Sprawozdanie z drugiego projektu z przedmiotu „Sztuczna Inteligencja w Automatyce”

Numer zadania: 10

Wykonawcy:
Daniel Giełdowski
Piort Chachuła

Spis treści

1. Symulacja procesu

1.1. Charakterystyka statyczna

Zadany układ opisany jest równaniami:

$$\begin{cases} x_1(k) = -\alpha_1 x_1(k-1) + x_2(k-1) + \beta_1 g_1(u(k-3)) \\ x_2(k) = -\alpha_2 x_1(k-1) + \beta_2 g_1(u(k-3)) \\ y(k) = g_2(x_1(k)) \end{cases} \tag{1.1}$$

gdzie u-sygnał wejściowy, y-sygnał wyjściowy, x_1, x_2 - zmienne stanu, $\alpha_1 = -1,422574$, $\alpha_2 = 0,466776$, $\beta_1 = 0,017421$, $\beta_2 = 0,013521$ oraz

$$g_1(u(k-3)) = \frac{exp(5u(k-3)) - 1}{exp(5u(k-3)) + 1}, \quad g_2(x_1(k)) = 1 - exp(-1.5x_1(k)) \tag{1.2}$$

Podany punkt pracy układu to $u = y = x_1 = x_2 = 0$, więc w wersji statycznej:

$$\begin{cases} x_1 = -\alpha_1 x_1 + x_2 + \beta_1 g_1(u) \\ x_2 = -\alpha_2 x_1 + \beta_2 g_1(u) \\ y = g_2(x_1) \end{cases} \tag{1.3}$$

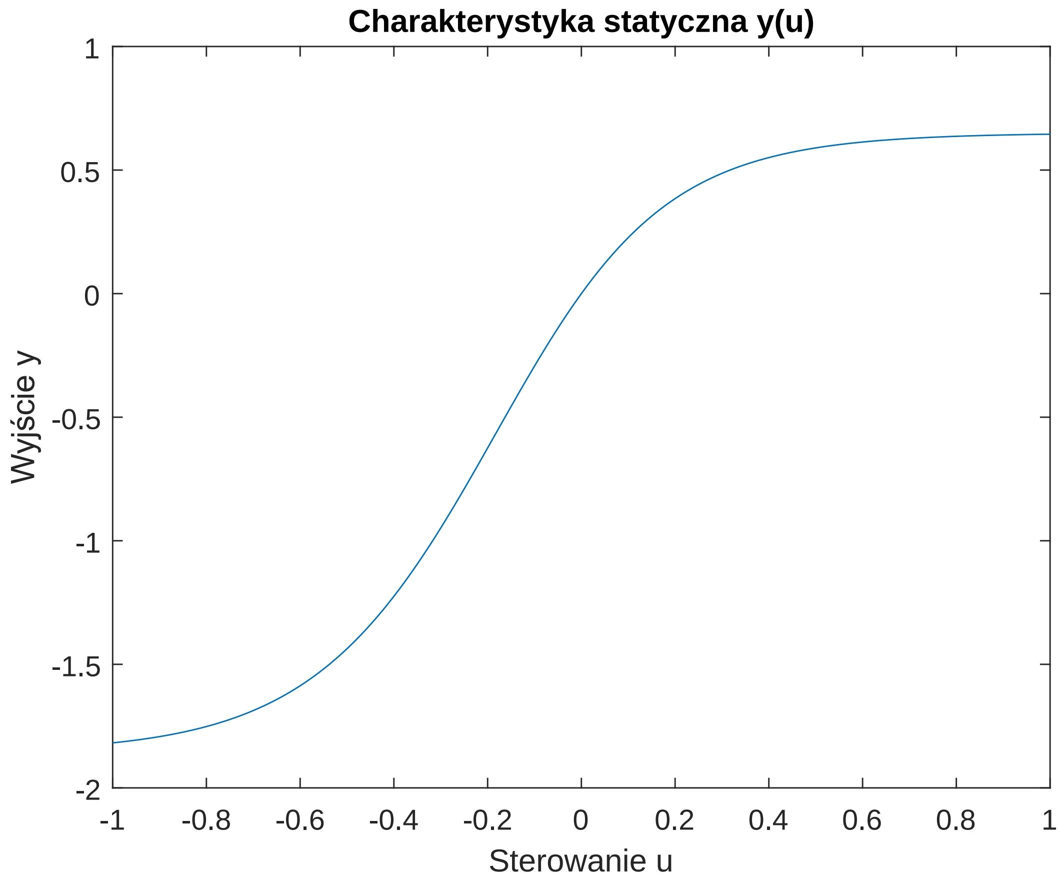
Po przekształceniach:

$$x_1 = \frac{(\beta_1 + \beta_2) g_1(u)}{1 + \alpha_1 + \alpha_2} \tag{1.4}$$

Po podstawieniu równania (1.4) do y otrzymujemy

$$y(u) = g_2\left(\frac{(\beta_1 + \beta_2) g_1(u)}{1 + \alpha_1 + \alpha_2}\right) \tag{1.5}$$

