**Laboratorium 1  
Symulacje deterministyczne**

**Imię i nazwisko:** Franciszek Jarek, Marcin Napieralski

1. **Wstęp**

Celem ćwiczenia było zaimplementowanie algorytmu Rungego-Kutty IV rzędu (RK4) do symulacji deterministycznej modelu biologicznego regulacji białek w komórce, a następnie wykonanie czterech scenariuszy testowych obejmujących odpowiedź komórki na uszkodzenia DNA, działanie mechanizmu PTEN oraz terapii siRNA. Model został dostarczony przez prowadzącego i opisuje dynamikę czterech białek: p53, MDM2 (w cytoplazmie i jądrze) oraz PTEN.

Symulacja została przeprowadzona zarówno z użyciem **RK4 o stałym kroku całkowania**, jak i wersji **RK4 ze zmiennym krokiem**, wykorzystując adaptacyjną kontrolę błędu. Końcowe wyniki zostały przedstawione w formie wykresów obrazujących liczbę cząsteczek każdego z białek w czasie (do 48 godzin).

**2. Implementacja algorytmu**

**2.1 Warianty RK4**

**W ramach projektu zaimplementowano dwa warianty RK4:**

1. RK4 ze stałym krokiem całkowania – krok wynosił 0.5 minuty, a całkowity czas symulacji to 2880 minut (czyli 48 godzin).
2. RK4 ze zmiennym krokiem całkowania – krok był dynamicznie dostosowywany na podstawie różnicy pomiędzy rozwiązaniem RK4 i RK2. Zastosowano podstawowy mechanizm sterowania krokiem oparty na progu błędu (domyślnie 1e-2) oraz granice minimalnego i maksymalnego kroku (od 0.01 do 5 minut).

**2.2 Struktura programu**

**Program napisano w języku Python z wykorzystaniem bibliotek:**

* numpy – do obliczeń numerycznych,
* matplotlib – do generowania wykresów.

Całość znajduje się w pliku main.py z funkcją main() jako punktem wejścia. Gotowy plik .exe został wygenerowany narzędziem pyinstaller i nie wymaga dodatkowego oprogramowania do uruchomienia. Dołączyliśmy też plik requirements.txt, który zawiera informację o wszystkich potrzebnych bibliotekach.

**2.3 Dane wejściowe**

Wartości początkowe cząsteczek białek zostały dobrane po wcześniejszej stabilizacji modelu (symulacja 208 dni z krokiem 5 minut). Uzyskane wartości końcowe wykorzystano jako warunki początkowe do wszystkich czterech scenariuszy testowych.

**Wartości przed stabilizacją:**

* p53: 90000,
* MDMcyto: 200000,
* MDMn: 250000,
* PTEN: 550000.

A graph of different colored lines

AI-generated content may be incorrect.

**Wartości po stabilizacji:**

* p53: 26853.93,
* MDMcyto: 11173.46,
* MDMn: 154378.21,
* PTEN: 17287.56.

**3. Wyniki symulacji**

**3.1 Scenariusze:**

* A) Podstawowy: Brak uszkodzeń DNA, działający PTEN, brak siRNA.
* B) Uszkodzenie DNA: Aktywne PTEN, uszkodzenia DNA.
* C) Nowotwór: Wyłączony PTEN, uszkodzenia DNA, brak siRNA.
* D) Terapia: Wyłączony PTEN, uszkodzenia DNA, włączone siRNA.

