

**Wydział Elektroniki i Technik Informacyjnych
Politechnika Warszawska**

Sztuczna inteligencja w automatyce

Sprawozdanie z projektu nr 1, zadanie DDD

Krystian Piszczela, Władysław Młynik

Warszawa, 2023

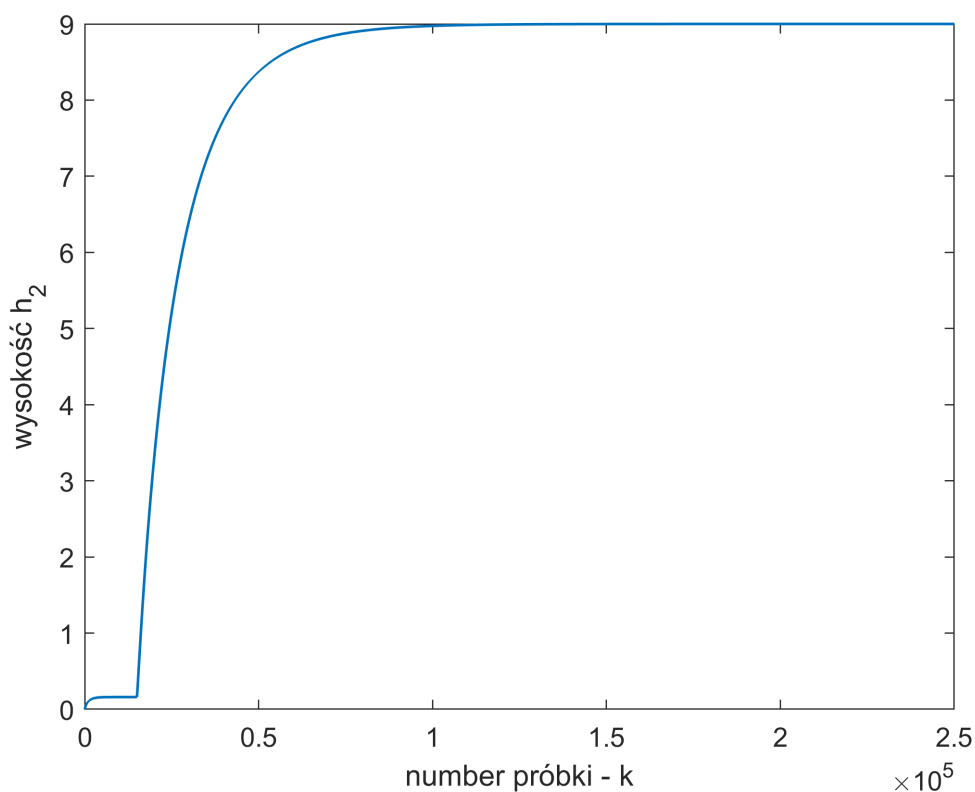
Spis treści

1. Zadanie pierwsze	2
1.1. Symulacja działania obiektu	2
1.2. Tworzenie modelu liniowego	2
1.3. Porównanie działania modelu liniowego z działaniem modelu nieliniowego	4
1.4. Nerozmyty regulator DMC	5
2. Zadanie drugie	7
2.1. Rozmyte modele Takagi-Sugeno	7
2.2. Poprawa wybranego modelu rozmytego za pomocą manipulacji kształtem funkcji przynależności	10
2.3. Projektowanie rozmytego algorytmu regulacji predykcyjnej w wersji analitycznej	12
2.4. Porównanie regulatora DMC w wersji rozmytej do regulatora DMC w wersji podstawowej (z zadania pierwszego)	12
2.5. Eksperymenty ze zmianą kształtu funkcji przynależności w regulatorze	14
3. Zadanie trzecie	18
3.1. Algorytm regulacji predykcyjnej typu SL	18
3.2. Porównanie algorytmu regulacji predykcyjnej typu SL z poprzednim wariantem	19
4. Zadanie czwarte	21
4.1. Uzupełnienie nerozmytego algorytmu DMC o mechanizm uwzględniania pomiaru zakłócenia	21

1. Zadanie pierwsze

1.1. Symulacja działania obiektu

Implementacja symulacji działania obiektu jest dostępna w pliku `step.m`. Wynik działania tego programu został przedstawiony na rys. 1.1. Jest to sprawdzenie poprawności wartości h_2 podanej w punkcie pracy, czyli $F_1 = 52 \text{ cm}^3/\text{s}$, $F_D = 8 \text{ cm}^3/\text{s}$, $\tau = 150 \text{ s}$. Jak można odczytać z rys. 1.1, nasz obiekt stabilizuje się na poziomie $h_2 = 9 \text{ cm}$. Oznacza to, iż symulacja jest zgodna z oczekiwaniami. Przy uruchamianiu przyjęty został czas próbkowania $T = 0,01 \text{ s}$.

