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.

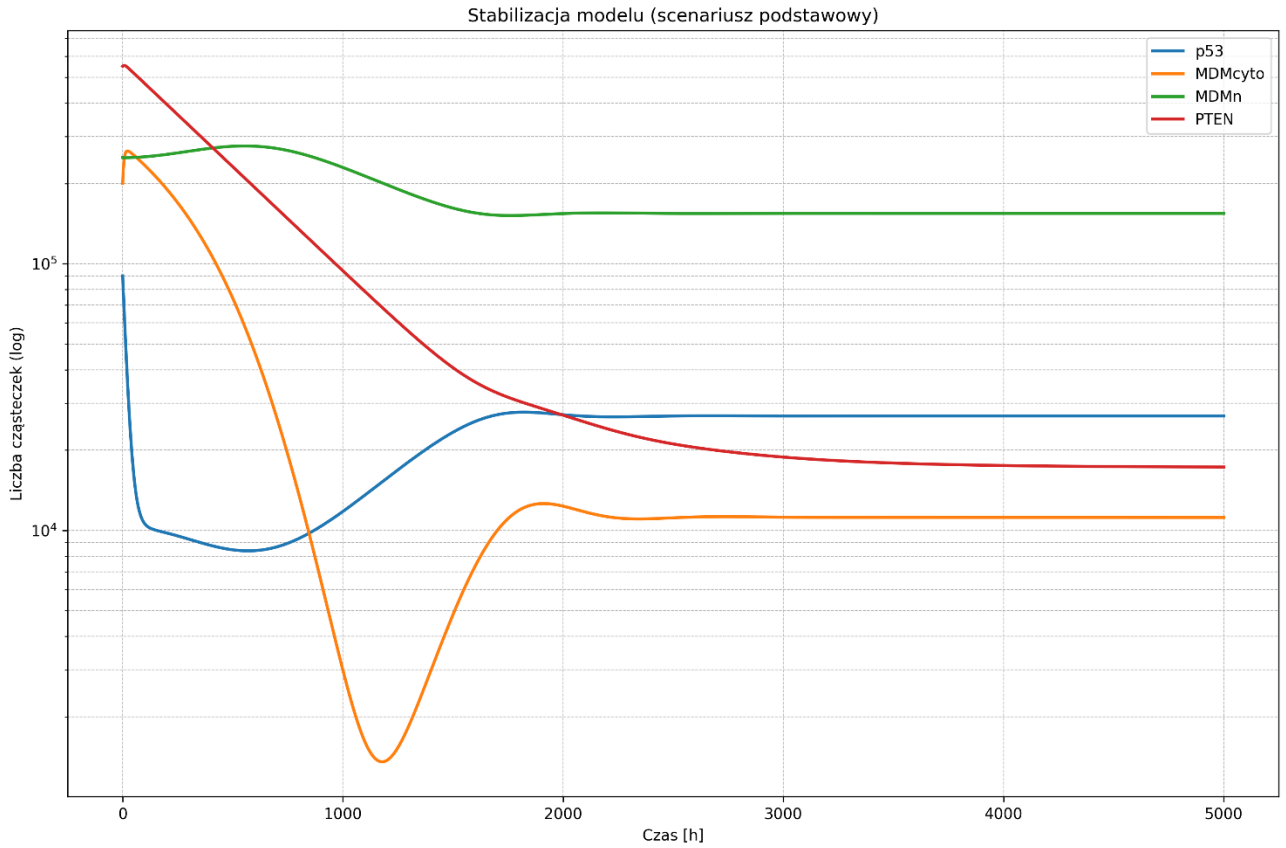
Całość znajduje się w pliku main.py z funkcją main() jako punktem wejścia. Gotowy plik .exe został wygenerowany narzędziem pyinstaller i nie wymaga dodatkowego oprogramowania do uruchomienia. Dołączyliśmy też plik requirements.txt, który zawiera informację o wszystkich potrzebnych bibliotekach.

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.