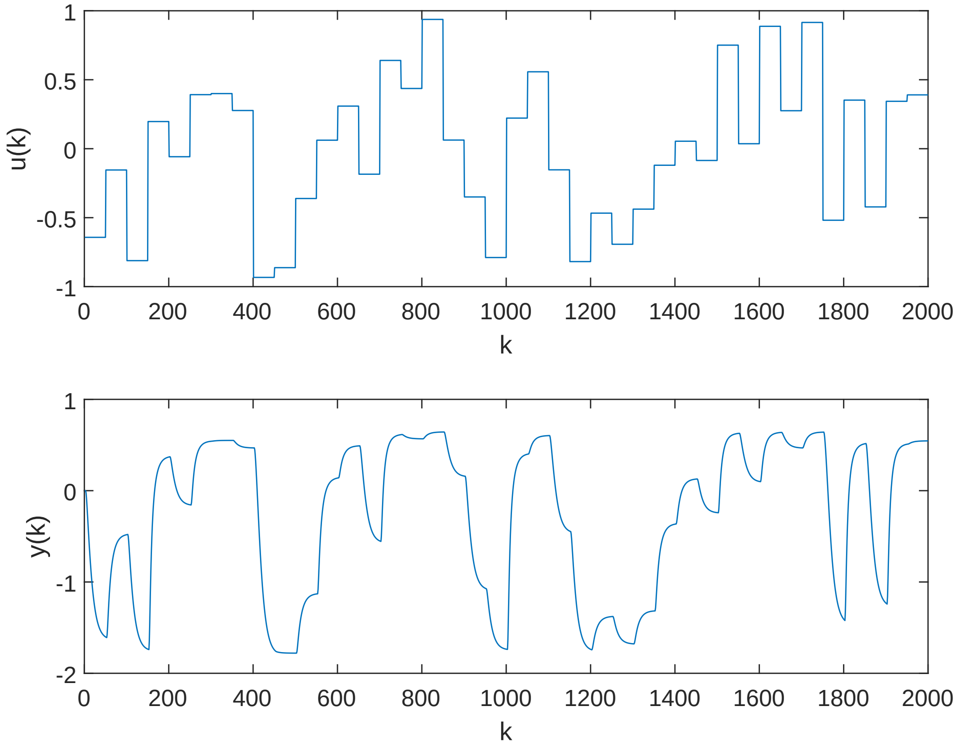
Wykres wyznaczonej charakterystyki statycznej dla zadanego zakresu wartości sterowania ($u^{min} = -1, u^{max} = 1$) przedstawiony został na wykresie 1.1. Wykres został wygenerowany za pomocą skryptu *charakterystyka_statyczna.m*.



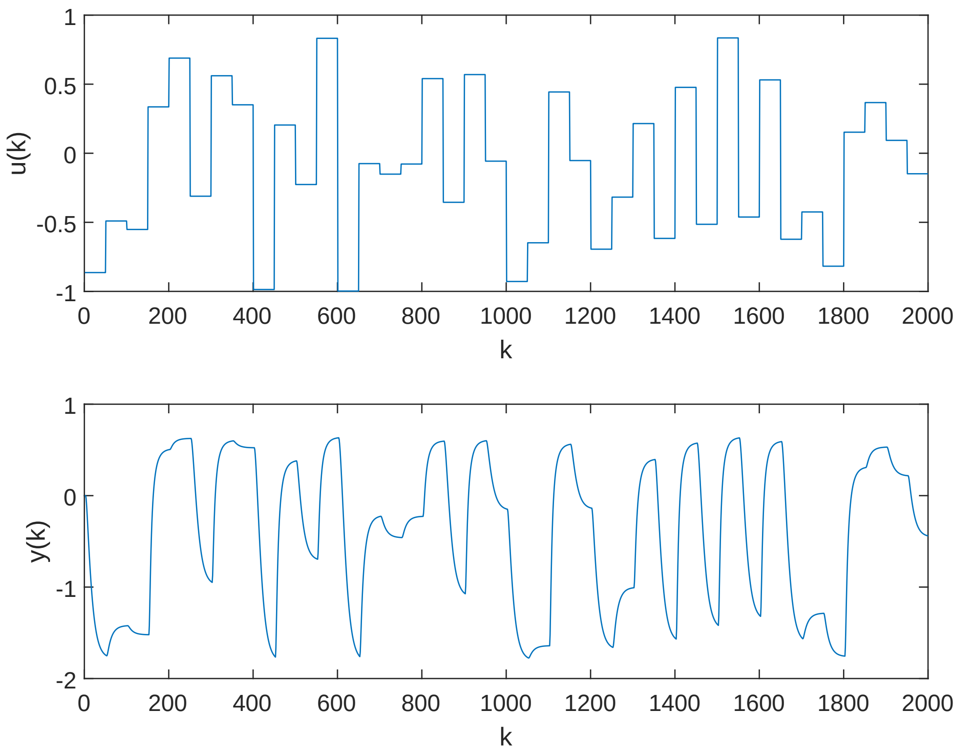
Rys. 1.1. Charakterystyka statyczna procesu

