Laboratorium 3
Emilia Gnatiuk i Julia Skoroszewska

Program służy do przeprowadzenia lokalnej i globalnej analizy wrażliwości deterministycznego modelu regulacji białka p53 w komórkach w odpowiedzi na uszkodzenia DNA.
Program analizuje wpływ parametrów modelu na poziom bialka p53 w dwóch scenariuszach.

lokalna analiza wrażliwości - obliczana jest funkcja wrażliwości, aproksymowana numerycznie przez różnice skończone. Zmiany poziomu p53 są porównywane z wartością bazową, wrażliwość jest normalizowana.

globalna analiza wrażliwości - metoda Sobola. Generowane są dwie macierze losowych zestawów parametrów. Dla każdego parametru i próbki obliczany jest wpływ zamiany kolumny A[i] na B[i].

Po uruchomieniu wyświetlane są wykresy wrażliwości lokalne w czasie, rankingi lokalne i globalne oraz poziom p53 przy zmianach parametrów.

```
import numpy as np
import matplotlib.pyplot as plt

#parametry modelu i scenariusze
params_nominal = {
    '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
}

sensitivity_constants = {
    'value_siRNA': 0.02,
    'value_PTEN_off': 0,
    'value_no_DNA_damage': 0.1
}

initial_conditions = {
    'p53': 26854,
    'mdmcyto': 11173,
    'mdmn': 17245,
    'pten': 154378
```

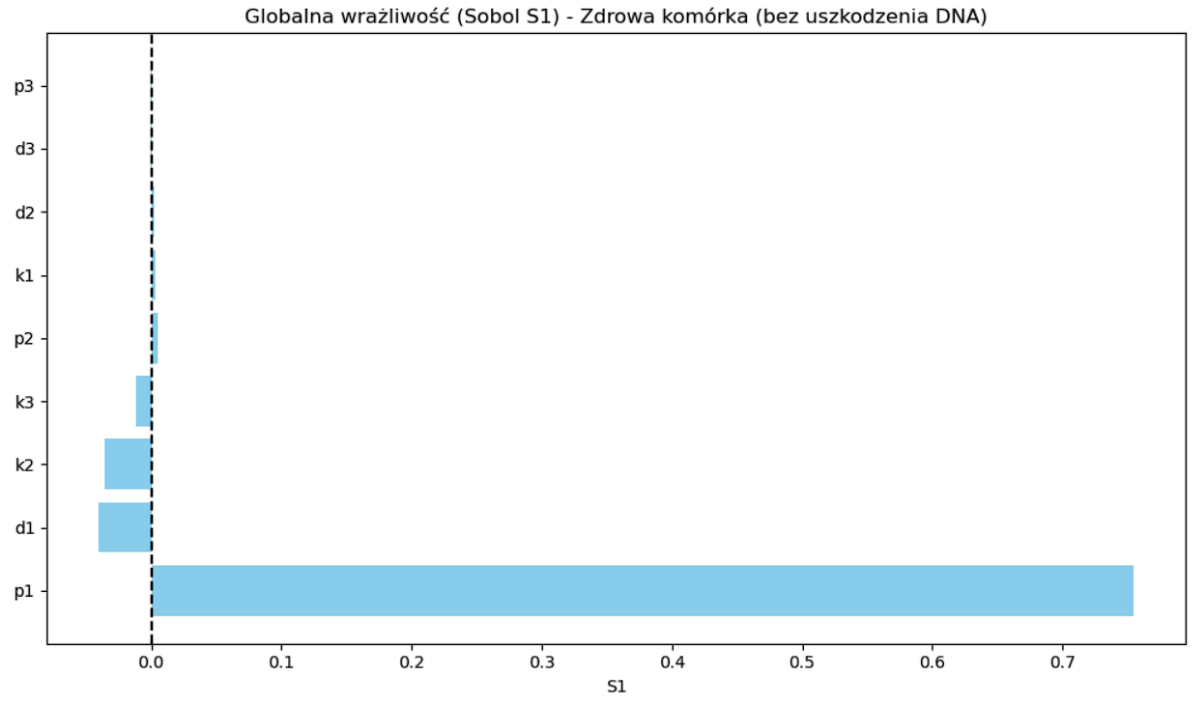
```
}

conditions = {
   "Zdrowa komórka (bez uszkodzenia DNA)": (False, False, True),
   "Nowotworowa komórka (z uszkodzeniem DNA)": (False, True,
False)
}

h = 6
iterations = int(48 * 60 / h)

#funkcje modelu
def f_p53(params, p53, mdmn):
   return params['p1'] - params['d1'] * p53 * mdmn**2

def f_mdmcyto(params, p53, mdmcyto, pten, siRNA=False,
no_DNA_damage=False):
   s_factor = sensitivity_constants['value_siRNA'] if siRNA else 1
   d_factor = sensitivity_constants['value_no_DNA_damage'] if
no_DNA_damage else 1
   return (
       params['p2'] * s_factor * (p53**4) / ((p53**4) +
(params['k2']**4))
       - params['k1'] * (params['k3']**2) / ((params['k3']**2) +
(pten**2)) * mdmcyto
       - params['d2'] * d_factor * mdmcyto
   )

def f_mdmn(params, mdmn, mdmcyto, pten, no_DNA_damage=False):
   d_factor = sensitivity_constants['value_no_DNA_damage'] if
no_DNA_damage else 1
   return (
       params['k1'] * (params['k3']**2) / ((params['k3']**2) +
(pten**2)) * mdmcyto
       - params['d2'] * d_factor * mdmn
   )

def f_pten(params, pten, p53, pten_off=False):
   if pten_off:
       return -params['d3'] * pten
   return params['p3'] * (p53**4) / ((p53**4) + (params['k2']**4))
- params['d3'] * pten

def RK4_step(params, state, h, siRNA=False, pten_off=False,
no_DNA_damage=False):
   p53, mdmcyto, mdmn, pten = state

   def calc_derivs(p53, mdmcyto, mdmn, pten):
```