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.

```python
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()
```

```python
if __name__ == "__main__":
    main()
```

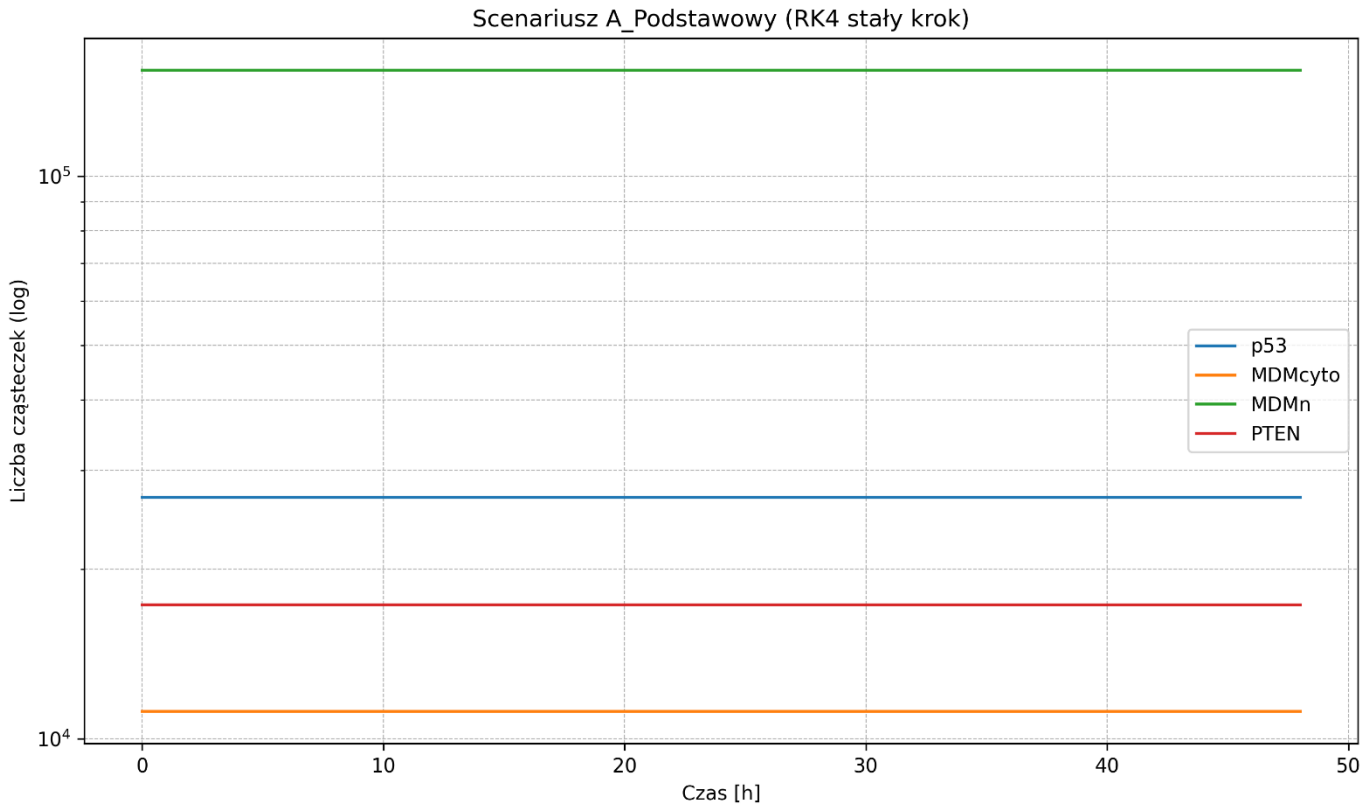
---

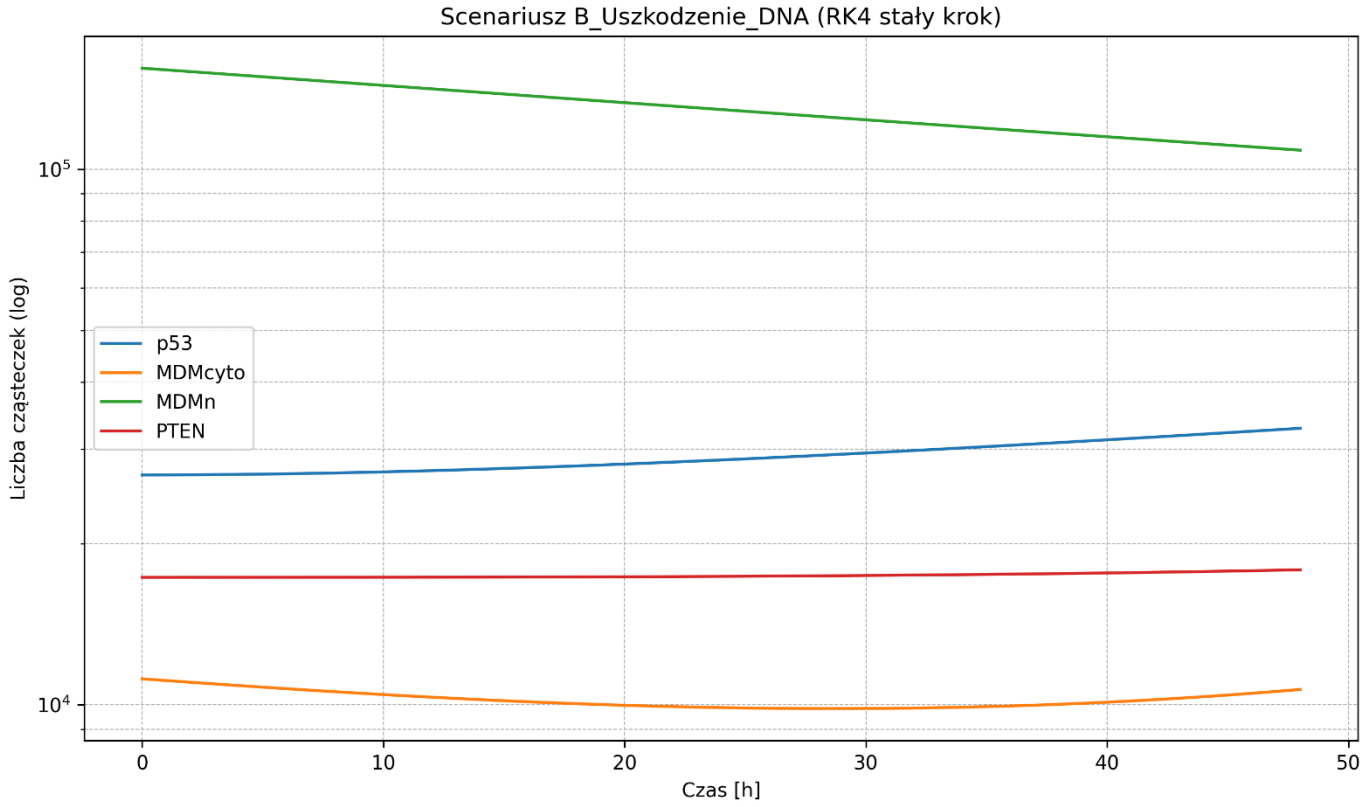
4. Wykresy wynikowe (48h)

