Politechnika Wrocławska

Sieci neuronowe i neurosterowniki

Sprawozdanie z projektu

Michał Leś, Jędrzej Kozal

prowadzący
Dr inż. Piotr Ciskowski

2017-10-01

1 Wstęp

Podstawowym założeniem projetku jest zaznajomienie się z neurosterownikami, ich zasadami działania oraz przygotowanie modelu predykcyjnego prostego obiektu dynamicznego. Dodatkowo przyjęto że bardziej interesujące będzie przyjęcie jakiegoś rzeczywistego obiektu niż badanie reakcji bardziej teoretycznego modelu o przyjętych charakterystykach.

Neurosterowniki są sposobem zaadresowania w automatyce kwestii sterowania obiektami mocno nieliniowymi, z którymi tradycyjne metody sterowania jak sterowniki PID nie dają sobie rady. U podstaw działania neurosterowników leży idea działania sieci neuronowej, które ostatnio zdominowały pole uczenia maszynowego. Są one wykorzystywane w wielu dziedzinach nauki i techniki do rozpoznawania obrazów (computer vision), klasyfikacji, sterownia ruchem ulicznym, wspomagania użytkowników różnych aplikacji (jak np. podpowiadanie słów w trakcie pisania na smartphonie), czy nawet prowadzenie eksperymentów społecznych. W niniejszej pracy postawiono zbadać w jaki sposób szerokie możliwości oferowane przez sieci neuronowe mogą być wykorzystane w automatyce.

2 Podstawy teoretyczne

2.1 Przyjęty model obektu

Obiekt w automatyce jest traktowany jako czarna skrzynka, która posiada wejścia i wyjścia. Na pobudzenie na wejściu reaguje odpowiedzią na wyjściu. Odpowiedź systemów dynamicznych zależy nie tylko od aktualnej wartości wejścia, ale także od stanu obiektu w poprzednich chwilach. Zgodnie z tym można zapisać model nieliniowego dyskretnego obiektu dynamicznego jako:

$$\begin{cases} \mathbf{x}(k+1) & = \phi(\mathbf{x}(k), \mathbf{u}(k)) \\ \mathbf{y}(k) & = \psi(\mathbf{x}(k)) \end{cases} \tag{1}$$

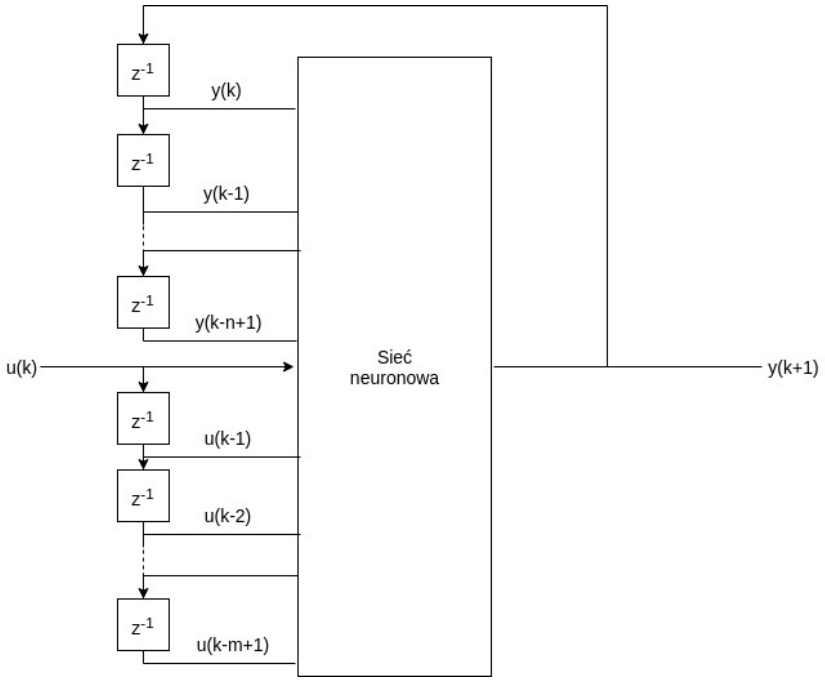
Układ 1 przedstawia równania stanu. Pierwsze równanie wiąże wewnętrzny stan obiektu z pobudzeniem, a drugie równanie wiąże stan obiektu z wyjściem. System opisywany tymi równaniami jest niezmienny w czasie (angl. time-invariant) - ϕ i ψ nie zależą bezpośrednio od czasu.

