**Politechnika Krakowska**
**im. Tadeusza Kościuszki**
Wydział Mechaniczny

**mgr inż. Łukasz Łacny**

# Neuronowe algorytmy aktywnego tłumienia drgań wybranych układów dyskretnych i ciągłych

Praca doktorska wykonana pod kierunkiem
dr hab. inż. Piotra Cupiała, prof. AGH

**Kraków 2016**

BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

# Spis treści


**1. Wprowadzenie** **2**

1.1. Sztuczne sieci neuronowe . . . 2

1.2. Cel i teza pracy . . . 4

**2. Identyfikacja neuronowa** **5**

2.1. Architektura i modele DSSN . . . 6

2.2. Wybrane modele emulatorów neuronowych . . . 9

2.2.1. Model NNFIR . . . 9

2.2.2. Model NNARX . . . 10

2.2.3. Model NNARMAX . . . 10

2.3. Algorytmy uczenia sieci neuronowych . . . 16

2.4. Emulacja neuronowa na przykładzie wybranych modeli dynamicznych . . . 17

2.4.1. Wahadło matematyczne . . . 17

2.4.2. Giroskop . . . 21

2.4.3. Belka typu „smart” . . . 24

**3. Sterowanie neuronowe** **33**

3.1. Architektura sieci wybranych algorytmów sterowania neuronowego . . . 35

3.1.1. Sterowanie typu „inverse” (odwrotne) . . . 35

3.1.2. Sterowanie typu NARMA-L2 . . . 37

3.1.3. Sterowanie typu FF . . . 39

3.1.4. Sterowanie predykcyjne . . . 40

3.1.5. Emulacja istniejącego układu sterowania . . . 41

3.2. Przykłady sterowania neuronowego . . . 41

3.2.1. Wahadło matematyczne . . . 42

3.2.2. Belka typu „smart” . . . 59

3.2.3. Wahadło podwójne nieliniowe . . . 68

3.2.4. Giroskop . . . 75

3.3. Zagadnienia uzupełniające . . . 83

3.3.1. Realizacja praktyczna . . . 83

3.3.2. Analiza stabilności . . . 84

**4. Podsumowanie i wnioski** **86**

**A. Dodatek** **89**




BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

# 1. Wprowadzenie

## 1.1. Sztuczne sieci neuronowe

Chociaż pojęcie „biomimetyka" (znane również jako „biomimikra" bądź „bionaśladownictwo") [1], [2], [3] oraz powiązane z nim ściśle pojęcie „bionika" są pojęciami stosunkowo młodymi, jako że zdefiniowane zostały dopiero w połowie wieku XX-go, sama koncepcja polegająca na rozwoju technologii bazując na „zapożyczonych" rozwiązaniach zaczerpniętych z natury pojawiła się znacznie wcześniej.

Początki zagadnienia sztucznych sieci neuronowych sięgają lat 40-tych XX-go wieku, kiedy to przeprowadzona przez McCullocha i Pittsa próba opisu matematycznego aktywności mózgowej będącej wynikiem przewodzenia informacji neuronowej pomiędzy poszczególnymi neuronami, zaowocowała stworzeniem pierwszego formalnego modelu sztucznego neuronu [4]. Kilka lat później, dokładnie w 1949 roku, psycholog Donald Hebb, zaproponował w swojej książce „The Organization of Behaviour" [5] metodę uczenia sieci neuronowych poprzez modyfikację wartości wag na powiązaniach pomiędzy poszczególnymi neuronami. Metoda ta, nazywana obecnie „metodą Hebba", stanowiła pierwszy krok na drodze do właściwego uczenia sztucznych sieci neuronowych.

Lata pięćdziesiąte przyniosły kolejne osiągnięcia w zakresie sztucznych sieci neuronowych. Szczególne znaczenie miało zbudowanie w 1958 roku przez Franka Rosenblatta, pierwszego modelu perceptronu - prostej sieci umożliwiającej klasyfikację obrazów bazując na wzorcach zdefiniowanych w procesie jej uczenia [6]. Co ciekawe, perceptron był w stanie zachować zdolność poprawnego działania nawet w przypadku gdy część z jego połączeń bądź elementów uległa uszkodzeniu.

Układ o nazwie ADALINE, skonstruowany w 1960 roku przez Bernarda Widrowa z Uniwersytetu Stanford, a następnie jego rozbudowana wersja określana mianem MADALINE (many ADALINEs), stanowiły dalsze praktyczne rozszerzenie koncepcji sztucznych sieci neuronowych. Zastosowania sieci MADALINE obejmowały między innymi rozpoznawanie obrazów oraz adaptacyjne przetwarzanie sygnałów.

Niestety z końcem lat 60-tych entuzjazm związany z technologią sztucznych sieci neuronowych oraz metodami ich uczenia znacznie się zmniejszył, czego zasadniczym powodem było opublikowanie w 1969 roku przez Minsky'ego oraz Paperta książki „Perceptrons : an Introduction to Computational Geometry" [7]. W książce tej autorzy poddali w wątpliwość możliwość uczenia sieci wielowarstwowych. Dopiero w połowie lat osiemdziesiątych wraz z rozwojem algorytmów uczenia sieci oraz wzrostem szybkości przetwarzania oraz możliwości obliczeniowych



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

komputerów, udało się obalić postawioną przez nich tezę.

Wznowione z początkiem lat 90-tych zainteresowanie sztucznymi sieciami neuronowymi przyniosło ze sobą zarówno rozwój metod i technik uczenia jak również zaowocowało rozszerzeniem typów architektury sieci poczynając od sieci prostych wielowarstwowych, poprzez sieci radialne, aż po sieci rekurencyjne. W przeciągu ostatnich 25 lat zakres działania oraz zastosowania sieci rozszerzył się znacznie obejmując obecnie między innymi zagadnienia rozpoznawania wzorców i sekwencji, przetwarzania i filtracji danych, aproksymacji funkcji nieliniowych czy też wreszcie identyfikacji oraz sterowania procesami dynamicznymi [8], [9], [10], [11], [12].

Jedną z kluczowych prac ściśle poświęconych zagadnieniu identyfikacji oraz sterowania układami dynamicznymi z wykorzystaniem sztucznych sieci neuronowych jest praca opublikowana przez Narendrę oraz Parthasarathy'ego [13] w pierszym wydaniu „IEEE Transactions on Neural Networks" w 1990 roku. Praca oraz zawarte w niej rozwiązania stanowi podstawę rozwoju współcześnie stosowanych neuronowych algorytmów i metod sterowania, bazujących albo na samodzielnie działającej sieci neuronowej albo na sieci w powiązaniu ze sterowaniem klasycznym [14], [15], [16]. Na początku XXI-go wieku pojawiły się dodatkowo propozycje sterowania neuronowego w powiązaniu z innymi metodami inteligentymi, przykładowo sterowanie neuronowo-rozmyte w oparciu o logikę rozmytą [17], [18], czy też pomysł polegający na uczeniu sieci neuronowej z wykorzystaniem algorytmów genetycznych oraz ewolucyjnych [19], [20].

Podobnie jak zagadnienie sterowania z wykorzystaniem sztucznych sieci neuronowych, koncepcja tak zwanych „konstrukcji inteligentych" spotkała się w przeciągu ostatnim kilkunastu lat ze sporym zainteresowaniem środowiska naukowego [21], [22], [23], [24], [25], [26], [27], [28], [29]. Dla koncepcji tej przyjmuje się założenie, że wykorzystując nowoczesną technologię oraz materiały w powiązaniu z „inteligentnymi" metodami sterowania, możliwe jest uzyskanie konstrukcji będących w stanie wykrywać i jednocześnie aktywnie reagować na zmiany zachodzące w ich otoczeniu. Jako że zagadnienie „konstrukcji inteligentnych" jest zagadnieniem stosunkowo obszernym, w literaturze doszukać się można sporej liczby publikacji dotyczących tego tematu, obejmujących m.in. monitorowanie stanu układów bądź konstrukcji [30], [31], aktywne tłumienie drgań i hałasu [32], [33], [34] czy też „energy-harvesting" czyli odzyskiwanie energii z ruchu i drgań [35].



BPK BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

## 1.2. Cel i teza pracy

Za główny cel pracy doktorskiej postawiono przetestowanie szeregu algorytmów sterowania neuronowego pod kątem zastosowania ich do aktywnego tłumienia drgań układów oscylacyjnych, w szczególności układów wykazujących zachowania nieliniowe.

W pracy przyjęto dodatkowo założenie, że tłumienie drgań za pomocą sztucznych sieci neuronowych ma się odbywać w sposób aktywny bądź hybrydowy (w powiązaniu z układami pasywnymi), lecz nie w sposób semi-aktywny (poprzez modyfikację parametrów sterowanego układu).

Teza jaką postawiono dla niniejszej pracy doktorskiej brzmi: „Sztuczne sieci neuronowe mogą, przy przyjęciu odpowiedniej architektury i przeprowadzeniu właściwego procesu uczenia, znaleźć zastosowanie w aktywnych bądź hybrydowych układach redukcji drgań układów liniowych i nieliniowych."



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

# 2. Identyfikacja neuronowa

Zarówno zakres zastosowań jak i możliwości oferowanych przez sztuczne sieci neuronowe jest obszerny, co zostało już wspomniane we wprowadzeniu do niniejszej pracy. Należy przy tym zaznaczyć, iż różnorodność zastosowań sztucznych sieci neuronowych jest bez wątpienia bezpośrednio powiązana z różnorodnością typów i struktur tychże sieci. Poczynając od prostych sieci typu „perceptron" bądź „Adaline", poprzez sieci dynamiczne i radialne, aż po złożone wielowarstwowe sieci rekurencyjne - każda z nich zaprojektowana jest oraz dostosowana do konkretnego typu celów i zadań.

Domyślać się można zatem, iż do celów sterowania, bądź bardziej ogólnie, modelowania układów dynamicznych skorzystać należy ze sztucznych sieci neuronowych o odpowiednio zaprojektowanej strukturze oraz właściwościach. Istotnie, tzw. sieci dynamiczne (zarówno typu „feedforward" jak i rekurencyjne) znajdują swoje zastosowania w przypadku gdy przedmiotem zainteresowania jest obiekt bądź zjawisko o charakterze dynamicznym. Identyfikacja neuronowa (zwana niekiedy emulacją neuronową), będąca tematem niniejszego rozdziału, jest niczym innym jak techniką nadzorowanego uczenia sieci neuronowej, na bazie danych uzyskanych z rzeczywistego obiektu bądź jego modelu matematycznego. Tak nauczona sieć, nosząca miano emulatora, jest w stanie z odpowiednią dokładnością oraz w odpowiednim zakresie naśladować zachowanie wybranego obiektu dynamicznego. Przykładowo, podając na wejście sieci wartości stanu układu oraz wartości wymuszenia dla chwili obecnej oraz chwil ją poprzedzających, sieć taka dokona predykcji wartości stanu układu w kolejnych, przyszłych chwilach, bazując na wzorcach, które posłużyły do jej nauczenia.

W rozdziale niniejszym przedstawiony zostanie dokładnie proces uczenia tego typu sieci, włącznie z zagadnieniami, których znajomość niezbędna jest do prawidłowego nauczenia sieci, a co za tym idzie do skutecznego jej funkcjonowania. Wyjaśnione zostaną kolejno:

- struktura oraz właściwości podstawowych typów dynamicznych sztucznych sieci neuronowych
- proces pobierania, bądź generacji próbek niezbędnych do uczenia sieci, jak również ograniczenia jakie nieodpowiedni dobór próbek nakłada na nauczoną sieć
- algorytmy uczenia sieci neuronowych oraz ich skuteczność dla przypadku sieci dynamicznych
- emulacja ciągłych i dyskretnych układów dynamicznych

Wszystkie zagadnienia przedstawione w tym rozdziale znajdują swoje rozszerzenie w rozdziale kolejnym dotyczącym metod sterowania neuronowego.

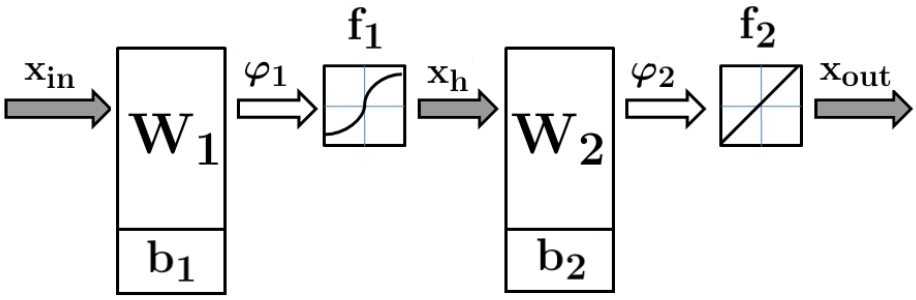


BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

Warto przy tej okazji zaznaczyć, że realizacja numeryczna wszystkich opisywanych w pracy układów oraz funkcji odbywała się w środowisku MATLAB/Simulink z wykorzystaniem narzędzi z pakietu „Neural Network Toolbox". Bazując na istniejących w pakiecie narzędziach, możliwym była implementacja przez autora w środowisku MATLAB szeregu opisywanych dalej w pracy metod emulacji oraz sterowania neuronowego. Ponadto implementacja ta obejmowała również przygotowanie modeli dynamicznych rozpatrywanych układów w oparciu o opisujące je równania analityczne (zarówno liniowe jak i nieliniowe), realizację funkcji służących do generacji wzorców uczących na bazie wspomnianych modeli, jak również zaprogramowanie szeregu funkcji i bloków pomocniczych (m.in. bloku powolnego zejścia do celu czy bloku opóźnień).

## 2.1. Architektura i modele DSSN

Pod względem architektury i struktury, dynamiczne sztuczne sieci neuronowe (DSSN) nie różnią się w zasadniczy sposób od ich wielowarstwowych odpowiedników typu „feedforward". Podobnie jak i w tamtym przypadku, sieć zbudowana jest z kolejno następujących po sobie bloków, tzw. warstw, przez które przepuszczana jest krok po kroku informacja wejściowa. Na rys. 2.1 przedstawiona jest przykładowa struktura sieci tego typu.

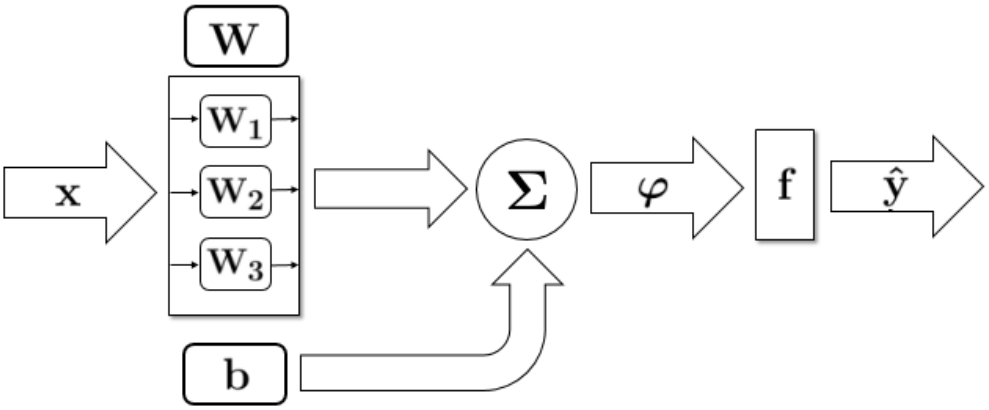



Rysunek 2.1.: Przykładowa struktura SSN typu „feedforward"

Powyżej przedstawiona sieć jest tzw. siecią dwuwarstwową tzn. siecią o jednej warstwie ukrytej i jednej wyjściowej. Warstwa ukryta stanowi element pośredni pomiędzy warstwą wejściową, na którą podawana jest informacja, a warstwą wyjściową, z której pobierana jest odpowiedź sieci. W zależności od potrzeb możliwe jest rozszerzenie sieci o dodatkowe warstwy ukryte, przy czym wiąże się to ze wzrostem złożoności układu, a co za tym idzie ze znacznym zwiększeniem kosztów obliczeniowych. Dla większości przypadków sieci 2- i 3-warstwowe (czyli sieci o odpowiednio jednej bądź dwóch warstwach ukrytych) są jednak wystarczające.



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ




Rysunek 2.2.: Uproszczony schemat pojedynczej warstwy sieci neuronowej

Ogólny algorytm działania sieci przedstawionej na rys.2.1 opisać można posiłkując się bardziej szczegółowym modelem pojedyńczej warstwy (rys. 2.2) w następujący sposób. Na wejście sieci podawana jest informacja wejściowa w postaci n-wymiarowego wektora $\mathbf{x}$. Wektor ten przemnażany jest przez tzw. macierz wag $\mathbf{W}$ o wymiarach $m \times n$, a do wyniku dodawany jest tzw. wektor odchyleń $\mathbf{b}$. Tak otrzymany wektor, poddawany jest następnie transformacji poprzez funkcję aktywacji $\mathbf{f}$, dając w rezultacie pożądaną odpowiedź w postaci m-wymiarowego wektora wyjściowego $\hat{\mathbf{y}}$. Algorytm ten przedstawić można w skrócie za pomocą następującego wzoru:

$$\hat{\mathbf{y}} = \mathbf{f}(\boldsymbol{\varphi}) = \mathbf{f}(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{2.1}$$

Propagacja wektora wejściowego przez kolejne warstwy sieci z wykorzystaniem powyższego wzoru, skutkuje otrzymaniem pożądanej wartości wyjściowej, przy założeniu, że sieć neuronowa jest poprawnie nauczona. Zależność pomiędzy wektorem wyjścia i wejścia dla sieci przedstawionej na rys. 2.1 zapisać można w następujący sposób:

$$\hat{\mathbf{y}} = \hat{\mathbf{y}}_\mathbf{2} = \mathbf{f_2}(\mathbf{W_2}\hat{\mathbf{y}}_\mathbf{1} + \mathbf{b_2}) = \mathbf{f_2}(\mathbf{W_2}\mathbf{f_1}(\mathbf{W_1}\mathbf{x} + \mathbf{b_1}) + \mathbf{b_2}) \tag{2.2}$$

Zarówno ilość warstw danej sieci neuronowej, jak i rozmiar macierzy wag oraz wektorów odchyleń w każdej z warstw dobierane są odgórnie w zależności od potrzeb oraz zastosowania danej sieci. Najczęściej dobór ten uzależniony jest od stopnia złożoności obiektu bądź zjawiska, które ma być modelowane przez sieć. Podobnie sprawa ma się z funkcjami aktywacji. Do najczęściej stosowanych funkcji aktywacji w przypadku sieci dynamicznych zalicza się:

- funkcję liniową

$$f(\varphi) = \varphi \tag{2.3}$$

- funkcję sigmoidalną

$$f(\varphi) = \frac{1}{1 + e^{\varphi}} \tag{2.4}$$

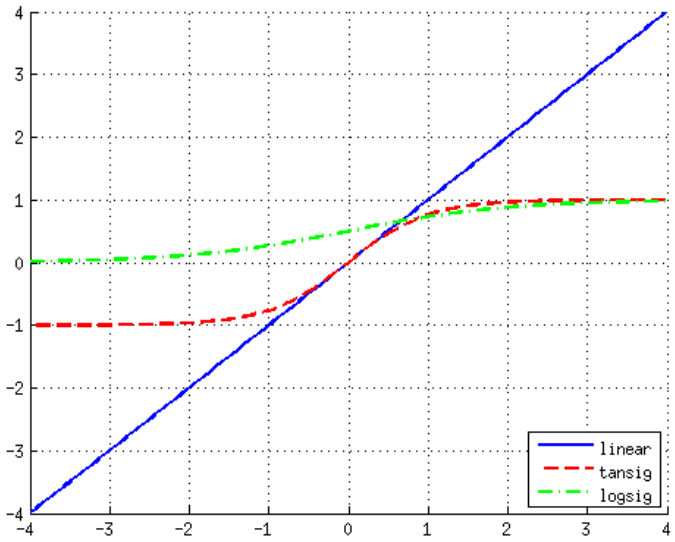


BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

- funkcję tangensoidalną

$$f(\varphi) = \mathrm{tgh}(\varphi) = \frac{e^{2\varphi} - 1}{e^{2\varphi} + 1} \tag{2.5}$$

Na rys. 2.3 przedstawione są wykresy wymienionych powyżej funkcji aktywacji. Z uwagi na fakt, że wartość funkcji liniowej 2.3 pokrywa się z wartością podanego na nią argumentu, funkcja ta jest najczęściej stosowana w warstwie wyjściowej sieci, przy uzyskiwaniu ostatecznej odpowiedzi sieci. Pozostałe dwie funkcje aktywacji (sigmoidalną i tangensoidalną) stosuje się najczęściej do transformacji wartości w warstwach ukrytych. Wart zauważenia jest fakt, iż nieliniowości wynikające z kształtu tych funkcji przenoszą się na zachowania nieliniowe sieci, rozszerzając tym samym efektywny zakres jej działania.




Rysunek 2.3.: Wykresy wybranych funkcji aktywacji

Do tego miejsca opis architektury odpowiada zarówno sieciom wielowymiarowym typu „feedforward" jak i sieciom dynamicznym. Różnica pomiędzy nimi przejawia się głównie w odpowiednim doborze wektora wejściowego i wyjściowego. W przypadku klasycznych sieci statycznych, na wejście sieci podawane są wartości wybranych kluczowych parametrów modelu, natomiast na jej wyjściu otrzymuje się wartość dodatkowego bądź dodatkowych nieznanych parametrów, zależnych w sposób pośredni bądź bezpośredni od podanych wartości wejściowych.

W przypadku sieci neuronowych dynamicznych, sieć ma za zadanie dokonać predykcji stanu układu $\hat{\mathbf{y}}$ dla kolejnych kroków czasowych, bazując na wprowadzonych do niej regresorów tzn. wartości stanu układu oraz wymuszeń, którym układ jest poddawany w chwilach poprzednich. Zachowanie sieci można zatem w ogólny sposób przedstawić za pomocą funkcji:

$$\hat{\mathbf{y}}(n) = \mathbf{f}(\mathbf{y}(n-1), \ldots, \mathbf{y}(n-k), \mathbf{u}(n), \ldots, \mathbf{u}(n-m)) \tag{2.6}$$



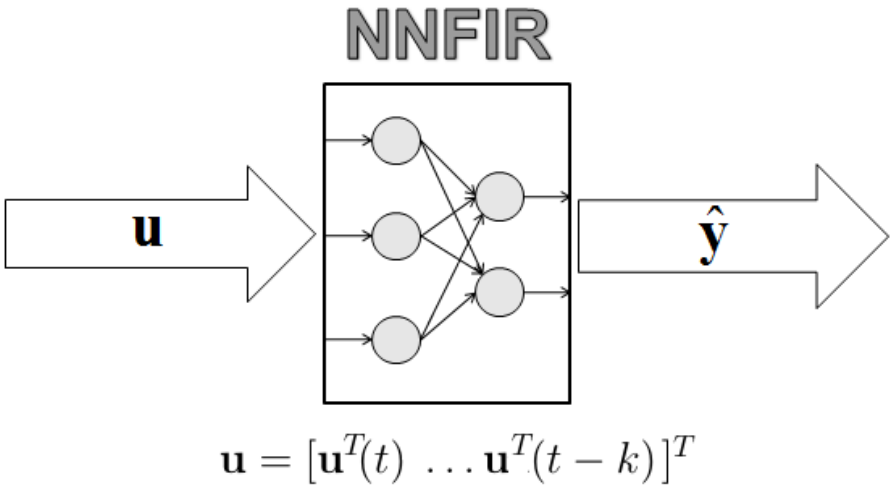
BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

## 2.2. Wybrane modele emulatorów neuronowych

Równanie 2.6 przedstawione w poprzednim paragrafie, opisujące w sposób ogólny specyfikę zachowania dynamicznych sztucznych sieci neuronowych, stanowi podstawę definicji dla modeli emulatorów neuronowych. W niniejszej pracy do celów emulacji neuronowej wykorzystywane są trzy podstawowe modele, bazujące na znanych modelach parametrycznych stosowanych w procesach identyfikacji klasycznej oraz predykcji statystycznej. Wybrane modele to kolejno NNFIR, NNARX oraz NNARMAX. Podstawowa ich zaleta w stosunku do odpowiadających im klasycznych liniowych modeli FIR, ARX oraz ARMAX, przejawia się w możliwości skutecznej emulacji wybranych modeli dynamicznych, zarówno w zakresie liniowym jak i poza nim. Zaleta ta, co zostało już uprzednio wspomniane, wynika w głównej mierze z zastosowania w nich nieliniowych funkcji aktywacji. Dodatkowo, zgodnie z założeniami postawionymi w pracy, każdy z tych modeli uczony jest w trybie „off-line", choć możliwe jest również rozszerzenie ich do tryby „on-line", przez co rozumie się ciągłe ich douczanie już w trakcie pracy emulatora. Dokładny opis każdego z rozpatrywanych modeli przedstawiony jest poniżej.

### 2.2.1. Model NNFIR

Pierwszy z prezentowanych modeli emulatorów neuronowych, jest jednocześnie najprostszym pod względem zastosowanej struktury. Skrót NNFIR oznacza Nonlinear Neural Finite Impulse Response. Wartość wyjściowa układu $\hat{\mathbf{y}}$ w tym przypadku,wyznaczana jest tylko i wyłącznie na podstawie kolumnowego wektora wymuszeń $\mathbf{u}$, którym poddawany jest układ w chwili obecnej oraz w $k$ poprzednich chwilach czasowych, jak to zostało przedstawione na rys. 2.4.




Rysunek 2.4.: Struktura sieci typu NNFIR

Równanie emulatora można zatem w ogólny sposób przedstawić w postaci:

$$\hat{\mathbf{y}}(t) = \mathbf{f}(\mathbf{u}) = \mathbf{f}(\mathbf{u}(t), \mathbf{u}(t-1) \ldots, \mathbf{u}(t-k)) \tag{2.7}$$



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

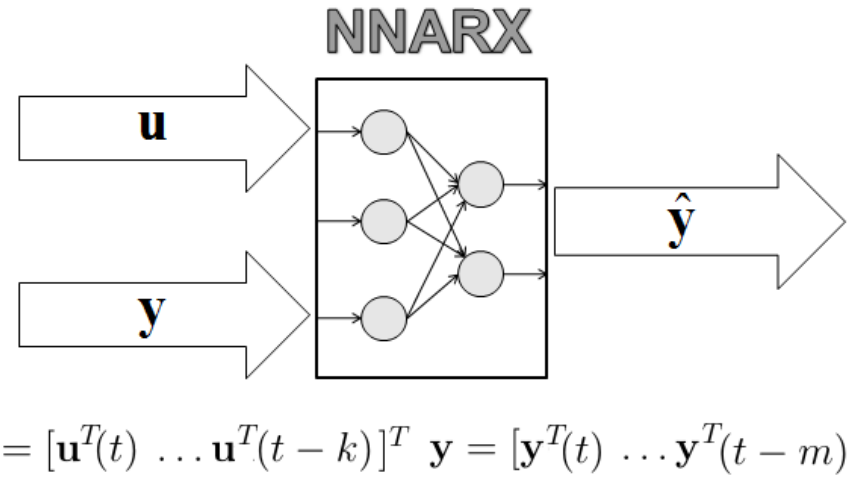
Niestety prostota struktury modelu NNFIR, przekłada się równocześnie na ograniczony zakres jego zastosowań. W tym przypadku pominięcie przy wyznaczaniu kolejnego stanu układu, poprzednich jego stanów, wiąże się z ograniczeniem możliwości predykcji do układów, w których nie występuje zależność pomiędzy kolejnymi i poprzednimi stanami układu. W modelach liniowych przekłada się to na układy nie posiadające biegunów transmitancji, a co za tym idzie takich, gdzie stan układu uzależniony jest wyłącznie od poprzednich wymuszeń.