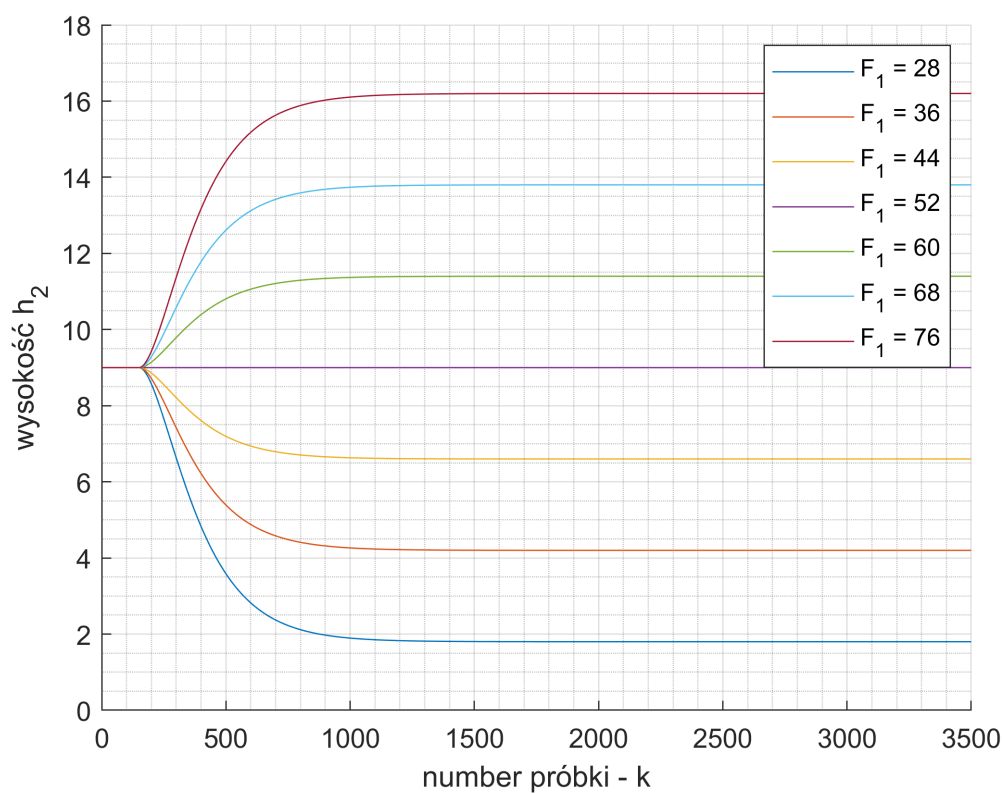


Rys. 1.1. Przebieg wysokości h_2 w punkcie pracy

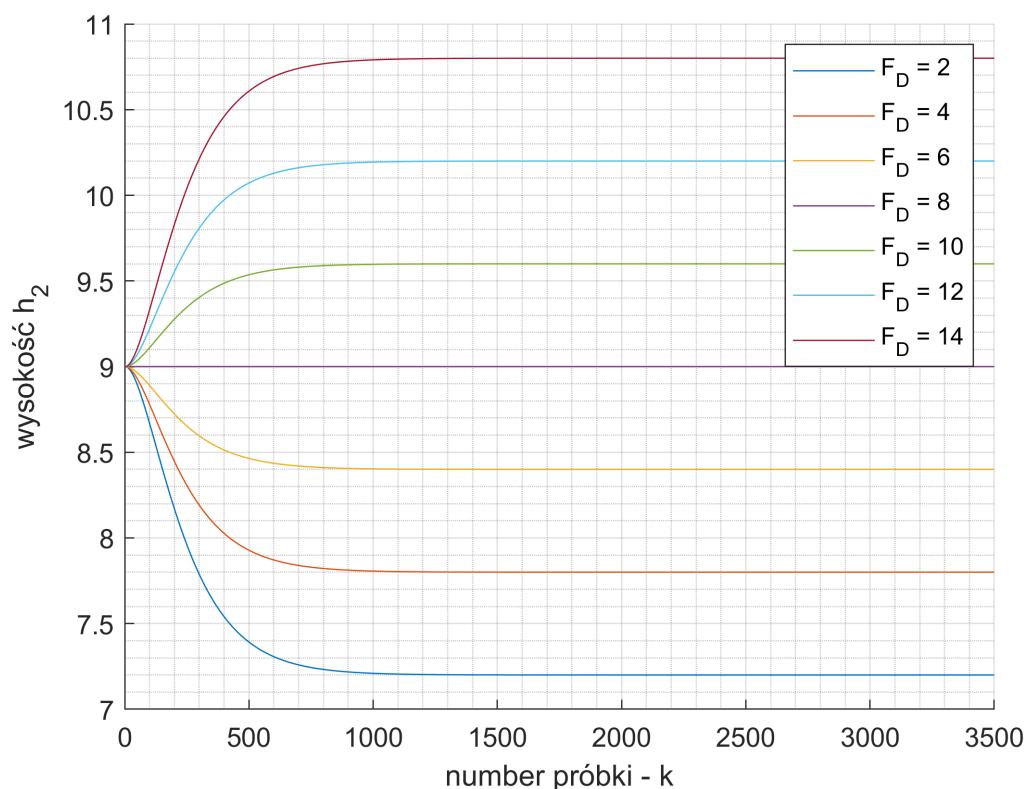
1.2. Tworzenie modelu liniowego

Na potrzeby znalezienia modelu liniowego stworzony został plik `step_lin.m`. Pozwala on na otrzymanie wykresu przedstawiającego odpowiedź skokową zlinearyzowanego modelu na skok wartości sterowania (rys. 1.2) oraz odpowiedź skokową na skok wartości zakłócenia (rys. 1.3). Zbadane zostało przez nas zachowanie układu na skok sterowania kolejno do: $F_1 = 28 \text{ cm}^3/\text{s}$, $F_1 = 36 \text{ cm}^3/\text{s}$, $F_1 = 44 \text{ cm}^3/\text{s}$, $F_1 = 60 \text{ cm}^3/\text{s}$, $F_1 = 68 \text{ cm}^3/\text{s}$, $F_1 = 76 \text{ cm}^3/\text{s}$. Każde dodatkowe $8 \text{ cm}^3/\text{s}$ powodowało wzrost wysokości h_2 o $2,4 \text{ cm}$, również każdy spadek wartości F_1 o $8 \text{ cm}^3/\text{s}$ skutkował spadkiem wysokości h_2 o $2,4 \text{ cm}$, co potwierdza liniowość naszego modelu.

W przypadku badania zachowania układu na skok zakłócenia przetestowane zostały skoki kolejno do wartości: $F_D = 2 \text{ cm}^3/\text{s}$, $F_D = 4 \text{ cm}^3/\text{s}$, $F_D = 6 \text{ cm}^3/\text{s}$, $F_D = 10 \text{ cm}^3/\text{s}$, $F_D = 12 \text{ cm}^3/\text{s}$, $F_D = 14 \text{ cm}^3/\text{s}$. W wyniku otrzymaliśmy zmianę h_2 o 0,6 cm na każde $2 \text{ cm}^3/\text{s}$ różnicy w wartości F_D .



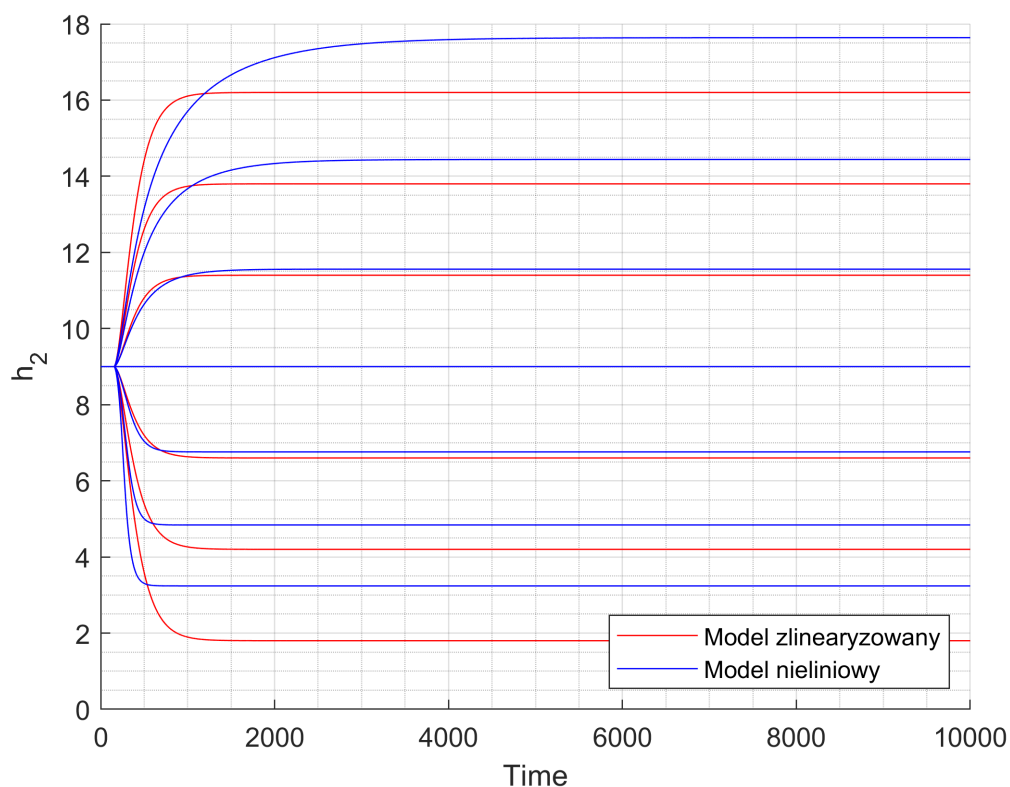
Rys. 1.2. Odpowiedź skokowa modelu liniowego na różne wartości skoku sterowania



Rys. 1.3. Odpowiedź skokowa modelu liniowego na różne wartości skoku zakłócenia

1.3. Porównanie działania modelu liniowego z działaniem modelu nieliniowego

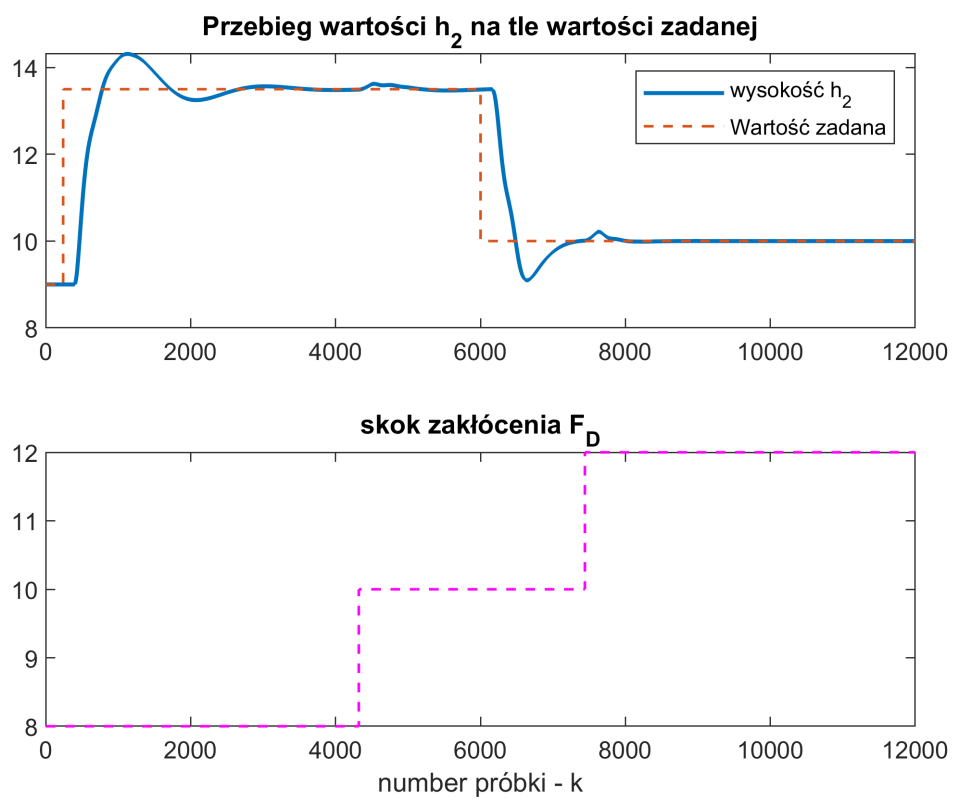
Porównanie działania modelu liniowego z działaniem modelu nieliniowego zostało zaimplementowane w pliku `test_lin.m`. Korzysta on również z symulacji, które zaimplementowane zostały w plikach o nazwach: `step.m` oraz `step_lin.m`. Do porównania zostały użyte skoki, z których korzystaliśmy również w poprzedniej sekcji, czyli: $F_1 = 28 \text{ cm}^3/\text{s}$, $F_1 = 36 \text{ cm}^3/\text{s}$, $F_1 = 44 \text{ cm}^3/\text{s}$, $F_1 = 60 \text{ cm}^3/\text{s}$, $F_1 = 68 \text{ cm}^3/\text{s}$, $F_1 = 76 \text{ cm}^3/\text{s}$. Wynik zaprezentowany został na rys. 1.4. Wynik był zgodny z naszymi przewidywaniami: dla skoków niewiele odbiegających od punktu pracy, względem którego dokonana została linearyzacja dysproporcja jest niewielka, wręcz pomijalna. Jednakże dla większej zmiany wartości sterującej - przykładowo z $52 \text{ cm}^3/\text{s}$ do $76 \text{ cm}^3/\text{s}$ różnica wynosi $1,43 \text{ cm}$, co w porównaniu między wartościami otrzymanymi z obu modeli stanowi około 8%, więc nie powinno być pomijane.



