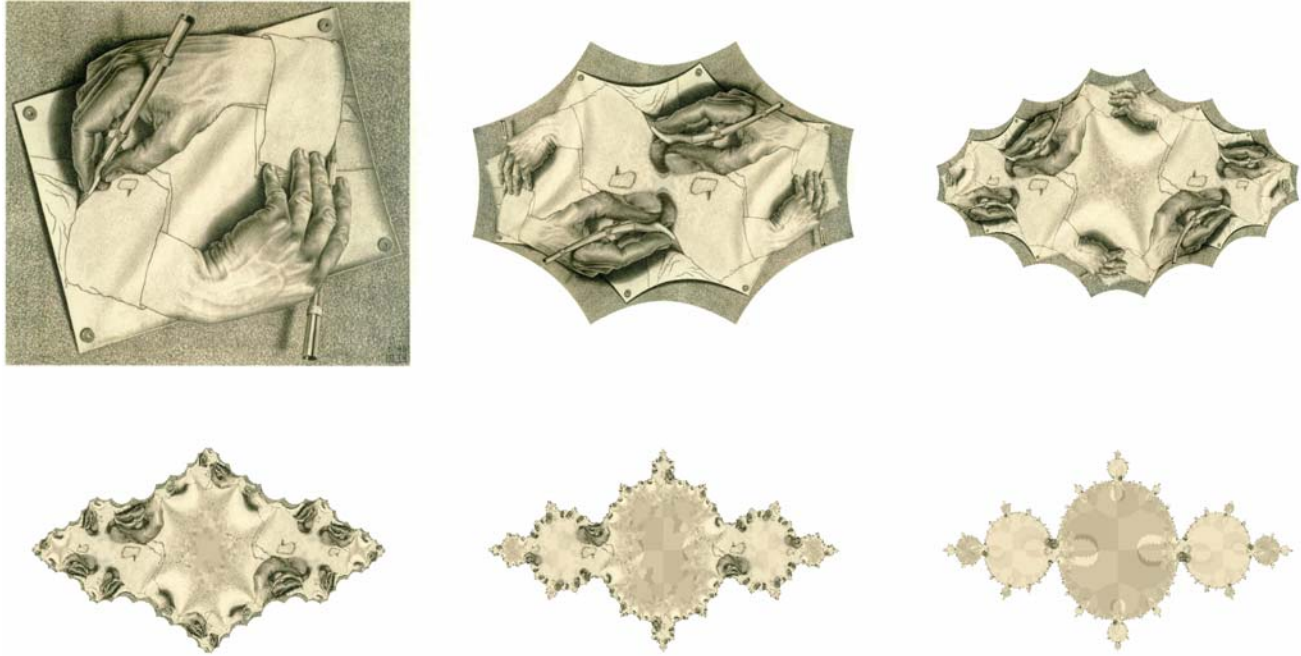


Fig. 4. Original image, and iteration of  $z^2 + C$  after 1, 2, 4, 8 and 16 steps.

the user may set any parameter for animation, the amount of steps, the frame size, and other animation parameters. Some examples of these animations can be downloaded from [www.lip.uns.edu.ar/gmg](http://www.lip.uns.edu.ar/gmg).

- [8] Claudio Delrieux, Julián Domínguez, and Andrés Repetto. Advanced Techniques for Real-time Flow Visualization. In *SPIE Proceedings Vol. 4716*, pages 375–385, The International Society for Optical Engineering Press, ISBN: 0-8194-4466-9, [www.spie.org/web/abstracts/4700/4716.html](http://www.spie.org/web/abstracts/4700/4716.html), 2002.

## REFERENCES

- [1] Kreyszig E., *Advanced Mathematics for Engineering* Ed. Limusa, 1990.
- [2] Churchill R., Brown J. W. *Complex Analysis* McGraw-Hill, 1992.
- [3] Spiegel M. *Complex Calculus Series Schaum*, McGraw-Hill, 1989.
- [4] H.-O. Peitgen and D. Saupe, *The Science of Fractal Images*, Springer-Verlag, New York, 1986.
- [5] Thomas Strothotte, *Computational Visualization*, Springer, Berlin, 1998.
- [6] Peter Keller and Mary Keller, *Visual Cues: Practical Data Visualization*, IEEE Computer Society Press, 1990.
- [7] Claudio Delrieux, Julián Domínguez, and Andrés Repetto. Towards a CLIC in Vector Field Visualization. In *Proceedings of the CISST 2001 Conference*, pages 695–702, CSREA Press, ISBN 1-892512-73-4, 2001.