Wydział Elektroniki i Technik Informacyjnych
Politechnika Warszawska

Przygotowanie pracy magisterskiej

(semestr 25L)

Raport - luty

Autor:
Wojciech Rogalski

Promotor:
dr hab. inż. Piotr Marusak

Warszawa, 2024

Spis treści

1. Wstęp 2
2. Hammerstein 3
2.1. Następniki liniowe 4
2.2. Następniki nieliniowe 6
3. Wiener 8
3.1. Następniki liniowe 8
3.2. Następniki nieliniowe 10
4. Testowania modeli 11
5. DMC 17
6. Odpowiedź skokowa obiektu rzeczywistego 24
Spis rysunków 26
Spis tabel 27

1. Wstęp

Przygotowałem raport z prac w lutym. Na wstępie zaznaczę co udało się zrobić:

- Rozmycie statyki w modelach Hammersteina i Wienera z następnikami liniowymi.
- Rozmycie statyki modelach Hammersteina i Wienera z następnikami nieliniowymi - hiperbolicznymi (sinh).
- Porównanie wyników dla przykładowych wymuszeń.
- Zaprojektowanie regulatorów DMC-analitycznego, DMC-numerycznego, DMC-SL, DMC-NPL oraz FDMC.
- Porównanie wyników otrzymanych dla poszczególnych regulatorów.

Oprócz tego starałem się otrzymać odpowiedź skokową z obiektu w pracy, natomiast tutaj sytuacja jest trochę bardziej skomplikowana i chciałbym prosić o pomoc w identyfikacji tego obiektu. Dokładne informacje w dalszej części raportu.

2. Hammerstein

W modelu Hammersteina wprowadziłem nieliniowość na wejściu rozmywając sygnał sterujący. W przypadku następników liniowych było to pięć zbiorów rozmytych, natomiast w przypadku następników nieliniowych w postaci sinusa hiperbolicznego, były to już tylko trzy zbiory. Podjąłem próbę z dwoma zbiorami i jest to możliwe, natomiast czas jaki należy poświęcić na dostrajanie modelu bardzo się wydłuża, a za cel postawiłem sobie dostrojenie modelu z następnikami liniowymi i nieliniowymi w podobnym czasie i udało się spełnić to założenie.

2.1. Następniki liniowe

Po lepszym zapoznaniu się z Fuzzy Logic Toolbox postanowiłem wykorzystać oferowane przez to narzędzie funkcje. Zbudowałem model rozmyty:

```
function linearFuzzy(obj)
U_center = linspace(obj.U_min, obj.U_max, 5);
obj.linear_fis = sugfis('Name', 'Linear_Hammerstein',...
                        'Type', 'sugeno');
obj.linear_fis = addInput(obj.linear_fis, [obj.U_min obj.U_max],...
                        'Name', 'U_linear');

% Definiowanie funkcji przynależności (gaussmf)
for i = 1:length(U_center)
   obj.linear_fis = addMF(obj.linear_fis, 'U_linear', 'gaussmf',...
                        [12, U_center(i)]);
end

% Definiowanie wyjścia i początkowych następników (a_i * u + b_i)
obj.linear_fis = addOutput(obj.linear_fis, [obj.U_min obj.U_max],...
                                'Name', 'U_fuzzy');

% Współczynniki (a_i, b_i) następników
a_param = [0.7895 0.8982 1.0933 1.0489 1.2034];
b_param = [0.0001 0.0002 0.0001 0 0.001];

% Dodanie reguł TS w postaci liniowej
for i = 1:length(U_center)
   obj.linear_fis = addMF(obj.linear_fis, 'U_fuzzy', 'linear',...
                        [a_param(i), b_param(i)]);
end

% Reguły Takagi-Sugeno: [inputMF, outputMF, weight]
ruleList = [1 1 1 1;  % Reguła 1: wejście MF1 -> wyjście Out1
            2 2 1 1;  % Reguła 2: wejście MF2 -> wyjście Out2
            3 3 1 1;  % Reguła 3: wejście MF3 -> wyjście Out3
            4 4 1 1;  % Reguła 4: wejście MF4 -> wyjście Out4
            5 5 1 1]; % Reguła 5: wejście MF5 -> wyjście Out5

% Dodanie reguł do systemu
obj.linear_fis = addRule(obj.linear_fis, ruleList);
end
```

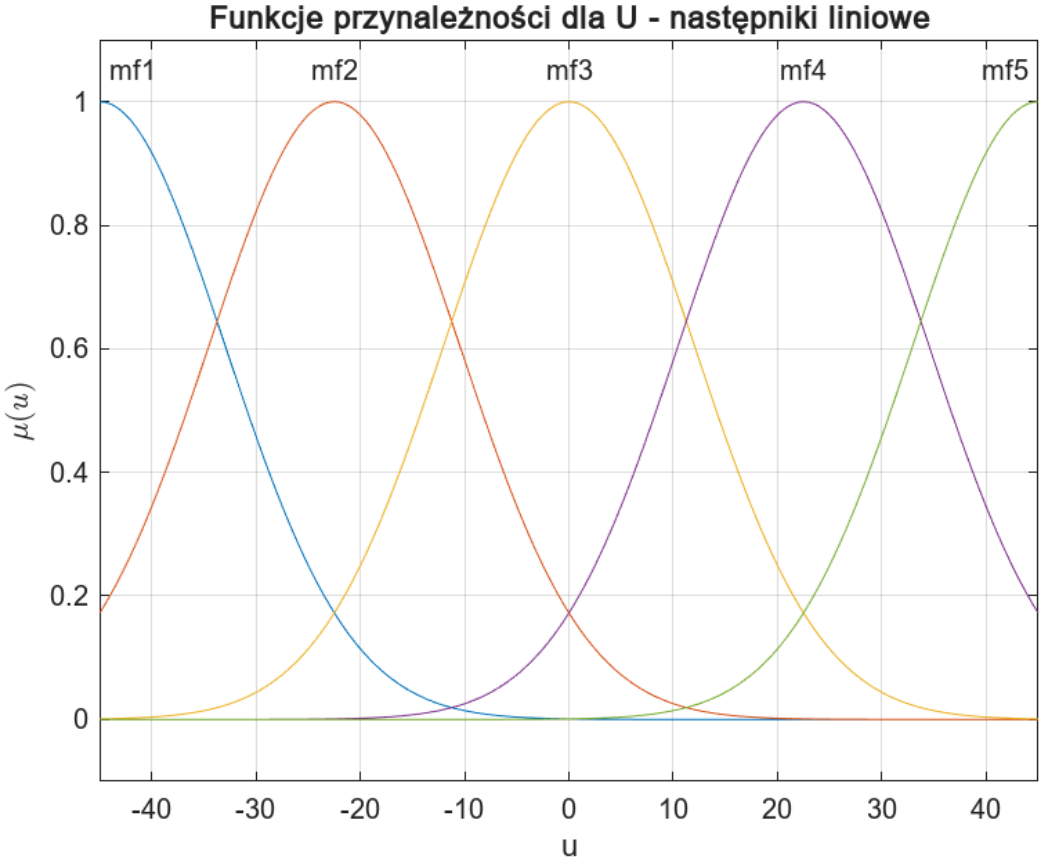
Następniki modelu liniowego były postaci:

$$\begin{aligned}
&\text{Jeśli } u \text{ jest } U^1 \text{ to } u^1_{fuzzy} = a^1 u + b^1 \\
&\text{Jeśli } u \text{ jest } U^2 \text{ to } u^2_{fuzzy} = a^2 u + b^2 \\
&\text{Jeśli } u \text{ jest } U^3 \text{ to } u^3_{fuzzy} = a^3 u + b^3 \\
&\text{Jeśli } u \text{ jest } U^4 \text{ to } u^4_{fuzzy} = a^4 u + b^4 \\
&\text{Jeśli } u \text{ jest } U^5 \text{ to } u^5_{fuzzy} = a^5 u + b^5
\end{aligned} \quad c \tag{2.1}$$

Zbiory rozmyte rozmieściłem równomiernie na całym zakresie wartości sterowania:

```
U_center = linspace(obj.U_min, obj.U_max, 5);
```

Zbadałem trzy kształty funkcji przynależności: `gaussmf`, `gbellmf` oraz `sigmf`, jednak szybko się okazało, że otrzymywane wyniki nie różnią się w sposób znaczący, a w przypadku funkcji `gaussmf` potrzeba najmniej parametrów, żeby dobrze zdefiniować taki zbiór, dlatego w przyjąłem tylko taki kształt funkcji przynależności modyfikując jedynie parametry σ oraz μ odpowiadające wariancji oraz wartości średniej. Zbiory rozmyte prezentują się następująco:



Rys. 2.1: Zbiory rozmyte - model Hammersteina z następnikami liniowymi.

2.2. Następniki nieliniowe

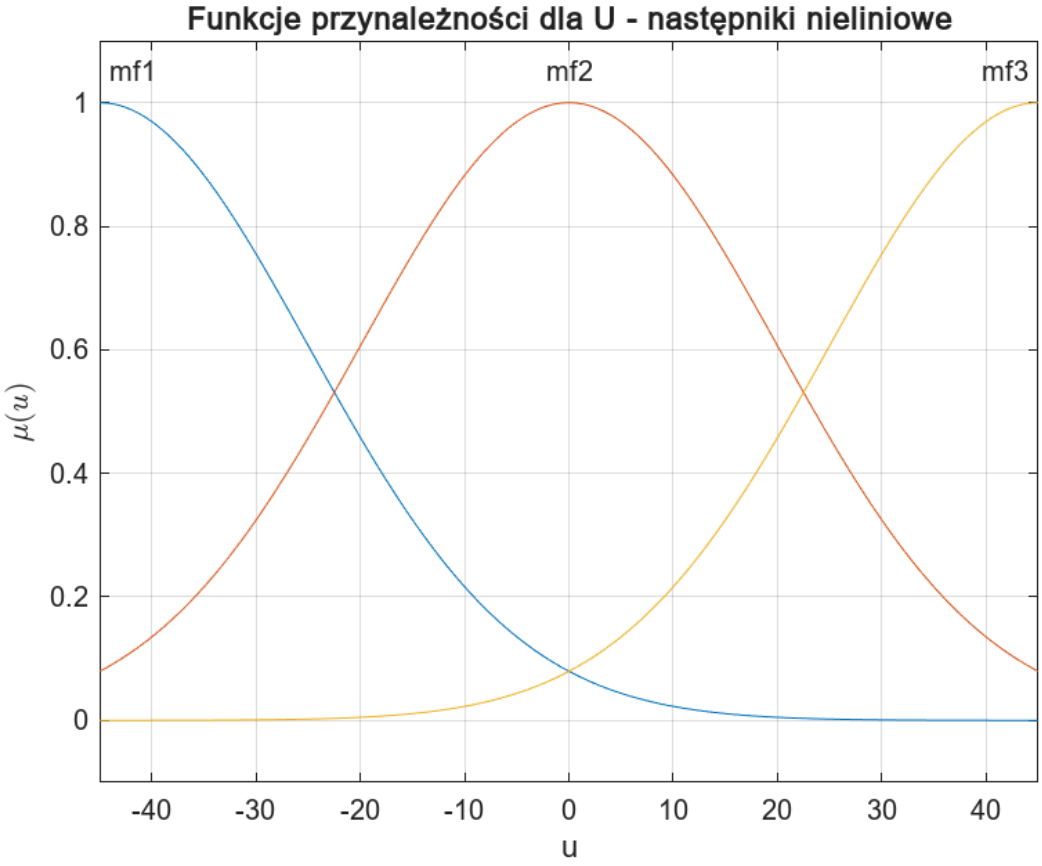
W przypadku następników hiperbolicznych zdecydowałem się na taką postać:

$$\begin{aligned} &\text{Jeśli } u \text{ jest } U^1 \text{ to } u^1_{fuzzy} = a^1 \sinh\left(\tfrac{u}{b^1}\right) \\ &\text{Jeśli } u \text{ jest } U^2 \text{ to } u^2_{fuzzy} = a^2 \sinh\left(\tfrac{u}{b^1}\right) \\ &\text{Jeśli } u \text{ jest } U^3 \text{ to } u^3_{fuzzy} = a^3 \sinh\left(\tfrac{u}{b^3}\right) \end{aligned} \tag{2.2}$$

