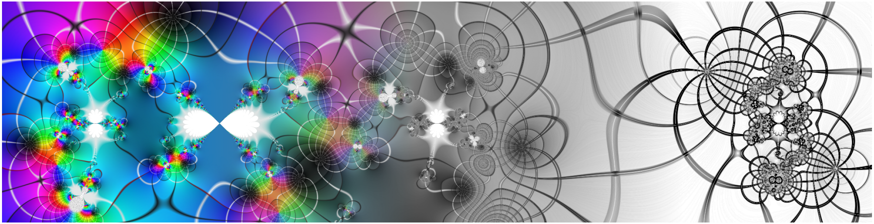


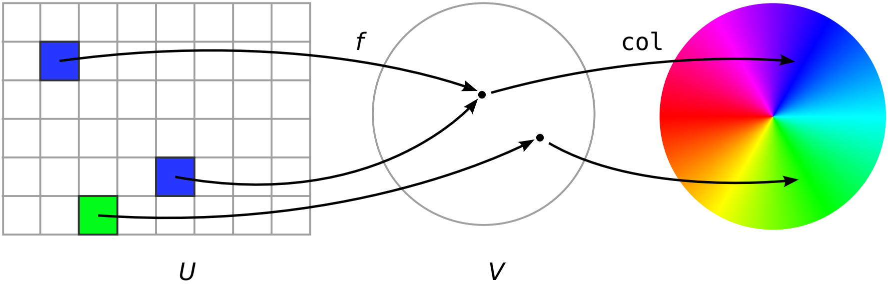
Lab - Domain Coloring of Complex Functions

Konstantin Poelke
Freie Universität Berlin

ABV Visualization 2014

Outline

1. Discretize domain $U \subset \mathbb{C}$ into pixels
2. Define color scheme function $\texttt{col} : \mathbb{C} \cup \{\infty\} \to \text{RGB}$
3. For each pixel (i, j) compute domain point $z_{ij} \in U$
4. Compute color $c_{ij} := \texttt{col} \circ f(z_{ij})$
5. Assign color to pixel: $img[i][j] = c_{ij}$



Recap: Notions from Complex Analysis

- Recall or look up the following definitions and notions from complex analysis.

- *Holomorphic functions*
- *Zeros* of a holomorphic function and their *multiplicities*
- *Singularities:*
  - *Poles*: $\frac{1}{(z-p)^k}$
  - *Essential singularities*: $\exp(\frac{1}{z}),\ldots$
- *Conformality:* preservation of angles and orientation
- *Complex Logarithm*, *principal branch*, *complex powers and roots*
- *Branching:* $\log z$, $\sqrt{z}$, ...
- *Riemann Surface*

Outline

Recap

Implementing Your Color Schemes
- Python Hacking
- Implementing Domain Coloring

Quizzes
- Quiz 1
- Quiz 2
- Quiz 3
- Quiz 4
- Quiz 5

IPython

- We will use the *IPython* environment
- Python programming language in Mathematica-like notebooks
- Easy programming in browser
- Heavily used in scientific community
- Many additional python libraries available, e.g. for: visualization, numerical computations, CAS, ODE/PDE solving, integration, symbolic differentiation,...

Installing IPython

- On Linux-Systems *IPython* usually comes with your distribution:
  `> sudo apt-get install ipython3`
- Cross-Platform: There are (I)Python bundles for easy installation of all dependencies, e.g. *Anaconda* for Windows, Mac or Linux:
  `http://continuum.io/downloads`
- Whatever you choose, try to get one with Python 3.x support!

Starting IPython

- *IPython* is pre-installed in PC pool
- Open terminal and type
  `user@pc> ipython3 notebook -pylab`
- This starts the IPython notebook viewer for Python 3
- The -pylab option loads the *matplotlib* stack for visualization
- Open a browser (e.g. Iceweasel) and go to
  `http://localhost:8888`

Common Python(/Math) Pitfalls

- Division of integers is integer division with rounding ($2/5 = 0$) in Python versions < 3.0, but floating point division ($2/5 = 0.4$) in Python 3.x (see also / vs. *//*)
- Make sure that you have correct spacing/indentation
- Statements like *def*, *if*, *for*,... all end with a colon ':'. The next line always starts with indentation
- Complex math module `cmath` vs. math module `math`
- Use as much *Numpy* as possible for speed-up

Python/IPython Literature

- *Python programming language*
  - Python 3.3: https://docs.python.org/3.3/
  - Python 2.7.8: https://docs.python.org/2/
  - Python 3: https://en.wikibooks.org/wiki/Non-Programmer%27s_Tutorial_for_Python_3
  - Python 3: http://www.diveintopython3.net/
- *IPython*
  - IPython: http://ipython.org/notebook.html
  - IPython 0.13.1 (deprecated PC pool version): http://ipython.org/ipython-doc/rel-0.13.1/index.html
  - Demo notebooks: https://github.com/ipython/ipython/wiki/A-gallery-of-interesting-IPython-Notebooks
- Some important libraries
  - NumPy: http://www.numpy.org/
  - SciPy: http://www.scipy.org/
  - matplotlib: http://matplotlib.org/
- Other:
  - Lambda expressions: http://openbook.galileocomputing.de/python/python_kapitel_10_005.htm
  - Math functions in Python: http://docs.python.org/library/math.html