Rys. 1.4. Porównanie odpowiedzi modelu liniowego i nieliniowego

1.4. Nerozmyty regulator DMC

Implementacja regulatora DMC została dokonana w plikach `dmc_single.m` oraz `simulate_dmc_single.m`. Proces doboru nastaw został zakończony z następującymi wartościami: $\lambda = 5$, $N = 1500$, $Nu = 50$ oraz $D = 5000$. Przykładowy przebieg, z dwoma skokami: z 9 do 13,5, a następnie z 13,5 do 10, został przedstawiony na rys. 1.5.

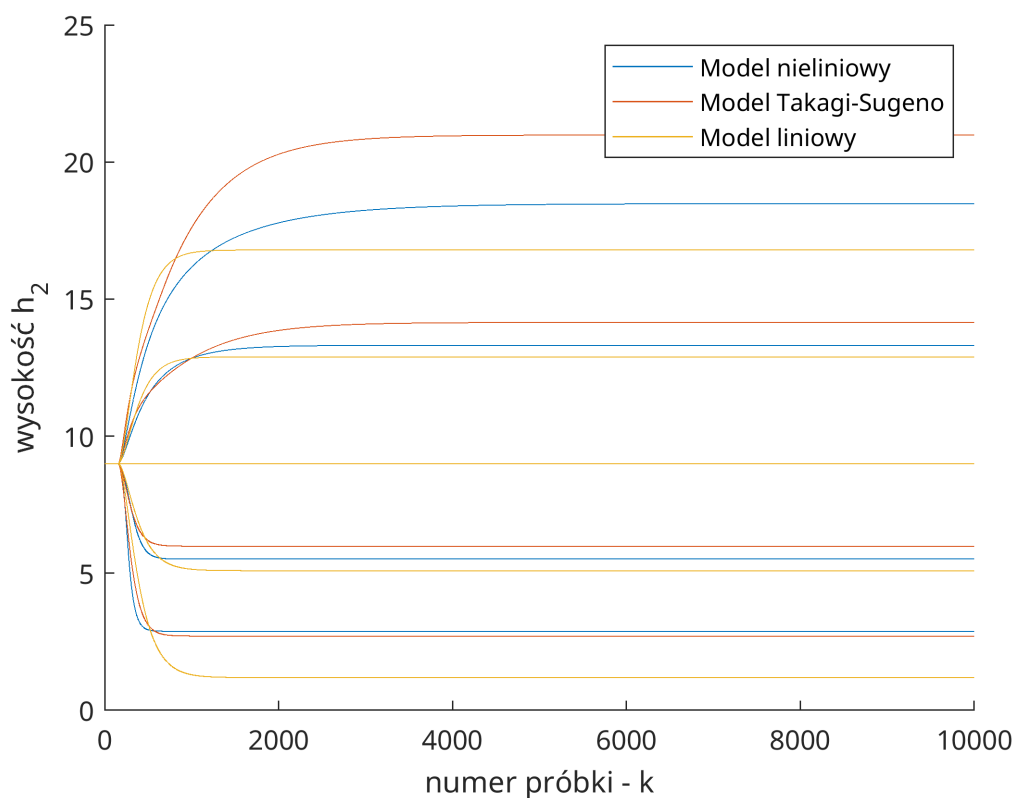
Rys. 1.5. Regulator DMC z λ 5

2. Zadanie drugie

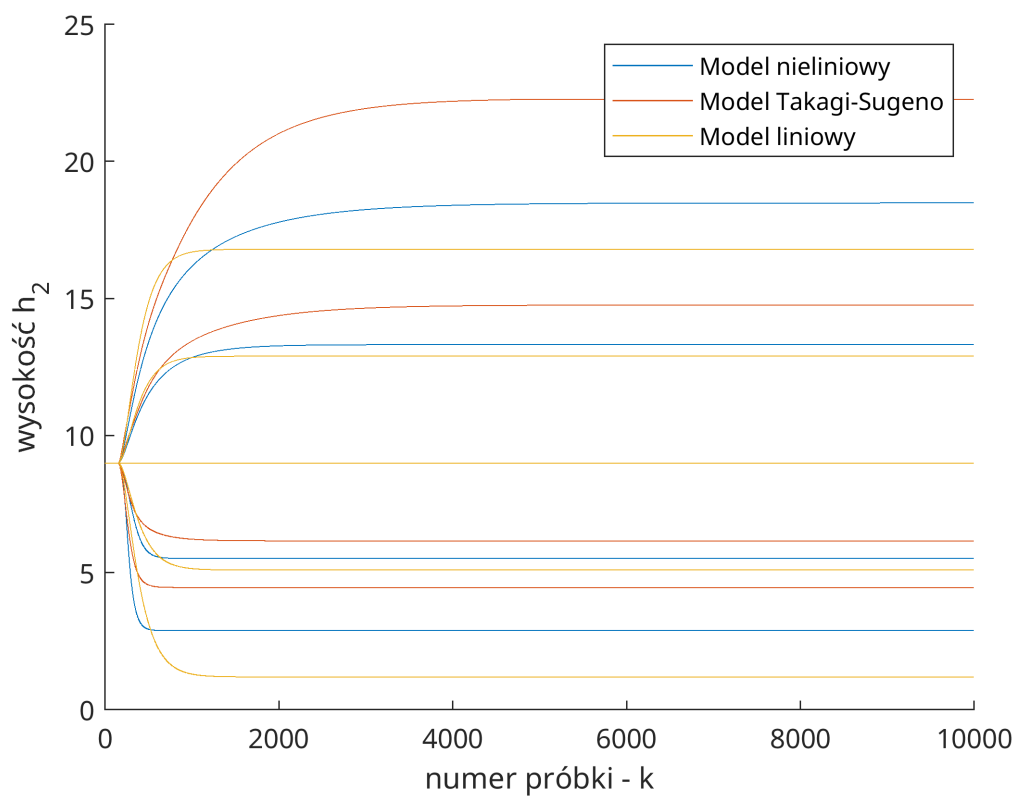
2.1. Rozmyte modele Takagi-Sugeno

W celu utworzenia rozmytych modeli Takagi-Sugeno utworzony został skrypt `test_fuzzy.m`, w którym za pomocą parametru n wybieramy liczbę modeli lokalnych. Wyniki jego działania dla $n = 2$, $n = 3$, $n = 4$ oraz $n = 5$ zostały przedstawione kolejno na rysunkach: 2.1, 2.2, 2.3 oraz 2.4. Początkowym kształtem funkcji przynależności była funkcja trójkątna oparta na równomiernym podziale wyjścia. Zdecydowaliśmy się na taki wybór zmiennej, gdyż w tym przypadku wyjście zmienia się wolniej od sterowania, co usprawni proces sterowania obiektem. Spowoduje on również zmniejszenie szansy na uzyskanie niepożądanego stanu oscylacji, które mogą wystąpić podczas przełączania się sterowania między dwoma modelami lokalnymi.

