Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Politechnika Świętokrzyska
Wydział Elektrotechniki, Automatyki i Informatyki

Kierunek studiów:
Elektrotechnika

Paweł Strząbała

Materiały dydaktyczne do przedmiotu

Komputerowe wspomaganie projektowania w układach sterowania

opracowane w ramach realizacji Projektu
„**Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki**”
FERS.01.05-IP.08-0234/23

Kielce, 2025

Spis treści

1. Wprowadzenie do Pakietu Control System Toolbox

Control System Toolbox to pakiet narzędzi dostępny w środowisku MATLAB, służący do analizy, modelowania i projektowania układów dynamicznych. Umożliwia m.in. wyznaczanie odpowiedzi czasowych systemów na różne wymuszenia oraz wykreślanie charakterystyk częstotliwościowych. Pakiet ten pozwala na pracę zarówno z układami SISO (Single Input Single Output), jak i MIMO (Multiple Input Multiple Output), a także udostępnia funkcje wspomagające projektowanie regulatorów i układów sterowania.

1.1. Definiowanie modelu systemu

W MATLAB-ie systemy dynamiczne można definiować za pomocą transmitancji (ang. transfer function), w postaci równań stanu (ang. state-space) oraz w formie zer i biegunów (ang. zero-pole-gain).

tf - **transfer function**, funkcja służy do definiowania systemu w postaci transmitancji:

$$\mathrm{G(s)} = \frac{\mathrm{NUM(s)}}{\mathrm{DEN(s)}} \tag{1}$$

gdzie: NUM i DEN to odpowiednio macierze współczynników licznika oraz mianownika transmitancji.

Przykład 1 - definiowanie obiektu w postaci transmitancji

Transmitancja obiektu:

$$G(s) = \frac{s}{s^2 + 2s + 10} \tag{2}$$

Program w MATLAB-ie:

Metoda 1 - zapis symboliczny z użyciem tf('s'):

```
clc; clear;

s = tf('s');                    % operator Laplace'a
SYS = s / (s^2 + 2*s + 10)      % definicja transmitancji
```

Metoda 2 - zapis za pomocą współczynników licznika i mianownika:

Politechnika Świętokrzyska Kielce University of Technology
Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki" nr FERS.01.05-IP.08-0234/23

```
NUM = [1 0];                        % licznik: s
DEN = [1 2 10];                     % mianownik: s^2 + 2*s + 10
SYS2 = tf(NUM, DEN)                 % definicja transmitancji
```

ss - **state space**, funkcja służy do definiowania systemu w postaci zmiennych stanu:

$$\dot{x} = Ax(t) + Bu(t) \tag{3}$$

$$y = Cx(t) + Du(t) \tag{4}$$

gdzie A,B,C oraz D to macierze równań stanu oraz wyjścia.

Składnia w MATLAB-ie:

$$SYS = ss(A, B, C, D) \tag{5}$$

Przykład 2

Dany jest obiekt opisany równaniami:

$$\dot{x} = \begin{bmatrix} -2 & -1 \\ 1 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t) \tag{6}$$

$$y = [1 \quad 2]x(t) + [1]u(t) \tag{7}$$

Program w MATLAB-ie:

```
clc; clear;

A = [-2 -1;1 0];    % macierz stanu
B = [1;0];          % macierz sterowania
C = [1, 2];         % macierz wyjścia
D = 1;              % macierz sprzężenia bezpośredniego

SYS = ss(A, B, C, D)   % definicja modelu w przestrzeni stanu
```

Ta reprezentacja jest szczególnie przydatna w przypadku systemów wielowymiarowych (MIMO).

zpk - funkcja służy do definiowania systemu w postaci transmitancji danej w postaci zer i biegunów oraz wzmocnienia:

Politechnika Świętokrzyska Kielce University of Technology
Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki" nr FERS.01.05-IP.08-0234/23

$$G(s) = K \cdot \frac{(s - z_1) \dots (s - z_m)}{(s - p_1) \dots (s - p_n)} \tag{8}$$

Składnia w MATLAB-ie:

$$SYS = zpk(Z, P, K) \tag{9}$$

gdzie:

- Z = [z_1, z_2, z_3, … ,z_m] - wektor zer (pierwiastki licznika),
- P = [p_1, p_2, p_3, … ,p_n] - pierwiastki mianownika (p_1 − p_n),
- K - wzmocnienie.

Przykład 3 - definiowanie systemu w postaci zer, biegunów oraz wzmocnień:

Obiekt opisany jest transmitancją:

$$G(s) = \frac{2s + 1}{(s + 2)(s + 5)} = \frac{2 \cdot (s + 0{,}5)}{(s + 2)(s + 5)} \tag{10}$$

Kod w MATLAB-ie:

```
clc; clear;

Z = [-0.5];          % zera
P = [-2 -5];         % bieguny
K = 2;               % wzmocnienie

SYS = zpk(Z, P, K)   % definicja systemu w postaci ZPK
```

1.2. Zmiana postaci modelu

Każda z trzech reprezentacji (**tf**, **ss**, **zpk**) może opisywać ten sam system, ale jest użyteczna w innych kontekstach. W praktyce często stosuje się konwersje między nimi w zależności od analizowanego problemu.

- **tf2ss** - konwersja z transmitancji do równań stanu,
- **ss2tf** - konwersja z równań stanu do transmitancji.

Przykład 4

```
clc; clear;

NUM = [1 0];
DEN = [1 2 10];
```

```
SYS1 = tf(NUM, DEN)              % transmitancja: s / (s^2 + 2s + 10)

[A,B,C,D] = tf2ss(NUM, DEN);     % tf → ss
SYS2 = ss(A,B,C,D);

[NUM2, DEN2] = ss2tf(A,B,C,D);   % ss → tf
SYS3 = tf(NUM2, DEN2)
```

- **zp2ss** - wyznaczanie macierzy równań stanu na podstawie zer, biegunów i wzmocnienia,
- **ss2zp** - wyznaczanie zer, biegunów i wzmocnienia na podstawie równań stanu

Przykład 5

```
clc; clear;

Z = -0.5;
P = [-2 -5];
K = 2;

SYS1 = zpk(Z,P,K)                % postać ZPK

[A,B,C,D] = zp2ss(Z,P,K);        % zpk → ss
SYS2 = ss(A,B,C,D);

[Z2,P2,K2] = ss2zp(A,B,C,D);     % ss → zpk
SYS3 = zpk(Z2,P2,K2)
```

- **tf2zp** - *transfer function to zero pole*, funkcja służy do wyznaczenia zer, biegunów i wzmocnień na podstawie transmitancji,
- **zp2tf** - *zero pole to transfer function*, funkcja służy do wyznaczania transmitancji obiektu na podstawie zer, biegunów oraz wzmocnień.

Przykład 6

```
clc; clear;

NUM = [1 0];
DEN = [1 2 10];
SYS1 = tf(NUM, DEN)              % transmitancja

[Z,P,K] = tf2zp(NUM, DEN);       % tf → zpk
SYS2 = zpk(Z,P,K);
```


Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki" nr FERS.01.05-IP.08-0234/23

```
[NUM2,DEN2] = zp2tf(Z,P,K);      % zpk → tf
SYS3 = tf(NUM2, DEN2)
```

Funkcje tf2ss, ss2tf, zp2ss, ss2zp, tf2zp, zp2tf umożliwiają łatwe przechodzenie między różnymi postaciami modelu obiektu. W MATLAB-ie reprezentacje te są równoważne i można je stosować zamiennie w zależności od rodzaju analizy.