### 2.2.2. Model NNARX

Emulator typu NNARX (Nonlinear Neural AutoRegressive with eXogenous input) stanowi logiczne rozwinięcie modelu NNFIR, w którym uwzględnione są poprzednie stany układu. Równanie dla modelu NNARX jest identyczne jak przedstawione już ogólne równanie 2.8 dla sieci dynamicznych:

$$\hat{\mathbf{y}}(t) = \mathbf{f}(\mathbf{u}, \mathbf{y}) = \mathbf{f}(\mathbf{u}(t), \mathbf{u}(t-1), \ldots, \mathbf{u}(t-k), \mathbf{y}(t-1), \ldots, \mathbf{y}(t-m)) \tag{2.8}$$

gdzie predykcja dla obecnego stanu układu dokonywana jest na bazie $k$ poprzednich wymuszeń i $m$ poprzednich stanów.




Rysunek 2.5.: Struktura sieci typu NNARX

Testy przeprowadzone z wykorzystaniem sieci typu NNARX, potwierdzają ich skuteczność w zastosowaniach dotyczących emulacji zachowań obiektów dynamicznych.

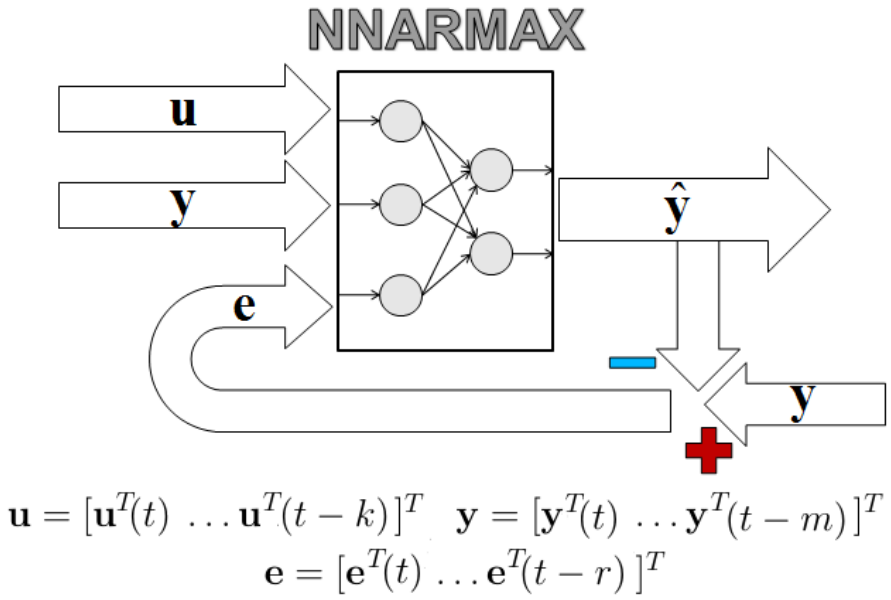
### 2.2.3. Model NNARMAX

Sieć typu NNARMAX (Nonlinear Neural AutoRegressive, Moving Average with eXogenous input) stanowi dodatkowe rozszerzenie sieci NNARX. Jak obrazuje to rys. 2.6, na wejście sieci podawany jest dodatkowo wektor uchybu $\mathbf{e}$ będący różnicą pomiędzy wartościami wyjściowymi stanu z rzeczywistego modelu, a wartościami otrzymanymi w wyniku predykcji dokonanej



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

przez sieć. Wektor ten jest wyznaczany dla założonej odgórnie liczby $r$ poprzednich chwil czasowych.




Rysunek 2.6.: Struktura sieci typu NNARMAX

Zależność pomiędzy wektorem wyjścia i wejścia dla sieci typu NNARMAX przedstawić można zatem za pomocą funkcji:

$$\hat{\mathbf{y}}(t) = \mathbf{f}(\mathbf{u}, \mathbf{y}, \mathbf{e}) = \mathbf{f}(\mathbf{u}(t), \mathbf{u}(t-1), \ldots, \mathbf{u}(t-k), \mathbf{y}(t), \mathbf{y}(t-1), \ldots, \mathbf{y}(t-m), \mathbf{e}(t), \mathbf{e}(t-1), \ldots, \mathbf{e}(t-r)) \quad (2.9)$$

która w porównaniu z formułą dla sieci NNARX, zawiera dodatkowo $r$ poprzednich wartości uchybu $\mathbf{e}$. Wektor uchybu $\mathbf{e}$ dla każdej z chwil czasowych $r$ określana jest jako różnica pomiędzy wektorem wyjściowych sieci $\hat{\mathbf{y}}$, a rzeczywistym wektorem wyjściowym układu $\mathbf{y}$. Predykcja obecnego stanu układu $\hat{\mathbf{y}}$ dokonywana jest zatem na podstawie $k$ poprzednich wymuszeń, $m$ poprzednich stanów oraz $r$ poprzednich uchybów. Jak można się domyślać, dokładność sieci tego typu jest lepsza, głównie dzięki uwzględnieniu w obliczeniach wspomnianej korygującej wartości błędu. Związane jest to jednak z pewnymi dodatkowymi problemami natury technicznej. Przede wszystkim, sieć tego typu zawiera więcej parametrów wejściowych, a co za tym idzie jest ona obszerniejsza - wymaga zatem większej mocy obliczeniowej. Ponadto, próby bezpośredniego nauczenia sieci na bazie posiadanych próbek mogą okazać się bezowocne, głównie z uwagi na znaczne i niepoprawne wartości różnicy $e$ w pierwszych stadiach nauki. Standardowa metoda uczenia sieci typu NNARMAX sprowadza się zatem do nauczenia sieci NNARX, a następnie douczenia jej rozszerzonej wersji uwzględniającej wartości uchybu, wykorzystując w tym celu wartości wyjściowe uzyskanych w procesie testów sieci NNARX.

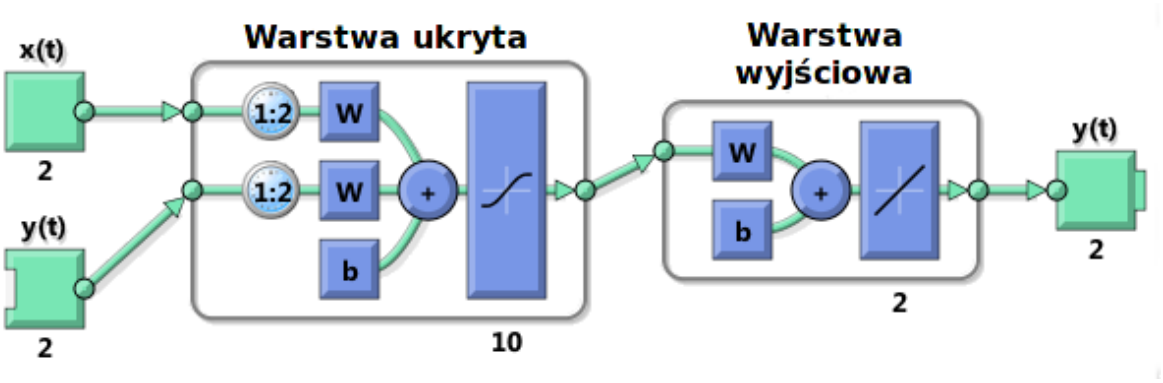


BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

### Modyfikacja powiązań między wejściami i wyjściami sieci

Dla zaprezentowanych powyżej modeli sieci neuronowych typu NNARX i NNARMAX, wprowadzana jest często dodatkowa modyfikacja umożliwiająca zwiększenie autonomiczności emulatora kosztem jego dokładności.

Standardowo nauczona sieć typu NNARX (Rys. 2.7), na wejście przyjmuje wektor wymuszeń (w tym wypadku $\mathbf{x}(t)$) oraz wektor stanu układu $\mathbf{y}(t)$. Po przeliczeniu podanych wartości poprzez nauczone macierze wag sieci, na wyjściu otrzymuje się estymowaną wartość stanu układu dla chwili następnej, z dokładnością uzależnioną od poziomu nauczenia sieci. Sieć działająca w ten sposób wymusza jednak dalsze zastosowanie modelu bądź układu, na bazie którego została nauczona - w każdej chwili czasowej pobierane są z niego wartości stanu układu, które podawane są na wejście sieci. Powiązanie tego typu nosi miano „Series-Parrallel" (SP).



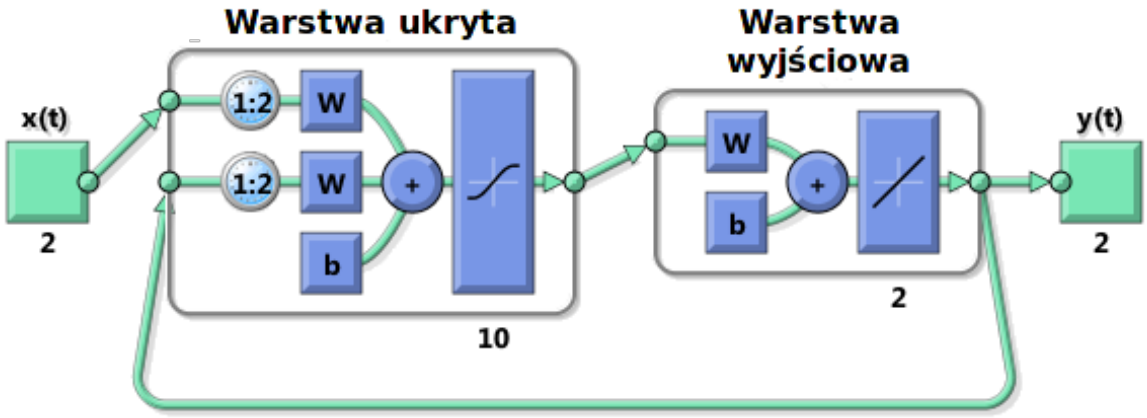

Rysunek 2.7.: Struktura powiązań „Series Parallel" (SP) sieci typu NNARX

Modyfikacja polegająca na wyprowadzeniu wartości estymowanej z wyjścia sieci, a następnie podaniu jej na wejście sieci w miejsce wektora stanu pobranego z zewnętrznego modelu nosi miano „Parallel". W takim przypadku pojawiają się jednak dwa problemy. Po pierwsze, podobnie jak w przypadku sieci typu NNARMAX, nie jest możliwe nauczenie sieci typu „Parallel" w sposób bezpośredni, z uwagi na brak powiązania z rzeczywistym modelem w początkowym stadium uczenia. Rozwiązanie w tym wypadku polega na nauczeniu najpierw sieci z powiązaniem typu SP, a następnie przetransformowanie już nauczonej sieci do postaci P poprzez modyfikację powiązań pomiędzy warstwami.

Po drugie, w przypadku powiązania typu P, jeżeli sieć nie jest dostatecznie dobrze nauczona, albo wartości stanu układu zbliżą się do wartości granicznych (wynikających z próbek służących do uczenia sieci), możliwym jest, że błędy wynikające z niedokładności sieci nawarstwią się, a co za tym idzie wartości wyjściowe sieci staną się rozbieżne z rzeczywistymi. Jedynym rozwiązaniem w takim wypadku, jest porównanie odpowiedzi sieci z rzeczywistym stanem modelu, w celu upewnienia się o prawidłowym jej funkcjonowaniu.



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ




Rysunek 2.8.: Struktura powiązań „Parallel" sieci typu NNARX

### Ogólne wprowadzenie do rodzajów sygnałów wymuszenia wykorzystywanych do uczenia sieci

Poniżej przedstawione zostały podstawowe metody generacji sygnałów wymuszenia. Zarówno wielkości wejściowe (wymuszenie) jak i wyjściowe (odpowiedź) w ten sposób uzyskane, wykorzystywane są następnie w procesie nauki odpowiedniej sieci neuronowej.

- **Generacja ze stałym krokiem**

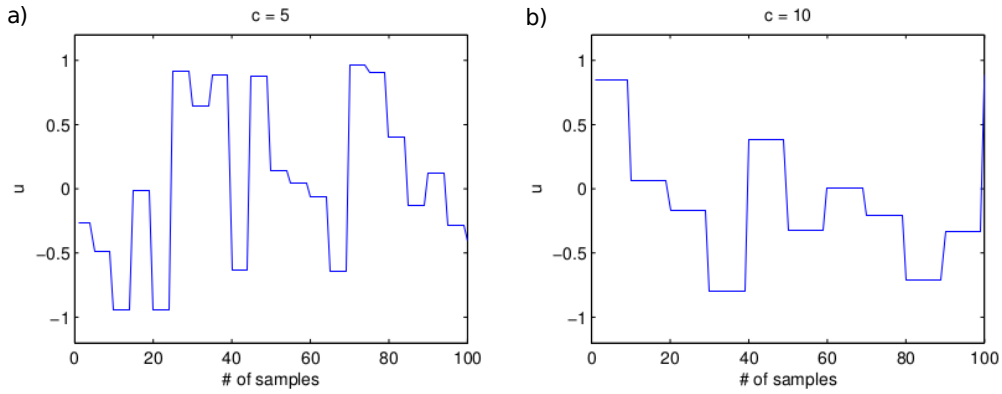
  Założenie metody generacji sygnału wymuszenia ze stałym krokiem jest proste - nowa wartość sygnału losowana jest spośród wartości ze zdefiniowanego przedziału, co stałą (odgórnie zadaną) liczbę próbek. Jednakże pomimo swojej prostoty, metoda ta nie jest często stosowana do generacji wzorców uczących, głównie z uwagi na pomijanie niektórych częstotliwości w wygenerowanym sygnale, będące wynikiem zastosowania stałego, a nie losowo zmiennego kroku czasowego. Na rysunku 2.9 przedstawione są dwa wygenerowane tą metodą sygnały, różniące się między sobą ilością próbek $c$, co które losowana jest nowa wartość sygnału.

- **Generacja ze zmiennym krokiem**

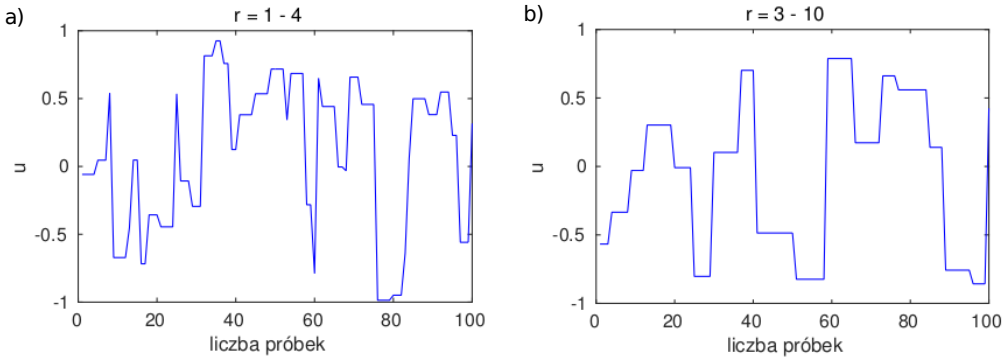
  W przypadku generacji sygnału ze zmiennym krokiem czasowym, koniecznym jest zdefiniowanie minimalnej oraz maksymalnej wartości próbek, przy których następować będzie zmiana wartości sygnału. Za każdym razem, gdy zmiana taka następuje, z podanego zakresu losowana jest jedna wartość, która determinuje dla ilu kolejnych próbek wartość sygnału pozostanie niezmieniona. Przykład sygnałów wygenerowanych tą metodą przedstawia rysunek 2.10. Zaznaczyć należy, iż dla przedstawionego przykładu założono jednorodny rozkład prawdopodobieństwa dla wartości z losowanego przedziału.



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ




Rysunek 2.9.: Przykładowy sygnał ze stałym krokiem czasowym zmiany $c$
a) zmiana co 5 próbek
b) zmiana co 10 próbek



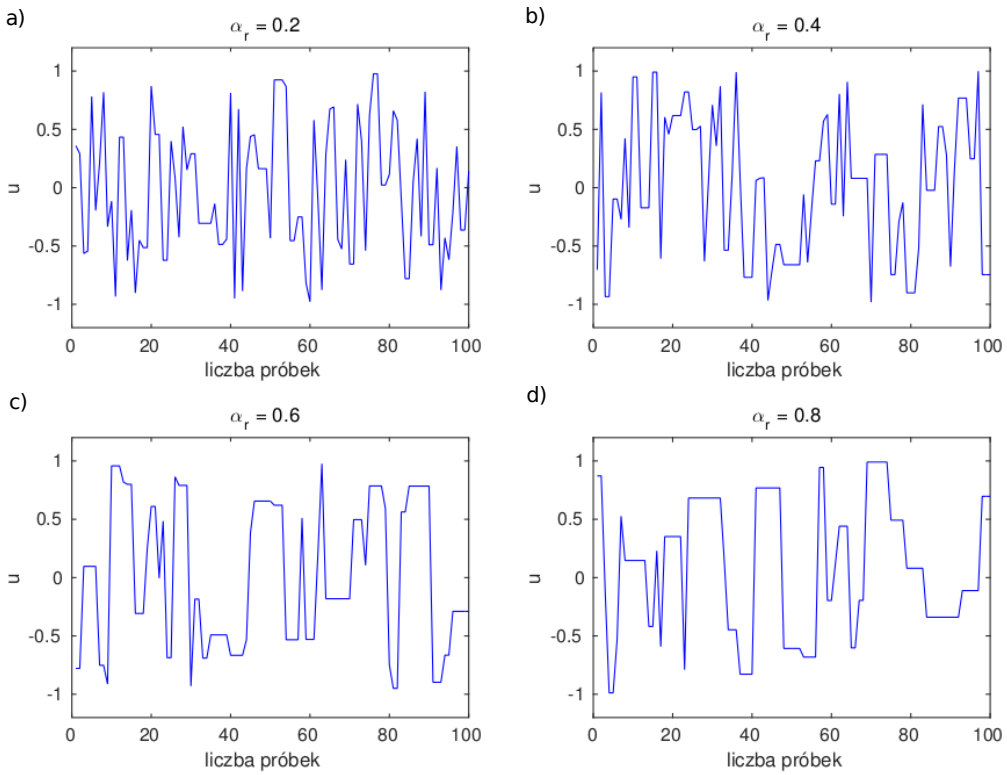

Rysunek 2.10.: Przykładowy sygnał ze zmiennym krokiem czasowym zmiany, z losowego przedziału $r$
a) zmiana co 1 - 4 próbek
b) zmiana co 3 - 10 próbek

- **Generacja ze zmiennym krokiem, parametr $\alpha_r$**

  Podobnie jak i w przedstawionej powyżej metodzie generacji sygnału ze zmiennym krokiem, zmiana wartości następuje co losowo wybraną liczbę próbek. Różnica w tym wypadku polega na uzależnieniu tej zmiany nie od wartości losowanej z zadanego przedziału, lecz od wybranego odgórnie parametru $\alpha_r$. Dla każdego kroku czasowego $\delta t$ losowana jest wartość $\alpha$ z przedziału $(0, 1)$. Jeżeli $\alpha \leq \alpha_r$ wartość sygnału pozostaje niezmieniona, w przeciwnym wypadku losowana jest nowa wartość. Rysunek 2.11 obrazuje w jaki sposób zmiana parametru $\alpha_r$ wpływa na ostateczny wygląd wygenerowanego sygnału.



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ




Rysunek 2.11.: Przykładowy sygnał ze zmiennym krokiem czasowym zmiany, uzależnionym od parametru $\alpha_r$
a) $\alpha_r = 0.2$, b) $\alpha_r = 0.4$, c) $\alpha_r = 0.6$, d) $\alpha_r = 0.8$

- **Wymuszenie typu „chirp”**

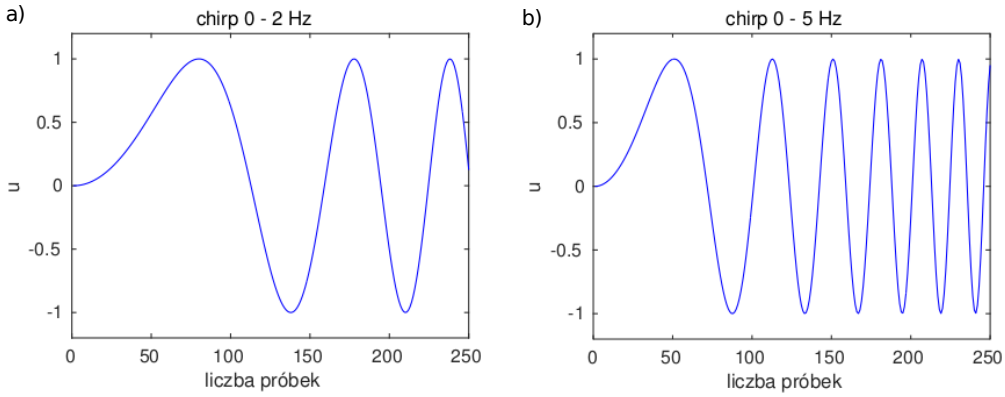
  W przeciwieństwie do przedstawionych powyżej wymuszeń, wymuszenie typu „chirp” nie jest losowe, lecz w pełni zdeterminowane - przedstawić je można za pomocą następującej formuły:

$$u(t) = u_0 + A\sin(2\pi f t) = u_0 + A\sin\left[2\pi(f_0 + (f_k - f_0)\frac{t}{T})t\right] \tag{2.10}$$

  gdzie $u_0$ i $A$ - to odpowiednio - wartość średnia i amplituda wymuszenia, natomiast $f_0$, $f_k$ i $T$ to kolejno częstotliwość początkowa, częstotliwość końcowa oraz czas przejścia od początkowej do końcowej częstotliwości. Wymuszenie tego typu znajduje swoje zastosowanie w uczeniu sieci neuronowej głównie z uwagi na możliwość zawarcia w nim pełnego zakresu częstotliwości dla konkretnego przypadku uczenia. Przykładowe wymuszenie typu „chirp” przedstawione jest na rysunku 2.12.



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ




Rysunek 2.12.: Wymuszenie typu „chirp"
a) zmiana w zakresie $0 - 2$ Hz
b) zmiana w zakresie $0 - 5$ Hz

## 2.3. Algorytmy uczenia sieci neuronowych

Ostatnim z kroków niezbędnych do nauczenia sieci neuronowej, jest dobór adekwatnego algorytmu uczącego, który bazując na przygotowanych uprzednio wzorcach uczących doprowadzi do minimalizacji błędu sieci, poprzez stopniową modyfikację jej macierzy wag $\mathbf{W_n}$. Najczęściej stosowanym kryterium optymalizacji w przypadku dynamicznych sztucznych sieci neuronowych jest kryterium minimalizacji błędu średniokwadratowego, zapisywane dla układów o jednym wyjściu zazwyczaj w postaci formuły [12]:

$$E = \frac{1}{2N}\sum_{n=1}^{N} e^2 = \frac{1}{2N}\sum_{n=1}^{N}[y - \hat{y}]^2 \tag{2.11}$$

gdzie $N$ jest ilością wzorców uczących, natomiast $y$ oraz $\hat{y}$ to odpowiednio wartość wyjściowa z modelu oraz wartość wyjściowa z sieci. Warunek ten jest często stosowany w uczeniu sieci neuronowych z uwagi na prostotę jego implementacji w algorytmach służących do modyfikacji poszczególnych wartości macierzy wag. W przypadku, gdy wyjście układu nie jest skalarem lecz wektorem, kryterium minimalizacji zakłada się w formie:

$$E = \frac{1}{2N}\sum_{n=1}^{N} \mathbf{e}^T\mathbf{e} = \frac{1}{2N}\sum_{n=1}^{N}[\mathbf{y} - \hat{\mathbf{y}}]^T[\mathbf{y} - \hat{\mathbf{y}}] \tag{2.12}$$

Na potrzeby niniejszej pracy zaimplementowano oraz przetestowano szereg tego typu algorytmów, poczynając od algorytmu gradientowego spadku, przez klasyczny algorytm Gaussa - Newtona, zmodyfikowaną jego wersję (quasi-Newtona), oraz algorytm Lavenberga - Margquartda ze zmiennym krokiem zbieżności [36]. Wszystkie wymienione algorytmy działają na zasadzie „propagacji wstecznej", co oznacza że błąd wyjściowy propagowany jest wstecz poprzez wszystkie warstwy sieci (wyjściową i ukryte), gdzie następnie na jego bazie dokonywana jest modyfikacja macierzy wag pomiędzy warstwami. W załączniku przedstawiono dokładny opis



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

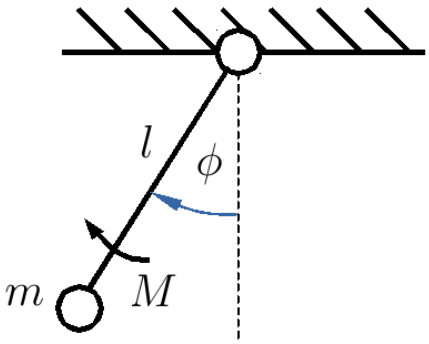
dla podstawowego algorytmu gradientowego spadku. Pozostałe algorytmy stanowią rozszerzenie tego algorytmu, w szczególności rozpatrując średniokwadratowe kryterium minimalizacji z wykorzystaniem aproksymacji drugiego, a nie pierwszego rzędu. Oznacza to, że w kryterium oprócz pochodnych cząstkowych pierwszego rzędu (składających się na gradient), pojawiają się również pochodne cząstkowe rzędu drugiego, zebrane w tak zwaną macierz Hessego. Dokładny opis tych algorytmów znaleźć można w literaturze [10], [37], [38].

## 2.4. Emulacja neuronowa na przykładzie wybranych modeli dynamicznych

Bazując na przedstawionych powyżej informacjach na temat modeli, struktury oraz algorytmów uczenia dynamicznych sztucznych sieci neuronowych, niniejszy podrozdział zawiera przykłady emulacji neuronowej dla szeregu wybranych modeli układów dynamicznych, zarówno liniowych jak i nieliniowych. Emulowane modele to kolejno nieliniowe wahadło matematyczne, giroskop o dwóch stopniach swobody oraz belka typu „smart" z piezoelektrycznymi czujnikami i aktuatorami. Każdy z przedstawionych modeli znajdzie również swoje rozszerzenie w rozdziale poświęconym sterowaniu neuronowemu.

### 2.4.1. Wahadło matematyczne

Pierwszym z modeli dynamicznych wybranych na potrzeby zaprezentowania uczenia oraz dokładności emulacji za pomocą sieci neuronowych, jest model wahadła matematycznego (Rys. 2.13). Wybór ten podyktowany został w głównej mierze chęcią zbadania nieliniowych zachowań dynamicznych wahadła oraz sprawdzenia zakresu i możliwości emulacji wspomnianych zachowań przez sieć neuronową.