Jeśli zastąpimy funkcje nieliniowe przez odpowiednie macierze zyskamy system liniowy:

$$\begin{cases} \mathbf{x}(k+1) & = A\mathbf{x}(k) + B\mathbf{u}(k) \\ \mathbf{y}(k) & = C\mathbf{x}(k) + D\mathbf{u}(k) \end{cases} \tag{2}$$

W rzeczywistych systemach w równaniach oprócz stanu $\mathbf{x}$ i pobudzenia $\mathbf{u}$ pojawiają się także zakłócenia $\mathbf{z}$, które nie będą tutaj rozważane.

Klasa systemów liniowych jest od dawna dobrze znana. Znaleziono wiele sposobów badania i algorytmów sterowania obiektami liniowymi. O wiele trudniejsze w sterowaniu są obiekty nieliniowe. Nawet znając funkcję ψ oraz $x(k)$ samo stworzenie modelu mogącego dokonać identyfikacji takiego obiektu jest



Rysunek 1: Model identyfikacji obiektu z wykorzystaniem sieci neuronowej.

trudne, ze względu na kumulujące się z czasem błędy numeryczne, oraz poprzez konieczność poczynienia założeń co do stabilności systemu.

Najprostsza reprezentacja systemu dynamicznego nie zawiera wewnętrznego stanu, który w wielu przypadkach może być pominięty

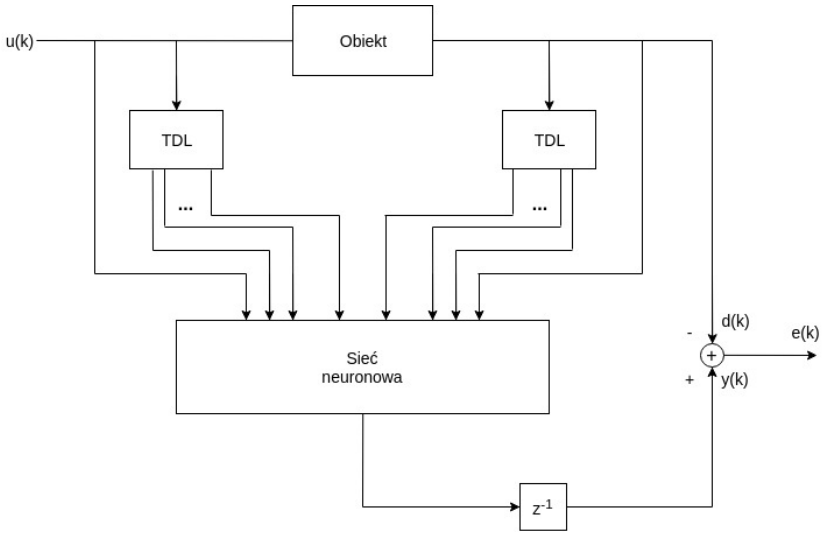
$$\mathbf{y}(k+1) = f(\mathbf{y}(k), \mathbf{u}(k)) \tag{3}$$