1.3. Łączenie modeli obiektów

Modele dynamiczne można łączyć w bardziej złożone układy, korzystając z operatorów (*, +) lub funkcji (series, parallel, feedback).

Szeregowe połączenie obiektów:

$$SYS = SYS_1 \cdot SYS_2 \tag{11}$$

Funkcja w MATLAB-ie:

SYS = series(SYS1,SYS2,'name')

Równoległe połączenie obiektów:

$$SYS = SYS_1 + SYS_2 \tag{12}$$

Funkcja w MATLAB-ie:

SYS = parallel(SYS1,SYS2,'name')

Układ ze sprzężeniem zwrotnym można zdefiniować jako:

$$SYS = \frac{SYS_1}{1 \pm SYS_1 \cdot SYS_2} \tag{13}$$

Funkcja w MATLAB-ie:

SYS = feedback(SYS1, SYS2, SIGN)

Znak SIGN = -1 dla sprzężenia ujemnego (domyślnie), SIGN = +1 dla sprzężenia dodatniego.

Przykład 7

```
clc; clear;

s = tf('s');                    % zmienna Laplace'a

% Dwa modele obiektów
```

```
SYS1 = 1/(0.2*s+1);
SYS2 = 1/(0.4*s+1);

% Połączenia systemów
SYS3 = series(SYS1, SYS2)       % połączenie szeregowe
SYS4 = parallel(SYS1, SYS2)     % połączenie równoległe
SYS5 = feedback(SYS1, SYS2)     % ujemne sprzężenie zwrotne
SYS6 = feedback(SYS2, 1, +1)    % dodatnie sprzężenie jednostkowe
```

Dzięki powyższym funkcjom można budować złożone schematy blokowe bez konieczności rysowania ich w Simulinku.

1.4. Badanie parametrów modelu obiektu

Do analizy modeli systemów dynamicznych MATLAB udostępnia zestaw funkcji umożliwiających badanie m.in.: zer, biegunów, zapasu stabilności oraz charakterystyk częstotliwościowych.

Wybrane funkcje:

- **zero** - służy do wyznaczania zer transmitancji oraz wzmocnienia układu:

  [ZERO, GAIN] = zero(SYS)

- **pzmap** - służy do graficznego przedstawienia rozmieszczenia zer i biegunów systemu w płaszczyźnie zespolonej:

  pzmap(SYS1,SYS2, ...)

- **margin** - służy do wyznaczania zapasu modułu Gm [dB], zapasu fazy Pm [°] (w stopniach) oraz związanych z nimi częstotliwości krytycznych amplitudy Wcg i fazy Wcp.

  [Gm, Pm, Wcg, Wcp] = margin(SYS)

- **allmargin** - funkcja służy do wyznaczania informacji o stabilności układu:

  S = allmargin(SYS)

  gdzie: S jest strukturą, która zawiera pola: GainMargin (zapas amplitudy), PhaseMargin (zapas fazy), GMFrequency (częstotliwość dla zapasu amplitudy), PMFrequency (częstotliwość dla zapasu fazy), Stable (informacja o stabilności).

Przykład 8

```
clc; clear;
```


Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki" nr FERS.01.05-IP.08-0234/23

```
s = tf('s');              % operator Laplace'a
SYS = s/(s^2 + 3*s + 2)   % transmitancja: G(s) = s / (s^2 + 3s + 2)

[ZERO, GAIN] = zero(SYS)            % zera układu i wzmocnienie
[Gm, Pm, Wcg, Wcp] = margin(SYS)    % zapas amplitudy, fazy i
częstotliwości krytyczne

S = allmargin(SYS);         % szczegółowe informacje o stabilności
disp(S)                     % wyświetlenie struktury 'S'

pzmap(SYS);                 % wykres rozmieszczenia zer i biegunów
```

1.5. Wyznaczanie charakterystyk częstotliwościowych układu

W analizie układów regulacji istotne znaczenie mają charakterystyki częstotliwościowe. W MATLAB-ie dostępne są dedykowane funkcje do ich wyznaczania i wizualizacji:

- **bode** - służy do wykreślania charakterystyk logarytmicznych amplitudowo-fazowych (Bodego):

  bode(SYS1, SYS2, ... , WMIN, WMAX)

  gdzie: SYS1, SYS2, ... - systemy, które mają być porównane, WMIN, WMAX - opcjonalnie zakres częstotliwości.

- **nyquist** - służy do wykreślania charakterystyki amplitudowo - fazowej Nyquista:

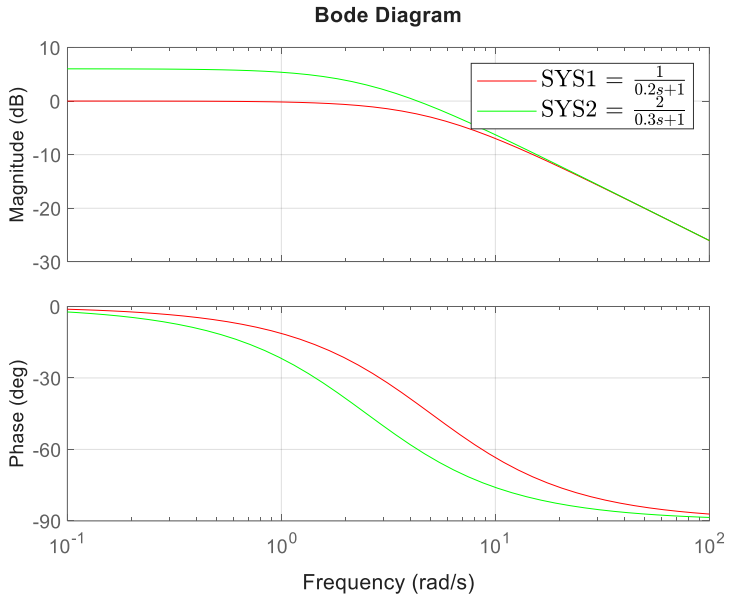
  nyquist(SYS1, SYS2, ..., WMIN, WMAX)

Przykład 9 - program wyznacza charakterystyki częstotliwościowe dla dwóch obiektów inercyjnych I rzędu:

```
clc; clear; close all;

s = tf('s');
SYS1 = 1/(0.2*s + 1);
SYS2 = 2/(0.4*s + 1);

% --- Charakterystyki Bodego ---
figure(1);
bode(SYS1, 'r', SYS2, 'g');
grid on; hold on;
leg = legend('SYS1 = $\frac{1}{0.2s+1}$', ...
```

```
             'SYS2 = $\frac{2}{0.4s+1}$');
leg.Interpreter = 'latex';
leg.FontSize = 12;

% --- Charakterystyka Nyquista ---
figure(2);
nyquist(SYS1, 'r', SYS2, 'g');
grid on; hold on;
leg = legend('SYS1 = $\frac{1}{0.2s+1}$', ...
             'SYS2 = $\frac{2}{0.4s+1}$');
leg.Interpreter = 'latex';
leg.FontSize = 12;
```

