

Raport z projektu w ramach przedmiotu Modelowanie Matematyczne

Bartosz Chabros 198404

Beniamin Cymanowski 198067

31 maja 2025

1 Cel projektu

Celem projektu było stworzenie aplikacji umożliwiającej symulację odpowiedzi układu dynamicznego z regulatorem PI na różne typy sygnałów wejściowych. Użytkownik może zmieniać parametry transmitancji, regulatora oraz sygnału wejściowego z poziomu interfejsu graficznego.

2 Opis układu

Układ składa się z transmitancji obiektu $G_p(s)$ oraz regulatora PI $G_c(s)$. Całość została zamknięta w pętli sprzężenia zwrotnego.

2.1 Transmitancja obiektu

$$G_p(s) = \frac{a_1 s + a_0}{b_2 s^2 + b_1 s + b_0}$$

2.2 Regulator PI

$$G_c(s) = K_p + \frac{K_i}{s}$$

3 Interfejs użytkownika

Aplikacja została napisana w języku Python z użyciem bibliotek PyQt6 oraz Matplotlib. Umożliwia ona:

- Wprowadzanie parametrów transmitancji i regulatora,
- Wybór typu sygnału wejściowego (sinusoidalny, prostokątny, trójkątny),
- Ustawienie częstotliwości, fazy, czasu i kroku symulacji,
- Wizualizację sygnału wejściowego oraz odpowiedzi układu.
- Sprawdza stabilność układu i wyświetla o niej komunikaty w terminalu.

4 Opis działania programu

```
Gp_licznik = [a1, a0]
Gp_mianownik = [b2, b1, b0]
Gp = ct.tf(Gp_licznik, Gp_mianownik)
Gc_licznik = [Kp, Ki]
Gc_mianownik = [1, 0]

Gc = ct.tf(Gc_licznik, Gc_mianownik)
Go = ct.series(Gc, Gp)
Gz = ct.feedback(Go, 1)
```

Rysunek 1: Po przekazaniu do funkcji `run_simulation` są tworzone poszczególne transmitancje, są łączone szeregowo i zamykane w ujemnej pętli sprzężenia zwrotnego. W ten sposób otrzymujemy transmitancję całego układu.

