Assignment

Exercise sheet 1 Task 4

Name TN 1: Tim Peinkofer

tim.peinkofer@student.uni-tuebingen.de

Name TN 2: Fabian Kostow

fabian.kostow@student.uni-tuebingen.de

Tutor: Jose Carlos Olvera Meneses

Date: 1. November 2024

Inhaltsverzeichnis

1 Task 4

In this assignment we want to discuss and figure out the convergence schema of different fractals. The basic code for the investigation of convergence and iterations is as follows:

```python
import numpy as np
import matplotlib.pyplot as plt

#Define Function
def func(x):
    return x**3-1

#Define derivative
def derivative(x):
    return 3*x**2

# finding the roots viay newton-raphson
def newton_raphson(x_0, f, df, tolerance=1e-9, max_iterations
                                   =200):
    x_t = x_0
    iteration = 0

    while abs(f(x_t)) >= tolerance and iteration <
                                          max_iterations: #Checking
                                          for convergence
        if abs(df(x_t))< 1e-12: #Solving the problem dividing
                                              by zero
            break
        x_t = x_t - f(x_t) / df(x_t)
        iteration += 1

    #
    return x_t if abs(f(x_t)) < tolerance else 0, iteration #
                                           This line returns the
                                          value after a convergence
                                          check
```

```
def find_roots_in_grid(f, df, N=400, epsilon=1e-9, max_steps=
                               200, x_range=(-2, 2), y_range=
                               (-2, 2)): # Generating grid to
                                plot
    x_values = np.linspace(x_range[0], x_range[1], N)
    y_values = np.linspace(y_range[0], y_range[1], N)

    grid_roots = np.zeros((N, N), dtype=complex)# Generating
                                  a grid for iterations and
                                  convergence
    iterations_grid = np.zeros((N, N))

    for i, y in enumerate(y_values): #Generating the values
                                  for every pixel of the
                                  grid
        for j, x in enumerate(x_values):
            z = complex(x, y)
            root, iterations = newton_raphson(z, f, df,
                                      epsilon, max_steps
                                      )
            grid_roots[i, j] = root
            iterations_grid[i, j] = iterations

    return grid_roots, iterations_grid


def plot_imaginary_part(grid_roots, x_range=(-2, 2), y_range=
                               (-2, 2)):
    plt.figure(figsize=(8, 8))
    plt.imshow(np.imag(grid_roots), extent=(x_range[0],
                                  x_range[1], y_range[0],
                                  y_range[1]), cmap="plasma"
                                  )
    plt.colorbar(label="Im(k(z))")
    plt.ylabel("Re(z)")
    plt.xlabel("Im(z)")
    plt.title("Imaginary part of roots in (x, y) plane")
    plt.show()
```

```python
def plot_log_iterations(iterations_grid, x_range=(-2, 2),
                                 y_range=(-2, 2)):
    log_iterations = np.log(iterations_grid, where=
                                  iterations_grid > 0) #
                                  Just plotting the values
                                  where number of iterations
                                   != 0
    plt.figure(figsize=(8, 8))
    plt.imshow(log_iterations, extent=(x_range[0], x_range[1]
                                  , y_range[0], y_range[1]),
                                   cmap="viridis")
    plt.colorbar(label="log10(Number of Iterations)")
    plt.xlabel("Re(z)")
    plt.ylabel("Im(z)")
    plt.title("Logarithm of iterations in (x, y) plane")
    plt.show()


if __name__ == "__main__":
    grid_roots, iterations_grid = find_roots_in_grid(func,
                                  derivative)
    plot_imaginary_part(grid_roots)
    plot_log_iterations(iterations_grid)
```

As a result we get get the roots and a convergence and iteration plot. Let us first take a look at the roots:

$$
\begin{aligned}
r_1 &= 1 \\
r_2 &\approx -0.5 + 0.866i \\
r_2 &\approx -0.5 - 0.866i
\end{aligned}
$$

We now compare this with the convergence plot:

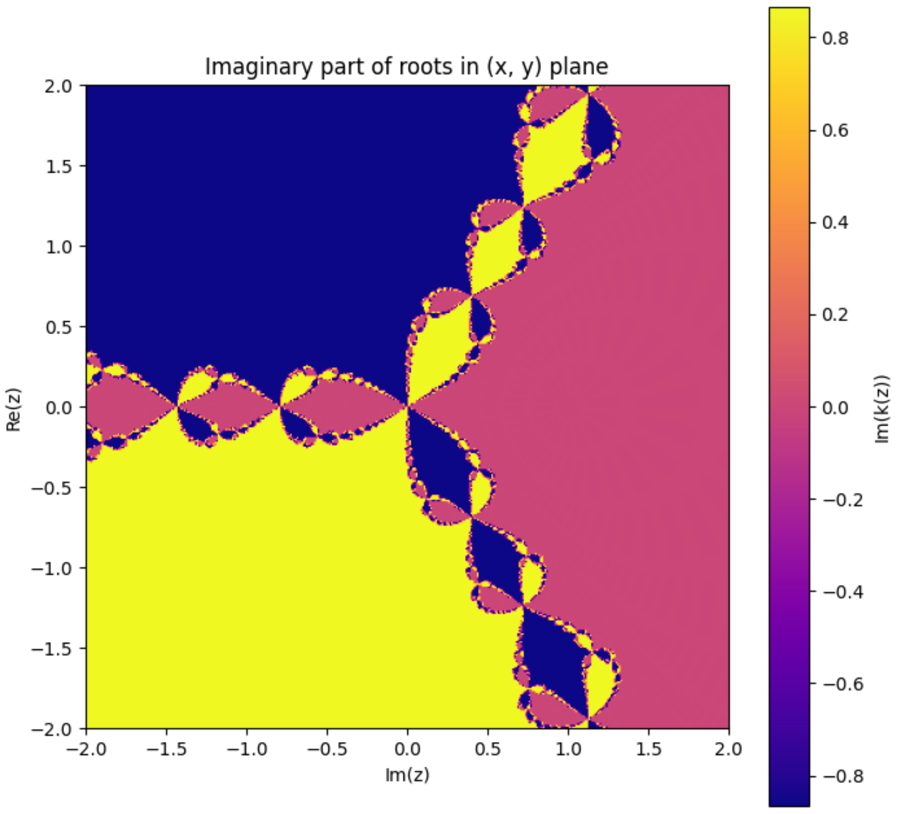

Abbildung 1: Convergence Plot $z^3 - 1$

We recognize that our roots are arranged at 120 degrees to each other as in the unit circle. The fractal pattern are based on our used method. Because of the convergence schema of the newton method, different values converge against differnt roots. This leads us to the fractal structure and the subdivision into differnt convergence areas.

Let us now zoom a little bit into a specific covergence area:

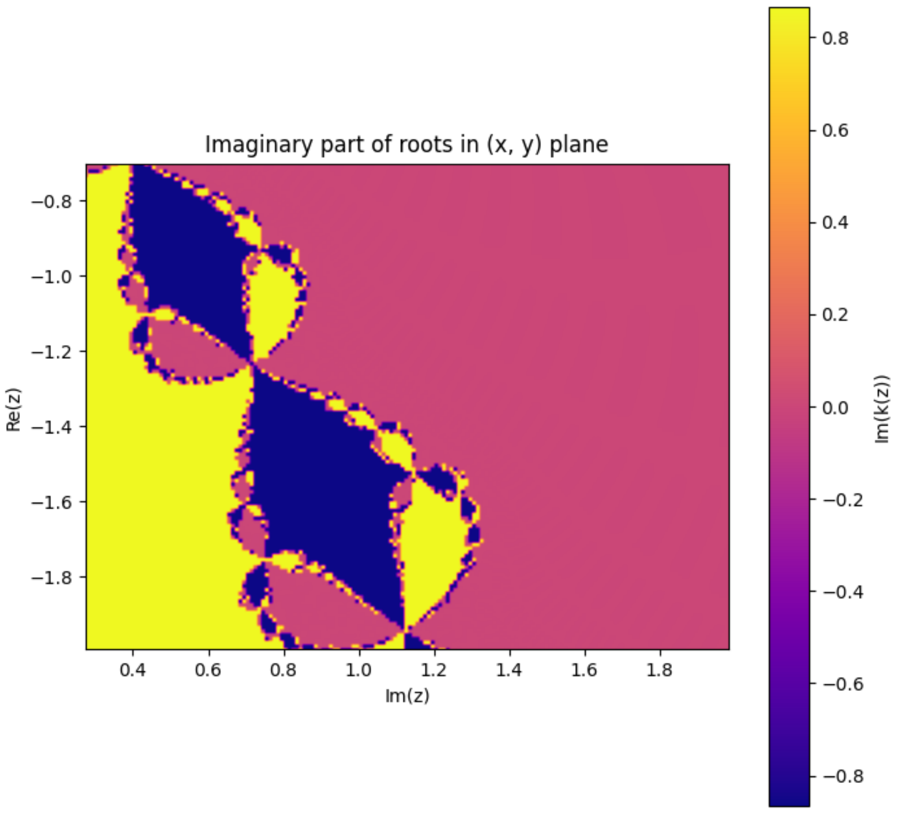

Abbildung 2: Convergence Zoomed Plot $z^3 - 1$

An important aspect can be recognized immediately from the excerpt: The different convergence behavior depending on the selected point. We see that the middle part of the fractal converges to r_2, while the points around it converge to r_1 or r_3. That is a typical behaviour of a fractal structure.

Next we want to take a look at the iteration grid:

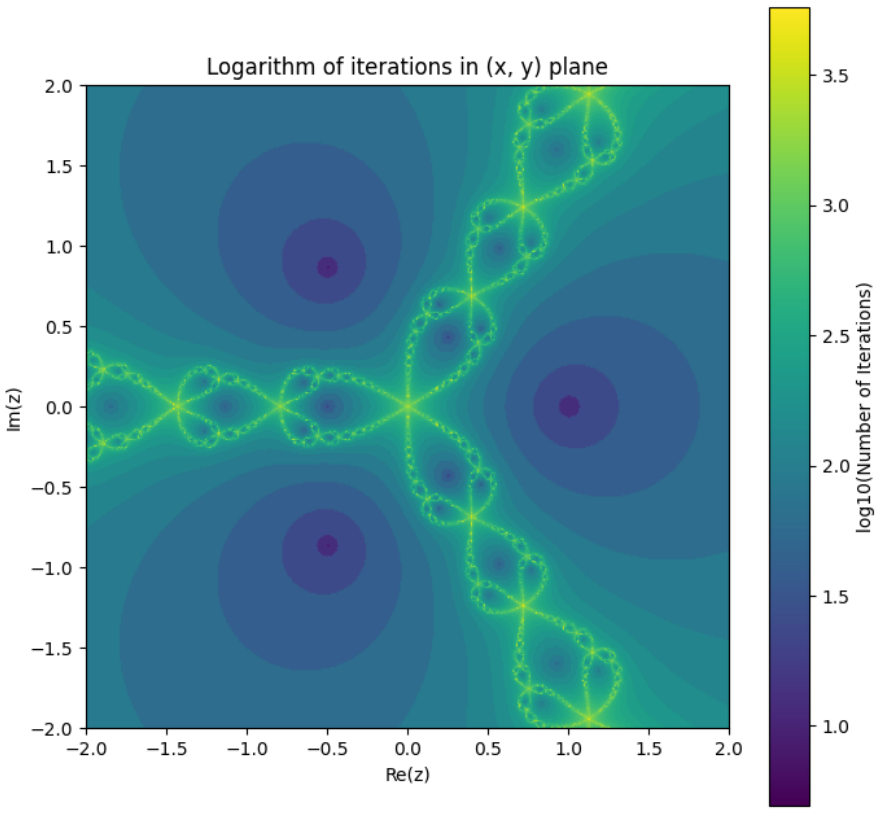

Abbildung 3: Iteration grid $z^3 - 1$

The three roots can be clearly seen as dark dots in the picture above. The number of iterations at these points is zero. The diverging areas of the fractal, which are shown in light green, are also clearly visible. These divergent areas occur at the edges of the areas of different convergence.

Let's take a closer look at the divergent areas:

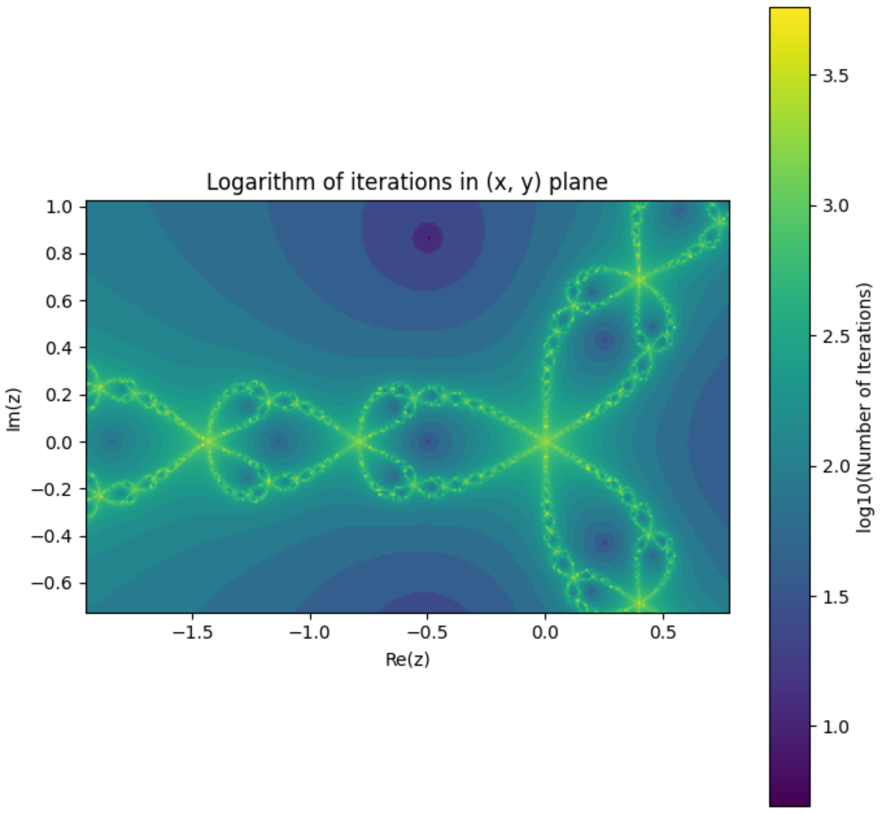

Abbildung 4: Zoomed Iteration grid $z^3 - 1$

Based on our excerpt, we can clearly see the divergent behavior of our sequence. Around each convergent area there is a divergent line, which represents the boundary between two areas of different convergence.

Now let's take a look at the convergence behavior of other functions.