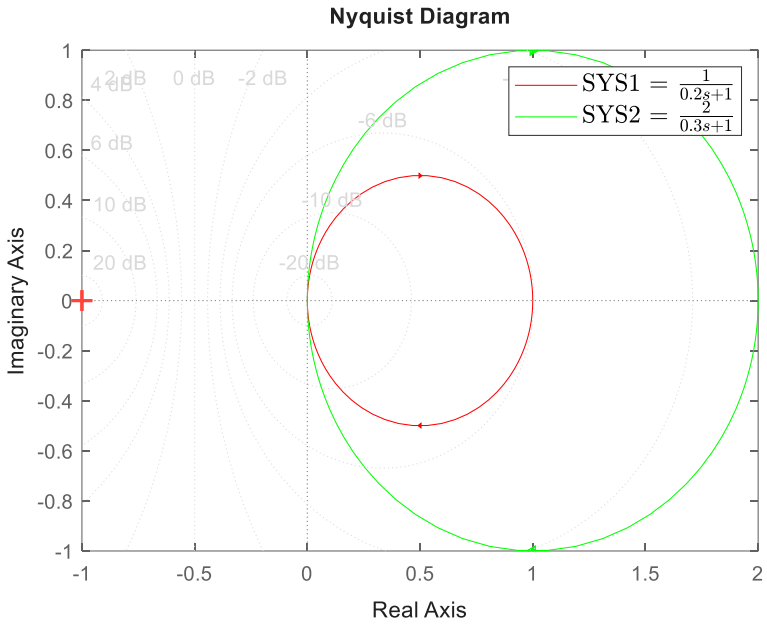
[Tekst alternatywny. Wykres dwuwymiarowy. Oś X przedstawia częstotliwość (w skali logarytmicznej), oś Y przedstawia odpowiednio: amplitudę w decybelach oraz fazę w stopniach. Na wykresie znajdują się charakterystyki Bode'go członów inercyjnych I rzędu - krzywe obrazujące spadek amplitudy oraz przesunięcie fazowe w funkcji częstotliwości.]



Rys. 1. Charakterystyki Bode'go członów inercyjnych I rzędu

[Tekst alternatywny. Wykres dwuwymiarowy. Oś X i oś Y przedstawiają część rzeczywistą i urojoną transmitancji zespolonej. Na wykresie znajdują się charakterystyki Nyquista członów inercyjnych I rzędu - krzywe pokazujące trajektorie wektora transmitancji w płaszczyźnie zespolonej w funkcji częstotliwości.]

Politechnika Świętokrzyska
Kielce University of Technology
Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki" nr FERS.01.05-IP.08-0234/23



Rys. 2. Charakterystyki Nyquist'a członów inercyjnych I rzędu

1.6. Wyznaczanie charakterystyk czasowych układu

Do analizy dynamiki systemów regulacji używa się odpowiedzi czasowych na skok jednostkowy oraz na impuls Diraca. MATLAB udostępnia do tego celu dedykowane funkcje.

- **step** - służy do wyznaczania odpowiedzi układu na wymuszenie skokowe jednostkowe:

  step(SYS1, SYS2, ..., TFINAL)

  step(SYS1, SYS2, ..., T)

  gdzie: TFINAL - czas końcowy symulacji (od 0 do TFINAL), T - wektor chwil czasowych, w których wyznaczana jest odpowiedź, SYS1, SYS2, ... - systemy do porównania na jednym wykresie.

- **impulse** - służy do wyznaczania odpowiedzi układu na wymuszenie impulsowe (Dirac'a):

  impulse(SYS1, SYS2, ..., TFINAL)

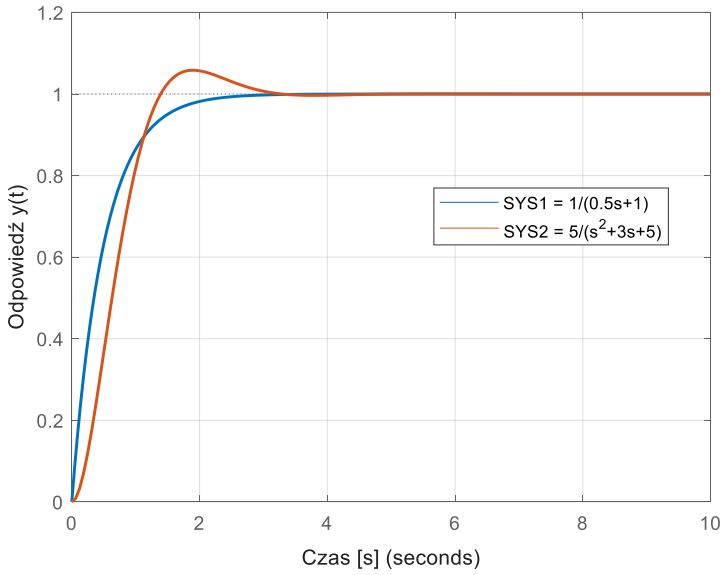
  impulse(SYS1, SYS2, ..., T)

Przykład 10 - program wyznacza odpowiedzi skokowe i impulsowe obiektów inercyjnych I rzędu:

Politechnika Świętokrzyska
Kielce University of Technology
Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki" nr FERS.01.05-IP.08-0234/23

```
clc; clear; close all;

s = tf('s');

SYS1 = 1/(0.5*s + 1);      % obiekt inercyjny I rzędu
SYS2 = 5/(s^2 + 3*s + 5);  % obiekt II rzędu

% --- Odpowiedzi skokowe ---
figure(1);
step(SYS1, SYS2, 10);   % odpowiedzi skokowe do T = 10s
grid on;
legend('SYS1 = 1/(0.5s+1)', 'SYS2 = 5/(s^2+3s+5)');

% --- Odpowiedzi impulsowe ---
figure(2);
impulse(SYS1, SYS2, 10); % odpowiedzi impulsowe do T = 10s
grid on;
legend('SYS1 = 1/(0.5s+1)', 'SYS2 = 5/(s^2+3s+5)');
```

[Tekst alternatywny. Wykres dwuwymiarowy. Oś X przedstawia czas [s], oś Y przedstawia odpowiedź układu y(t). Na wykresie znajdują się dwie krzywe: odpowiedź skokowa obiektu inercyjnego I rzędu oraz odpowiedź skokowa obiektu inercyjnego II rzędu.]

