Symulacja procesów biologicznych

Ćwiczenie: 2 – Symulacje stochastyczne
Autor: Wojciech Laskowski, Bioinformatyka, rok III

Uwaga: Aby plik *gillespie_model.exe* zadziałał – należy w tym samym folderze mieć plik wejściowy *bio_model.json*

Symulacje stochastyczne:
Zaimplementowano podstawowy algorytm Gillespiego jako funkcja przyjmująca na wejściu dowolny model zapisany za pomocą równań reakcji.

Wersja implementacji 1+2

Dokonano symulacji modelu matematycznego o poniższych parametrach, a następnie zilustrowano wyniki załączonymi poniżej wykresami.

Parametr	Znaczenie	Opis biologiczny
p1	produkcja p53	Stała szybkość produkcji białka p53 w komórce.
p2	produkcja MDM2	Szybkość transkrypcji MDM2 indukowana przez p53.
p3	produkcja PTEN	Szybkość transkrypcji PTEN aktywowana przez p53.

Parametr	Znaczenie	Opis biologiczny
d1	degradacja p53	Zależna od poziomu MDM2 w jądrze. Im więcej MDM2, tym silniejsze hamowanie p53.
d2	degradacja MDM2	Degradacja MDM2 (zarówno jądrowego, jak i cytoplazmatycznego).
d3	degradacja PTEN	Zmniejszenie ilości PTEN z czasem.

Parametr	Znaczenie	Opis biologiczny
k1	transport MDM2	Kontroluje przepływ MDM2 z cytoplazmy do jądra.
k2	aktywacja transkrypcji	Stała nasycenia dla aktywacji transkrypcji MDM2/PTEN przez p53.
k3	hamowanie transportu MDM2	Zależność od poziomu PTEN — hamuje migrację MDM2.

Modyfikacja	Znaczenie	Efekt w modelu
dna_damage	uszkodzenie DNA	Gdy True, stabilizacja p53 przez obniżenie d2. Gdy False, degradacja MDM2 (a tym samym p53) jest silniejsza.
siRNA	terapia siRNA	Gdy True, tłumiona produkcja MDM2 (p2 *= 0.02).
pten_active	aktywność PTEN	Gdy False, wyłączona jest produkcja PTEN (p3 = 0).

Dla każdego scenariusza możliwe jest włączenie lub wyłączenie:

- uszkodzeń DNA (dna_damage)
- działania genu PTEN (pten_active)
- działania terapii siRNA (siRNA)

Dynamicznie modyfikowane parametry:

- p2 *= 0.02 przy obecności siRNA (spadek produkcji MDM2)
- p3 = 0 przy nieaktywnym PTEN (brak produkcji)
- d2 *= 0.1 przy braku uszkodzenia DNA (mniejsza degradacja MDM2)

Każdy scenariusz generuje osobny wykres, który składa się z trzech realizacji. Każda z nich przedstawia przedstawiający czasowy przebieg ilości cząsteczek białek w logarytmicznej skali, z osią X wyrażoną w minutach (zakres 0–2880 min).

Wygenerowane pliki PNG są zapisywane w tym samym folderze, w którym znajduje się program.

Wyniki symulacji i wykresy

Symulacja została przeprowadzona w czterech scenariuszach:

