Sprawozdanie – Symulacja układu RLC metodą RK4

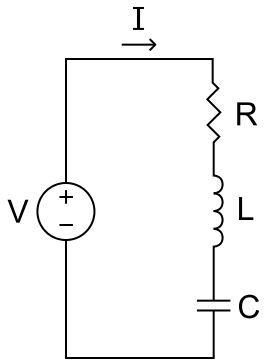
1. Cel

Celem zadania było numeryczne rozwiązanie równań różniczkowych opisujących obwód RLC za pomocą metody Rungego-Kutty IV rzędu oraz wizualizacja wyników.

---

2. Model fizyczny

Układ RLC to szeregowe połączenie rezystora (R), cewki (L) oraz kondensatora (C).



3. Wyprowadzenie równań

Podstawowe równanie dla obwodu RLC:

$$\frac{d^2I(t)}{dt^2} + \frac{R}{L}\frac{dI(t)}{dt} + \frac{1}{LC}I(t) = 0$$

Podstawiamy:

- $x_1 = I$
- $x_2 = \frac{dI}{dt}$

Otrzymujemy układ dwóch równań pierwszego rzędu:

$$\begin{cases} \frac{dx_1}{dt} = x_2 \\ \frac{dx_2}{dt} = -\frac{R}{L}x_2 - \frac{1}{LC}x_1 \end{cases}$$

---

4. Metoda numeryczna: Rungego-Kutty IV rzędu (RK4)

Ogólny wzór RK4:

$$y_{n+1} = y_n + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4)$$

$$K_1 = f(t_n, y_n) \cdot h$$

$$K_2 = f\left(t_n + \frac{h}{2}, y_n + \frac{K_1}{2}\right) \cdot h$$

$$K_3 = f\left(t_n + \frac{h}{2}, y_n + \frac{K_2}{2}\right) \cdot h$$

$$K_4 = f(t_n + h, y_n + K_3) \cdot h$$

Dla układu RLC funkcja f to:

$$f(t, [x_1, x_2]) = \begin{bmatrix} x_2 \\ \\ -\frac{R}{L}x_2 - \frac{1}{LC}x_1 \end{bmatrix}$$

---

5. Implementacja

Symulację wykonałem w języku **Rust** z użyciem bibliotek **egui** oraz **plotters**.

Fukcja pochodnej w kodzie:

```rust
fn rlc_derivs(r: f64, l: f64, c: f64, _t: f64, state: &[f64; 2]) -> [f64; 2] {
    // state[0] = x1 = I
    // state[1] = x2 = dI/dt
    let x1 = state[0]; // current I
    let x2 = state[1]; // dI/dt

    // dI/dt = x2 (by definition)
    // d^2I/dt^2 = -R/L*x2 - 1/(L*C)*x1
    let dx1_dt = x2;
    let dx2_dt = -r / l * x2 - 1.0 / (l * c) * x1;
    [dx1_dt, dx2_dt]
}
```