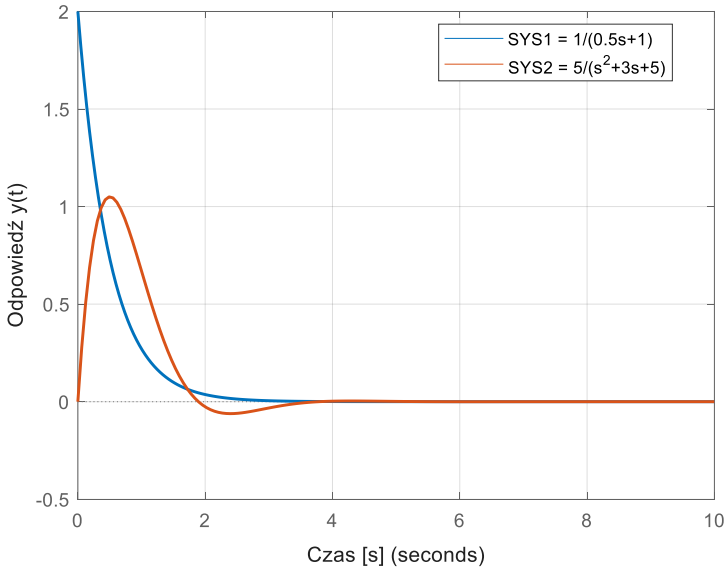


Rys. 3. Odpowiedzi skokowe obiektu inercyjnego I i II rzędu

[Tekst alternatywny. Wykres dwuwymiarowy. Oś X przedstawia czas [s], oś Y przedstawia odpowiedź układu y(t). Na wykresie znajdują się dwie krzywe:

Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki" nr FERS.01.05-IP.08-0234/23

odpowiedź impulsowa obiektu inercyjnego I rzędu oraz odpowiedź impulsowa obiektu inercyjnego II rzędu, bardziej złożona, mogąca przyjmować przebieg oscylacyjny.]



Rys. 4. Odpowiedzi impulsowe obiektu inercyjnego I i II rzędu

2. Identyfikacja parametrów modelu matematycznego obiektu sterowania

W celu wyznaczenia parametrów modelu matematycznego obiektu sterowania można zastosować metody optymalizacji funkcji wielu zmiennych. Metody te pozwalają na lokalizację punktów krytycznych funkcji celu, przede wszystkim jej minimów, co ma kluczowe znaczenie w procesie identyfikacji parametrów modeli obiektów sterowania. Najczęściej przyjmowaną funkcją celu jest kryterium najmniejszych kwadratów, minimalizujące błędy pomiędzy odpowiedzią modelu a rzeczywistymi danymi pomiarowymi:

$$S = \sum_{i=1}^{n} (v_i - y_i)^2 \tag{14}$$

gdzie:

- v_i - wartość zmierzona (eksperymentalna odpowiedź obiektu),
- y_i - wartość uzyskana z modelu matematycznego (symulacja przy zadanych parametrach),
- n - liczba próbek pomiarowych.

Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki" nr FERS.01.05-IP.08-0234/23

Minimalizacja powyższej funkcji pozwala na dopasowanie odpowiedzi modelu do danych rzeczywistych i w konsekwencji na identyfikację parametrów obiektu. Wartości y_i są obliczane jako odpowiedź obiektu na wymuszenie skokowe.

Przykład 11

W ramach przykładu dokonano identyfikacji parametrów modelu obiektu sterowania na podstawie zasymulowanych danych pomiarowych v_i. W tym celu przyjęto model matematyczny opisany transmitancją o nieznanych parametrach K, T_1, T_2, T_3:

$$G_o = \frac{K}{(1+sT_1)(1+sT_2)(1+sT_3)} \tag{15}$$

Proces identyfikacji polegał na dopasowaniu odpowiedzi modelu do danych pomiarowych poprzez minimalizację funkcji kryterium najmniejszych kwadratów (14), co umożliwiło wyznaczenie wartości parametrów modelu.

Rozwiązanie:
Poniższy skrypt w MATLAB-ie generuje dane pomiarowe w postaci odpowiedzi skokowej z dodanym szumem, a następnie identyfikuje parametry modelu za pomocą funkcji optymalizacji fminsearch. `Wynikiem działania programu są zidentyfikowane parametry` K, T_1, T_2, T_3, które pozwalają opisać transmitancję badanego obiektu.

```
clc; clear; close all;

% ================ Parametry obiektu ================
wzmocnienie = 1;     % wzmocnienie statyczne K
tau1 = 2;            % stała czasowa T1
tau2 = 3;            % stała czasowa T2
tau3 = 4;            % stała czasowa T3

% ================ Parametry symulacji ================
czasSymulacji = 50;       % czas trwania symulacji [s]
liczbaPunktow = 1000;     % liczba punktów w wektorze czasu
amplitudaSzum = 0.02;     % amplituda szumu

% ================ Generowanie danych ================
% Odpowiedź skokowa układu z dodanym szumem
[wektorCzasu, odpowiedzZSzumem] =
odpowiedzSkokowaZSzumem(wzmocnienie, tau1, tau2, tau3, ...
    czasSymulacji, liczbaPunktow, amplitudaSzum);
```


Politechnika Świętokrzyska Kielce University of Technology
Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki" nr FERS.01.05-IP.08-0234/23

```
% Dane do identyfikacji
t_dane = wektorCzasu;
y_dane = odpowiedzZSzumem;

% ================= Funkcja celu =================
% Minimalizacja sumy kwadratów błędów (SSE) zgodeni ze wzorem (1)
funkcjaCelu = @(parametry) obliczSSE(parametry, t_dane, y_dane);

% ================= Identyfikacja parametrów =================
% Wartości początkowe parametrów [K, T1, T2, T3]
wartosciPoczatkowe = [1, 3, 3, 2];

% Identyfikacja metodą optymalizacji (fminsearch)
parametryEstymowane = fminsearch(funkcjaCelu, wartosciPoczatkowe);

% ================= Wyniki identyfikacji =================
K_est  = parametryEstymowane(1);
T1_est = parametryEstymowane(2);
T2_est = parametryEstymowane(3);
T3_est = parametryEstymowane(4);

fprintf('Zidentyfikowane parametry modelu:\n');
fprintf('K  = %.4f\n', K_est);
fprintf('T1 = %.4f\n', T1_est);
fprintf('T2 = %.4f\n', T2_est);
fprintf('T3 = %.4f\n', T3_est);

% ================= Porównanie odpowiedzi =================
[t_model, y_model] = odpowiedzSkokowaModelu(parametryEstymowane,
t_dane);

figure;
plot(t_dane, y_dane, 'b', 'LineWidth', 1.5); hold on;
plot(t_model, y_model, 'r--', 'LineWidth', 1.5);
xlabel('Czas [s]');
ylabel('Odpowiedź');
legend('Dane pomiarowe', 'Model zidentyfikowany', 'Location',
'Best');
grid on;

% ================= Funkcje pomocnicze =================
% --- Generowanie odpowiedzi skokowej z szumem ---
function [wektorCzasu, odpowiedzZSzumem] =
odpowiedzSkokowaZSzumem(wzmocnienie, tau1, tau2, tau3, ...
```

