SZAU Projekt 2

Janusz Kubiak, Krzysztof Jóskowiak

14.01.2021

1 Wstęp

Regulowany w zadaniu proces opisany jest równaniami:

$$\begin{aligned} x_1(k) &= -\alpha_1 x_1(k-1) + x_2(k-1) + \beta_1 g_1(u(k-3)) \\ x_2(k) &= -\alpha_2 x_1(k-1) + \beta_2 g_1(u(k-3)) \\ y(k) &= g_2(x_1(k)) \end{aligned}$$

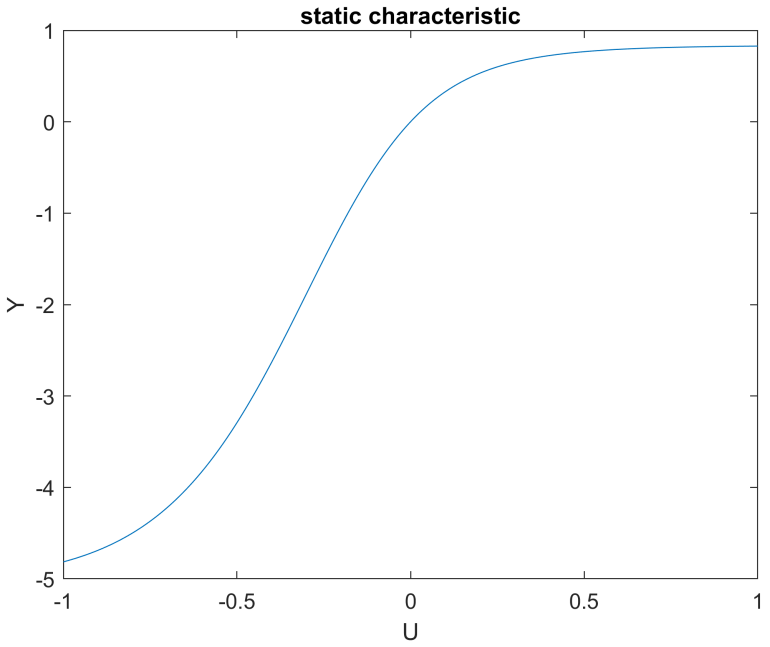
gdzie u - sygnał wejściowy, y - wyjściowy, x_1, x_2 - zmienne stanu, $\alpha_1 = -1,535262$, $\alpha_2 = 0.586646$, $\beta_1 = 0.02797$, $\beta_2 = 0.023414$, oraz:

$$g_1(u(k-3)) = \frac{exp(4.5u(k-3)) - 1}{exp(4.5u(k-3)) + 1}, g_2(u(k)) = 1 - exp(-1.8x_1(k))$$

2 Symulacja procesu

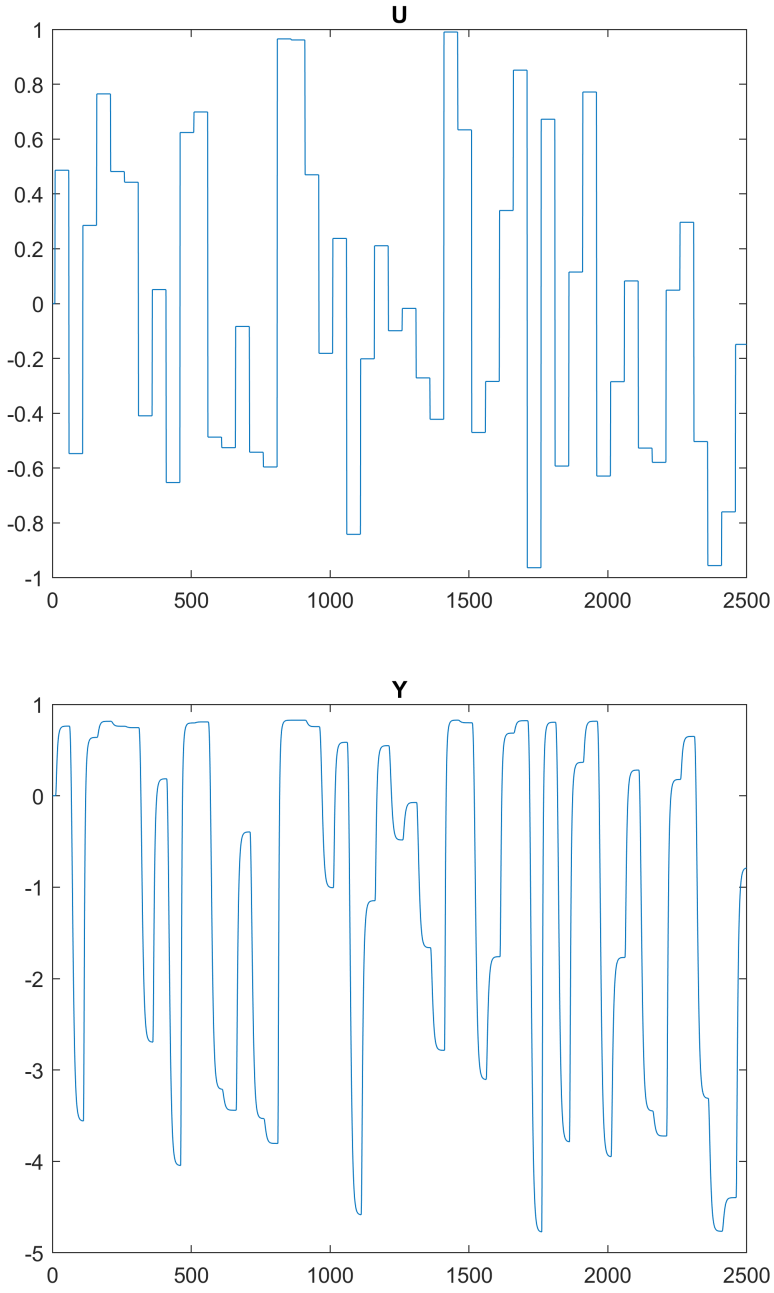
2.1 Charakterystyka statyczna

Charakterystyka statyczna procesu została wyznaczona metodą symulacyjną.

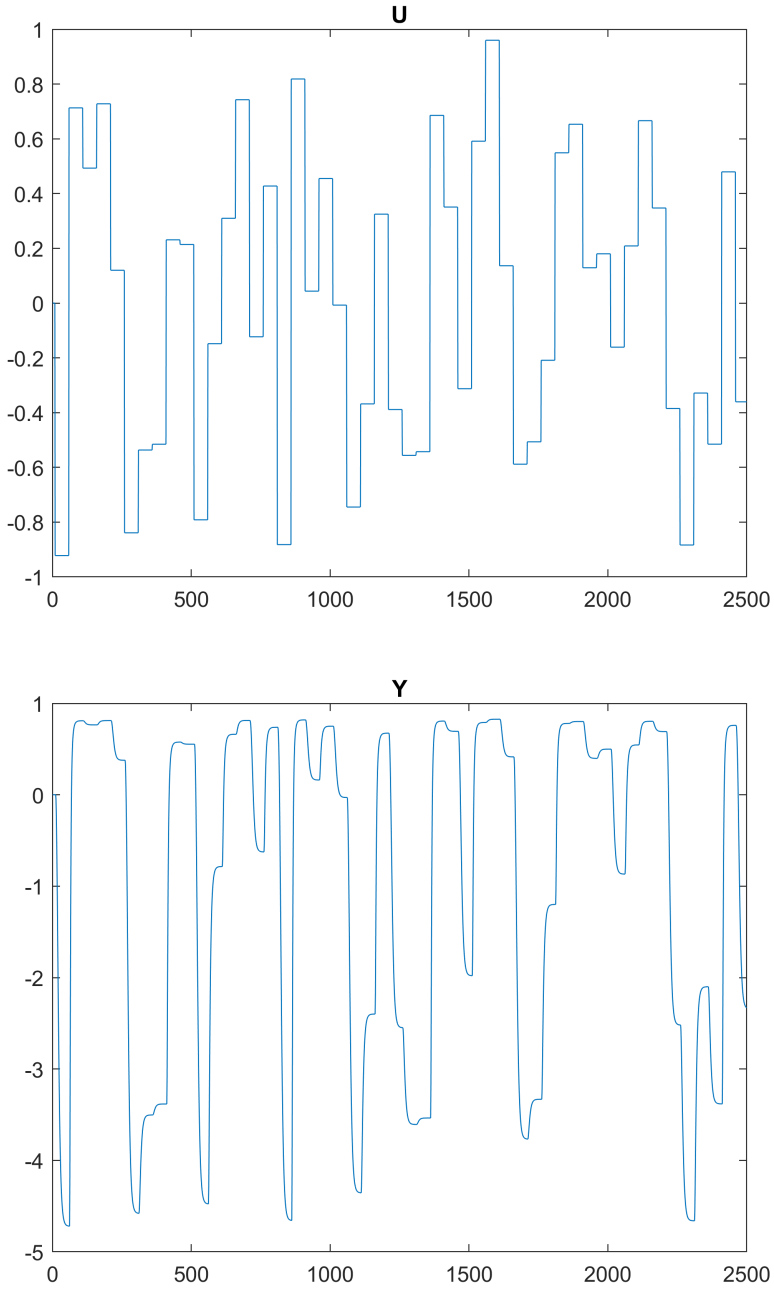


Rysunek 1: Charakterystyka statyczna obiektu.

