SPRAWOZDANIE 10

Modelowanie Systemów Dynamicznych

Temat ćwiczenia: Identyfikacja obiektu regulacji

Michał Midor, gr. 5, 17.12.2025 r. - Środa 13.15

Cel ćwiczenia:

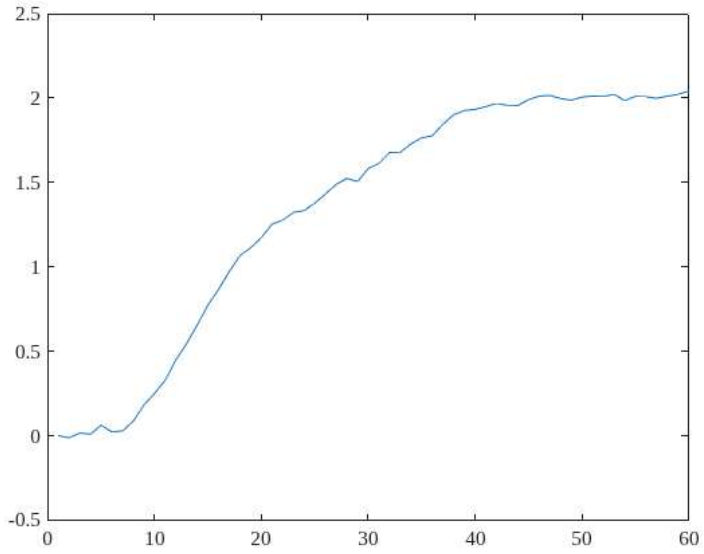
Celem ćwiczenia jest poznanie metod identyfikacji parametrów modelu rzeczywistego obiektu regulacji. Obiekty rzeczywiste są nieskończenie wymiarowe, posiadają zakłócenia i ich odpowiedzi nigdy nie będą tak głądkie, jak te z matematycznych obiektów, lecz w sterowaniu dla uproszczenia można opisać je różnymi modelami transmitancyjnymi, tak zwanymi obiektami inercyjnymi.

Najpierw ręcznie zostaną dobrane parametry modelu zastępczego i próba zminimalizowania błędu wyznaczonego za pomocą metody średniej kwadratów. Dalszym etapem będzie stworzenie funkcji zwracającej błąd dla różnego rodzaju zakładanego obiektu, a następnie użyta zostanie funkcja optymalizacyjna szukająca najlepszych parametrów (takich, dla których błąd jest najmniejszy) i porównane zostaną minimalne błędy z różnych modeli.

Rozwiązanie zadań:

Ćwiczenie rozpoczyna się od wczytania odpowiedzi skokowej rzeczywistego modelu oraz przedstawienia jej na wykresie.

```
load obiekt.mat
global t;
global y;
t = 0:59;

plot(y);
```



Następnie dobierane są parametry dla obiektu inercyjnego I rzędu z opóźnieniem o transmitancji:

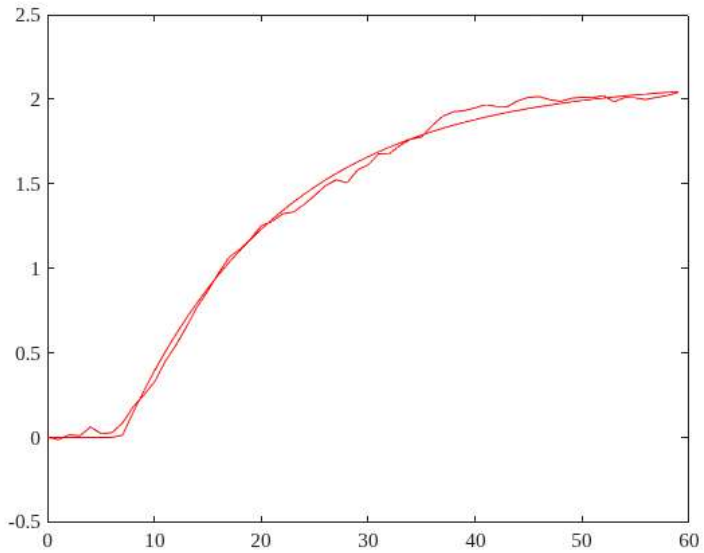
$$G(s) = \frac{ke^{-s\theta}}{Ts+1}.$$

Wzmocnienie k, stała czasowa T, oraz opóźnienie theta zostają dobrane "na oko", by wykresy odpowiedzi skokowych modelu rzeczywistego i symulowanego były zbliżone. Zaczyna się od prostych parametrów wstępnych parząc na wykres, a po paru eksperymentach można dojść do parametrów w miarę optymalnych.

```
display("Model A z ręcznie dobranymi parametrami.")
```

```
"Model A z ręcznie dobranymi parametrami."
```

```
% parametry wstępne
%k = 2;
%T = 10;
%theta = 8;

% parametry optymalne
k = 2.11;
T = 14.92;
theta = 6.9;

sys_order1 = tf(k, [T 1], "InputDelay", theta);
figure();
y_order1 = step(sys_order1, t);
plot(t, y, t, y_order1, "Color", "red"); drawnow;
```




```
error = (y-y_order1);
result_error = sum(error.^2)/length(error)
```

```
result_error =
0.0017
```

Dla tak dobranych parametrów błąd wynoski 0.0017, co jak się okaże poźniej, jest całkiem dobrym wynikiem.

Chcąc skorzystać z funkcji optymalizującej 'fminsearch' z biblioteki ‘Optimization Toolbox’, najpierw definiuje się funkcję zwracającą wartość błędu dla zadanych parametrów. Pierwsza funkcja 'identA' będzie przyjmowała parametry dla obiektu inercyjnego I rzędu z opóźnieniem. Wygląda ona następująco:

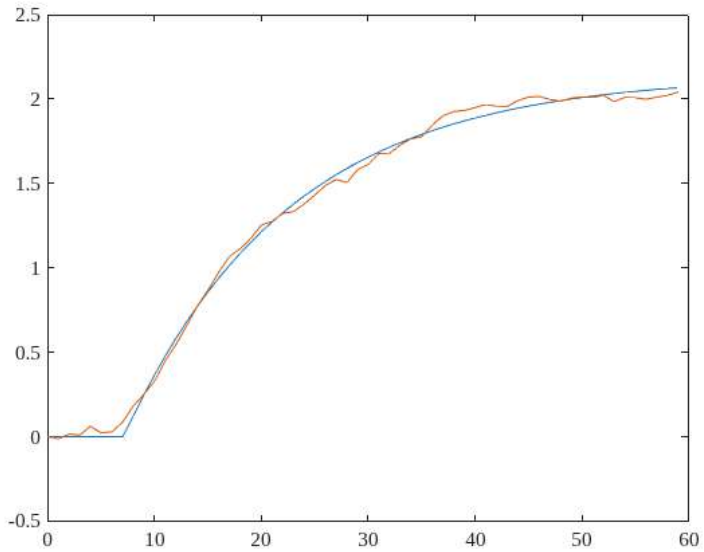
```
function blad = identA(X0)
global y;
global t;

K = X0(1);
T = X0(2);
theta = X0(3);

sys = tf(K, [T 1], "InputDelay", theta);
y_sym = step(sys, t);
plot(t, y_sym, t, y); % drawnow; % pause(0.1)

e = y - y_sym;
blad = sum(e.^2) / length(e);
end
```

Funkca 'fminsearch' wywoła funkcję 'identA' wiele razy z podanym wektorem parametrów 'X0_A' i zwróci parametry obiektu, dla których błąd jest minimalny.

```
X0_A = [k, T, theta];
% X0_A = [2 15 6];
display("Model A z parametrami zoptymalizowanymi automatycznie.")
```

```
"Model A z parametrami zoptymalizowanymi automatycznie."
```

```
[parametry, blad] = fminsearch('identA', X0_A);
```




```
blad_A = blad
```

```
blad_A =
0.0015
```

Błąd wynosi 0.0015, a zatem ręcznie dobrane parametry były już bardzo dobre. Przebieg odpowiedzi skokowej symulowanego modelu w porównianiu do rzeczywistego widoczny jest na wykresie.