2 Results for different functions

2.1 $z^7 - 1$

For the equation $z^7 - 1$ we get the following convergence plot:

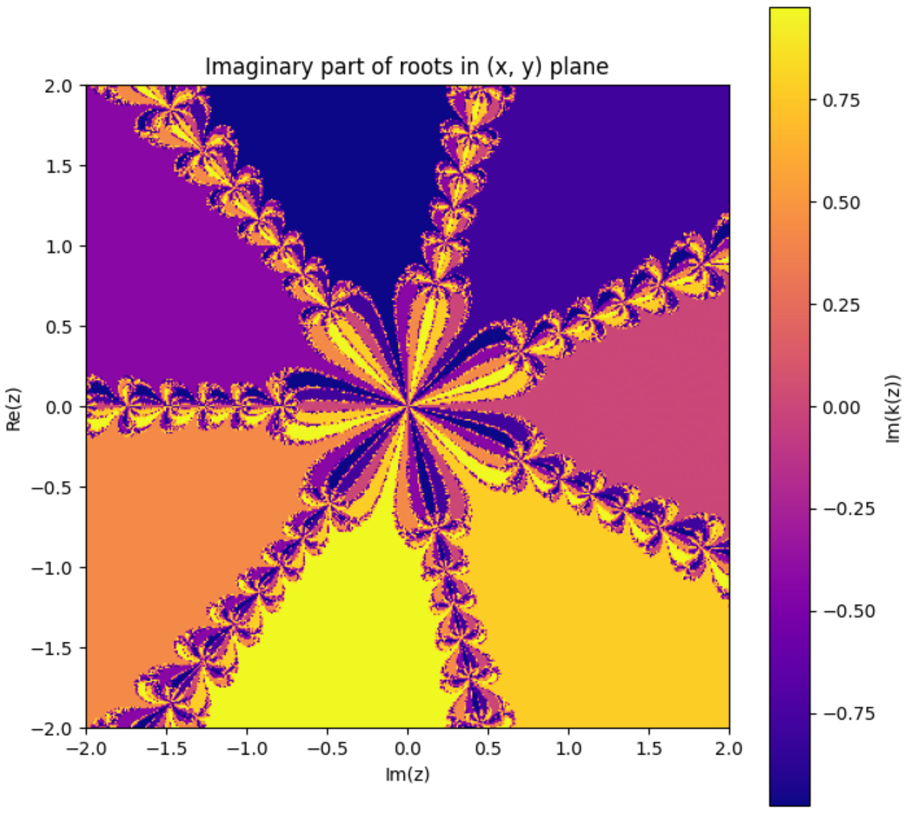

Abbildung 5: Convergence $z^7 - 1$

As we can see, the equation has seven roots, which we can examine because of the seven different convergence fields.

If we zoom a little bit in, we get:

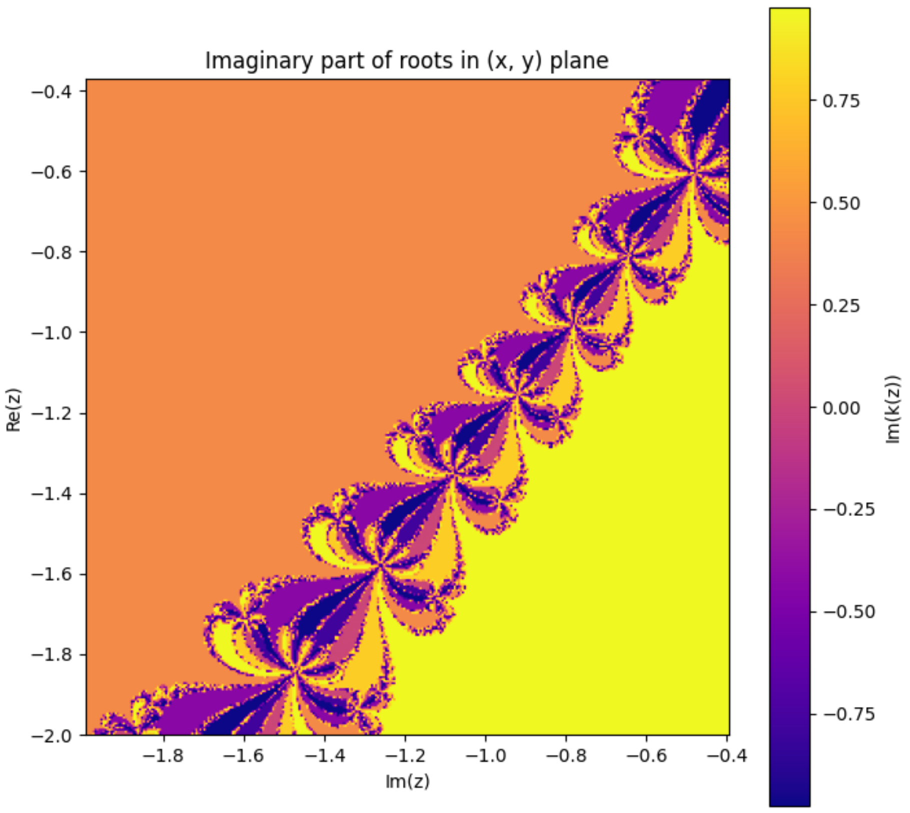

Abbildung 6: Convergence zoomed $z^7 - 1$

As we can see, the equation has seven roots, which we can examine because of the seven different convergence fields. This lets us conclude that it is a fractal.