Było to podyktowane kształtem funkcji sinh i faktowi, że funkcja ta rośnie wykładniczo. Okazało się, że kluczową rolę odgrywa tutaj parametr normalizujący, tj. b^i. Jego wartość pozwoliła dobrze modelować wartości sterowania zarówno z niskiego, średniego, jak i wysokiego przedziału.

```
function nonlinearFuzzy(obj)
U_center = linspace(obj.U_min, obj.U_max, 3);
obj.nonlinear_fis = sugfis('Name', 'Nonlinear_Hammerstein',...
                        'Type', 'sugeno');
obj.nonlinear_fis = addInput(obj.nonlinear_fis,...
                        [obj.U_min obj.U_max], 'Name', 'U_linear');

% Definiowanie funkcji przynależności (gaussmf)
for i = 1:length(U_center)
   obj.nonlinear_fis = addMF(obj.nonlinear_fis, 'U_linear',...
                        'gaussmf', [20, U_center(i)]);
end

% Definiowanie wyjścia i początkowych następników (a_i * u + b_i)
obj.nonlinear_fis = addOutput(obj.nonlinear_fis,...
                        [obj.U_min obj.U_max], 'Name', 'U_fuzzy');

% Początkowe współczynniki (a_i, b_i)
a_param = [60.5 50.7 83.5];
b_param = [80 50 75];

% Dodanie reguł TS w postaci liniowej
for i = 1:length(U_center)
   obj.nonlinear_fis = addMF(obj.nonlinear_fis, 'U_fuzzy',...
                        'linear', [a_param(i), b_param(i)]);
end

% Reguły Takagi-Sugeno: [inputMF, outputMF, weight]
ruleList = [1 1 1 1;  % Reguła 1: wejście MF1 -> wyjście Out1
            2 2 1 1;  % Reguła 2: wejście MF2 -> wyjście Out2
            3 3 1 1];  % Reguła 3: wejście MF3 -> wyjście Out3

% Dodanie reguł do systemu
obj.nonlinear_fis = addRule(obj.nonlinear_fis, ruleList);
end
```

Funkcje przynależności dla modelu Hammersteina z następnikami hiperbolicznymi przedstawiłem na Rys. 2.2.



Rys. 2.2: Zbiory rozmyte - model Hammersteina z następnikami nieliniowymi.

3. Wiener

W przypadku modelu Wienera nieliniowość dodałem na wyjściu, rozmywając sygnał wyjściowy. Procedura analogiczna, jak w przypadku modeli Hammersteina, pięć zbiorów z następnikami liniowymi oraz trzy z następnikami hiperbolicznymi.

3.1. Następniki liniowe

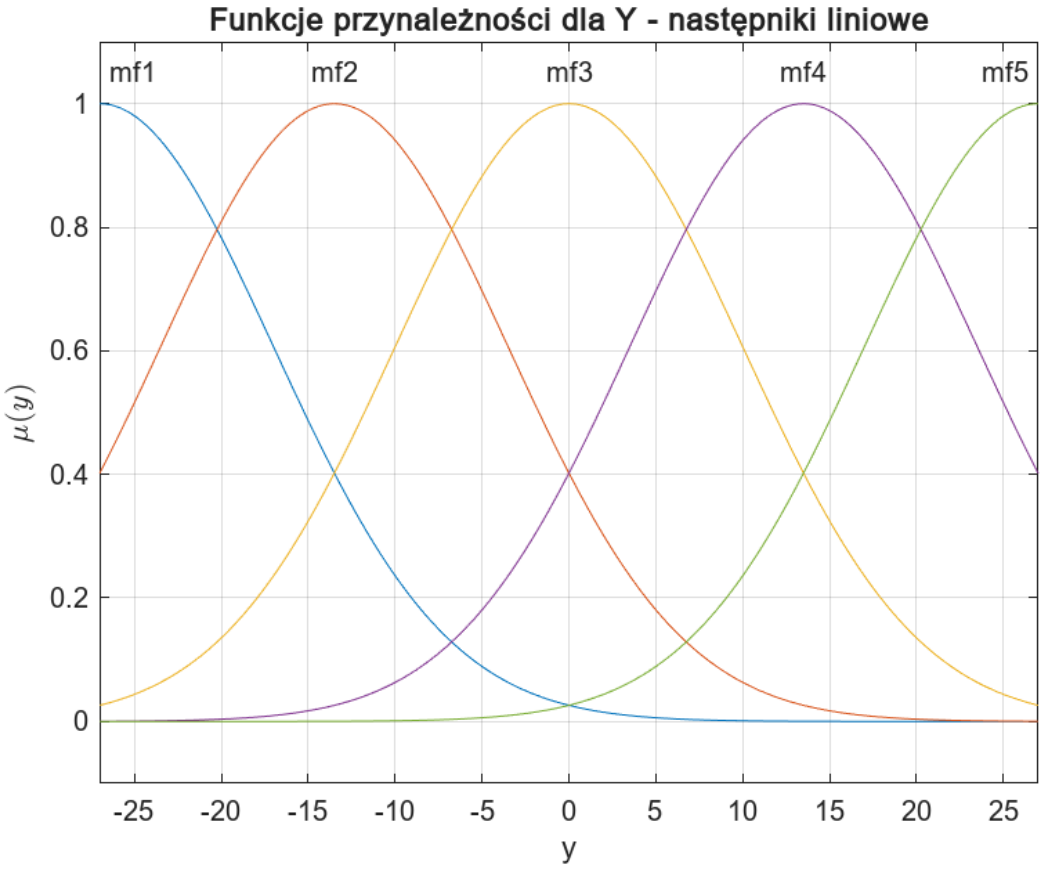
Zbudowałem system rozmyty:

```
function linearFuzzy(obj)
Y_center = linspace(obj.Y_min, obj.Y_max, 5);
obj.linear_fis = sugfis('Name', 'Linear_Wiener', 'Type', 'sugeno');
obj.linear_fis = addInput(obj.linear_fis, [obj.Y_min obj.Y_max],...
                          'Name', 'Y_linear');

% Definiowanie funkcji przynależności (gaussmf)
for i = 1:length(Y_center)
   obj.linear_fis = addMF(obj.linear_fis, 'Y_linear', 'gaussmf',...
                          [10, Y_center(i)]);
end

% Definiowanie wyjścia i początkowych następników (a_i * y + b_i)
obj.linear_fis = addOutput(obj.linear_fis, [obj.Y_min obj.Y_max],...
                          'Name', 'Y_fuzzy');

% Początkowe współczynniki (a_i, b_i)
a_param = [0.77 0.92 1 1.05 1.24]; % Można dostroić
b_param = [0 0 0 0 0];

% Dodanie reguł TS w postaci liniowej
for i = 1:length(Y_center)
   obj.linear_fis = addMF(obj.linear_fis, 'Y_fuzzy', 'linear',...
                          [a_param(i), b_param(i)]);
end

% Reguły Takagi-Sugeno: [inputMF, outputMF, weight]
ruleList = [1 1 1 1;
            2 2 1 1;
            3 3 1 1;
            4 4 1 1;
            5 5 1 1];
% Dodanie reguł do systemu
obj.linear_fis = addRule(obj.linear_fis, ruleList);
end
```

Jak widać postać następników jest taka sama, jak dla modelu Hammersteina z tą różnicą, że udało się całkowicie wyeliminować stałą b^i. Zatem ostateczna postać następników liniowych w modelu Wienera prezentuje się następująco:

$$\begin{aligned}
&\text{Jeśli } y \text{ jest } Y^1 \text{ to } y^1_{fuzzy} = a^1 y \\
&\text{Jeśli } y \text{ jest } Y^2 \text{ to } y^2_{fuzzy} = a^2 y \\
&\text{Jeśli } y \text{ jest } Y^3 \text{ to } y^3_{fuzzy} = a^3 y \\
&\text{Jeśli } y \text{ jest } Y^4 \text{ to } y^4_{fuzzy} = a^4 y \\
&\text{Jeśli } y \text{ jest } Y^5 \text{ to } y^5_{fuzzy} = a^5 y
\end{aligned} \tag{3.1}$$

Funkcje przynależności zaprezentowano na Rys. 3.1



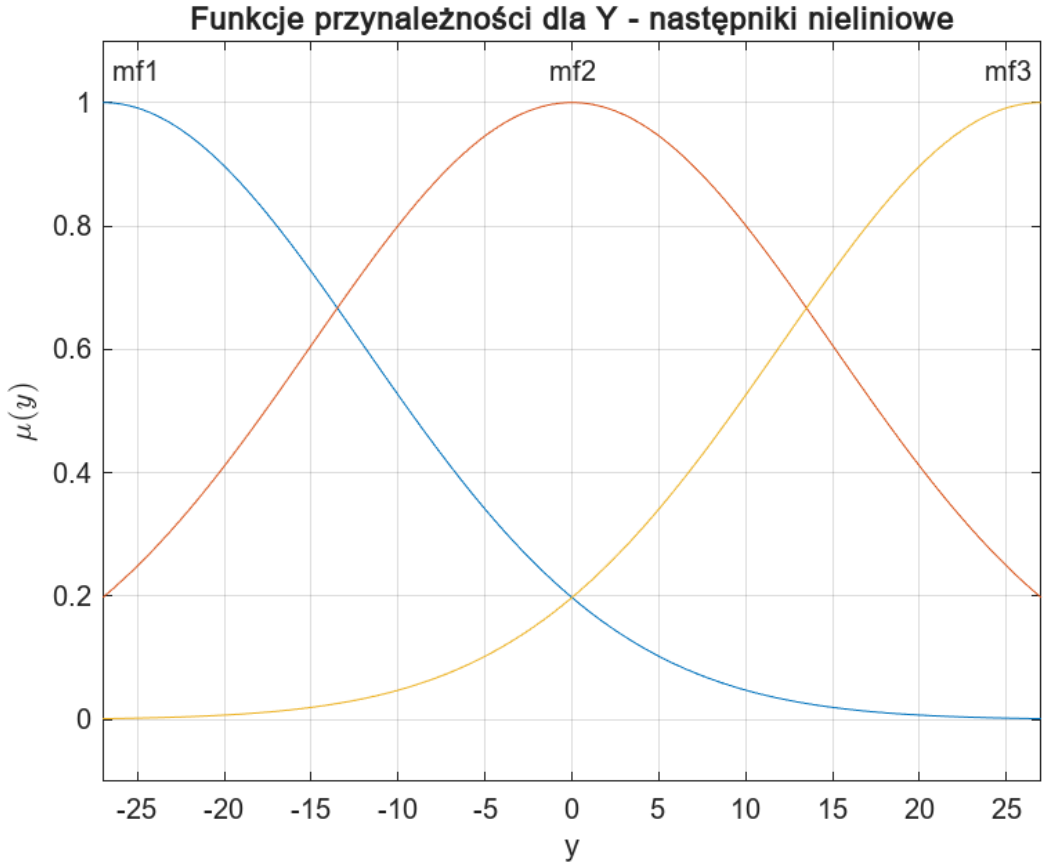
Rys. 3.1: Zbiory rozmyte - model Wienera z następnikami liniowymi.