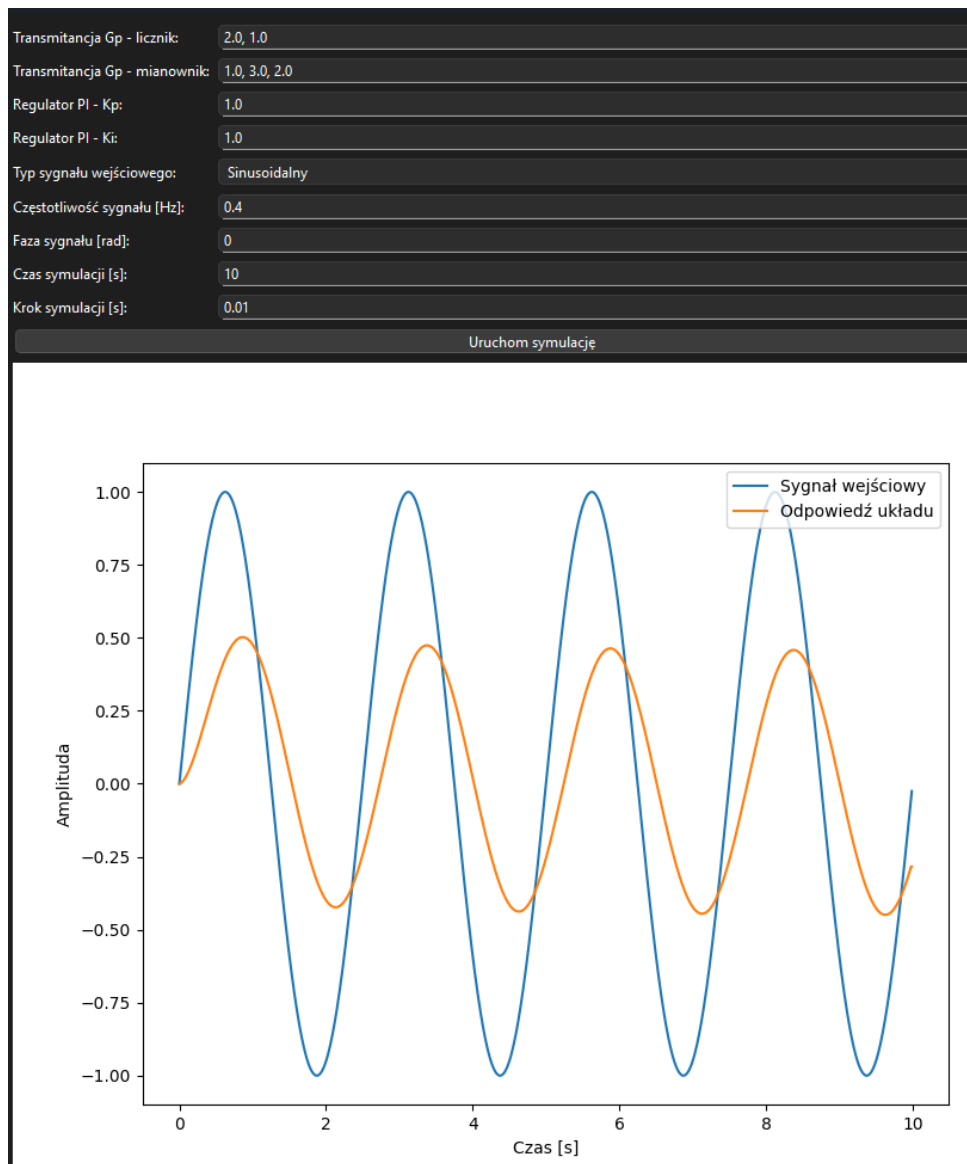
```
ss_model = ct.tf2ss(Gz)
A, B, C, D = ss_model.A, ss_model.B, ss_model.C, ss_model.D
```

Rysunek 2: Transmitancja całego układu jest następnie przekształcana na model stanowy.

```
for i in range(num_elements-1):
    #krok próbny
    Xp = Xp + A @ Xp + B * U[i]
    #krok właściwy
    X = Xp + 0.5 * (A @ X + B * U[i] + A @ Xp + B * U[i+1])
    Y = C @ X + D * U[i]
    output_signal.append(Y.item())

# Dodanie ostatniego punktu do output_signal
Y = C @ X + D * input_signal[-1]
output_signal.append(Y.item())
```

Rysunek 3: Rozwiązywanie równań stanu jest realizowane przy pomocy metody Tustina.



Rysunek 4: Przykładowa odpowiedź układu na sygnał sinusoidalny.

5 Przykładowe wyniki

6 Wnioski

Projekt pozwolił na praktyczne zastosowanie wiedzy z zakresu teorii sterowania oraz programowania. Aplikacja umożliwia szybkie testowanie wpływu parametrów na odpowiedź układu.

Symulacja Układu - Parametry

Transmitancja Gp - licznik:

2.0, 1.0

Transmitancja Gp - mianownik:

1.0, 3.0, 2.0

Regulator PI - Kp:

50.0

Regulator PI - Ki:

0.0

Typ sygnału wejściowego:

Prostokątny

Częstotliwość sygnału [Hz]:

0.5

Faza sygnału [rad]:

1.57

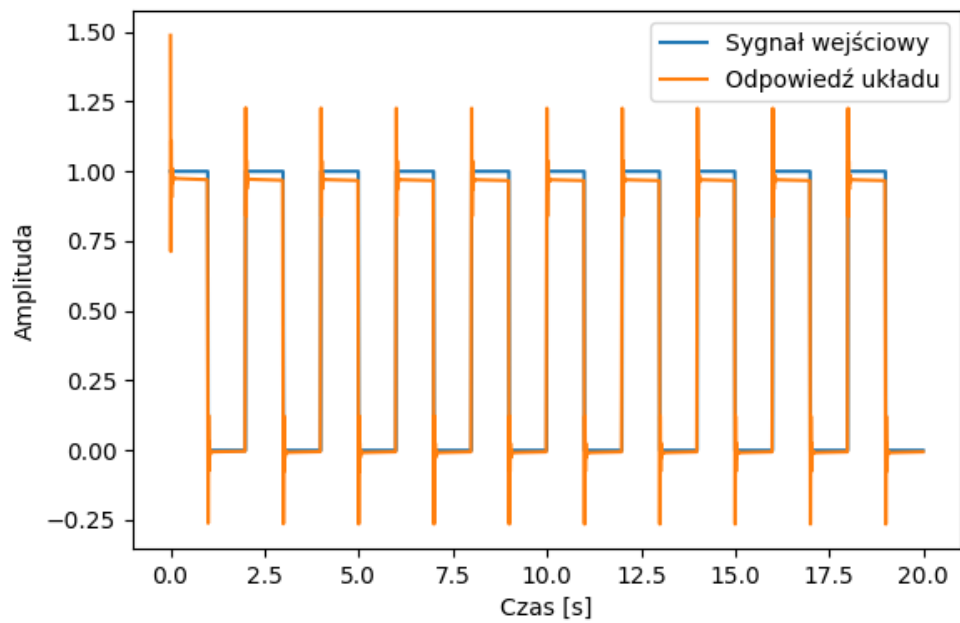
Czas symulacji [s]:

20

Krok symulacji [s]:

0.01

Uruchom symulację



Rysunek 5: Przykładowa odpowiedź układu na sygnał prostokątny. Ilustracja przeregulowania.

Symulacja Układu - Parametry

Transmitancja G_p - licznik:

2.0, 1.0

Transmitancja G_p - mianownik:

1.0, 3.0, 1.0

Regulator PI - K_p :

3.0

Regulator PI - K_i :

4.0

Typ sygnału wejściowego:

Trójkątny

Częstotliwość sygnału [Hz]:

0.5

Faza sygnału [rad]:

1.57

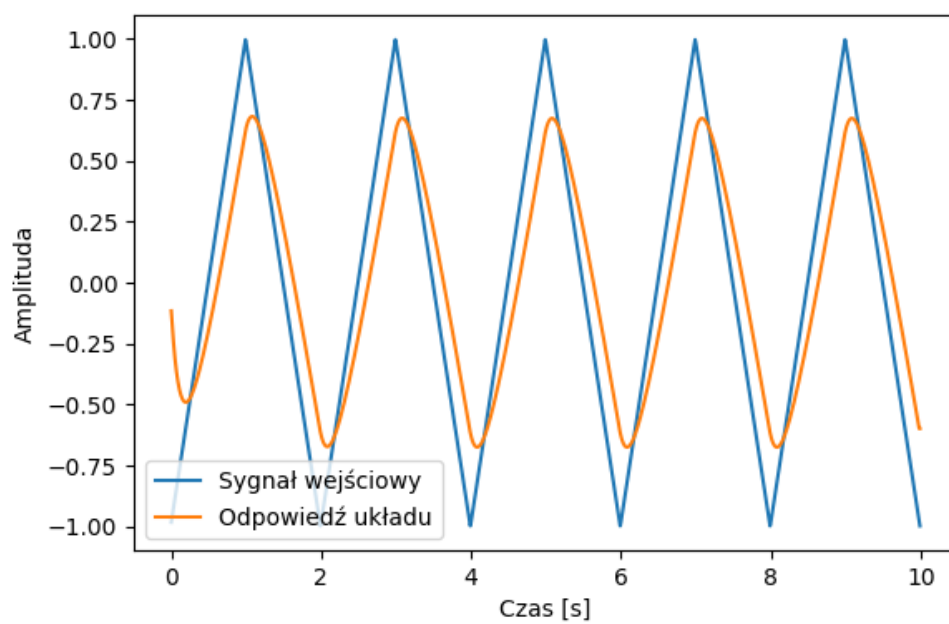
Czas symulacji [s]:

10

Krok symulacji [s]:

0.01

Uruchom symulację



Rysunek 6: Przykładowa odpowiedź układu na sygnał trójkątny.