Rysunek 2.13.: Model wahadła matematycznego

Równanie nieliniowe modelu (z pominięciem elementów dyssypacyjnych) przedstawić można w następującej postaci:



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

$$\ddot{\phi} + \frac{g}{l}\sin\phi = \frac{M}{ml^2} \tag{2.13}$$

gdzie $l$ i $m$ - długość i masa wahadła, $g$ - przyspieszenie ziemskie, natomiast $M$ - wymuszenie w postaci momentu, przyłożone zgodnie z kątem $\phi$. Po linearyzacji wokół punktu zerowego (odpowiadającemu kątowi $\phi = 0$ rad), zlinearyzowane równanie jednorodne wahadła przyjmuje postać:

$$\ddot{\phi} + \frac{g}{l}\phi = 0 \tag{2.14}$$

Z równania 2.14 w bezpośredni sposób wyznaczyć można częstość własną drgań wahadła, która w zakresie liniowym jest stała i zależna wyłącznie od długości wahadła $l$ oraz wartości przyspieszenia ziemskiego $g$:

$$\omega = \sqrt{\frac{l}{g}} \tag{2.15}$$

Wyznaczenie częstości własnej wahadła w ujęciu nieliniowym stanowi bardziej złożony problem - jego rozwiązanie znaleźć można przykładowo w pracy [39]. Z rozwiązania równania (2.13) w postaci jednorodnej oraz przy dodatkowym założeniu zerowej prędkości początkowej dla kąta $\phi$, wynika że okres drgań wahadła, a co za tym idzie również jego częstość własna, zależą od przyjętych warunków początkowym, w tym wypadku od wartości początkowej kąta $\phi$. Wzór na okres drgań otrzymuje się w postaci całki eliptycznej

$$T = 2\sqrt{\frac{l}{g}}\int_0^{x_0} \frac{dx}{\sqrt{\sin^2\frac{x_0}{2} - \sin^2\frac{x}{2}}} \tag{2.16}$$

którą to po dodatkowych przekształceniach rozwinąć można ostatecznie do postaci następującej sumy nieskończonej:

$$T = 2\pi\sqrt{\frac{l}{g}}\left[1 + \left(\frac{1}{2}\right)^2 \sin^2\frac{\phi_0}{2} + \left(\frac{1\cdot 3}{2\cdot 4}\right)^2 \sin^4\frac{\phi_0}{2} + \ldots\right] \tag{2.17}$$

gdzie $\phi_0$ jest kątem początkowym wahadła. Celem naszym w tym przypadku jest sprawdzenie czy emulator neuronowy nauczony w pełnym zakresie kątowym wahadła jest w stanie w dokładny sposób odtworzyć nieliniowe jego zachowanie. Na potrzeby obliczeń założono długość wahadła $l = 0.2$ m, natomiast jego masę przyjęto równą $m = 0.1$ kg.

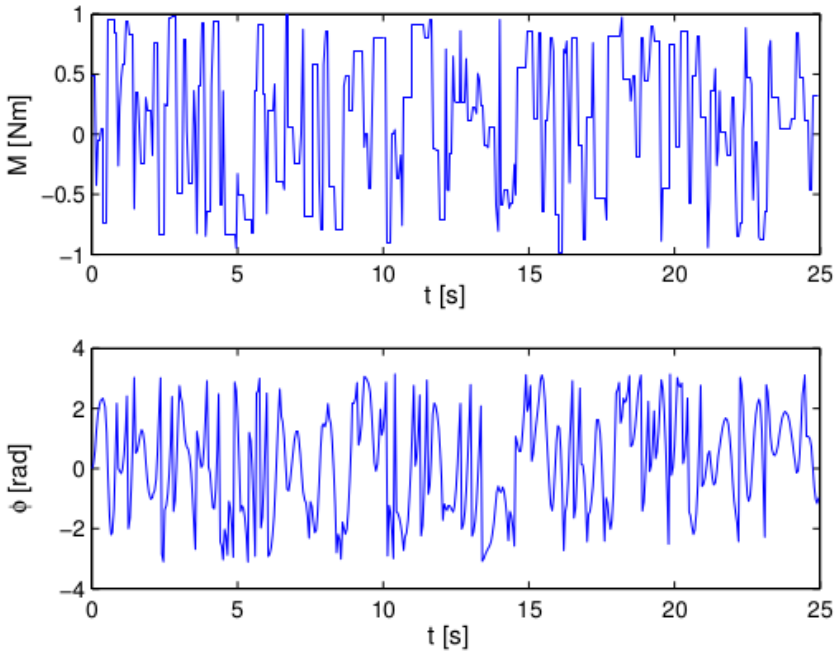
### Uczenie sieci neuronowej

Przed przystąpieniem do uczenia sieci neuronowej, konieczne jest wygenerowanie wzorców, na bazie których sieć zostanie nauczona. W tym przypadku wymuszenie momentem $M$ wygenerowane zostało za pomocą metody ze zmiennym krokiem przy parametrze $\alpha_r = 0.65$. Odpowiedź układu wyznaczono na bazie modelu matematycznego z wykorzystaniem metody



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

Rungego-Kutty 4-go rzędu, z krokiem czasowym $\Delta t = 10$ ms. Wzorce uczące w ten sposób wygenerowane pokazane są na rysunku 2.14




Rysunek 2.14.: Wygenerowane wzorce uczące dla wahadła matematycznego

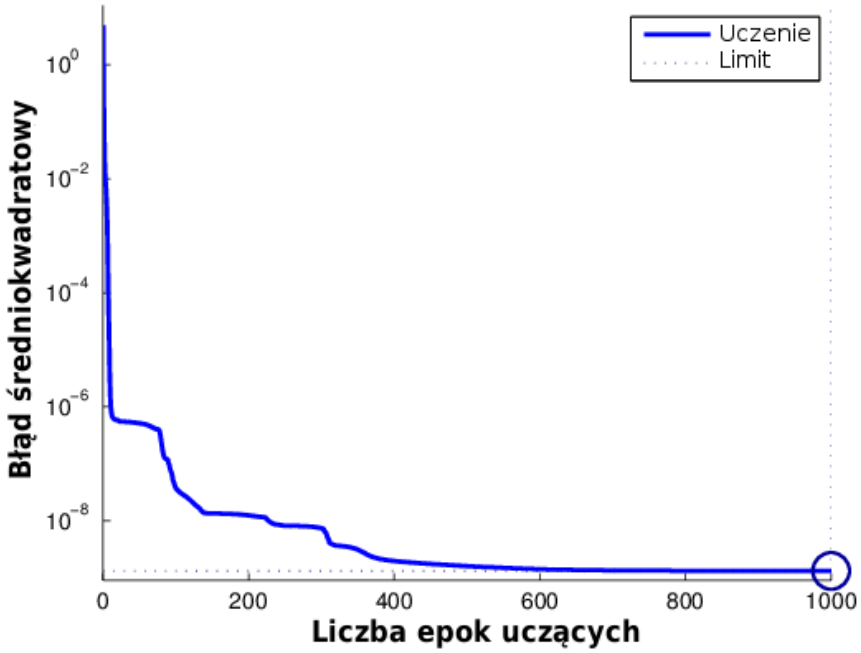
Zakładając model sieci typu NNARX 15-1 (15 neuronów w warstwie ukrytej, 1 neuron w wyjściowej) oraz identyczną ilość regresorów dla wymuszenia oraz dla położenia kąta $\phi$ ($k = m = 2$), można przystąpić do uczenia sieci. Poniżej (Rys. 2.15) przedstawiona jest krzywa uczenia emulatora, z wykorzystaniem metody uczenia Lavenberga-Margquardta z warunkiem minimalizacji błędu średniokwadratowego. Warunek maksymalnej ilości epok uczących (1000) został na sieć narzucony odgórnie. Przy tym warunku wyraźnie widać zbieżność rozwiązania sieci, włącznie z gwałtownym oraz stopniowym wzrostem dokładności uzyskiwanym przez sieć w odpowiednio początkowej oraz kolejnych fazach uczenia.

### Emulator neuronowy wahadła

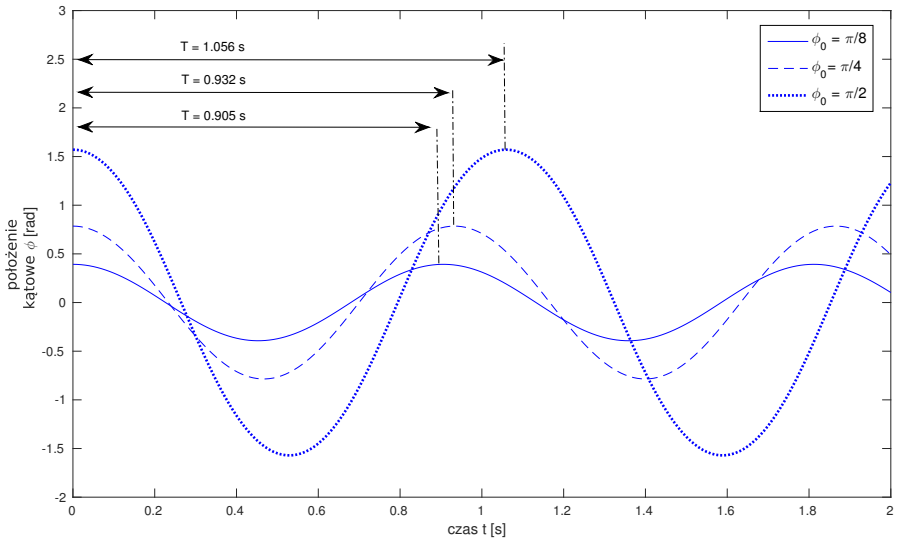
Rysunek 2.16 przedstawia wykresy odpowiedzi sieci neuronowej emulującej nieliniową dynamikę analizowanego wahadła matematycznego. Z wykresów tych widać, że okres drgań emulatora zależny jest od początkowego położenia kąta $\phi$, a jego wartość pokrywa się z wartością otrzymaną z rozwiązania analitycznego (2.17). Wyznaczone wartości okresu oraz częstości własnych dla przedstawionych położeń startowych kąta $\phi$ zebrane zostały w tabeli 2.1.



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ




Rysunek 2.15.: Krzywa uczenia emulatora neuronowego wahadła matematycznego (błąd średniokwadratowy)




Rysunek 2.16.: Odpowiedź emulatora neuronowego wahadła matematycznego przy różnych warunkach początkowych (położenie startowe $\phi_0$)



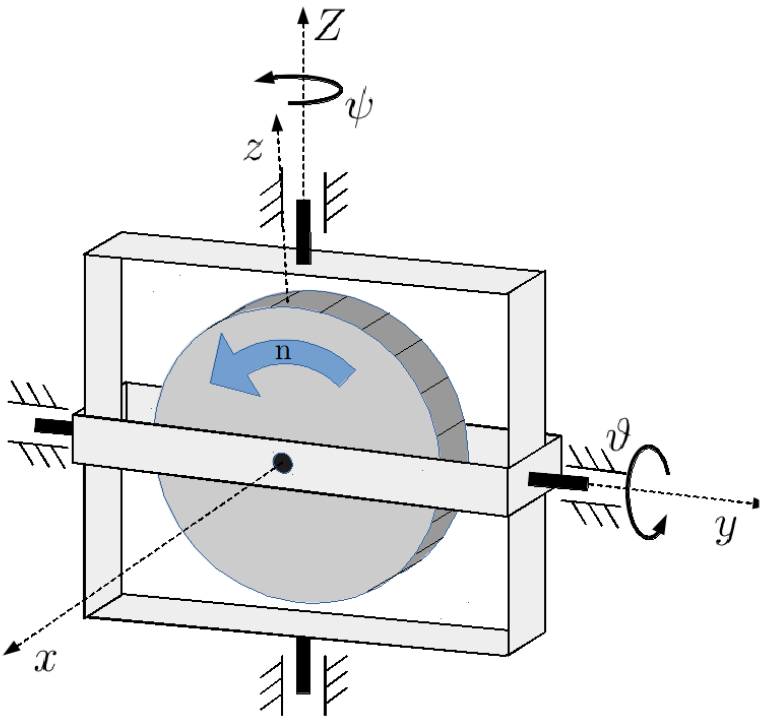
BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

Tablica 2.1.: Wartości okresu i częstości własnych wahadła, przy zadanych początkowych położeniach kąta $\phi$

| $\phi_0$ [rad] | T [s] | $\omega$ [rad/s] |
|---|---|---|
| $\pi/8$ | 0.905 | 6.396 |
| $\pi/4$ | 0.932 | 6.734 |
| $\pi/2$ | 1.056 | 5.950 |

### 2.4.2. Giroskop

Drugim z modeli rozpatrywanych pod kątem emulacji neuronowej jest giroskop w zawieszeniu Cardana o dwóch osiach obrotu przedstawiony na rysunku 2.17 [40]. Opis oraz zachowanie dynamiczne tego typu giroskopów są szeroko dostępne w literaturze [41], [42], [43], [44], włącznie z publikacjami poświęconymi implementacji klasycznych metod sterowania do zmiany orientacji kątów ramki wewnętrznej i zewnętrznej [42], [45].




Rysunek 2.17.: Model giroskopu w zawieszeniu Cardana z dwiema osiami obrotu

Model giroskopu wybrany został na potrzeby emulacji oraz sterowania neuronowego z uwagi na występujący w nim efekt giroskopowy. Wraz ze wzrostem prędkości kątowej wirnika giroskopu, zwiększeniu ulega również jego częstość własna, w związku z tym zwiększa się stopień trudności aktywnego sterowania. Dodatkową zaletą wybranego modelu jest jego zachowanie nieliniowe, co pozwala na przetestowanie zarówno emulacji jak i skuteczności sterowania



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

neuronowego w pełniejszym nieliniowym zakresie.

Nieliniowe równania dynamiki przedstawionego giroskopu, z ich wyprowadzeniem za pomocą metody d'Alemberta, zostały podane w pracy [41] i mają one postać:

$$\begin{cases} I_y\ddot{\vartheta} + h\dot{\psi}\cos\vartheta + (I_z - I_x)\dot{\psi}^2\sin\vartheta\cos\vartheta = M_y \\ (I_{oZ} + I_z\cos^2\vartheta + I_x\sin^2\vartheta)\ddot{\psi} - h\dot{\vartheta}\cos\vartheta + 2(I_x - I_z)\dot{\psi}\dot{\vartheta}\sin\vartheta\cos\vartheta = M_Z \end{cases} \tag{2.18}$$

gdzie $\vartheta$ i $\psi$ są odpowiednio kątami obrotu ramki wewnętrznej i zewnętrznej, $I_x$, $I_y$ oraz $I_z$ to momenty bezwładności ramki wewnętrznej i wirnika wokół osi $x$, $y$ i $z$, $I_{oZ}$ - moment bezwładności ramki zewnętrznej względem osi $Z$, natomiast $M_y$ oraz $M_Z$ - wymuszenia w postaci momentów przyłożone względem osi $y$ i $Z$. Parametr $h$ w równaniu (2.18) jest to moment pędu wirnika i może być przedstawiony za pomocą zależności:

$$h = I_{rx}n \tag{2.19}$$

w której $I_{rx}$ jest momentem bezwładności wirnika, natomiast $n$ jest prędkością kątową wirowania.

Na potrzeby niniejszej emulacji, przyjęto następujące parametry giroskopu:

$$I_x = 1.6 \cdot 10^{-4}\ \mathrm{kg}\cdot\mathrm{m}^2,\ I_y = 1.2 \cdot 10^{-4}\ \mathrm{kg}\cdot\mathrm{m}^2$$

$$I_z = 1.2 \cdot 10^{-4}\ \mathrm{kg}\cdot\mathrm{m}^2,\ I_{rx} = 10^{-4}\ \mathrm{kg}\cdot\mathrm{m}^2,\ I_{oZ} = 10^{-5}\ \mathrm{kg}\cdot\mathrm{m}^2$$

Ponadto, w celu uproszczenia modelu, założono że prędkość kątowa wirnika $n$ jest stała i niezmienna w czasie, wymuszenie w postaci momentu $M_y$ przyłożone jest wyłącznie względem osi $y$, moment $M_Z$ jest natomiast pominięty.

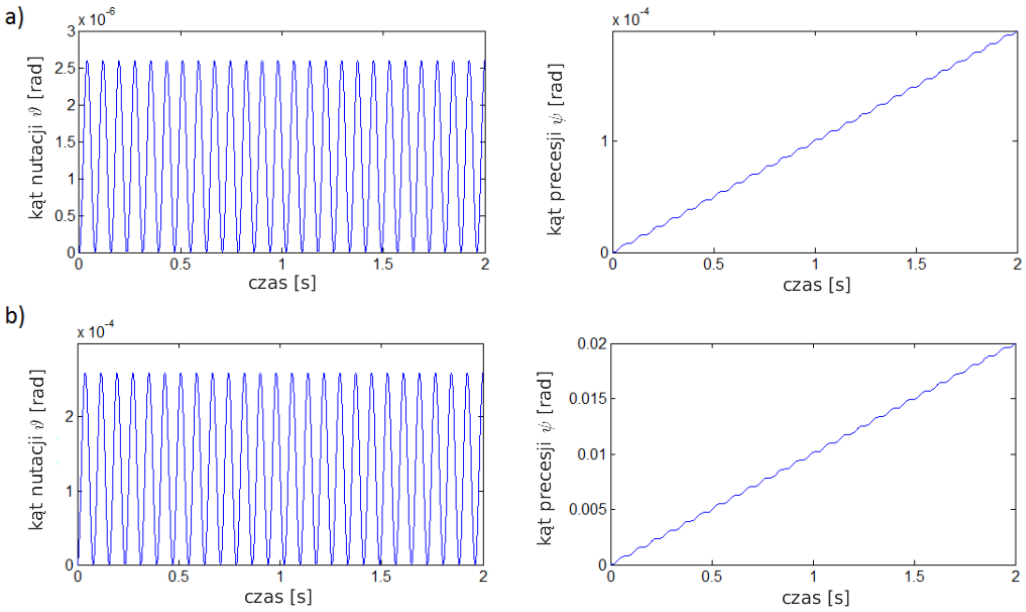
Podobnie jak w przypadku emulacji wahadła matematycznego, wzorce uczące wygenerowane zostały wykorzystując model giroskopu, korzystając z metody Rungego-Kutty z krokiem czasowym $\Delta t = 0.1$ ms oraz przyjmując że moment $M_y$ zawiera się w przedziale $\left(-10^{-4}, 10^{-4}\right)$ Nm. Przykładowa odpowiedź giroskopu, w postaci kąta nutacji $\vartheta$ oraz precesji $\psi$, na wymuszenie momentem skokowym $M_y = 10^{-6}$ oraz $M_y = 10^{-4}$ Nm przy prędkości kątowej wirnika wynoszącej $n = 100$ rad/s, przedstawiona jest na rysunku 2.18.

### Emulator giroskopu

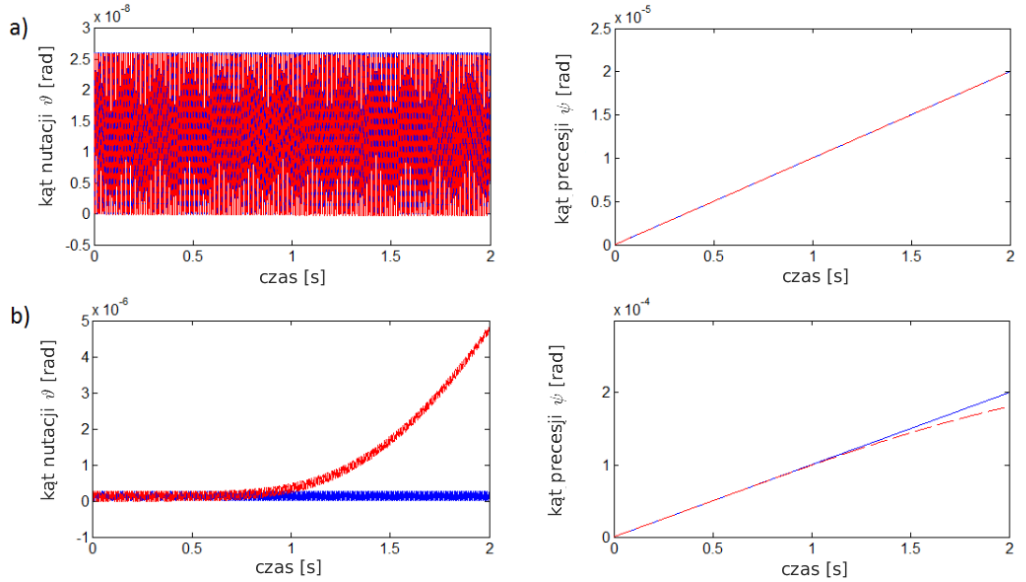
Na potrzeby emulacji neuronowej giroskopu, przetestowano wszystkie z zaprezentowanych uprzednio modeli sieci: NNFIR, NNARX oraz NNARMAX. Dla każdego z modeli przyjęto identyczną ilość regresorów: $k = 2$, $m = 2$ oraz $r = 2$. Jak można się było spodziewać emulacja neuronowa nie powiodła się w przypadku sieci NNFIR. W pozostałych przypadkach, sieć była w stanie poprawnie nauczyć się zachowań dynamicznych giroskopu oraz emulować je z dobrą dokładnością w zakresie kątowym, w którym przeprowadzono uczenie sieci.



BPK BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ




Rysunek 2.18.: Odpowiedź giroskopu w postaci kątów nutacji $\vartheta$ oraz precesji $\psi$ na wymuszenie skokowym momentem $M_y$
a) $n = 100$ rad/s, $M_y = 10^{-6}$ Nm
b) $n = 100$ rad/s, $M_y = 10^{-4}$ Nm




Rysunek 2.19.: Odpowiedź giroskopu (niebieska linia) oraz emulatora (czerwona linia) w postaci kątów nutacji $\vartheta$ oraz precesji $\psi$ na wymuszenie skokowym momentem $M_y$
a) $n = 1000$ rad/s, $M_y = 10^{-6}$ Nm
b) $n = 1000$ rad/s, $M_y = 10^{-5}$ Nm

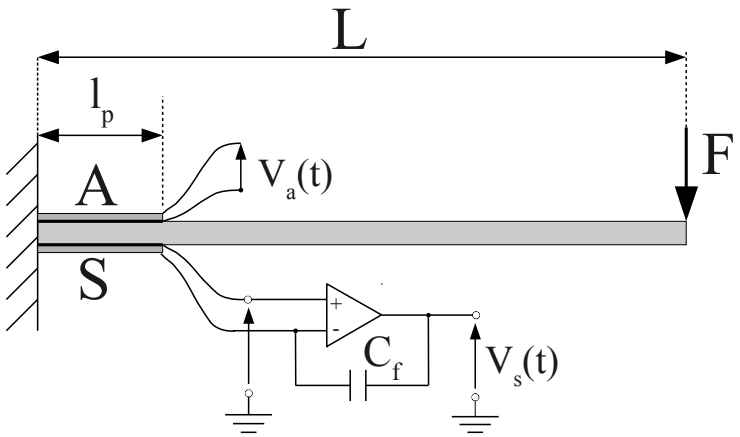


BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

Na rysunku 2.18 przedstawiono odpowiedź emulatora neuronowego oraz giroskopu na wymuszenie stałym momentem $M_y$. Z rysunku tego widać, że w przypadku gdy odpowiedź emulatora (kąt nutacji $\vartheta$ oraz precesji $\psi$) zawiera się w przedziale, w którym sieć została nauczona, wykresy otrzymane z emulatora neuronowego pokrywają się z wykresami uzyskanymi w wyniku całkowania równań ruchu 2.18. W momencie gdy zakres ten zostanie przekroczony, odpowiedź emulatora zaczyna rozbiegać się z wartościami wyznaczonymi z modelu (Rys. 2.19 b). W przypadku dyskutowanej emulacji, zdecydowano się na ograniczony zakres uczenia sieci, głównie w celu zaprezentowania ograniczeń narzuconych na emulator neuronowy wynikających z przyjętych kryteriów i parametrów jego nauki. Szerszy zakres uczenia giroskopu zawarty jest w dalszej części pracy, w rozdziale poświęconym neuronowym metodom jego sterowania.

### 2.4.3. Belka typu „smart"

W celu zbadania procesu emulacji w przypadku układu o parametrów rozłożonych (układu ciągłego), w obecnym punkcie będzie rozważany model belki wspornikowej z elementami piezoelektrycznymi umieszczonymi w sąsiedztwie punktu utwierdzenia belki wspornikowej, pokazany na rysunku 2.20. Belka taka nosi niekiedy miano „belki inteligentnej" (ang. „smart beam"), z uwagi na wykorzystanie elementów piezoelektrycznych oraz metod aktywnego tłumienia drgań. Piezoelektryki działają w badanym układzie na zasadzie powiązanej pary - jeden z nich funkcjonuje jak czujnik (S), drugi jako aktuator (A) [27].




Rysunek 2.20.: Uproszczony model belki wspornikowej z elementami piezoelektrycznymi (A - aktuator, S - czujnik) oraz przedwzmacniaczem ładunku ($C_f$)



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

Wymiary oraz własności materiałowe belki dla podanej analizy zostały przyjęte jak następuje:

$$\begin{aligned} L &= 1.5 \cdot 10^{-1} \text{ m} - \text{długość belki} \\ h &= 1.5 \cdot 10^{-3} \text{ m} - \text{szerokość belki} \\ b &= 10^{-2} \text{ m} - \text{wysokość belki} \\ \rho &= 7.9 \cdot 10^{3} \text{ kg/m}^3 - \text{gęstość materiału belki,} \\ E &= 2.1 \cdot 10^{11} \text{ N/m}^2 - \text{moduł Younga} \end{aligned}$$

Równania dla piezoelektrycznego aktuatora i czujnika, przedstawić można odpowiednio w następującej postaci [27]:

$$M(t) = cV_a(t) \tag{2.20}$$

$$V_s(t) = s \left[ \left. \frac{\partial \mathrm{w(x,t)}}{\partial \mathrm{x}} \right|_{x=x_b} - \left. \frac{\partial \mathrm{w(x,t)}}{\partial \mathrm{x}} \right|_{x=x_a} \right] \tag{2.21}$$

Równanie 2.20 opisuje zależność pomiędzy napięciem $V_a(t)$ przyłożonym do aktuatora piezoelektrycznego $A$, oraz wytworzonym w wyniku tego działającym na końcach aktuatora momentem zginającym $M(t)$ proporcjonalnym do wartości przyłożonego napięcia. Z kolei równanie 2.21 definiuje zależność pomiędzy kątem ugięcia na obu końcach piezoelektrycznego czujnika ($x_a$, $x_b$) oraz wygenerowanym w wyniku tego odkształcenia napięciem $V_s(t)$. Wartości współczynników proporcjonalności $c$ i $s$ występujących w równaniach 2.20 i 2.21 są określone przez następujące wyrażenia [27]:

$$c = \frac{E_p \left| d_{31} \right| b_p (h + h_p)}{2} \tag{2.22}$$

$$s = \frac{E_p \left| d_{31} \right| b_p (h + h_p)}{2} \cdot \frac{1}{C_f} \tag{2.23}$$

gdzie $h$ oznacza wysokość belki, $E_p$ i $d_{31}$ odpowiadają modułowi Younga oraz stałej piezoelektrycznej czujnika i aktuatora, natomiast $b_p$ i $h_p$ są odpowiednio szerokością i wysokością elementów piezoelektrycznych. Parametr $C_f$ występujący w równaniu 2.23 oznacza pojemność elektryczną przedwzmacniacza ładunku widocznego na rysunku 2.20. Na potrzeby niniejszej analizy wartości podanych parametrów przyjęto w następujący sposób:

$$\begin{aligned} &E_p = 50 \cdot 10^{9} \text{ N/m}^2, && d_{31} = -150 \cdot 10^{-12} \text{ C/V}, && b_p = 10^{-2} \text{ m}, \\ &h_p = 1.5 \cdot 10^{-3} \text{ m}, && l_p = 2 \cdot 10^{-2} \text{ m}, && C_f = 10^{-7} \text{ C} \end{aligned}$$

Dodatkowe informacje na temat elementów piezoelektrycznych oraz ich zastosowań jako czujników i aktuatorów w układach inteligentnych, dostępne są w literaturze [27], [46], [47].