2.2 Zadanie identyfikacji obiektu

Zadanie identyfikacji sprowadza się do wyznaczeniu modelu reagującego na pobudzenie w sposób zbliżony do reakcji rzeczywistego obiektu:

$$||\hat{y} - y|| < \epsilon \tag{4}$$

Uzyskanie takiego modelu umożliwia sterowanie adaptacyjne, co znacznie ułatwia pracę z obiektami nieliniowymi. Celem sterowania w ogólności jest uzyskanie na wyjściu obiektu $\mathbf{y}(k)$ wartości jak najbardziej zbliżonej do wartości zadanej $\mathbf{y}_m(k)$. W przypadku sterowania adaptacyjnego oznacza to znalezienie takiego pobudzenia $\mathbf{u}(k)$ aby została spełniona zależność:



Rysunek 2: Schemat podłączenia sieci neuronowej w trakcie uczenia.

$$\lim_{k\to\infty} ||\mathbf{y}(k) - \mathbf{y}_m(k)|| < \epsilon \tag{5}$$

Często do takiego zadania są wykorzystywane sieci neuronowe przedstawione na rys. 1. Na wejście sieci podaje się wejścia systemu, oraz wyjścia z poprzednich chwil. Zadaniem sieci jest przewidzenie zachowania systemu w kroku $k+1$. Szereg bloków opóźniających będzie dalej oznaczany w pracy jako TDL(Tapped Dleay Line).

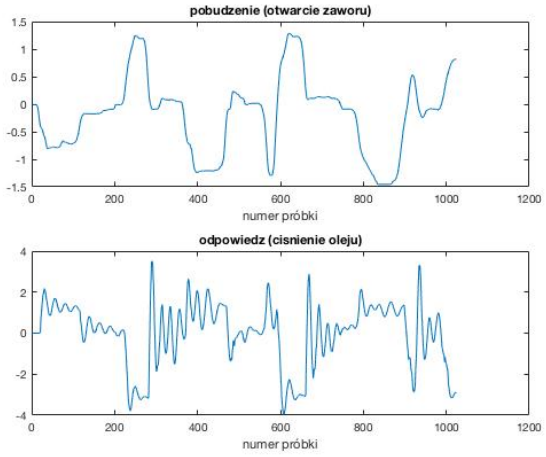
W trakcie identyfikacji sieć neuronowa jest zwykle włączana sposób szeregowo-równoległy, co zostało zilustrowane na rys 2. Taka konfiguracja umożliwia badanie błędu i minimalizację błędu zgodnie z nierównością 4. Sieć neuronowa nieustanie próbuje przewidzieć wyjście obiektu w następnej chwili. Różnica $e(k)$ jest wykorzystywana w procesie uczenia sieci. W [1] zaproponowano taką metodę oraz zaprezentowano wykorzystanie modelu do predykcji wyjścia filtru Butterwortha szóstego rzędu dla pobudzenia sygnałem trójkątnym oraz tłumionym sygnałem sinusoidalnym.

3 Zrealizowane zadania oraz otrzymane wyniki

Zadanie predykcji może zostać zrealizowane za pomocą modelowania liniowego ARX - modelu auteregresywnego z zewnętrznym wejściem. Jeżeli do modelowania zostały użyte sieci neuronowe model taki nazywa się NNARX (Neural Network ARX).

Zadanie zostało podzielone na dwie części - pierwsza część zakłada użycie domyślnych narzędzi do tworzenia sieci neuronowych w środowisku Matlab. Wymaga to nieco więcej nakładu w przygotowaniu zbiorów uczących i zarządzania sieciami. Druga metoda opiera się na zastosowaniu specjalnego dodatku do Matlaba służącego do tworzenia modelów liniowych ARX za pomocą sieci neuronowych (toolbox NNARX).

Na podstawie danych zawartych w pliku 'actuator.m' żądany model został załadowany do środowiska programu Matlab. Wejściem obiektu (u) jest otwarcie zaworu, zaś odpowiedzią systemu(y) jest ciśnienie oleju.