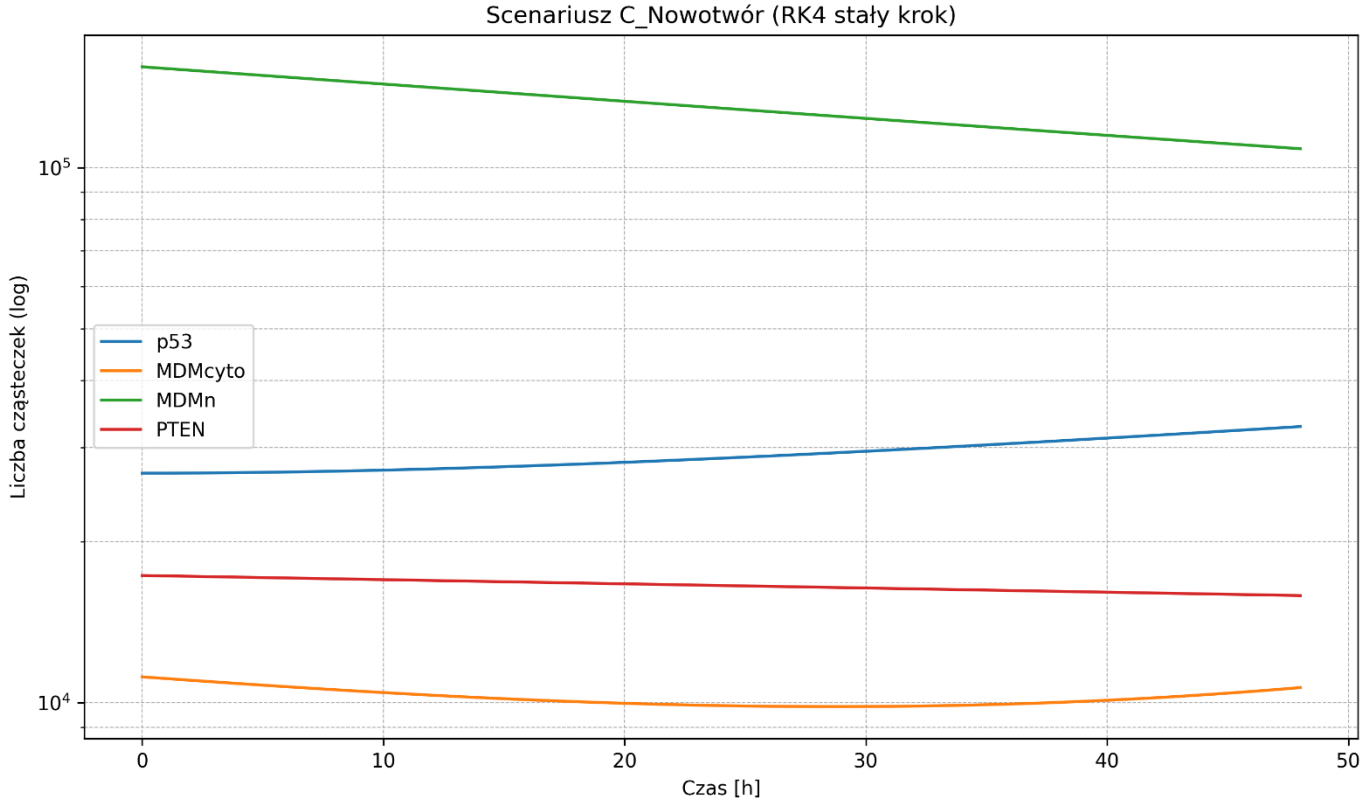
Dołączono osobno po dwa wykresy dla każdego scenariusza:

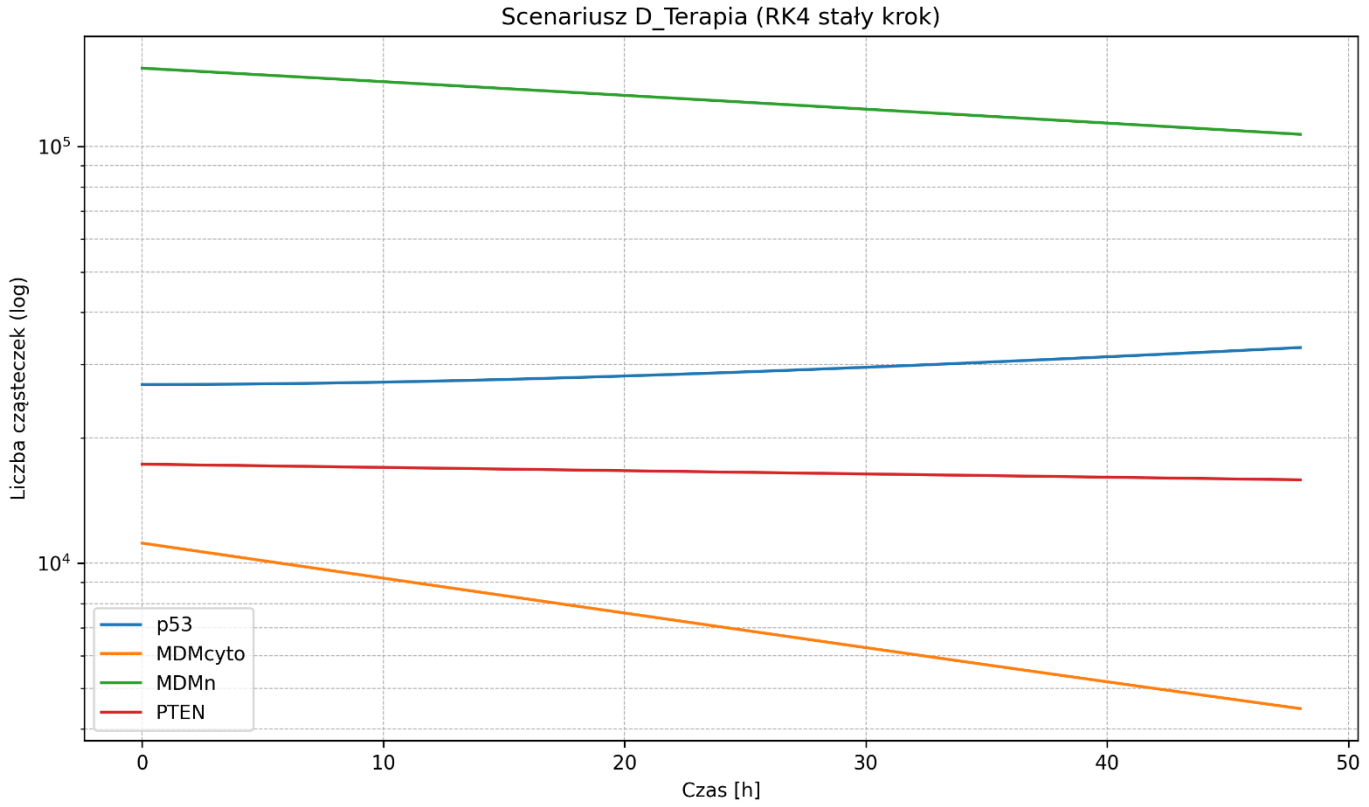
- scenariusz_X_staly_krok.png – wynik z RK4 (stały krok)
- scenariusz_X_adaptacyjny.png – wynik z RK4 (zmienny krok)

