Programowanie II R

Zadania – seria 10.

Szablony funkcji i klas

1 Wstęp

Dynamikę wielu układów fizycznych można opisać za pomocą układów równań różniczkowych zwyczajnych (ODE) pierwszego rzędu:

$$\frac{d\mathbf{u}}{dt} = f(\mathbf{u}) \tag{1}$$

gdzie $\mathbf{u}$ jest wektorem stanu w N-wymiarowej przestrzeni fazowej. Celem ćwiczeń jest stworzenie narzędzia, które dzięki mechanizmowi szablonów pozwoli rozwiązać dowolny taki układ przy użyciu algorytmu Rungego-Kutty 4. rzędu (RK4).

2 Zadanie 1: Szablon klasy `Vector<T, N>`

Zaimplementuj klasę reprezentującą N-wymiarowy wektor o elementach typu `T`. Klasa powinna:

- Przechowywać dane wewnątrz statycznej tablicy (np. `std::array<T, N>`).
- Zawierać konstruktor bez argumentów, który wypełnia tablicę zerami.
- Zawierać konstruktor przyjmujący argument typu `std::initializer_list<T>` (co pozwala np. napisać `Vector<double, 3> v = {1.0, 2.0, 3.0};`).
- Przeciążać operatory niezbędne do wykonywania operacji wektorowych:
  - `operator[]` – dostęp do elementów (dwie wersje: zwracająca `T&` i zwracająca `const T&`),
  - `operator+` – dodawanie dwóch wektorów,
  - `operator*` – mnożenie wektora przez skalar typu `T`.

3 Zadanie 2: Algorytm RK4

Napisz szablon funkcji `rk4_step`, która wykonuje jeden krok całkowania algorytmem RK4:

$$\begin{aligned}
\mathbf{u}_{t+\Delta t} &= \mathbf{u}_t + \frac{\Delta t}{6}(\mathbf{k_1} + 2\mathbf{k_2} + 2\mathbf{k_3} + \mathbf{k_4}) \\
\mathbf{k_1} &= f(\mathbf{u_t}) \\
\mathbf{k_2} &= f(\mathbf{u_t} + \mathbf{k_1}\frac{\Delta t}{2}) \\
\mathbf{k_3} &= f(\mathbf{u_t} + \mathbf{k_2}\frac{\Delta t}{2}) \\
\mathbf{k_4} &= f(\mathbf{u_t} + \mathbf{k_3}\Delta t)
\end{aligned}$$

- **Argumenty szablonu:** typ `T`, liczba całkowita `N` (obydwa tak jak w klasie `Vector`), typ `derivative_func`.
- **Argumenty funkcji:** obecny stan (typ `const Vector<T, N>&`), krok czasowy (typ `T`) oraz funkcja obliczająca wektor pochodnych (typ `derivative_func`). Zakładamy, że jeśli obiekt typu `derivative_func` wywołany zostanie z argumentem typu `Vector<T, N>` opisującym stan układu, to zwrócony zostanie obiekt typu `Vector<T, N>` zawierający pochodne po czasie.
- **Wartość zwracana:** Stan układ po kroku (typ `Vector<T, N>`).

4 Zadanie 3: Dynamika populacji (model Lotki-Volterry)

Użyj funkcji `rk4_step` do symulowania dynamiki układu drapieżnik-ofiara ($N = 2$, $\mathbf{u} = \{x, y\}$) opisanego równaniami:

$$\frac{dx}{dt} = \alpha x - \beta xy \tag{2}$$

$$\frac{dy}{dt} = \delta xy - \gamma y \tag{3}$$

Parametry: $\alpha = 1.1, \beta = 0.4, \delta = 0.1, \gamma = 0.4$. Stan początkowy: $x = 10, y = 5$. Czas symulacji: 50 s, krok czasowy: 0.1 s.

Zapisuj wartości t, x i y do pliku `lotka.csv`. Wygeneruj wykres populacji od czasu oraz diagram fazowy (y względem x).
Do ich narysowania można użyć następujących komend programu `gnuplot`:

```
set datafile separator ","
plot "lotka.csv" using 1:2 with lines title "Ofiary", \
     "lotka.csv" using 1:3 with lines title "Drapiezniki"
plot "lotka.csv" using 2:3 with lines title "Diagram"
```

5 Zadanie 4: Chaos deterministyczny (atraktor Lorenza)

Przeprowadź symulację układu Lorenza ($N = 3$):

$$\frac{dx}{dt} = \sigma(y - x)$$

$$\frac{dy}{dt} = x(\rho - z) - y$$

$$\frac{dz}{dt} = xy - \beta z$$

Parametry: $\sigma = 10, \rho = 28, \beta = 8/3$. Stan początkowy: $[1, 1, 1]$. Czas symulacji: 100 s, krok czasowy: 0.01 s.

Zapisuj wartości x, y i z do pliku `lorenz.csv`. Narysuj diagram fazowy, np:

```
set datafile separator ","
splot "lorenz.csv" using 1:2:3 with lines lw 1 lc rgb "blue" title "Trajektoria"
```

Opracowanie: Piotr Dziekan