Rysunek 3: Dane wejściowe zależne od czasu

Pierwszym elementem potrzebnym do wykonania zadania była konstrukcja prawidłowego ciągu uczącego. W modelu ARX potrzebujemy stworzyć układ z opóźnionymi wejściami i odpowiedziami. W zależności od potrzeb modelu, możemy dostosować wielkość opóźnienia i ilość opóźnionych próbek. W naszym przypadku skonstruowana została następującą macierz wejściowa dla sieci:

$$u_s = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & u' & 0 & 0 \\ 0 & 0 & u' & 0 \\ 0 & p' & 0 & 0 \\ 0 & 0 & p' & 0 \\ 0 & 0 & 0 & p' \end{bmatrix} \tag{6}$$

gdzie u' i y' odpowiadają transponowanym wektorom otrzymanym z modelu. Analogicznie skonstruowany został wektor pożądanych wyjść dla sieci:

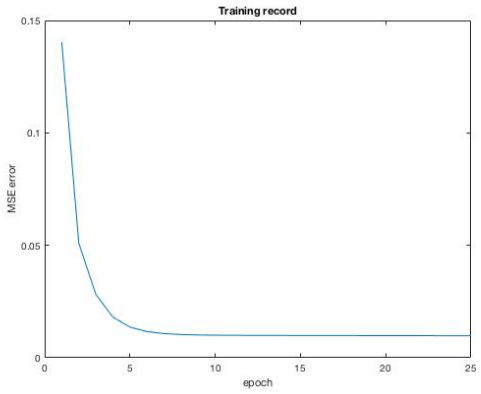
$$y_s = \begin{bmatrix} p' & 0 & 0 & 0 \end{bmatrix}' \tag{7}$$

Dane zostały pogrupowane na zbiór uczący (pierwsze 512 próbek), oraz zbiór sprawdzający(kolejne 512 próbek).

Tak przygotowane dane zostały poddane procesowi uczenia przyrostowego. Przykładowe wartości dla minimalnego błędu kwadratowego (MSE) wahały się w okolicach 0.01 co widać na rysunku 5.

```
Krok     1 - błąd: 0.140394
Krok     2 - błąd: 0.050943
Krok     3 - błąd: 0.028158
Krok     4 - błąd: 0.018099
Krok     5 - błąd: 0.013627
Krok     6 - błąd: 0.011630
Krok     7 - błąd: 0.010732
Krok     8 - błąd: 0.010323
Krok     9 - błąd: 0.010133
Krok    10 - błąd: 0.010040
Sprawdzam sieć ARX na zbiorze sprawdzającym...
Symuluję siećARX na zbiorze sprawdzającym...
```

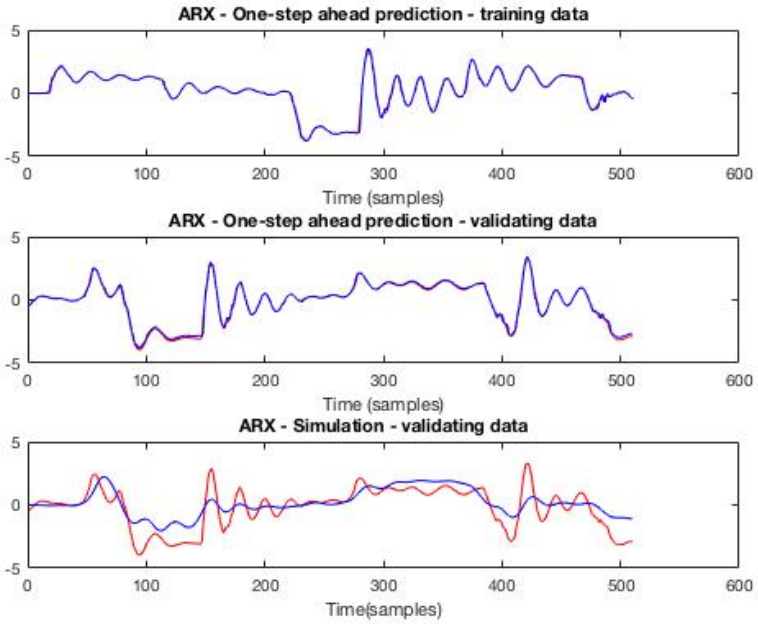
Rysunek 4: Przykładowe wartości błędów



Rysunek 5: Zależność błędu od ilości epok

Wraz ze wzrostem ilości epok błąd malał bardzo wolno - na rysunku nr 5 jest dobrze widoczne, że już po 10 epokach otrzymane zostały wartości błędu rzędu błędu po 25 epokach.