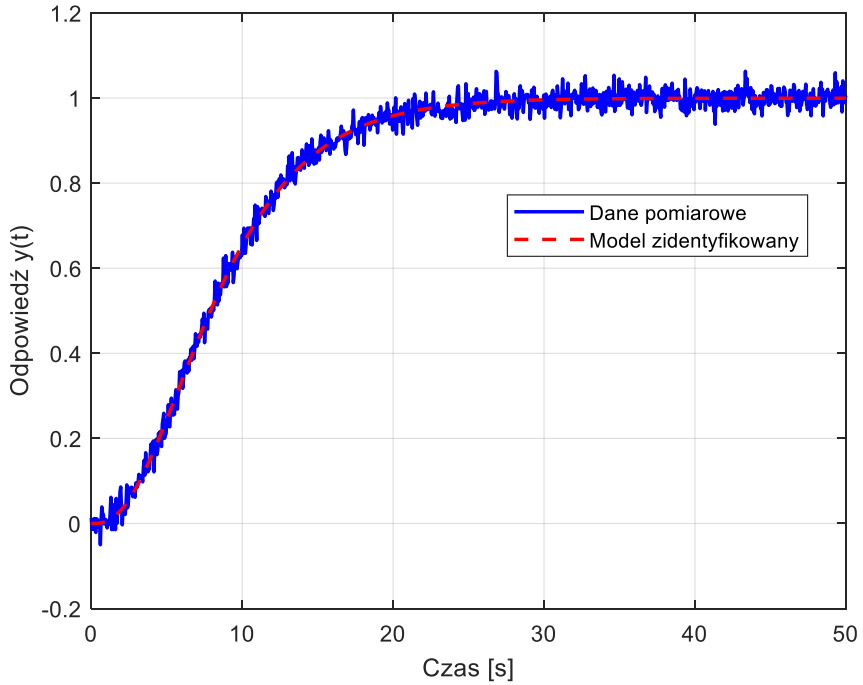

Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki" nr FERS.01.05-IP.08-0234/23

```
        czasSymulacji, liczbaPunktow, amplitudaSzum)

% Transmitancja układu inercyjnego III rzędu
    licznik = wzmocnienie;
    mianownik = [tau1*tau2*tau3, tau1*tau2 + tau1*tau3 + tau2*tau3,
tau1 + tau2 + tau3, 1];
    uklad = tf(licznik, mianownik);

    % Wektor czasu symulacji
    wektorCzasu = linspace(0, czasSymulacji, liczbaPunktow);

    % Odpowiedź skokowa układu
    [odpowiedz, ~] = step(uklad, wektorCzasu);

    % Dodanie szumu losowego
    szum = amplitudaSzum * randn(size(odpowiedz));
    odpowiedzZSzumem = odpowiedz + szum;
end

% --- Funkcja obliczająca SSE ---
function sse = obliczSSE(parametry, t, y_rzeczywiste)

    % Przypisanie parametrów
    K  = parametry(1);
    T1 = parametry(2);
    T2 = parametry(3);
    T3 = parametry(4);

    % Transmitancja modelu
    licznik = K;
    mianownik = [T1*T2*T3, T1*T2 + T1*T3 + T2*T3, T1 + T2 + T3, 1];
    uklad = tf(licznik, mianownik);

    % Odpowiedź modelu dla tego samego wektora czasu
    [y_model, ~] = step(uklad, t);

    % Obliczenie sumy kwadratów błędów
    bledy = y_rzeczywiste - y_model;
    sse = sum(bledy.^2);
end

% --- Generowanie odpowiedzi modelu dla podanych parametrów ---
function [t_model, y_model] = odpowiedzSkokowaModelu(parametry, t)
    % Rozpakowanie parametrów
    K  = parametry(1);
```


Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki" nr FERS.01.05-IP.08-0234/23

```
    T1 = parametry(2);
    T2 = parametry(3);
    T3 = parametry(4);

    % Transmitancja modelu
    licznik = K;
    mianownik = [T1*T2*T3, T1*T2 + T1*T3 + T2*T3, T1 + T2 + T3, 1];
    uklad = tf(licznik, mianownik);

    % Odpowiedź skokowa modelu
    [y_model, t_model] = step(uklad, t);
end
```

[Tekst alternatywny. Wykres dwuwymiarowy. Oś X przedstawia czas [s], oś Y przedstawia odpowiedź układu y(t). Na wykresie znajdują się dwie krzywe: niebieska linia przedstawia odpowiedź skokową uzyskaną z danych pomiarowych (z uwzględnieniem szumu), natomiast czerwona linia przerywana odpowiada odpowiedzi skokowej modelu obiektu o zidentyfikowanych parametrach. Wykres ilustruje stopień dopasowania modelu matematycznego do danych pomiarowych.]



Rys. 5. Odpowiedzi skokowe dla danych pomiarowych i modelu otrzymanego w wyniku identyfikacji

Przeprowadzona identyfikacja pozwoliła uzyskać model matematyczny dobrze odwzorowujący dynamikę badanego obiektu. Odpowiedź zidentyfikowanego modelu jest zbliżona do danych pomiarowych, co potwierdza skuteczność zastosowanej

metody. Niewielkie różnice wynikają głównie z obecności szumu w danych, jednak nie wpływają one znacząco na poprawność estymowanych parametrów.

3. Optymalny dobór parametrów regulatora

Do optymalnego doboru nastaw regulatorów, takich jak PI czy PID, można wykorzystać metody optymalizacji numerycznej, np. funkcję fminsearch w środowisku MATLAB. Proces ten polega na minimalizacji wybranego kryterium jakości regulacji. Najczęściej stosowane są kryteria całkowe:

a) IAE (Integral of Absolute Error):

$$IAE = \int_0^\infty |e(t)|dt \tag{16}$$

gdzie: *e(t)* to uchyb zamkniętego układu regulacji.

Kryterium traktuje wszystkie błędy równomiernie, niezależnie od ich wielkości, dzięki czemu jest mniej wrażliwe na duże odchylenia i łatwe w implementacji. Jednak nie uwzględnia czasu występowania błędu, co może prowadzić do wydłużenia czasu regulacji.

b) ISE (Integral of Squared Error):

$$ISE = \int_0^\infty e^2(t)dt \tag{17}$$

Minimalizuje duże błędy, co jest korzystne w systemach, gdzie istotne są znaczne odchylenia od wartości zadanej. Nie uwzględnia jednak czasu ich występowania, co może powodować wydłużenie czasu regulacji.

c) ITSE (Integral of Time-weighted Squared Error):

$$ITSE = \int_0^\infty t \cdot e^2(t)dt \tag{18}$$

Kryterium sprzyja szybszemu tłumieniu oscylacji i jest korzystne w systemach, w których priorytetem jest stabilność. Z drugiej strony, jest bardziej złożone obliczeniowo i wrażliwe na błędy pojawiające się w późniejszych etapach odpowiedzi.

d) ITAE (Integral of Time-weighted Absolute Error):

$$ITAE = \int_0^\infty t \cdot |e(t)|dt \tag{19}$$

Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki" nr FERS.01.05-IP.08-0234/23

Kryterium jest ważone czasem, co sprzyja minimalizacji błędów w późniejszych etapach regulacji i zapewnia szybsze tłumienie oscylacji oraz krótszy czas regulacji. Jednocześnie jest bardziej złożone w implementacji i wrażliwe na zakłócenia pojawiające się w późniejszych fazach odpowiedzi.

Przykład 12

Rozważono obiekt inercyjny III rzędu opisany transmitancją:

$$G_o = \frac{K}{(1+sT_1)(1+sT_2)(1+sT_3)} \tag{20}$$

gdzie: K = 1, T_1 = 3,38, T_2 = 3,4, T_3 = 2,17.

Celem prezentowanego przykładu jest wyznaczenie nastaw regulatora PI, które minimalizują błąd regulacji według kryterium ITAE (19).