1.2. Zbiory danych

W celu przygotowania do uczenia sieci neuronowych wygenerowaliśmy dwa zbiory danych. Dane zostały wygenerowane poprzez zasymulowanie zadanego procesu dla sygnału sterowania złożonego o wartości zmieniającej się skokowo co 50 próbek. Obydwa zbiory danych mają po 2000 próbek. Zostały one przedstawione na wykresach 1.2 i 1.3. Użyte zostały skrypty: *generowanie_danych.m* (do wygenerowania danych) oraz *wykres_dancyh.m* (do narysowania wykresów).



Rys. 1.2. Dane uczące

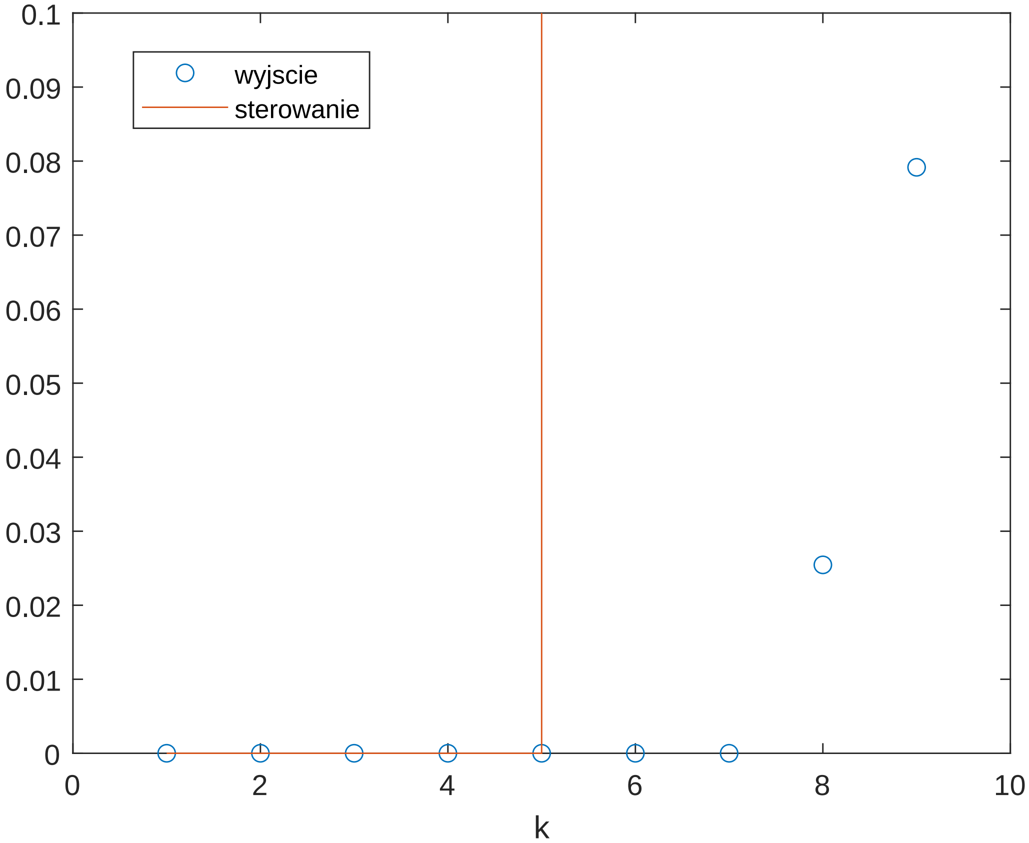


Rys. 1.3. Dane weryfikujące

2. Modelowanie procesu

2.1. Opóźnienie

W celu zdefiniowania opóźnienia τ procesu zasymulowaliśmy go dla pojedynczego skoku sterowania. Wyniki symulacji przedstawione są na wykresie 2.1. Skok sterowania nastąpił w 5 kroku działania programu, natomiast wyjście procesu zmieniło się dopiero w kroku 8. Oznacza to, że poszukiwane przez nas opóźnienie wynosi $\tau = 3$. Użyty przez nas skrypt to *tauwiz.m*.