3.2. Następniki nieliniowe

Tak jak w poprzednim przypadku, w następnikach hiperbolicznych wykorzystano funkcję sinh.

$$\begin{aligned} &\text{Jeśli } y \text{ jest } Y^1 \text{ to } y^1_{fuzzy} = a^1 \sinh\left(\tfrac{y}{b^1}\right) \\ &\text{Jeśli } y \text{ jest } Y^2 \text{ to } y^2_{fuzzy} = a^2 \sinh\left(\tfrac{y}{b^1}\right) \\ &\text{Jeśli } y \text{ jest } Y^3 \text{ to } y^3_{fuzzy} = a^3 \sinh\left(\tfrac{y}{b^3}\right) \end{aligned} \tag{3.2}$$

Po raz kolejny okazało się, że to właśnie odpowiednia normalizacja sygnału wyjściowego zapewnia satysfakcjonującą dokładność modelowania. Zbiory rozmyte dla modelu Wienera z następnikami nieliniowymi zaprezentowałem poniżej.



Rys. 3.2: Zbiory rozmyte - model Wienera z następnikami nieliniowymi.

4. Testowania modeli

Wygenerowałem losowe wartości wymuszeń z przedziału $\left[-45 \quad 45\right]$. Na potrzeby testów przyjąłem stałą wartość zakłócenia, tj.

$$F_D = F_{D0} \tag{4.1}$$

gdzie F_{D0} - wartość z punktu pracy

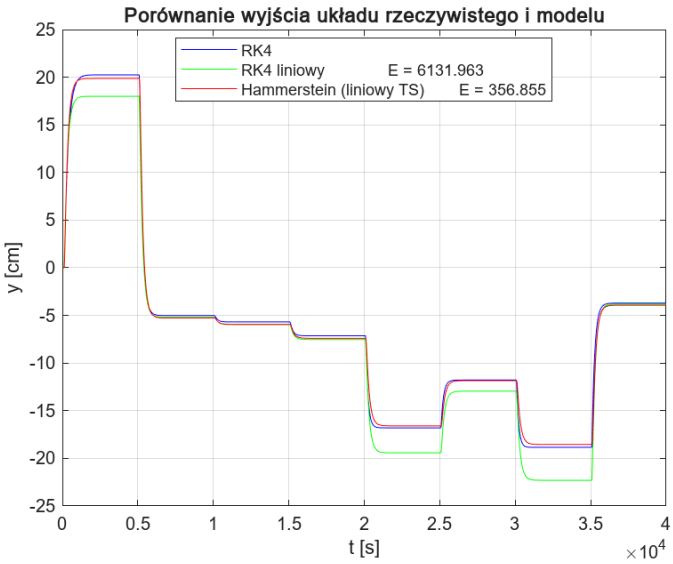
```
for i = 1:5
U = [repelem((rand(1, obiekt.kk/250) * 90 - 45), 250);
        zeros(1, obiekt.kk)];

hammerstein.testLinearModel(U, a, b, obiekt.delay, obiekt.kk,...
                        obiekt.Tp, @(x, y) obiekt.rk4(x, y), i);
hammerstein.testNonlinearModel(U, a, b, obiekt.delay, obiekt.kk,...
                        obiekt.Tp, @(x, y) obiekt.rk4(x, y), i);

wiener.testLinearModel(U, a, b, obiekt.delay, obiekt.kk,...
                        obiekt.Tp, @(x, y) obiekt.rk4(x, y), i);
wiener.testNonlinearModel(U, a, b, obiekt.delay, obiekt.kk,...
                        obiekt.Tp, @(x, y) obiekt.rk4(x, y), i);
end
```

Wykonano pięć takich testów. Udało się bez utraty dokładności zbudować model obiektu z wykorzystaniem hiperbolicznych następników w modelach Takagi - Sugeno. Rezultaty przedstawiono poniżej.

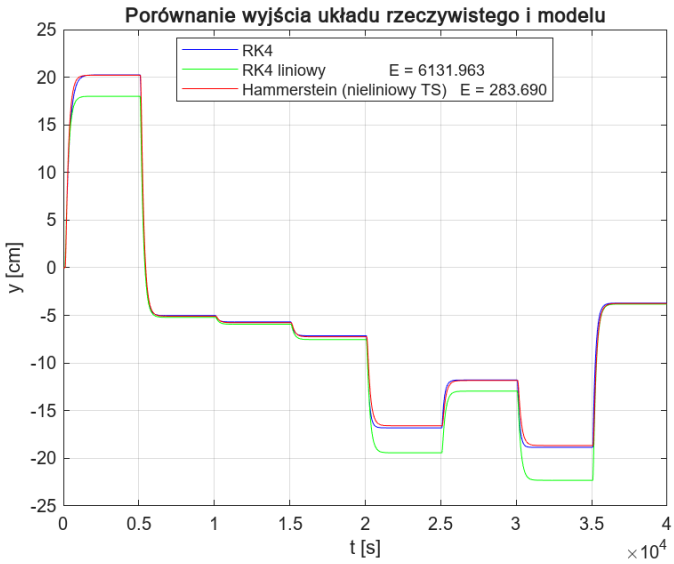
*W legendach obok opisu danego przebiegu zamieściłem wartość wskaźnika jakości w postaci błędu kwadratowego, tj.

$$E = \sum_{i=1}^{N} (y(i) - y_{mod}(i))^2 \tag{4.2}$$

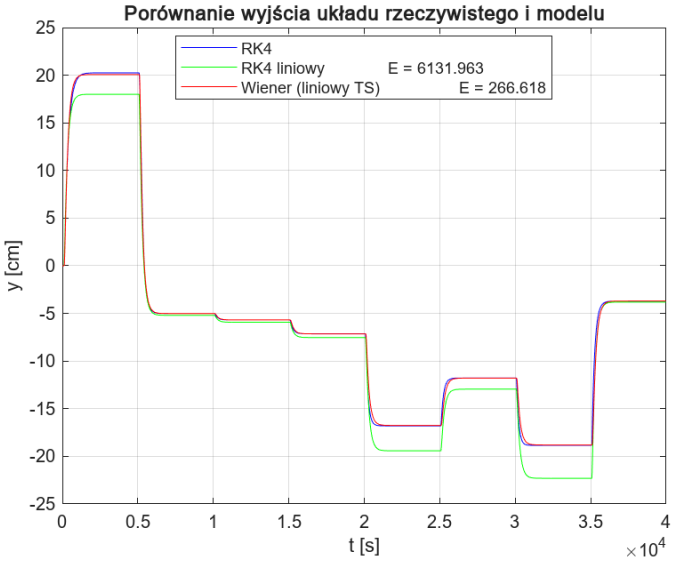
gdzie y - wartość otrzymana dla algorytmu Rungego-Kutty 4 rzędu



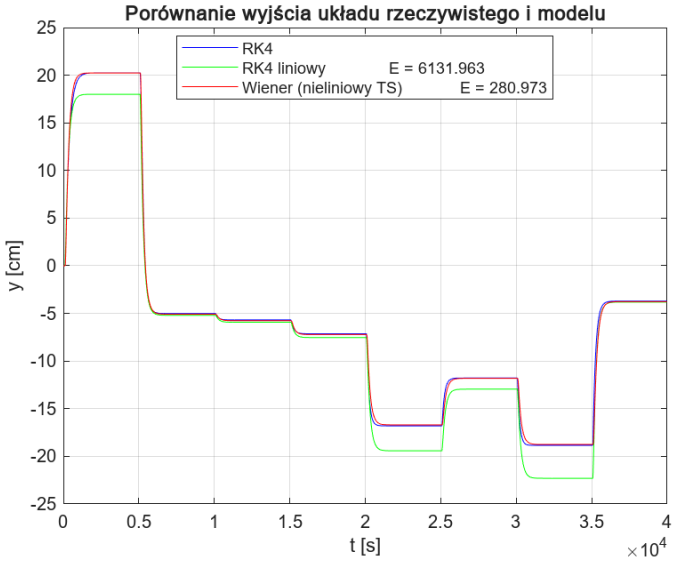
(a) Hammerstein - następniki liniowe.



(b) Hammerstein - następniki nieliniowe.

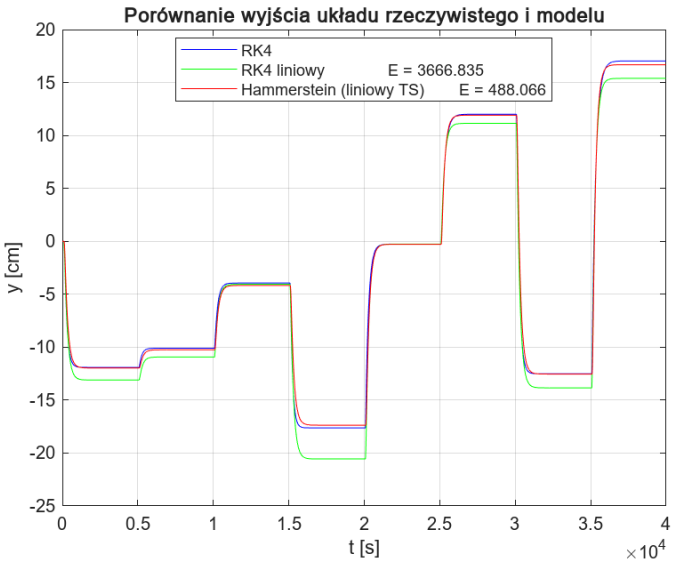


(c) Wiener - następniki liniowe.

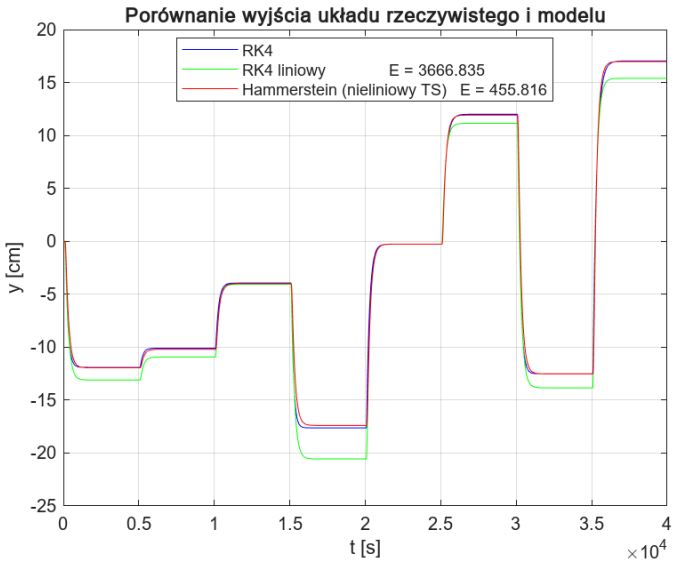


(d) Wiener - następniki nieliniowe.