Rozwiązanie:

Optymalizację przeprowadzono w środowisku MATLAB z użyciem funkcji fminsearch, która poszukiwała wartości minimalnej wskaźnika ITAE.

```
clc; clear; close all

% ================= Parametry symulacji =================
zakresCzasu = [0 100];  % przedział czasu symulacji odpowiedzi
skokowej
s = tf('s');            % operator Laplace'a

% ================= Parametry obiektu =================
% Obiekt inercyjny III rzędu o znanych stałych czasowych
T1 = 3.38;
T2 = 3.40;
T3 = 2.17;

% Transmitancja operatorowa obiektu
G = 1 / ((T1*s + 1)*(T2*s + 1)*(T3*s + 1));

% ================= Optymalizacja regulatora PI =================
% Funkcja celu – minimalizacja wskaźnika ITAE
funkcjaCelu = @(parametry) obliczITAE(parametry, G, zakresCzasu);

% Parametry początkowe regulatora [Kp, Ki]
parametryPoczatkowe = [1, 1];

% Tablica do zapisywania historii wartości ITAE
```


Politechnika Świętokrzyska Kielce University of Technology

Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki" nr FERS.01.05-IP.08-0234/23

```
global wartosciITAE
wartosciITAE = [];

% Opcje optymalizacji – zapis wartości w każdej iteracji
opcje = optimset('OutputFcn', @zapiszIteracje);

% Optymalizacja parametrów regulatora PI za pomocą fminsearch
parametryOptymalne = fminsearch(funkcjaCelu, parametryPoczatkowe,
opcje);
Kp_optymalne = parametryOptymalne(1);
Ki_optymalne = parametryOptymalne(2);

% ================ Wyniki ================
fprintf('\n=== WYNIKI OPTYMALIZACJI ===\n');
fprintf('Optymalne parametry regulatora PI:\n');
fprintf('Kp = %.4f\n', Kp_optymalne);
fprintf('Ki = %.4f\n', Ki_optymalne);

% ================ Symulacje ================
% Odpowiedź obiektu bez regulatora
[y_bez, t_bez] = step(G, zakresCzasu);
info_bez = stepinfo(G);

% Regulator PI w postaci transmitancji
C = Kp_optymalne + Ki_optymalne / s;

% Zamknięty układ regulacji (sprzężenie zwrotne)
T = feedback(C*G, 1);

% Odpowiedź układu z regulatorem
[y, t] = step(T, zakresCzasu);
info_reg = stepinfo(T);

% ================ Analiza porównawcza ================
fprintf('\nPorównanie parametrów dynamicznych:\n');
fprintf('====================================\n');
fprintf('Parametr            |  Bez regulatora   |  Z regulatorem
PI\n');
fprintf('-----------------------------------------------------------
-\n');
fprintf('Czas narastania [s] |  %.4f            |  %.4f\n',
info_bez.RiseTime, info_reg.RiseTime);
fprintf('Czas ustalania [s]  |  %.4f            |  %.4f\n',
info_bez.SettlingTime, info_reg.SettlingTime);
```

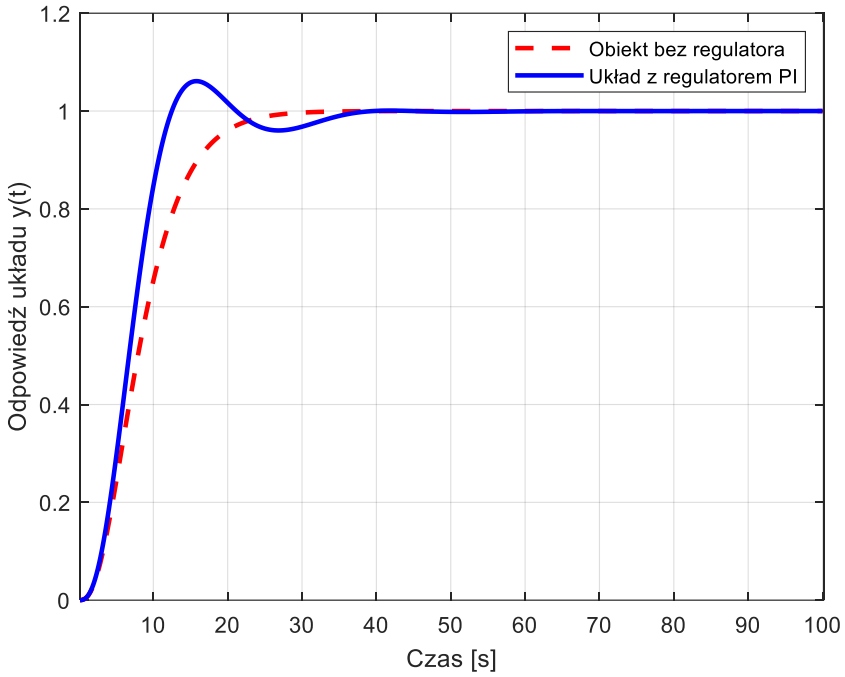

Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki" nr FERS.01.05-IP.08-0234/23

```
fprintf('Przeregulowanie [%%] |  %.2f              |  %.2f\n',
info_bez.Overshoot, info_reg.Overshoot);
fprintf('Czas szczytu [s]    |  %.4f              |  %.4f\n',
info_bez.PeakTime, info_reg.PeakTime);
fprintf('Wartość szczytu     |  %.4f              |  %.4f\n',
info_bez.Peak, info_reg.Peak);

% ================ Wizualizacja wyników ================
% Porównanie odpowiedzi obiektu i układu z regulatorem
figure;
plot(t_bez, y_bez, 'r--', 'LineWidth', 2); hold on;
plot(t, y, 'b', 'LineWidth', 2);
grid on;
title('Odpowiedź obiektu bez regulatora i z regulatorem PI');
xlabel('Czas [s]');
ylabel('Sygnał wyjściowy y(t)');
legend('Obiekt bez regulatora', 'Układ z regulatorem PI');

% Wykres zbieżności metody fminsearch
figure;
plot(wartosciITAE, '-o', 'LineWidth', 2);
grid on;
title('Zbieżność metody fminsearch (minimalizacja ITAE)');
xlabel('Iteracja');
ylabel('Wartość ITAE');

% ============================================================
% Funkcje pomocnicze
% ============================================================

% Funkcja obliczająca wskaźnik ITAE dla regulatora PI
function itae = obliczITAE(parametry, G, zakresCzasu)
    s = tf('s');
    Kp = parametry(1);
    Ki = parametry(2);
    C = Kp + Ki / s;        % regulator PI
    T = feedback(C*G, 1);   % układ zamknięty
    [y, t] = step(T, zakresCzasu);
    e = 1 - y;              % uchyb regulacji
    itae = trapz(t, t .* abs(e)); % całkowanie ITAE
end

% Funkcja zapisująca wartości ITAE w każdej iteracji
function stop = zapiszIteracje(~, optymalizacja, stan)
    stop = false;
```


Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki" nr FERS.01.05-IP.08-0234/23

```
    global wartosciITAE
    if isequal(stan, 'iter')
        wartosciITAE(end+1) = optymalizacja.fval;
    end
end
```