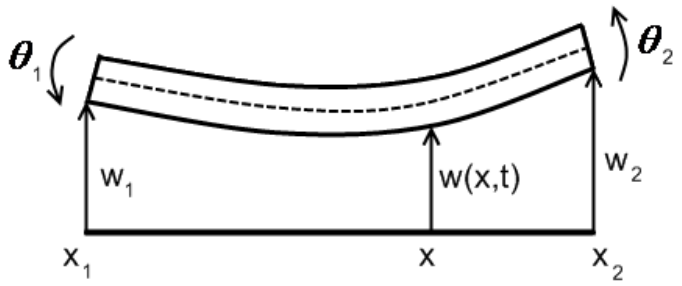


BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

### Uproszczenia modelu belki inteligentnej

Na potrzeby generacji próbek uczących dla emulatora neuronowego, przeprowadzono szereg dodatkowych uproszczeń dla modelu belki przedstawionego na rysunku 2.20. Pierwszy z kroków polegał na dyskretyzacji ciągłego modelu belki wspornikowej, korzystając z metody elementów skończonych.

Belka podzielona została na $N$ elementów o równej długości - pojedyńczy element skończony pokazano na rysunku 2.21. Dla każdego elementu zdefiniowane zostały cztery zmienne lokalne: po jednym przemieszczeniu i kącie obrotu, na początku i na końcu elementu skończonego.




Rysunek 2.21.: Założone przemieszczenia $w$ oraz kąty obrotu $\vartheta$ węzłów dla pojedynczego elementu skończonego belki

Macierze mas i sztywności dla pojedynczego elementu, dla przypadku belki zginanej przyjęto w postaci [48]:

$$M_e = \frac{\rho A l_e}{420}\begin{bmatrix} 156 & 22 & 54 & -13 \\ 22 & 4 & 13 & -3 \\ 54 & 13 & 156 & -22 \\ -13 & -3 & -22 & 4 \end{bmatrix}$$

$$K_e = \frac{EI}{l_e^3}\begin{bmatrix} 12 & 6 & -12 & 6 \\ 6 & 4 & -6 & 2 \\ -12 & -6 & 12 & -6 \\ 6 & 2 & -6 & 4 \end{bmatrix}$$

z parametrem $l_e$ odpowiadającym długości elementu skończonego oraz parametrami $A$ i $I$ będącymi odpowiednio polem przekroju poprzecznego belki oraz momentem bezwładności przekroju względem osi belki. Dokonując agregacji otrzymanych macierzy mas $\mathbf{M_e}$ i sztywności $\mathbf{K_e}$ poszczególnych elementów otrzymano macierze globalne $\mathbf{M}$ i $\mathbf{K}$ całej belki. Uwzględniając warunki brzegowe, zdyskretyzowane równanie dynamiki przedstawić można w postaci macierzowej:

$$\mathbf{M}\ddot{\mathbf{w}} + \mathbf{K}\mathbf{w} = \mathbf{F}(t) \tag{2.24}$$



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

gdzie **w** jest globalnym wektorem przemieszczeń oraz kątów obrotu:

$$\mathbf{w} = \begin{bmatrix} w_1 & \theta_1 & w_2 & \theta_2 & \dots & w_N & \theta_N \end{bmatrix}^T_{2N\times 1} \tag{2.25}$$

powstałym w wyniku agregacji współrzędnych lokalnych poszczególnych elementów. Rozmiary uzyskanych w wyniku agregacji globalnych macierzy mas **M** i sztywności **K**, zależą od liczby elementów skończonych, na które została podzielona belka podczas dyskretyzacji. Co za tym idzie, wraz ze wzrostem liczby elementów skończonych wzrasta również ilość równań opisujących dynamikę belki. Równanie (2.24) przekształcić można następnie do postaci równania stanu, które w powiązaniu z właściwie dobranym równaniem wyjścia, daje układ równań:

$$\begin{cases} \dot{\mathbf{z}} = \mathbf{A}\mathbf{z} + \mathbf{B}u + \mathbf{E}s \\ y = \mathbf{C}\mathbf{z} \end{cases} \tag{2.26}$$

gdzie **z** jest wektorem stanu układu **u** - sygnałem sterującym, natomiast **s** wymuszeniem zewnętrznym. Macierze **A**, **B**, **C** oraz **E** dla podanego układu zdefiniowane są w postaci:

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}_{2N\times 2N} & \mathbf{I}_{2N\times 2N} \\ \mathbf{M^{-1}K}_{2N\times 2N} & \mathbf{0}_{2N\times 2N} \end{bmatrix}_{4N\times 4N}$$

$$\mathbf{B} = \begin{bmatrix} 0 \\ \vdots \\ b_a \\ \vdots \\ 0 \end{bmatrix}_{4N\times 1} \qquad b_a = \frac{c}{l_e}$$

$$\mathbf{C} = \begin{bmatrix} 0 & \dots & c_s & \dots & 0 \end{bmatrix}_{1\times 4N}, \quad c_s = \frac{s}{l_e}$$

$$\mathbf{E} = \begin{bmatrix} 0 \\ \vdots \\ e_{2N-1} \\ \vdots \\ 0 \end{bmatrix}_{4N\times 1} \qquad e_{2N-1} = F$$

Wartość $b_a$ i $c_s$ odpowiadają za interakcję pomiędzy piezoelektrycznym aktuatorem bądź czujnikiem a belką wspornikową, są one zatem uwzględnione w macierzach $B$ i $C$ jako wyrazy odpowiadające położeniom końcowych elementów piezoelektrycznych. Z kolei wartość zaburzenia $e_{2N-1}$ odpowiada wymuszeniu przyłożonemu na końcu belki wspornikowej .

Jak można zauważyć na przykładzie przedstawionych macierzy, w szczególności macierzy stanu układu **A**, rozmiar jej gwałtownie wzrasta wraz ze wzrostem liczby zastosowanych ele-



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

mentów skończonych. Rozwiązaniem zastosowanym w celu redukcji ilości równań dynamiki belki, jest przetransformowanie zdyskretyzowanych globalnych macierzy mas $\mathbf{M}$ i sztywności $\mathbf{K}$ do postaci w przestrzeni modalnej, a następnie zachowanie wyłącznie równań odpowiadających $n$ najniższym częstościom własnym układu. Dodatkową zaletą przejścia do przestrzeni modalnej, jest możliwość wprowadzenia do równań dynamiki układu dodatkowej macierzy modalnej tłumienia $\mathbf{C}$, opisującej dyssypację energii w belce przy założeniu wiskotycznego charakteru tłumienia. Przejście do przestrzeni modalnej usyskuje się poprzez zastosowanie przekształcenia:

$$\mathbf{z} = \mathbf{V}\mathbf{z}_\xi$$

$$\mathbf{z} = \begin{bmatrix} \mathbf{w}(x,t) \\ \dot{\mathbf{w}}(x,t) \end{bmatrix}_{4N\times 1} \quad \mathbf{z}_\xi = \begin{bmatrix} \xi(t) \\ \dot{\xi}(t) \end{bmatrix}_{4N\times 1} \quad \mathbf{V} = \begin{bmatrix} \mathbf{T}_{2N\times 2N} & \mathbf{O}_{2N\times 2N} \\ \mathbf{0}_{2N\times 2N} & \mathbf{T}_{2N\times 2N} \end{bmatrix}_{4N\times 4N} \tag{2.27}$$

gdzie macierz $\mathbf{T}$ jest macierzą złożoną z wektorów własnych jednorodnego równania (2.24), ułożonych zgodnie z warunkiem wzrostu odpowiadających im wartości własnych. Równania stanu i wyjścia dla tak zdefiniowanych wektorów i macierzy, przekształcają się zatem do postaci:

$$\begin{cases} \dot{\mathbf{z}}_\xi = \mathbf{A_m}\mathbf{z}_\xi + \mathbf{B_m}\mathbf{u} + \mathbf{E_m}\mathbf{s} \\ y = \mathbf{C_m}\mathbf{z}_\xi \end{cases} \tag{2.28}$$

gdzie:

$$\mathbf{A_m} = \mathbf{V^{-1}}\mathbf{A}\mathbf{V}, \quad \mathbf{B_m} = \mathbf{V^{-1}}\mathbf{B}$$
$$\mathbf{C_m} = \mathbf{C}\mathbf{V}, \quad \mathbf{E_m} = \mathbf{V^{-1}}\mathbf{E}$$

Ostateczna postać macierzy modalnej $\mathbf{A_m}$ z uwzględnieniem modalnej macierzy tłumienia $\mathbf{C}$ powstałej poprzez dołączenie tłumienia modalnego do rozprzężonych w wyniku przejścia do przestrzeni modalnej równań stanu, wygląda zatem następująco:

$$\mathbf{A_m} = \begin{bmatrix} \mathbf{0}_{2N\times 2N} & \mathbf{I}_{2N\times 2N} \\ -\mathbf{\Omega^2}_{2N\times 2N} & -\mathbf{C}_{2N\times 2N} \end{bmatrix}_{4N\times 4N} \tag{2.29}$$

z macierzą kwadratów częstości własnych $\mathbf{\Omega^2}$ oraz macierzą tłumienia $\mathbf{C}$ w postaci:

$$\mathbf{\Omega^2} = \begin{bmatrix} \omega_1^2 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \omega_{2N}^2 \end{bmatrix} \tag{2.30}$$

$$\mathbf{C} = \begin{bmatrix} 2\omega_1\zeta_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 2\omega_{2N}\zeta_{2N} \end{bmatrix} \tag{2.31}$$



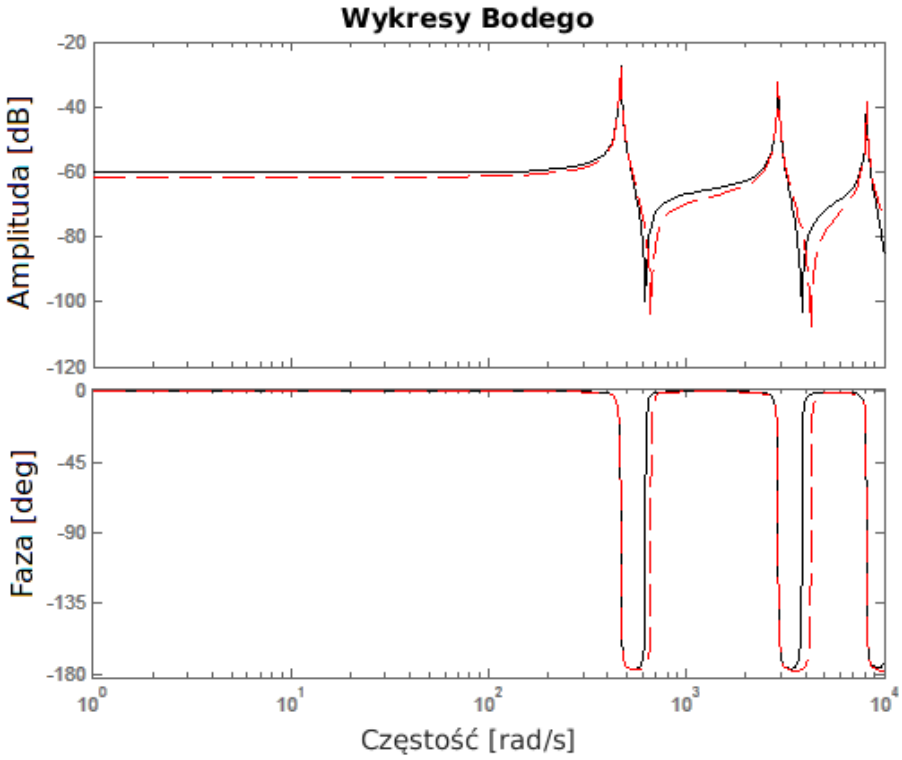
BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

z częstościami własnymi $\omega_n$ ułożonymi w kolejności rosnącej wzdłuż przekątnych obu macierzy.

Jak wcześniej wspomniano podane macierze modalne na potrzeby analizy oraz generacji wzorców uczących zredukować można dodatkowo do postaci uwzględniającej wyłącznie $m$ pierwszych form własnych. Przykładowo, poniżej pokazana jest macierz modalna $\mathbf{A_m}$ już po wspomnianej redukcji:

$$\mathbf{A_m} = \begin{bmatrix} \mathbf{0}_{m\times m} & \mathbf{I}_{m\times m} \\ -\mathbf{\Omega^2}_{m\times m} & \mathbf{C}_{m\times m} \end{bmatrix}_{2m\times 2m} \tag{2.32}$$

Na rysunku (2.22) pokazane jest porównanie wykresów Bode'go wzmocnienia i fazy w zakresie $10^0 - 10^4$ rad/s, dla pełnego układu (linia ciągła) oraz dla układu po redukcji do czterech pierwszych form własnych (linia przerywana). Z porównania wykresów widać, że charakterystyka układu po redukcji jest zbliżona do charakterystyki układu przed redukcją, a co za tym idzie możliwe jest zastosowanie jej do naszych potrzeb w zaprezentowanym powyżej zakresie częstości.




Rysunek 2.22.: Porównanie wykresu Bode'go wzmocnienia i fazy dla belki wspornikowej przed redukcją (czarna ciągła linia) i po redukcji modalnej z uwzględnieniem 3 pierwszych form własnych (czerwona przerywana linia)

### Emulator neuronowy belki inteligentnej

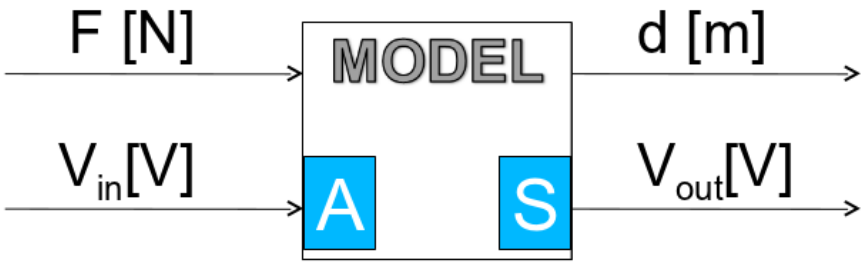
Wzorce uczące zastosowane do nauczenia sieci neuronowej wygenerowano korzystając z przedstawionego powyżej uproszczonego modelu sieci neuronowej przy założonej redukcji do pierw-



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

szej formy własnej oraz tłumieniu modalnym $\zeta = 0.005$. Standardowo, wygenerowane wzorce poddane zostały normalizacji w celu uniknięcia wpływu dużych wartości na proces uczenia sieci, przy jednoczesnym przyspieszeniu zbieżności rozwiązania oraz wzroście dokładności emulacji nauczonej sieci.

Uproszczony schemat wejść i wyjść dla modelu belki inteligentnej pokazany jest na rysunku 2.23. Standardowo belka uczona jest na bazie wartości $V_{in}$ oraz $V_{out}$, będącymi odpowiednio napięciem podawanym na aktuator piezoelektryczny oraz napięciem wyjściowym z czujnika piezoelektrycznego. Zarówno wymuszenie zewnętrzne $F$ jak i przyłożone do aktuatora piezoelektrycznego napięcie $V_{in}$ mają wpływ na wartość przemieszczenie końca belki $d$, a co za tym idzie również na wartość napięcia wejściowego $V_{out}$ na czujniku piezoelektrycznym.




Rysunek 2.23.: Uproszczony schemat wymuszeń (wejść) i odpowiedzi (wyjść) modelu belki inteligentnej z piezoelektrykami

Dla podanego przypadku emulacji neuronowej zdecydowano się na zastosowanie sieci typu NNARX. Parametry wybrane do nauczenia podanej sieci neuronowej zebrano w poniższej tabeli:

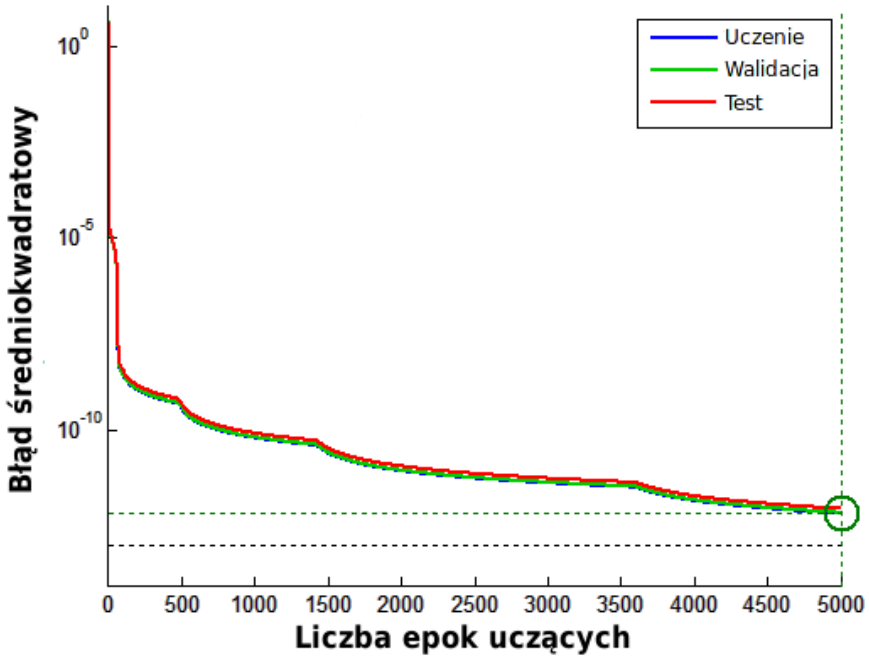
| | |
|---|---|
| częstotliwość próbkowania | $f_s = 1000$ Hz |
| liczba próbek | $N = 2000$ |
| liczba regresorów wymuszenia | $k = 2$ |
| liczba regresorów stanu | $m = 2$ |
| algorytm uczenia sieci | Lavenberga-Margquardta |
| wskaźnik jakości uczenia | mse (mean square error) |
| liczba cykli uczenia | 5000 |
| założona dokładność sieci | 1e-13 |

Krzywa uczenia dla emulatora belki inteligentnej pokazana jest na rysunku 2.24. Na rysunku obok krzywej uczenia („Train") przedstawione są również dwie inne krzywe („Validation" oraz „Test"). Zazwyczaj przy uczeniu sieci neuronowej, wzorce uczące dzielone są na trzy wymienionie podgrupy. Pierwsza z nich, czyli „Train" odpowiada właściwym wzorcom zastosowanym do nauczenia sieci. Druga, czyli „Validation" służy do weryfikacji stopnia przeuczenia sieci, wstrzymując dalszy proces uczenia w przypadku gdy stopień ten zostanie przekroczony. Ostatnia z grup nie bierze udziału w procesie nauki sieci, jej celem jest natomiast sprawdzenie



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

dokładności działania już nauczonej sieci, z wykorzystaniem innych próbek niż zastosowane we wzorcach uczących. W przypadku emulacji belki inteligentnej, dostępne wzorce podzielono w proporcji 6:2:2.

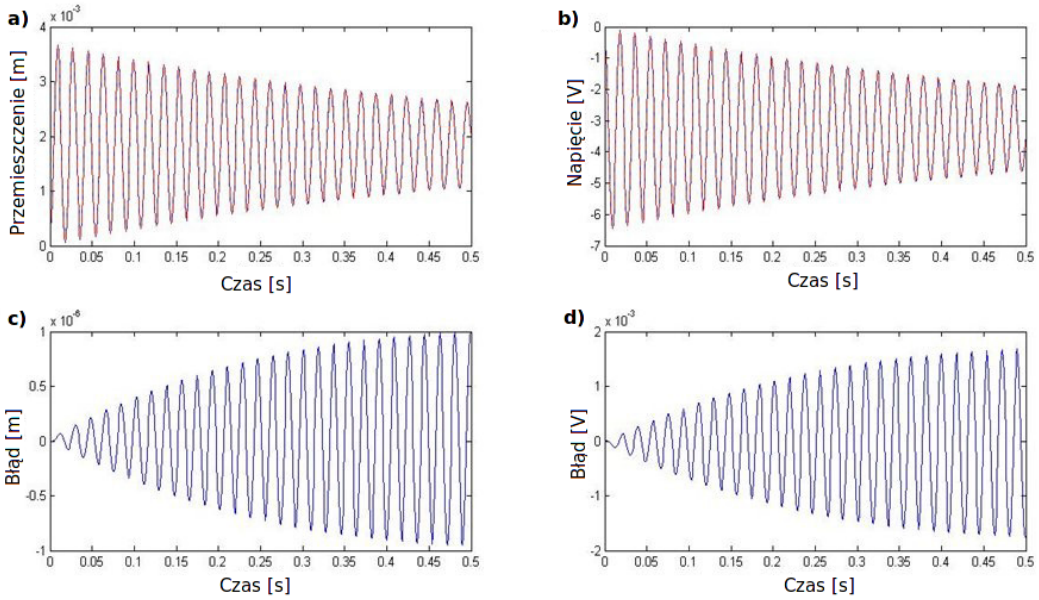



Rysunek 2.24.: Krzywa uczenia emulatora neuronowego belki inteligentnej

Wykresy na rysunku 2.25 obrazują odpowiedź emulatora belki inteligentnej na wymuszenie skokiem siły o wartości 1 N. Jak widać z wykresów błędu, różnica pomiędzy odpowiedzią uzyskaną z modelu belki oraz odpowiedzią z jej emulatora jest znikoma (ok. 0.1%) w stosunku do właściwej odpowiedzi, można zatem na tej podstawie przyjąć założenie, że sieć została poprawnie nauczona dla pierwszej formy własnej przy założonych dla niej uproszczeniach oraz w przyjętym podczas nauki zakresie jej działania. Wyższe formy własne belki dyskutowane będą w dalszej części pracy przy okazji przedstawienia możliwości aktywnego neuronowego tłumienia jej drgań.



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ




Rysunek 2.25.: Odpowiedź skokowa emulatora belki inteligentnej na wymuszenie sygnałem skokowym o wartości 1 N
a) przemieszczenie końca belki
b) napięcie wyjściowe z czujnika piezoelektrycznego
c) błąd pomiędzy przemieszczeniem końca belki wyznaczonym z modelu i z emulatora
d) błąd pomiędzy napięciem wyjściowym z czujnika piezo wyznaczonym z modelu i z emulatora



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

# 3. Sterowanie neuronowe

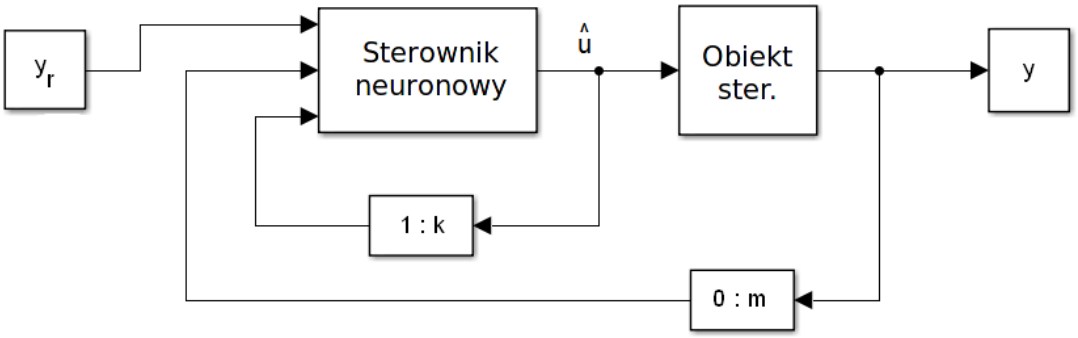
Podobnie jak w przypadku opisywanych w poprzednim rozdziale emulatorów neuronowych, różnorodność oraz zakres zastosowań sieci służących do regulacji oraz sterowania wybranymi procesami bądź obiektami dynamicznymi jest obszerny. W literaturze dostępna jest znaczna ilość prac poświęconych zagadnieniu sterowania neuronowego, poczynając od tych gdzie sieć neuronowa emuluje zachowanie już istniejącego algorytmu sterującego [49], poprzez sieci sterujące typu rekurencyjnego [50], aż po sieci których zadaniem jest optymalizacja sterowania, np. przez znalezienie optymalnych wartości wzmocnienia $K_P$, $K_I$ oraz $K_D$ dla sterownika typu PID [51].

Standardowo, metody sterowania neuronowego dzielą się na dwie podgrupy: tak zwane sterowanie „bezpośrednie" oraz sterowanie „pośrednie". Pierwsza z tych podgrup odpowiada sytuacji, gdy sieć neuronowa uczona jest do roli elementu sterującego, a zatem jest w stanie w bezpośredni sposób wpływać na zachowanie sterowanego układu. Do grupy tej zaliczamy przykładowo sterowanie przez inwersję oraz sterowanie typu NARMA-L2. W drugiej z podgrup, sieć uczona jest jako suplement do istniejącego sterowania, gdzie zadaniem jej jest optymalizacja parametrów algorytmu sterującego. Wynika stąd, że nauczona sieć neuronowa wpływa w sposób pośredni na zachowanie układu. Przykładem tego typu sterowania jest nieliniowe sterowanie predykcyjne, gdzie sieć neuronowa działająca jako emulator sterowanego obiektu, dokonuje predykcji jego zachowania dla kilku przyszłych kroków, na bazie których optymalizowane jest zachowanie właściwego elementu sterującego.

Dodatkowy podział typów sterowania neuronowego dokonywany jest na bazie sposobu w jaki sieć jest uczona. W podziale tym rozróżnia się uczenie typu „off-line" oraz uczenie typu „on-line". W pierwszym z tych przypadków zakłada się, że sieć neuronowa nauczona jest wyłącznie raz na bazie dostarczonych do niej próbek, natomiast w trakcie pracy jej struktura oraz parametry (wartości wag i odchyleń) pozostają niezmienione. W drugim przypadku, sieć douczana jest w sposób ciągły w trakcie pracy, na bazie stale dostarczanych do niej próbek. Pomijając dodatkowo odmienny sposób uczenia, każda z tych metod posiada swoje zalety i wady: o ile w przypadku uczenia „off-line" otrzymujemy kompletnie działający sterownik neuronowy dla podanego układu, o tyle uczenie typu „on-line" umożliwia dostosowanie sterownika dla przypadku zmiany parametrów układu, kosztem jednak wymaganej większej mocy obliczeniowej oraz szybkości przetwarzania danych.



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ




Rysunek 3.1.: Schemat ogólny układu ze sterowaniem neuronowym

Rysunek 3.1 obrazuje uproszczony schemat układu ze sterowaniem neuronowym. W ogólnym przypadku sygnał sterujący $\hat{u}$ wyznaczany jest przez sieć na bazie $k$ poprzednich wartości $u$, $m$ poprzednich stanów wyjściowych obiektu $y$ oraz pożądanej wartości stanu wyjściowego $y_r$, co można zapisać za pomocą ogólnej funkcji postaci:

$$\hat{u}(t) = g\left(y_r(t), y(t-1), \cdots, y(t-m), u(t-1), \cdots, u(t-k)\right) \tag{3.1}$$

Przed przystąpieniem do uczenia regulatora neuronowego, koniecznym krokiem podobnie jak przy emulacji neuronowej jest odpowiedni dobór parametrów zarówno sieci jak i procesu jej nauki. Krok ten jest niezbędny z uwagi na złożoną naturę sieci, których właściwe funkcjonowanie uzależnione jest od szeregu różnych czynników. Spośród czynników, które zostały dokładnie opisane w poprzednim rozdziale rozpatrzyć należy:

- **architekturę sieci** - w szczególności liczbę oraz wymiary poszczególnych warstw sieci (zwłaszcza warstw ukrytych), liczbę regresorów oraz typ funkcji aktywacji dla każdej z warstw
- **dobór wzorców uczących** - które podobnie jak przy emulacji powinny obejmować pełen zakres, w którym sieć ma działać, w tym przypadku zakres jej sterowania
- **dobór algorytmu uczącego** - z uwzględnieniem kryterium optymalizacji; kluczowym jest osiągnięcie odpowiedniego poziomu nauczenia sieci - w przypadku gdy sieć jest niedouczona, sterowanie jest nieefektywne; z kolei dla sieci przeuczonej sterowanie jest super-efektywne, lecz fizycznie nierealizowalne

Ponadto przy projektowaniu sterowania neuronowego, konieczne jest uwzględnienie:

- **stopnia złożoności sterowania** - przez co rozumie się ilość zmiennych sterowanych oraz wymuszeń sterujących, w najprostszym przypadku układ SISO o jednym wejściu i jedynym wyjściu, w bardziej złożonym - MIMO o wielu wejściach i wielu wyjściach; jak



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

to zostanie dalej przedstawione - niektóre z typów sterowania neuronowego typu SISO, rozszerzyć można po odpowiednich modyfikacjach na sterowanie typu MIMO

- **architekturę sieci sterującej** - w zależności od potrzeb oraz rodzaju sterowanego układu, architekturę sieci sterującej można wybrać spośród szeregu dostępnych typów opisanych w literaturze [10], [12], [37]; wybór architektury sieci determinuje stopień jej złożoności, a co za tym idzie zakres możliwych zastosowań

Z uwagi na obszerność zagadnienia sterowania za pomocą sztucznych sieci neuronowych, na potrzeby niniejszej pracy podjęto decyzję o ograniczeniu ilości stosowanych typów regulatorów neuronowych, w zamian koncentrując się na implementacji oraz modyfikacji szeregu z nich na potrzeby aktywnego tłumienia drgań układów oscylacyjnych. Poniżej znajduje się szczegółowy opis typów architektury sieci dla algorytmów sterowania neuronowego wybranych na potrzeby niniejszej pracy.