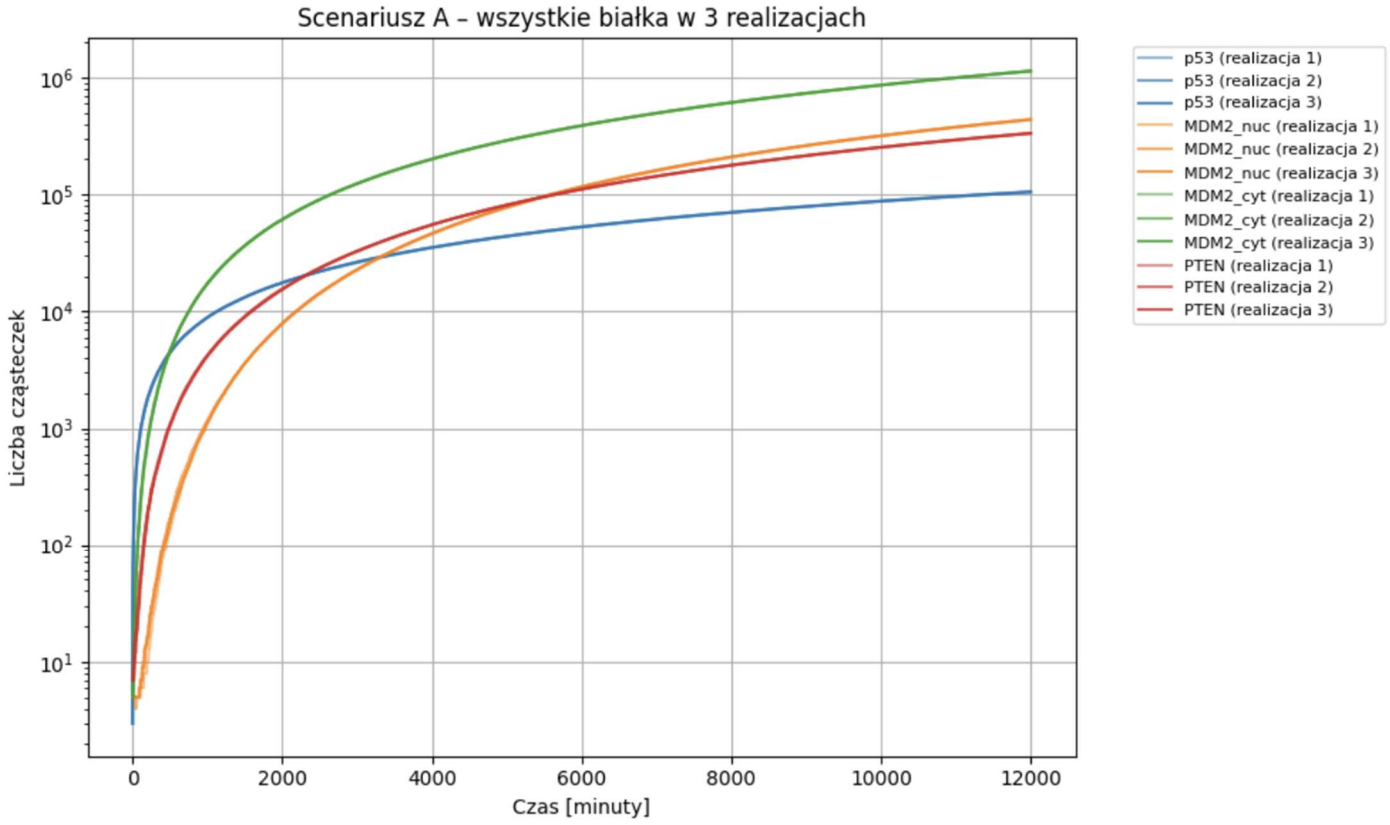
1. Scenariusz A – Podstawowy:
    - Brak uszkodzeń DNA
    - Aktywny PTEN
    - Brak siRNA