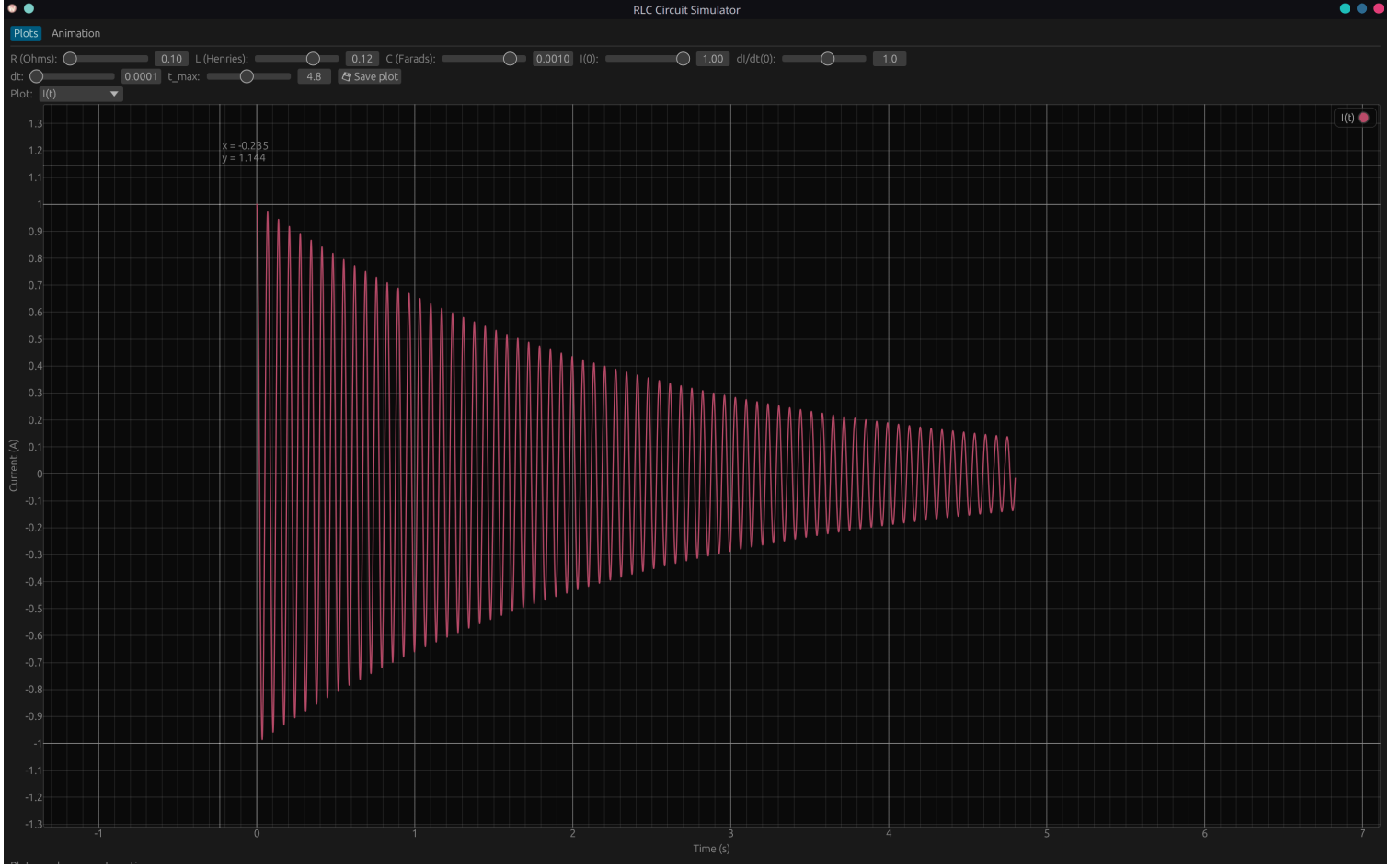
Funckja RK4 w kodzie:

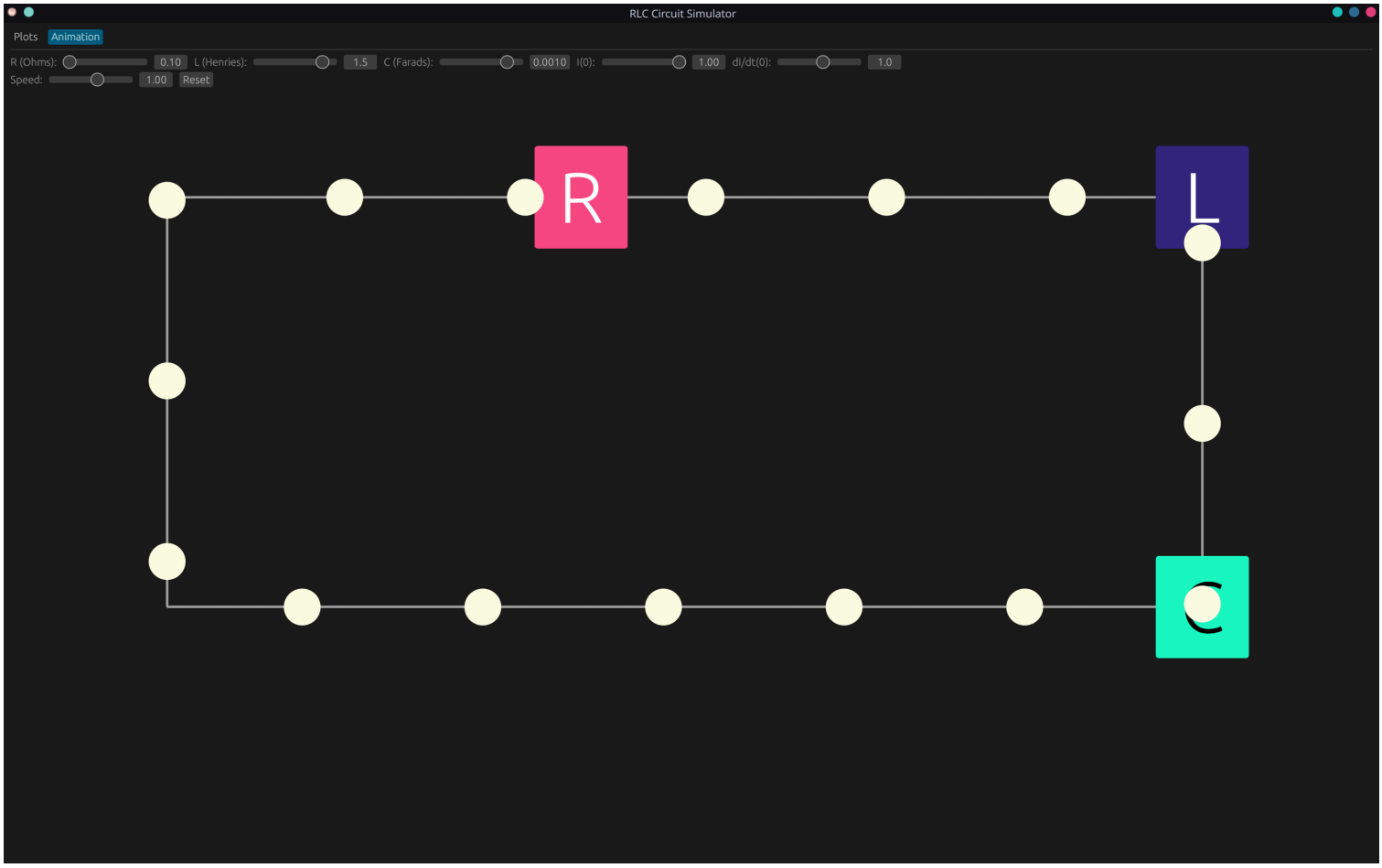
```
fn rk4_step(
    r: f64,
    l: f64,
    c: f64,
    t: f64,
    state: &[f64; 2],
    dt: f64,
) -> [f64; 2] {
```

```
    // K1
    let k1 = rlc_derivs(r, l, c, t, state);

    // K2 state = state + dt/2 * k1
    let state_k2 = [
        state[0] + dt / 2.0 * k1[0],
        state[1] + dt / 2.0 * k1[1],
    ];
    let k2 = rlc_derivs(r, l, c, t + dt / 2.0, &state_k2);

    // K3 state = state + dt/2 * k2
    let state_k3 = [
        state[0] + dt / 2.0 * k2[0],
        state[1] + dt / 2.0 * k2[1],
    ];
    let k3 = rlc_derivs(r, l, c, t + dt / 2.0, &state_k3);

    // K4 state = state + dt * k3
    let state_k4 = [
        state[0] + dt * k3[0],
        state[1] + dt * k3[1],
    ];
    let k4 = rlc_derivs(r, l, c, t + dt, &state_k4);

    // weighted sum x+1 = dt / 6 * (K1 + 2K2 + 2K3 + K4)
    [
        state[0] + dt / 6.0 * (k1[0] + 2.0 * k2[0] + 2.0 * k3[0] + k4[0]),
        state[1] + dt / 6.0 * (k1[1] + 2.0 * k2[1] + 2.0 * k3[1] + k4[1]),
    ]
}
```

Opis GUI

Program umożliwia wybór parametrów R, L, C, warunków początkowych, kroku czasowego i czasu symulacji, a następnie generuje trzy typy wykresów: $I(t)$ (prąd w funkcji czasu), $\frac{dI}{dt}(t)$ (pochodna prądu względem czasu) oraz wykres fazowy I względem $\frac{dI}{dt}$. Wszystkie wykresy można zapisać do pliku PNG z nazwą zawierającą typ wykresu oraz wszystkie parametry symulacji, np. `phase_plot_R=5_L=1_C=0.01_I0=1_dI0=0_dt=0.01_tmax=10.png`.





7. Wnioski

- Metoda Rungego-Kutty IV rzędu skutecznie rozwiązuje układ równań różniczkowych opisujących obwód RLC.

- Zmiana parametrów R, L, C znacząco wpływa na charakter przebiegu prądu – w szczególności na szybkość tłumienia i częstotliwość drgań.
  Wartość kroku czasowego dt ma istotny wpływ na jakość symulacji: dla $dt < 0.005$ wyniki są już znacząco zniekształcone, a dla $dt < 0.01$ stają się całkowicie bezużyteczne.
- Płynność animacji może być ograniczona przez wydajność biblioteki graficznej lub koszt obliczeń metody RK4.

---

8. Źródła

- Wikipedia – Obwód RLC
- Dokumentacja eframe, egui, egui_plot, plotters