For the iteration grid we get:

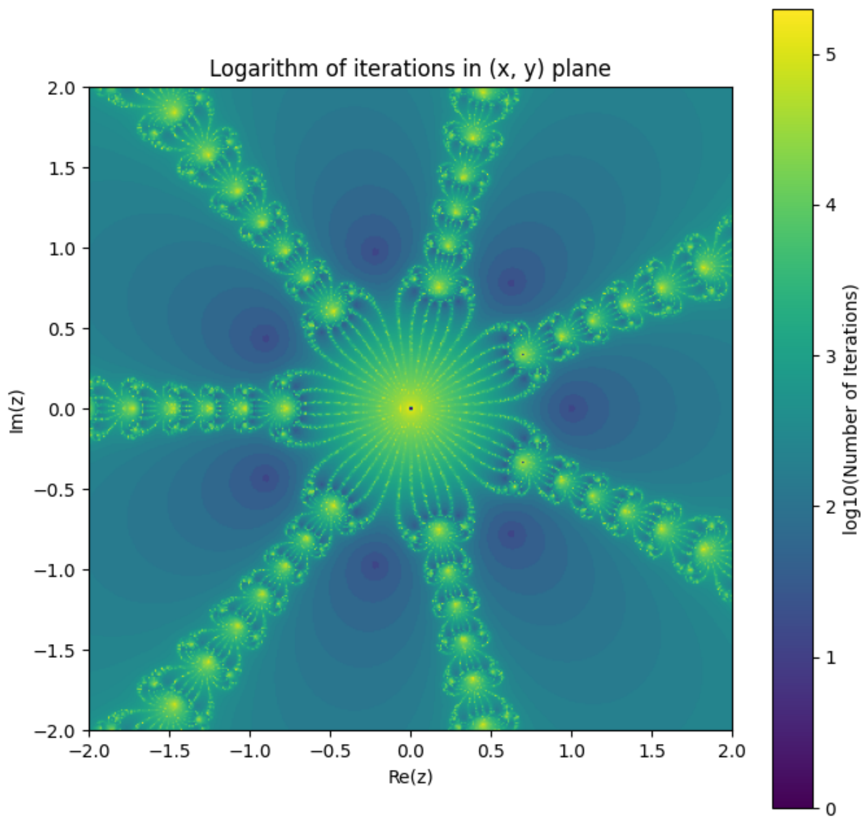

Abbildung 7: Iteration grid $z^7 - 1$

The light green lines are dividing the grid in sectors with different convergence behaviors, while the dark dots represent the roots. The reason for the latter is that the number of our iterations is zero.

2.2 $z^4 - 1$

For the equation $z^4 - 1$ we get the following convergence plot:

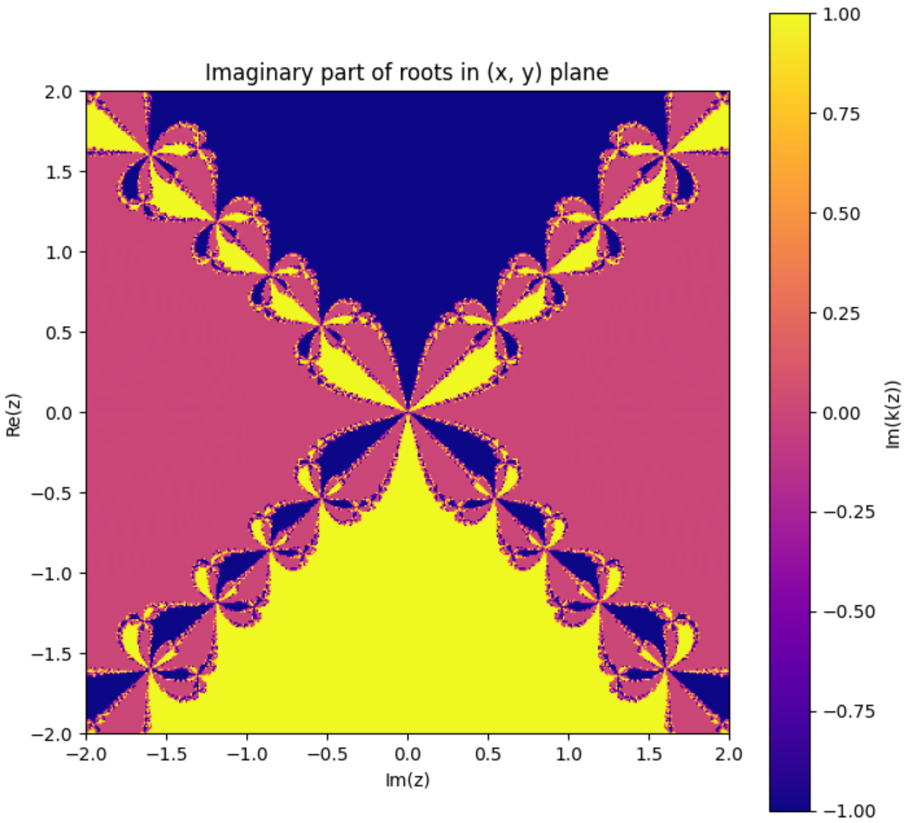

Abbildung 8: Convergence $z^4 - 1$

We also have four divided section because of the four roots of the equation. The different convergence behavior of different points can also be seen very clearly.