2. Scenariusz B – Uszkodzenie DNA (komórka zdrowa):
    - Uszkodzone DNA
    - Aktywny PTEN
    - Brak siRNA

3. Scenariusz C – Komórka nowotworowa:
    - Uszkodzone DNA
    - Nieaktywny PTEN
    - Brak siRNA

4. Scenariusz D – Terapia:
    - Uszkodzone DNA
    - Nieaktywny PTEN
    - Terapia siRNA włączona

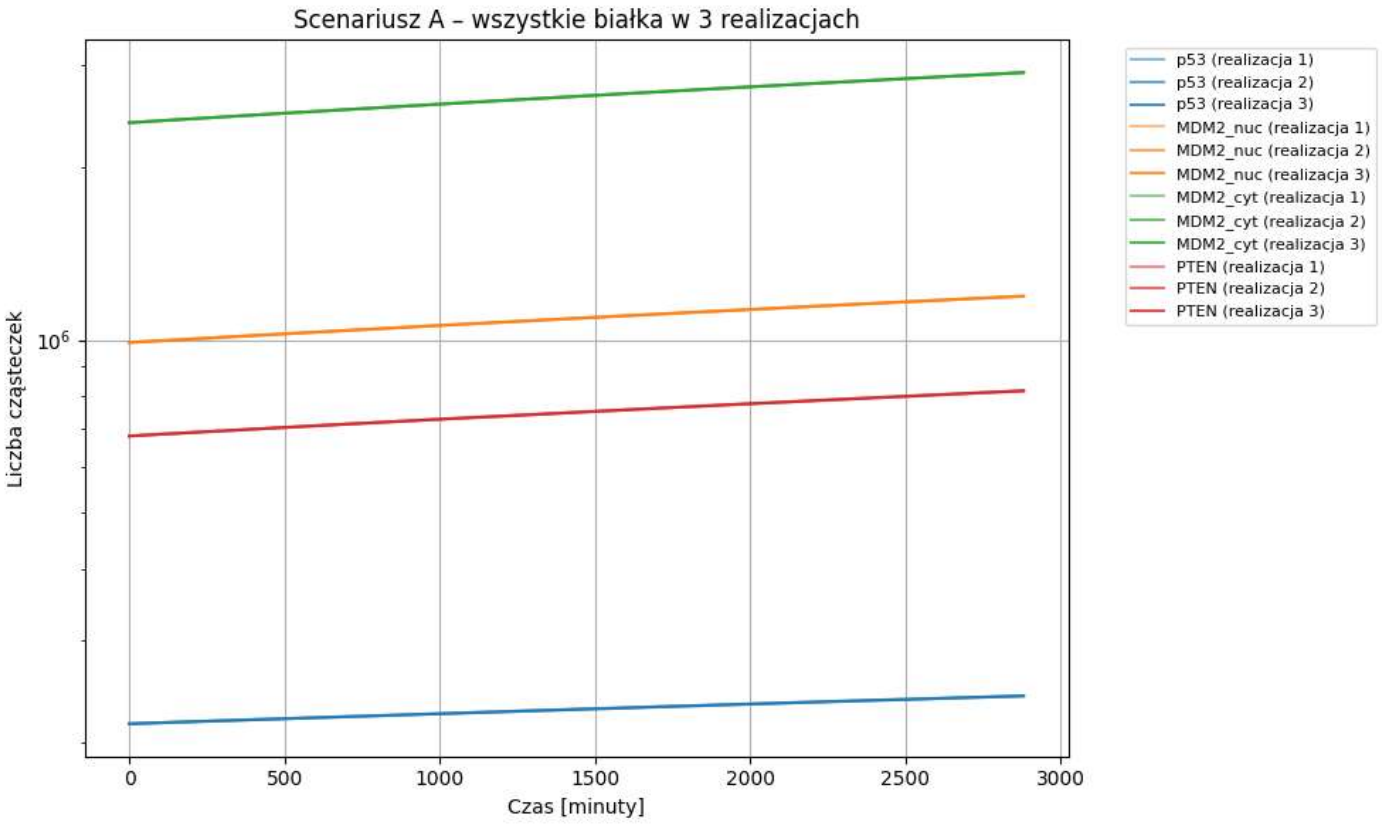
Wykres do ustalenia stanu początkowego – poniższy wykres przedstawia symulację dla 12000 minut.