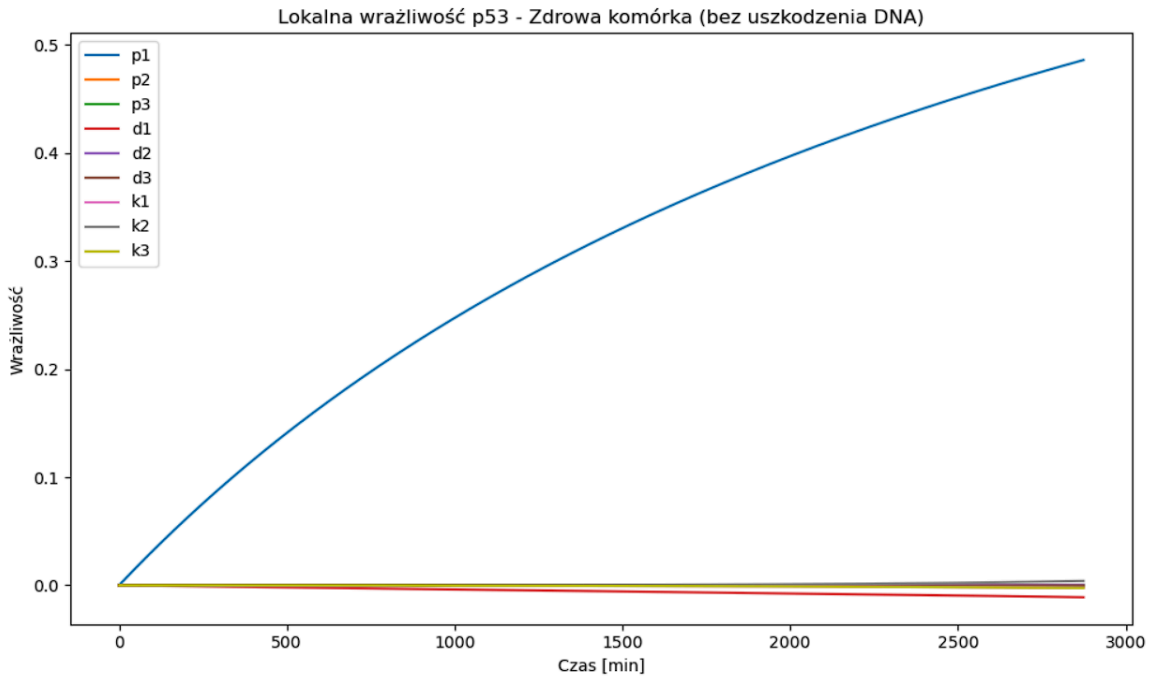
```
        return (
            f_p53(params, p53, mdmn),
            f_mdmcyto(params, p53, mdmcyto, pten, siRNA,
no_DNA_damage),
            f_mdmn(params, mdmn, mdmcyto, pten, no_DNA_damage),
            f_pten(params, pten, p53, pten_off)
        )

    k1 = calc_derivs(p53, mdmcyto, mdmn, pten)
    k2 = calc_derivs(*(p + h / 2 * k for p, k in zip((p53, mdmcyto,
mdmn, pten), k1)))
    k3 = calc_derivs(*(p + h / 2 * k for p, k in zip((p53, mdmcyto,
mdmn, pten), k2)))
    k4 = calc_derivs(*(p + h * k for p, k in zip((p53, mdmcyto,
mdmn, pten), k3)))

    return tuple(p + h / 6 * (k1i + 2 * k2i + 2 * k3i + k4i) for p,
k1i, k2i, k3i, k4i in zip((p53, mdmcyto, mdmn, pten), k1, k2, k3,
k4))

#analiza RK4

def RK4_run(params, initial_state, scenario, iterations, h):
    siRNA, pten_off, no_DNA_damage = scenario
    state = tuple(initial_state.values())
    time_series = {k: [] for k in ['t', 'p53', 'mdmcyto', 'mdmn',
'pten']}

    for i in range(iterations):
        time_series['t'].append(i * h)
        for name, val in zip(['p53', 'mdmcyto', 'mdmn', 'pten'],
state):
            time_series[name].append(val)
        state = RK4_step(params, state, h, siRNA, pten_off,
no_DNA_damage)

    return time_series

#analiza lokalna
def local_sensitivity(params_nominal, initial_state, scenario,
delta=1e-4):
    base = RK4_run(params_nominal, initial_state, scenario,
iterations, h)
    p53_base = np.array(base['p53'])

    sensitivities = {}
    for key in params_nominal:
        perturbed = params_nominal.copy()
```