Wykresy przedstawione na rysunku 6 przedstawiają testy otrzymanej sieci neuronowej. W pierwszych dwóch przypadkach dokonano sprawdzenia działania sieci na zbiorze uczącym oraz walidacyjnym za pomocą metody one-step-ahead.

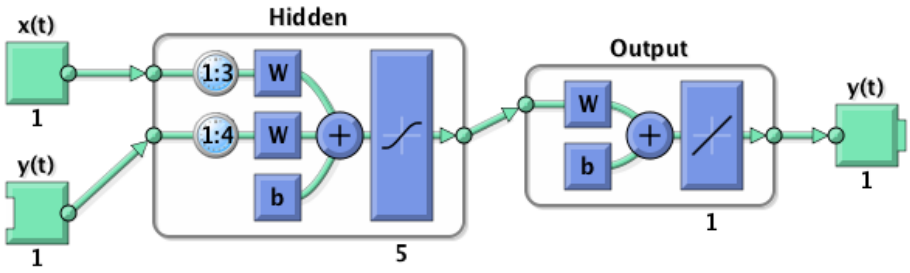


Rysunek 6: Wykresy predykcji one-step-ahead dla zbioru uczącego i walidacyjnego oraz symulacja obiektu

Metoda ta próbuje przewidzieć wyjście obiektu w następnym dyskretnym kroku na podstawie pełnej informacji o wyjściu obiektu i jego wejściu w przeszłości. Ostatni wykres przedstawia symulację obiektu za pomocą modelu stworzonego na podstawie sieci. Model ten powinien modelować badany wcześniej obiekt. Na wykresie widzimy, że zachowanie obiektu modelowanego w ten sposób jest podobne do rzeczywistego, jednak jest on mniej podatny na wykrywanie zmian.

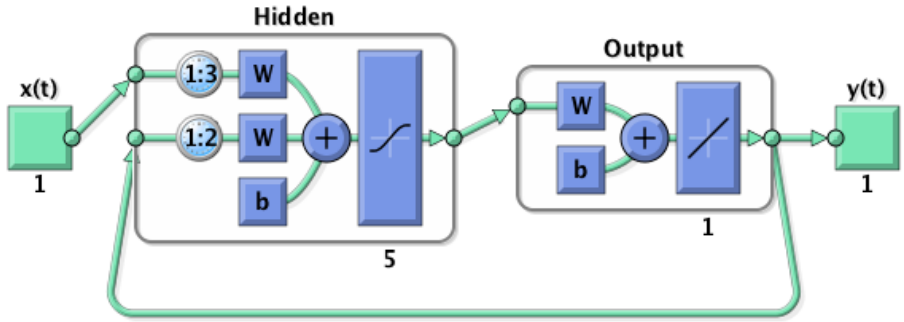
Druga część zadania polegała na stworzeniu modelu za pomocą toolboxa zawierającego gotowe rozwiązania dla sieci typu NNARX. Stworzono nowy obiekt zawierający te same dane, oraz załadowano je do niego. Bardzo wygodną opcją modelowania za pomocą obiektu nnarxnet jest możliwość łatwej manipulacji wielkością opóźnień wejść i wyjść dla uczenia sieci. W poprzednim modelu opóźnienia były generowane już na poziomie przygotowania danych do uczenia, co wymagało nakładu w postaci modelowania wejścia i wyjścia (rysunek 3 na stronie 4). Sieci te są również szybsze w działaniu. Przy użyciu mechanizmów optymalizacyjnych Matlaba przetwarzanie tych samych struktur w połączeniu z metodą uczenia Levenberga Marquardta pozwala na uzyskanie znacznie lepszych efektów.