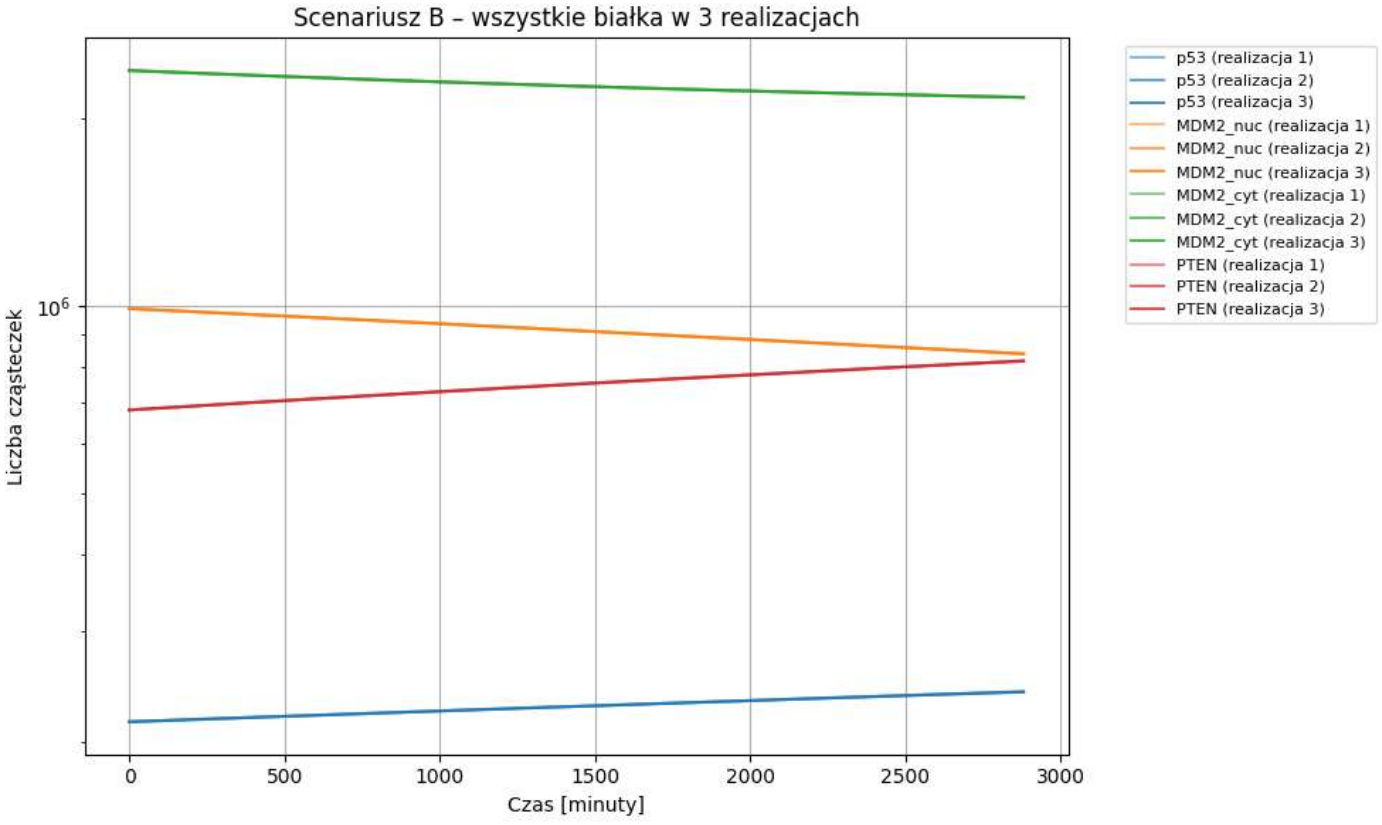


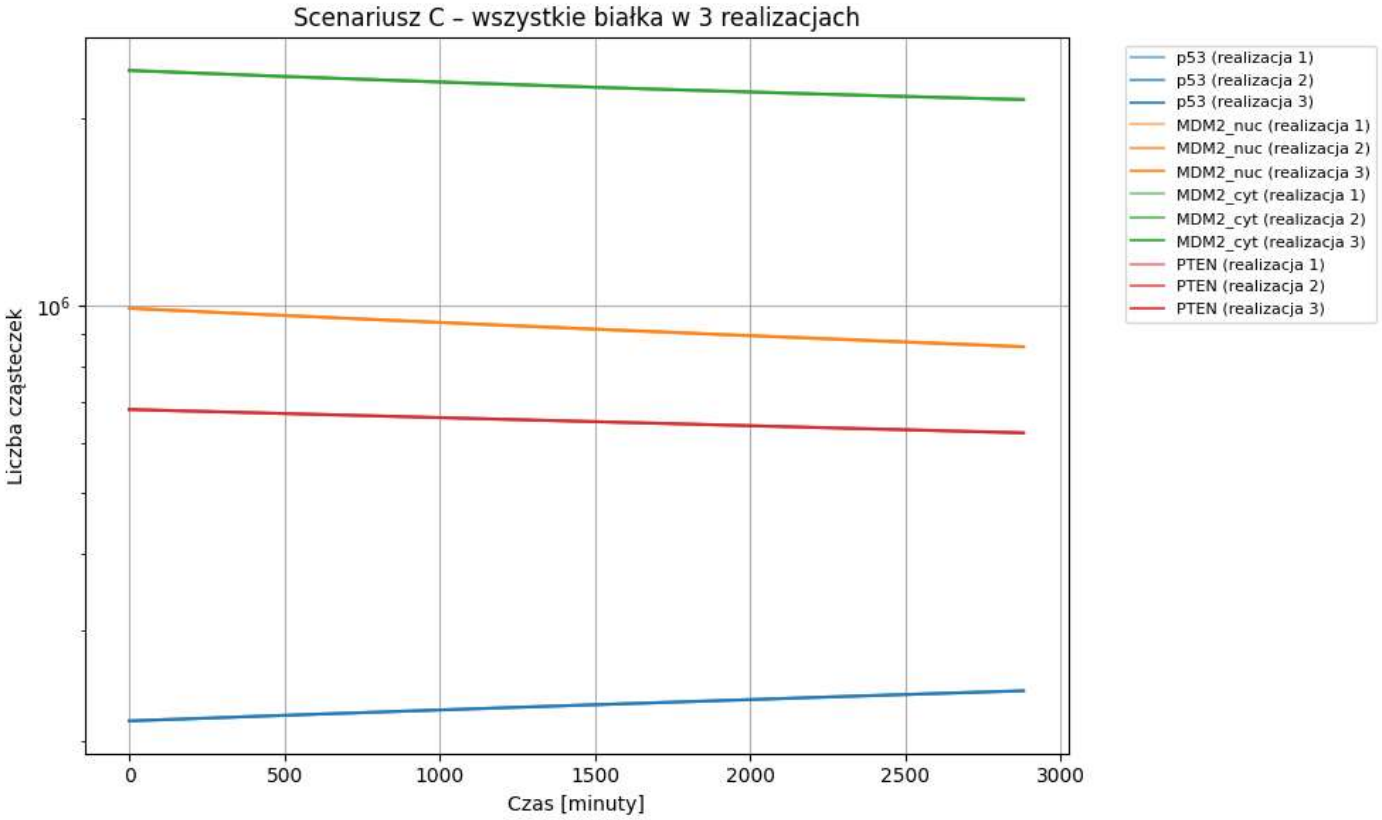
Wykres stanu podstawowego komórki (scenariusz A) był wykonane również dla czasu 20 000 minut, który dał poniższe wartości początkowe po względnym unormowaniu krzywych:

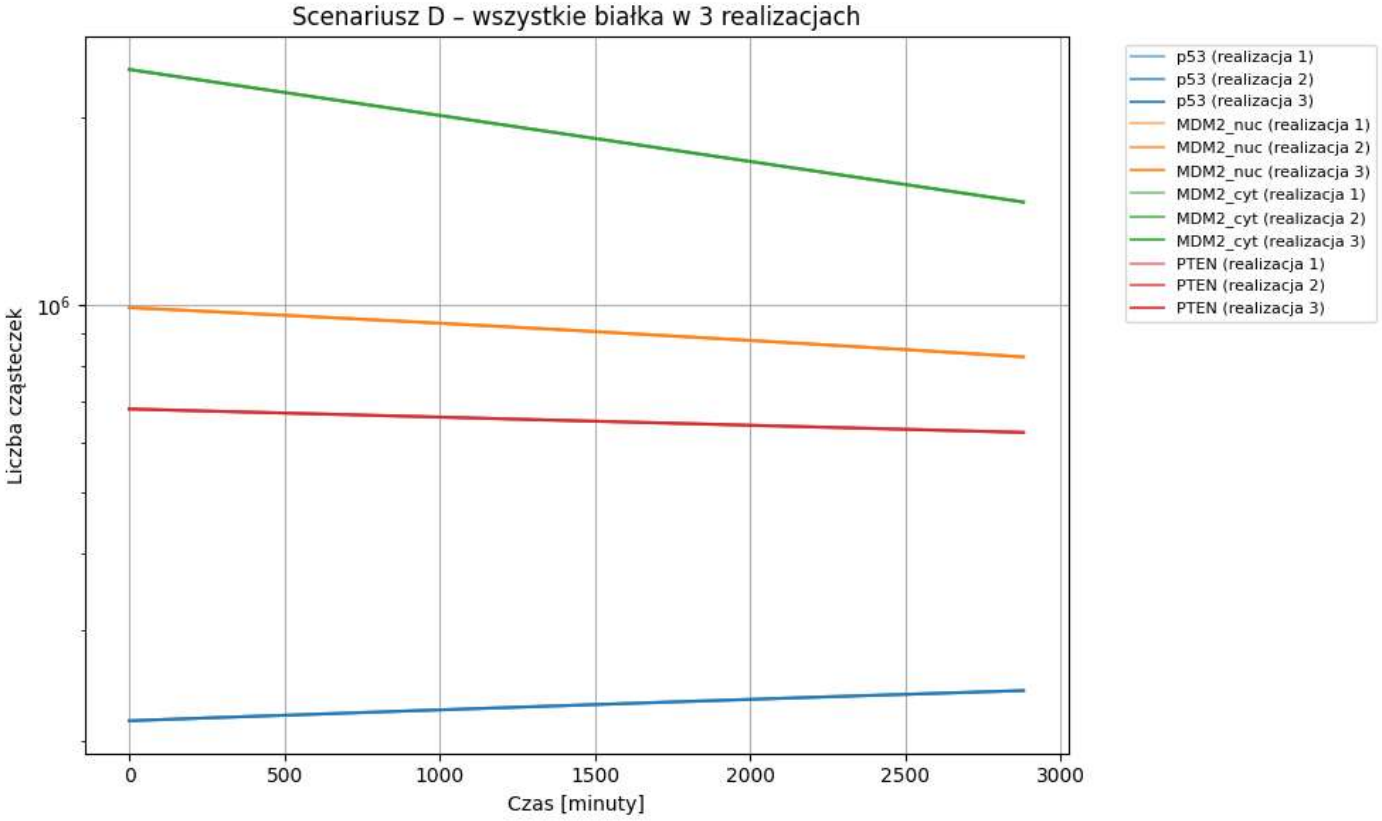
"initial_state": {
"p53": 215100,
"MDM2_nuc": 989800,
"MDM2_cyt": 2386500,
"PTEN": 680700
},