Każdy scenariusz był zasymulowany zarówno z krokiem stałym, jak i zmiennym. Przebiegi zostały zapisane do plików PNG.

import numpy as np

import matplotlib.pyplot as plt

# Parametry modelu

p1 = 8.8

p2 = 440

p3 = 100

d1 = 1.375e-14

d2 = 1.375e-4

d3 = 3e-5

k1 = 1.925e-4

k2 = 1e5

k3 = 1.5e5

# Model

def model(t, y, config):

    p53, mdmcyto, mdmn, pten = y

    siRNA = config.get("siRNA", False)

    DNA\_damage = config.get("DNA\_damage", False)

    PTEN\_off = config.get("PTEN\_off", False)

    siRNA\_factor = 0.02 if siRNA else 1.0

    DNA\_factor = 1.0 if DNA\_damage else 0.1

    PTEN\_factor = 0.0 if PTEN\_off else 1.0

    dp53\_dt = p1 - d1 \* p53 \* mdmn \*\* 2

    dmdmcyto\_dt = (p2 \* siRNA\_factor \* (p53 \*\* 4) / ((p53 \*\* 4) + k2 \*\* 4)

                   - k1 \* (k3 \*\* 2) / ((k3 \*\* 2) + pten \*\* 2) \* mdmcyto

                   - d2 \* DNA\_factor \* mdmcyto)

    dmdmn\_dt = k1 \* (k3 \*\* 2) / ((k3 \*\* 2) + pten \*\* 2) \* mdmcyto - d2 \* DNA\_factor \* mdmn

    dpten\_dt = p3 \* PTEN\_factor \* (p53 \*\* 4) / ((p53 \*\* 4) + k2 \*\* 4) - d3 \* pten

    return np.array([dp53\_dt, dmdmcyto\_dt, dmdmn\_dt, dpten\_dt])

# RK4 z krokiem stałym

def rk4\_fixed(f, y0, t0, tf, h, config):

    times = [t0]

    results = [y0]

    y = y0.copy()

    t = t0

    while t < tf:

        k1 = f(t, y, config)

        k2 = f(t + h / 2, y + h / 2 \* k1, config)

        k3 = f(t + h / 2, y + h / 2 \* k2, config)

        k4 = f(t + h, y + h \* k3, config)

        y = y + h / 6 \* (k1 + 2 \* k2 + 2 \* k3 + k4)

        t += h

        times.append(t)

        results.append(y)

    return np.array(times), np.array(results)

# RK4 adaptacyjny

def rk4\_adaptive(f, y0, t0, tf, h0, config, tol=1e-2, h\_min=0.01, h\_max=5):

    times = [t0]

    results = [y0.copy()]

    t = t0

    y = y0.copy()

    h = h0

    while t < tf:

        if t + h > tf:

            h = tf - t

        k1 = f(t, y, config)

        k2 = f(t + h / 2, y + h / 2 \* k1, config)

        k3 = f(t + h / 2, y + h / 2 \* k2, config)

        k4 = f(t + h, y + h \* k3, config)

        y\_rk4 = y + h / 6 \* (k1 + 2\*k2 + 2\*k3 + k4)

        k1s = f(t, y, config)

        k2s = f(t + h, y + h \* k1s, config)

        y\_rk2 = y + h / 2 \* (k1s + k2s)

        error = np.max(np.abs(y\_rk4 - y\_rk2))

        if error < tol or h <= h\_min:

            t += h

            y = y\_rk4

            times.append(t)

            results.append(y.copy())

        s = min(2, max(0.5, 0.9 \* (tol / error)\*\*0.5)) if error != 0 else 2

        h = max(h\_min, min(h \* s, h\_max))

    return np.array(times), np.array(results)

# Główna część programu

def main():

    # Stabilizacja

    y0 = np.array([90000, 200000, 250000, 550000])

    t0 = 0

    tf = 300000  # minut

    h = 5

    config\_base = {"siRNA": False, "DNA\_damage": False, "PTEN\_off": False}

    times, results = rk4\_fixed(model, y0, t0, tf, h, config\_base)