Rys. 2.1. Wizualizacja opóźnienia procesu

2.2. Dobór liczby neuronów

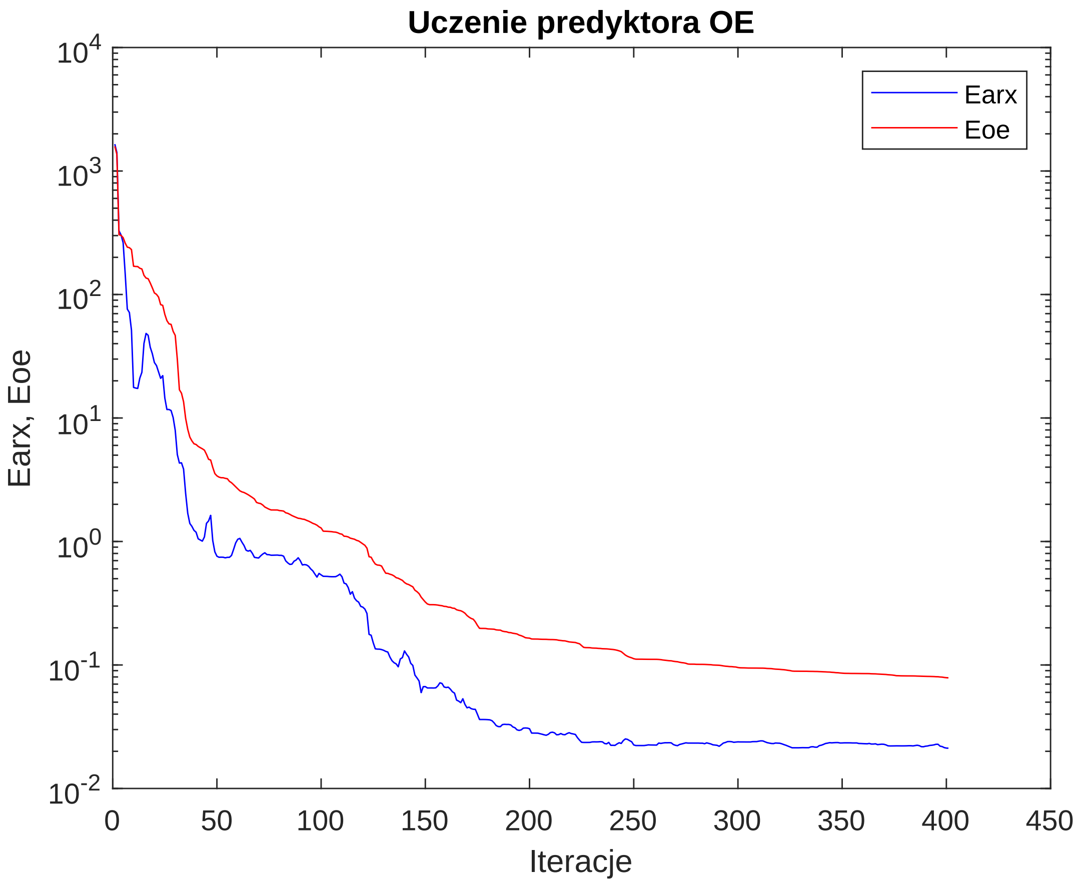
W celu dobrania odpowiedniej liczby neuronów dla sieci zastosowaliśmy wielokrotne uczenie z użyciem programu *sieci.exe*. Dla każdej ilości neuronów ukrytych od 1 do 10 dokonaliśmy 5 procesów uczenia za pomocą algorytmu BFGS z wykorzystaniem rekurencji. W tym celu wykorzystaliśmy skrypt *modelowanie.m*. Najmniejszy uzyskany błąd uczenia wraz ze skojarzonym z nim błędem weryfikacji przedstawiony został w tabeli poniżej. Najmniejszy błąd dla obydwu zbiorów występuje dla 9 neuronów. Ostatecznie jednak zdecydowaliśmy się na używanie sieci z pięcioma neuronami ukrytymi. Powodem tego jest mała poprawa w stosunku do większych ilości neuronów oraz chęć zmniejszenia nakładu obliczeń. Dodatkowo sieci o zbyt dużej ilości neuronów ukrytych mają tendencję do przetrenowywania się, w wyniku którego sieć przystosowuje się nie tyle do procesu co do samych danych uczących.