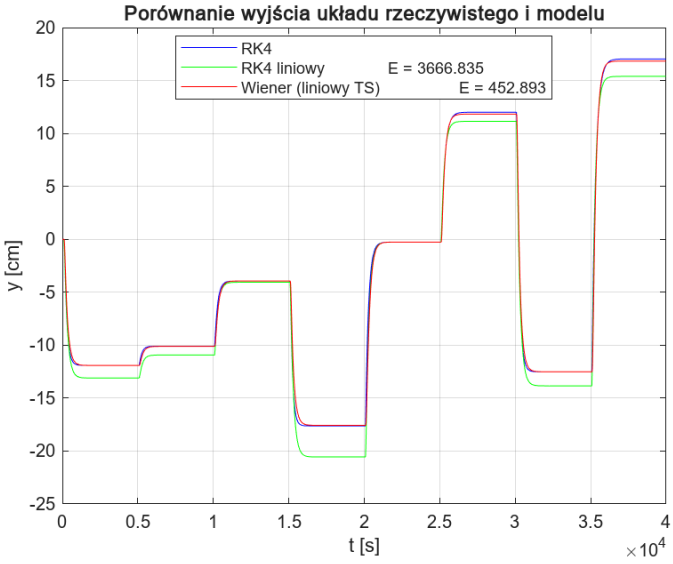
Rys. 4.1: Pierwsza seria modeli.



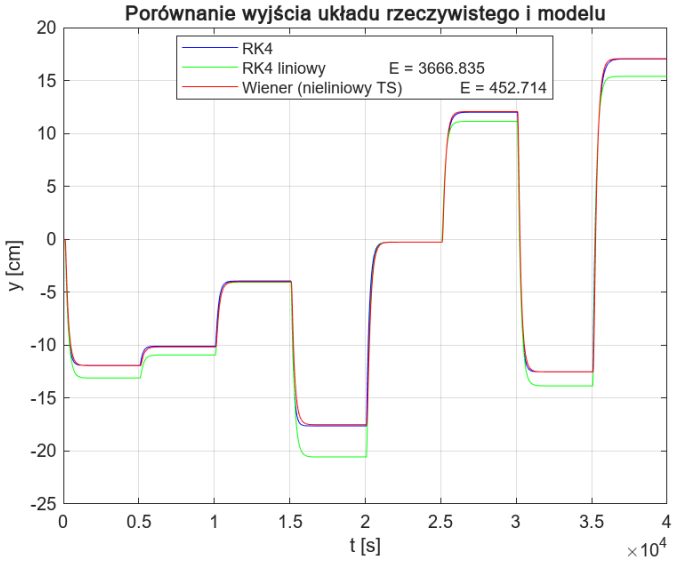
(a) Hammerstein - następniki liniowe.



(b) Hammerstein - następniki nieliniowe.

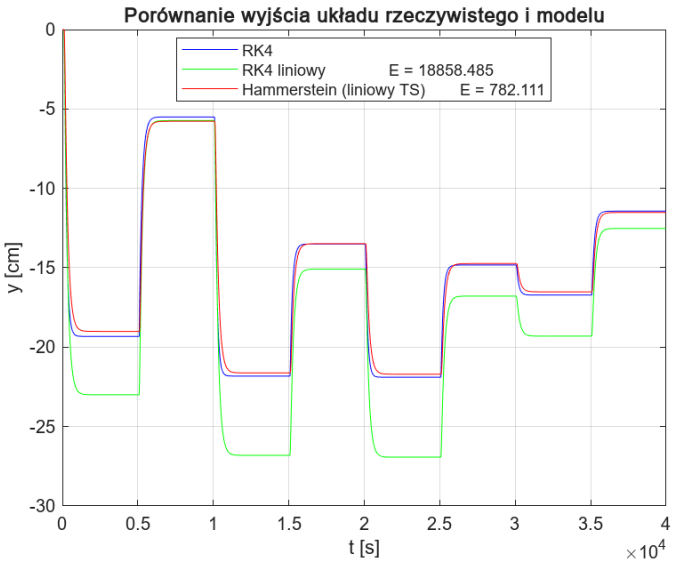


(c) Wiener - następniki liniowe.

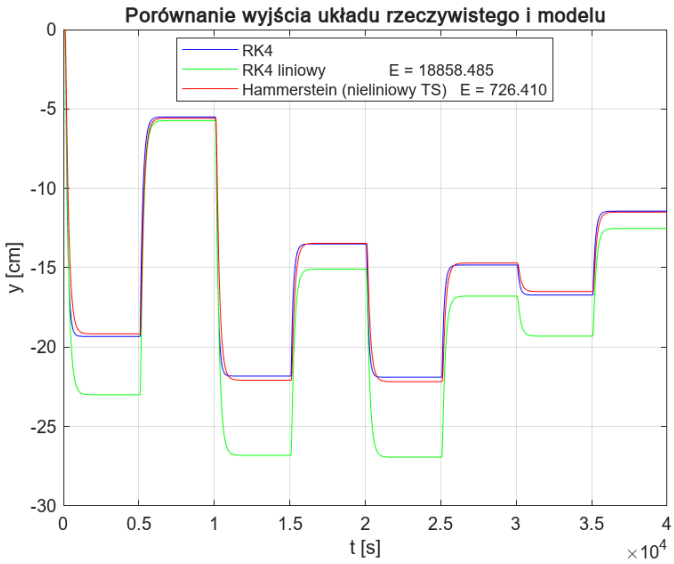


(d) Wiener - następniki nieliniowe.

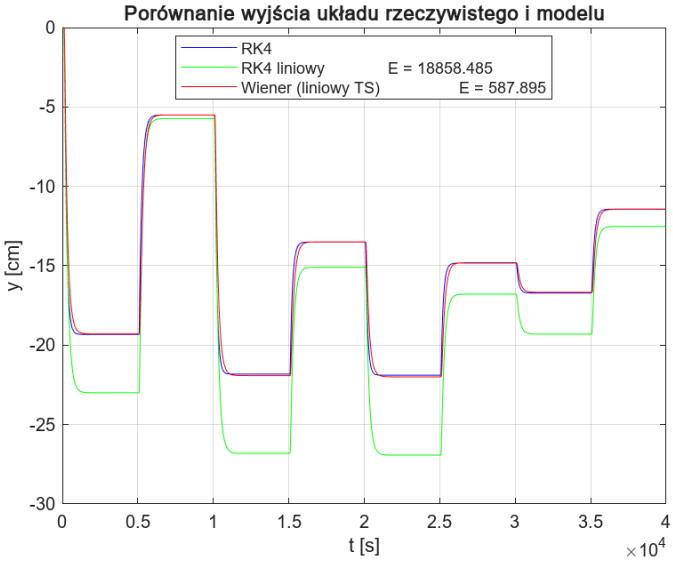
Rys. 4.2: Druga seria modeli.



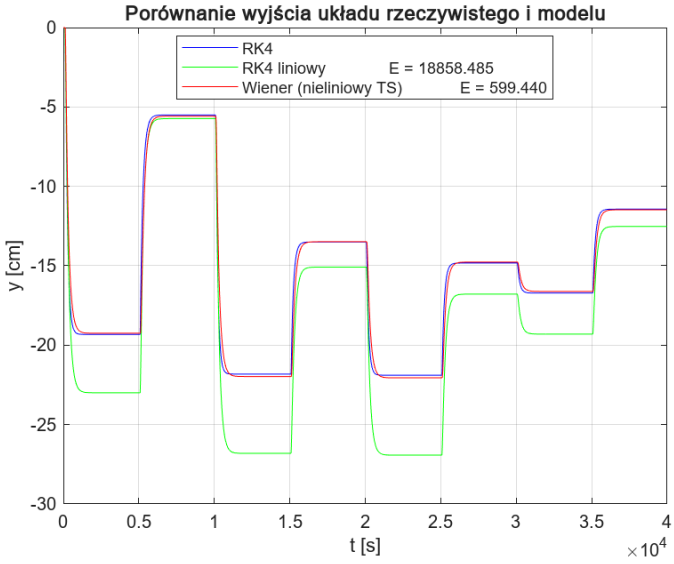
(a) Hammerstein - następniki liniowe.



(b) Hammerstein - następniki nieliniowe.

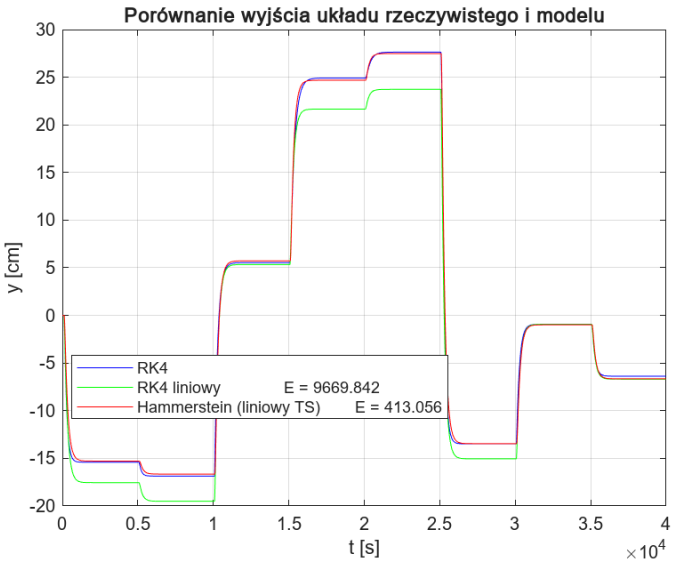


(c) Wiener - następniki liniowe.

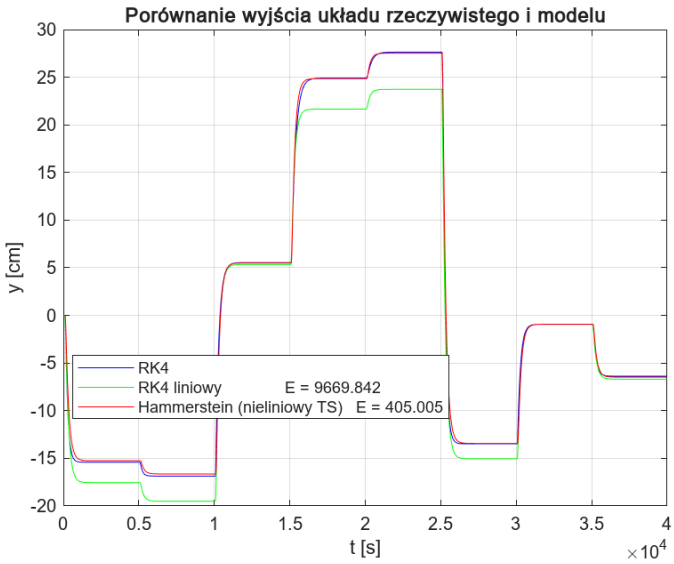


(d) Wiener - następniki nieliniowe.

