Optymalizacja	Informatyka Technicza	Grupa 5
Imię i nazwisko: Jakub Świerczyński, Jakub Wawrzyczek, Oliwia Wiatrowska, Karolina Starzec	Temat: LAB 6 - Optymalizacja metodami niedeterministycznymi	

1. Cel

Celem ćwiczenia jest zapoznanie się z niedeterministycznymi metodami optymalizacji poprzez ich implementację oraz wykorzystanie do wyznaczenia minimum podanej funkcji celu.

2. Kod źródłowy

Funkcja testowa *test_fun(x):*

```
def test_fun(x):
    global FCALLS
    FCALLS += 1
    return x[0] ** 2 + x[1] ** 2 - np.cos(2.5 * np.pi * x[0]) - np.cos(2.5
* np.pi * x[1]) + 2
```

Parametry:

```
# Parametry
N = 2  # liczba zmiennych
limits = np.array([[-5, -5], [5, 5]])  # limity dla każdej zmiennej jako
lista dwuelementowych zakresów
epsilon = 0.001
Nmax = 1000

best_solution, best_fitness = EA(N, limits, epsilon,test_fun ,Nmax)
print(f"Best solution found: {best_solution}, with fitness:
{best_fitness}")
```

Algorytm EA:

```
ef EA(N, limits, epsilon, fun: Callable, Nmax):
    mi, lambda_ = 20, 40
    P_x = np.random.uniform(limits[0], limits[1], (mi + lambda_, N))
    P_y = np.array([fun(x) for x in P_x[:mi]])
    gen = np.random.default_rng(seed=int(datetime.now().timestamp()))

    for _ in range(Nmax):
        if np.min(P_y) < epsilon:
            return P_x[np.argmin(P_y)], np.min(P_y)

        IFF = 1 / (P_y + 1e-10)
        parents_indices = np.searchsorted(np.cumsum(IFF) / np.sum(IFF),
gen.random(lambda_))
        P_x[mi:] = P_x[parents_indices] + gen.normal(0, 1, (lambda_, N))

        for i in range(0, lambda_, 2):
            cp = gen.integers(1, N)
```

```
            P_x[mi + i, cp:], P_x[mi + i + 1, cp:] = P_x[mi + i + 1,
cp:].copy(), P_x[mi + i, cp:].copy()

        new_fitness = np.array([fun(x) for x in P_x[mi:]])
        if np.min(new_fitness) < epsilon:
            return P_x[mi + np.argmin(new_fitness)], np.min(new_fitness)

        combined_fitness = np.hstack((P_y, new_fitness))
        best_indices = np.argsort(combined_fitness)[:mi]
        P_x[:mi], P_y = np.vstack((P_x, P_x[mi:]))[best_indices],
combined_fitness[best_indices]

    return P_x[np.argmin(P_y)], np.min(P_y)
```

Symulacja:

```
def symulacja(B):
    t_end, dt = 100, 0.1
    m1, m2, k1, k2, F = 5, 5, 1, 1, 1
    x1_initial, x2_initial, v1_initial, v2_initial = 0, 0, 0, 0

    def equations(t, y, b1, b2):
        x1, v1, x2, v2 = y
        return [
            v1,
            (-b1 * v1 - b2 * (v1 - v2) - k1 * x1 - k2 * (x1 - x2)) / m1,
            v2,
            (F + b2 * (v1 - v2) + k2 * (x1 - x2)) / m2
        ]

    sol = solve_ivp(equations, [0, t_end], [x1_initial, v1_initial,
x2_initial, v2_initial],
                    t_eval=np.arange(0, t_end + dt, dt), args=(B[0], B[1]))
    return pd.DataFrame({'x1': sol.y[0], 'x2': sol.y[2]}, index=sol.t)
```

3. Omówienie wyników

Tabela przedstawia wyniki eksperymentu dotyczącego wpływu początkowego zakresu mutacji na optymalizację funkcji celu za pomocą algorytmu ewolucyjnego:

Początkowa wartość zakresu mutacji	x_1^*	x_2^*	y^*	Liczba wywołań funkcji celu	Liczba minimów globalnych
0,01	0,000018	0,000321	0,001583	34159	1
0,1	-0,000383	0,000685	0,001564	30125,8	1
1	-0,000276	-0,000901	0,001537	30544,6	1
10	-0,000214	0,000935	0,001567	28162,2	1
100	-0,000314	0,000077	0,00138	28989,4	1

Według danych optymalnym kompromisem między dokładnością a liczbą wywołań funkcji celu wydaje się zakres mutacji 10, który zapewnia efektywną konwergencję przy stosunkowo niewielkiej liczbie obliczeń.