2.2 Symulacje



Rysunek 2: Dane uczące.



Rysunek 3: Dane weryfikujące.

3 Modelowanie procesu

3.1 Opóźnienie

Opóźnienie procesu można odczytać bezpośrednio z równań. Jest to odcinek czasu po którym obiekt zaczyna odpowiadać na zmianę sterowania. Dla tego procesu wynosi ono $\tau = 3$.

3.2 Modele neuronowe

Modele neuronowe uczone były w trybie rekurencyjnym, maksymalnie 1000 iteracji, oraz błędzie granicznym 0.00001 algorytmem BFGS, przy pomocy dostarczonego programu "sieci". W poniższej tabelce przedstawiono błędy średniokwadratowe, gdzie dla najlepszego modelu z każdej grupy dla danych weryfikujących w trybie rekurencyjnym. Błąd ten dany jest równaniem

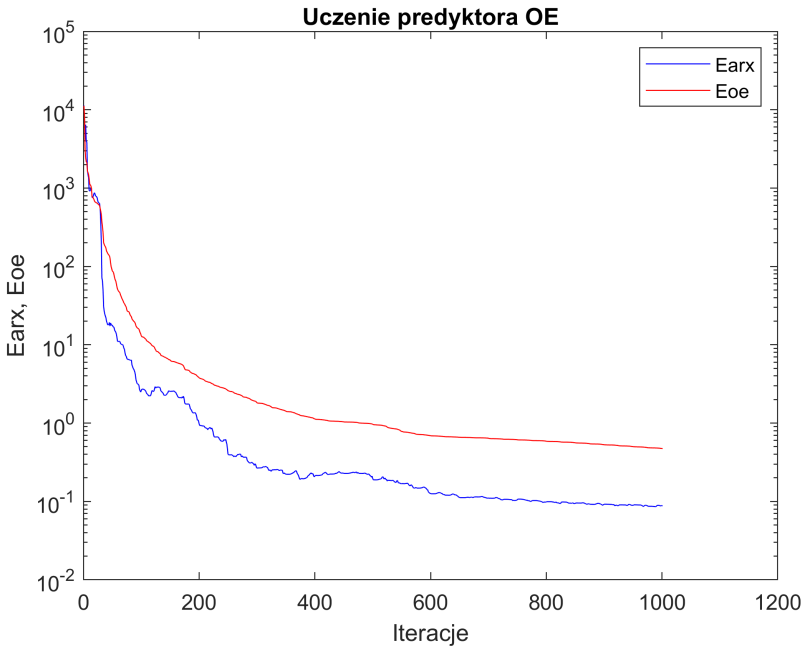
$$E = \frac{\sum_{k=1}^{L}(y - y_{mod})}{L},$$

gdzie y - wyjście procesu, y_{mod} - wyjście modelu, L - długość symulacji.
Wzór ten podany został, ponieważ program "sieci" najwyraźniej inaczej obliczał błąd (prawdopodobnie nie dzielony był on przez długość symulacji), więc błędy na rysunkach z uczenia modelu mogą odbiegać od podanych w tabelkach itp. Nie zmienia to uszeregowania modeli pod względem błędu, więc nie powinno być problemem. Poza modelami z 1 i 2 neuronami uczenie wszystkich pozostałych wykorzystało maksymalną liczbę iteracji, było więc dość powolne.

Neurony	Błąd OE
1	7.612144e-02
2	3.112128e-03
3	1.229195e-03
4	8.506235e-04
5	5.153278e-04
6	3.153605e-04
7	3.945854e-04
8	2.302020e-04
9	2.257136e-04
10	1.545001e-04

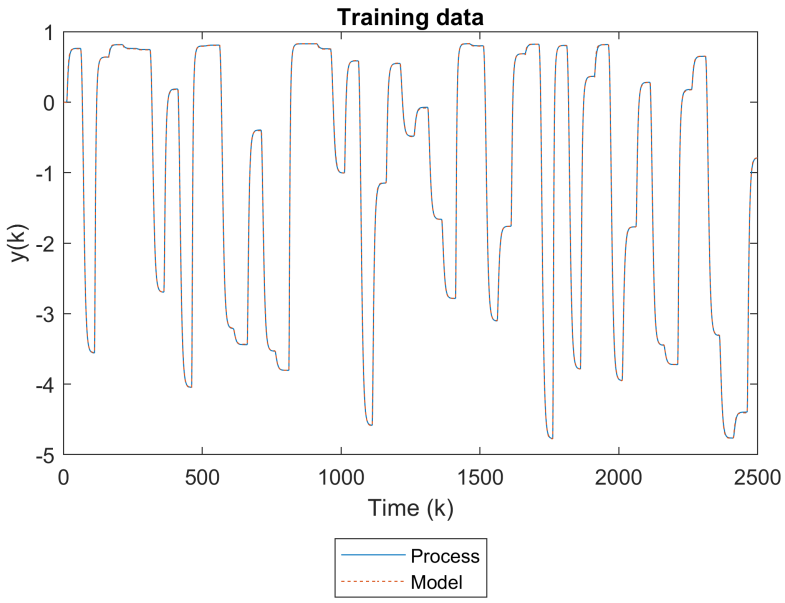
3.3 Wybór najlepszego

W związku ze wzrostem błędu między modelem z 6 a 7 neuronami uznano model z 6 neuronami za najlepszy kompromis między dokładnością a ilością obliczeń.

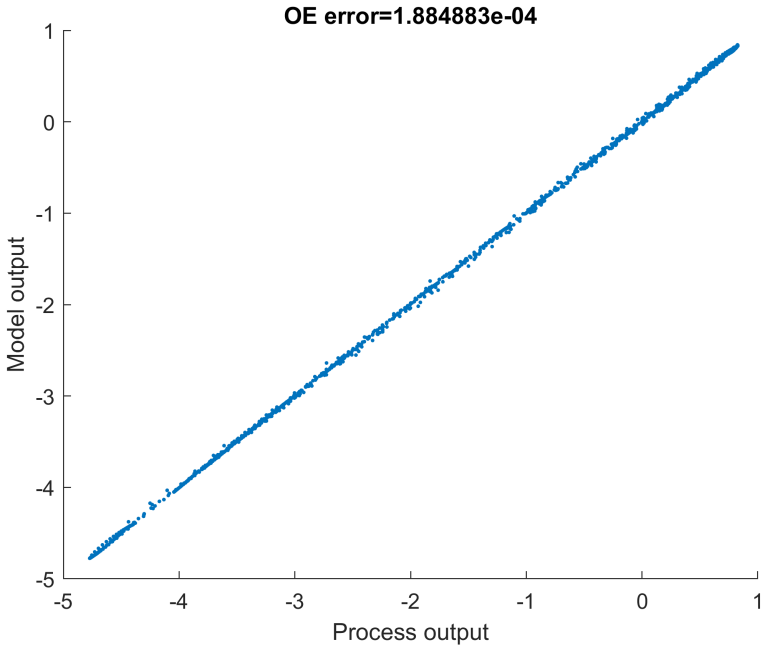


Rysunek 4: Przebieg błędów podczas uczenia najlepszego modelu z sześcioma neuronami.