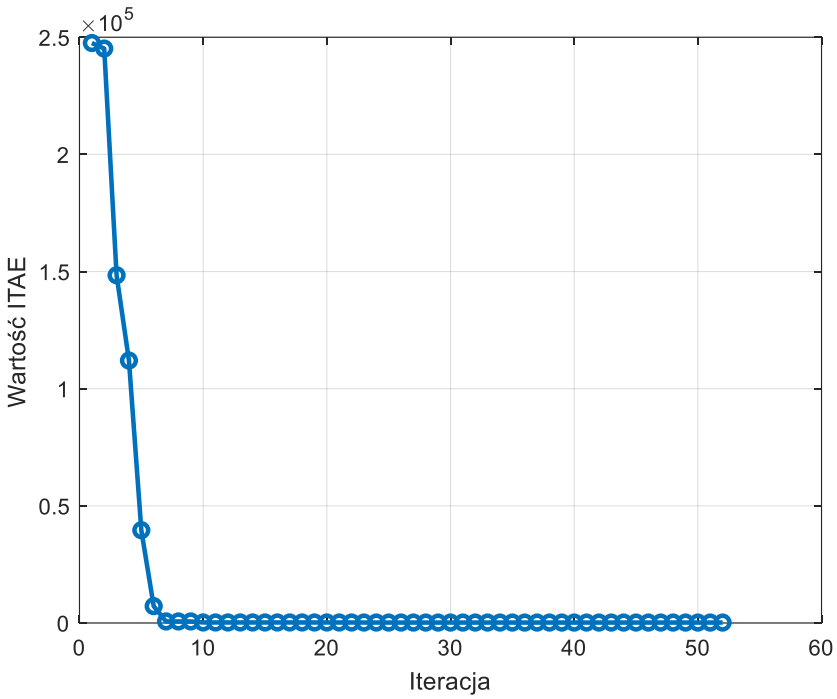
[Tekst alternatywny. Wykres dwuwymiarowy. Oś X przedstawia czas [s], oś Y przedstawia odpowiedź układu y(t). Na wykresie znajdują się dwie krzywe: czerwona linia przerywana obrazuje odpowiedź skokową układu otwartego (bez regulatora), natomiast niebieska linia ciągła przedstawia odpowiedź układu zamkniętego z regulatorem PI. Wykres umożliwia porównanie dynamiki obu układów.] Widoczna jest poprawa dynamiki - czas narastania skrócił się z ok. 12,75 s do 7,64 s, co oznacza szybszą reakcję układu na wymuszenie skokowe. Jednocześnie czas ustalania wydłużył się (z 22,78 s do 32,61 s), a dodatkowo pojawiło się przeregulowanie rzędu 6,1 %, którego nie obserwowano dla obiektu bez regulatora. Wartość maksymalna odpowiedzi wzrosła z 0,999 do 1,061, co wiąże się z przeregulowaniem.



Rys. 6. Porównanie odpowiedzi skokowych układu otwartego (bez regulatora) i układu zamkniętego z regulatorem PI

[Tekst alternatywny. Wykres dwuwymiarowy. Oś X przedstawia numer iteracji algorytmu, oś Y przedstawia wartość funkcji celu ITAE. Na wykresie znajduje się linia obrazująca zmiany wartości ITAE w kolejnych iteracjach, co ilustruje proces zbieżności algorytmu optymalizacji do minimum.]

Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki" nr FERS.01.05-IP.08-0234/23



Rys. 7. Zbieżność algorytmu optymalizacji

W trakcie optymalizacji zauważono, że początkowa wartość wskaźnika ITAE była bardzo wysoka, co wskazywało na duży błąd regulacji przy parametrach startowych regulatora. Już w pierwszych kilku iteracjach algorytmu fminsearch nastąpił gwałtowny spadek wartości funkcji celu, co świadczy o szybkim odnalezieniu obszaru bliskiego optimum. Od około 7. iteracji wartości ITAE ustabilizowały się, a dalsze obliczenia nie przynosiły istotnej poprawy. Ostatecznie proces optymalizacji przebiegł sprawnie i doprowadził do wyznaczenia dobrze dopasowanych parametrów regulatora PI zgodnych z kryterium ITAE.

4. Optymalizacja parametrów regulatora PI z wykorzystaniem algorytmu PSO

Algorytm PSO (*Particle Swarm Optimization*) to metoda optymalizacji inspirowana naturalnym zachowaniem stadnym, obserwowanym m.in. w stadach ptaków czy ławicach ryb. W procesie tym cząstki poruszają się w przestrzeni rozwiązań, aktualizując swoją pozycję i prędkość na podstawie dotychczasowych najlepszych wyników - zarówno indywidualnych (lokalnych), jak i globalnych. Dzięki temu możliwe jest odnalezienie rozwiązania bliskiego optimum w problemach nieliniowych i wielowymiarowych.

Przykład 13

Celem przykładu jest zastosowanie algorytmu PSO do doboru optymalnych nastaw regulatora PI dla obiektu inercyjnego 3. rzędu. Jako kryterium jakości przyjęto wskaźnik ITAE. Przyjęto obiekt regulacji opisany wcześniej transmitancją (20).

Program w środowisku MATLAB realizuje zadanie poprzez:

- definicję transmitancji obiektu,
- zbudowanie funkcji celu computeITAE, obliczającej wartość kryterium ITAE,
- minimalizację tej funkcji z użyciem wbudowanej metody particleswarm,
- symulację odpowiedzi obiektu bez regulatora i układu z regulatorem PI,
- analizę podstawowych parametrów dynamicznych odpowiedzi skokowych za pomocą funkcji stepinfo.

```
clear; clc; close all;

% ================= Definicja układu =================
s = tf('s'); % operator Laplace'a

% Parametry obiektu (układ inercyjny III rzędu)
K = 1;
T1 = 3.38;
T2 = 3.40;
T3 = 2.17;

obiekt = 1 / ((T1*s + 1)*(T2*s + 1)*(T3*s + 1));

% ================= Optymalizacja metodą PSO =================
% Zakresy poszukiwań dla parametrów regulatora PI [Kp, Ki]
lb = [0.01, 0.01];   % dolne ograniczenia
ub = [10,   10];     % górne ograniczenia

% Funkcja celu – minimalizacja ITAE
fun = @(x) obliczITAE(obiekt, x(1), x(2), 100);

% Opcje algorytmu PSO
options = optimoptions('particleswarm', ...
    'SwarmSize', 30, ...         % liczba cząstek
    'MaxIterations', 50, ...     % maksymalna liczba iteracji
    'Display', 'iter');          % podgląd postępu

% Optymalizacja parametrów regulatora PI
[x_opt, fval] = particleswarm(fun, 2, lb, ub, options);

% ================= Wyniki =================
Kp_opt = x_opt(1);
```


Politechnika Świętokrzyska Kielce University of Technology
Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki" nr FERS.01.05-IP.08-0234/23