W trybie uczenia sieci (rysunek 7), przyjmuje ona wejścia pochodzące od prawdziwego obiektu.



Rysunek 7: Schemat sieci NNARX dokonującej predykcji pochodzącej z rzeczywistego obiektu

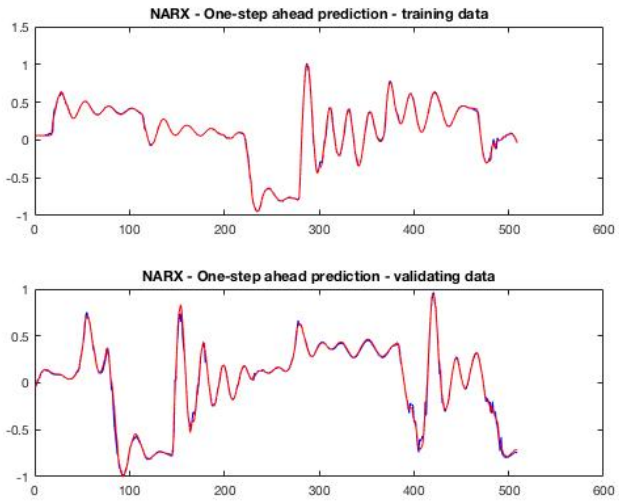
Jeżeli badany model ma służyć do identyfikacji obiektu (rysunek 8), wprowadzenie sprzężenia zwrotnego spowoduje zachowanie się wyjścia dokładnie tak jak obiekt



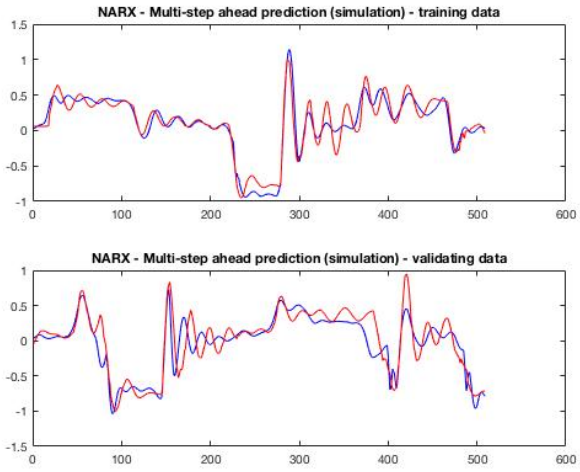
Rysunek 8: Schemat sieci NNARX symulującej obiekt

Sieć otrzymana w wyniku działania na obiekcie nnarx jest siecią wielowarstwową, można dzięki temu za jej pomocą łatwo manipulować ilością neuronów w warstwie ukrytej (szybki czas przetwarzania pozwala na eksperymentowanie z długością uczenia i ilością neuronów w celu osiągnięcia żądanej dokładności modelu). Toolbox Matlaba udostępnia bardzo wiele statystyk na temat procesu uczenia i analizy sieci. Głównym interesującym nas wykresem jest skuteczność metody predykcji typu one-step-ahead oraz symulacja obiektu czyli poprawność wykonania zadania identyfikacji.

Dla 5 neuronów w warstwie ukrytej, 1000 epokach uczenia i opóźnieniem wejść o 2 i wyjść o 3 na wejściu sieci otrzymałem wykresy przedstawione na rysunku 9. Wygenerowanie symulacji obiektu wynikające z zamknięcia pętli sprzężenia zwrotnego zostało przedstawione na rysunku 10.