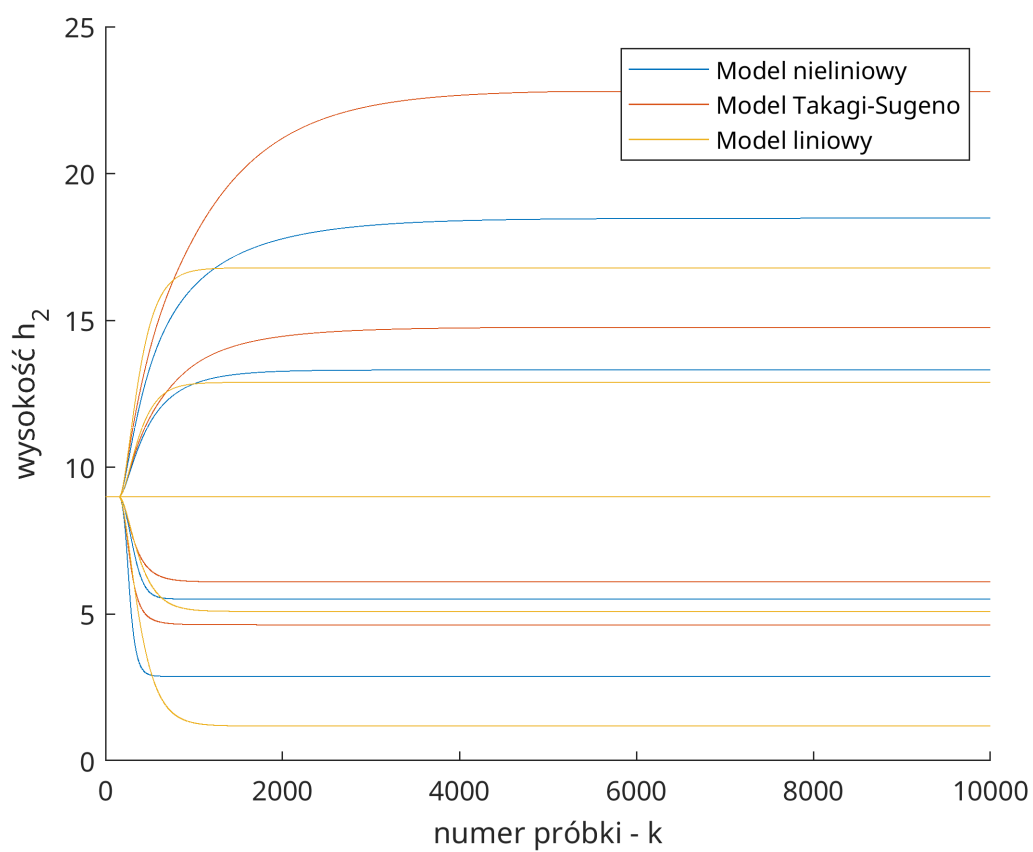
Jak można zobaczyć na przebiegach, w każdym z czterech wymienionych przypadków, wykres model Takagi-Sugeno się nieco wyminął z modelem nieliniowym. Natomiast na rysunku przedstawiającym model korzystający z pięciu modeli lokalnych widać, iż najdłużej trzyma się on okolicy modelu nieliniowego w początkowym etapie narastania lub spadku od punktu pracy. Dlatego też zdecydowaliśmy się na wybór tego modelu do dalszych badań.



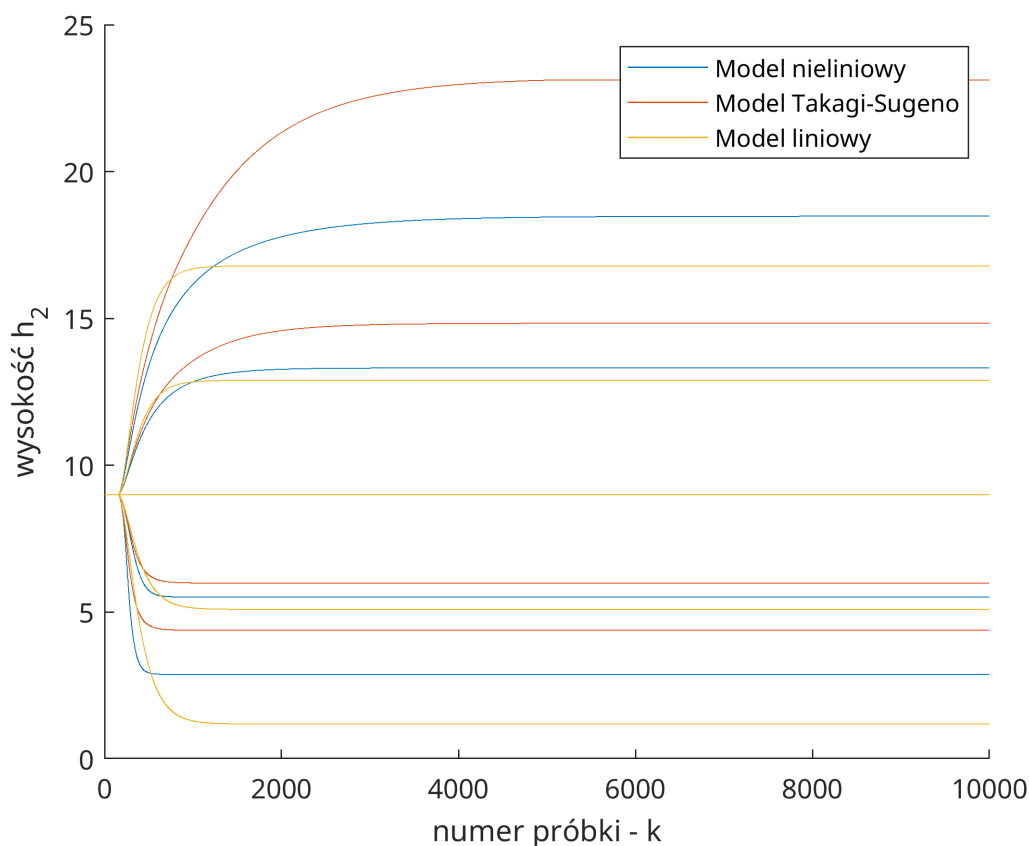
Rys. 2.1. Model Takagi-Sugeno z dwoma modelami lokalnymi



Rys. 2.2. Model Takagi-Sugeno z trzema modelami lokalnymi



Rys. 2.3. Model Takagi-Sugeno z czterema modelami lokalnymi



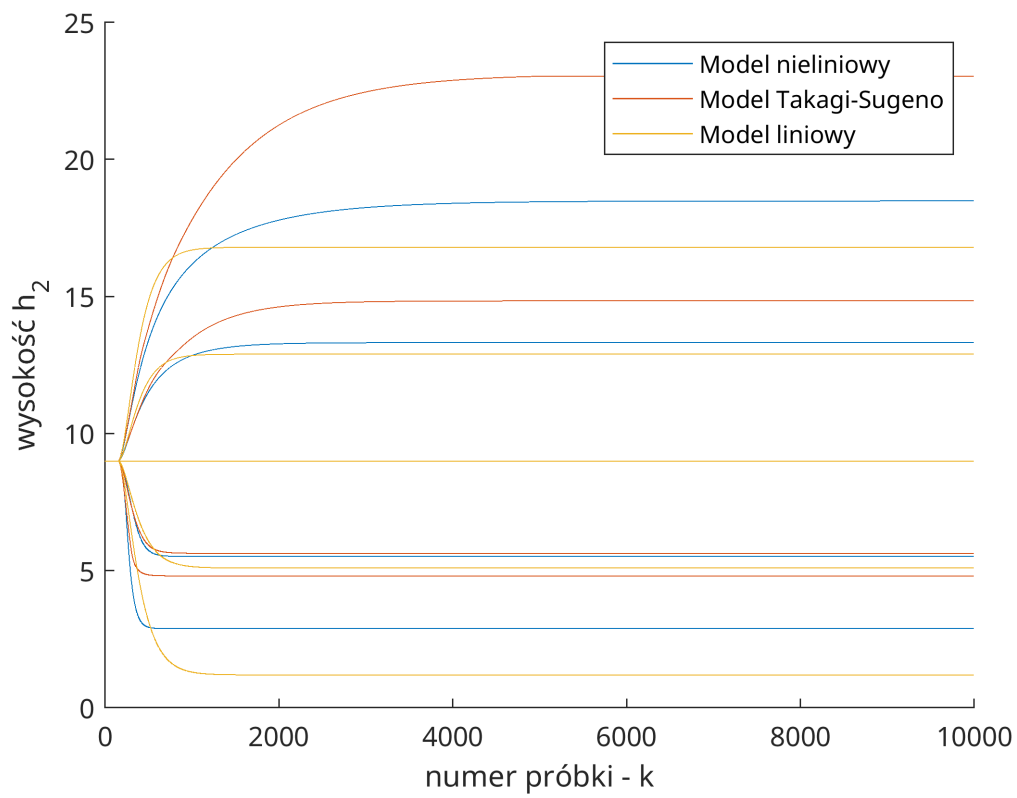
Rys. 2.4. Model Takagi-Sugeno z pięcioma modelami lokalnymi