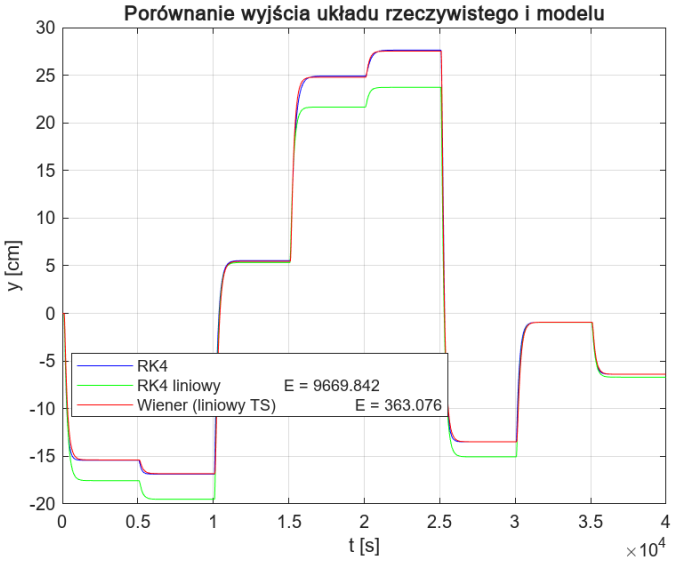
Rys. 4.3: Trzecia seria modeli.



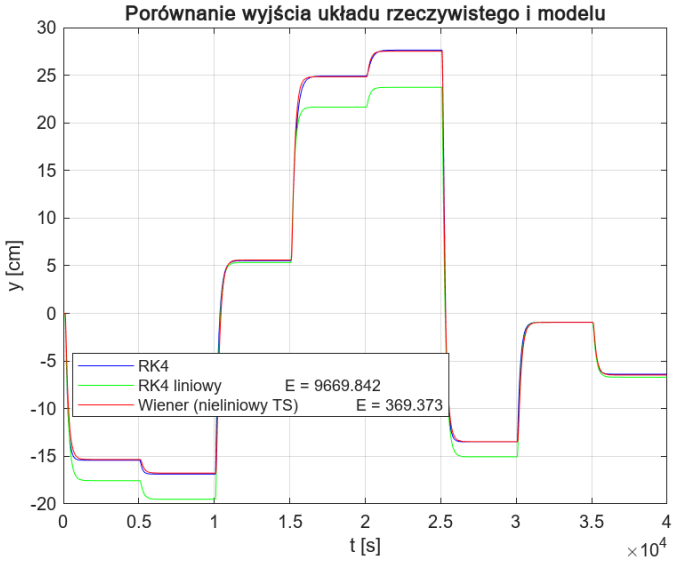
(a) Hammerstein - następniki liniowe.



(b) Hammerstein - następniki nieliniowe.

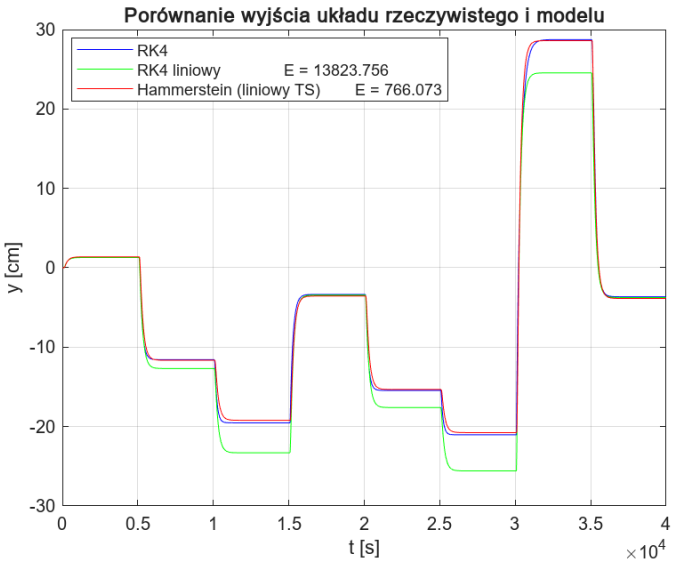


(c) Wiener - następniki liniowe.

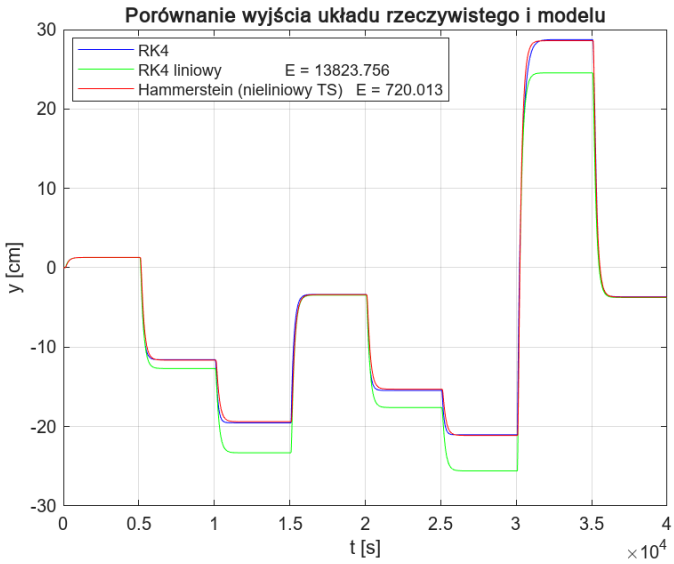


(d) Wiener - następniki nieliniowe.

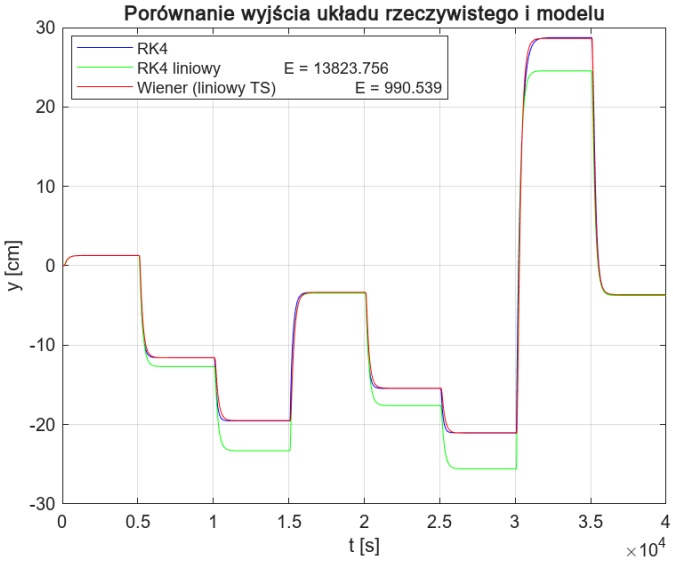
Rys. 4.4: Czwarta seria modeli.



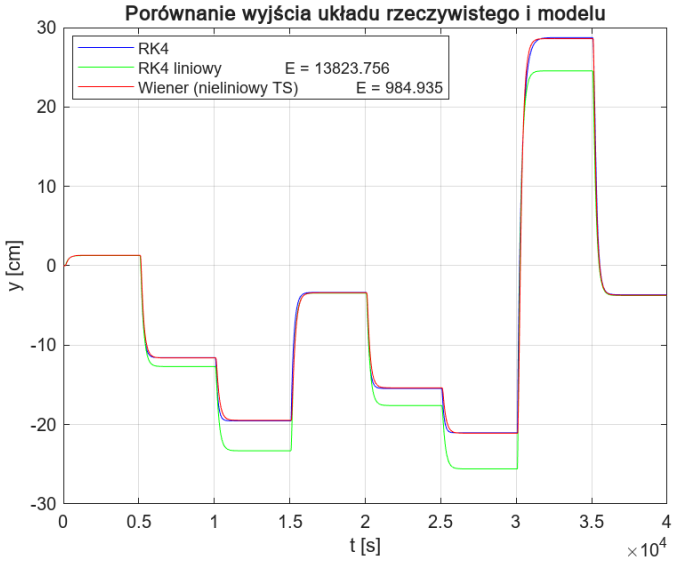
(a) Hammerstein - następniki liniowe.



(b) Hammerstein - następniki nieliniowe.



(c) Wiener - następniki liniowe.



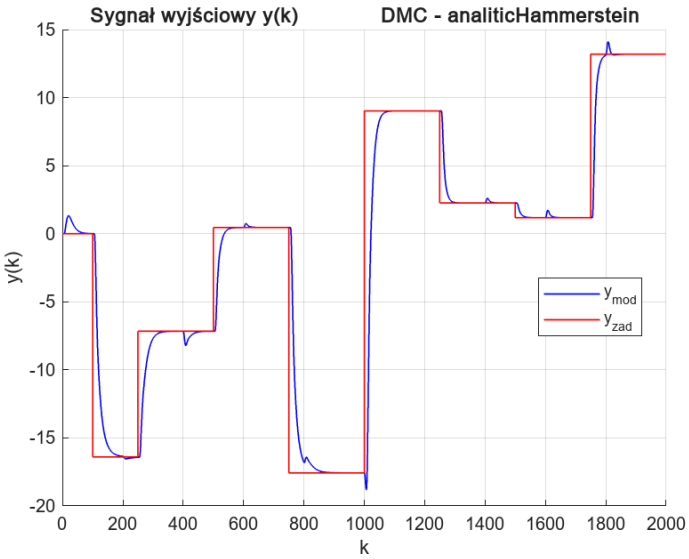
(d) Wiener - następniki nieliniowe.

Rys. 4.5: Piąta seria modeli.

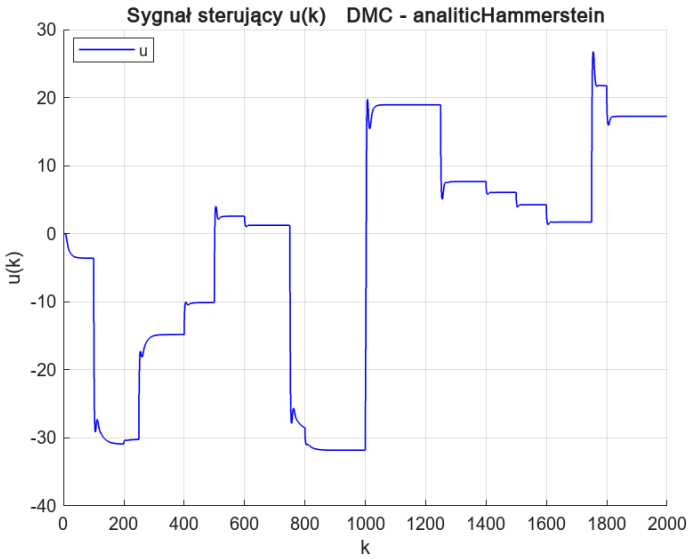
5. DMC

Zaimplementowałem pięć regulatorów, tj. analityczny, numeryczny, z sukcesywną linearyzacją, nieliniową predykcją i linearyzacją oraz regulator rozmyty. Pominąłem regulator z nieliniową optymalizacją, ze względu trudność jego praktycznego zastosowania - niemożliwy do oszacowania czas znalezienia minimum. Wygenerowano sekwencję sygnału zadanego, taką samą dla każdego z regulatorów. W każdym z regulatorów uwzględniłem pomiar zakłócenia.