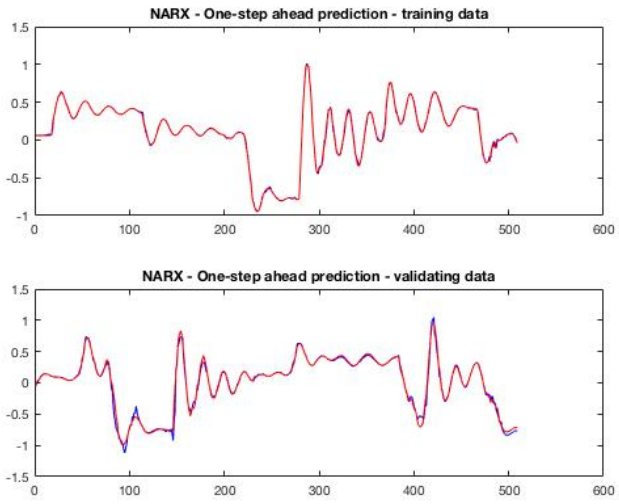


Rysunek 9: Wykresy predykcji one-step-ahead dla zbioru uczącego i walidacyjnego

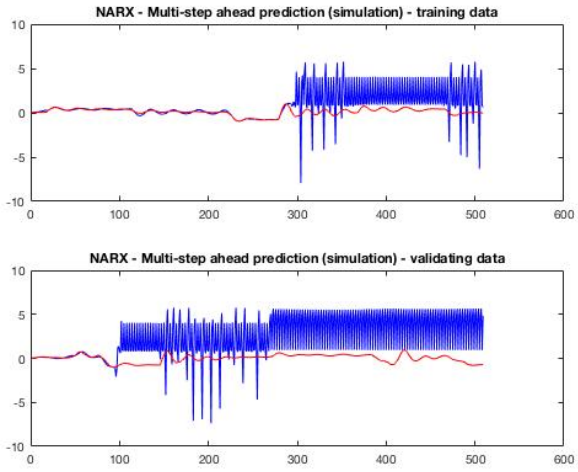


Rysunek 10: Symulacja rzeczywistego obiektu

Dla 10 neuronów w warstwie ukrytej, 1000 epokach uczenia i opóźnieniem wejść o 2 i wyjść o 3 na wejściu sieci otrzymałem wykresy przedstawione na rysunku 11. Wygenerowanie symulacji obiektu wynikające z zamknięcia pętli sprzężenia zwrotnego zostało przedstawione na rysunku 12.