2.2. Poprawa wybranego modelu rozmytego za pomocą manipulacji kształtem funkcji przynależności

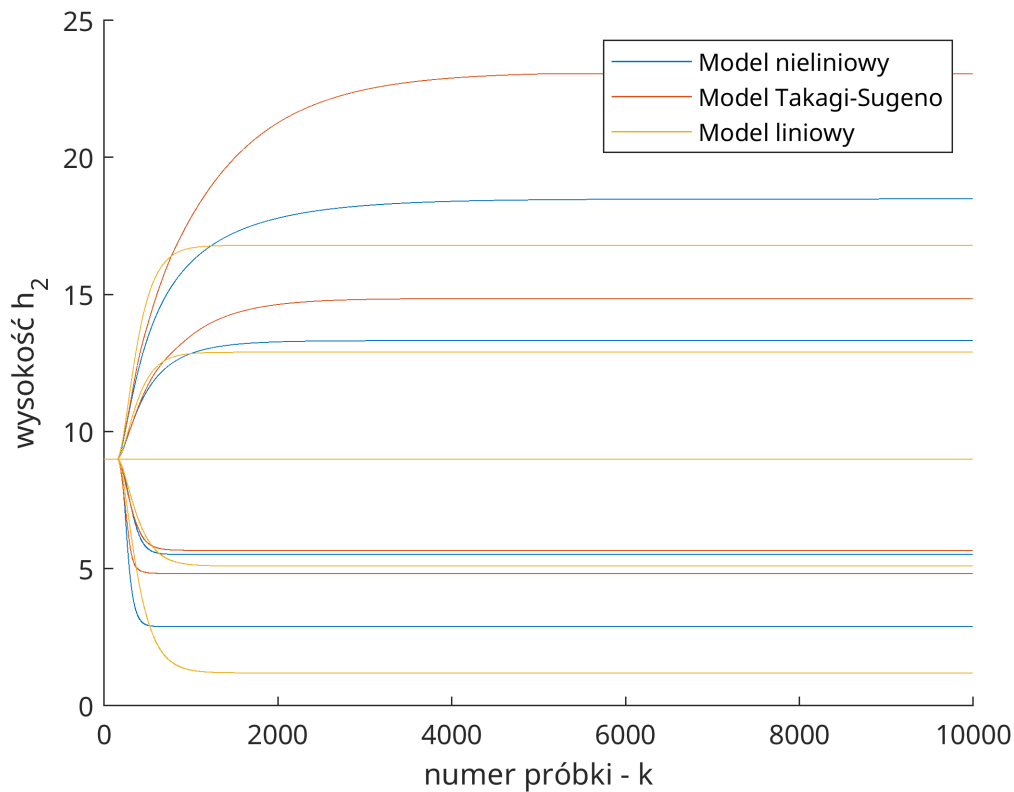
W celu poprawy modelu rozmytego z pięcioma modelami lokalnymi za pomocą manipulacji kształtem funkcji przynależności zostały przetestowane trzy różne kształty tejże funkcji: sigmoidalny (rys. 2.5), trapezowy (rys. 2.6) oraz trójkątny (wygenerowany w poprzedniej sekcji, na rys. 2.4). Test polegał na uruchomieniu skryptu `test_fuzzy.m` z edytowaną linią odpowiedzialną za wybór funkcji przynależności – w każdym kolejnym uruchomieniu programu wybór padał na inną z trzech poniższych:

```
f_przyn = @przyn_triangle;
f_przyn = @przyn_trap;
f_przyn = @przyn_sigmoid;
```

Porównując te przebiegi między sobą, na pierwszy rzut oka ciężko znaleźć różnicę, gdyż są one bardzo niewielkie. Najłatwiej jest ją dostrzec na przykładzie trzeciego skoku - z wartości $h_2 = 9$ cm do okolic $h_2 = 6$ cm. W porównaniu do modelu nieliniowego najbardziej odbiega model używający funkcji trójkątnej. Natomiast funkcja trapezowa daje wynik na tyle zbliżony do funkcji sigmoidalnej, iż określenie zwycięzcy wymaga przybliżenia lub sprawdzenia dokładnych danych z przebiegu. Ostatecznie najlepszą okazała się ostatnia z wymienionych - funkcja sigmoidalna.



Rys. 2.5. Model Takagi-Sugeno z pięcioma modelami lokalnymi oparty na sigmoidalnej funkcji przynależności



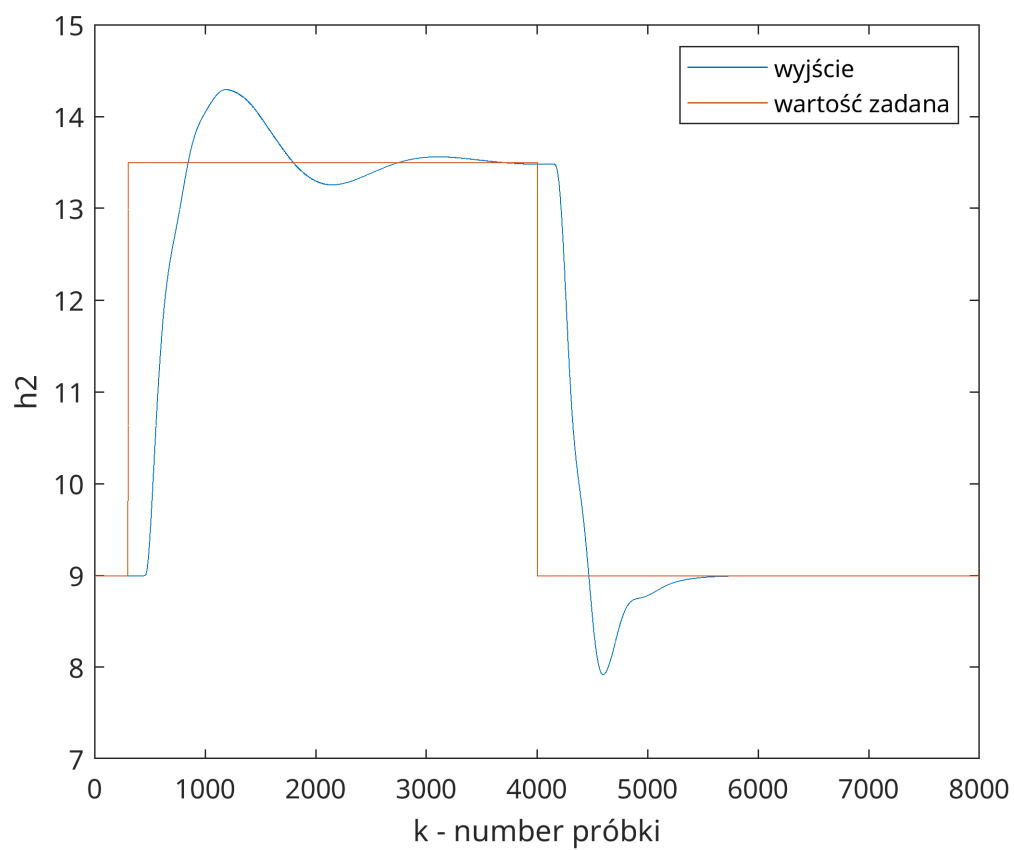
Rys. 2.6. Model Takagi-Sugeno z pięcioma modelami lokalnymi oparty na trapezowej funkcji przynależności