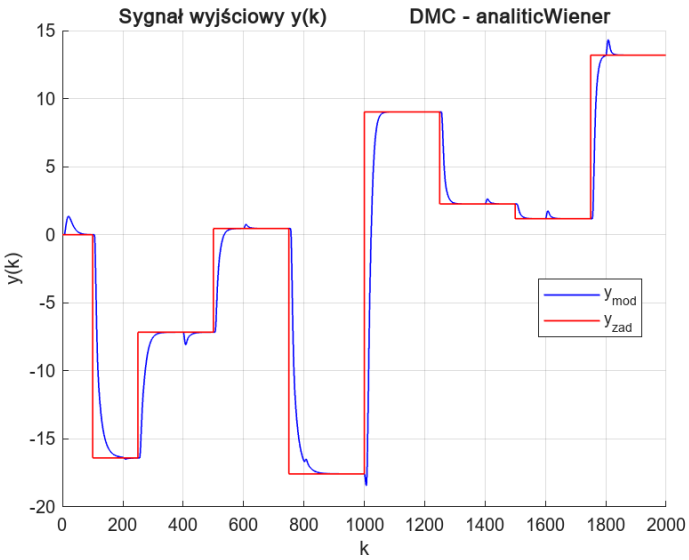
```
%% DMC(N, Nu, D, D_disturbance, lambda)
dmc = DMC(100, 2, 150, 150, 1);
% Pierwsza iteracja jest wspólna dla wszystkich algorytmów DMC
dmc.dynamic_matrix(s);
dmc.past_matrix(s);
dmc.matrix_disturbance(s_disturbance);

% Test
y_zad = [repelem((rand(1, obiekt.kk/250) * 40 - 20), 250)];
y_zad(1:100) = 0;
u_D = [repelem((rand(1, obiekt.kk/200) * 10 - 5), 200)];

pool = gcp();
f.analiticHammerstein = parfeval(pool, ...);
f.analiticWiener = parfeval(pool, ...);
f.numericHammerstein = parfeval(pool, ...);
f.numericWiener = parfeval(pool, ...);
f.slHammerstein = parfeval(pool, ...);
f.slWiener = parfeval(pool, ...);
f.nplHammerstein = parfeval(pool, ...);
f.nplWiener = parfeval(pool, ...);
f.fuzzyHammerstein = parfeval(pool, ...);
f.fuzzyWiener = parfeval(pool, ...);

fprintf("\nFunkcje uruchomione asynchronicznie...\n\n");
```



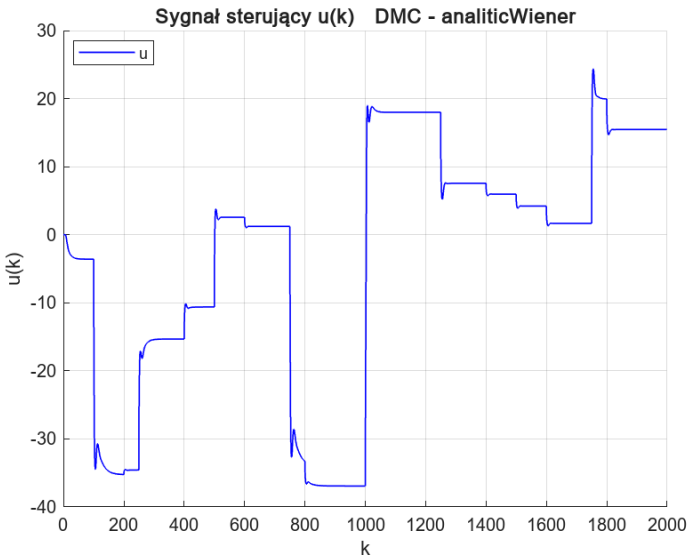
(a) Hammerstein - sygnał wyjściowy $y(k)$.



(b) Hammerstein - sygnał sterujący $u(k)$.

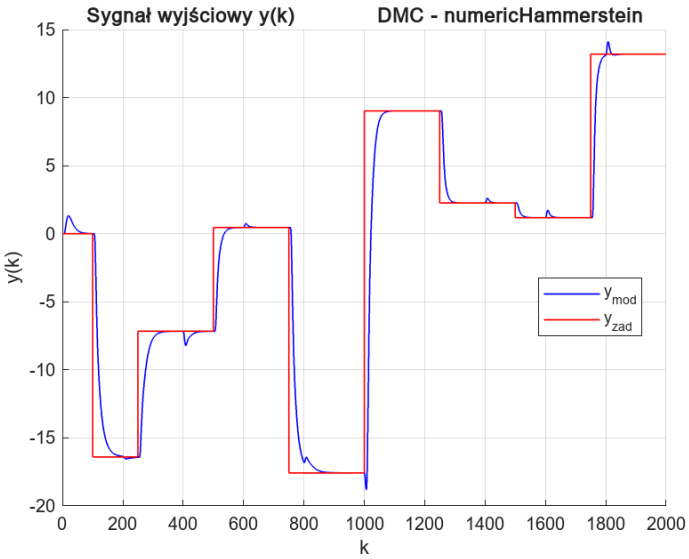


(c) Wiener - sygnał wyjściowy $y(k)$.

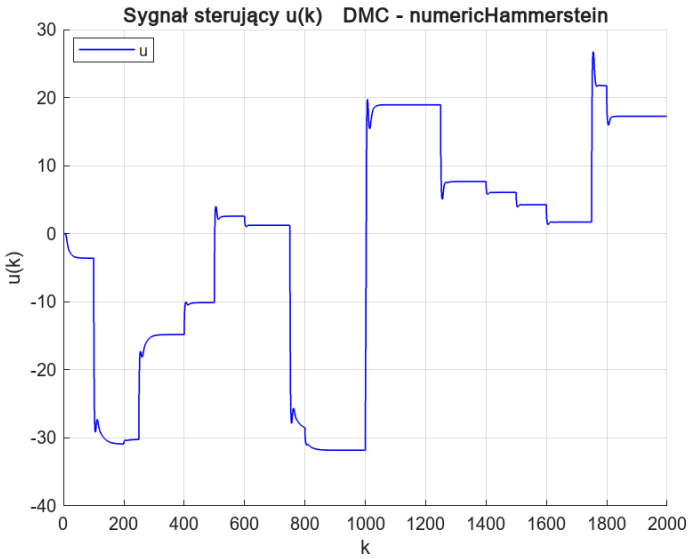


(d) Wiener - sygnał sterujący $u(k)$.

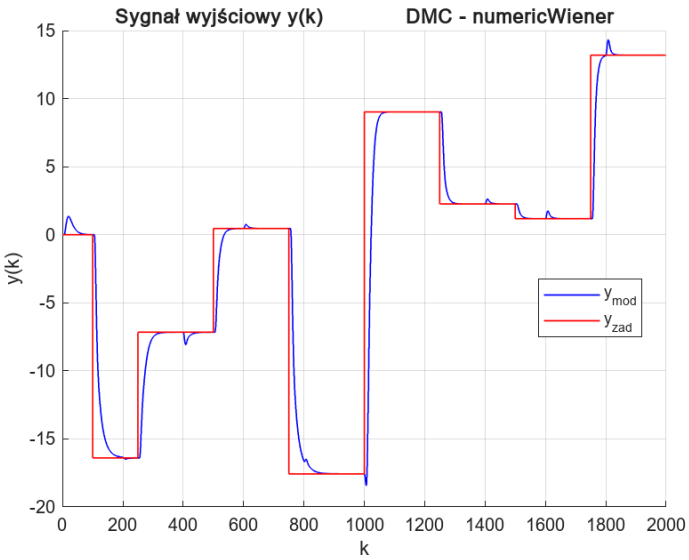
Rys. 5.1: Regulator analityczny.



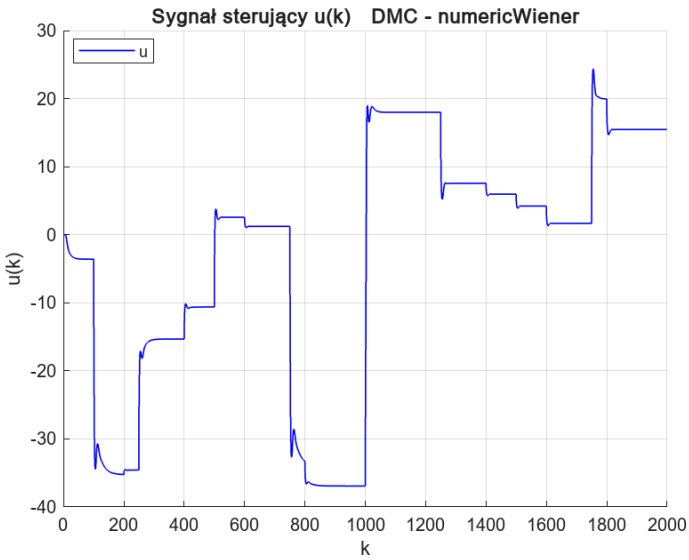
(a) Hammerstein - sygnał wyjściowy $y(k)$.



(b) Hammerstein - sygnał sterujący $u(k)$.

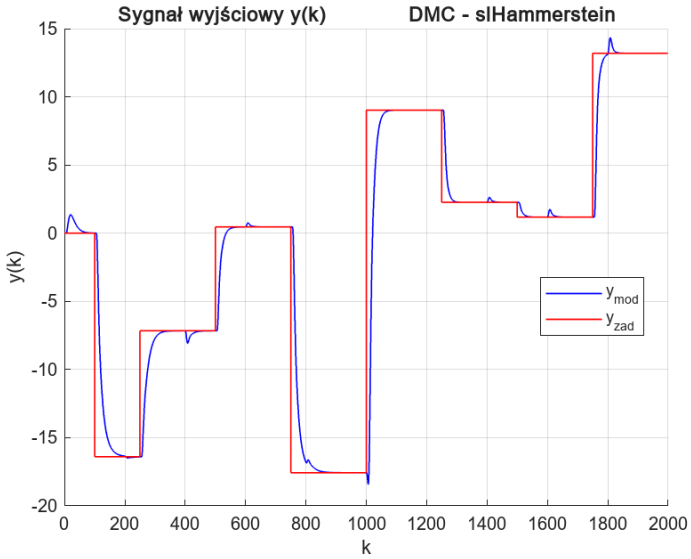


(c) Wiener - sygnał wyjściowy $y(k)$.

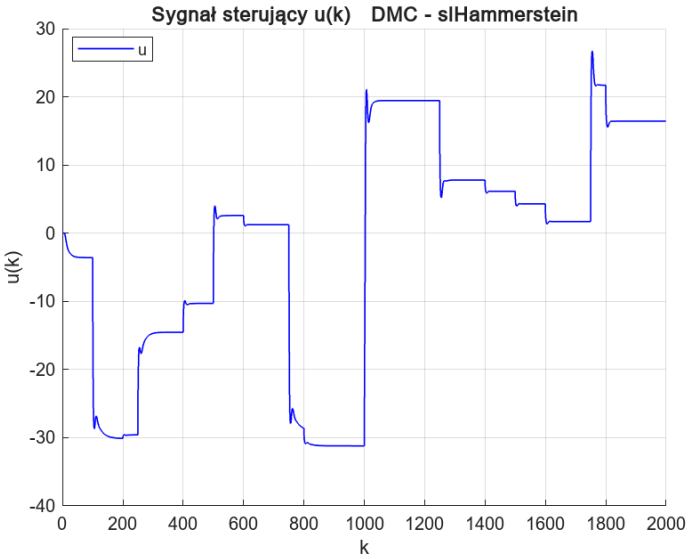


(d) Wiener - sygnał sterujący $u(k)$.