Może się jednak okazać, że zaproponowany obiekt nie jest najlepszym dopasowaniem do obiektu rzeczywistego, więc należy sprawdzić kolejne możliwości. Jedną z nich jest obiekt inercyjny II rzędu o transmitancji:

$$G(s) = \frac{ke^{-s\theta}}{(T_1 s + 1)(T_2 s + 1)}$$

Należy wybrać parametry początkowe (punkt startowy), stworzyć funkcję zwracającą błąd dla zadanego typu obiektu, tym razem wygląda ona następująco:

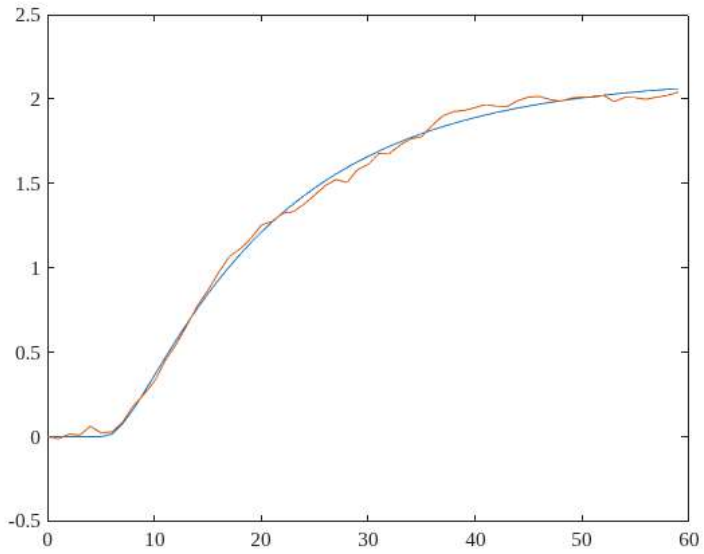
```
function blad = identB(X0)
K = X0(1);
T = X0(2);
theta = X0(3);
Ksi = X0(4);
global y;
global t;

sys = tf(K, [T.^2 2.*T.*Ksi 1]);
set(sys, 'outputdelay', theta);
y_sym = step(sys, t);

plot(t, y_sym, t, y); % drawnow; % pause(0.1)

e = y - y_sym;
blad = sum(e.^2) / length(e);
end
```

Następnie wywoływana jest metoda 'fminsearch'.

```
X0_B = [2 5 5 1.5];
display("Model B z parametrami zoptymalizowanymi automatycznie.")
```

```
"Model B z parametrami zoptymalizowanymi automatycznie."
```

```
[parametry, blad] = fminsearch('identB', X0_B); % Punkt startowy bardzo
ważny, bo może prowadzić do błędu
```




```
blad_B = blad
```

```
blad_B =
0.0013
```

```
parametry_B = parametry
```

```
parametry_B = 1×4
    2.1292    5.5212    5.2836    1.5300
```

Tym razem wynik błędu wynosi 0.0013, co jest poprawą w porównaniu do pierszego obietku. Dopasowanie można zaobserwować na wykresie.

Obiekt regulacji może być również modelem wieloinercyjnym bez opóźnienia, opisanym następującą transmitancją:

$$G(s) = \frac{k}{(Ts+1)^n}$$

Problemem jest potęga mianownika transmitancji, co uniemożliwa użycie funkcji tf(). Możliwe natomiast jest wykorzystanie zpk(). Transmitancja po przekształceniu nie posiada zer, a bieguny są jednym pierwiastkiem wielokrotnym (-1/T), a wzmocnienie wynosi k/(T^n).

Funkcja błędu wygląda zatem następująco:

```
function blad = identC(X0)
global y;
global t;
global n;

k = X0(1);
T = X0(2);

zeros = [];
ones_to_scale = ones(1, n);
poles_C = -1/T * ones_to_scale;
K = k/(T.^n);

sys = zpk(zeros, poles_C, K);
y_sym = step(sys, t);
% plot(t, y_sym, t, y);  drawnow; % pause(0.1)

e = y - y_sym;
blad = sum(e.^2) / length(e);
end
```

Chcąc przeprowadzić analizę błędu dla różnych wartości n, nie można podać 'n' jako kolejnego elementu wektora argumentów, gdyż funkcja optymalizująca starałaby się również ten parametr zoptymalizować, co nie jest fizycznie możliwe, 'n' musi być liczbą naturalną, powinno być również większe od 2, ponieważ inaczej wyszłyby poprzednie obiekty.

