AP 7

Loewner differential equation

The Loewner differential equation is written as:

$$\frac{\partial g_t(z)}{\partial t} = \frac{2}{g_t(z) - \lambda(t)},$$

where:

- $g_t(z)$ is the Loewner chain,
- $\lambda(t)$ is the driving function (often continuous or piecewise).

For numerical purposes, this equation can be reformulated into a system of ODEs. Let's assume a toy driving function like $\lambda(t) = 2t$ to simplify implementation.

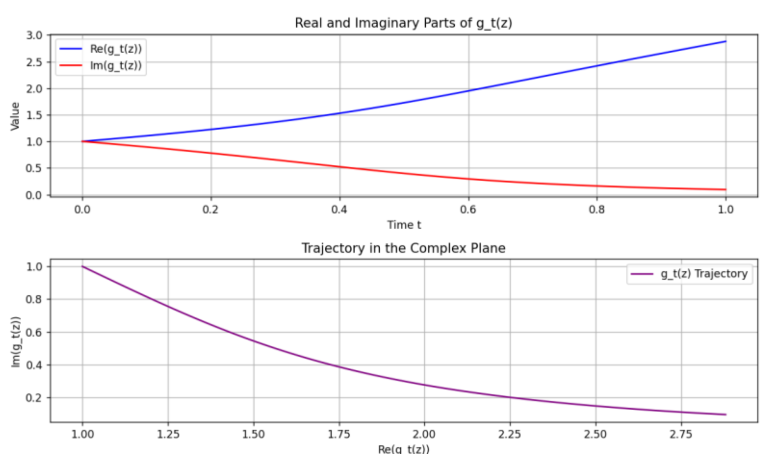
Rewriting:

$$g_t(z) = y(t), \quad \text{so the system becomes:} \quad \frac{dy}{dt} = \frac{2}{y - \lambda(t)}.$$

https://en.wikipedia.org/wiki/List_of_nonlinear_ordinary_differential_equations

Result:

The numerical solution and visualization of the Loewner differential equation demonstrate the impact of the driving function λ(t)=2t\lambda(t) = 2tλ(t)=2t on the evolution of gt(z)g_t(z)gt (z). The results highlight the structured yet dynamic nature of the Loewner chains, with the driving function dictating the behavior of the solution in both the real-imaginary space and the complex plane. These visualizations provide insights into how Loewner's theory models the growth of conformal maps over time.



Code:

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

# Loewner differential equation
def loewner_rhs(y, t, lambda_t):
    return 2 / (y - lambda_t)

# driving function
def driving_function(t):
    return 2 * t

# Adams-Bashforth 2-step method implementation
def adams_bashforth_2step(f, y0, t0, t_end, h, lambda_func):
    t = np.arange(t0, t_end + h, h)
    n_steps = len(t)

    y = np.zeros(n_steps, dtype=complex)
    y[0] = y0

    lambda_0 = lambda_func(t[0])
    y[1] = y[0] + h * f(y[0], t[0], lambda_0)

    for n in range(1, n_steps - 1):
        lambda_n = lambda_func(t[n])
        lambda_nm1 = lambda_func(t[n - 1])

        y[n + 1] = y[n] + h * (
            (3 / 2) * f(y[n], t[n], lambda_n) - (1 / 2) * f(y[n - 1], t[n - 1],
lambda_nm1)
        )

    return t, y

# Parameters for the toy model
initial_condition = 1 + 1j
t0, t_end = 0, 1
h = 0.01

# call Loewner differential equation
```

```python
t, y = adams_bashforth_2step(loewner_rhs, initial_condition, t0, t_end, h, driving_function)

# Visualization
plt.figure(figsize=(10, 6))

# Plot the real and imaginary parts
plt.subplot(2, 1, 1)
plt.plot(t, y.real, label="Re(g_t(z))", color="blue")
plt.plot(t, y.imag, label="Im(g_t(z))", color="red")
plt.xlabel("Time t")
plt.ylabel("Value")
plt.title("Real and Imaginary Parts of g_t(z)")
plt.legend()
plt.grid()

# Plot the trajectory in the complex plane
plt.subplot(2, 1, 2)
plt.plot(y.real, y.imag, label="g_t(z) Trajectory", color="purple")
plt.xlabel("Re(g_t(z))")
plt.ylabel("Im(g_t(z))")
plt.title("Trajectory in the Complex Plane")
plt.legend()
plt.grid()

plt.tight_layout()
plt.show()
```