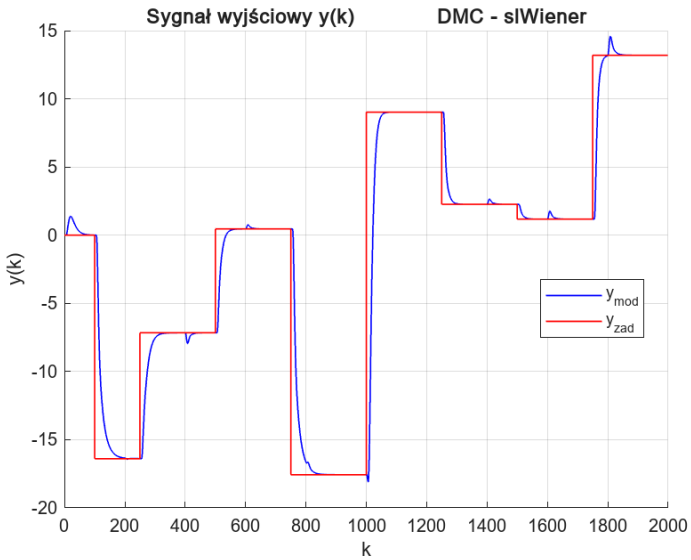
Rys. 5.2: Regulator numeryczny.



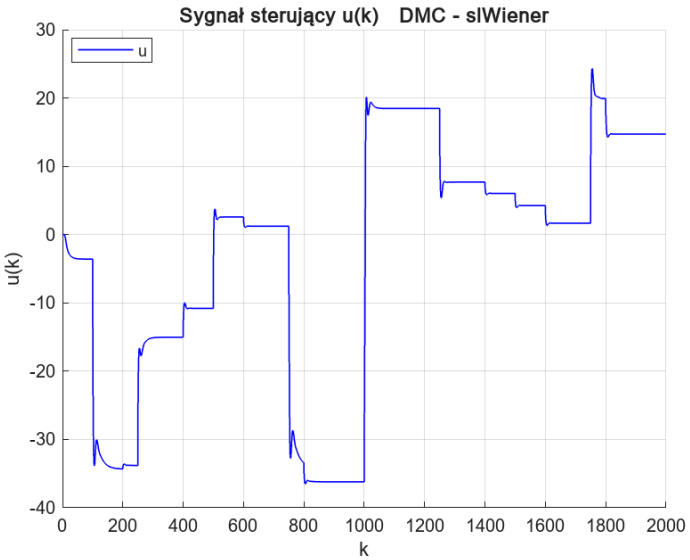
(a) Hammerstein - sygnał wyjściowy $y(k)$.



(b) Hammerstein - sygnał sterujący $u(k)$.

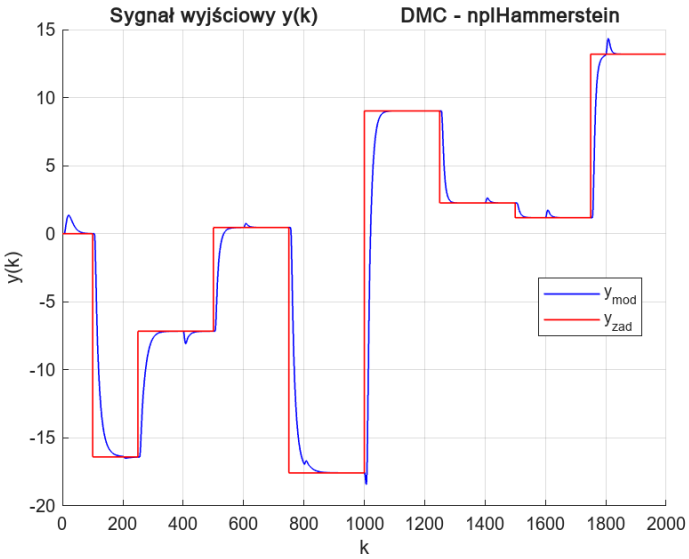


(c) Wiener - sygnał wyjściowy $y(k)$.

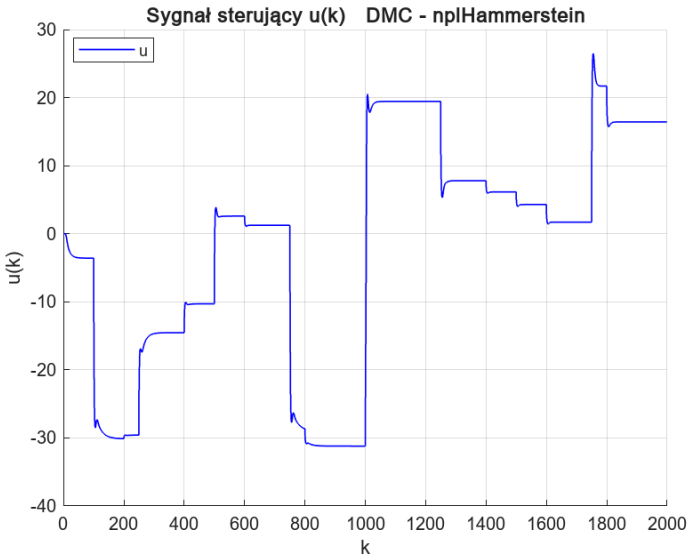


(d) Wiener - sygnał sterujący $u(k)$.

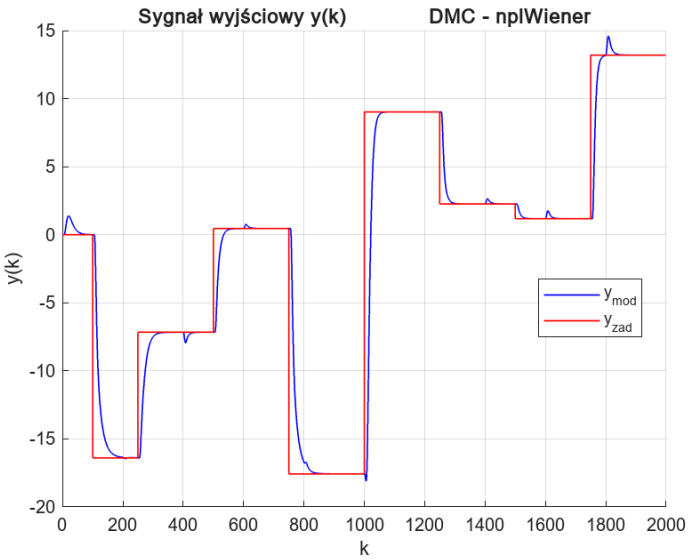
Rys. 5.3: Regulator DMC-SL.



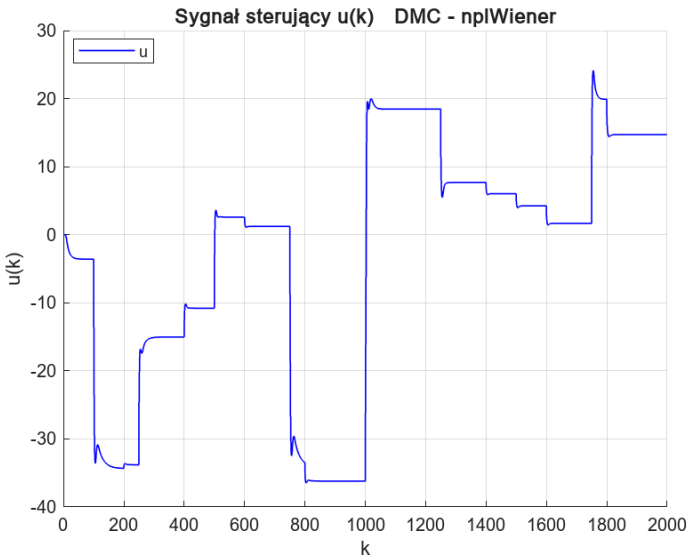
(a) Hammerstein - sygnał wyjściowy $y(k)$.



(b) Hammerstein - sygnał sterujący $u(k)$.

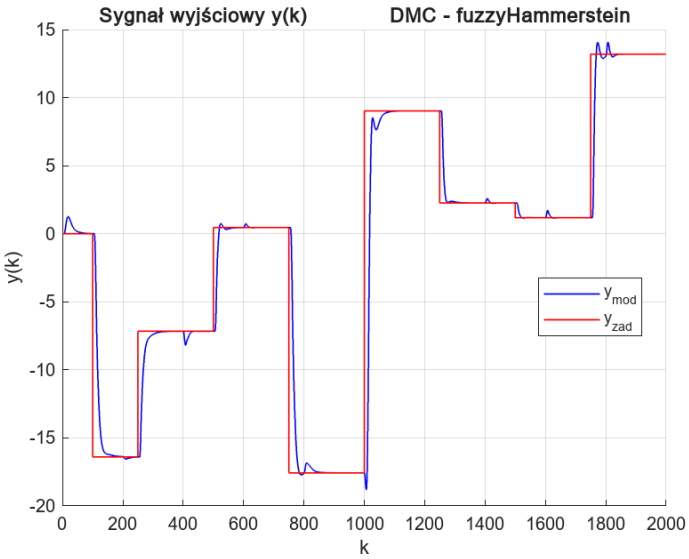


(c) Wiener - sygnał wyjściowy $y(k)$.

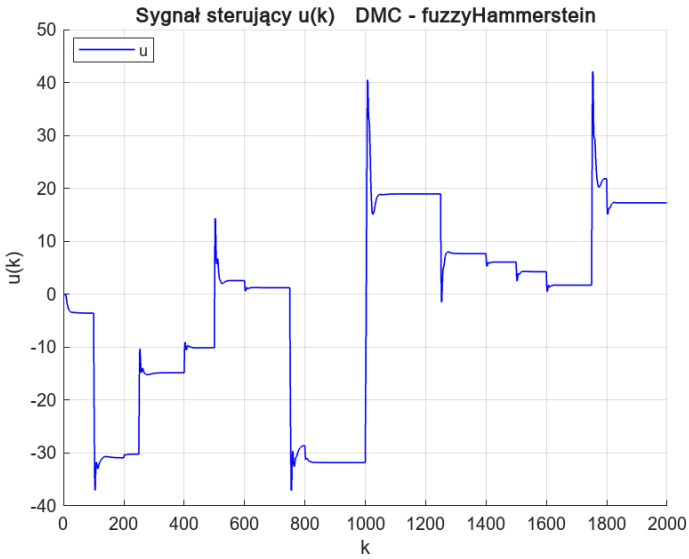


(d) Wiener - sygnał sterujący $u(k)$.

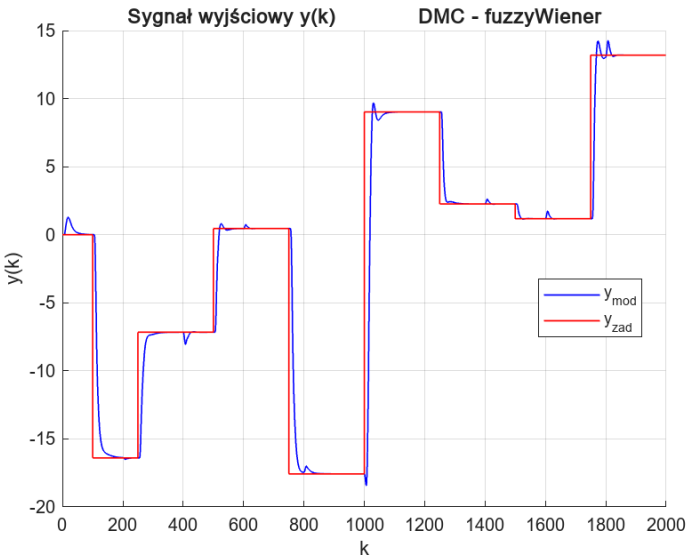
Rys. 5.4: Regulator DMC-NPL.