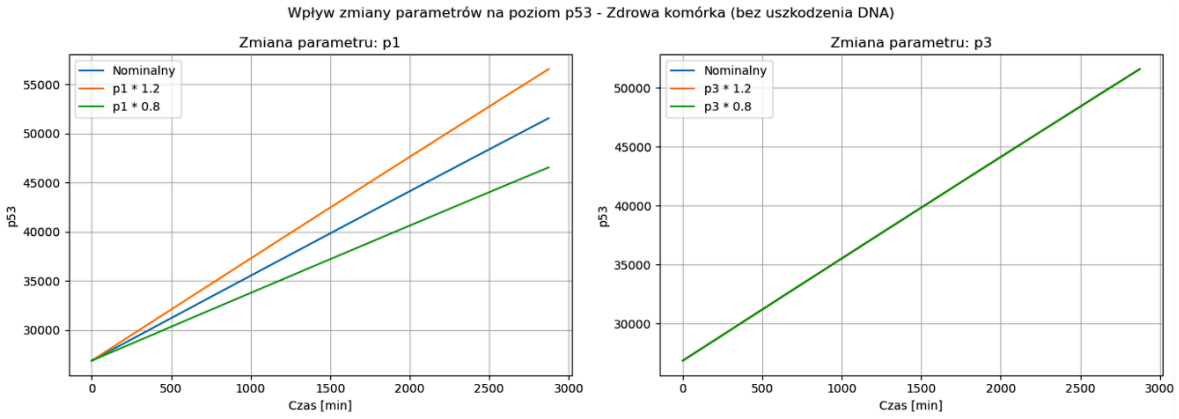
```
        perturb = params_nominal[key] * delta if
params_nominal[key] != 0 else delta
        perturbed[key] += perturb
        result = RK4_run(perturbed, initial_state, scenario,
iterations, h)
        p53_perturbed = np.array(result['p53'])
        S = (p53_perturbed - p53_base) / perturb
        S_norm = (params_nominal[key] / p53_base) * S
        sensitivities[key] = S_norm.tolist()

    return base['t'], sensitivities, p53_base

#analiza globalna (Sobol S1)

def sample_parameters(bounds, N):
    return np.random.uniform(
        low=[b[0] for b in bounds],
        high=[b[1] for b in bounds],
        size=(N, len(bounds))
    )

def run_model_end_value(param_set, initial_state, scenario,
param_names):
    p = dict(zip(param_names, param_set))
    p.update(sensitivity_constants)
    result = RK4_run(p, initial_state, scenario, iterations, h)
    return result['p53'][-1]

def global_sensitivity(bounds, param_names, initial_state,
scenario, N=512):
    A = sample_parameters(bounds, N)
    B = sample_parameters(bounds, N)
    fA = np.array([run_model_end_value(row, initial_state,
scenario, param_names) for row in A])
    fB = np.array([run_model_end_value(row, initial_state,
scenario, param_names) for row in B])
    var_f = np.var(fA, ddof=1)
    S1 = []

    for i in range(len(bounds)):
        AB = A.copy()
        AB[:, i] = B[:, i]
        fAB = np.array([run_model_end_value(row, initial_state,
scenario, param_names) for row in AB])
        S1.append(np.mean(fB * (fAB - fA)) / var_f)

    return S1
```