## 3.1. Architektura sieci wybranych algorytmów sterowania neuronowego

Spośród szeregu dostępnych w literaturze typów sterowania neuronowego, na potrzeby tłumienia drgań układów przedstawionych w niniejszej pracy zdecydowano się za zastosowanie pięciu wybranych algorytmów, kolejno:

- sterowania typu **„inverse”** (inaczej: sterowania odwrotnego)
- sterowania typu **„NARMA-L2”** znanego również pod nazwą Feedback Linearization czyli sterowania z linearyzacją w pętli sprzężenia zwrotnego
- sterowania typu **„feedforward”**, stanowiącego połączenie sterowania neuronowego poprzez inwersję z klasycznym sterowaniem typu PID
- sterowania poprzez emulację istniejącego układu sterującego
- sterowania predykcyjnego

Nawiązując do klasyfikacji przedstawionej we wstępie do niniejszego rozdziału, każdy z wymienionych powyżej algorytmów, za wyjątkiem moteody sterowania predykcyjnego, jest typu „bezpośredniego”, dla każdego zdecydowano się również na uczenie w trybie „off-line”. Bardziej szczegółowy opis wybranych algorytmów znaleźć można w literaturze [10], [11], [13], [52].

### 3.1.1. Sterowanie typu „inverse” (odwrotne)

Sterowanie poprzez inwersję stanowi jedną z pierwszych zaproponowanych metod sterowania neuronowego. W metodzie tej przyjmuje się, że wartość wyjściową $y$ sterowanego układu dla chwili $t+1$ można wyznaczyć z ogólnej zależności:



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

$$y(t+1) = f\left(y(t), \cdots, y(t-m), u(t), \cdots, u(t-n)\right) \tag{3.2}$$

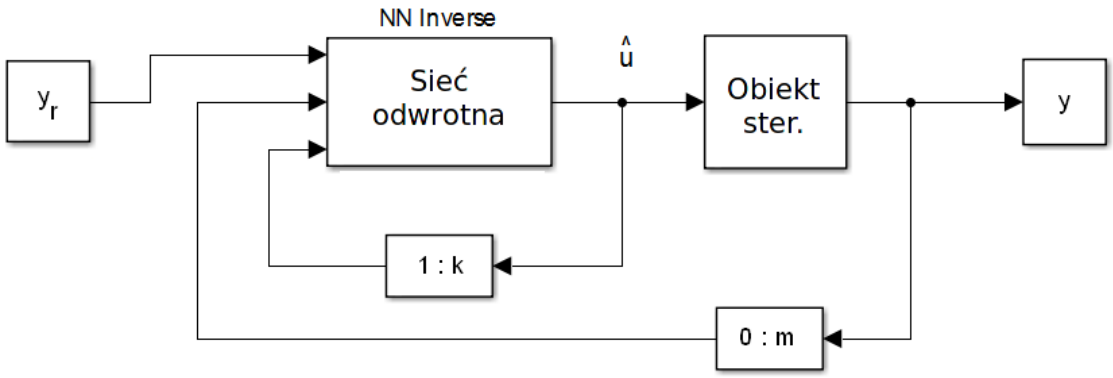
Przyjmując następnie założenie, że funkcja 3.2 jest odwracalna, możliwe jest poprzez odpowiednie jej przekształcenie, wyseparowanie ostatniego sygnału wymuszenia $u$, otrzymując tym samym zależność odwrotną:

$$u(t) = f^{-1}\left(y(t+1), y(t), \cdots, y(t-m), u(t-1), \cdots, u(t-n)\right) \tag{3.3}$$

Podstawiając następnie w miejsce $y(t+1)$ wartość pożądaną $y_r(t+1)$ oraz zakładając, że funkcja $f^{-1}$ reprezentuje sieć neuronową $g$, a zatem wyjście $u$ jest tak naprawdę wartością wyjściową $\hat{u}$ sieci neuronowej, otrzymujemy:

$$\hat{u}(t) = g\left(y_r(t+1), y(t), \cdots, y(t-m), u(t-1), \cdots, u(t-n)\right) \tag{3.4}$$

Wartość wyjściowa $y$ sterowanego obiektu, przy tak założonym sygnale sterującym $\hat{u}$ podawanym przez prawidłowo nauczoną sieć, wykazywać powinna zbieżność do zadanej wartości $y_r$.




Rysunek 3.2.: Schemat blokowy dla układu sterowania typu „inverse”

Rysunek 3.2 przedstawia ogólny schemat blokowy dla sterowania typu „inverse”, z uwzględnieniem regulatora neuronowego, obiektu sterowanego oraz bloków opóźnienia dla sygnałów $u$ i $y$. Proces uczenia kontrolera neuronowego tego typu przebiega identycznie jak proces uczenia emulatora typu NNARX z tą różnicą, że wzorce uczące zamienione są miejscami - wzorce wejściowe stają się wyjściowymi i vice versa. Warto przy tej okazji wspomnieć, że możliwe rozszerzenie zakresu sterowania typu „inverse” na układy MIMO, poprzez zastąpienie wartości sterującej $u$ oraz wyjściowej $y$, odpowiednio wektorami $\mathbf{u}$ oraz $\mathbf{y}$, otrzymując w ten sposób:

$$\hat{\mathbf{u}}(t) = \mathbf{g}\left(\mathbf{y_r}(t+1), \mathbf{y}(t), \cdots, \mathbf{y}(t-m), \mathbf{u}(t-1), \cdots, \mathbf{u}(t-n)\right) \tag{3.5}$$

Chociaż metoda neuronowego sterowania przez inwersję jest prosta pod względem archi-



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

tektury sieci oraz jej implementacji, a uczenie kontrolera neuronowego przebiega stosunkowo szybko, metoda ta posiada również szereg wad. Przede wszystkim, zastosowanie jej ograniczone jest do układów, które wykazują cechę odwracalności, czyli układów dla których wartość wejściową układu wyznaczyć można w sposób jednoznaczny ze znanej wartości wyjściowej oraz jej regresorów. Ponadto nauczony sterownik neuronowy bez narzuconych zewnętrznych ograniczeń, przejawia tendencję do podawania nieosiągalnych wartości sterujących, w szczególności w początkowych chwilach od uruchomienia sterowania. Stąd konieczność narzucenia dodatkowych ograniczeń na sterowanie, przykładowo wprowadzenie bloku saturacji, który ograniczy wartość sterowania do realizowalnego przedziału.

### 3.1.2. Sterowanie typu NARMA-L2

Podstawę metody sterowania neuronowego typu NARMA-L2, stanowi model NARMA, czyli nieliniowy auto-regresyjny model o średniej ruchomej. Bazując na tym modelu, w metodzie tej podobnie jak przy sterowaniu typu „inverse" przyjmuje się, że sygnał sterujący sprowadzić można do postaci nieliniowej funkcji uzależnionej od poprzednich stanów wyjściowych oraz wartości sterujących układu:

$$u(t) = g\left(y(t+1), y(t), \cdots, y(t-m), u(t-1), \cdots, u(t-n)\right) \tag{3.6}$$

W przypadku sterowania typu NARMA-L2, wprowadza się jednak dodatkowe założenie umożliwiające skrócenie czasu potrzebnego na minimalizację błędu średniokwadratowego sieci, a co za tym idzie przyspieszenie procesu jej uczenia. Zakłada się mianowicie, że wartość wyjściową układu $y$ dla chwili $t+1$ aproksymować można za pomocą zlinearyzowanej funkcji zależnej od sygnału sterującego $u(t)$:

$$\tilde{y}(t+1) = f(t) + g(t) \cdot u(t) \tag{3.7}$$

gdzie $f(t)$ oraz $g(t)$ są wartościami wyznaczonymi dla chwili czasowej $t$, zależnymi w sposób nieliniowy od wartości wyjściowych oraz sterujących układu:

$$\begin{aligned} f(t) &= f(y(t), \cdots, y(t-m), u(t-1), \cdots, u(t-k)) \\ g(t) &= f(y(t), \cdots, y(t-m), u(t-1), \cdots, u(t-k)) \end{aligned} \tag{3.8}$$

Przekształcając następnie równanie 3.7 w celu wyznaczenia wartości $u(t)$:

$$u(t) = \frac{\tilde{y}(t+1) - f(t)}{g(t)} \tag{3.9}$$

a następnie podstawiając w miejsce $\tilde{y}(t+1)$ wartość $y_r(t+1)$ odpowiadającą pożądanej trajektorii układu, otrzymujemy sygnał sterujący sieci $\hat{u}(t)$ w formie:

$$\hat{u}(t) = \frac{y_r(t+1) - f(t)}{g(t)} \tag{3.10}$$



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

Zwrócić przy tym należy uwagę na fakt, iż do wyznaczenia wartości sterującej z równania 3.10, konieczna jest znajomość obu funkcji $f(t)$ oraz $g(t)$, które na obecnym etapie metody, znane są wyłącznie w ogólnej postaci 3.8. Pamiętając jednak o możliwości zastosowania algorytmów neuronowych, równanie 3.7 sprowadzić można do postaci emulatora neuronowego:

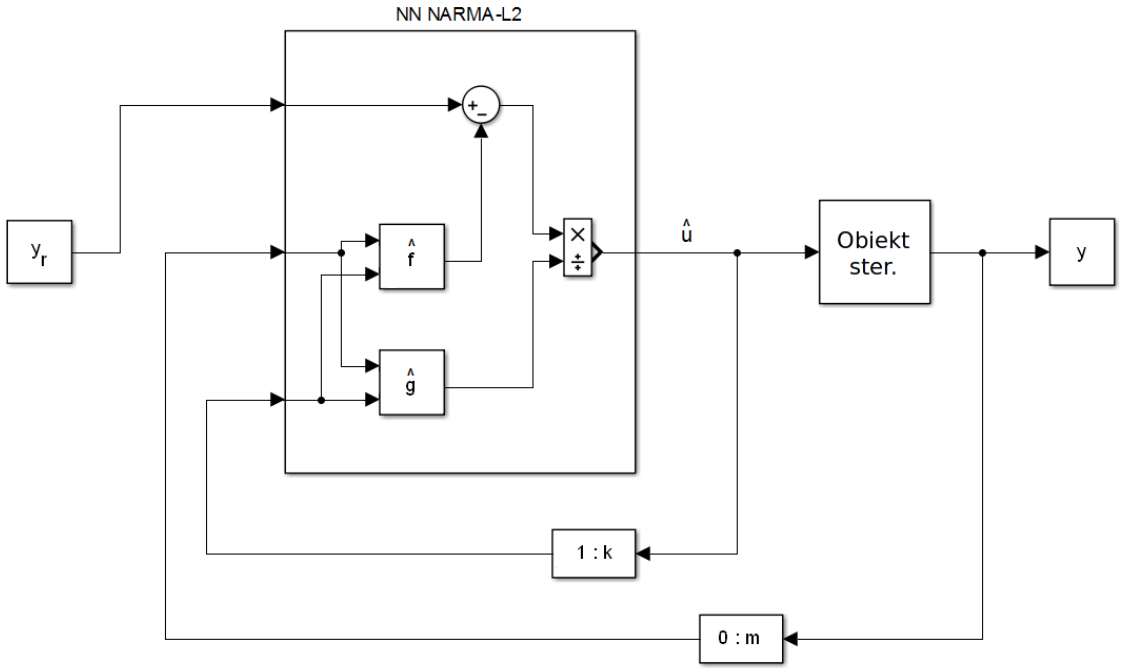
$$\hat{y}(t+1) = \hat{f}(t) + \hat{g}(t) \cdot u(t) \tag{3.11}$$

który następnie uczony jest korzystając z wzorców uczących wygenerowanych z układu będącego obiektem sterowania. Funkcje $f(t)$ oraz $g(t)$ po nauczeniu tego typu emulatora neuronowego, reprezentowane są zatem przez dwie odrębne i niezależne od siebie sieci neuronowe $\hat{f}(t)$ oraz $\hat{g}(t)$.

Tak nauczone sieci zaimplementować można następnie w sterowaniu neuronowym, w wyniku czego otrzymuje się ostateczną postać sygnału sterującego $\hat{u}$:

$$\hat{u}(t) = \frac{y_r(t+1) - \hat{f}(t)}{\hat{g}(t)} \tag{3.12}$$

Rysunek 3.3 obrazuje schemat blokowy dla tak zdefiniowanego sterowania neuronowego typu NARMA-L2.




Rysunek 3.3.: Schemat blokowy dla układu sterowania typu NARMA-L2

Z uwagi na liniową postać równania 3.7, sterowanie typu NARMA-L2 występuje również w literaturze pod nazwą „Feedback Linearization" [10], czyli sterowania z linearyzacją w pętli sprzężenia zwrotnego.

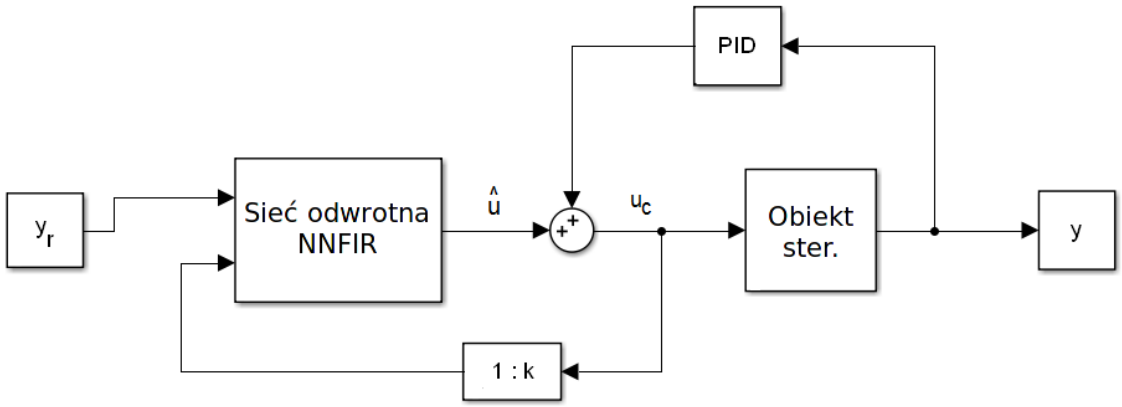


BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

### 3.1.3. Sterowanie typu FF

W klasycznej teorii sterowania, sterowanie typu „feedforward" zachodzi w przypadku gdy sygnał sterujący podawany przez sterownik na wejście układu, nie jest zależny od jego stanu wyjściowego (włącznie z możliwym błędem sterowania), generowany jest natomiast na podstawie dobrej znajomości dynamiki układu, a co za tym idzie umiejętności predykcji jego zachowania.

Dla przypadku sieci neuronowych, sterowanie typu „feedforward" realizowane przez sieć, powiązane jest standardowo ze sterowaniem typu „feedback" (w pętli sprzężenia zwrotnego), realizowanym przez klasyczny układ PID, o odpowiednio dobranych parametrach wzmocnienia $k_P$, $k_I$, oraz $k_D$. Głównym zadaniem dołączonego układu PID, jest korekcja (minimalizacja) uchybu na wyjściu układu powstałego w wyniku sterowania za pomocą sieci neuronowej. Najczęściej w neuronowym sterowaniu typu „feedforward" zastosowanie znajduje sieć typu NNFIR (opisana szczegółowo w podrozdziale 2.2.1), której nauka oraz sterowanie odbywa się na identycznej zasadzie jak w opisanej wyżej metodzie typu „inverse". Schemat blokowy układu sterowania, dla tak zdefiniowanej sieci FFFIR, przedstawiono na rysunku 3.4.

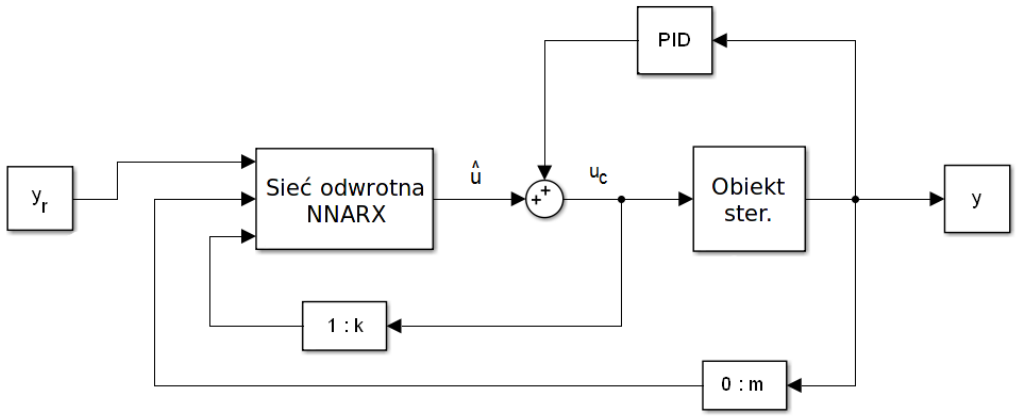



Rysunek 3.4.: Schemat sterowania typu FFFIR

Alternatywą dla sterowania FFFIR, rozszerzającą układ o dodatkowy element sprzężenia zwrotnego, jest zastosowanie w miejsce sieci typu NNFIR - sieci NNARX, otrzymując tym samym dokładniejsze sterowanie FFARX. Schemat blokowy dla tego typu sterowania pokazano na rysunku 3.5.



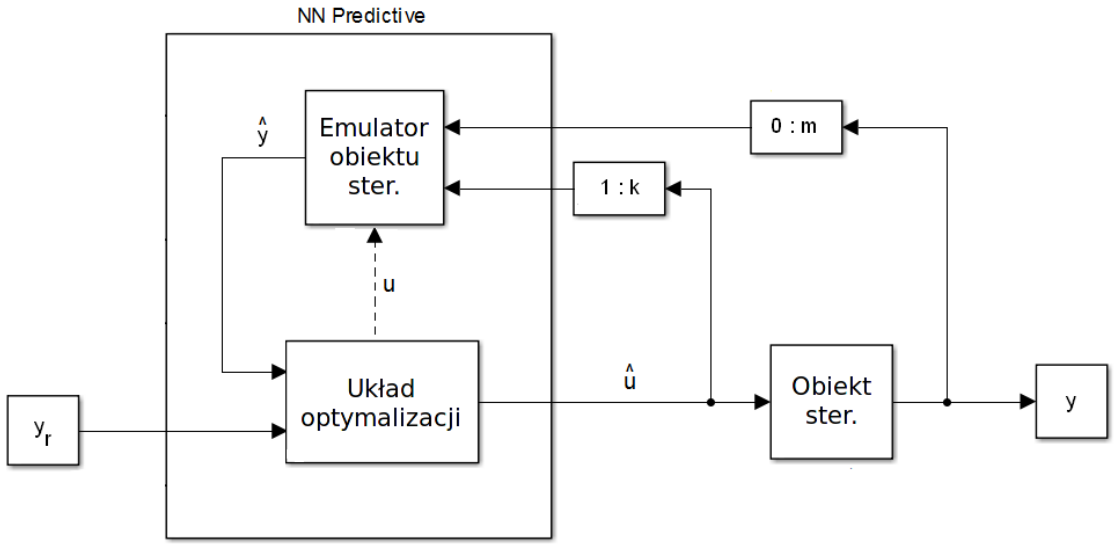
BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ




Rysunek 3.5.: Schemat sterowania typu FFARX

### 3.1.4. Sterowanie predykcyjne

W przeciwieństwie do opisanych wyżej algorytmów sterowania neuronowego, metoda predykcyjna zaliczana jest do grupy pośrednich algorytmów sterowania neuronowego. Oznacza to, że sieć neuronowa nie jest wykorzystywana w sposób bezpośredni do wyznaczenia właściwego sygnału sterującego, służy ona natomiast do optymalizacji zachowania właściwego bloku sterującego. Na rysunku 3.6 przedstawiono schemat blokowy dla sterowania predykcyjnego z wyraźnie wyszczególnionym blokiem emulatora neuronowego oraz blokiem optymalizacji.




Rysunek 3.6.: Schemat sterowania typu predykcyjnego



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

Optymalizacja sterowania odbywa się poprzez minimalizację wskaźnika jakości $J$ zdefiniowanego wzorem:

$$J = \sum_{j=N_1}^{N_2} (y_r(t+j) - \hat{y}(t+j))^2 + \rho \sum_{j=1}^{N_u} (u(t+j-1) - u(t+j-2))^2 \tag{3.13}$$

gdzie $y_r$, $\hat{y}$ oraz $u$ są odpowiednio pożądaną wartością wyjściową rzeczywistego układu, wartością wyjściową z emulatora neuronowego oraz zaproponowaną wartością sygnału sterującego. Minimalizacja wskaźnika jakości $J$ dla przypadku sterowania predykcyjnego przeprowadzana jest na bazie wartości wyjściowych $\hat{y}$ przewidywanych przez emulator neuronowy w zakresie „horyzontu błędu" $[N_1, N_2]$ oraz przewidywanych wartości sygnału sterującego ograniczonych tak zwanym „horyzontem sterowania" $N_u$. Parametr $\rho$ uwzględniony we wzorze 3.13, pozwala na dodatkową modyfikację sterowania poprzez zmianę kontrybucji części odpowiadającej wartościom sygnału sterującego.

Dokładny opis metody neuronowego sterowania predykcyjnego znaleźć można w pracy [10] oraz w dokumentacji [53].

### 3.1.5. Emulacja istniejącego układu sterowania

Ostatnia z przedstawionych w pracy metod sterowania neuronowego, polega na wykorzystaniu już istniejącego układu sterowania (PID, LQR, etc.), a następnie na bazie wzorców pobranych podczas sterowania, nauczenie sieci wiernej emulacji oryginalnego układu. Najczęściej do tego celu w przypadku sieci dynamicznych wykorzystuje się architekturę typu NNARX lub NNARMAX (opis w podrozdzałach 2.2.2, 2.2.3). Oczywistą wadą przedstawionego rozwiązania jest konieczność posiadania już działającego sterownika dla sterowanego układu. Dodatkowo zakres działania sieci neuronowej ograniczony jest zakresem działania oryginalnego układu sterującego. Z drugiej strony jak można się przekonać testując nauczony sterownik/emulator neuronowy, jedną z cech jaką on wykazuje jest zdolność do prawidłowego (choć z gorszą dokładnością) działania w przypadku gdy parametry sterowanego układu ulegną zmianie.

## 3.2. Przykłady sterowania neuronowego

Poniżej przedstawiona została numeryczna implementacja neuronowego sterowania, obejmująca wszystkie z opisanych szczegółowo powyższych metod. Sieci neuronowe oraz algorytmy ich uczenia zaprogramowane zostały w środowisku Matlab, natomiast symulacja właściwego ich działania przeprowadzona została w graficznym module Simulink. Ponadto, nauczone sieci neuronowe, rozszerzone zostały o szereg dodatkowych bloków oraz układów, mających na celu usprawnienie ich funkcjonowania. Dokładny ich opis wraz z wprowadzonymi przez nie zmianami, przedstawiony jest przy okazji opisu sterowania poszczególnych modeli.

Specyfikę oraz skuteczność neuronowego sterowania, w szczególności w zastosowaniu do aktywnego tłumienia drgań, zdecydowano się przedstawić na przykładzie wybranych modeli



BPK BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

dynamicznych układów oscylacyjnych. Modele, na które się zdecydowano w niniejszej pracy to kolejno:

- nieliniowe wahadło matematyczne
- belka inteligentna z elementami piezoelektrycznymi
- nieliniowe wahadło podwójne
- giroskop z zawieszeniem Cardana o dwóch osiach obrotu

Ponadto, opisane metody sterowania neuronowego przetestowane zostały również dla innych, nie zawartych w pracy, układów dynamicznych, między innymi dla nieliniowego układu testowego (z nieliniowością w postaci trzeciej potęgi przemieszczenia w równaniu różniczkowym dynamiki) oraz dla często spotykanego w zagadnieniach teorii sterowania, układu wahadła odwróconego.

### 3.2.1. Wahadło matematyczne

Pierwszym i jednocześnie najprostszym z układów testowanych pod kątem skuteczności sterowania neuronowego, jest dynamiczny model wahadła matematycznego. Układ ten opisany już został dokładnie w podrozdziale 2.4.1, przy okazji wprowadzenia do zagadnienia emulacji neuronowej. Model ten wybrany został zarówno ze względu na prostotę jak i nieliniową dynamikę, przejawiającą się w zmiennej częstości drgań w zależności od warunków początkowych - prędkości oraz położenia. Z uwagi na wspomniane zalety modelu wahadła matematycznego, posłuży on jako podstawowy przykład, na którym przetestowane będą opisane wyżej metody sterowania neuronowego.

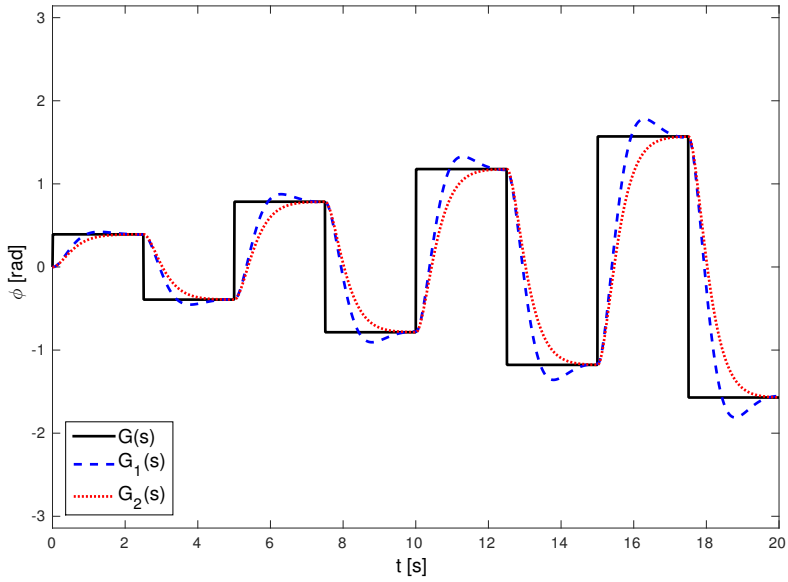
W niniejszych przykładach, w celu przetestowania zachowania nauczonego sterownika neuronowego, jako funkcję docelową dla położenia kątowego wahadła, zastosowano funkcję prostokątną o rosnącej wartości (określaną w pracy mianem „Rising Square"), zdefiniowaną za pomocą zależności:

$$x(t) = \left\lceil \frac{t}{5} \right\rceil (-1)^{\lfloor t/2.5 \rfloor} \cdot \pi/8 \tag{3.14}$$

gdzie symbole $\lceil\ \rceil$ oraz $\lfloor\ \rfloor$ odpowiadają odpowiednio funkcji „ceiling" (zaokrąglenie w górę) oraz „floor" (zaokrąglenie w dół). Przy tak zdefiniowanych parametrach, zmiana wartości docelowej funkcji (włącznie z jej znakiem) następuje co 2.5 s, natomiast sama jej wartość bezwzględna wzrasta w każdym kroku o $\pi/8$. Wykres powyżej założonej funkcji docelowej kąta pokazany jest na rysunku 3.7:



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ




Rysunek 3.7.: Wykres pożądanego położenia kątowego wahadła w postaci funkcji prostokątnej o rosnącej wartości („Rising Square") dla przypadków:
- bez filtru analogowego $G(s) = 1$
- z filtrem analogowym II-go rzędu $G_1(s) = 1/(0.1s^2 + 0.4s + 1)$
- z filtrem analogowym II-go rzędu $G_2(s) = 1/(0.1s^2 + 0.6s + 1)$

Funkcja ta jest o tyle użyteczna, że dzięki niej możliwym jest sprawdzenie odpowiedzi sterownika na zmianę skokową wartości docelowej kąta, pozwalając jednocześnie na sprawdzenie działania sterownika neuronowego w pełnym nieliniowym zakresie wartości. Przykładowo, dla czasu sterowania równego 20 s (stosowanego w poniższych przykładach), zmiana położenia kątowego wahadła zawiera się w przedziale $[-\pi/2, \pi/2]$. Wydłużając ten czas do 40 s, położenie kąta $\phi$ obejmować będzie pełen możliwy zakres wartości.