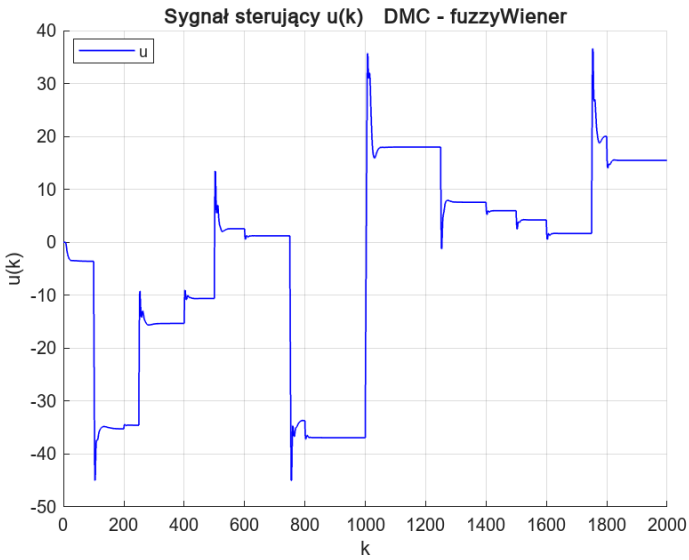
(a) Hammerstein - sygnał wyjściowy $y(k)$.



(b) Hammerstein - sygnał sterujący $u(k)$.



(c) Wiener - sygnał wyjściowy $y(k)$.



(d) Wiener - sygnał sterujący $u(k)$.

Rys. 5.5: Regulator FDMC.

Wskaźnik jakości użyty do określenia jakości wyznaczonego sterowania był postaci:

$$J = \sum_{i=1}^{N}(y_{zad}(i) - y(i))^2 + \lambda \sum_{i=1}^{N} \Delta u(i)^2 \tag{5.1}$$

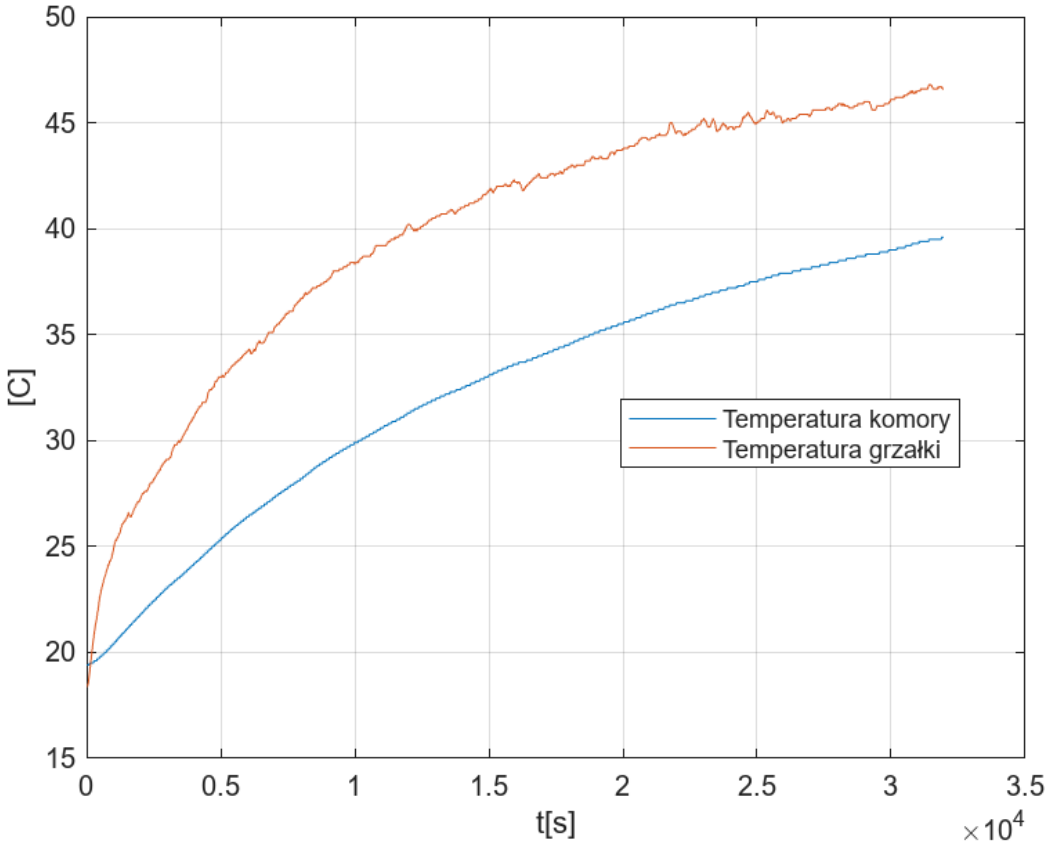
Zatem uwzględniłem zarówno uchyby regulacji, jak i zmiany sygnału sterującego. Wartości wskaźnika zamieściłem w tabeli poniżej.

Regulator	Hammerstein	Wiener
DMC - analityczny	25214.753	25130.851
DMC - numeryczny	25214.753	25130.851
DMC - SL	24605.791	24568.536
DMC - NPL	24490.601	24464.042
FDMC	23555.099	23636.443

Tab. 5.1: Porównanie regulatorów.

Zgodnie z wiedzą teoretyczną, bardziej złożone regulatory pozwoliły z większą dokładnością odwzorować sygnał zadany. **Zastanawiam się nad faktem, że regulatory analityczne i numeryczne dały dokładnie te same wyniki oraz przebiegi sygnałów wyjściowego i sterującego.**

6. Odpowiedź skokowa obiektu rzeczywistego

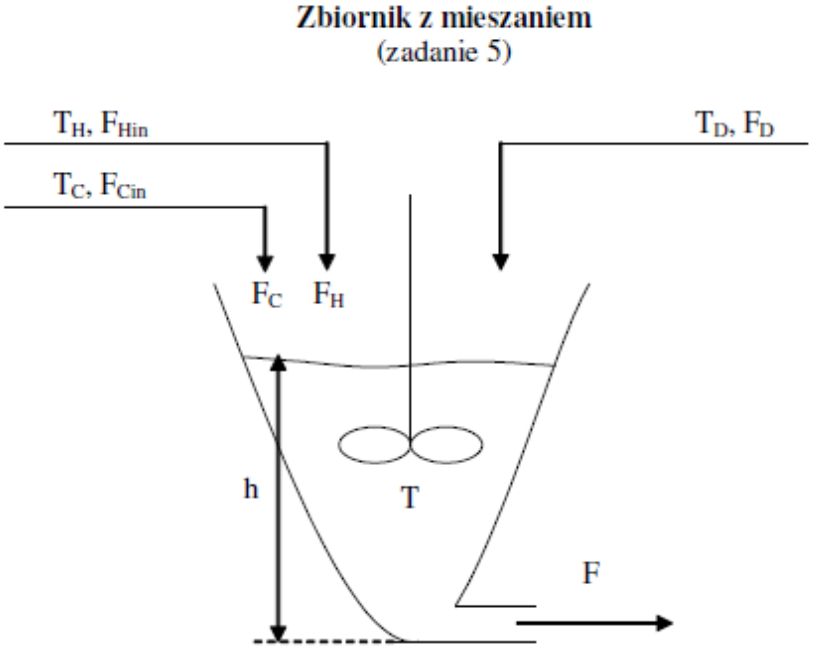


Rys. 6.1: Odpowiedź skokowa obiektu rzeczywistego.

Identyfikowanym obiektem była komora, która ogrzewana jest grzałką, która z kolei jest sterowana sygnałem PWM. Na Rys. 6.1 przedstawiłem odpowiedź układu na wymuszenie w postaci 5% wypełnienia sygnału PWM. Wymuszenie było tak małe ponieważ wcześniej wykonałem test na 10% wymuszeniu i grzałka osiągnęła ok. 120^oC (dostałem pozwolenie na testowanie do 150^oC), a na wykresie nie było widać wypłaszczenia temperatury, więc byłem zmuszony przerwać testy. Zastanawiam się jak rozwiązać ten problem, ponieważ steruję sygnałem PWM, a wejściem jest temperatura, jak skorelować ograniczenia temperaturowe z sygnałem PWM? Jednocześnie nie mogę ograniczyć sygnału sterującego w ramach +5, -5, bo regulator będzie bardzo wolny. Prosiłbym o wskazówki.

Postaram się ponowić testy, na mniejszej komorze, ponieważ obecnie testowana jest względnie duża i bardzo długo się nagrzewa i chłodzi (skończyłem testy o 21.00 i temperatura wskazywała ok. 40^oC, rano zachodząc do pracy o 7.30 temperatura utrzymywała się na poziomie ok. 33^oC).

Zastanawiałem się również co w przypadku gdyby obiekt rzeczywisty był jednak liniowy. Na przedmiocie TAP realizowaliśmy projekt, którego tematem był obiekt MISO o silnych nieliniowościach.



Mieszanie wody gorącej (T_H, F_H) z zimną (T_C, F_C) z dopływem zakłócającym (T_D, F_D).

$$\begin{cases} \dfrac{dV}{dt} = F_H + F_C + F_D - F(h) \\ V\dfrac{dT}{dt} = F_H \cdot T_H + F_C \cdot T_C + F_D \cdot T_D - F(h) \cdot T \\ F(h) = \alpha\sqrt{h}, \quad V(h) = C \cdot h^2, \quad F_H(t) = F_{Hin}(t - \tau_H), \quad F_C(t) = F_{Cin}(t - \tau_C) \end{cases}$$

Stałe:

$C = 0{,}7$, $\alpha = 18$;

Punkt pracy:

$T_C = 27$ °C, $T_H = 79$ °C, $T_D = 38$ °C,

$F_C = 53$ cm^3/s, $F_H = 17$ cm^3/s, $F_D = 15$ cm^3/s,

$\tau_C = 30$ s, $\tau_H = 20$ s, $h = 22{,}30$ cm, T = 39,34 °C;

Wielkości regulowane: h, T;

Wielkości sterujące: F_{Hin}, F_{Cin}.

Rys. 6.2: Obiekt MISO.

Czy w razie gdyby obiekt rzeczywisty okazał się liniowy, jako alternatywa mógłbym zamodelować obiekt z zadania czy jednak wybierzemy jeszcze bardziej złożony?s

Spis rysunków

2.1 Zbiory rozmyte - model Hammersteina z następnikami liniowymi. 5
2.2 Zbiory rozmyte - model Hammersteina z następnikami nieliniowymi. 7

3.1 Zbiory rozmyte - model Wienera z następnikami liniowymi. 9
3.2 Zbiory rozmyte - model Wienera z następnikami nieliniowymi. 10

4.1 Pierwsza seria modeli. 12
4.2 Druga seria modeli. 13
4.3 Trzecia seria modeli. 14
4.4 Czwarta seria modeli. 15
4.5 Piąta seria modeli. 16

5.1 Regulator analityczny. 18
5.2 Regulator numeryczny. 19
5.3 Regulator DMC-SL. 20
5.4 Regulator DMC-NPL. 21
5.5 Regulator FDMC. 22

6.1 Odpowiedź skokowa obiektu rzeczywistego. 24
6.2 Obiekt MISO. 25

Spis tabel

5.1 Porównanie regulatorów. 23