2.3. Projektowanie rozmytego algorytmu regulacji predykcyjnej w wersji analitycznej

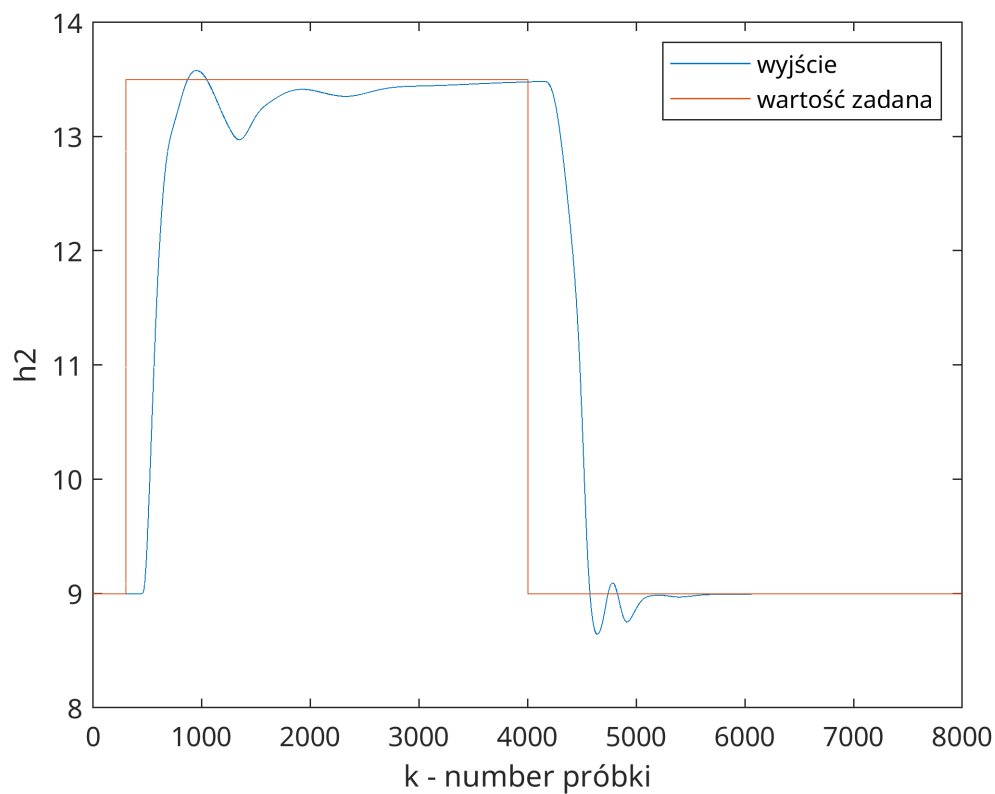
Algorytm DMC w wersji rozmytej został zaimplementowany w pliku `simulate_dmc_fuzzy.m`. Korzysta on z sigmoidalnej funkcji przynależności. Proces jego strojenia został zakończony z następującymi wartościami λ : $\lambda_1 = 400$, $\lambda_2 = 30$, $\lambda_3 = 1$, $\lambda_4 = 900$, $\lambda_5 = 600$. Środki przedziałów rozmywania znajdują się w punktach: 3.8, 7.4, 11, 14.6 oraz 18.2. Przykładowy przebieg wraz ze skokiem został przedstawiony na rys. 2.8.

2.4. Porównanie regulatora DMC w wersji rozmytej do regulatora DMC w wersji podstawowej (z zadania pierwszego)

Rysunek 2.7 przedstawia podstawową wersję regulatora DMC, którą uzyskaliśmy w zadaniu pierwszym. Rozmyta wersja przedstawiona jest na rys. 2.8. Obydwie wersje napotykały na skok wartości zadanej z 9 cm do 13,5 cm na początku pokazanego przebiegu. Rozmyta wersja znacząco zmniejszyła przeregulowanie, czego kosztem było osiągnięcie wartości zadanej kilkanaście próbek później. W drugim skoku wartości zadanej – powrocie do wartości 9 cm – podstawowa wersja spadła aż do wartości około 8 cm, podczas gdy rozmytemu DMC udało się ograniczyć ten błąd o około 0,5 cm. Czas, po którym następuje ostateczna stabilizacja jest taki sam w obu przypadkach. Podsumowując, rozmyty DMC pozwolił nam na zmniejszenie niedokładności podczas procesu sterowania, co w wielu przypadkach może być bardzo dużym benefitem.



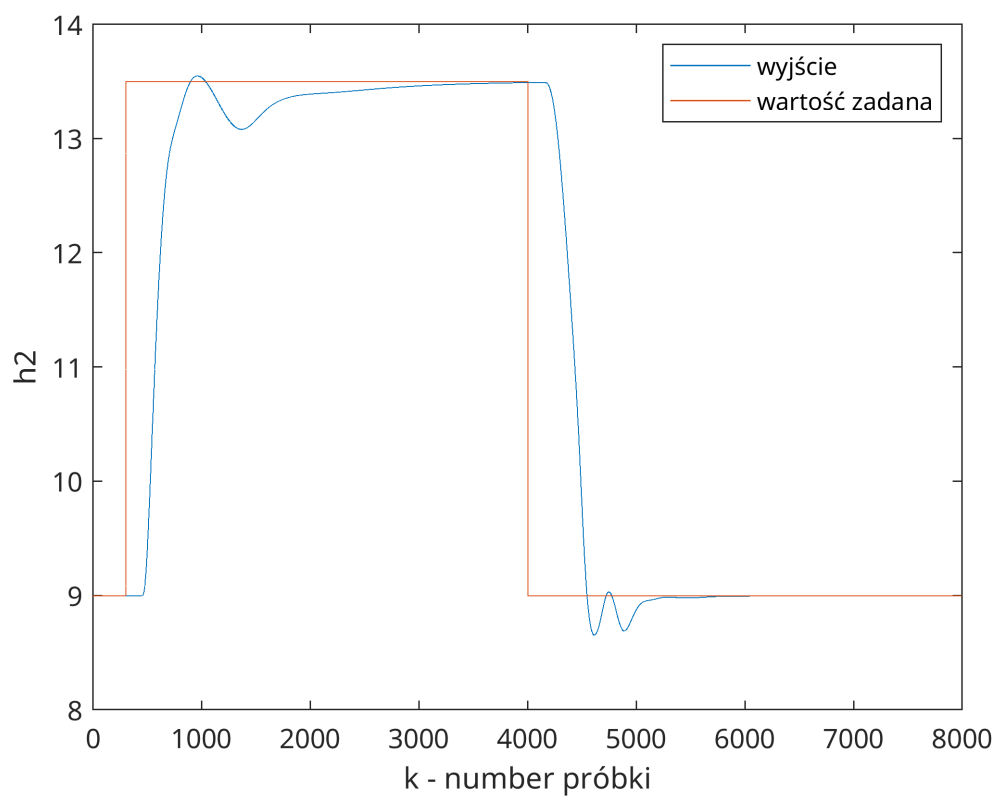
Rys. 2.7. Przykładowy przebieg dla DMC w wersji podstawowej (z zadania pierwszego)



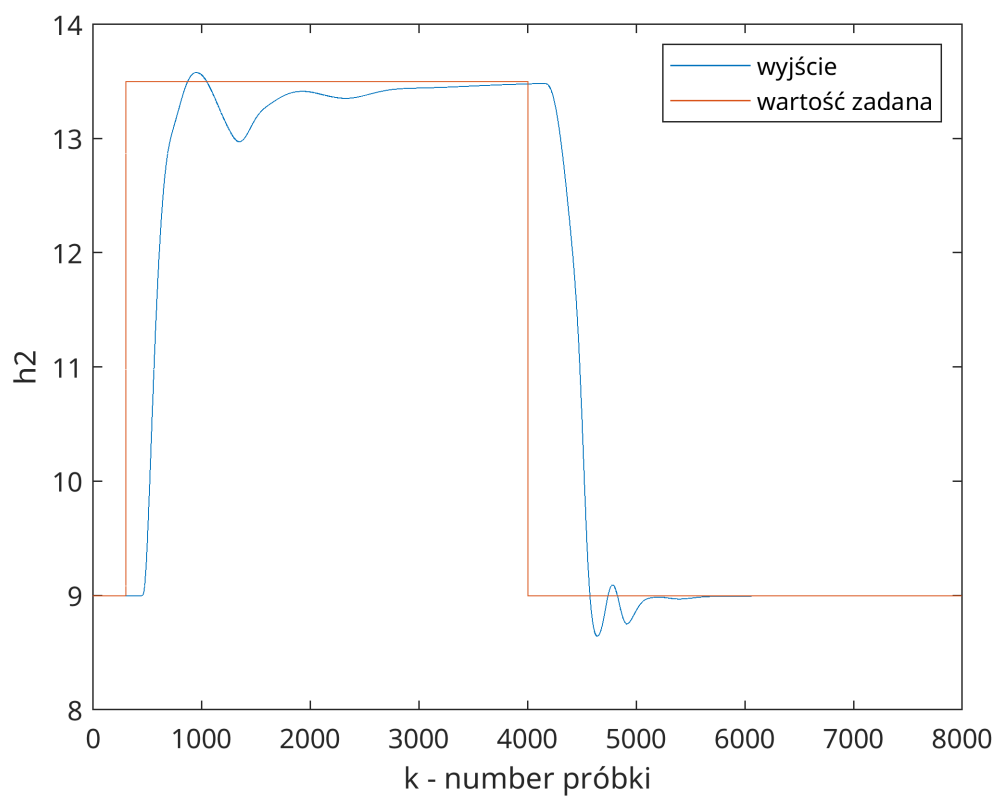
Rys. 2.8. Przykładowy przebieg dla DMC w wersji rozmytej