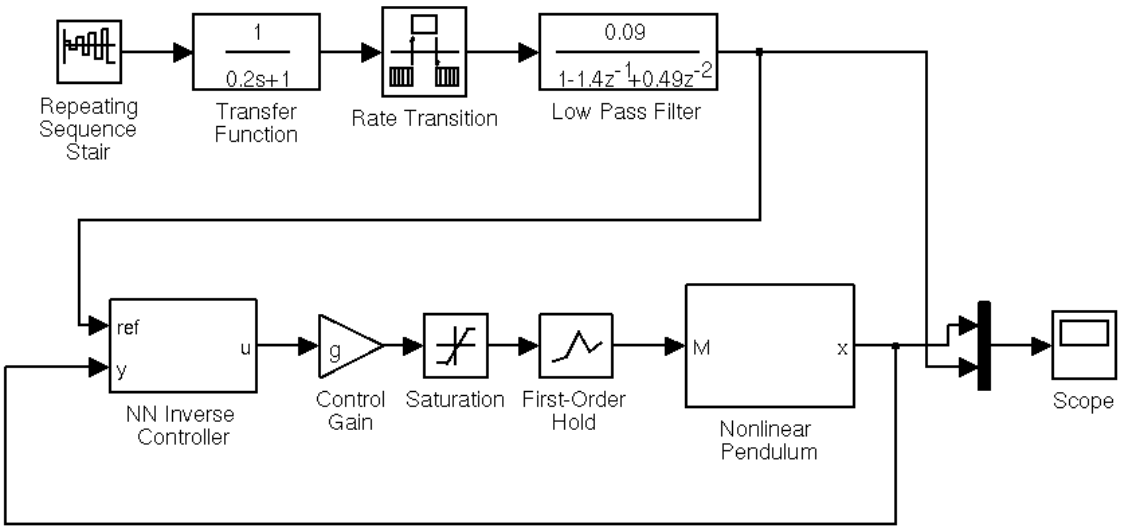
### Sterowanie typu „inverse"

Na rysunku 3.8 przedstawiono przykładowy schemat blokowy sterowania neuronowego typu „inverse" dla układu wahadła matematycznego. Sieć neuronowa dla podanego układu nauczona została przy następujących parametrach:

- parametry wahadła: $m = 0.1$ kg, $l = 0.2$ m
- architektura typu **NNARX**, 2-warstwowa, 15-1
- regresory: $m = 2$, $k = 2$
- zakres wymuszenia/sygnału sterującego $M = [-1, 1]$ Nm
- zakres położenia kątowego $\phi = [-\pi, \pi]$ rad



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ




Rysunek 3.8.: Schemat blokowy sterowania neuronowego typu "inverse" dla wahadła matematycznego, rozszerzony o dodatkowe bloki korygujące (środowisko Matlab/Simulink)

Powyższy schemat, poza elementami odpowiadającymi sterownikowi neuronowemu, modelowi wahadła matematycznego oraz funkcji zadanego położenia docelowego kąta $\phi$ („repeating sequence stair") , zawiera szereg dodatkowych bloków korygujących mających na celu usprawnienie działania neuronowego układu sterowania. Bloki te, opisane szczegółowo w dalszej części pracy przy okazji analizy ich wpływu na skuteczność neuronowego sterowania wahadłem matematycznym, zawierają kolejno:

- **filtr analogowy** - na wyjściu funkcji definiującej położenie docelowe kąta, mający na celu wygładzenie kształtu docelowej krzywej, a tym samym jak zostanie dalej pokazane, usprawnienie sterowania
- **dolnoprzepustowy filtr cyfrowy** - wykorzystywany w tym przypadku do spowolnienia dojścia do celu, poprzez eliminację nagłych skoków wartości kąta, przy jednoczesnym ograniczeniu skokowych zmian momentu sterującego $M$
- **wzmocnienie** $g$ - na wyjściu układu sterującego
- **saturacja** - wprowadzająca ograniczenie na moment sterujący, narzucone przez przyjęty podczas nauki zakres wzorców uczących

Analiza skuteczności działania sterowania neuronowego typu „inverse" rozpocznięta zostanie od domyślnie zdefiniowanego układu, który poza blokiem saturacji, nie zawiera żadnego z wymienionych powyżej elementów. Na rysunku 3.9 pokazane są wykresy sygnału sterującego $M$ oraz kąta $\phi$ wahadła, przy prostokątnej rosnącej funkcji docelowej opisanej podanym

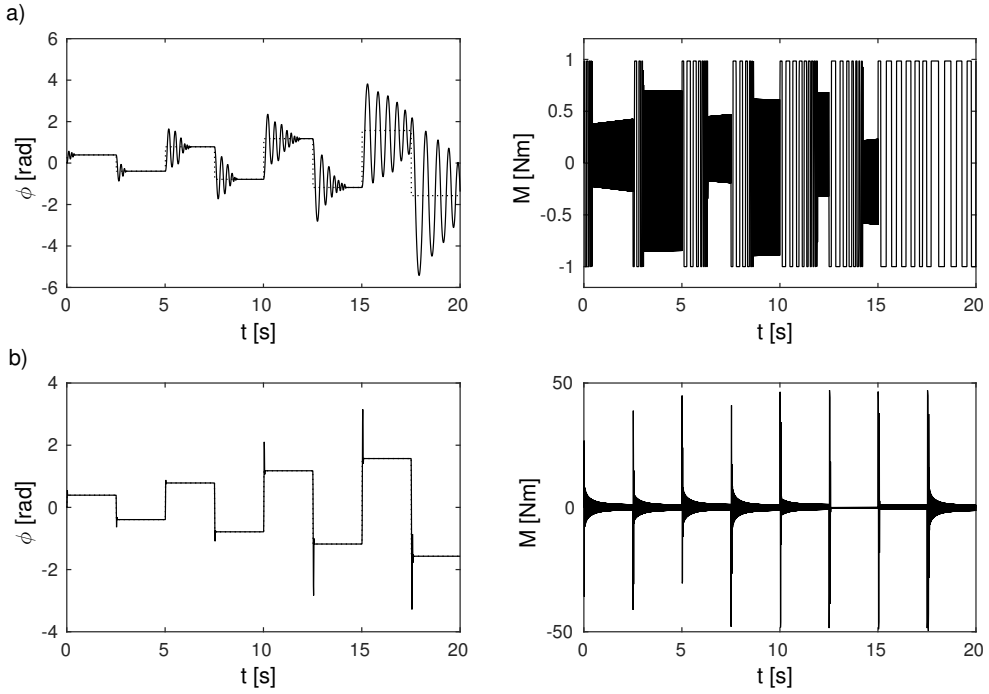


BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

wcześniej równaniem 3.14, dla dwóch przypadków: bez nałożonej saturacji na moment sterujący $M$ oraz z saturacją w zakresie $M = [-1, 1]$ Nm, zgodnie z przyjętym zakresem wzorców uczących.

Analizując przedstawione sterowanie neuronowe wysnuć można szereg następujących wniosków. Po pierwsze, dla każdego z obu przypadków zdefiniowane przez funkcję prostokątną rosnącą sterowanie jest osiągnięte, co więcej również dla zakresu nieliniowego. Jednakże, dla przypadku z saturacją nałożoną na wartość sygnału sterującego, zauważyć można wyraźne przesterowanie w połączeniu z wolno zanikającymi oscylacjami wokół położenia docelowego. Jak widać na wykresie, sygnał sterujący dla tego przypadku wykazuje silnie zmienną naturę, nieustannie przeskakując pomiędzy narzuconą odgórnie wartością minimalną i maksymalną sterowania.

Efekt ten przypisywać można sterownikowi neuronowemu, który pomimo poprawnego nauczenia, napotykając skokową zmianę pożądanej wartości położenia kątowego, domyślnie proponuje wartość sterującą $M$ umożliwiającą dodarcie do położenia docelowego w minimalnie krótkim czasie, znacznie jednak wybiegającą ponad nałożone podczas nauki sieci ograniczenia. Co gorsze, wartość ta wprowadzana jest następnie do sieci w postaci regresora, determinując tym samym wartość sygnałów sterujących dla kolejnych chwil czasowych, a co za tym idzie sterowanie jak pokazano na rysunku 3.9, a nieosiągalne w praktycznej realizacji.




Rysunek 3.9.: Moment sterujący $M$ oraz położenie kątowe $\phi$ wahadła matematycznego dla sterowania neuronowego typu „inverse"
a) sterowanie z nałożoną saturacją na wartość momentu $M = [-1, 1]$ Nm
b) sterowanie bez saturacji wartości momentu sterującego $M$
Ustawienia sterowania: domyślne (bez dodatkowych bloków korygujących)



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

Z drugiej strony, w przypadku gdy sygnał sterujący $M$ nie jest ograniczany w żaden sposób, jak na rysunku 3.9 b, wahadło wykazuje zdolność do osiągania zadanego położenia kątowego w znacznie krótszym czasie (choć nadal z wyraźnie obecnym przesterowaniem). Dzieje się to jednak kosztem wzrostu wartości momentu sterującego poza zakres, w którym sieć była uczona. Dodatkowo wartość sygnału sterującego nadal wykazuje niepożądaną i fizycznie niezrealizowaną tendencję do nagłej, skokowej zmiany. Podobny problem zaobserwować można również dla przypadku sterowania neuronowego typu „inverse" dla innego prostego układu nieliniowego opisanego w pracy [10], nie jest tam jednak zaproponowane żadne rozwiązanie umożliwiające korekcję sygnału sterującego. Przy tej okazji należy zwrócić uwagę na fakt, iż sterowanie jest w stanie wyjść (nawet daleko) poza ramy, w których została nauczona sieć. Własność ta jest typowa dla neuronowego sterowania typu „inverse".

Jednym z zaproponowanych rozwiązań umożliwiających usprawnienie sterowania neuronowego, niezależnie od zastosowanego jego typu, jest bądź to modyfikacja architektury uczonej sieci (przez co rozumie się zwiększenie liczby warstw ukrytych, ilości neuronów w poszczególnych warstwach lub wyprowadzenie sterowania na bazie większej ilości regresorów), bądź ponowne nauczenie sieci przy zmienionej częstotliwości próbkowania wzorców uczących. Jednakże, jak wykazały testy przeprowadzone dla zmodyfikowanej struktury sterownika neuronowego, wpływ modyfikacji wprowadzonych do architektury sieci jest nieznaczny i pomijalny.

Co do zmiany częstotliwości próbkowania, pamiętać należy o konieczności znalezienia optymalnej jej wartości dla rozpatrywanego układu sterowanego. W przypadku gdy wartość ta jest zbyt niska pojawia się ryzyko pominięcia obecnych w dynamice sterowanego obiektu wysokich częstotliwości co wiąże się z możliwym pogorszeniem skuteczności sterowania. Z kolei gdy częstotliwość próbkowania jest zbyt wysoka, dynamika sterowanego układu jest lepiej odzwierciedlona, kosztem jednak konieczności szybszego przetwarzania informacji przez sterownik neuronowy, a co za tym również idzie - szybszą jego dynamiką (jak pokazano na powyższym przykładzie - rysunek 3.9).

Alternatywnym rozwiązaniem przedstawionego problemu sterowania neuronowego jest zastosowanie odpowiednich bloków korygujących w układzie sterowania. Co jednak warto zaznaczyć, bloki te w żaden sposób nie ingerują w architekturę oraz działanie nauczonej sieci - sterownika neuronowego. Poniżej, na wybranych przykładach przedstawiono w jaki sposób zmiana parametrów w poszczególnych z wymienionych elementów wpływa na właściwości sterowania badanym układem wahadła. Ponieważ jednak właściwości te uzależnione są od wielu współzależnych od siebie czynników, dla każdego z przedstawionych poniżej przykładów przyjmuje się konkretne parametry działania pozostałych elementów (zdefiniowane jako „ustawienia sterowania" w opisie pod adekwatnymi wykresami)

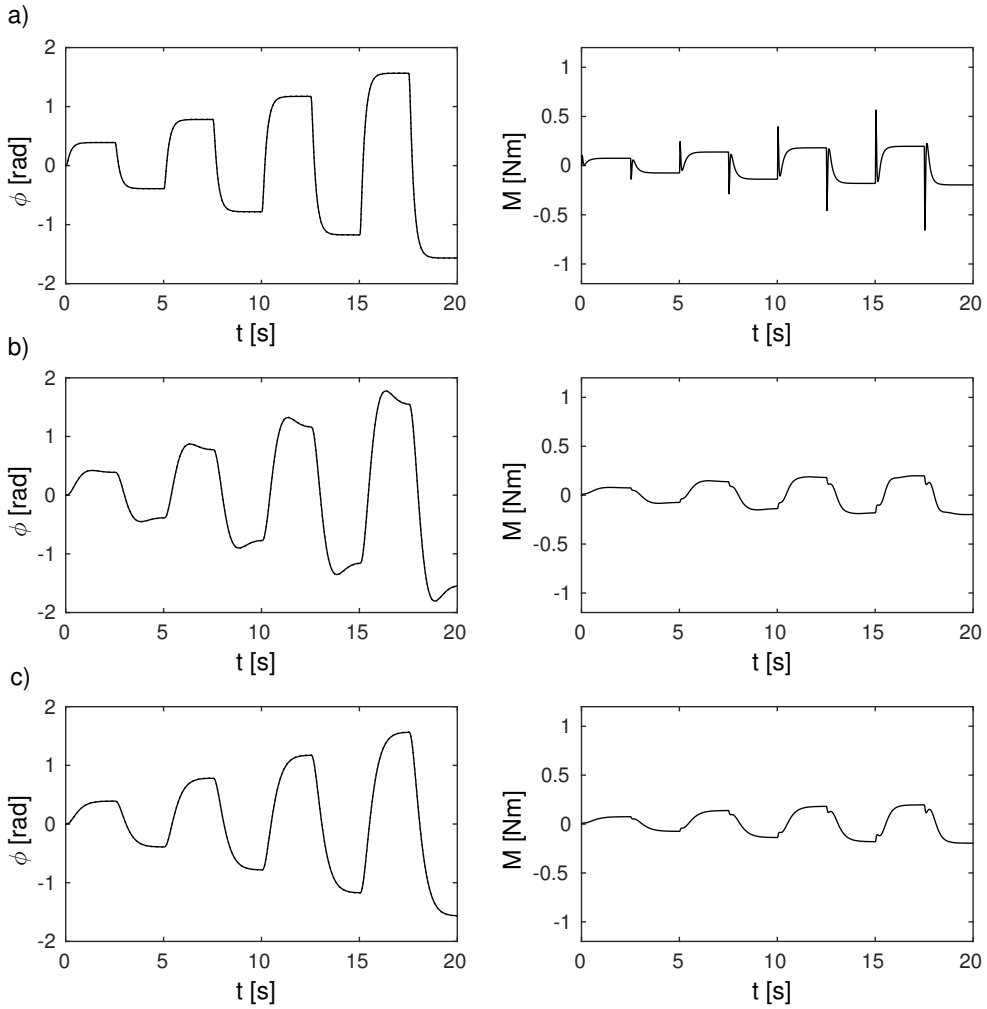
- **Filtr analogowy I-go, II-go rzędu**

  Ponieważ, jak to przedstawiono powyżej, tendencja do skokowej zmiany wartości sterowania przez sterownik neuronowy wynika ze skokowo zdefiniowanej funkcji docelowego położenia kątowego, jednym z możliwych rozwiązań jest wygładzenie tej funkcji tak, by



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

przejście między kolejnymi położeniami kątowymi następowało w sposób płynny. Jest to możliwe do osiągnięcia poprzez wprowadzenie do układu dodatkowego bloku z filtrem analogowym o odpowiednio dobranych parametrach, zaraz za blokiem ze zdefiniowaną funkcją położenia docelowego (rysunek 3.8). Kontroler w takim wypadku otrzymuje dodatkową informację nie tylko o tym jakie położenie powinien osiągnąć, ale również po jakiej krzywej powinien do niego dotrzeć.




Rysunek 3.10.: Moment sterujący $M$ oraz położenie kątowe $\phi$ wahadła matematycznego dla sterowania neuronowego typu „inverse"
a) filtr analogowy I-go rzędu $G(s) = 1/(0.2s + 1)$
b) filtr analogowy II-go rzędu $G(s) = 1/(0.1s^2 + 0.4s + 1)$
c) filtr analogowy II-go rzędu $G(s) = 1/(0.1s^2 + 0.6s + 1)$
Ustawienia sterowania: wzmocnienie $g = 0.3$, saturacja $M = [-1, 1]\,\mathrm{Nm}$

Na rysunku 3.10 pokazano w jaki sposób dobór odpowiedniego filtra wpływa na specyfikę sterowania neuronowego - z wyraźnie widoczną poprawą sygnału sterującego, wynikającą

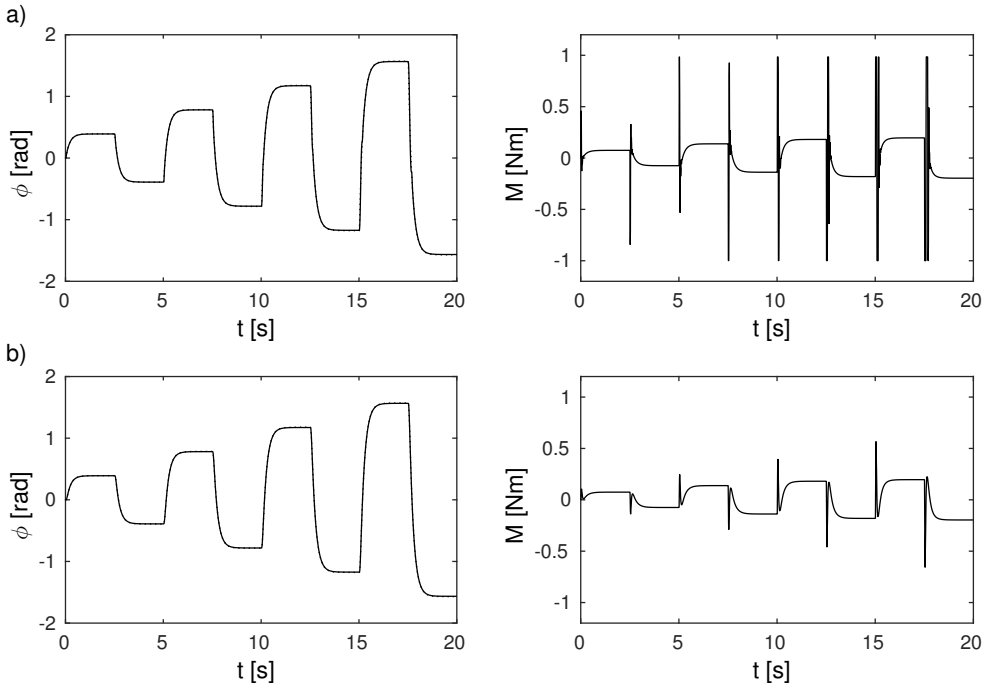


BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

z wprowadzenia ciągłego (w przeciwieństwie do skokowego) charakteru zmiany położenia kątowego.

- **Dolnoprzepustowy filtr cyfrowy**

  Na rysunku 3.11 pokazano wykresy położenia kątowego wahadła $\phi$ oraz momentu sterującego $M$ dla przypadku z uwzględnieniem dolnoprzepustowego filtru cyfrowego w układzie sterowania (b) oraz dla przypadku gdzie jest on pominięty (a). Zastosowany filtr o transmitancji w płaszczyźnie $z$ zdefiniowanej za pomocą zależności $H(z) = 0.09/(1 - 1.4z^{-1} + 0.49z^{-2})$, odpowiada podwójnemu biegunowi $((0.7, 0)$ leżącemu wewnątrz okręgu jednostkowego tejże płaszczyzny.




Rysunek 3.11.: Moment sterujący $M$ oraz położenie kątowe $\phi$ wahadła matematycznego dla sterowania neuronowego typu „inverse"
a) bez dolnoprzepustowego filtru cyfrowego
b) z zastosowaniem dolnoprzepustowego filtru cyfrowego
$H(z) = 0.09/(1 - 1.4z^{-1} + 0.49z^{-2})$
Ustawienia sterowania: wzmocnienie $g = 0.3$, saturacja $M = [-1, 1]\,\mathrm{Nm}$, filtr analogowy I-go rzędu $G(s) = 1/(0.2s + 1)$

Z rysunku 3.11 widać, że wprowadzenie opisanej wyżej filtracji do układu, umożliwia poprzez eliminację wysokoczęstotliwościowych zmian założonego położenia $\phi$, ograniczenie nagłych zmian sygnału sterującego $M$, bez widocznego pogorszenia działania układu sterowania. Pamiętać jednak należy o tym, że z uwagi na odcięcie wysokich częstotliwości przez filtr, nie będzie możliwe jego zastosowanie dla przypadków gdzie wymagana jest



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

nagła zmiana wartości położenia kątowego - w takim wypadku dokładność sterowania ulegnie wyraźnemu pogorszeniu.

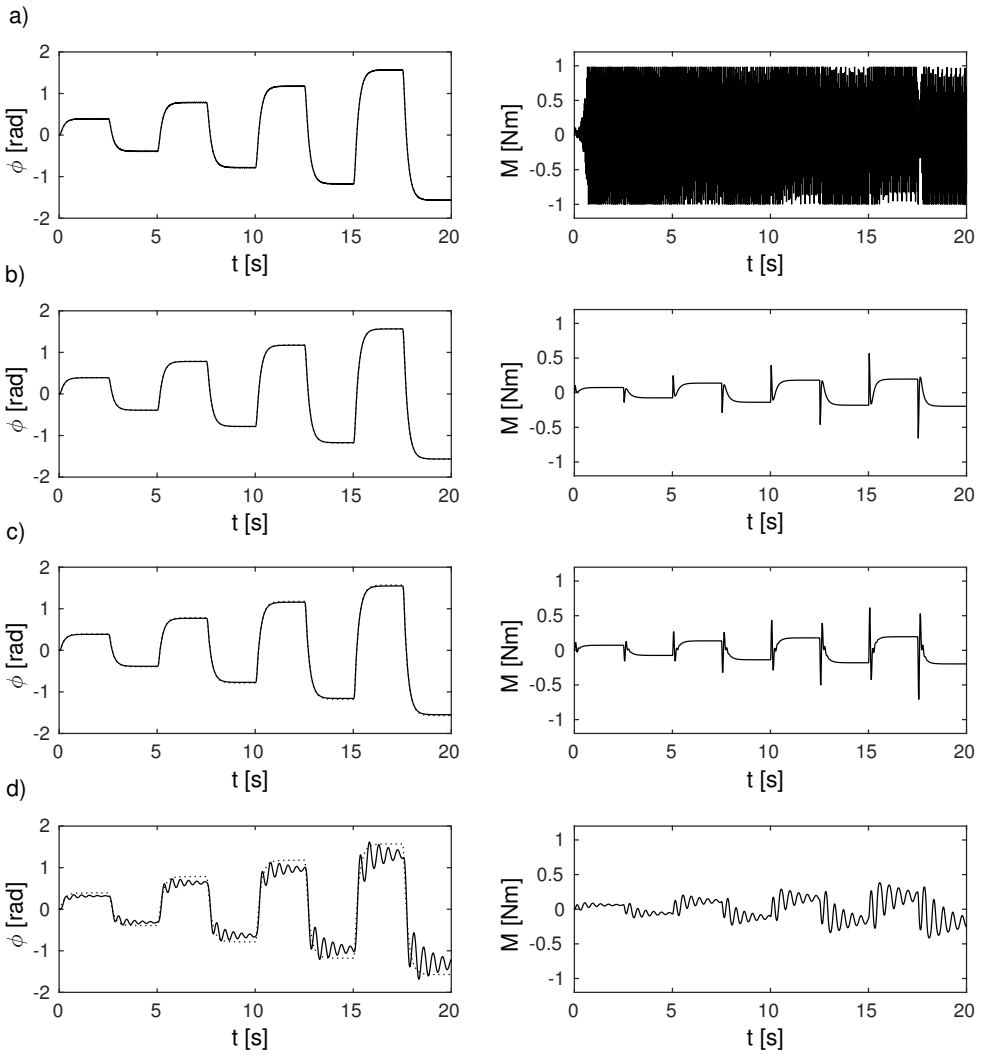
- **Wzmocnienie** g

  Poprzez zmianę parametru wzmocnienia $g$ sygnału sterującego $M$ podawanego na wyjściu ze sterownika neuronowego, możliwa jest poprawa właściwego sterowania bez ingerencji w architekturę nauczonej sieci. Właściwym jednak w tym przypadku słowem powinna być „atenuacja" a nie wzmocnienie, jako że zadaniem parametru $g$ jest ograniczenie wartości sygnału sterującego, którego zmiana dla przypadku domyślnego jest zbyt gwałtowna, jak to zostało uprzednio zademonstrowane na rysunku 3.9. Odpowiedź układu sterowania przy zastosowaniu szeregu różnych wartości $g$, poczynając od 0.5 (największa) po 0.01 (najmniejsza), pokazana została na rysunku 3.12. Wyraźnie widać, że zastosowanie poprawnie dobranej wartości atenuacji znacząco wpływa na poprawę działania neuronowego układu sterowania. W skrajnych przypadkach: jeżeli wartość ta jest zbyt wysoka ($g = 0.5$), powraca problem skokowej zmiany wartości sterującej. Jeżeli z kolei wartość ta jest zbyt niska ($g = 0.01$), w odpowiedzi układu pojawiają się wyraźnie oscylacje, pojawia się w niej również niepożądany uchyb ustalony (steady-state error).

  Wykresy przedstawione na rysunku 3.13 pokazują sterowanie położeniem wahadła matematycznego $\phi$, dla szeregu różnych wartości bądź funkcji docelowych, przy zastosowaniu sterownika neuronowego typu „inverse" z właściwie dobranymi parametrami poszczególnych elementów układu. Jak widać z wykresów, sterowanie odbywa się w sposób prawidłowy (również w zakresie nieliniowym) z wartością sygnału sterującego $M$ nie wybiegającą poza przedział wartości zdefiniowany podczas procesu uczenia sieci.