Rozwiązaniem jest wywołanie funkcji optymalizującej w pętli, dla różnych 'n', lecz 'n' jest przekazywana do funkcji jako zmienna globalna.

```
display("Model C")
```

```
    "Model C"
```

```
k_C = 1;
T_C = 1;
X0_C = [k_C, T_C];

n_val = [3, 4, 5, 6];
global n

display("Wyniki błędu modelu C dla różnych n.")
```

```
    "Wyniki błędu modelu C dla różnych n."
```

```
for i=1:length(n_val);
    n = n_val(i);
    display(strcat('n=',string(n),':'));
    [parametry, blad] = fminsearch('identC', X0_C) % Punkt startowy bardzo
ważny, bo może prowadzić do błędu
end
```

```
    "n=3:"
parametry = 1×2
    2.0396    6.6554
blad =
```

```
0.0024
    "n=4:"
parametry = 1×2
    1.9881    4.7881
blad =
0.0040
    "n=5:"
parametry = 1×2
    1.9567    3.7380
blad =
0.0071
    "n=6:"
parametry = 1×2
    1.9346    3.0613
blad =
0.0106
```

Najmniejszy błąd został uzyskany dla n = 3, każde zwiększenie tej wartości powoduje coraz większy błąd. Do wizualizacji odpowiedzi skokowej tego modelu należy użyć parametrów zwróconych właśnie dla tego n.

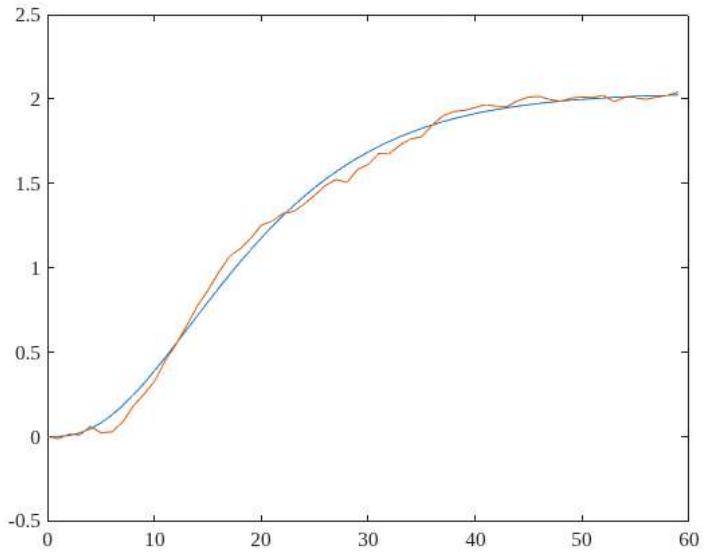
```
n = 3;
[parametry, blad] = fminsearch('identC',X0_C)
```

```
parametry = 1×2
    2.0396    6.6554
blad =
0.0024
```

```
k_C = parametry(1);
T_C = parametry(2);

zeros = [];
ones_to_scale = ones(1, n);
poles_C = -1/T_C * ones_to_scale;
K = k_C/(T_C.^n);

sys = zpk(zeros, poles_C, K);
y_sym = step(sys, t);
display("Model C z parametrami zoptymalizowanymi automatycznie dla n = 3.")
```

```
    "Model C z parametrami zoptymalizowanymi automatycznie dla n = 3."
```

```
plot(t, y_sym, t, y);
```



Najmniejszy błąd uzyskany dla tego modelu równy 0.0024 jest większy od błędów dla porzednich modeli. Dla modelu A wyniósł on 0.0015, a dla modelu B 0.0013. Analizując uzyskane wyniki błędów można stwierdzić, że obiekt rzeczywisty jest obiektem inercyjnym II rzędu z opóźnieniem równym 5,2836.