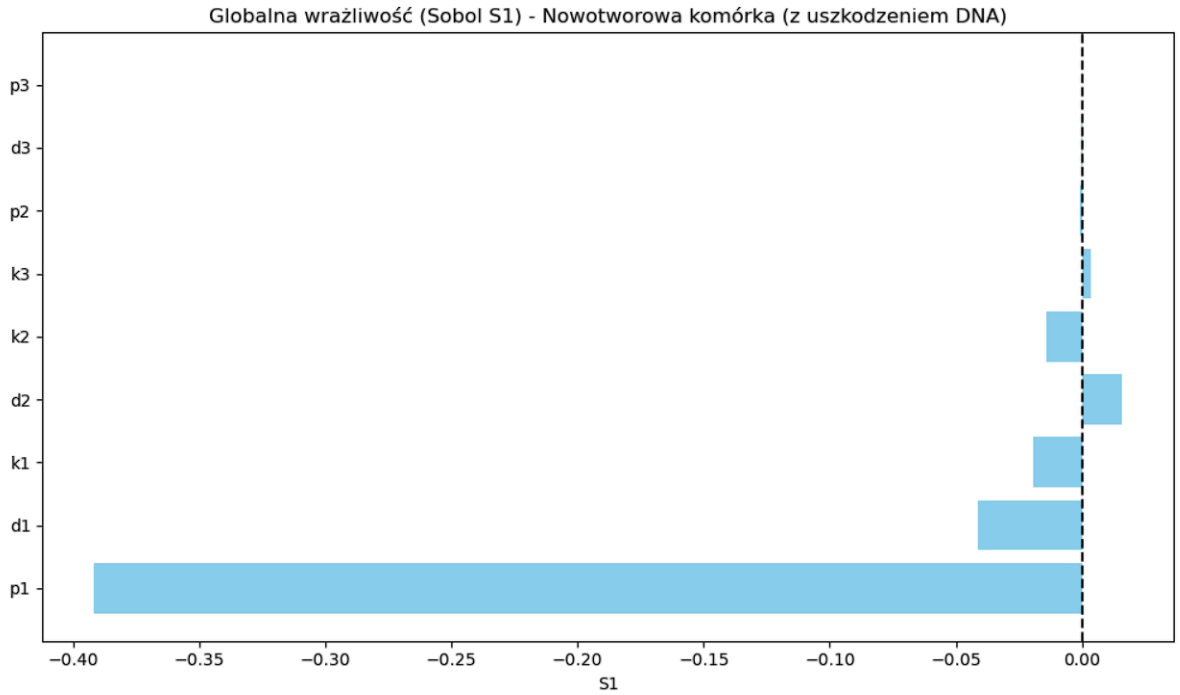
```
#uruchomienie main

def main():
   param_names = list(params_nominal.keys())
   bounds = [[params_nominal[k]*0.8, params_nominal[k]*1.2] for k
in param_names]

   for label, scenario in conditions.items():
       print(f"Analiza scenariusza: {label}")

       #globalna analiza wrażliwości
       S1 = global_sensitivity(bounds, param_names,
initial_conditions, scenario, N=128)
       ranked_global = sorted(zip(param_names, S1), key=lambda x:
abs(x[1]), reverse=True)

       plt.figure(figsize=(10, 6))
       plt.barh([k for k, _ in ranked_global], [v for _, v in
ranked_global], color='skyblue')
       plt.axvline(0, color='black', linestyle='--')
       plt.title(f"Globalna wrażliwość (Sobol S1) - {label}")
       plt.xlabel("S1")
       plt.tight_layout()
       plt.show()

       #lokalna analiza wrażliwości
       t, sensitivities, p53_base =
local_sensitivity(params_nominal, initial_conditions, scenario)
       ranking_mean = sorted({k: np.mean(np.abs(v)) for k, v in
sensitivities.items()}.items(), key=lambda x: x[1], reverse=True)
       ranking_end = sorted({k: abs(v[-1]) for k, v in
sensitivities.items()}.items(), key=lambda x: x[1], reverse=True)

       plt.figure(figsize=(10, 6))
       for k, v in sensitivities.items():
           plt.plot(t, v, label=k)
       plt.legend()
       plt.title(f"Lokalna wrażliwość p53 - {label}")
       plt.xlabel("Czas [min]")
       plt.ylabel("Wrażliwość")
       plt.tight_layout()
       plt.show()

       #zmiana poziomu p53
       change_params = [ranking_mean[0][0], ranking_mean[-1][0]]
       fig, axes = plt.subplots(1, 2, figsize=(14, 5))
       for i, p in enumerate(change_params):
```