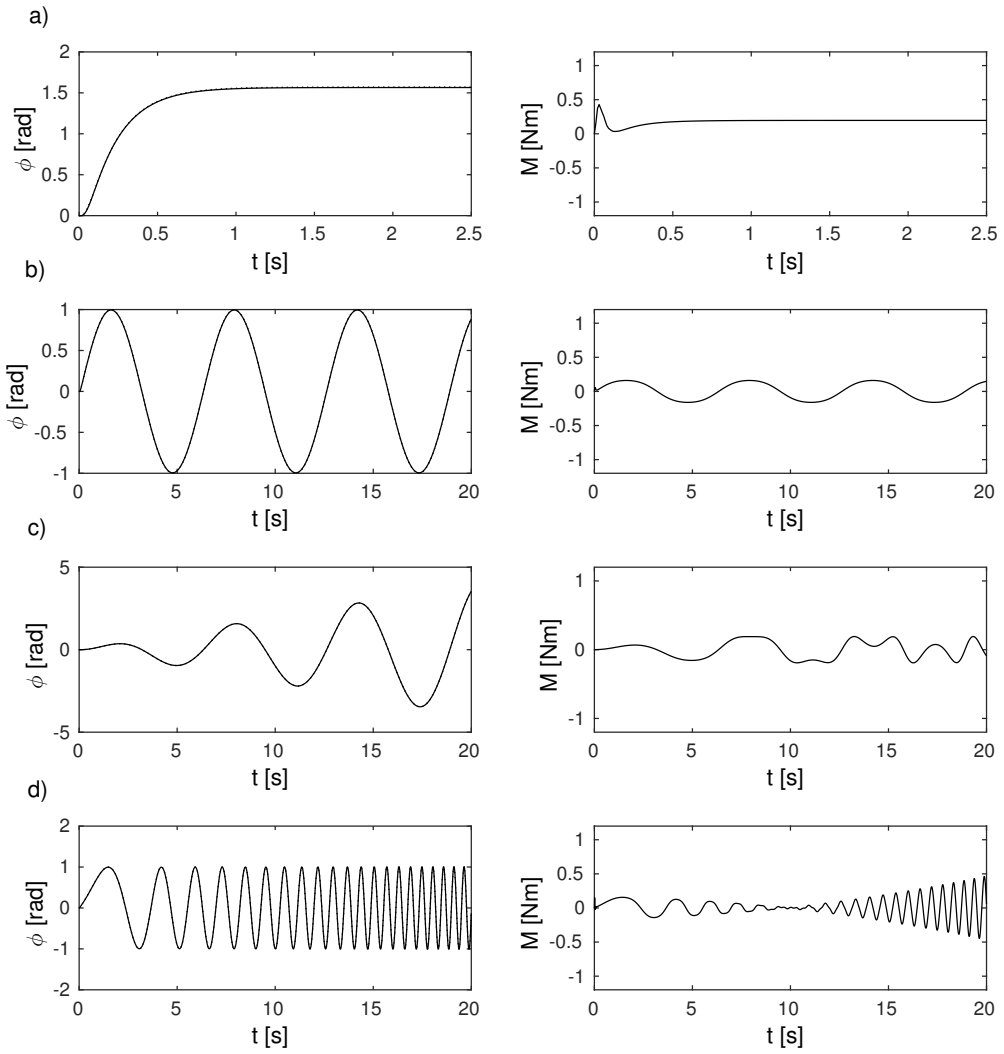
BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ




Rysunek 3.12.: Moment sterujący $M$ oraz położenie kątowe $\phi$ wahadła matematycznego dla sterowania neuronowego typu „inverse"
a) wzmocnienie $g = 0.5$
b) wzmocnienie $g = 0.3$
c) wzmocnienie $g = 0.1$
d) wzmocnienie $g = 0.01$
Ustawienia sterowania: saturacja $M = [-1, 1]\,\mathrm{Nm}$, filtr analogowy I-go rzędu $G(s) = 1/(0.2s + 1)$



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ



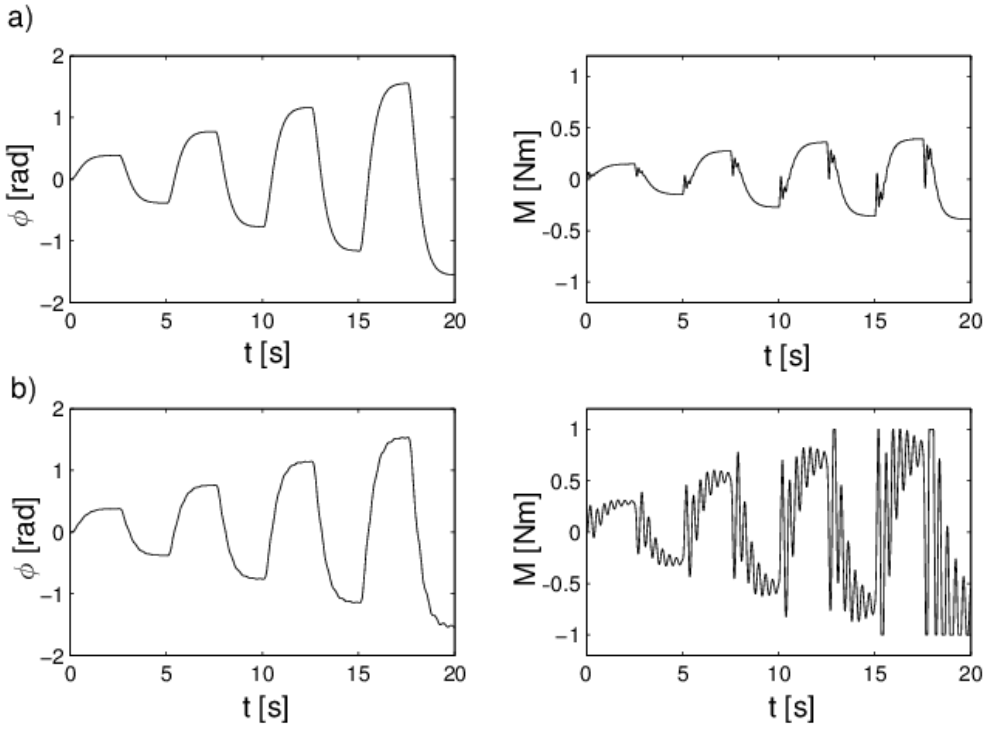

Rysunek 3.13.: Moment sterujący $M$ oraz położenie kątowe $\phi$ wahadła matematycznego dla sterowania neuronowego typu „inverse"; Docelowa wartość/krzywa położenia kątowego wahadła:
a) wartość stała ($A = \pi/2$ rad)
b) sinusoida ($A = 1$ rad, $f = 1$ Hz)
c) sinusoida o zmiennej amplitudzie ($A = 1$ rad, $f = 1$ Hz, d = 0.2 rad/s)
d) sinusoida o zmiennej częstotliwości ($A = 1$ rad, $f = 0.1 - 2$ Hz w 20 s)
Ustawienia sterowania: wzmocnienie $g = 0.3$, saturacja $M = [-1, 1]$ Nm



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

### Sterowanie typu „inverse" - wrażliwość na zmiany parametrów wahadła

Dodatkowym testem jaki można przeprowadzić dla sterowania neuronowego, w tym przypadku sterowania typu „inverse" jest zbadanie w jakim stopniu sieć neuronowa nauczona pod kątem sterowania układem o odgórnie zdefiniowanych parametrach jest w stanie reagować na zmianę wspomnianych parametrów. Wykresy na rysunku 3.14 pokazują w jaki sposób uprzednio nauczona sieć neuronowa działa w przypadku wahadła o innej długości, niż ta przy której sieć została domyślnie nauczona ($l = 0.2$ m). Jak widać z wykresów sieć jest nadal w stanie wysterować układ do pożądanego położenia kątowego $\phi$. Wiąże się to jednak ze wzrostem wartości wymaganego momentu sterującego $M$, wynikającym ze wzrostu długości $l$ wahadła matematycznego, a co za tym idzie równoczesnym wzrostem jego momentu bezwładności.



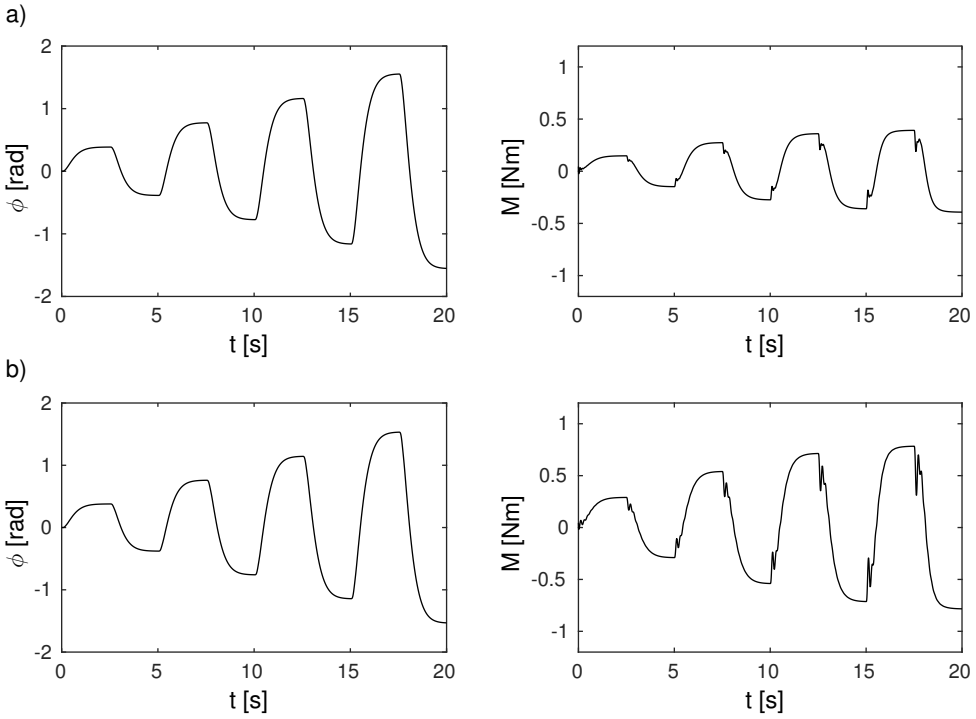

Rysunek 3.14.: Moment sterujący $M$ oraz położenie kątowe $\phi$ wahadła matematycznego dla sterowania neuronowego typu „inverse" przy zmienionej długości wahadła (domyślnie $l = 0.2$ m):
a) długość $l = 0.4$ m, masa $m = 0.1$ kg
b) długość $l = 0.8$ m, masa $m = 0.1$ kg
Ustawienia sterowania: wzmocnienie $g = 0.3$, saturacja $M = [-1, 1]$ Nm, filtr analogowy II-go rzędu $G(s) = 1/(0.1s^2 + 0.6s + 1)$

Dodatkowe oscylacje wartości sygnału sterującego $M$ na wyjściu sieci (3.14 b) przy wydłużeniu wahadła matematycznego, wynikają najprawdopodobniej z powiązanego z nim wzrostu częstości własnej drgań wahadła. Jako, że tak nauczona sieć neuronowa przystosowana jest do sterowania w niższym zakresie częstości, w przypadku gdy zakres ten się rozszerza sieć stara się dostosować do zaistniałej zmiany.



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

Nauczony sterownik neuronowy jest w stanie sterować układem również w przypadku, gdy zmianie ulegnie masa $m$ wahadła matematycznego, jak widać na rysunku 3.15. Ponieważ podobnie jak dla przypadku wydłużonego wahadła, przy wzroście masy wahadła wzrasta również jego moment bezwładności. Logicznym jest zatem, że i moment sterujący $M$ konieczny do jego wysterowania musi być większy.



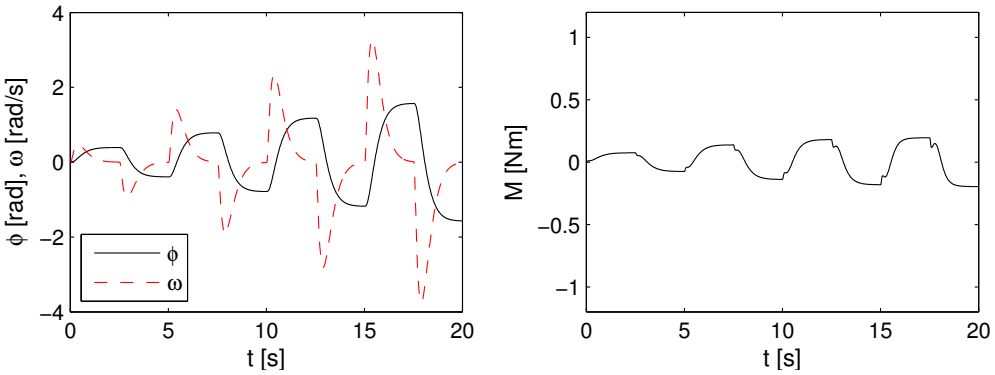

Rysunek 3.15.: Moment sterujący $M$ oraz położenie kątowe $\phi$ wahadła matematycznego dla sterowania neuronowego typu „inverse" przy zmienionej masie wahadła (domyślnie $m = 0.1$ kg):
a) długość $l = 0.2$ m, masa $m = 0.2$ kg
b) długość $l = 0.2$ m, masa $m = 0.4$ kg
Ustawienia sterowania: wzmocnienie $g = 0.3$, saturacja $M = [-1, 1]$ Nm, filtr analogowy II-go rzędu $G(s) = 1/(0.1s^2 + 0.6s + 1)$

### Sterowanie typu „inverse" - MISO

Dla rozpatrywanego modelu wahadła matematycznego możliwym jest również rozszerzenie działania sterownika neuronowego typu „inverse" na sterowanie typu MISO (multi-input single-output). W przeciwieństwie do zaprezentowanego uprzednio sterowania typu SISO, gdzie wartość sygnału sterującego wyznaczana była przez sieć neuronową wyłącznie na bazie wartości położenia kątowego $\phi$ wahadła (oraz jej regresorów), w przedstawionym sterowaniu typu MISO sieć uwzględnia dodatkowo wartość prędkości kątowej $\omega$ wahadła. Na rysunku 3.16 przedstawiono przykładowe sterowanie tego typu z wyszczególnionymi wykresami położenia kątowego $\phi$, prędkości kątowej $\omega$ oraz momentu sterującego $M$ wahadła matematycznego.



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ




Rysunek 3.16.: Położenie $\phi$ i prędkość kątowa $\omega$ wahadła matematycznego oraz moment sterujący $M$ dla sterowania neuronowego typu „inverse" z uwzględnieniem pełnego wektora stanu wahadła;
Parametry sterowania: wzmocnienie $g = 1$, saturacja $M = [-1, 1]\,\mathrm{Nm}$, filtr analogowy II-go rzędu $G(s) = 1/(0.1s^2 + 0.6s + 1)$

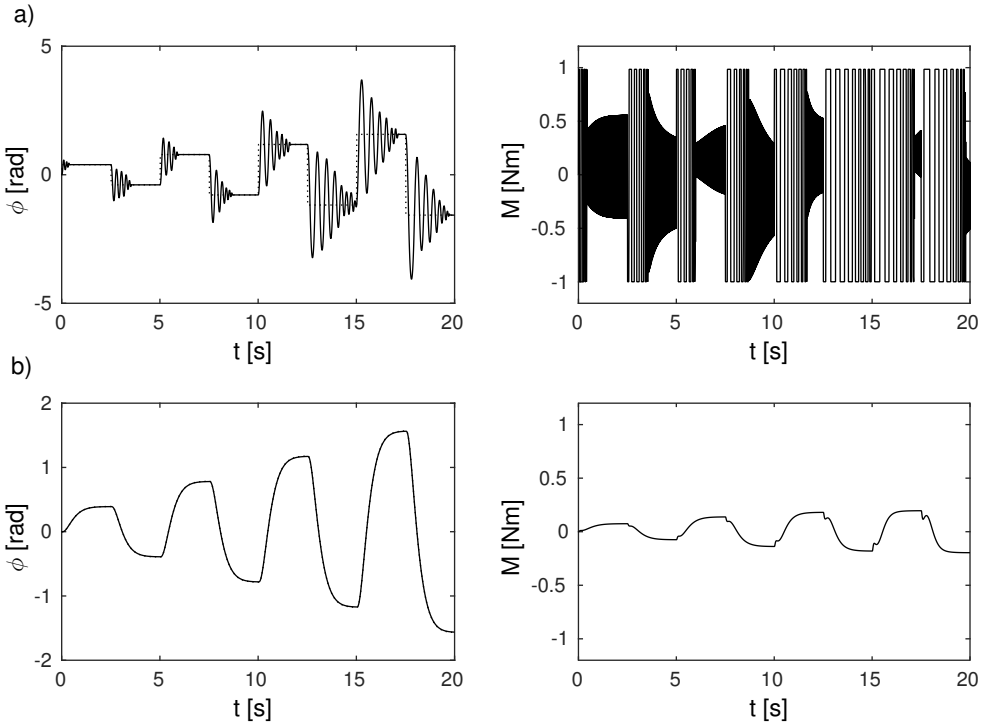
O ile w przypadku sterowania typu SISO sieć neuronowa uzyskiwała w sposób pośredni informację na temat prędkości kątowej $\omega$ wahadła (poprzez wartości regresorów położenia kątowego $\phi$), o tyle wprowadzenie tej informacji w sposób bezpośredni do sieci neuronowej umożliwia usprawnienie działania sieci, a przez to częściową poprawę skuteczności właściwego sterowania. Porównując wykresy na rysunku 3.16 z wykresami na rysunku 3.10 widać, że wartość momentu sterującego $M$ jest porównywalna dla obu typów sterowań przy identycznie zdefiniowanych parametrach sterowania. Zaznaczyć należy, że dla przypadku SISO, wartość ta uwzględnia dodatkowo zmodyfikowany parametr wzmocnienia $g = 0.3$. Dla testowanego przypadku MISO modyfikacja ta nie jest konieczna, jako że sieć jest w stanie bardziej dokładnie wyznaczyć wymaganą wartość sygnału sterującego $M$, korzystając z podanej na nią dodatkowej informacji wejściowej. Uwzględnienie wspomnianej informacji na wejściu sieci wiąże się dodatkowo z koniecznością zwiększenia jej rozmiarów, a co za tym idzie wydłużenia czasu jej uczenia oraz późniejszych obliczeń.

### Sterowanie typu „NARMA-L2"

W przypadku sterowania neuronowego typu NARMA-L2, zdecydowano się na skorzystanie z sieci o architekturze (15-1)-(15-1)-1, stanowiącej reprezentację dwóch funkcji nieliniowych $f$ oraz $g$, z każdą z nich zamodelowaną poprzez 2-warstwową sieć neuronową typu NNARX o 15 neuronach w warstwie ukrytej oraz 1 neuronem w warstwie wyjściowej. Ilość regresorów stanu wyjściowego $y$ oraz wymuszenia $u$ przyjęta została tak samo jak dla sterowania typu „inverse". Podobnie jak i w przypadku tamtego sterowania, sieć neuronowa nie była w stanie w sposób bezpośredni poprawnie sterować położeniem kątowym $\phi$ rozpatrywanego wahadła.



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ




Rysunek 3.17.: Moment sterujący $M$ oraz położenie kątowe $\phi$ wahadła matematycznego dla sterowania neuronowego typu „NARMA-L2"
a) sterowanie domyślne
b) sterowanie z zastosowaniem bloków korygujących

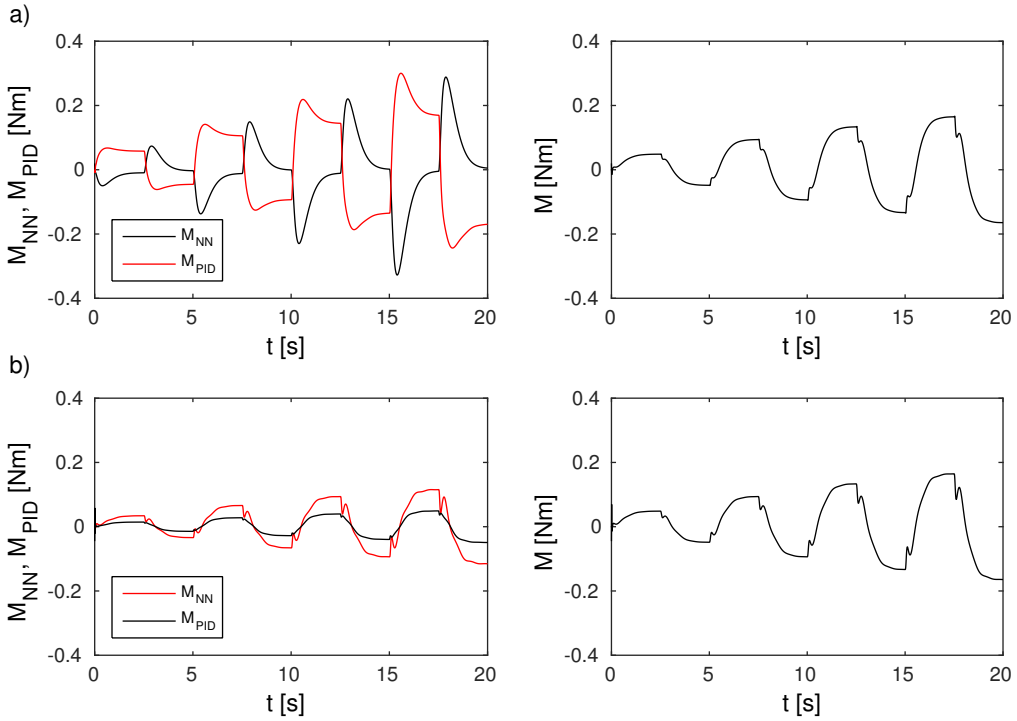
Jak widać na rysunku 3.17 a, przy sterowaniu występuje zarówno przesterowanie wartości $\phi$ jak i skokowa zmiana momentu wymuszenia $M$. Modyfikując zachowanie układu sterowania w analogiczny jak poprzednio sposób poprzez zastosowanie bloków korygujących o optymalnie dobranych parametrach udało się znacznie usprawnić działanie sieci neuronowej (rys. 3.17). Podobnie jak poprzednio, przetestowano również nauczoną sieć pod kątem zmiany położenia kątowego $\phi$ podług zadanej odgórnie krzywej. Uzyskane w ten sposób wykresy pokrywają się z tymi które uzyskano przy sterowaniu typu „inverse" (rysunek 3.13).

### Sterowanie typu „Feedforward"

Rysunek 3.18 przedstawia wykresy momentu sterującego $M$ (z wyszczególnieniem składowych momentu sieci $M_{NN}$ i momentu PID $M_{PID}$) dla obu przypadków sterowania neuronowego typu „Feedforward" (NNFIR, NNARX) opisanych wcześniej w rozdziale 2.4.



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ




Rysunek 3.18.: Wykresy poszczególnych momentów sterujących ($M_{PID}, M_{NN}$) oraz momentu całkowitego $M$ dla sterowania neuronowego typu „feedforward";
a) sterownik neuronowy typu NNFIR
b) sterownik neuronowy typu NNARX
Ustawienia sterowania: wzmocnienie $g = 0.3$, saturacja $M = [-1, 1]\,\mathrm{Nm}$, filtr analogowy II-go rzędu $G(s) = 1/(0.1s^2 + 0.6s + 1)$
Parametry PID: $K_P = 1$, $K_I = 20$, $K_D = 0.25$

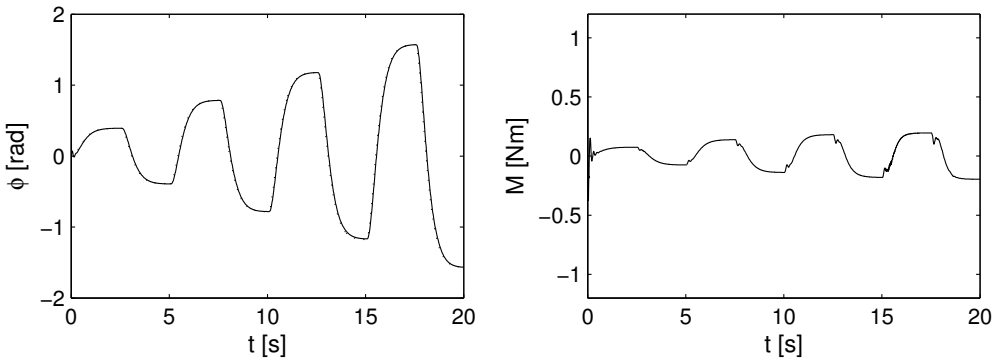
Wykresy położenia kątowego $\phi$ pokrywają się z wykresem przedstawionym na rysunku 3.10 przy zachowanych identycznych parametrach sterowania. Warto zauważyć, że właściwie nauczona sieć neuronowa typu NNFIR, w powiązaniu z działającym układem PID jest w stanie stosunkowo dobrze sterować położeniem kątowym $\phi$ analizowanego wahadła, nawet w zakresie nieliniowym.

### Sterowanie predykcyjne

Na potrzeby neuronowego sterowania predykcyjnego położeniem kątowym $\phi$ wahadła matematycznego skorzystano z bloku dostępnego w środowisku MATLAB/Simulink wraz z wbudowanymi funkcjami optymalizacji, przygotowanego pod kątem sterowania typu SISO. Wykresy na rysunku 3.19 obrazują położenie kątowe $\phi$ oraz moment sterujący $M$ dla analizowanego układu przy zastosowaniu parametrów sterowania identycznych jak dla przypadku sterowania typu „inverse" MISO (Rys. 3.16). Na rysunku 3.20 pokazano dodatkowo wykresy sterowania typu predykcyjnego dla szeregu różnych funkcji docelowego położenia kątowego $\phi$ wahadła.