Na wszystkich wykresach uwzględniono 3 realizacje dla każdego z białek.
Są mało widoczne, gdyż nie różnią się znacząco od siebie, a zatem krzywe się nachodzą.


Scenariusz A – wszystkie białka w 3 realizacjach
Liczba cząsteczek
10^6
0
500
1000
1500
2000
2500
3000
Czas [minuty]
p53 (realizacja 1)
p53 (realizacja 2)
p53 (realizacja 3)
MDM2_nuc (realizacja 1)
MDM2_nuc (realizacja 2)
MDM2_nuc (realizacja 3)
MDM2_cyt (realizacja 1)
MDM2_cyt (realizacja 2)
MDM2_cyt (realizacja 3)
PTEN (realizacja 1)
PTEN (realizacja 2)
PTEN (realizacja 3)


Scenariusz B – wszystkie białka w 3 realizacjach
Liczba cząsteczek
10^6
0
500
1000
1500
2000
2500
3000
Czas [minuty]
p53 (realizacja 1)
p53 (realizacja 2)
p53 (realizacja 3)
MDM2_nuc (realizacja 1)
MDM2_nuc (realizacja 2)
MDM2_nuc (realizacja 3)
MDM2_cyt (realizacja 1)
MDM2_cyt (realizacja 2)
MDM2_cyt (realizacja 3)
PTEN (realizacja 1)
PTEN (realizacja 2)
PTEN (realizacja 3)


Scenariusz C – wszystkie białka w 3 realizacjach
Liczba cząsteczek
10^6
0
500
1000
1500
2000
2500
3000
Czas [minuty]
p53 (realizacja 1)
p53 (realizacja 2)
p53 (realizacja 3)
MDM2_nuc (realizacja 1)
MDM2_nuc (realizacja 2)
MDM2_nuc (realizacja 3)
MDM2_cyt (realizacja 1)
MDM2_cyt (realizacja 2)
MDM2_cyt (realizacja 3)
PTEN (realizacja 1)
PTEN (realizacja 2)
PTEN (realizacja 3)


Scenariusz D – wszystkie białka w 3 realizacjach
Liczba cząsteczek
10^6
0
500
1000
1500
2000
2500
3000
Czas [minuty]
p53 (realizacja 1)
p53 (realizacja 2)
p53 (realizacja 3)
MDM2_nuc (realizacja 1)
MDM2_nuc (realizacja 2)
MDM2_nuc (realizacja 3)
MDM2_cyt (realizacja 1)
MDM2_cyt (realizacja 2)
MDM2_cyt (realizacja 3)
PTEN (realizacja 1)
PTEN (realizacja 2)
PTEN (realizacja 3)

Kod:

Krótki opis działania:

1. Wczytanie modelu: Skrypt ładuje z pliku JSON dane modelu biologicznego, zawierające listę białek, reakcje chemiczne, parametry, stan początkowy oraz różne scenariusze eksperymentalne.
2. Symulacja Gillespie'go:
   - Dla wybranego scenariusza modyfikuje parametry modelu (np. zmienia szybkości reakcji w zależności od obecności uszkodzenia DNA, aktywności PTEN czy obecności siRNA).
   - Symuluje przebieg reakcji losując czas i typ kolejnej reakcji na podstawie bieżących stężeń i parametrów.
   - Aktualizuje stan systemu po każdej reakcji i zapisuje go wraz z aktualnym czasem.
3. Wielokrotne realizacje:
   - Dla każdego scenariusza wykonuje 3 niezależne realizacje symulacji, by uwzględnić stochastyczną zmienność wyników – mimo tego, wychodzą bardzo zbliżone do siebie, niemal niezauważalne na wykresach. Różnie widoczne są dopiero gdy przyjrzymy się parametrom widocznym w terminalu podczas przebiegu programu.
4. Wizualizacja wyników:
   - Rysuje wykresy przebiegu czasowego liczby cząsteczek poszczególnych białek (na skali logarytmicznej) dla wszystkich trzech realizacji i zapisuje je do plików PNG.

```
import json
import random
import math
import copy
import matplotlib.pyplot as plt

# algorytm
def gillespie(model_data, scenario_name, max_time):
    species = model_data["species"]                 # lista białek
    reactions = model_data["reactions"]             # reakcje chemiczne
    base_params = model_data["params"]              # bazowe parametry
    initial_state = model_data["initial_state"]     # stan początkowy
    scenarios = model_data["scenarios"]             # scenariusze

    # kopia parametrów, żeby nie nadpisać oryginału
    params = copy.deepcopy(base_params)
    flags = scenarios[scenario_name]

    # modyfikacje zależne od scenariusza
    if not flags["dna_damage"]:
        params["d2"] *= 0.1
    if not flags["pten_active"]:
        params["p3"] = 0.0
    if flags["siRNA"]:
        params["p2"] *= 0.02

    # symulacja
    state = copy.deepcopy(initial_state)            # oecny stan cząsteczek
    time_series = [(0.0, copy.deepcopy(state))]     # lista: czas, stan
    time = 0.0
    steps = 0
```

```
    # symulacja trwa dopóki pętla nie przekroczy określonego z góry czasu
    while time < max_time:
        propensities = []  # Lista prawdopodobieństw reakcji --> propensje
        for rxn in reactions:
            local_vars = {**state, **params}  # zmienne stanu, parametry
            try:
                a = eval(rxn["rate"], {}, local_vars)  # obliczanie szybkości reakcji
            except Exception:
                a = 0.0
            propensities.append(max(a, 0.0))  # propensja musi być większa lub równa 0

        a0 = sum(propensities)  # suma propensji
        if a0 <= 0:
            print(f"Propensje spadły do zera na {steps} kroku.")
            time += (max_time - time)  # wtedy --> skok do końca czasu
            time_series.append((time, copy.deepcopy(state)))
            break

        # losowanie czasu do następnej reakcji (tau),  wybóru reakcji
        r1 = random.random()
        r2 = random.random()
        tau = (1.0 / a0) * math.log(1.0 / r1)
        time += tau

        # losujemy która reakcja ma zajść
        threshold = r2 * a0
        cumulative = 0.0
        for i, a in enumerate(propensities):
            cumulative += a
            if cumulative >= threshold:
                selected_rxn = reactions[i]
                break

        # aktualizacja stanu po reakcji
        for sp, count in selected_rxn.get("reactants", {}).items():
            state[sp] -= count
        for sp, count in selected_rxn.get("products", {}).items():
            state[sp] += count

        # zapisywanie stanu i dodanie kroku
        time_series.append((time, copy.deepcopy(state)))
        steps += 1

    # informacja końcowa po symulacji
    print(f"Koniec: czas końcowy = {time:.2f} min, liczba kroków = {steps}")
    print(f"Ostatni stan: {state}")
    return time_series

# wykresy
def plot_realizations(realizations, species, scenario_name):
    plt.figure(figsize=(10, 6))
    colors = plt.cm.tab10.colors  # kolory
```

```
    # dla każdego białka i każdej realizacji rysujemy przebieg czasowy
    for i, sp in enumerate(species):
        for j, result in enumerate(realizations):
            times = [t for t, _ in result]
            values = [state[sp] for _, state in result]
            label = f"{sp} (realizacja {j+1})"
            plt.plot(times, values, label=label, color=colors[i % len(colors)], alpha=0.5
+ 0.15 * j)

    plt.title(f"Scenariusz {scenario_name} – wszystkie białka w 3 realizacjach")
    plt.xlabel("Czas [minuty]")
    plt.ylabel("Liczba cząsteczek")
    plt.yscale("log")
    plt.grid(True)
    plt.legend(fontsize=8, bbox_to_anchor=(1.05, 1), loc='upper left')
    plt.tight_layout()
    plt.savefig(f"glsp_scenario_{scenario_name}.png", bbox_inches="tight")
    plt.close()

# uruchomienie
if __name__ == "__main__":
    with open("bio_model.json", "r") as f:
        model_data = json.load(f)  # wczytanie danych z pliku JSON

    # dla każdego scenariusza --> 3 realizacje
    for scenario in model_data["scenarios"].keys():
        print(f"Symulacja scenariusza {scenario} w toku")
        realizations = []
        for i in range(3):
            result = gillespie(model_data, scenario, max_time=2880)  # 48 godzin (2880
minut)
            realizations.append(result)
            print(f"Realizacja {i+1} dla scenariusza {scenario} zakończona.\n")

        # wykres po 3 realizacjach
        plot_realizations(realizations, model_data["species"], scenario)
        print(f"Wykres zapisany do pliku: glsp_scenario_{scenario}.png\n")
```