Next we take a look at the iteration grid:

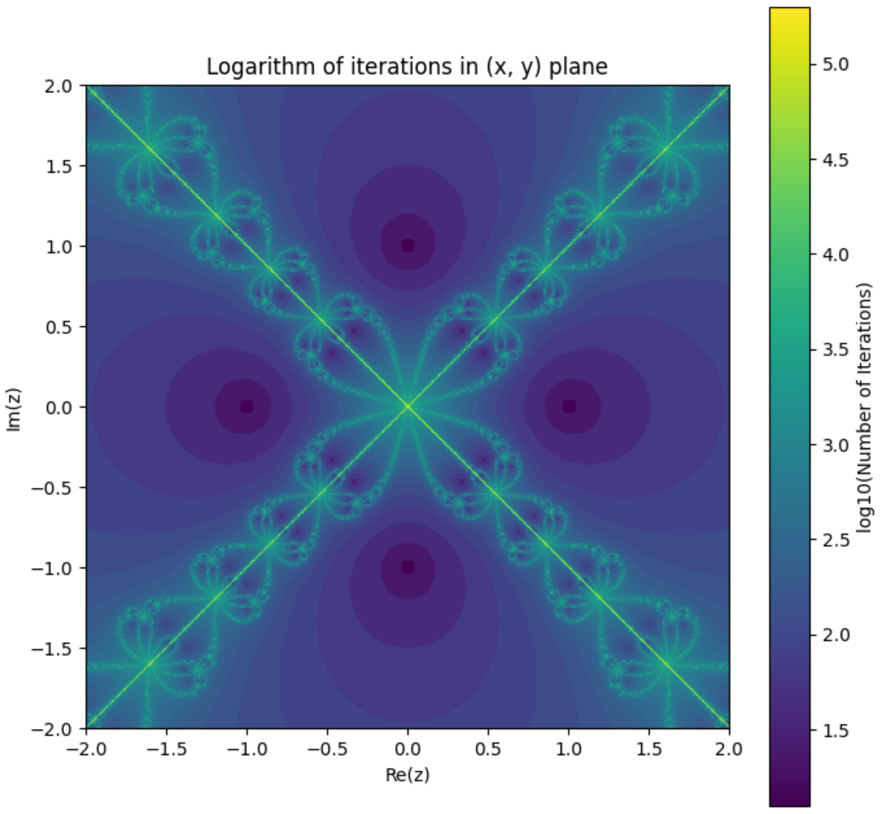

Abbildung 9: Iteration grid $z^4 - 1$

As above, we can also see the division into four sectors here (light green lines/ diverging zones). We also see four dark dots, which visualize our roots just like above.

2.3 $35z^9 - 180z^7 + 378z^5 - 420z^3 + 315z$

For the equation $35z^9 - 180z^7 + 378z^5 - 420z^3 + 315z$ we get the following convergence plot:

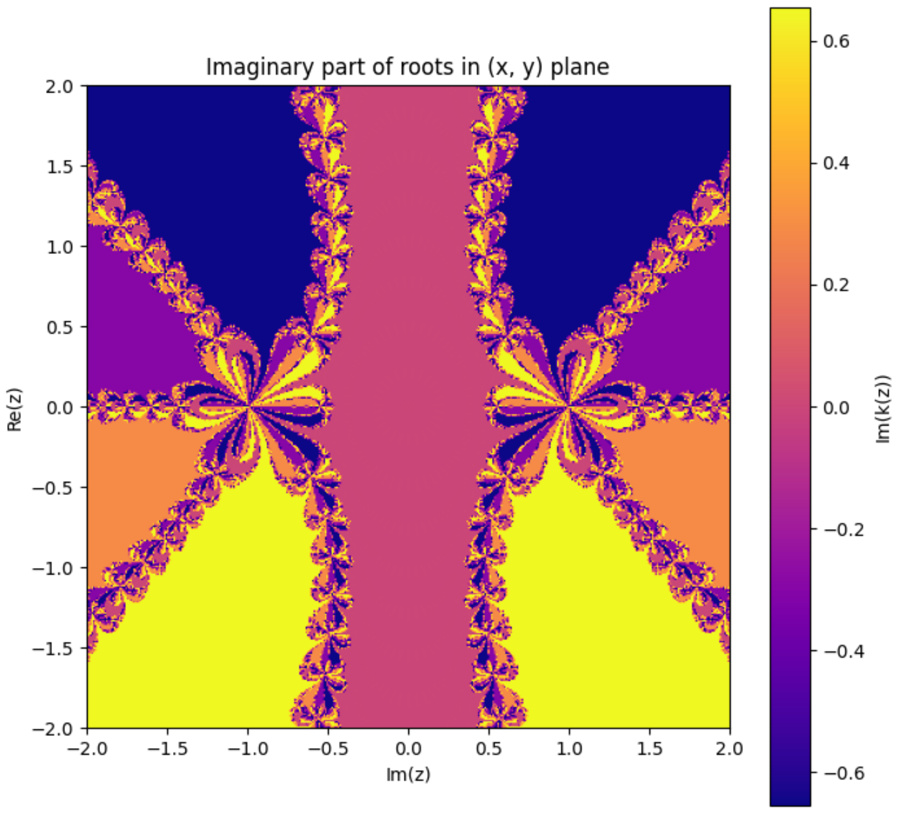

Abbildung 10: Convergence $35z^9 - 180z^7 + 378z^5 - 420z^3 + 315z$

The equation $35z^9 - 180z^7 + 378z^5 - 420z^3 + 315z$ does have seven roots, one real and six complex. This explains the seven different convergence areas, as well as the increasingly fractal structure at the boundaries of these areas.