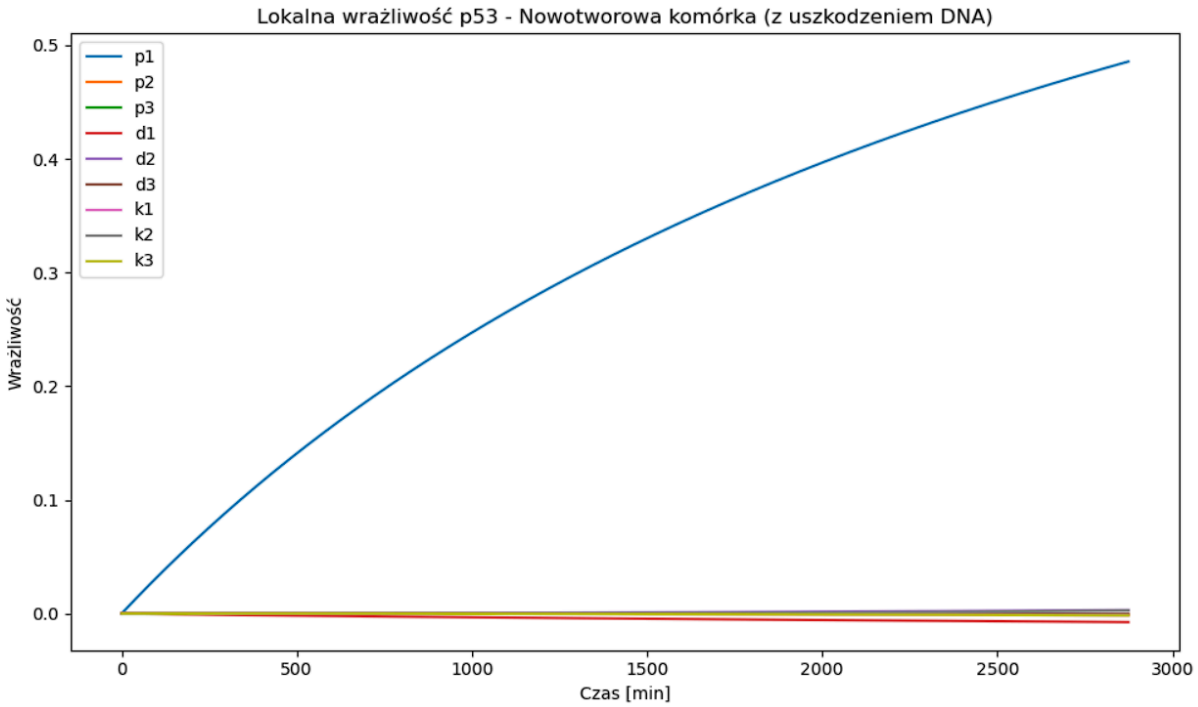
```
            axes[i].plot(t, p53_base, label='Nominalny')
            for factor in [1.2, 0.8]:
                perturbed = params_nominal.copy()
                perturbed[p] *= factor
                pert_result = RK4_run(perturbed,
initial_conditions, scenario, iterations, h)
                axes[i].plot(pert_result['t'], pert_result['p53'],
label=f'{p} * {factor}')
            axes[i].set_title(f'Zmiana parametru: {p}')
            axes[i].set_xlabel("Czas [min]")
            axes[i].set_ylabel("p53")
            axes[i].legend()
            axes[i].grid(True)
        plt.suptitle(f"Wpływ zmiany parametrów na poziom p53 -
{label}")
        plt.tight_layout()
        plt.show()

if __name__ == '__main__':
    main()
```