Plik JSON z wprowadzonymi danymi do modelu

Plik bio_model.json jest zapisany w formacie JSON i zawiera pełny opis układu biologicznego modelowanego przez algorytm Gillespiego. Struktura pliku dzieli się na pięć głównych sekcji:

1. **species** – lista nazw wszystkich cząsteczek (np. białek) w modelu
2. **initial_state** – stan początkowy układu (liczba cząsteczek)
   - Zawiera słownik, w którym kluczami są nazwy cząsteczek z sekcji species, a wartościami ich początkowe ilości.
3. **params** – parametry reakcji
   - Zawiera zestaw parametrów wykorzystywanych w wyrażeniach określających szybkości reakcji (rate). Są to m.in. stałe produkcji (p1, p2, p3), degradacji (d1, d2, d3) i inne parametry modelowe (k1, k2, k3).
4. **reactions** - lista reakcji chemicznych
   - Każda reakcja opisana jest jako słownik zawierający:
     - reactants: substraty (np. "p53": 1 oznacza 1 cząsteczkę p53)
     - products: produkty reakcji

   rate: wyrażenie matematyczne opisujące szybkość reakcji, zapisane jako string
5. **scenarios** – definicje warunków eksperymentalnych
   Każdy scenariusz symulacyjny (np. A, B, C, D) to zestaw flag, które wpływają na wartości parametrów w modelu:
   - dna_damage – czy występuje uszkodzenie DNA
   - siRNA – czy aktywna jest interferencja RNA (np. tłumienie ekspresji MDM2)
   - pten_active – czy aktywne jest białko PTEN

   W zależności od wartości tych flag, w kodzie modyfikowane są odpowiednie parametry (np. osłabienie degradacji, wyłączenie produkcji, itp.).


```
{
  "species": ["p53", "MDM2_nuc", "MDM2_cyt", "PTEN"],
  "initial_state": {
  "p53": 215100,
  "MDM2_nuc": 989800,
  "MDM2_cyt": 2386500,
  "PTEN": 680700
},
  "params": {
    "p1": 8.8,
    "p2": 440,
    "p3": 100,
    "d1": 1.375e-14,
    "d2": 0.0001375,
    "d3": 0.00003,
    "k1": 0.0001925,
    "k2": 100000,
    "k3": 150000
  },
  "reactions": [
    {
      "reactants": {},
      "products": { "p53": 1 },
      "rate": "p1"
    },
    {
      "reactants": { "p53": 1, "MDM2_nuc": 1 },
```

```
      "products": {},
      "rate": "d1 * p53 * MDM2_nuc"
    },
    {
      "reactants": { "p53": 1 },
      "products": { "p53": 1, "MDM2_cyt": 1 },
      "rate": "p2 * p53 / (p53 + k2)"
    },
    {
      "reactants": { "MDM2_cyt": 1 },
      "products": { "MDM2_nuc": 1 },
      "rate": "k1 * MDM2_cyt * (k3 / (k3 + PTEN))"
    },
    {
      "reactants": { "MDM2_cyt": 1 },
      "products": {},
      "rate": "d2 * MDM2_cyt"
    },
    {
      "reactants": { "MDM2_nuc": 1 },
      "products": {},
      "rate": "d2 * MDM2_nuc"
    },
    {
      "reactants": { "p53": 1 },
      "products": { "p53": 1, "PTEN": 1 },
      "rate": "p3 * p53 / (p53 + k2)"
    },
    {
      "reactants": { "PTEN": 1 },
      "products": {},
      "rate": "d3 * PTEN"
    }
  ],
  "scenarios": {
    "A": {
      "dna_damage": false,
      "siRNA": false,
      "pten_active": true
    },
    "B": {
      "dna_damage": true,
      "siRNA": false,
      "pten_active": true
    },
    "C": {
      "dna_damage": true,
      "siRNA": false,
      "pten_active": false
    },
    "D": {
      "dna_damage": true,
      "siRNA": true,
      "pten_active": false
```

```
    }
  }
}
```