2.5. Eksperymenty ze zmianą kształtu funkcji przynależności w regulatorze

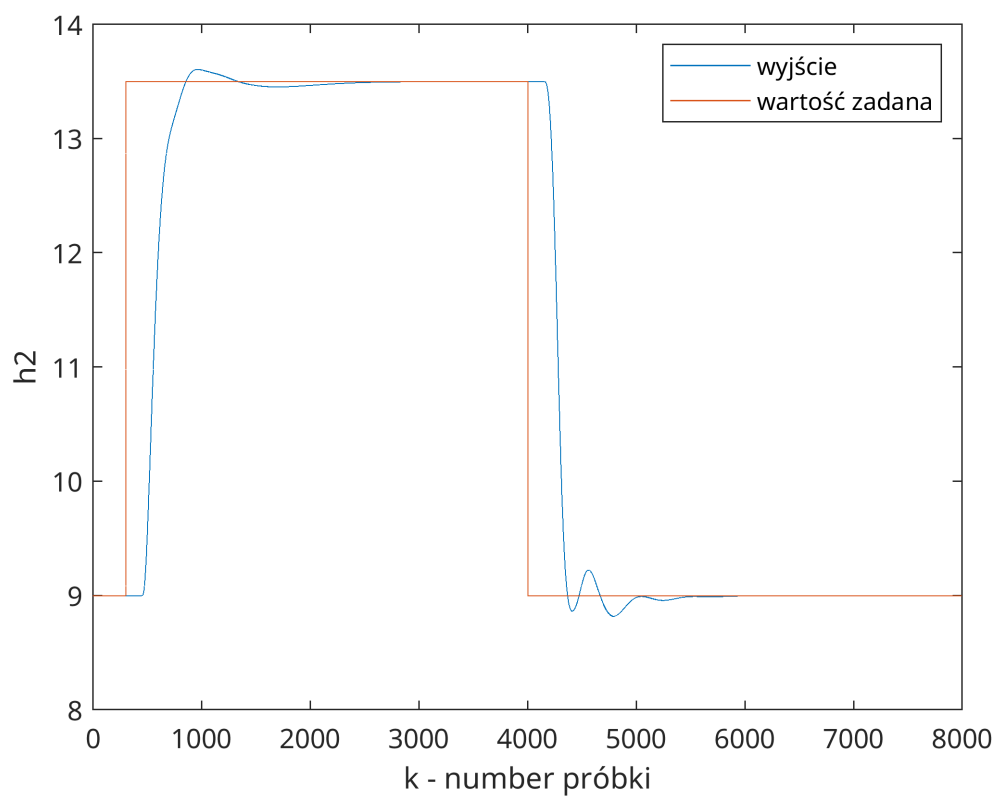
Zbadane zostały trzy przypadki. Pierwszym jest funkcja sigmoidalna – przedstawiona na rys. 2.9 – wskaźnik jakości regulacji wyniósł dla niej $E = 1,2228 \cdot 10^4$. Drugą sytuacją jest funkcja trapezowa - dla niej błąd wyniósł wartość $E = 1,2811 \cdot 10^4$. Jej przebieg znajduje się na rys. 2.10. Ostatnim przypadkiem jest funkcja trójkątna, która uzyskała $E = 9,8044 \cdot 10^3$ i została przedstawiona na rys. 2.11. Podsumowując, użycie funkcji trójkątnej okazało się świetnym sposobem na polepszenie jakości działania algorytmu DMC, co widać nie tylko na przykładzie numerycznym - wartości błędu, lecz także pozwoliła ona na wypłaszczenie znaczącej oscylacji, która pojawiała się podczas skoku wartości zadanej w górę.



Rys. 2.9. Przykładowy przebieg dla DMC w wersji rozmytej dla sigmoidalnej funkcji przynależności



Rys. 2.10. Przykładowy przebieg dla DMC w wersji rozmytej dla trapezowej funkcji przynależności

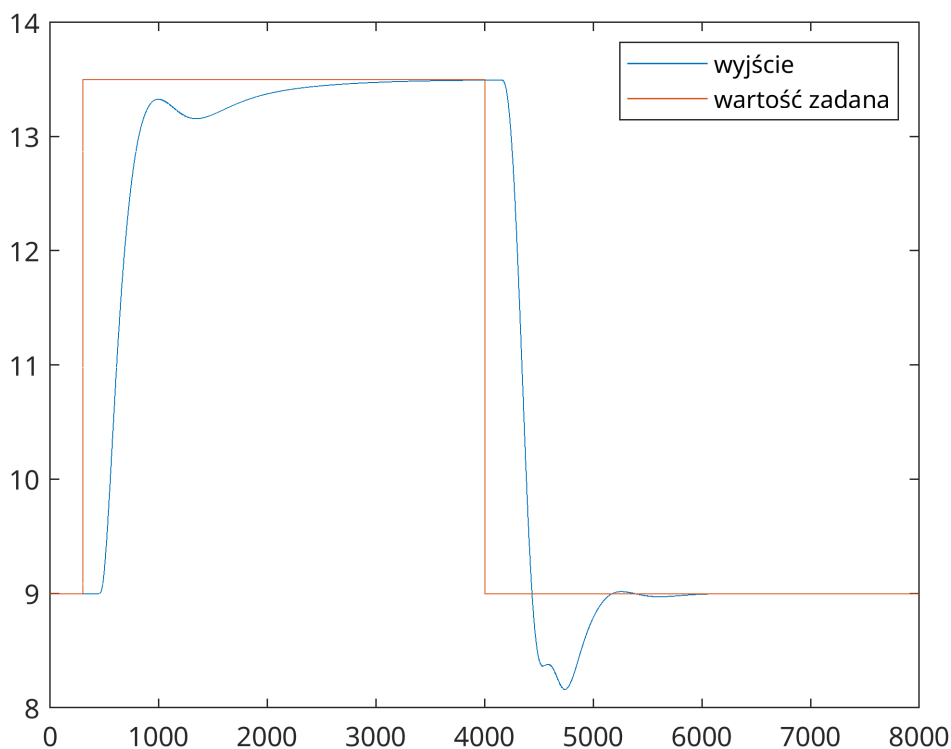


Rys. 2.11. Przykładowy przebieg dla DMC w wersji rozmytej dla trójkątnej funkcji przynależności

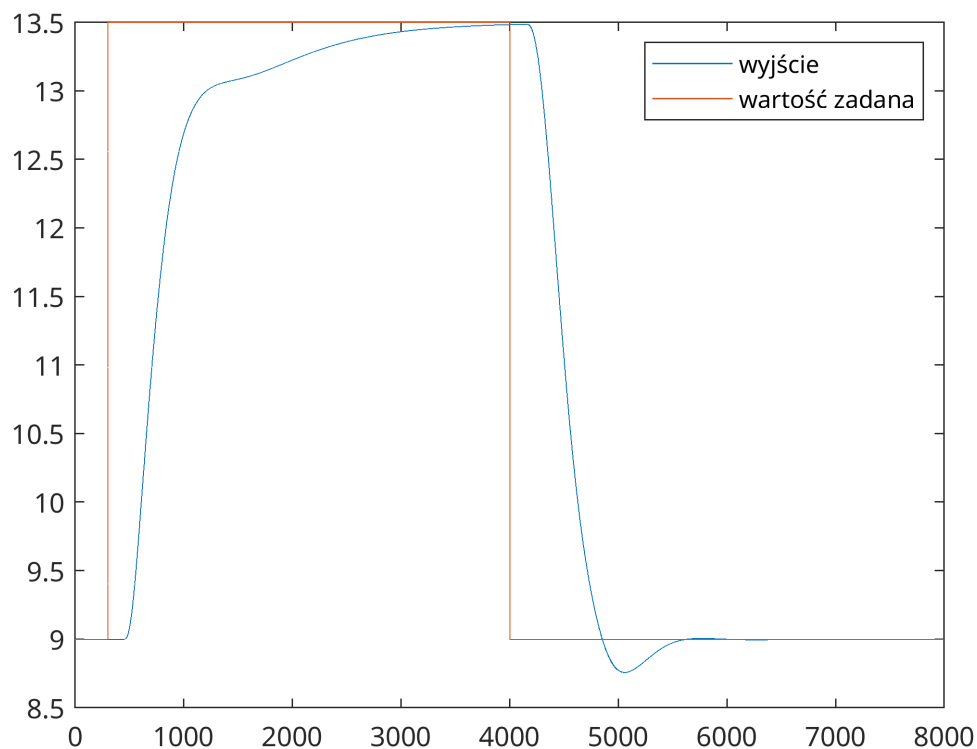
3. Zadanie trzecie

3.1. Algorytm regulacji predykcyjnej typu SL

Algorytm regulacji predykcyjnej typu SL został zaimplementowany w skrypcie `simulate_dmc_sl.m` oraz wywoływanej przez nią funkcji `dmc_sl.m`. Zostały nadane następujące ograniczenia na wartość sterowania: $0 < F_1 < 200$. Z procesu strojenia uzyskana została następująca wartość lambdy: $\lambda = 20$. Taka nastawa wraz z połączeniem z trapezową funkcją przynależności pozwalają nam uzyskać przebieg, który został przedstawiony na rys. 3.1. Nie byliśmy jednak w pełni usatysfakcjonowani jego zachowaniem podczas skoku wartości zadanej w dół, lecz próby poprawienia tego zachowania powodowały pogorszenie działania podczas pierwszego skoku. Na rys. 3.2 został przedstawiony drugi wariant proponowanych przez nas nastaw. W tym przypadku współczynnik kary $\lambda = 500$. Jako, iż regulator FDMC typu SL, ogranicza nas poprzez wybór tylko jednego współczynnika kary, wybór który z wyżej opisanych wariantów można uznać za lepszy powinien polegać na określeniu priorytetów działania naszego obiektu. Jeśli ważniejsza jest dokładność przy nalewaniu, to ostatecznym wyborem powinna być lambda równa 5, w przeciwnym wypadku – gdy proces opróżniania zbiornika stawiamy nad procesem napełniania powinniśmy wybrać lambda równą 500.