These roots can also be recognized in the iteration grid:

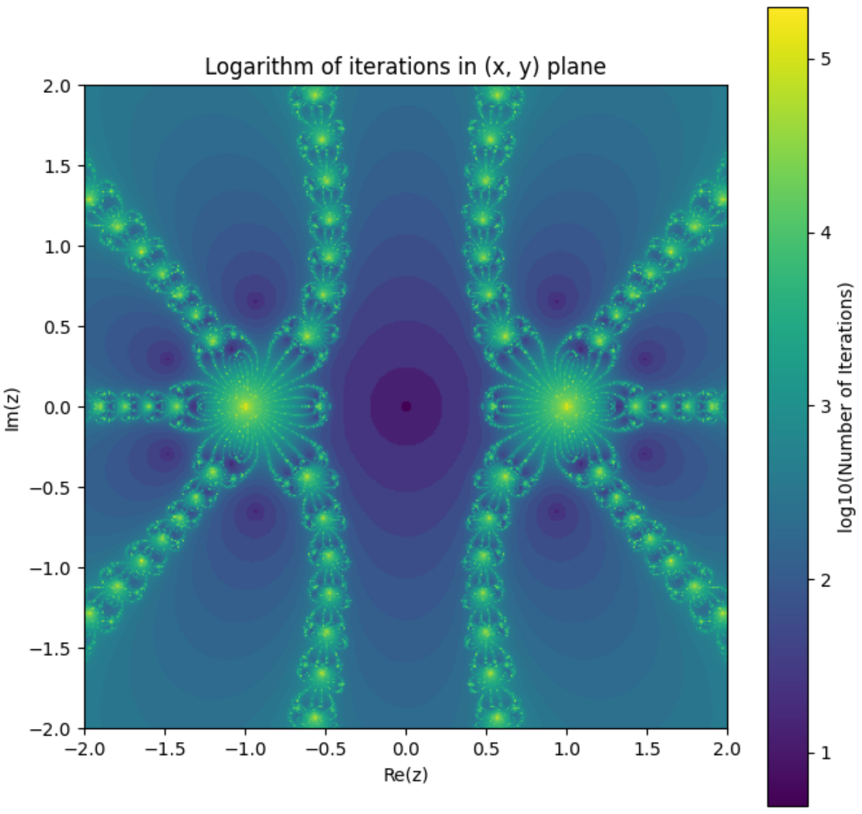

Abbildung 11: Iteration grid $35z^9 - 180z^7 + 378z^5 - 420z^3 + 315z$

2.4 Mandelbrot Set

We also tried to programm the Mandelbrot set. Because of it' s iterative definition we needed the following code to get good results:

```
import numpy as np
import matplotlib.pyplot as plt

def mandelbrot(N=400, max_steps=200, x_range=(-2, 0.7),
                               y_range=(-1.5, 1.2)):
    x_values = np.linspace(x_range[0], x_range[1], N)
    y_values = np.linspace(y_range[0], y_range[1], N)

    mandelbrot_set = np.zeros((N, N), dtype=complex)  #
                                   generate convergence grid
    iterations_grid = np.zeros((N, N))  # Generate iteration
                                   grid

    for i, y in enumerate(y_values):
        for j, x in enumerate(x_values):
            c = complex(x, y)
            z = 0
            iter_count = 0

            # Calculating the Mandelbrot set components
            while abs(z) < 2 and iter_count < max_steps:
                z = z**2 + c
                iter_count += 1

            # Save the results and the number of iterations
            mandelbrot_set[i, j] = z
            iterations_grid[i, j] = iter_count

    return mandelbrot_set, iterations_grid
```

```python
def plot_imaginary_part(grid_roots, x_range=(-2, 0.7),
                               y_range=(-1.5, 1.2)):
    plt.figure(figsize=(8, 8))
    plt.imshow(np.imag(grid_roots), extent=(x_range[0],
                                   x_range[1], y_range[0],
                                   y_range[1]), cmap="inferno
                                   ")
    plt.colorbar(label="Im(z)")
    plt.ylabel("Im(z)")
    plt.xlabel("Re(z)")
    plt.title("Imaginary part of roots in (x, y) plane")
    plt.show()

def plot_log_iterations(iterations_grid, x_range=(-2, 0.7),
                               y_range=(-1.5, 1.2)):
    log_iterations = np.log10(iterations_grid, where=
                                   iterations_grid > 0)  #
                                   Logarithmus Basis 10
    plt.figure(figsize=(8, 8))
    plt.imshow(log_iterations, extent=(x_range[0], x_range[1]
                                   , y_range[0], y_range[1]),
                                    cmap="viridis")
    plt.colorbar(label="log10(Number of Iterations)")
    plt.xlabel("Re(z)")
    plt.ylabel("Im(z)")
    plt.title("Logarithm of iterations in (x, y) plane")
    plt.show()


if __name__ == "__main__":
    grid_roots, iterations_grid = mandelbrot()
    plot_imaginary_part(grid_roots)
    plot_log_iterations(iterations_grid)
```