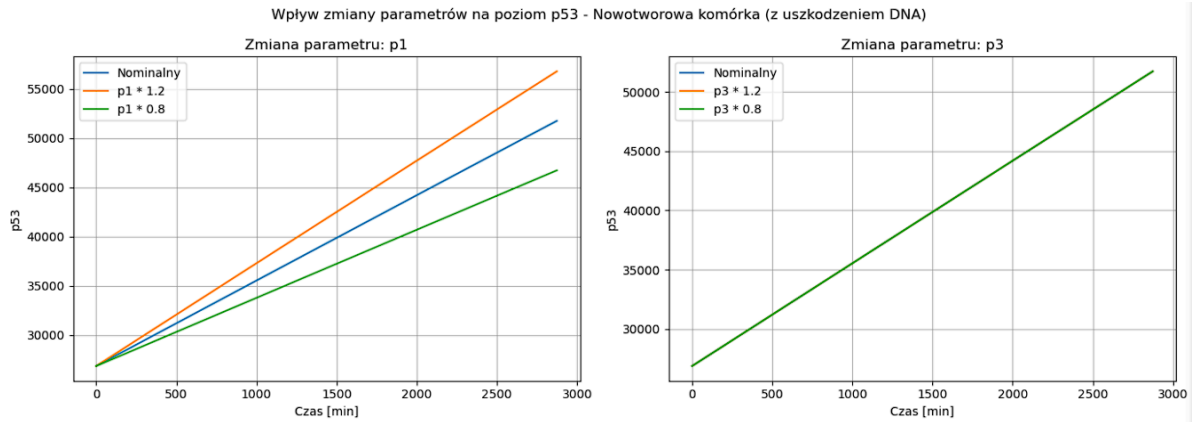



Lokalna wrażliwość p53 - Zdrowa komórka (bez uszkodzenia DNA)
p1
p2
p3
d1
d2
d3
k1
k2
k3
Wrażliwość
Czas [min]


Wpływ zmiany parametrów na poziom p53 - Zdrowa komórka (bez uszkodzenia DNA)
Zmiana parametru: p1
Nominalny
p1 * 1.2
p1 * 0.8
p53
Czas [min]
Zmiana parametru: p3
Nominalny
p3 * 1.2
p3 * 0.8
p53
Czas [min]


Globalna wrażliwość (Sobol S1) - Nowotworowa komórka (z uszkodzeniem DNA)
p3
d3
p2
k3
k2
d2
k1
d1
p1
S1


Lokalna wrażliwość p53 - Nowotworowa komórka (z uszkodzeniem DNA)
p1
p2
p3
d1
d2
d3
k1
k2
k3
Wrażliwość
Czas [min]


Wpływ zmiany parametrów na poziom p53 - Nowotworowa komórka (z uszkodzeniem DNA)
Zmiana parametru: p1
Nominalny
p1 * 1.2
p1 * 0.8
p53
Czas [min]
Zmiana parametru: p3
Nominalny
p3 * 1.2
p3 * 0.8
p53
Czas [min]