SYMULACJE W PYTHON

1. Zdefiniuj problem i model matematyczny

Rozpocznij od jasnego określenia problemu, który chcesz zasymulować. Następnie przekształć go w model matematyczny, na przykład poprzez sformułowanie równań różniczkowych opisujących dynamikę systemu.

2. Wybierz odpowiednie narzędzia i biblioteki

Python oferuje bogaty ekosystem bibliotek wspierających symulacje:

- **NumPy**: do efektywnych obliczeń numerycznych na tablicach wielowymiarowych.
- **SciPy**: zawiera funkcje do rozwiązywania równań różniczkowych, optymalizacji i innych zaawansowanych obliczeń.
- **Matplotlib** i **Seaborn**: do tworzenia wykresów i wizualizacji danych.
- **SymPy**: do obliczeń symbolicznych, takich jak różniczkowanie czy całkowanie.
- **Jupyter Notebook**: interaktywne środowisko do pisania i uruchamiania kodu oraz dokumentowania procesu symulacji.

3. Zaimplementuj model i przeprowadź symulację

Po przygotowaniu modelu matematycznego zaimplementuj go w Pythonie, korzystając z wybranych bibliotek. Na przykład, do rozwiązywania równań różniczkowych możesz użyć funkcji `odeint` z biblioteki SciPy.

4. Analizuj i wizualizuj wyniki

Po przeprowadzeniu symulacji, wykorzystaj biblioteki takie jak Matplotlib czy Seaborn do wizualizacji wyników. Dzięki temu łatwiej zrozumiesz zachowanie systemu i wyciągniesz wnioski.

5. Korzystaj z dostępnych zasobów edukacyjnych

Aby pogłębić swoją wiedzę na temat symulacji w Pythonie, warto skorzystać z dostępnych materiałów edukacyjnych:

- **Książki**: „Matematyczny Python. Obliczenia naukowe i analiza danych z użyciem NumPy, SciPy i Matplotlib” autorstwa Roberta Johanssona to doskonałe źródło wiedzy na temat obliczeń naukowych w Pythonie.

- **Kursy online**: Platformy edukacyjne oferują kursy dotyczące analizy danych i symulacji w Pythonie, które mogą być pomocne w nauce praktycznych umiejętności.
- **Materiały dydaktyczne**: Dostępne są również darmowe materiały i notatki, które omawiają różne aspekty analizy danych i symulacji w Pythonie.

PRZYKŁAD

1. Zdefiniuj parametry układu

Na początku określ parametry fizyczne systemu:

- `m1`, `m2`: masy obiektów
- `k1`, `k2`: sztywności sprężyn
- `b1`, `b2`: współczynniki tłumienia

Przykład:

```
m1 = 1.0      # masa 1
m2 = 1.0      # masa 2
k1 = 10.0     # sztywność sprężyny 1
k2 = 15.0     # sztywność sprężyny 2
b1 = 0.5      # tłumienie dla masy 1
b2 = 0.5      # tłumienie dla masy 2
```

2. Ustal warunki początkowe

Zdefiniuj początkowe pozycje i prędkości obu mas:

```
initial_state = np.array([1.0, 0.0, -1.0, 0.0])
# [x1, v1, x2, v2]
```

3. Zdefiniuj funkcję obliczającą pochodne

Stwórz funkcję `derivatives`, która oblicza pochodne stanu systemu w danym momencie czasu:

```
def derivatives(t, state):
    x1, v1, x2, v2 = state

    a1 = (-k1 * x1 - k2 * (x1 - x2) - b1 * v1) / m1
    a2 = (-k2 * (x2 - x1) - b2 * v2) / m2

    return np.array([v1, a1, v2, a2])
```

4. Zaimplementuj metodę Rungego-Kutty 4. rzędu

Stwórz funkcję `runge_kutta_4`, która wykorzystuje metodę RK4 do obliczenia kolejnego stanu systemu:

```
def runge_kutta_4(state, t, dt):
    k1 = derivatives(t, state)
    k2 = derivatives(t + dt / 2, state + dt / 2 * k1)
    k3 = derivatives(t + dt / 2, state + dt / 2 * k2)
    k4 = derivatives(t + dt, state + dt * k3)

    return state + (dt / 6) * (k1 + 2*k2 + 2*k3 + k4)
```

5. Skonfiguruj parametry symulacji

Określ czas trwania symulacji oraz krok czasowy:

```
t_max = 20.0
dt = 0.01
time = np.arange(0, t_max, dt)
```

6. Przeprowadź symulację

Zainicjalizuj tablicę do przechowywania stanów systemu i wykonaj pętlę symulacyjną:

```
states = np.zeros((len(time), 4))
states[0] = initial_state

for i in range(1, len(time)):
    states[i] = runge_kutta_4(states[i - 1], time[i - 1], dt)
```

7. Wizualizuj wyniki

Wykorzystaj bibliotekę Matplotlib do przedstawienia trajektorii ruchu obu mas:

```
plt.figure(figsize=(10, 6))
plt.plot(time, states[:, 0], label="Pozycja masy 1 (x1)")
plt.plot(time, states[:, 2], label="Pozycja masy 2 (x2)")
plt.xlabel("Czas [s]")
plt.ylabel("Pozycja [m]")
plt.title("Ruch dwóch mas połączonych sprężynami (Metoda
RK4)")
plt.legend()
plt.grid(True)
plt.show()
```