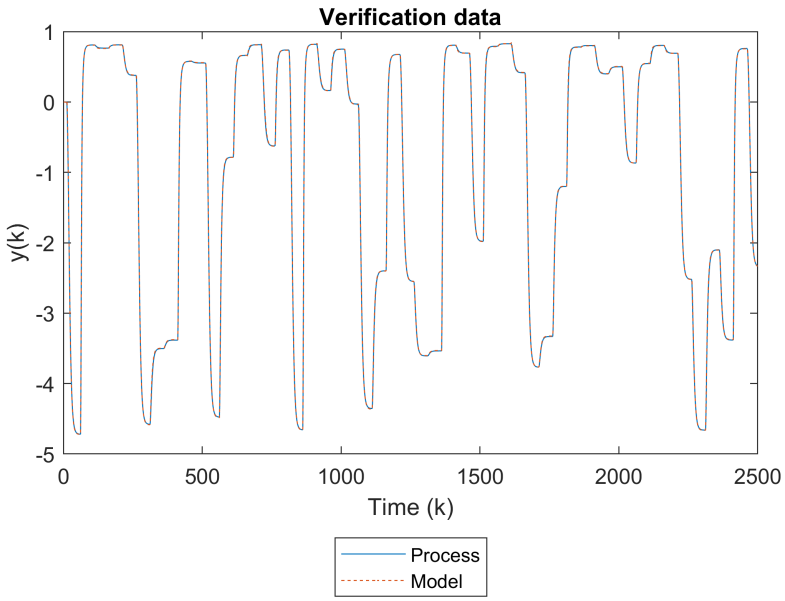
3.4 Porównanie procesu z modelem



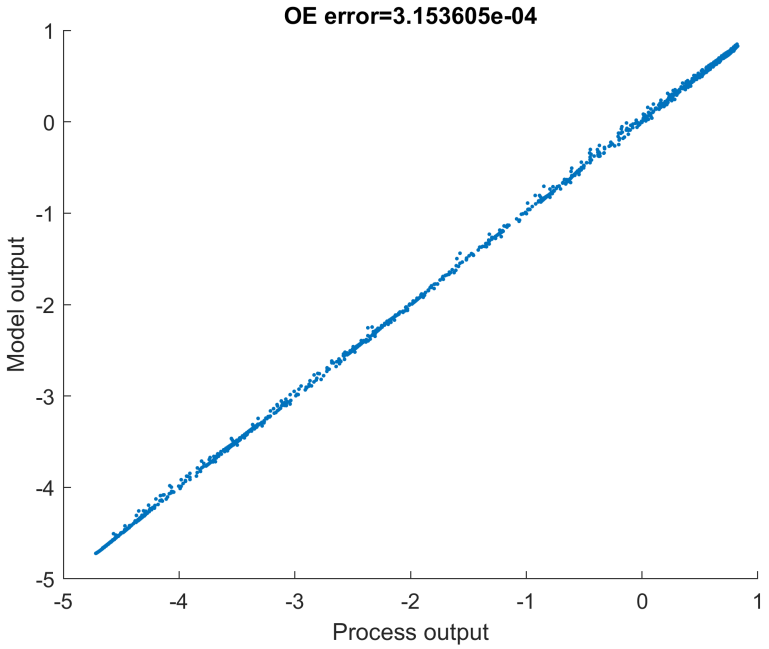
Rysunek 5: Porównanie wyjścia procesu z wyjściem modelu dla danych uczących.



Rysunek 6: Wyjścia modelu na tle wyjść procesu dla danych uczących.



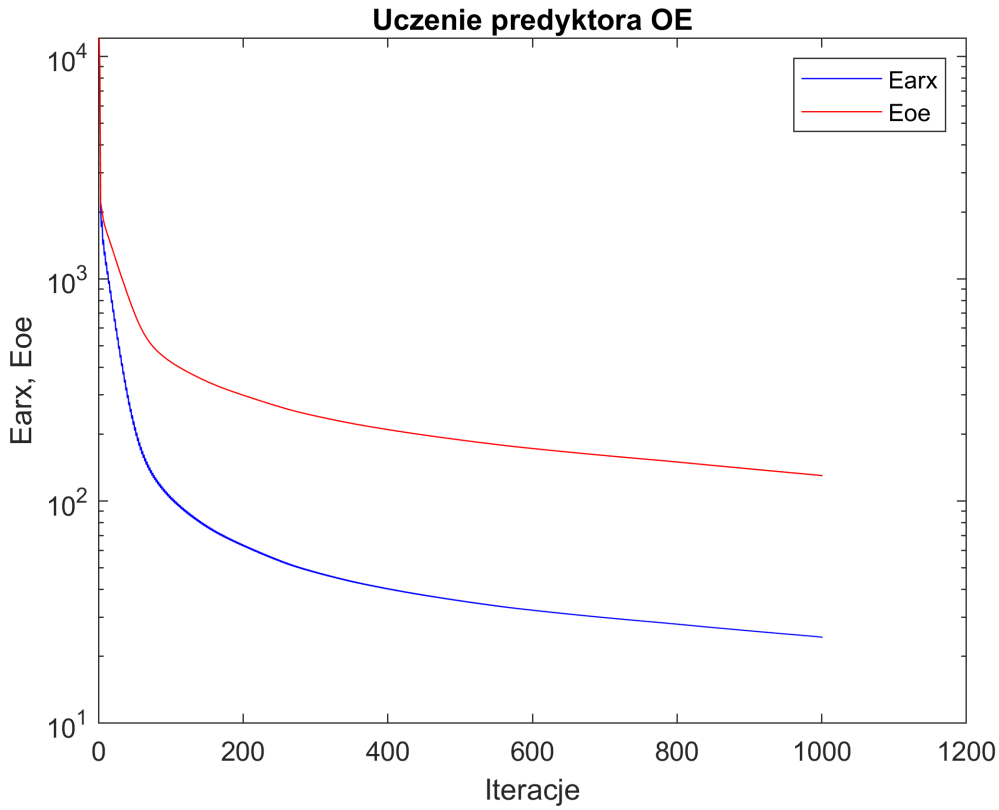
Rysunek 7: Porównanie wyjścia procesu z wyjściem modelu dla danych weryfikujących.



Rysunek 8: Wyjścia modelu na tle wyjść procesu dla danych weryfikujących.

3.5 Algorytm najszybszego spadku

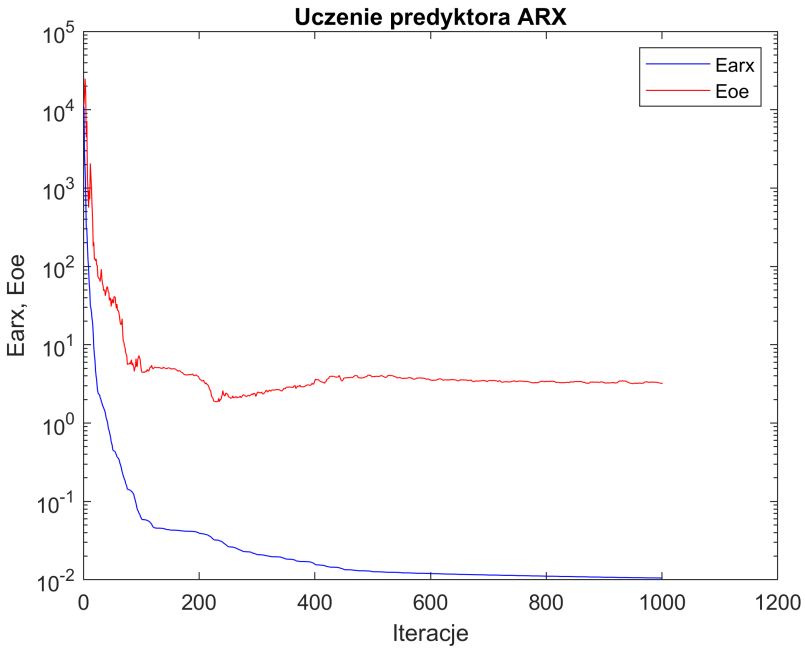
Najlepszy z modeli nauczonych algorytmem najszybszego spadku miał błąd w trybie rekurencyjnym dla zbioru weryfikującego równy 3.901022e-02.



Rysunek 9: Przebieg błędów podczas uczenia.

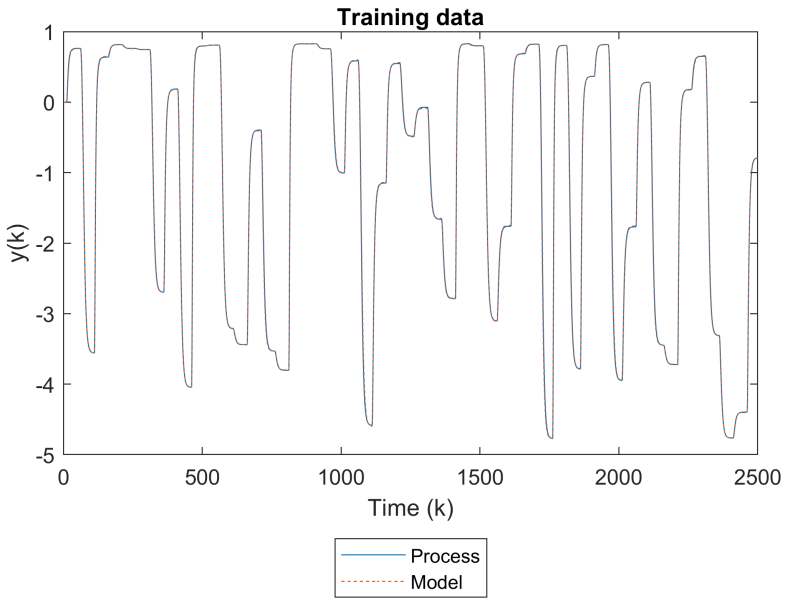
3.6 ARX

Najlepszy z modeli nauczonych algorytmem najszybszego spadku miał błąd w trybie bez rekurencji dla zbioru weryfikującego równy 4.428255e-06.