Adding the Template to IPython

- You find an IPython notebook file *domain_coloring_template.ipynb* on the course web page
- Save it to your home directory or workspace location
- Import it into your running IPython by going to the root tree
  `http://localhost:8888`
  and drag-and-drop or browse to the file location.
- Open this notebook
- In the menu go to *Cell* → *Run All*

Importing Python Modules

- Provide necessary functionality, math functions, plotting, etc.

```
# Used to embed plots into notebook view
# Note: Remove this line for IPython version < 1.0
%matplotlib inline

# We need a plot function from matplotlib
import matplotlib.pyplot as plt

# We may need a color conversion function
from colorsys import hsv_to_rgb

# Use numpy for arrays, much faster than python arrays/lists
import numpy as np

# We need some standard math functions and values for definition of test functions
from cmath import pi, sin, cos, exp, log
```

The Color Scheme Function

- $\texttt{col} : \mathbb{C} \cup \{\infty\} \rightarrow \text{RGB}$ returns a color for the given complex number z
- Return value is a triple (r, g, b) in RGB color space.

```
def my_color_scheme(z):

    # Error handling, occurs e.g. on division by zero
    # May be useful for coloring infinity
    if np.isnan(z):
        pass

    # get polar coordinates, angle in [0,2pi]
    phi = np.angle(z)
    r = np.absolute(z)

    # Conversion from hsv to rgb color space
    # This is needed since plotting function takes rgb pixel colors.
    return hsv_to_rgb(0.5,1.,1.)
```

Plotting Procedure

- Main plotting function: discretizes domain, evaluates function *f* and uses `my_color_scheme` to compute colors for each function value.

```
def plot(f, domain = [-1,1,-1,1], res = (200,200)):
    left = domain[0]
    right = domain[1]
    bottom = domain[2]
    top = domain[3]

    dx = (right-left)/res[0]
    dy = (top-bottom)/res[1]

    img = np.empty(shape=(res[0],res[1],3))

    for i in range(res[1]):
        y = top-dy*i
        for j in range(res[0]):
            x = left+dx*j
            z = np.complex64(x+1J*y)
            fz = f(z)
            col = my_color_scheme(fz)
            img[i][j] = col

    plt.imshow(img, origin="lower", extent=domain)
```

Test Objects and Plotting

- Test the domain coloring script with a simple domain and function object
- Lambda expressions: convenient way to define (mathematical) functions in Python

```
# Define a test domain
domain = [−0.25,0.25,−0.25,0.25]

# Define a test function
f = lambda z: sin(1/z)


# Compute the domain coloring plot
plot(f,domain)
```

General Information

- For every answer save your color plots
- Whenever you are asked to find a function, save your function definition, too! (.txt-file)
- Name your files *yourname_quiz-X_Y*
  - X: quiz number
  - Y: question number
- Try implementing your own coloring algorithms! Use different color schemes and play around with the settings to get interesting results.
- Python gurus can try to tweak the plotting procedure (keywords: numpy, broadcasting)
- You may look up wikipedia for *HSV color space*
- `Send your results to your tutor!`

Outline

Recap

Implementing Your Color Schemes
- Python Hacking
- Implementing Domain Coloring

Quizzes
- Quiz 1
- Quiz 2
- Quiz 3
- Quiz 4
- Quiz 5

Quiz 1 - Getting Started

- Plot the functions...

1. $f(z) = z + \frac{1}{z}$
2. $f(z) = z^4 - 1$
3. $f(z) = \frac{z^2+(i-1)z-i}{z^2+2z+1}$

- Using your domain coloring script, find the poles and zeroes of the given functions. What are there orders? Can you “see” them?

Freie Universität Berlin

Quiz 2 - Zeros and Poles

- Find a function with...

1. a simple pole at -1 and a simple zero at $+1$
2. a simple zero at $+1$, a pole of order two at $+i$, a zero of order three at -1, a pole of order four at $-i$

- Plot these functions using your color scheme and check if the plot reflects your theoretical result.

Quiz 3 - Series Expansion

- Adjust your domain size to a square with side length 8 centered at the origin

1. Make a series of plots of exp and its non-constant Taylor polynomials P_k (expanded in zero) of order $k = 1, \ldots, 5$ (which means you should get 6 pictures all in all). What do you observe?

- For the experts: make a plot of P_{50}. You may use SAGE [1] or whatever CAS you want to compute the Taylor expansion. You may also use some text editor (gedit, kate, kile) and find-and-replace to get the input formatting correct.

[1] http://www.sagemath.org/

Outline

Quiz 4 - Conformality

- Plot the functions...

1. $f(z) = \frac{1}{3}z^3 - z$
2. $f(z) = \frac{1}{5}z^5 - z$
3. $\sin z$ and $\cos z$

- Where are these functions locally conformal? Where not? How can you see this? Do you need to adjust the settings?

Outline

Quiz 5 - Branching

- Plot the functions...

1. $f(z) = \log iz$
2. $g_1(z) = \sqrt{z+1}\sqrt{z-1}$
3. $g_2(z) = \sqrt{z^2-1}$

- Where are the branch cuts? Can you explain the difference between g_1 and g_2? Shouldn't they look the same?