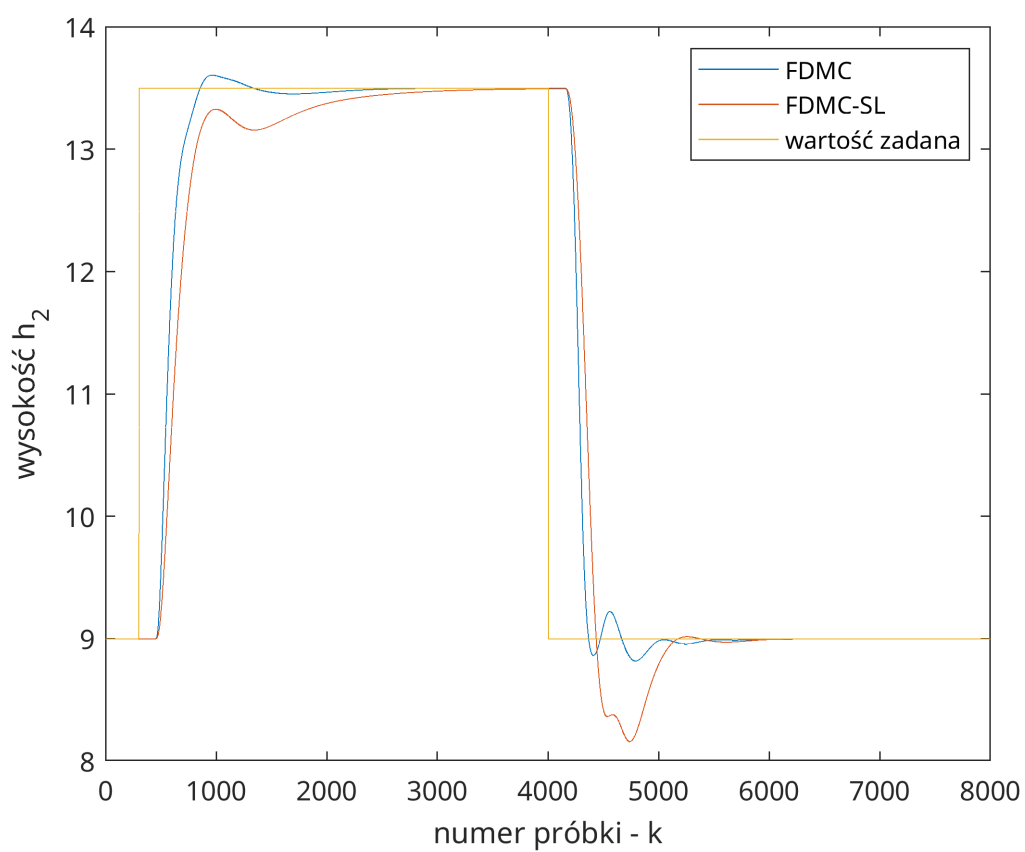
Rys. 3.1. Przykładowy przebieg procesu z użyciem algorytmu regulacji predykcyjnej typu SL - wariant pierwszy



Rys. 3.2. Przykładowy przebieg procesu z użyciem algorytmu regulacji predykcyjnej typu SL - wariant drugi

3.2. Porównanie algorytmu regulacji predykcyjnej typu SL z poprzednim wariantem

Porównanie jakości działania przedstawione jest na rys. 3.3. W tym przypadku nie udało się otrzymać poprawy regulacji za pomocą DMC w wersji SL względem poprzedniej wersji. Wynika to z faktu, iż odpowiednie dostrojenie współczynników kar wymagało dość dużej rozbieżności, które są wręcz niemożliwe do przybliżenia w akceptowalny sposób za pomocą jednej wartości.



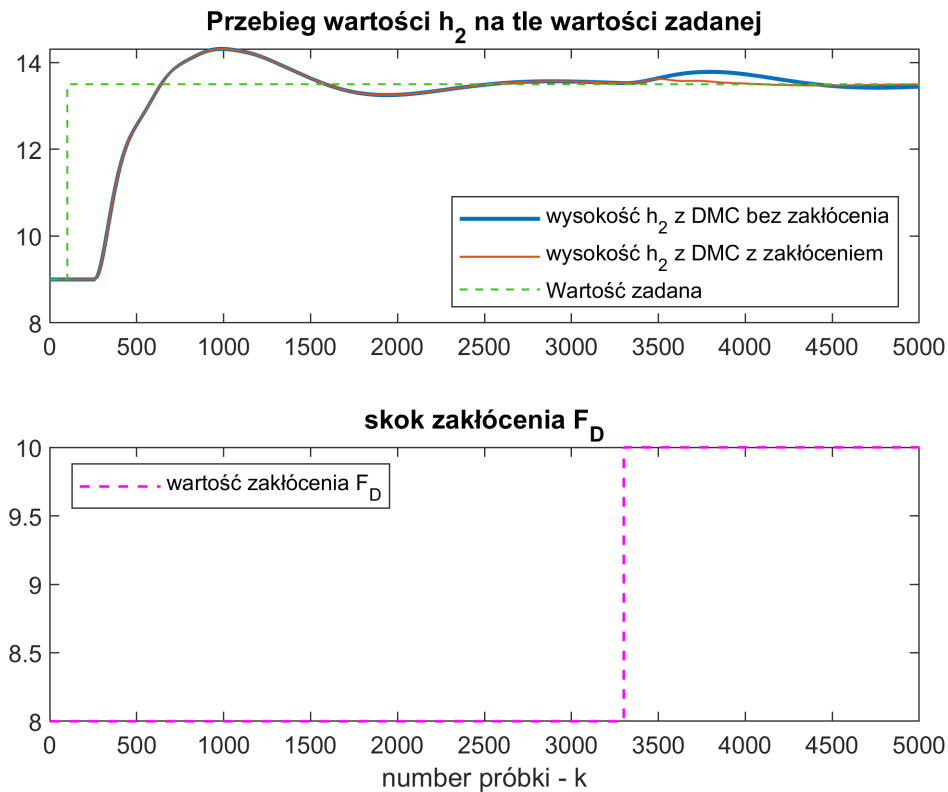
Rys. 3.3. Porównanie działania obu rozmytych wariantów algorytmu regulacji predykcyjnej

4. Zadanie czwarte

4.1. Uzupełnienie nierozmytego algorytmu DMC o mechanizm uwzględniania pomiaru zakłócenia

Implementacja mechanizmu uwzględniania pomiaru zakłócenia została wykonana w skrypcie `dmc_z_zakloeniem.m`, która bazuje na pliku używanym w zadaniu pierwszym. Zmiana polegała na doimplementowaniu macierzy M_z^P wraz z wektorem Δu_z^P .

Na rys. 4.1 przedstawione zostało porównanie przykładowego przebiegu, w którym występuje skok wartości zadanej w początkowej fazie, a następnie występuje skok zakłócenia, które ponownie zaburzyło stabilizację wysokość wody h_2 w zbiorniku drugim. Jak można zobaczyć, przebieg uzyskany z DMC uwzględniającego pomiar zakłócenia poradził sobie znacząco lepiej: wysokość niewiele przekroczyła wartość zadaną, podczas gdy podstawowa wersja pozwoliła na przeregulowanie o 30 centymetrów.



Rys. 4.1. Porównanie przebiegu wartości h_2 uzyskanej przez DMC wraz z mechanizmem uwzględniania pomiaru zakłócenia oraz bez niej

Jakość poprawy możemy policzyć jako różnica między sumą błędów uzyskanych z podstawowej wersji DMC (e_1), a sumą błędów uzyskaną przez regulator DMC wraz z mechanizmem uwzględniającym pomiar zakłócenia (e_2):

$$e_1 - e_2 = 988.1445 - 872.8850 = 115.2595$$

Poprawa błędu jest znacząca – jest to jego zmniejszenie o ponad 11%.