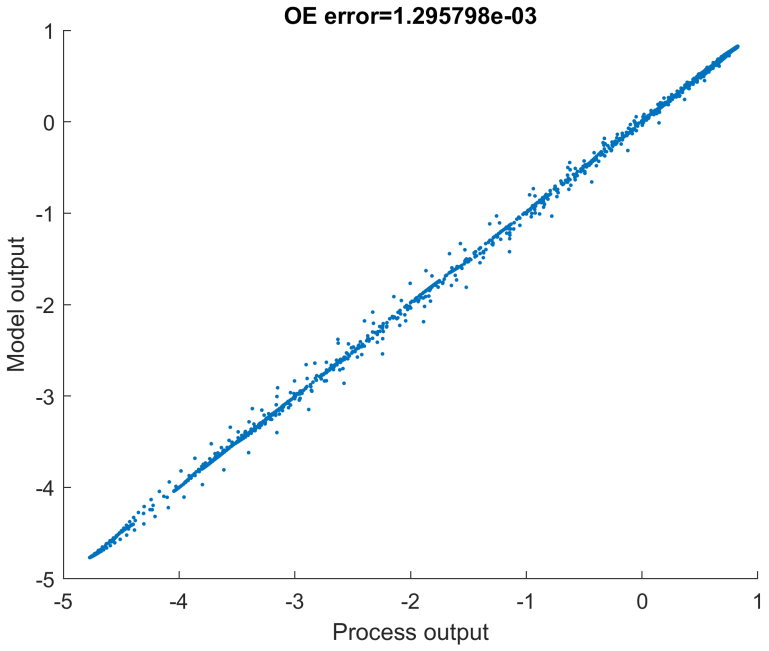


Rysunek 10: Przebieg błędów podczas uczenia.

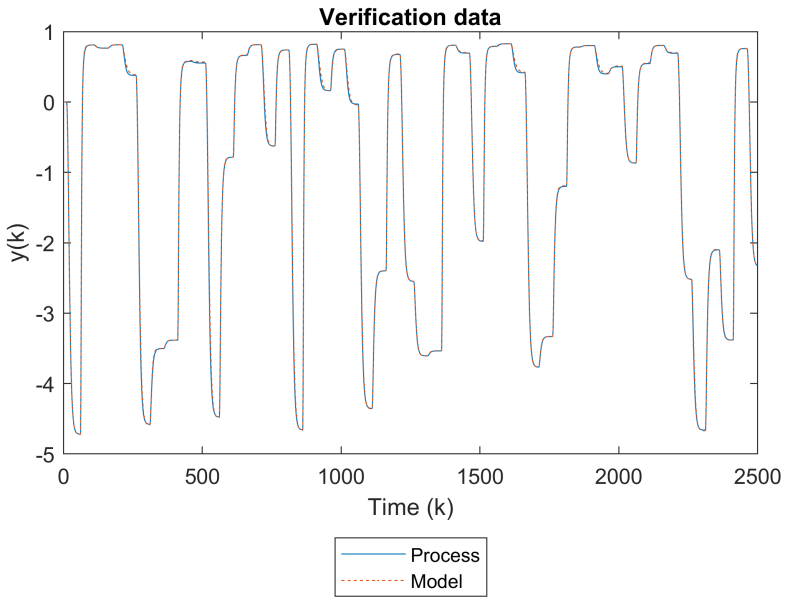
3.7 Porównanie modelu uczonego ARX z procesem



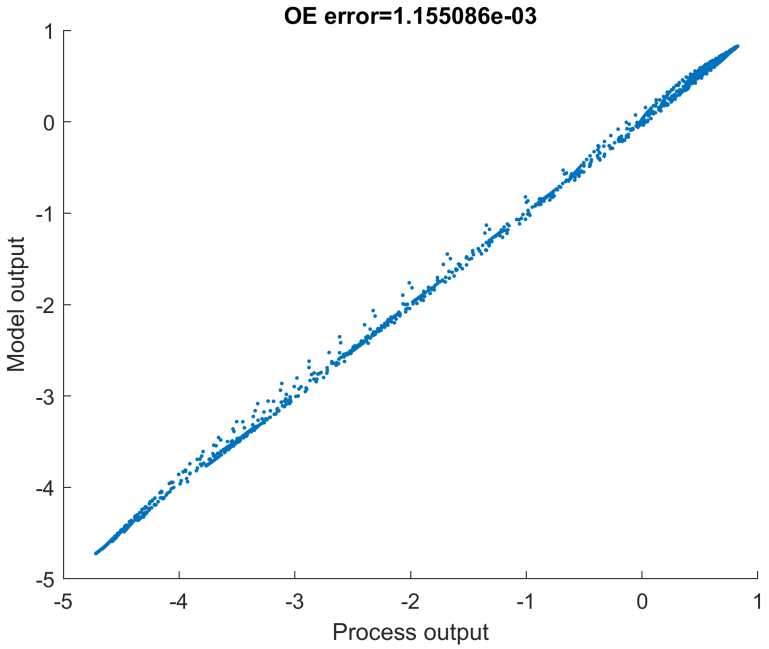
Rysunek 11: Porównanie wyjścia procesu z wyjściem modelu dla danych uczących.



Rysunek 12: Wyjścia modelu na tle wyjść procesu dla danych uczących.



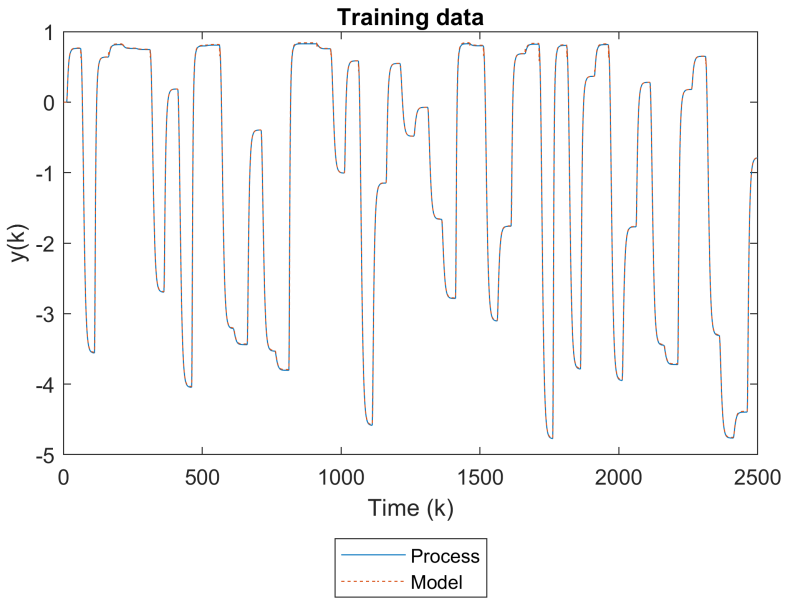
Rysunek 13: Porównanie wyjścia procesu z wyjściem modelu dla danych weryfikujących.



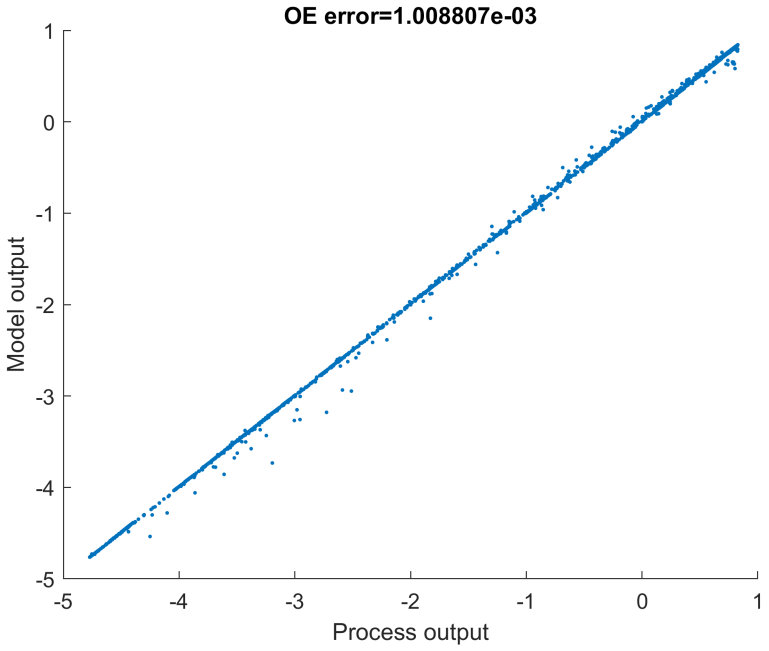
Rysunek 14: Wyjścia modelu na tle wyjść procesu dla danych weryfikujących.

Jak widać model uczony w trybie ARX ma znacznie większe problemy z poprawną symulacją procesu niż model uczony w trybie OE.