Liczba neuronów	Błąd uczenia	Błąd weryfikacji
1	3.070626e+01	5.548815e+01
2	4.977413e-01	1.060318e+00
3	3.206039e-01	5.111444e-01
4	1.479096e-01	2.625729e-01
5	8.734595e-02	1.534512e-01
6	7.765994e-02	2.087909e-01
7	2.614618e-02	1.727668e-01
8	1.509561e-02	1.095385e-01
9	1.355132e-02	6.725641e-02
10	2.105601e-02	1.136051e-01

Tab. 2.1. Błędy modelu dla różnej ilości neuronów

2.3. Model z algorytmu BFGS

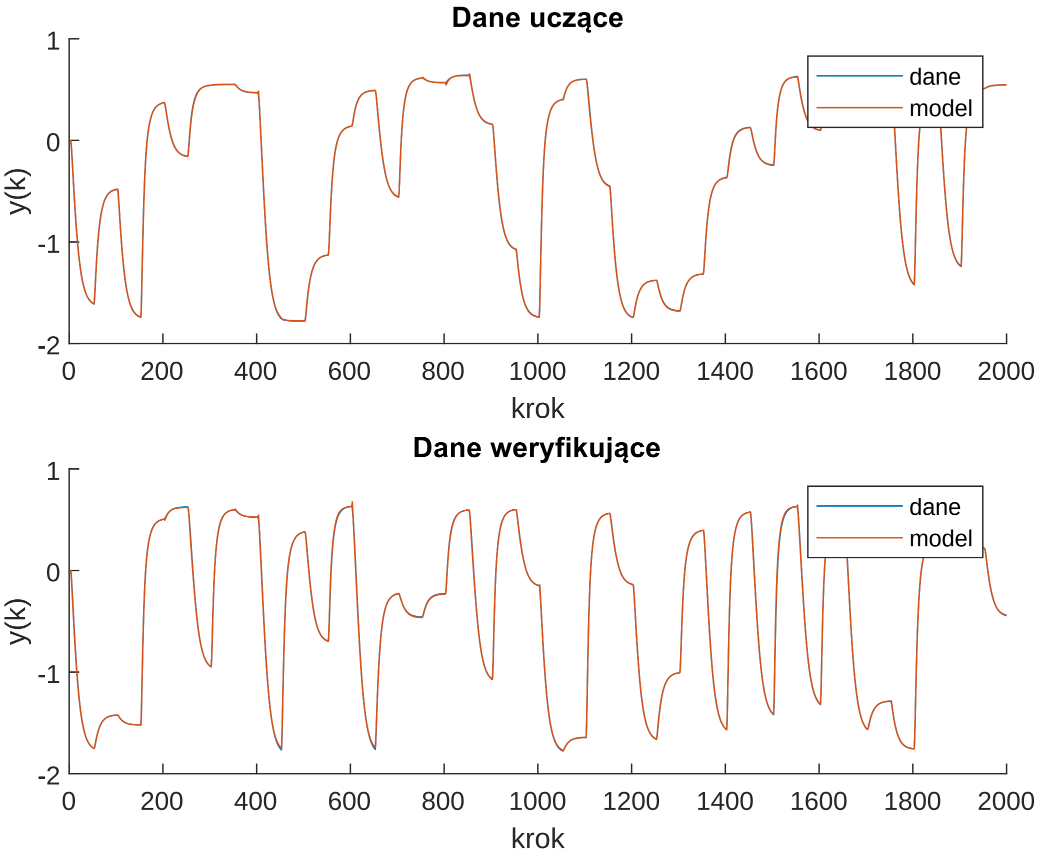
Na wykresie 2.2 przedstawione zostały błędy predykatorów ARX i OE dla kolejnych iteracji uczenia modelu. Zgodnie z ustaleniami z poprzednich punktów zastosowane zostały następujące parametry: $tau = 3$, $neurony\ ukryte = 5$. Końcowe błędy dla obydwu predykatorów wynosiły odpowiednio: $Eoe = 0.0787$ i $Earx = 0.0212$. Jak widać błędy te są dosyć małe jak na 2000 próbek co oznacza, że sieć dobrze nauczyła się modelu.