    # Wykres stabilizacji

    labels = ["p53", "MDMcyto", "MDMn", "PTEN"]

    index\_map = {label: i for i, label in enumerate(labels)}

    plt.figure(figsize=(12, 8))

    for label in labels:

        plt.plot(times / 60, results[:, index\_map[label]], label=label, linewidth=2)

    plt.yscale("log")

    plt.title("Stabilizacja modelu (scenariusz podstawowy)")

    plt.xlabel("Czas [h]")

    plt.ylabel("Liczba cząsteczek (log)")

    plt.legend()

    plt.grid(True, which="both", linestyle="--", linewidth=0.5)

    plt.tight\_layout()

    plt.savefig("stabilizacja\_podstawowy.png", dpi=300)

    plt.close()

    final\_values = results[-1]

    # Symulacje 4 scenariuszy

    initial\_conditions = final\_values

    t0 = 0

    tf = 2880

    h = 0.5

    scenarios = {

        "A\_Podstawowy": {"siRNA": False, "DNA\_damage": False, "PTEN\_off": False},

        "B\_Uszkodzenie\_DNA": {"siRNA": False, "DNA\_damage": True, "PTEN\_off": False},

        "C\_Nowotwór": {"siRNA": False, "DNA\_damage": True, "PTEN\_off": True},

        "D\_Terapia": {"siRNA": True, "DNA\_damage": True, "PTEN\_off": True},

    }

    for name, cfg in scenarios.items():

        # Stały krok

        t\_fixed, res\_fixed = rk4\_fixed(model, initial\_conditions, t0, tf, h, cfg)

        plt.figure(figsize=(10, 6))

        for label in labels:

            plt.plot(t\_fixed / 60, res\_fixed[:, index\_map[label]], label=label)

        plt.yscale("log")

        plt.title(f"Scenariusz {name} (RK4 stały krok)")

        plt.xlabel("Czas [h]")

        plt.ylabel("Liczba cząsteczek (log)")

        plt.legend()

        plt.grid(True, which="both", linestyle="--", linewidth=0.5)

        plt.tight\_layout()

        plt.savefig(f"scenariusz\_{name}\_stały\_krok.png", dpi=300)

        plt.close()

        # Adaptacyjny krok

        t\_adapt, res\_adapt = rk4\_adaptive(model, initial\_conditions, t0, tf, h, cfg)

        plt.figure(figsize=(10, 6))

        for label in labels:

            plt.plot(t\_adapt / 60, res\_adapt[:, index\_map[label]], label=label)

        plt.yscale("log")

        plt.title(f"Scenariusz {name} (RK4 adaptacyjny)")

        plt.xlabel("Czas [h]")

        plt.ylabel("Liczba cząsteczek (log)")

        plt.legend()

        plt.grid(True, which="both", linestyle="--", linewidth=0.5)

        plt.tight\_layout()

        plt.savefig(f"scenariusz\_{name}\_adaptacyjny.png", dpi=300)

        plt.close()

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**4. Wykresy wynikowe (48h)**

**Dołączono osobno po dwa wykresy dla każdego scenariusza:**

* scenariusz\_X\_staly\_krok.png – wynik z RK4 (stały krok)
* scenariusz\_X\_adaptacyjny.png – wynik z RK4 (zmienny krok)

Wykresy zawierają zmienne: p53, MDMcyto, MDMn, PTEN. Skala jest logarytmiczna, a oś czasu wyrażona w godzinach.

A graph with different colored lines

AI-generated content may be incorrect.

A graph of different colored lines

AI-generated content may be incorrect.

A graph of different colored lines

AI-generated content may be incorrect.

A graph of different colored lines

AI-generated content may be incorrect.

A graph with different colored lines

AI-generated content may be incorrect.

A graph of different colored lines

AI-generated content may be incorrect.

A graph of different colored lines

AI-generated content may be incorrect.

A graph of different colored lines

AI-generated content may be incorrect.

**5. Wnioski**

1. MDMn spada w każdym ze scenariuszy oprócz podstawowego A.

2. W scenariuszu Terapii obserwujemy gwałtowny spadek liczebności MDMcyto.

3. p53 rośnie w każdym ze scenariuszy oprócz podstawowego A.