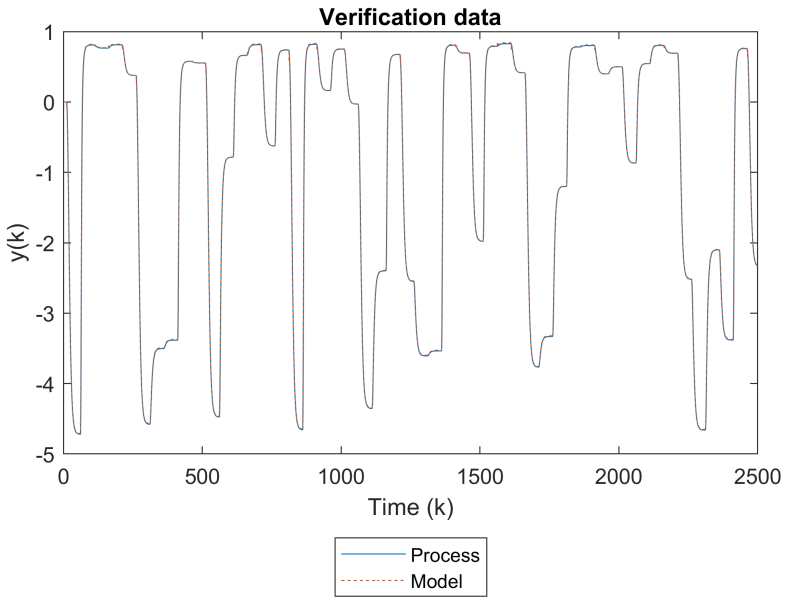
3.8 Model liniowy



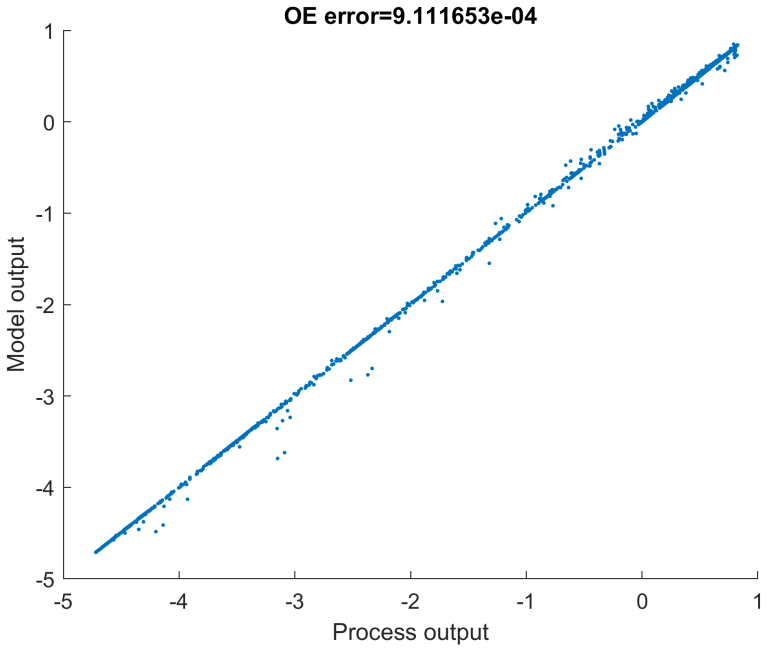
Rysunek 15: Porównanie wyjścia procesu z wyjściem modelu liniowego dla danych uczących.



Rysunek 16: Wyjścia modelu liniowego na tle wyjść procesu dla danych uczących.



Rysunek 17: Porównanie wyjścia procesu z wyjściem modelu liniowego dla danych weryfikujących.



Rysunek 18: Wyjścia modelu liniowego na tle wyjść procesu dla danych weryfikujących.

Model liniowy wyszedł całkiem niezły tym bardziej, że nie wymagał zbyt wielu obliczeń.

4 Modelowanie za pomocą przyborników programu MatLaba

4.1 Wybór przybornika

Do modelowania procesu za pomocą sieci neuronowych wewnątrz matlaba wybrany został przybornik "Deep Learning Toolbox". Dostarcza on między innymi struktury sieci neuronowej wraz ze wszystkimi istotnymi o niej informacjami, takimi jak liczba warstw i wagi poszczególnych neuronów, a także narzędzi do ich uczenia.

4.2 Uczenie modeli neuronowych za pomocą przybornika MatLaba

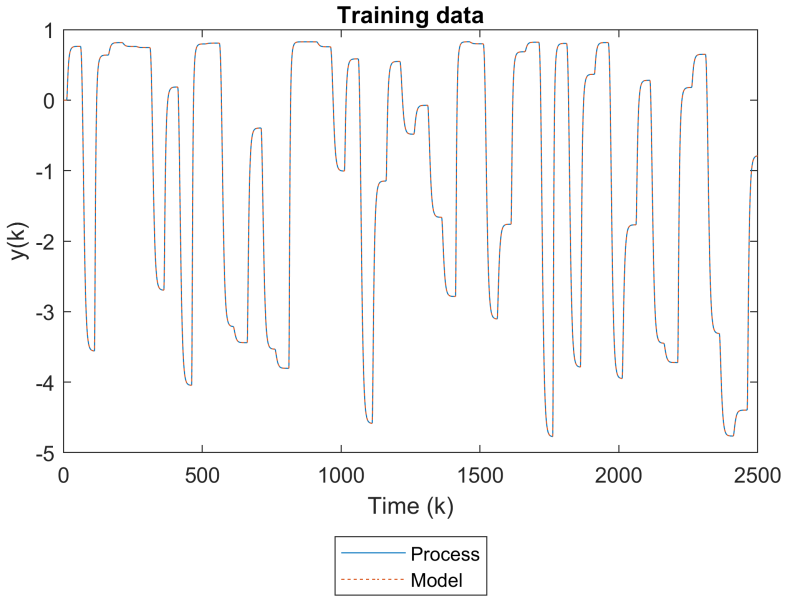
Uczenie przeprowadzono dla 1000 iteracji (pominięto ograniczenie błędu granicznego). Nauczono 10 modeli neuronowych w trybie bez rekurencji. Połowę uczono algorytmem skalowanych gradientów sprzężonych, a drugą algorytmem Levenberga-Marquardta. W poniższej tabeli znajdują się najlepsze błędy modeli z tej puli. Dla algorytmu skalowanych gradientów sprzężonych są to błędy dwóch różnych modeli, a dla algorytmu Levenberga-Marquardta są to błędy jednego modelu.

Algorytm	Średniokwadratowy błąd ARX	Średniokwadratowy błąd OE
SCG	4.342090e-05	3.942245e-03
LM	4.901803e-06	2.569114e-04

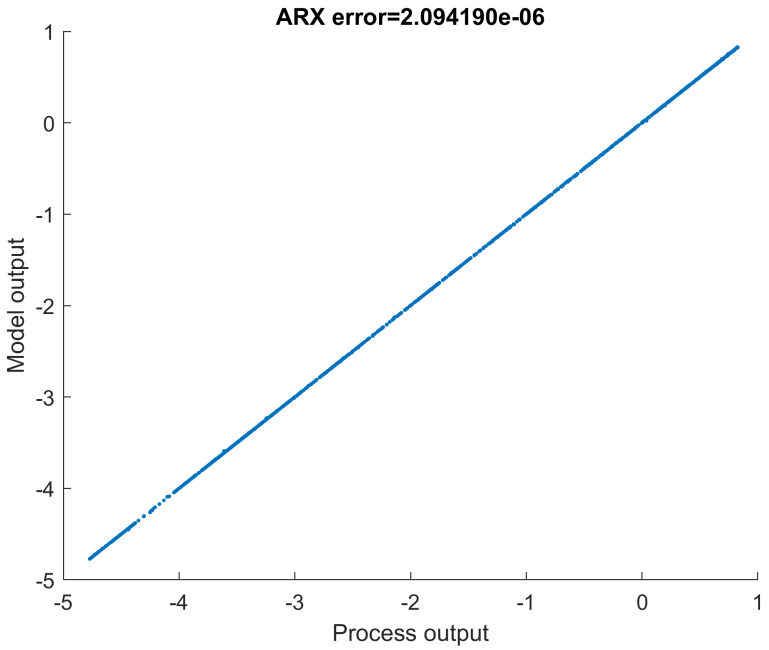
Z tabelki widać wyraźnie, że najlepszym modelem jest tu jeden z modeli uczonych metodą Levenberga-Marquardta.