Rys. 2.2. Zmiany błędów predykatora ARX i OE dla kolejnych iteracji uczenia modelu algorytmem BFGS z użyciem rekurencji

2.4. Symulacja modelu z algorytmu BFGS

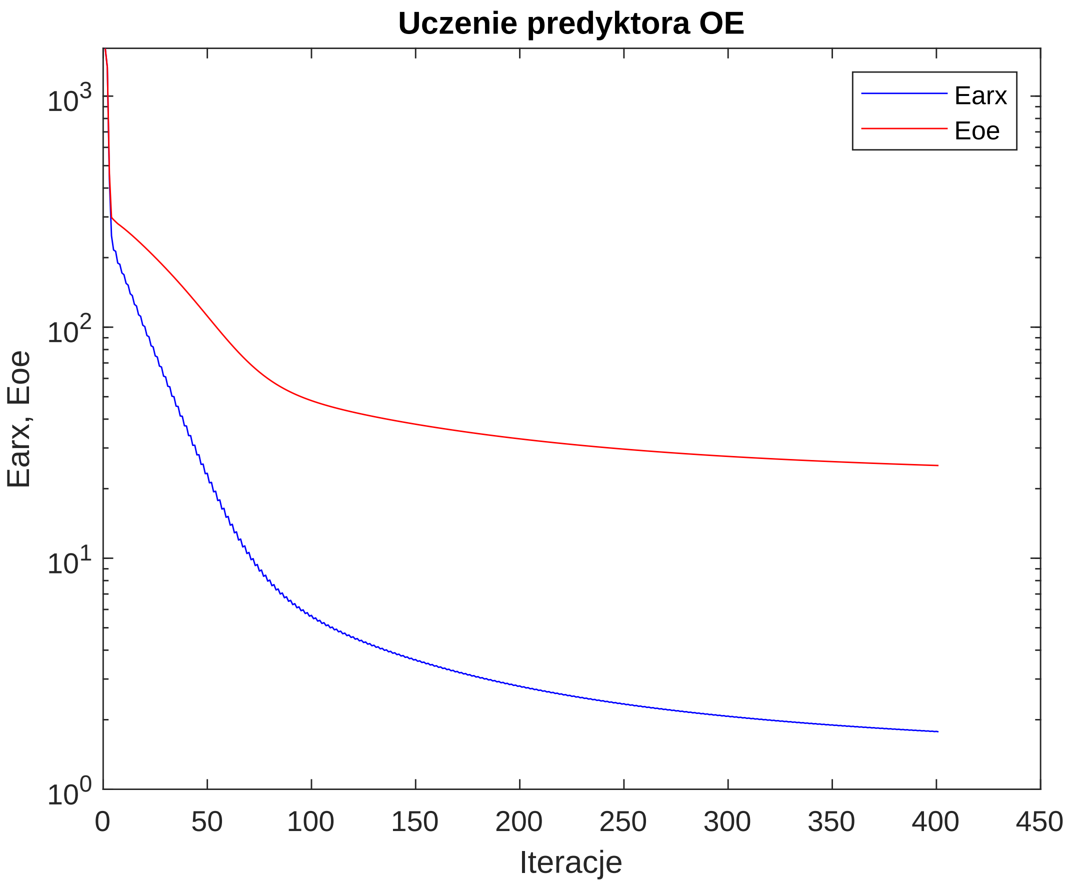
Model z poprzedniego punktu został zasymulowany w trybie rekurencyjnym dla uczącego oraz weryfikującego zboru danych. Eucz = 0.0787 Ewer = 0.2172



Rys. 2.3. Symulacja modelu uczonego algorytmem BFGS z rekurencją na danych uczących i weryfikujących