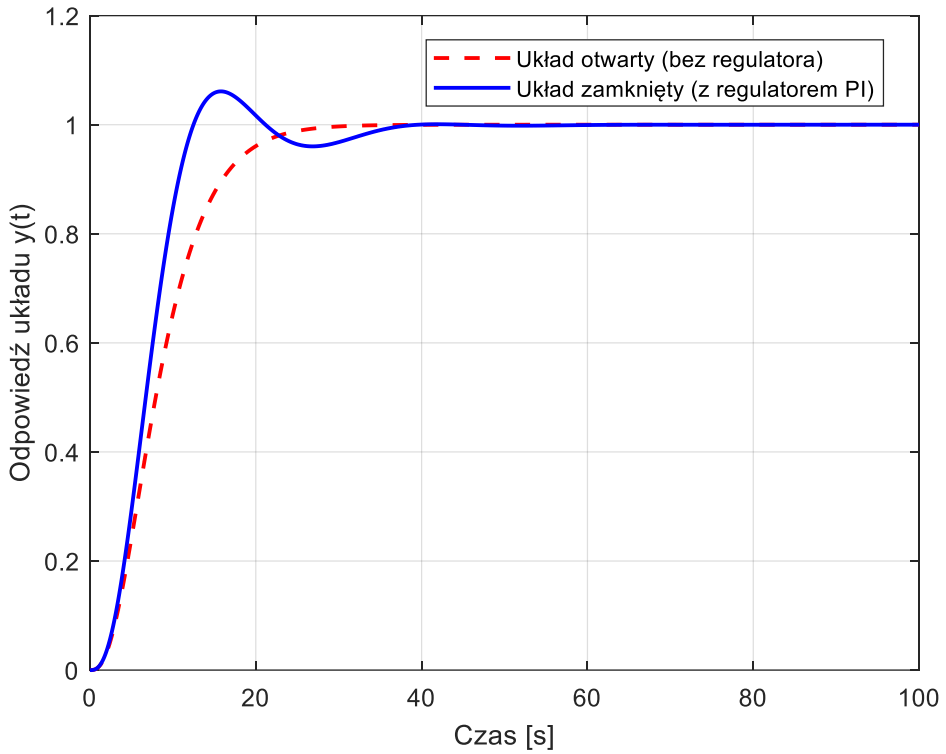
```
Ki_opt = x_opt(2);

fprintf('Optymalne nastawy regulatora PI:\n');
fprintf('Kp = %.4f, Ki = %.4f\n', Kp_opt, Ki_opt);

% ================ Symulacje ================
% Regulator PI w postaci operatorowej (Kp + Ki/s)
regulator = Kp_opt + Ki_opt/s;

% Układ zamknięty z regulatorem
ukladZamkniety = feedback(regulator * obiekt, 1);

% Symulacje odpowiedzi skokowych
czas = 0:0.01:100;
[y_bez,  t_bez] = step(obiekt, czas);           % odpowiedź obiektu
bez regulatora
[y_reg,  t_reg] = step(ukladZamkniety, czas);   % odpowiedź układu z
regulatorem PI

% Analiza dynamiczna
info_bez = stepinfo(y_bez, t_bez);
info_reg = stepinfo(y_reg, t_reg);

% Wyniki w konsoli
fprintf('\nPorównanie parametrów dynamicznych:\n');
fprintf('Parametr            |  Bez regulatora   |  Z regulatorem
PI\n');
fprintf('-------------------------------------------------------------
-\n');
fprintf('Czas narastania [s] |  %.4f            |  %.4f\n',
info_bez.RiseTime, info_reg.RiseTime);
fprintf('Czas ustalania [s]  |  %.4f            |  %.4f\n',
info_bez.SettlingTime, info_reg.SettlingTime);
fprintf('Przeregulowanie [%%] |  %.2f              |  %.2f\n',
info_bez.Overshoot, info_reg.Overshoot);
fprintf('Czas szczytu [s]    |  %.4f            |  %.4f\n',
info_bez.PeakTime, info_reg.PeakTime);
fprintf('Wartość szczytu     |  %.4f            |  %.4f\n',
info_bez.Peak, info_reg.Peak);

% ================ Wizualizacja ================
figure;
plot(t_bez, y_bez, 'r--', 'LineWidth', 1.5); hold on;
plot(t_reg, y_reg, 'b-', 'LineWidth', 1.5);
```


Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki" nr FERS.01.05-IP.08-0234/23

```
legend('Układ otwarty (bez regulatora)', 'Układ zamknięty (z
regulatorem PI)', 'Location', 'Best');
xlabel('Czas [s]');
grid on;

% ================ Funkcja pomocnicza ================
function ITAE = obliczITAE(obiekt, Kp, Ki, czasSymulacji)
    s = tf('s');
    regulator = Kp + Ki/s;              % regulator PI w postaci Kp
+ Ki/s
    uklad = feedback(regulator * obiekt, 1);
    [y, t] = step(uklad, czasSymulacji);
    e = 1 - y;                          % uchyb regulacji
    ITAE = trapz(t, t .* abs(e));       % kryterium ITAE
end
```

[Tekst alternatywny. Wykres dwuwymiarowy. Oś X przedstawia czas [s], oś Y przedstawia odpowiedź układu y(t). Na wykresie znajdują się dwie krzywe: czerwona linia przerywana obrazuje odpowiedź skokową układu otwartego (bez regulatora), natomiast niebieska linia ciągła przedstawia odpowiedź układu zamkniętego z regulatorem PI dobranym metodą PSO. Wykres umożliwia ocenę wpływu regulatora na dynamikę układu.] Na podstawie przeprowadzonej optymalizacji metodą PSO uzyskano optymalne parametry regulatora PI: wzmocnienie członu proporcjonalnego K_p = 1,0158 oraz wzmocnienie członu całkującego K_i = 0,1456. Na rysunku 8 przedstawiono porównanie odpowiedzi obiektu bez regulatora i układu z regulatorem PI. Regulator istotnie wpływa na charakterystykę dynamiczną układu. Czas narastania skrócił się z około 12,7 s do 7,6 s, co świadczy o szybszej reakcji na wymuszenie skokowe. Jednocześnie czas ustalania wydłużył się z 22,8 s do 32,6 s, co oznacza, że układ dłużej stabilizuje się wokół wartości zadanej. W odpowiedzi pojawiło się przeregulowanie na poziomie 6,1 %, którego obiekt bez regulatora nie wykazywał, a wartość szczytowa wzrosła z 1,00 do 1,0611. Podsumowując, regulator PI poprawił szybkość reakcji układu, ale kosztem dłuższego czasu ustalania i pojawienia się przeregulowania, co stanowi typowy kompromis w sterowaniu z użyciem tego typu regulatorów.

Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki" nr FERS.01.05-IP.08-0234/23



Rys. 8. Porównanie odpowiedzi skokowych układu otwartego (bez regulatora) i układu zamkniętego z regulatorem PI dla metody PSO

Do strojenia regulatorów można zastosować algorytm rojowy PSO (*Particle Swarm Optimization*) lub metodę Neldera–Meada (*fminsearch*). Algorytm PSO ma charakter globalny – jest odporny na minima lokalne, nie wymaga znajomości pochodnych i dobrze sprawdza się w problemach nieliniowych. Jego wadą jest większa czasochłonność oraz stochastyczność, co oznacza, że kolejne uruchomienia mogą dawać różne wyniki. Z kolei metoda *fminsearch* jest szybka i prosta w implementacji, jednak działa lokalnie i silnie zależy od wartości początkowych, co może skutkować uzyskaniem nieoptymalnych rezultatów.

5. Literatura

[1] Kochenderfer M.J., Wheeler T.A., (2019), Algorithms for Optimization, The MIT Press.
[2] Nise N., (2011), *Control System Engineering. International Student Version*, John Wiley&Sons.
[3] Pomoc programu MATLAB: https://www.mathworks.com/help/control/index.html?s_tid=CRUX_topnav, dostęp online 12.07.2025.