As result we get:

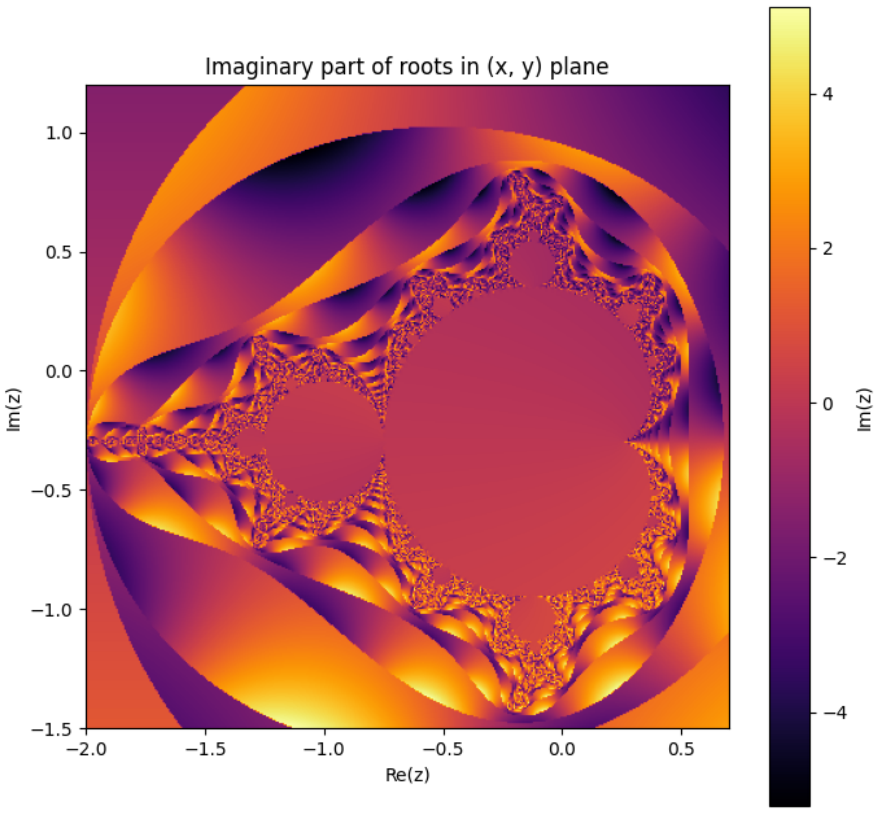

Abbildung 12: Convergence Mandelbrot

And also an iteration grid, which schows the Mandelbrot set behaviour perfectly:

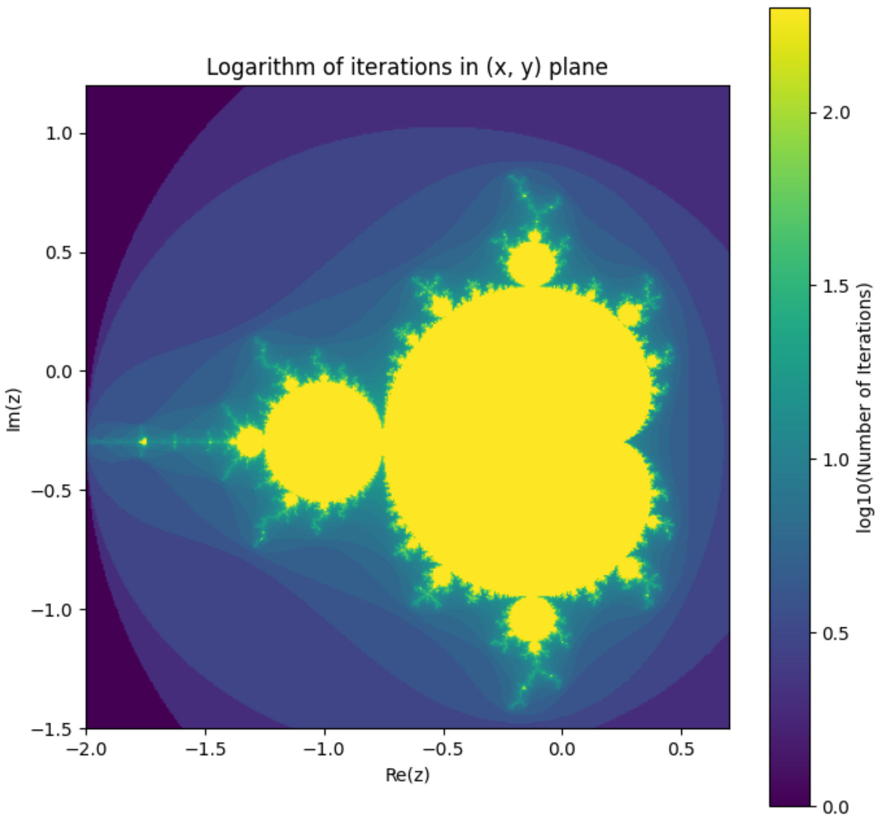

Abbildung 13: Iteration grid Mandelbrot