2.5. Model z algorytmu najszybszego spadku

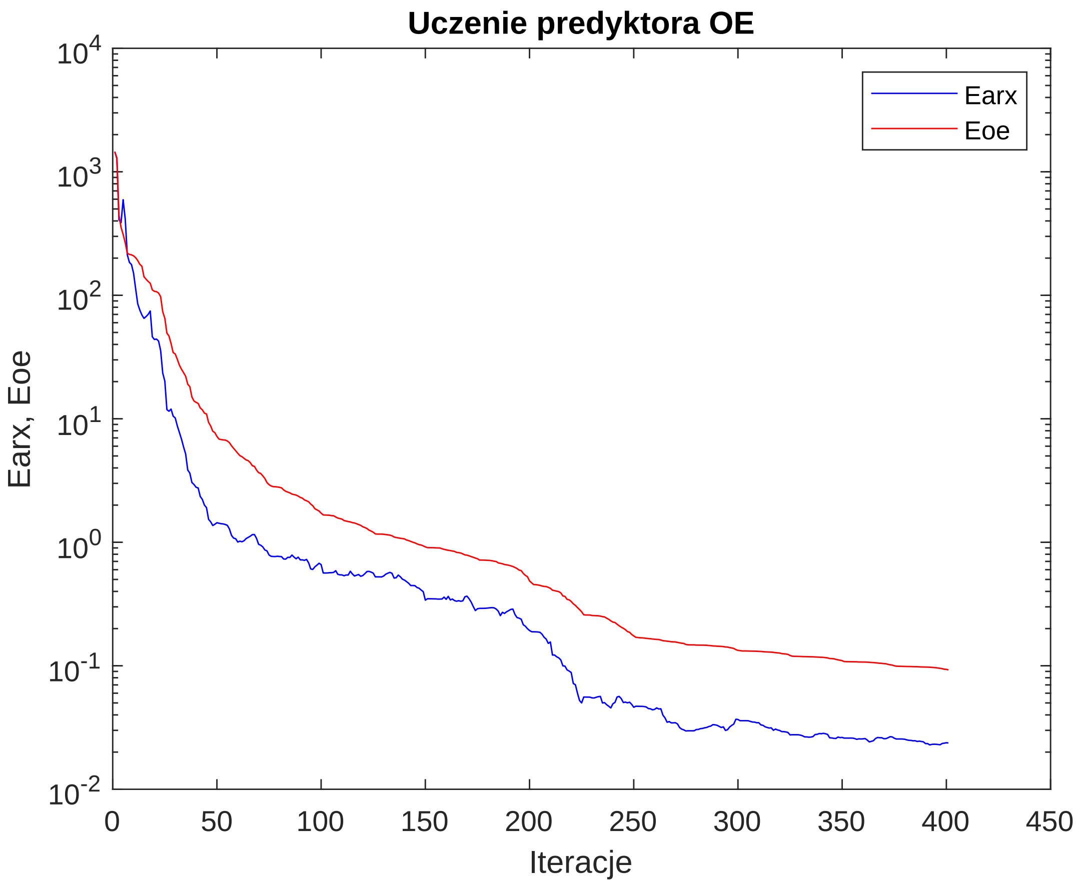
Eoe = 25.1864 Earx = 1.7730



Rys. 2.4. Zmiany błędów predykatora ARX i OE dla kolejnych iteracji uczenia modelu algorytmem najszybszego spadku z użyciem rekurencji