Kolejna tabela przedstawia wyniki optymalizacji parametrów b1 i b2 w celu minimalizacji funkcji celu:

b_1*	b_2*	y*	Liczba wywołań funkcji celu
1,50521412	2,50883599	1,93E-06	3340

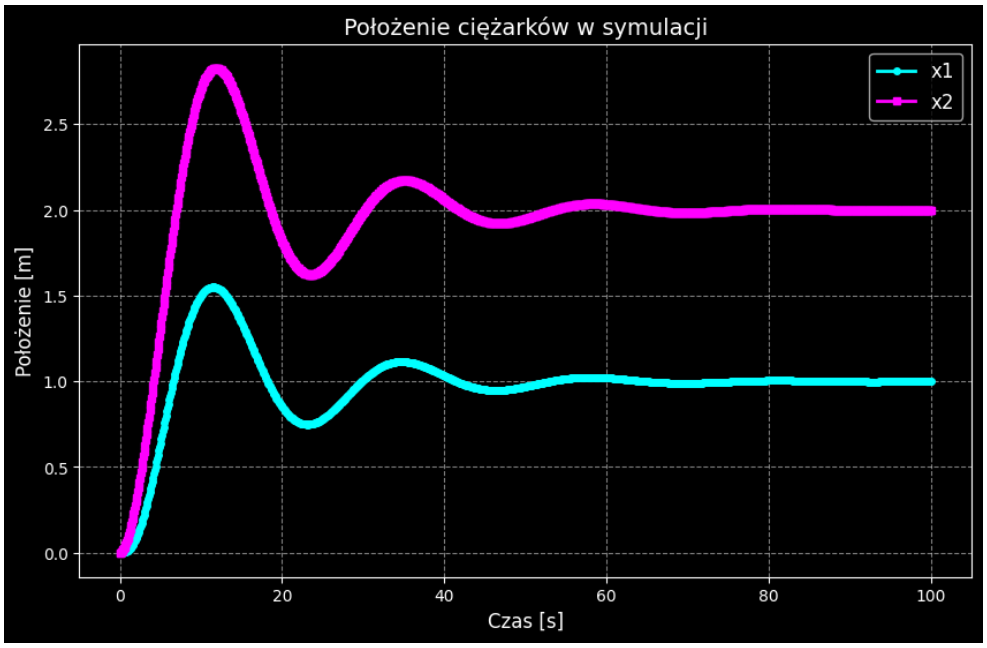
Znalezione wartości b1i b2 pozwoliły na osiągnięcie bardzo niskiej wartości funkcji celu, co sugeruje wysoką skuteczność procesu optymalizacji.

Niewielka liczba wywołań funkcji celu (3340) wskazuje na efektywność algorytmu, co oznacza, że rozwiązanie zostało znalezione relatywnie szybko.

Minimalna wartość funkcji celu w skali rzędu 10^{-6} wskazuje na bardzo precyzyjne dopasowanie modelu do optymalnych warunków.