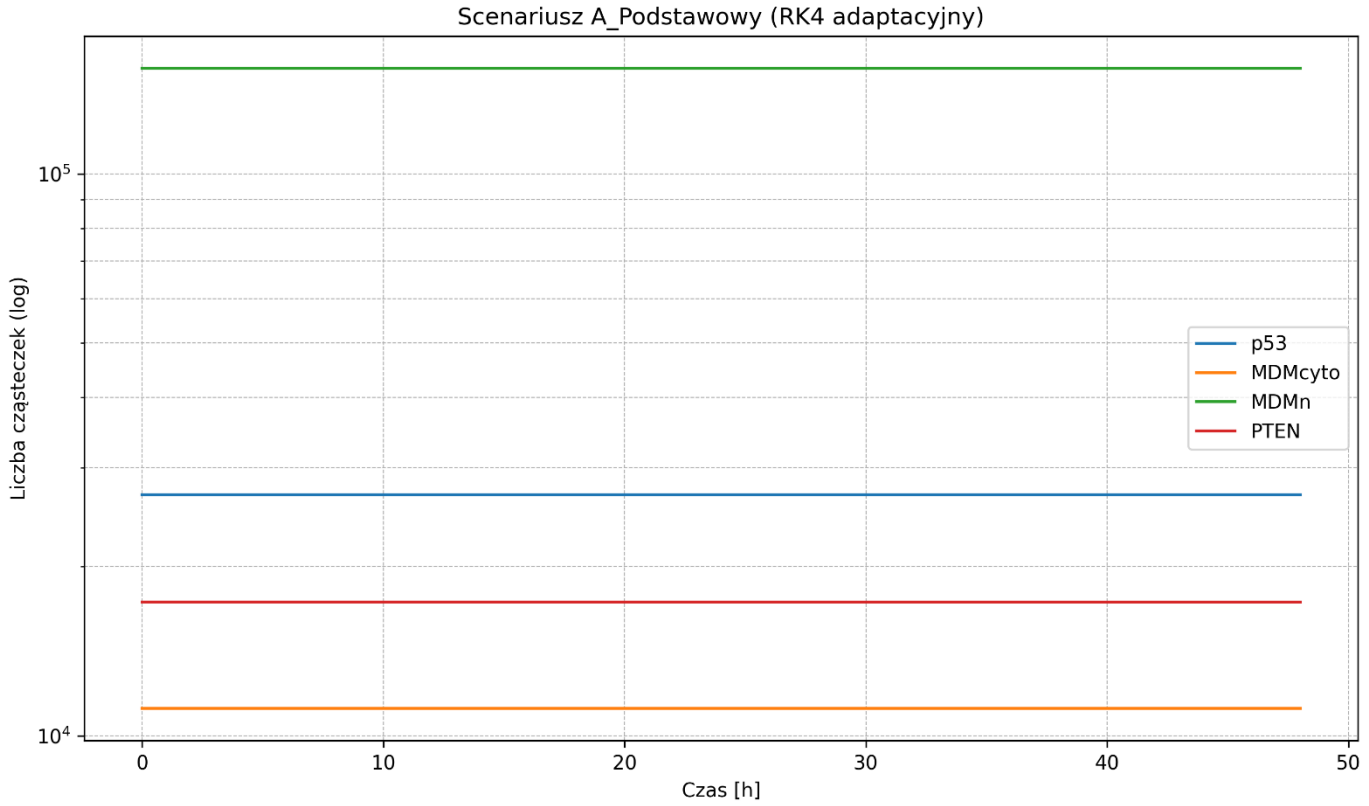
Wykresy zawierają zmienne: p53, MDMcyto, MDMn, PTEN. Skala jest logarytmiczna, a oś czasu wyrażona w godzinach.