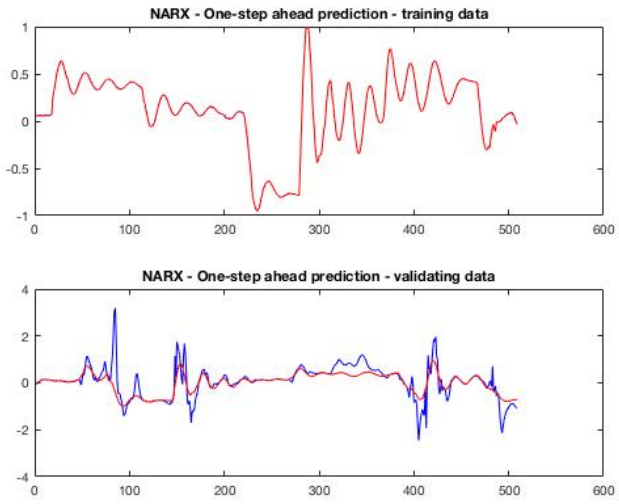


Rysunek 11: Wykresy predykcji one-step-ahead dla zbioru uczącego i walidacyjnego

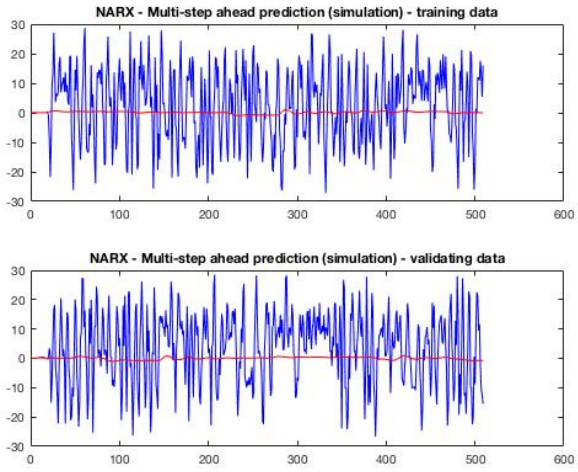


Rysunek 12: Symulacja rzeczywistego obiektu

Dla 100 neuronów w warstiwe ukrytej, 1000 epokach uczenia i opóźnieniem wejść o 2 i wyjść o 3 na wejściu sieci otrzymałem wykresy przedstawione na rysunku 13 Wygenerowanie symulacji obiektu wynikające z zamknięcia pętli sprzężenia zwrotnego zostało przedstawione na rysunku 14.



Rysunek 13: Wykresy predykcji one-step-ahead dla zbioru uczącego i walidacyjnego



Rysunek 14: Symulacja rzeczywistego obiektu

Porównując otrzymane w ten sposób wyniki można stwierdzić, że zwiększenie ilości neuronów w warstwie ukrytej nie zwiększa wydajności sieci. Po wynikach symulacji obiektu przy użyciu modelu można stwierdzić, że zwiększenie ilości neuronów w warstwie ukrytej powoduje większą czułość układu na zakłócenia. Układ rozpoczyna oscylacje z dużą częstotliwością oraz wydaje się być łatwy do wyprowadzenia z równowagi. Również właściwości predykcji one-step-ahead są gorsze dla modelu ze zbyt dużą ilością neuronów w warstwie ukrytej.

W celu zbadania wpływu opóźnień na właściwości układu zmienione zostały parametry opóźniające (konstruktor obiektu nnarx). Zmiana dotyczyła ilości opóźnień. Żadna konfiguracja opóźnień nie dała lepszych wyników niż konfiguracja opóźnień wyjść o 3 i wejść o 2.

4 Wnioski

Nieliniowe procesy są zwykle za bardzo skomplikowane dla modelowania tradycyjnymi metodami statystycznymi. Modelowanie za pomocą nowoczesnych metod opierających się o uczenie maszynowe czy sieci neuronowe pozwala na przybliżenie konkretnych właściwości badanego układu. Nie pozwala jednak na pełną predykcję zachowań układu. Należy zwrócić uwagę, że do właściwej identyfikacji systemu niezbędna jest duża liczba danych i głębsza ich analiza. Sieć będzie się zachowywać tylko tak, jak ją tego nauczymy i tylko na podstawie informacji, które zawarte były w danych uczących. W sprawozdaniu zostały opisane najprostsze scenariusze dotyczące działania i modelowania rzeczywistych obiektów. Modele wygenerowane za pomocą obiektu dedykowanego sieciom typu NNARX działały znacznie szybciej i były prostsze w użyciu. Samo narzędzie dysponuje ogromną ilością informacji na temat przebiegu uczenia, wartości wag czy właściwości wygenerowanej sieci. Sieci generowane ręcznie również służą jako dobry model dla sieci NNARX, nie dają jednak tak dokładnej informacji o obiekcie.

Literatura

[1] Stanisław Osowski. *Sieci Neuronowe w ujęciu algorytmicznym* Wydawnictwo Naukowo-Techniczne, Warszawa 1996

[2] Strona prowadzącego,
`http://staff.iiar.pwr.wroc.pl/piotr.ciskowski/skrypt/SNwMATLABie.htm`

[3] J. Rhim, S. W. Lee *A neural network approach for damage detection and identification of structures* Computational Mechanics, Springer 1995

[4] Jerzy Lipski *Identyfikacja procesów dynamicznych z zastosowaniem sieci neuronowych* Nauka i technika 2001