4.3 Symulacja najlepszego modelu

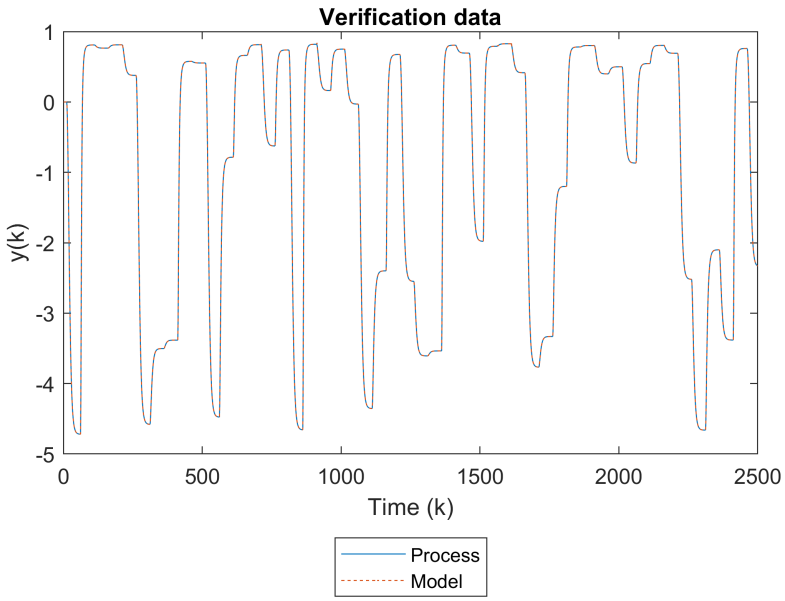
4.3.1 Tryb bez rekurencji



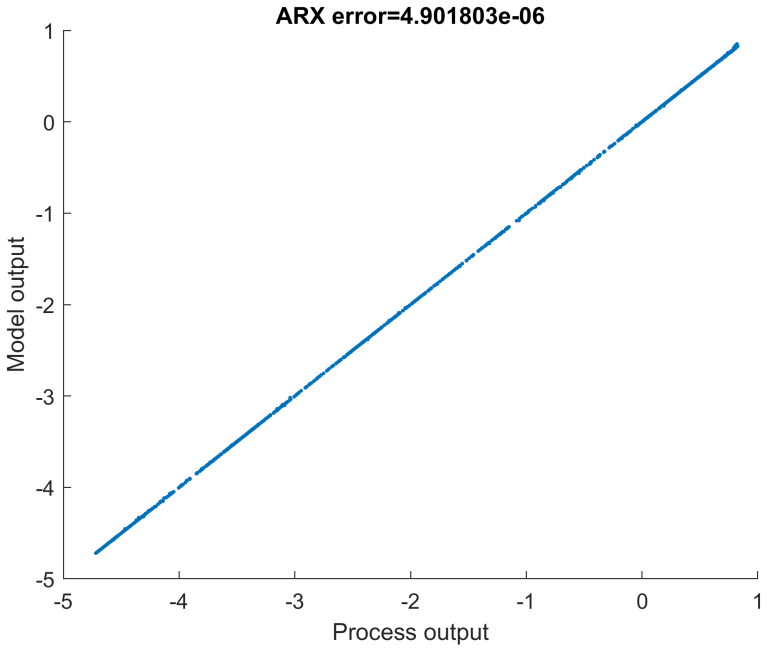
Rysunek 19: Porównanie wyjścia procesu z wyjściem modelu dla danych uczących.



Rysunek 20: Wyjścia modelu na tle wyjść procesu dla danych uczących.

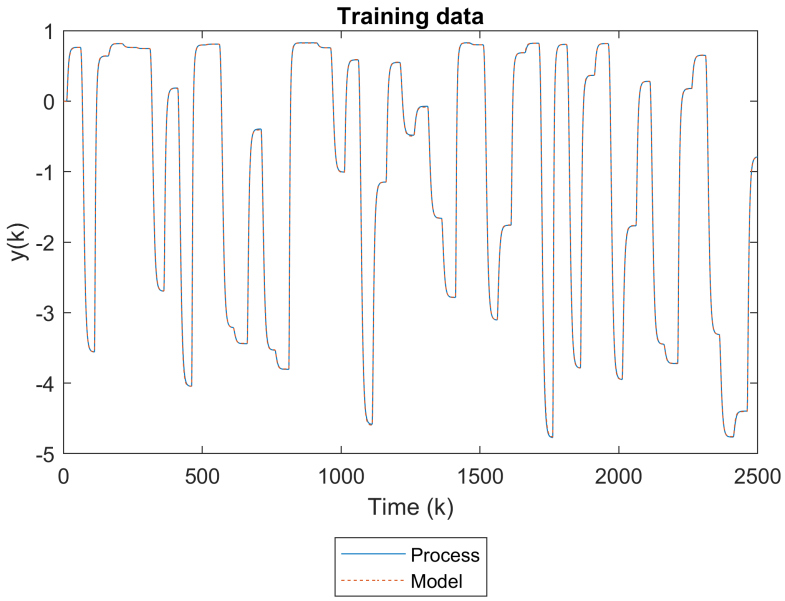


Rysunek 21: Porównanie wyjścia procesu z wyjściem modelu dla danych weryfikujących.

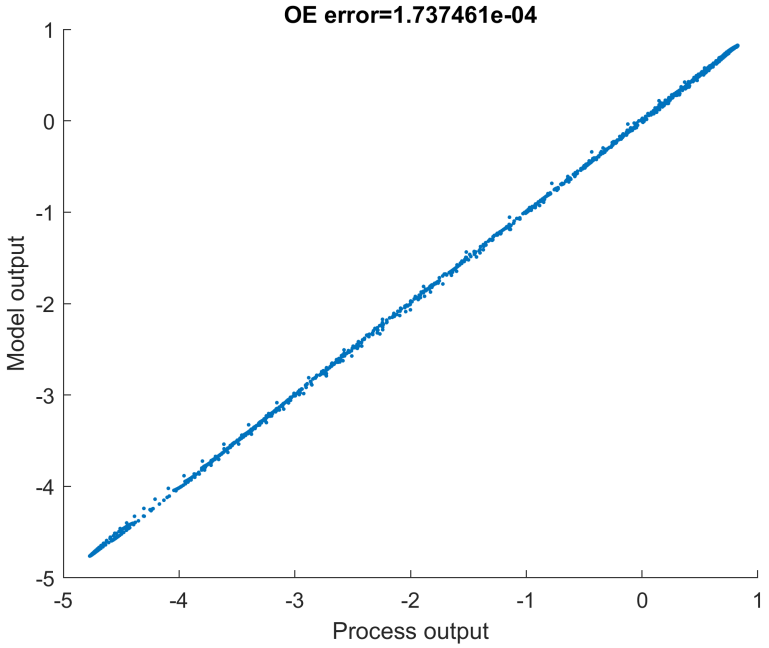


Rysunek 22: Wyjścia modelu na tle wyjść procesu dla danych weryfikujących.

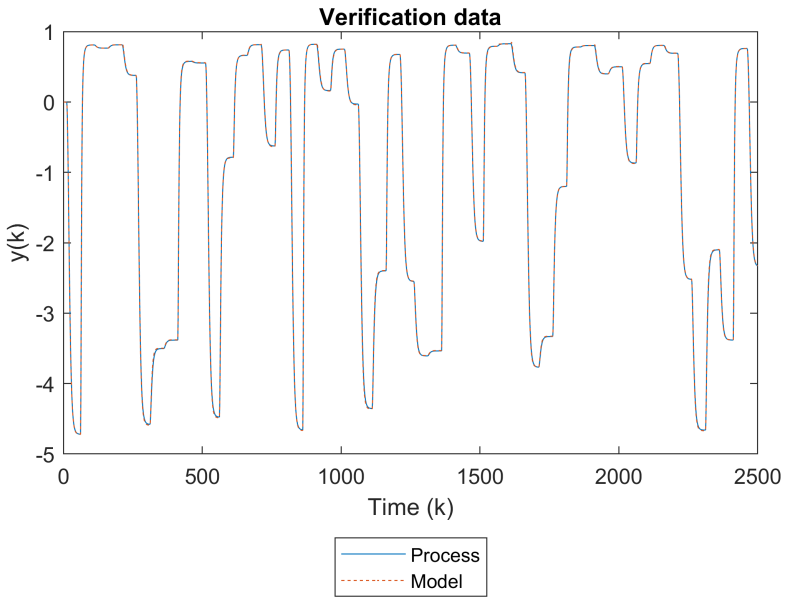
4.3.2 Tryb z rekurencją



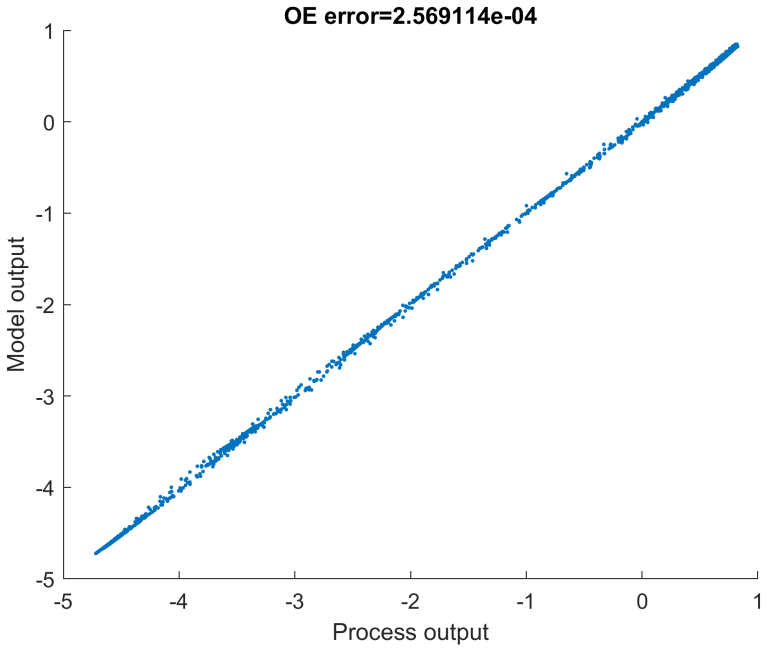
Rysunek 23: Porównanie wyjścia procesu z wyjściem modelu dla danych uczących.



Rysunek 24: Wyjścia modelu na tle wyjść procesu dla danych uczących.



Rysunek 25: Porównanie wyjścia procesu z wyjściem modelu dla danych weryfikujących.