2.6. Model z algorytmu BFGS z uczeniem bez rekurencji

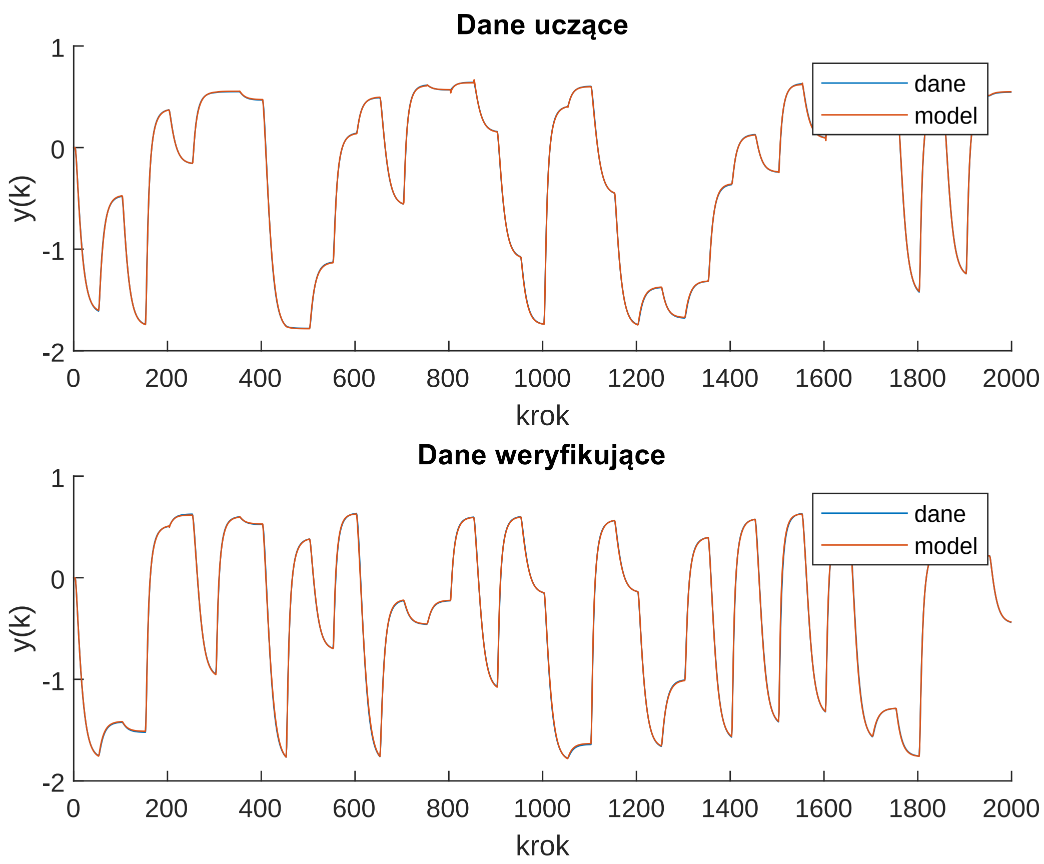
Eoe = 0.0925 Earx = 0.0237



Rys. 2.5. Zmiany błędów predykatora ARX i OE dla kolejnych iteracji uczenia modelu algorytmem BFGS bez użycia rekurencji

2.7. Symulacja modelu z algorytmu BFGS z uczeniem bez rekurencji

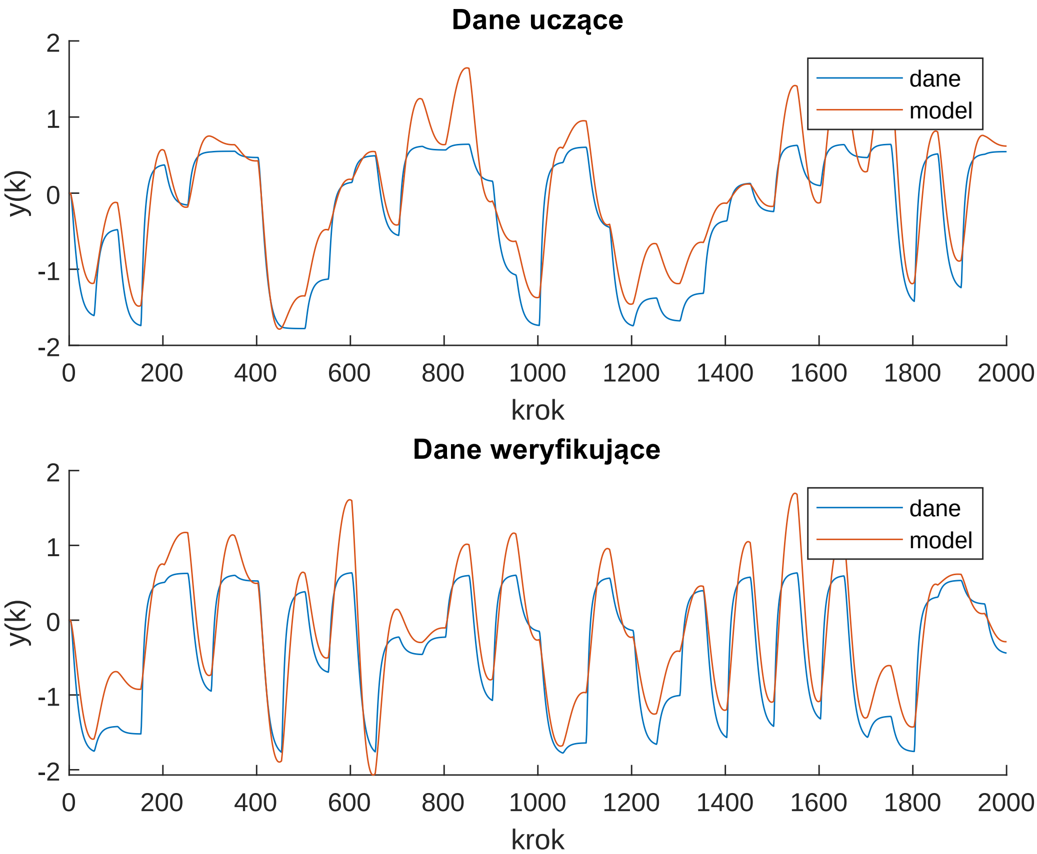
Eucz = 0.0925 Ewer = 0.3542



Rys. 2.6. Symulacja modelu uczonego algorytmem BFGS bez rekurencji na danych uczących i weryfikujących

2.8. Model metodą najmniejszych kwadratów

Ewer = 294.6949 Eucz = 299.7408



Rys. 2.7. Symulacja modelu wykonanego za pomocą metody najmniejszych kwadratów