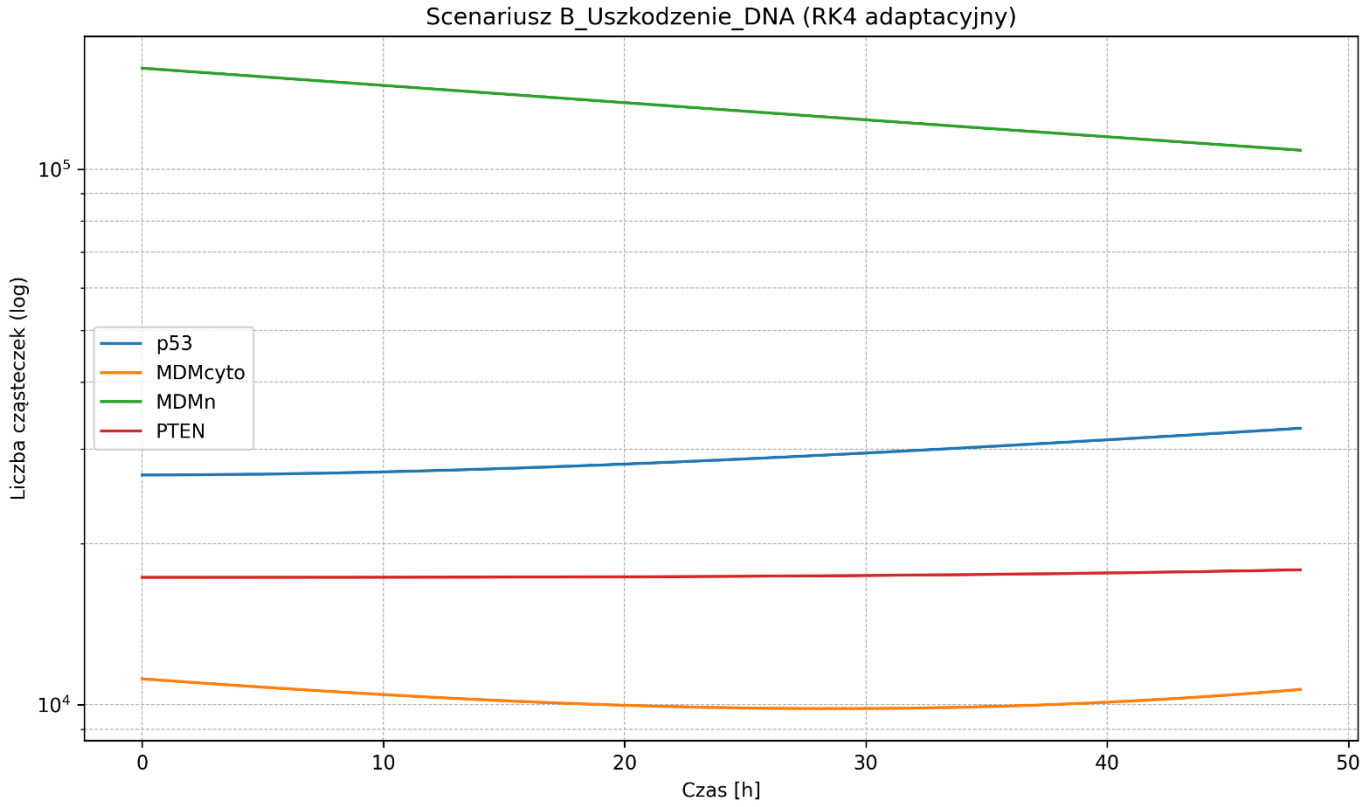


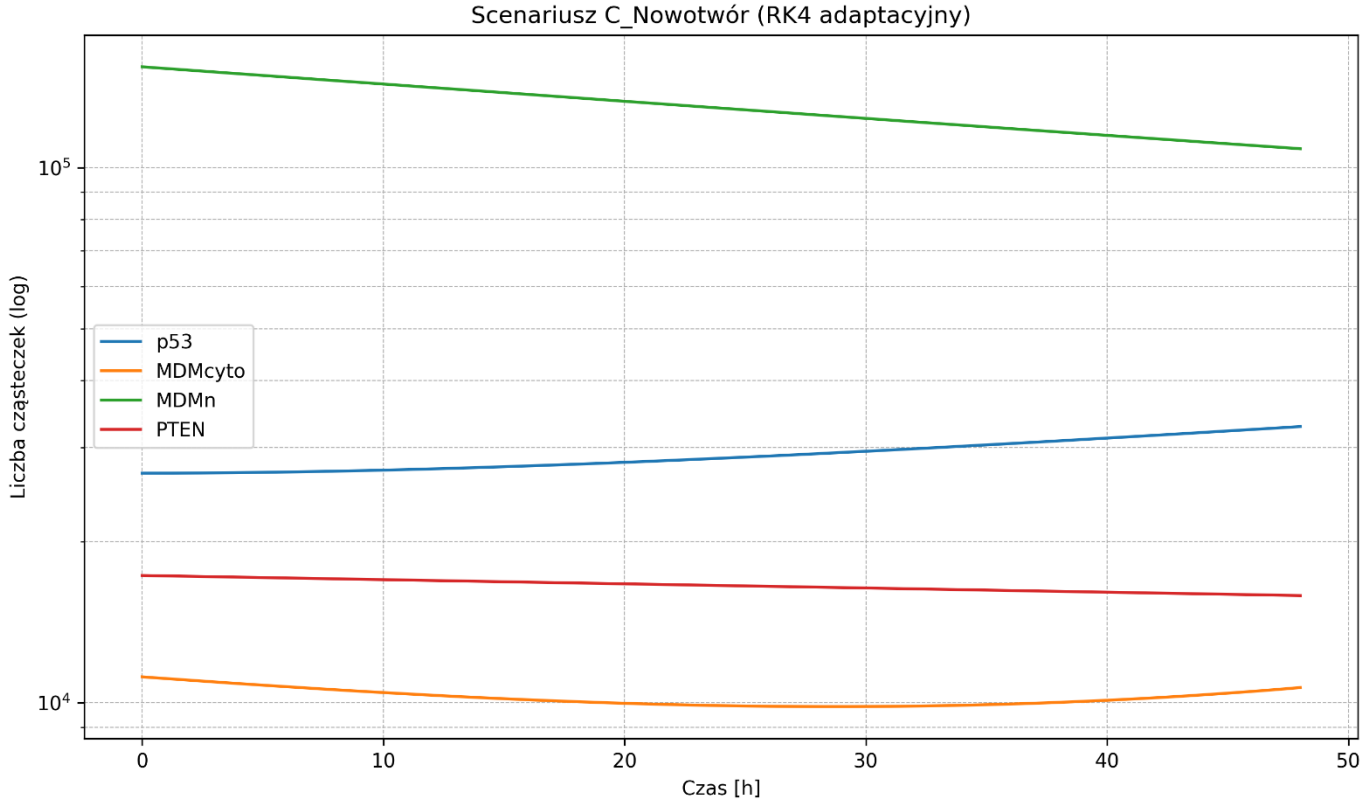

Scenariusz B_Uszkodzenie_DNA (RK4 stały krok)
Liczba cząsteczek (log)
10^5
10^4
p53
MDMcyto
MDMn
PTEN
0
10
20
30
40
50
Czas [h]