Rysunek 26: Wyjścia modelu na tle wyjść procesu dla danych weryfikujących.

4.4 Porównanie powyższego modelu z modelem uczonym algorytmem BFGS

Model uczony algorytmem LM był na pewno lepszy od najlepszego modelu uzyskanego algorytmem BFGS. Pomimo, że był uczony bez rekurencji miał mniejszy błąd w trybie rekurencyjnym, uczył się też dużo dużo krócej, mimo tej samej liczby iteracji.

5 Regulacja procesu

5.1 Algorytm regulacji predykcyjnej NPL

Zaimplementowano algorytm w wersji analitycznej. Wszystkie wykresy znajdują się na końcu sekcji.

5.2 Strojenie i regulacja

Po kilku próbach wartości nastaw algorytmu wyznaczono na: $N = 20$, $N_u = 2$, $\lambda = 30$. Nastawy te były dobierane na jak najmniejsze przy utrzymaniu zadowalającej jakości regulacji. Wartości zadane zostały wygenerowane losowo z przedziału od -4.8 do 0.8.

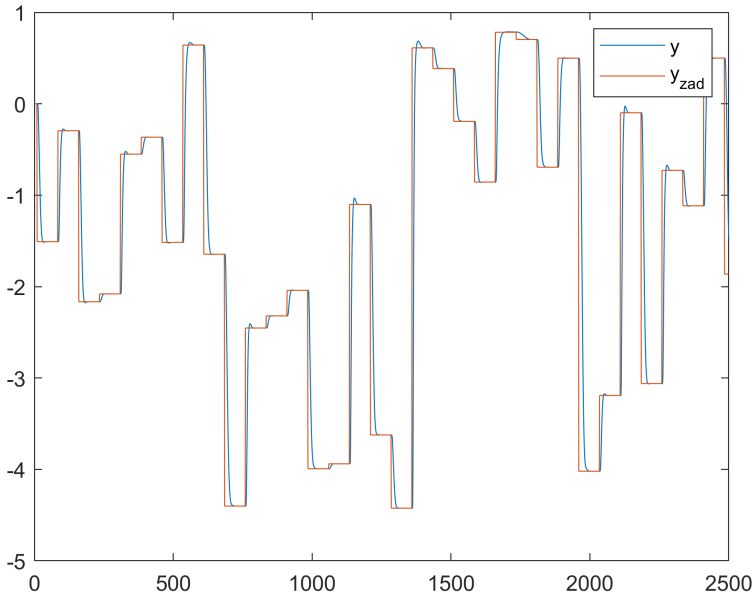
5.3 Algorytm GPC z modelem liniowym

Niestety, algorytm GPC z modelem liniowym nie był w stanie sobie poradzić z regulacją procesu przy takich samych nastawach, jak regulator NPL.

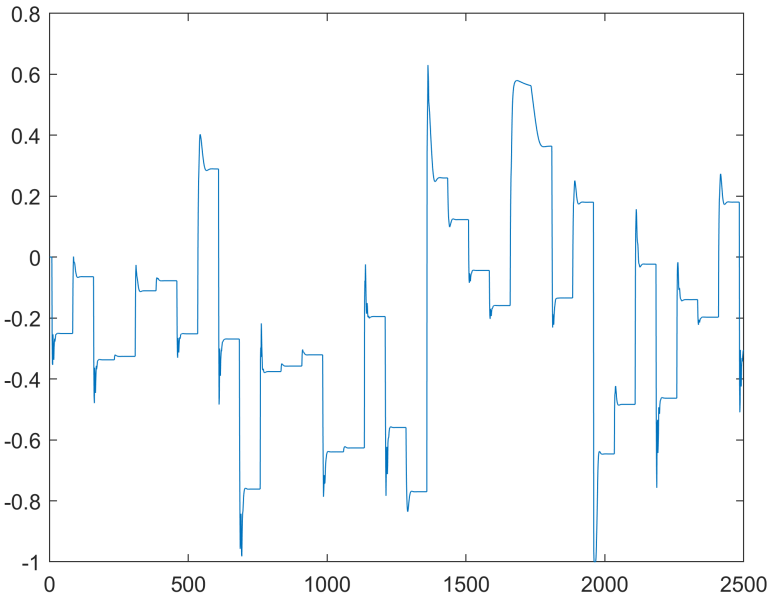
Sensowne sterowanie bez dużych oscylacji wyjścia osiągnięto dopiero po zwiększeniu horyzontu sterowania do 20, oraz λ do 1000. Przy takich nastawach regulator ma jednak problemy z nadążaniem z regulacją.

Dla takich ustawień regulator NPL również nie nadąża z regulacją, jednak sterowanie przez niego wyliczane jest bardziej stabilne.

Nastawy: $N = 20,\ N_u = 2,\ \lambda = 30$

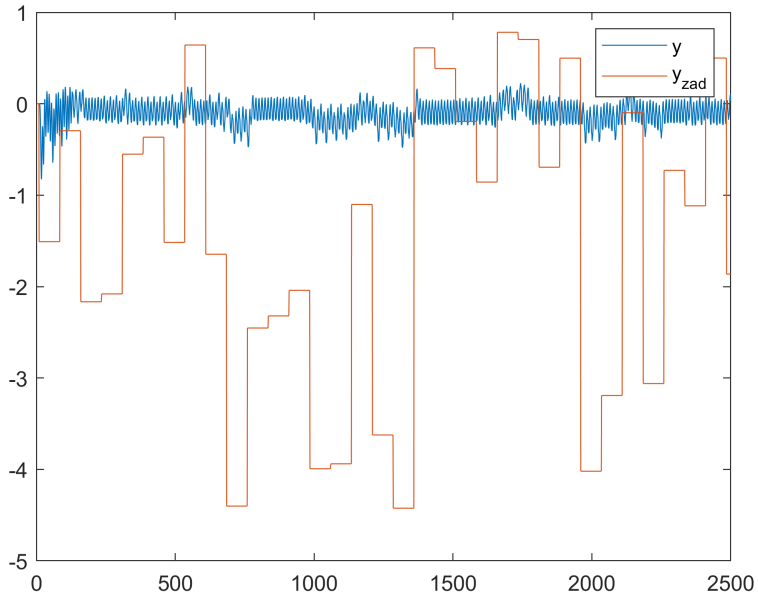


Rysunek 27: Wyjście procesu regulowanego NPL i wartość zadana.

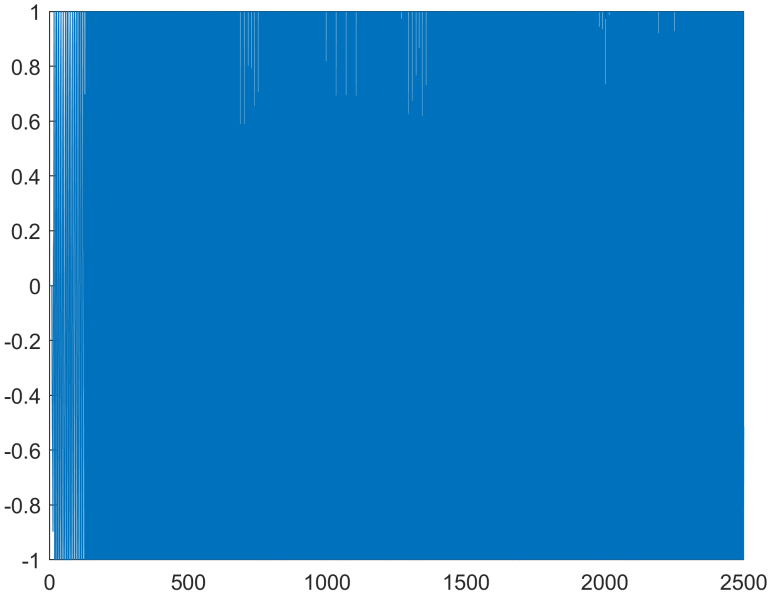


Rysunek 28: Sterowanie procesu NPL.

Nastawy: $N = 20,\ N_u = 2,\ \lambda = 30$

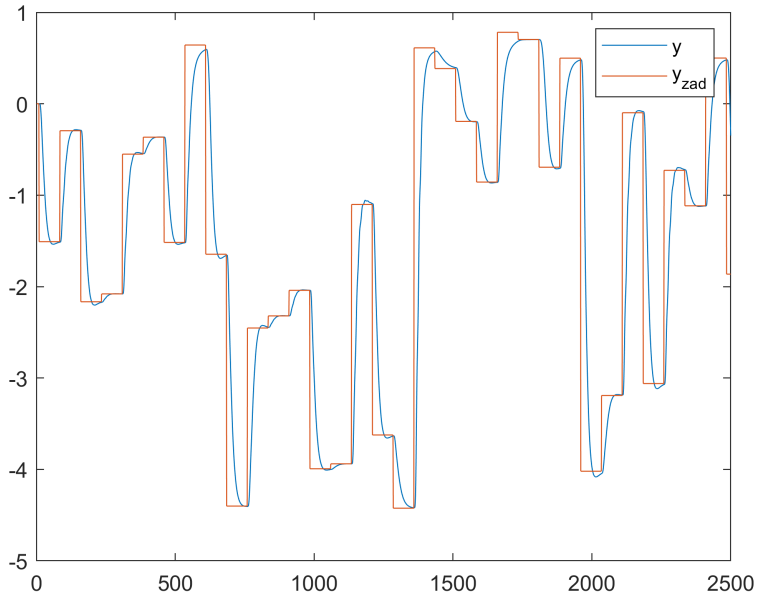


Rysunek 29: Wyjście procesu regulowanego GPC i wartość zadana.