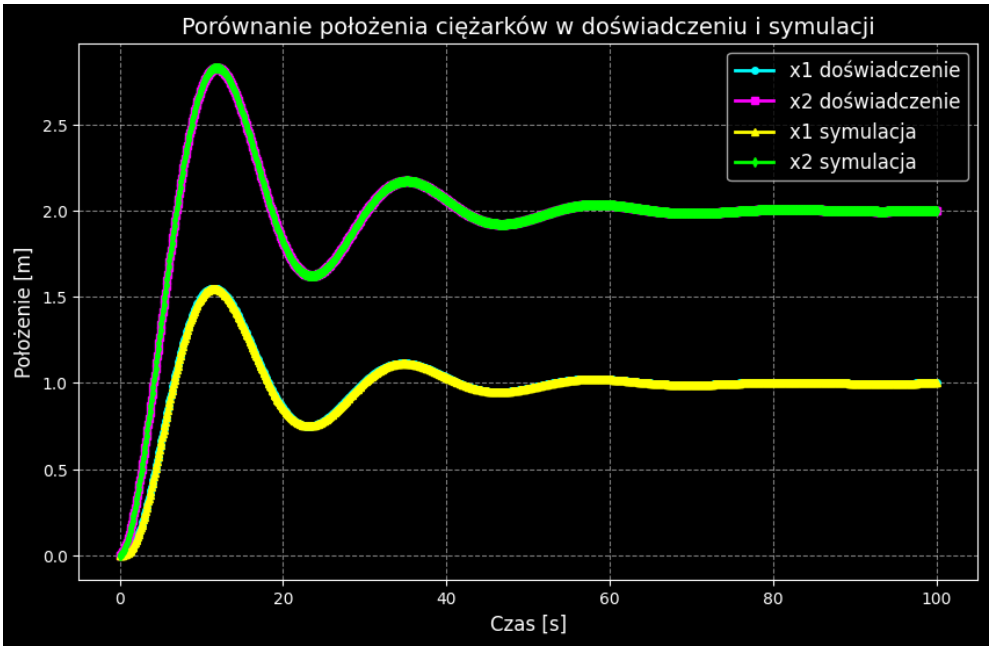
4. Wykresy

Pierwszy wykres przedstawia zmiany położenia dwóch ciężarków w czasie.



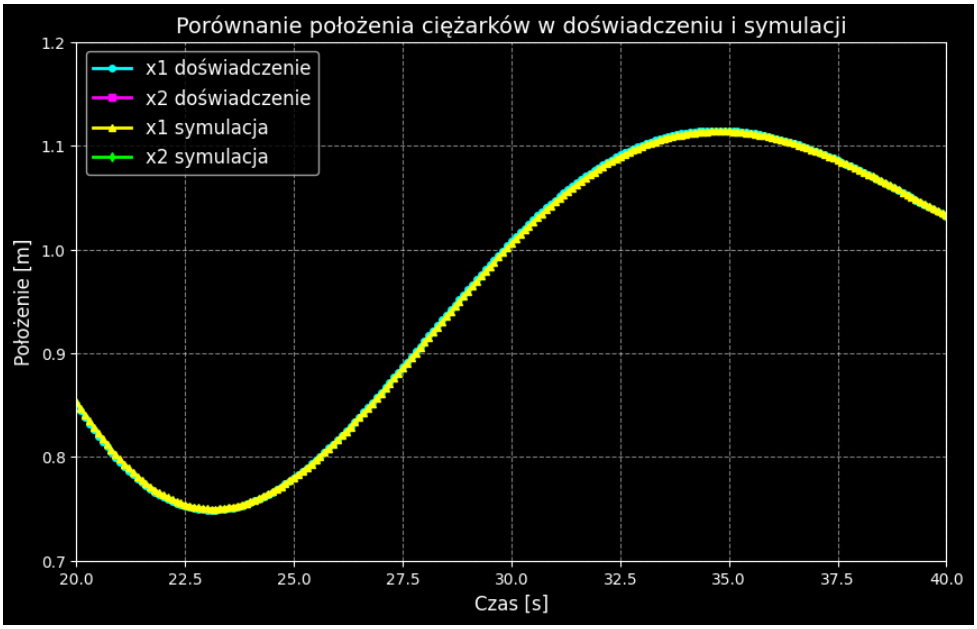
Obserwujemy stopniowe tłumienie oscylacji, co sugeruje obecność sił tłumiących w układzie. Ciężarek x2 wykazuje większe wychylenia niż x1, jednak jego drgania również ulegają stopniowemu wygaszaniu, co może wynikać z różnic w parametrach tłumienia lub sił działających na poszczególne elementy systemu. Po około 60 sekundach oba ciężarki osiągają stan równowagi, co wskazuje na stabilność układu oraz skuteczność mechanizmów tłumiących.

Wykres 2 przedstawia porównanie położenia ciężarków w doświadczeniu i symulacji w funkcji czasu:



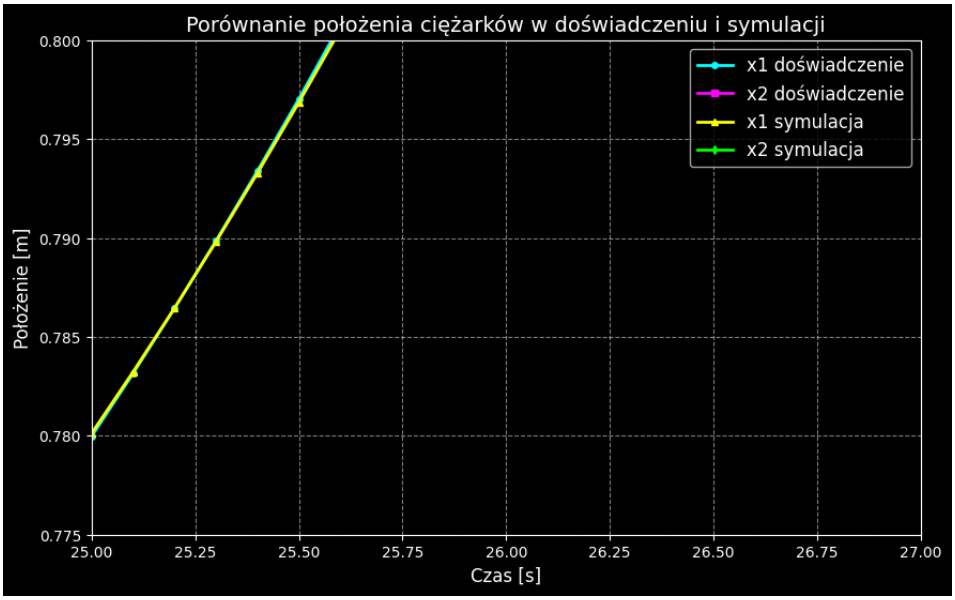
Wyniki symulacji bardzo dobrze pokrywają się z danymi eksperymentalnymi, co wskazuje na poprawność modelu matematycznego i jego potencjalne zastosowanie w dalszych analizach dynamicznych układu.

Wykres 3 przedstawia porównanie położenia ciężarków w doświadczeniu i symulacji w przedziale czasowym od 20 do 40 sekund.



Wykres wskazuje na bardzo dobrą zgodność symulacji z rzeczywistym eksperymentem, co potwierdza poprawność modelu i jego potencjalne zastosowanie do dalszych analiz i optymalizacji układu dynamicznego.

Ostatni wykres przedstawia porównanie położenia ciężarków w doświadczeniu i symulacji w zawężonym przedziale czasowym od 25 do 27 sekund.



Wykres potwierdza wysoką zgodność między symulacją a doświadczeniem, co oznacza, że model jest dobrze dopasowany do rzeczywistych warunków i może być używany do dalszych analiz dynamiki układu.

5. Wnioski

Podsumowując wyniki przeprowadzonych badań, można stwierdzić, że zastosowane metody niedeterministycznej optymalizacji, a w szczególności algorytm ewolucyjny, skutecznie znalazły optymalne wartości parametrów funkcji celu. Analiza wyników wskazuje, że optymalnym kompromisem między dokładnością a liczbą wywołań funkcji celu jest zakres mutacji wynoszący 10, który zapewnia efektywną konwergencję przy stosunkowo niewielkiej liczbie obliczeń. Dodatkowo, optymalizacja parametrów b1 i b2 pozwoliła na osiągnięcie bardzo niskiej wartości funkcji celu, co świadczy o wysokiej skuteczności procesu oraz efektywności algorytmu, który osiągnął wynik w zaledwie 3340 wywołaniach funkcji celu.

Wykresy obrazujące zachowanie ciężarków w czasie potwierdzają poprawność modelu matematycznego. Analiza zmian położenia wskazuje na stopniowe tłumienie oscylacji, co sugeruje obecność sił tłumiących w układzie. Ciężarek x2 wykazuje większe wychylenia niż x1, jednak jego drgania również ulegają stopniowemu wygaszaniu. Po około 60 sekundach oba ciężarki osiągają stan równowagi, co potwierdza stabilność układu oraz skuteczność mechanizmów tłumiących.

Porównanie wyników eksperymentalnych z symulacjami przeprowadzonymi w różnych przedziałach czasowych wykazuje bardzo dobrą zgodność. Wyniki symulacji pokrywają się z danymi eksperymentalnymi, co oznacza, że model dobrze odwzorowuje rzeczywiste zachowanie układu. Zarówno w pełnym zakresie czasowym, jak i w

wybranych fragmentach, takich jak przedział 20-40 sekund oraz zawężony zakres 25-27 sekund, zgodność wyników wskazuje na wysoką precyzję modelu i jego potencjalne zastosowanie do dalszych analiz i optymalizacji układu dynamicznego.

Podsumowując, przeprowadzone badania wykazały skuteczność zastosowanego algorytmu ewolucyjnego oraz potwierdziły poprawność modelu matematycznego, który dokładnie odwzorowuje rzeczywiste warunki eksperymentalne. Model może być używany do dalszych badań nad dynamiką układu, jego optymalizacją oraz przewidywaniem przyszłego zachowania systemu w różnych warunkach operacyjnych.