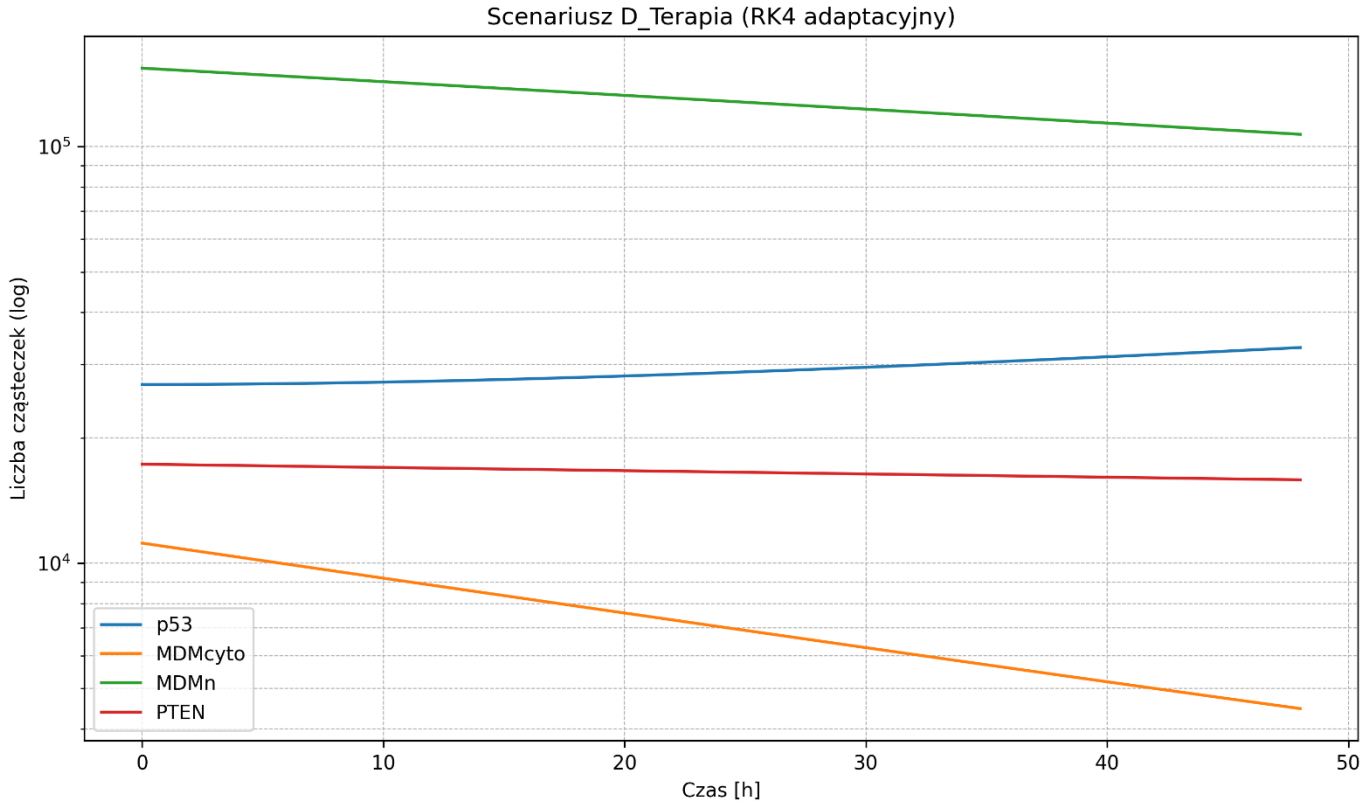

Scenariusz C_Nowotwór (RK4 stały krok)
Liczba cząsteczek (log)
10^5
10^4
p53
MDMcyto
MDMn
PTEN
0
10
20
30
40
50
Czas [h]


Scenariusz D_Terapia (RK4 stały krok)
Liczba cząsteczek (log)
10^5
10^4
p53
MDMcyto
MDMn
PTEN
0
10
20
30
40
50
Czas [h]


Scenariusz A_Podstawowy (RK4 adaptacyjny)
Liczba cząsteczek (log)
10^5
10^4
p53
MDMcyto
MDMn
PTEN
0
10
20
30
40
50
Czas [h]


Scenariusz B_Uszkodzenie_DNA (RK4 adaptacyjny)
Liczba cząsteczek (log)
10^5
10^4
p53
MDMcyto
MDMn
PTEN
0
10
20
30
40
50
Czas [h]


Scenariusz C_Nowotwór (RK4 adaptacyjny)
Liczba cząsteczek (log)
10^5
10^4
p53
MDMcyto
MDMn
PTEN
0
10
20
30
40
50
Czas [h]



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.