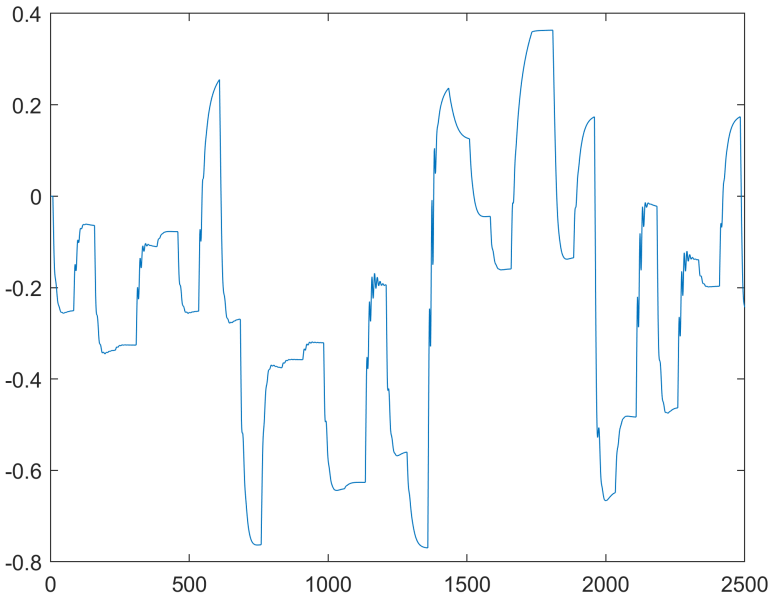


Rysunek 30: Sterowanie procesu GPC.

Nastawy: $N = 20,\ N_u = 20,\ \lambda = 1000$

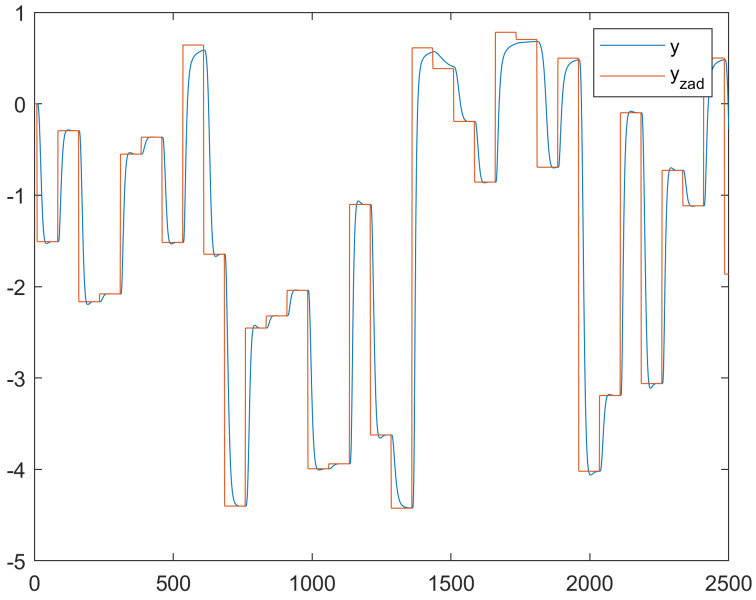


Rysunek 31: Wyjście procesu regulowanego GPC i wartość zadana dla zmienionych nastaw.

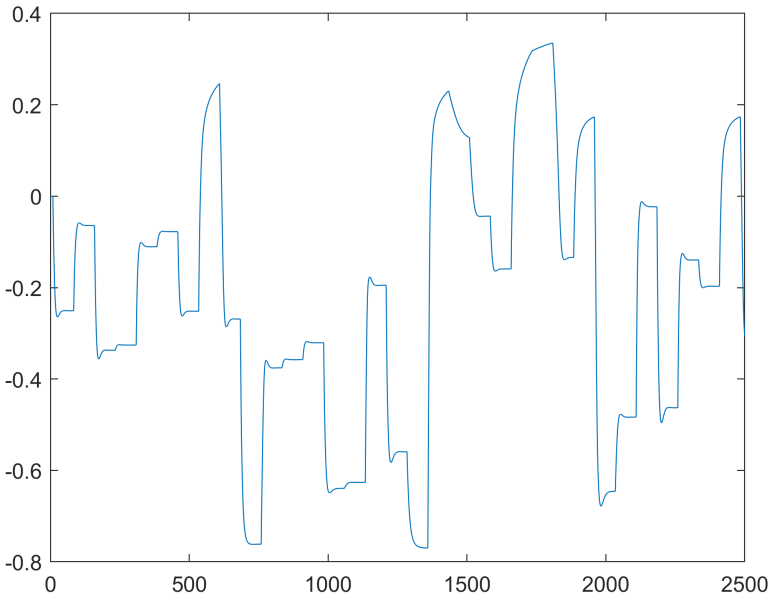


Rysunek 32: Sterowanie procesu GPC dla zmienionych nastaw.

Nastawy: $N = 20,\ N_u = 20,\ \lambda = 1000$



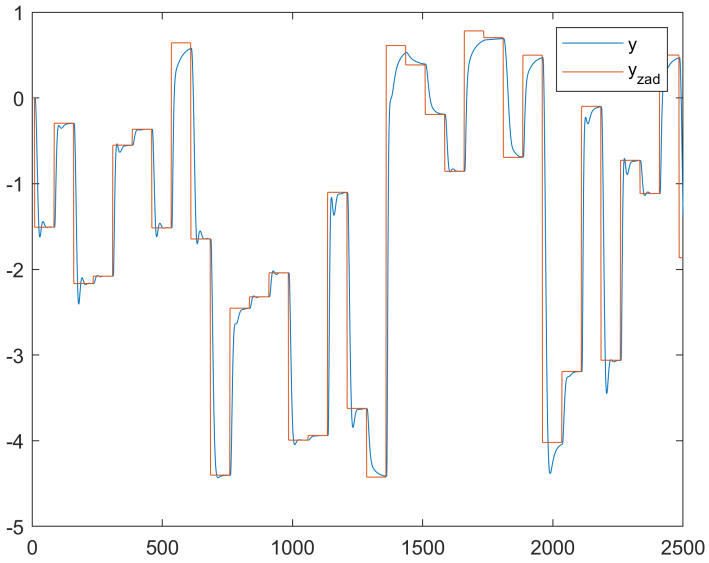
Rysunek 33: Wyjście procesu regulowanego NPL i wartość zadana dla zmienionych nastaw.



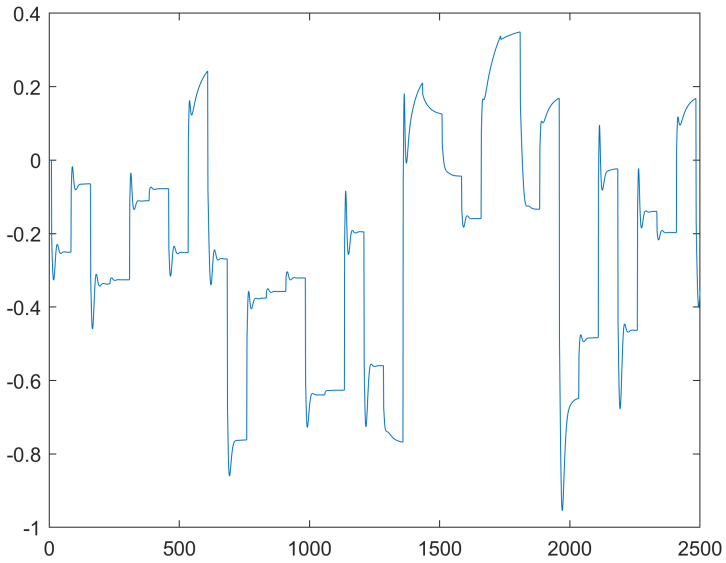
Rysunek 34: Sterowanie procesu NPL dla zmienionych nastaw.

5.4 PID (zadanie dodatkowe)

Jak można się było spodziewać, jakość regulacji PID'a nie jest zbyt dobra. Po przeprowadzeniu eksperymentu Zieglera-Nicholsa wyznaczono $K_k = 1$ i $T_k = 15$. Nastawy Zieglera-Nicholsa nie działały w ogóle, więc metodą prób i błędów wyznaczono nastawy: $K = 0.125K$, $T_s = 0.5T_k$ i $T_d = 0.01T_k$. Powodem problemów z regulatorem PID jest oczywiście nieliniowość procesu.



Rysunek 35: Wyjście procesu regulowanego PID.



Rysunek 36: Sterowanie procesu PID.