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ



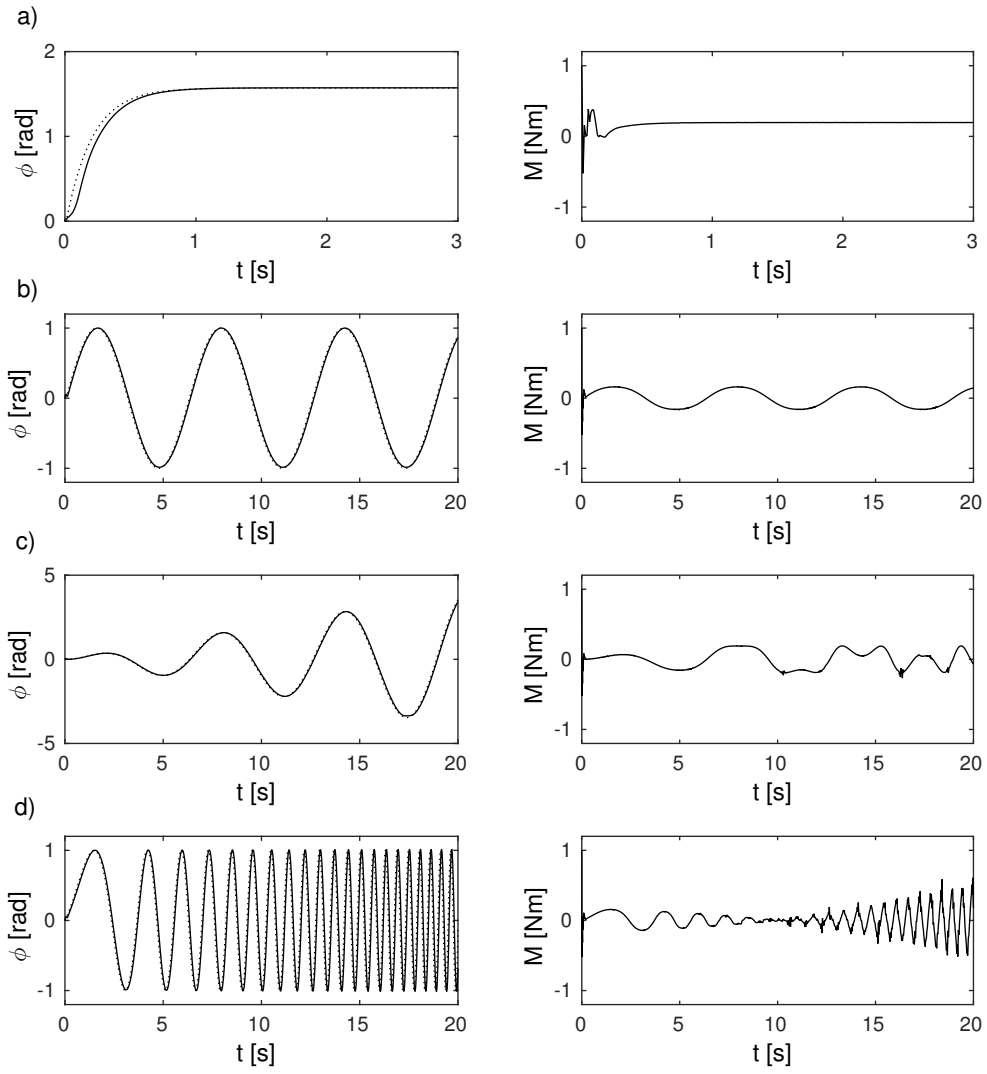

Rysunek 3.19.: Położenie kątowe $\phi$ wahadła matematycznego oraz moment sterujący $M$ dla przypadku sterowania neuronowego z wykorzystaniem metody predykcyjnej; Parametry sterowania: wzmocnienie $g = 1$, saturacja $M = [-1, 1]\,\mathrm{Nm}$, filtr analogowy II-go rzędu $G(s) = 1/(0.1s^2 + 0.6s + 1)$

Podstawową zaletą sterowania neuronowego typu predykcyjnego jest optymalne jego działanie wynikające z konieczności minimalizacji wskaźnika jakości $J$ zależnego od predykowanych przez sieć wartości stanu sterowanego układu oraz proponowanych przez sterownik wartości sygnału sterującego. W przeciwieństwie do sterowania neuronowego typu „inverse" oraz „NARMA-L2", unika się również problemu ze skokową zmianą wartości sygnału sterującego - wartość ta dla sterowania predykcyjnego nie wybiega poza zakres narzucony podczas procesu uczenia sieci neuronowej.

Niestety odbywa się to kosztem mocy obliczeniowej wymaganej dla sterowania typu predykcyjnego, w szczególności przy założonych dużych wartościach zarówno horyzontu błędu jak i horyzontu sterowania. W takim przypadku czas obliczeniowy wymagany do dokonania optymalizacji dla każdego kroku sterowania, ogranicza użyteczność wspomnianej metody sterowania do układów wolnozmiennych. Ponadto rozszerzenie metody neuronowego sterowania predykcyjnego na układy MIMO (multi-input multi-output) stanowi stosunkowo złożone zagadnienie.



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ




Rysunek 3.20.: Położenie kątowe $\phi$ wahadła matematycznego oraz moment sterujący $M$ dla przypadku sterowania neuronowego z wykorzystaniem metody predykcyjnej; Docelowa wartość/krzywa położenia kątowego wahadła:
a) wartość stała ($A = \pi/2$ rad)
b) sinusoida ($A = 1$ rad, $f = 1$ Hz)
c) sinusoida o zmiennej amplitudzie ($A = 1$ rad, $f = 1$ Hz, d = 0.2 rad/s)
d) sinusoida o zmiennej częstotliwości ($A = 1$ rad, $f = 0.1 - 2$ Hz w 20 s)
Ustawienia sterowania: wzmocnienie $g = 1$, saturacja $M = [-1, 1]$ Nm, filtr analogowy II-go rzędu $G(s) = 1/(0.1s^2 + 0.6s + 1)$



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

### 3.2.2. Belka typu „smart"

Kolejnym z układów poddanych analizie pod kątem zastosowania metod sterowania neuronowego oraz ich skuteczności w tłumieniu jego drgań, jest układ belki inteligentnej z elementami piezoelektrycznymi przestawiony szczegółowo w rozdziale 2.4.3. W tym przypadku, celem sterowania jest redukcja drgań swobodnego końca belki w możliwie jak najkrótszym czasie. Model belki inteligentnej zastosowany na potrzeby symulacji sterowania neuronowego w niniejszej pracy jest modelem liniowym. [54].

Wzorce uczące dla sieci neuronowej wygenerowane zostały na podstawie zdyskretyzowanego modelu belki inteligentnej dyskutowanego w podrozdziale 2.4.3. Przy generacji wzorców, przyjęto ograniczenie wartości napięcia sterującego do zakresu $[-250\text{V}, 250\text{V}]$. Ograniczenie to uwarunkowane jest własnościami materiału piezoelektrycznego zaimplementowanego w modelu belki. W przypadku symulacji w układzie sterowania zawarto ograniczający blok saturacji, podobnie jak dla przypadku sterowania położeniem kątowym wahadła matematycznego.

#### Sterowanie typu „inverse" / NARMA-L2

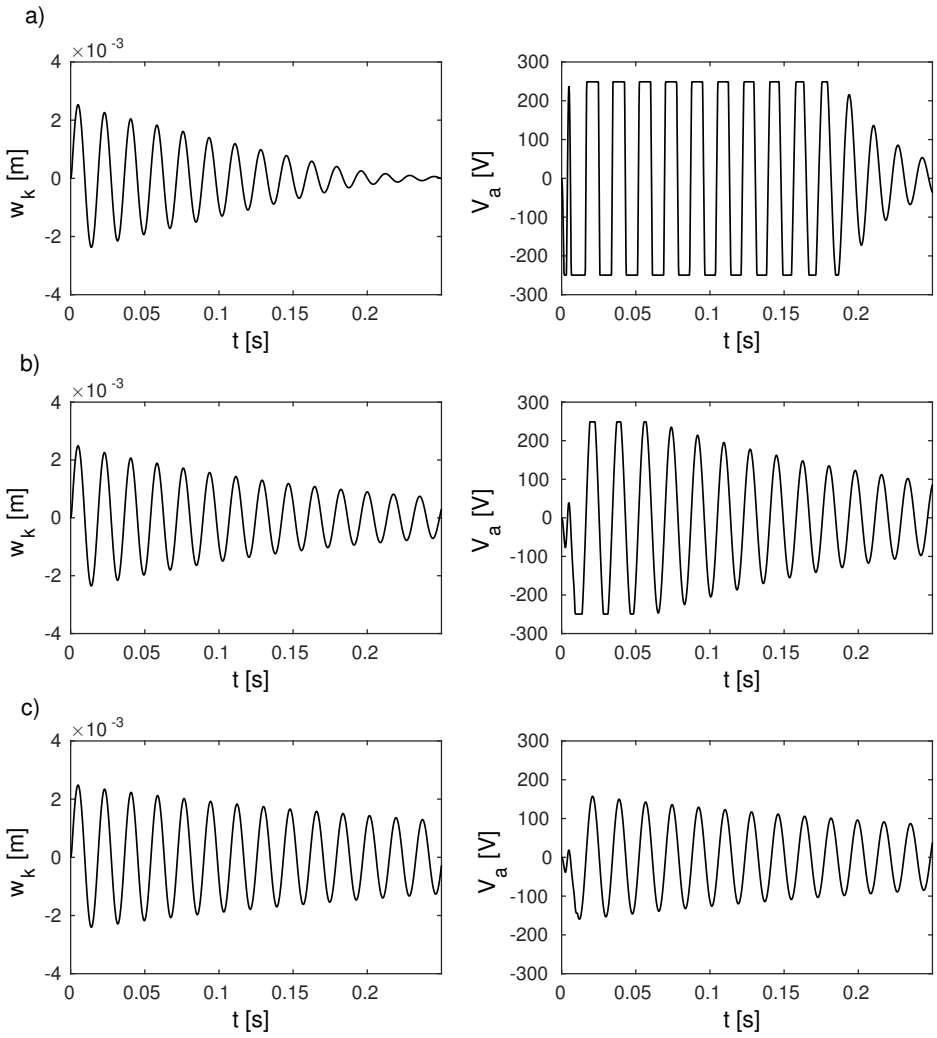
Choć specyfika oraz właściwości obu z zastosowanych metod sterowania neuronowego, w szczególności zastosowanych w nich sieci neuronowych oraz ich architektury, różnią się w sposób znaczący, sterowanie oraz otrzymane w jego wyniku wykresy przemieszczenia wolnego końca belki $w_k$ oraz napięcia wymuszającego na aktuatorze $V_a$, są dla obu przypadków podobne. Stąd też w dalszej części podrozdziału przedstawione są wielkości odpowiedzi układu oraz sygnału sterującego wyłącznie dla sterowania typu NARMA-L2. Poza właściwym sterowaniem aktywnym w układzie, dla każdej z form własnych belki przyjęte zostało tłumienie modalne o wartości $\zeta = 0.001$. Domyślnie wartości wzmocnienia $g$ oraz czasu próbkowania $\Delta t$ w zaprezentowanych testach przyjmuje się jak następuje: $g = 0.01$, $\Delta t = 0.002$ s.

- **Wzmocnienie $g$**

  Wyniki sterowania neuronowego przy uwzględnieniu trzech różnych wartości wzmocnienia $g$ na wyjściu sieci neuronowej przedstawiono na rysunku 3.21. Jak widać z wykresów, efektem zwiększonej wartości wzmocnienia $g$, jest wyraźna poprawa efektu tłumienia analizowanego układu. Odbywa się to jednak kosztem jednocześnie zwiększonej wartości wymaganego napięcia sterującego $V_a$ przyłożonego do elementu piezoelektrycznego pełniącego rolę wzbudnika. Dodatkowo jak widać z wykresu 3.21 (a), przy dobraniu zbyt dużych wartości wzmocnienia $g$, sieć wykazuje tendencję do wykraczania poza narzucony pierwotnie zakres wartości sterujących, które to następnie wartości ograniczane są przez zastosowany blok saturacji.



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ



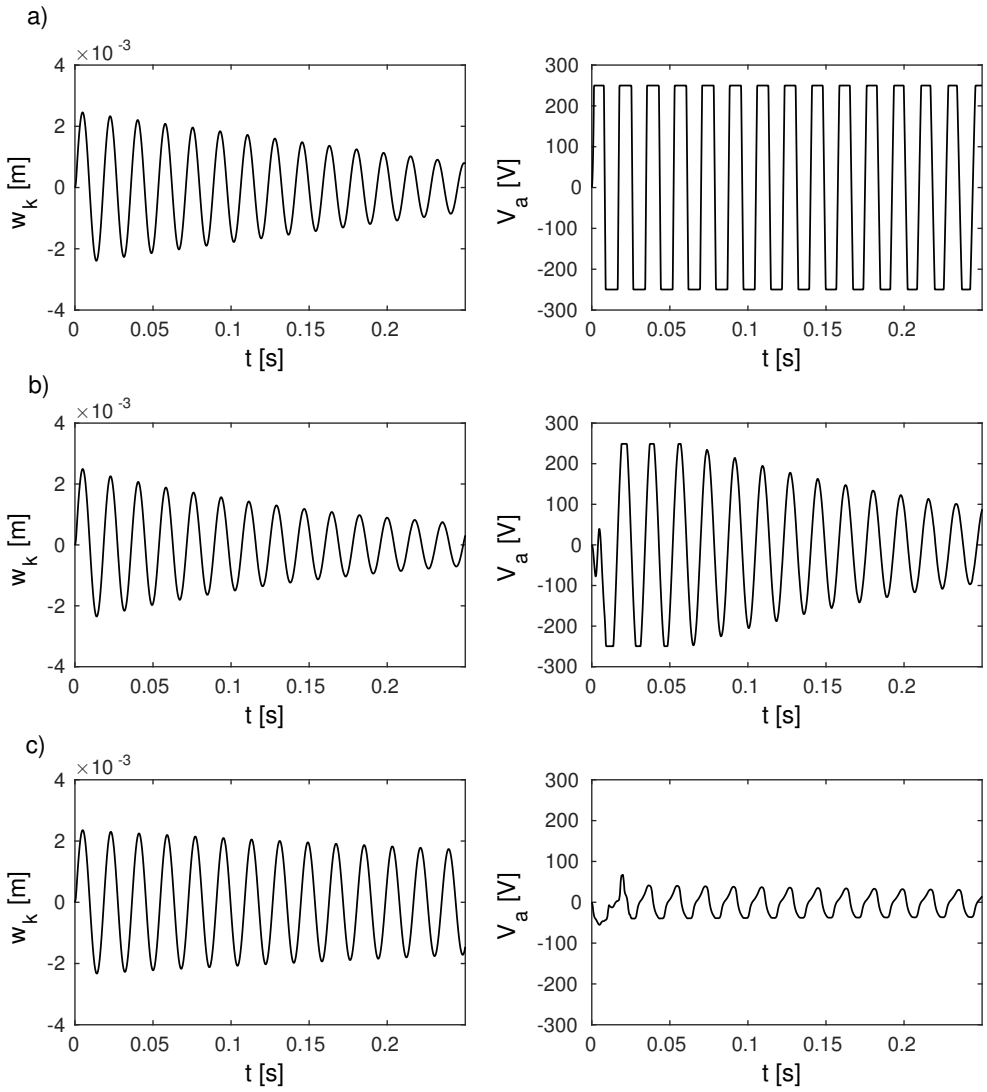

Rysunek 3.21.: Przemieszczenie końca belki $w_k$ oraz napięcie sterujące $V_a$ dla sterowania siecią neuronową typu NARMA-L2 dla różnych wartości wzmocnienia:
a) $g = 0.1$, b) $g = 0.01$, c) $g = 0.005$

- **Wpływ dobranego czasu próbkowania $\Delta t$**

  Na rysunku 3.22, pokazano w jaki sposób dobór czasu próbkowania $\Delta t$ wzorców (równoznaczny z doborem częstotliwości próbkowania $f_s$), wpływa na działanie sterownika neuronowego nauczonego na bazie tychże wzorców. Należy zaznaczyć, iż dla każdego z przedstawionych wykresów, zarówno architektura sieci jak i parametry uczenia (w tym ilość cykli oraz docelowa dokładność uczenia) są identyczne. Jedyna różnica sprowadza się do doboru wartości $\Delta t$, która determinuje szybkość działania sterownika i co ważne, musi być dobrana tak by uwzględniać dynamikę sterowanego układu z dostatecznie wysoką dokładnością.



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ




Rysunek 3.22.: Przemieszczenie końca belki $w_k$ oraz napięcie sterujące $V_a$ dla sterowania siecią neuronową typu NARMA-L2, nauczoną na bazie wzorców wygenerowanych z modelu, przy różnych czasach próbkowania:
a) $\Delta t = 0.001$ s, b) $\Delta t = 0.002$ s, c) $\Delta t = 0.005$ s

Dobranie zbyt niskiej wartości $\Delta t$ powoduje, że nauczony neuronowy układ stara się dotrzeć do założonej wartości zerowej w możliwie najkrótszym czasie, co przekłada się na wyznaczenie wartości napięcia sterującego $V_a$ wybiegających poza dopuszczalny dla elementu piezoelektrycznego zakres. Efekt ten jest wyraźnie widoczny na rysunku 3.22 a, gdzie z uwagi na ograniczenie wynikające z zastosowania bloku saturacji, napięcia sterujące oscyluje pomiędzy wartością maksymalną i minimalną. Z drugiej strony, zastosowanie zbyt dużej wartości $\Delta t$ (niskej częstotliwości $f_s$) skutkuje pogorszeniem działania układu sterowania (rys. 3.22 c). Stąd, jak widać, odpowiedni dobór wartości



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

czasu próbowania przy generacji wzorców uczących, przekłada się później na efektywność działania nauczonej na ich bazie sieci neuronowej. Konieczne jest zatem uwzględnienie możliwego wpływu tej wartości jeszcze przed rozpoczęciem właściwego procesu uczenia sieci (sterownika neuronowego).

- **Sterowanie z zastosowaniem bloku stopniowego dojścia do celu**

  Standardowo nauczony sterownik neuronowy typu „inverse" bądź „NARMA-L2" wykazuje tendencję do wyznaczania wartości sygnału sterującego, która w możliwie najkrótszym czasie pozwala na osiągnięcie pożądanego stanu układu. Niestety w przypadku gdy przejście pomiędzy położeniem początkowym a położeniem końcowych układu zadawane jest w sposób skokowy, bardzo często wartość sygnału sterującego ekstrapolowana przez sieć, wybiega poza dopuszczalny zakres narzucony w procesie uczenia sieci.

  Jednym z możliwych rozwiązań umożliwiających uniknięcie przedstawionego problemu jest implementacja dodatkowych krzywych przejścia definiujących w sposób ciągły, a nie skokowy, trajektorię zmiany sterowanych składowych stanu układu. Rozwiązanie to w niniejszej pracy zastosowane zostało w sposób pośredni dla opisanego wcześniej układu wahadła matematycznego (poprzez wprowadzenie dodatkowych bloków filtrów analogowych) oraz w sposób bezpośredni dla opisanych w dalszej części pracy układów wahadła podwójnego oraz giroskopu. Dla przypadku analizowanego układu belki inteligentnej zdecydowano się na przetestowanie podobnego rozwiązania polegającego na wprowadzeniu do układu sterowania dodatkowego bloku stopniowego dojścia do celu. Funkcjonowanie tego bloku polega na modyfikacji wartości docelowej sterowanego układu, tak że dla każdego kroku sterowania w miejsce pożądanej wartości stanu $y_r$, sieć ma za zadanie wysterować układ do wartości pośredniej $\hat{y}_r$, pomiędzy wartością obecną $y$ a wartością pożądaną $y_r$, zgodnie ze wzorem:

$$\hat{y}_r = y(1-d) + y_r d \tag{3.15}$$

  gdzie $d$ jest parametrem zmiany i zawiera się w przedziale $[0, 1]$. Dla przedstawionego przypadku tłumienia drgań belki inteligentnej, wartością sterowaną jest przemieszczenie końca belki $w_k$, tak więc przyjmując wartość docelową $w_{kr} = 0$, wzór (3.15) przyjmuje postać:

$$\hat{w}_{kr} = w_k(1-d) \tag{3.16}$$

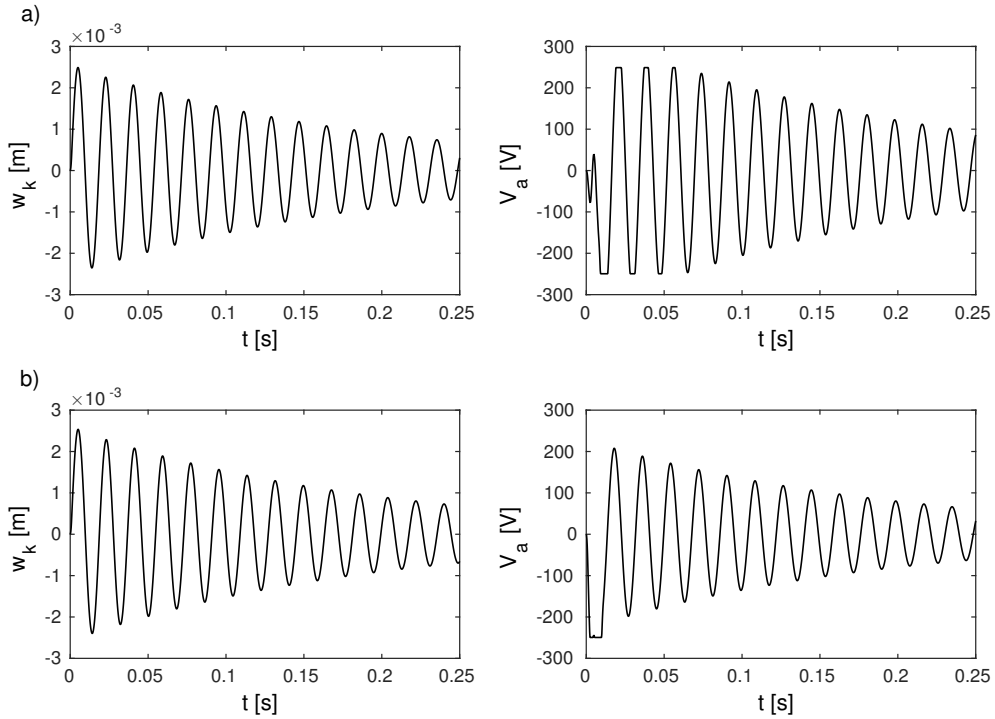
  Parametr $d$ determinuje różnicę pomiędzy wartością obecną $w_k$ a zmodyfikowaną wartością docelową $\hat{w}_{kr}$, w ten sposób wpływając na działanie sterowania neuronowego oraz szybkość przejścia pomiędzy obiema wartościami. W skrajnym przypadku gdy parametr $d = 1$, sterowanie odbywa się w sposób domyślny z wartością docelową $\hat{w}_{kr} = w_k = 0$. Z kolei dla wartości parametru $d = 0$, sterowanie jest niewykonalne, jako że zmody-



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

fikowana wartość docelowa $\hat{w}_{kr}$ pokrywa się z wartością aktualną $w_k$ dla danej chwili czasowej.

Na rysunku 3.23 pokazane są wykresy przemieszczenia końca belki $w_k$ oraz napięcia sterującego $V_a$ dla przypadku aktywnego neuronowego tłumienia drgań belki dla dwóch różnych wartości parametru $d$. Z wykresów widać, że przyjęcie wartości $d = 0.1$ tylko w niewielkim stopniu zmienia szybkość z jaką drgania belki są tłumione. Wyraźna różnica widoczna jest natomiast w wartościach napięcia sterującego $V_a$, dla zmodyfikowanego układu wartości te są niższe, jako że sieć stara się wyznaczyć wartość sterującą wyłącznie dla zredukowanego obszaru sterowania, w przeciwieństwie do pełnego zakresu dla układu bez zaproponowanej modyfikacji.




Rysunek 3.23.: Przemieszczenie końca belki $w_k$ oraz napięcie sterujące $V_a$ dla pierwszej formy własnej drgań przy sterowaniu siecią neuronową typu NARMA-L2 z wykorzystaniem bloku stopniowego dojścia do celu:
a) $d = 1$, b) $d = 0.1$

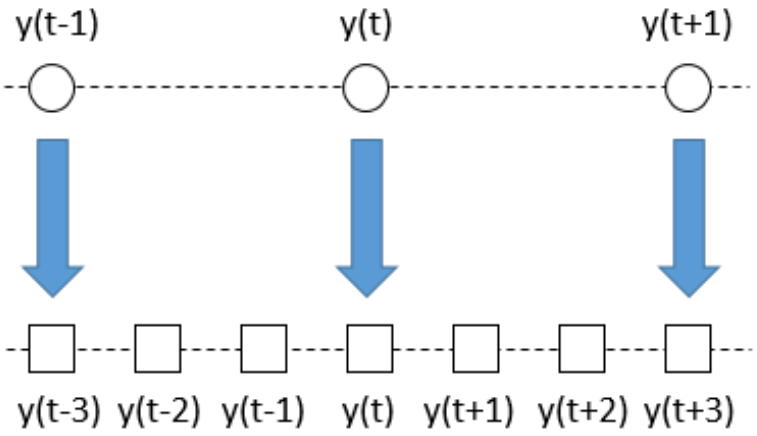
- **Tłumienie wyższych form własnych (rozszerzenie sterowania pierwszą formą na wyższe formy)**

  Dla rozpatrywanego układu belki inteligentnej z elementami piezoelektrycznymi, jedną z możliwych opcji aktywnego wytłumienia drgań wyższych form własnych jest zastosowanie sterownika neuronowego nauczonego uprzednio pod kątem tłumienia drgań pierwszej formy własnej belki. Działanie tak nauczonego sterownika neuronowego realizowalne jest



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

tylko i wyłącznie dla częstotliwości próbkowania $\Delta f$ narzuconej w procesie uczenia sieci. W celu rozszerzenia zakresu sterowania na wyższe formy własne układu, niezbędna jest zatem modyfikacja sterownika w taki sposób, by mógł on prawidłowo funkcjonować przy wyższej częstotliwości $\Delta\tilde{f}$. Dla częstotliwości tej muszą w tym wypadku zostać spełnione dwa warunki. Po pierwsze, musi ona być na tyle wysoka aby uwzględniona została dynamika sterowanych wyższych form własnych. Drugi warunek, zdefiniowany za pomocą wzoru $\Delta\tilde{f} = r \cdot \Delta f$, gdzie $r \in \mathbb{N}$, oznacza że nowa częstotliwość próbkowania $\Delta\tilde{f}$ dla sterowania jest wielokrotnością częstotliwości oryginalnej $f$.




Rysunek 3.24.: Rozszerzenie działania sieci na wyższą częstotliwość próbkowania poprzez modyfikację położenia jej regresorów

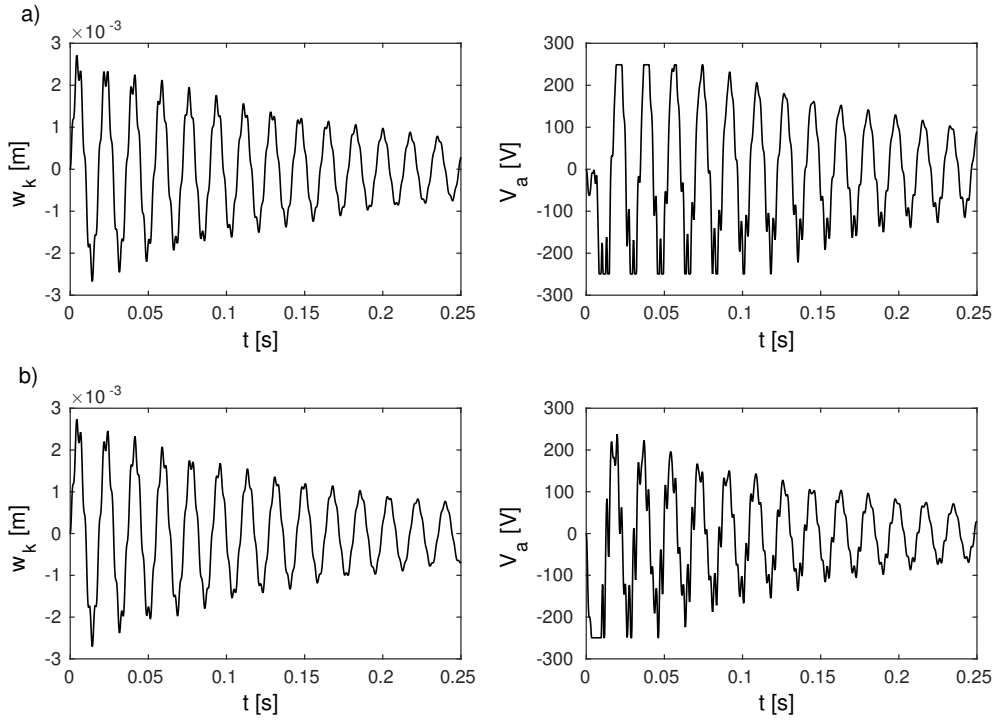
Dzięki temu, że częstotliwość $\Delta\tilde{f}$, zdefiniowana jest w taki, a nie inny sposób, możliwe jest prostsze dostosowanie sieci do zaistniałej zmiany, poprzez zmianę położenia regresorów do których odnosi się sieć, jak pokazano na rysunku 3.24. Wyższa częstotliwość próbkowania układu powoduje, że regresor $y(t-1)$ zastąpić należy regresorem $y(t-r)$, bądź w bardziej ogólnym przypadku regresor $y(t-rn)$ wprowadzany jest na wejście sieci w miejsce regresora $y(t-n)$. Podobna sytuacja ma również miejsce dla regresorów sygnału sterującego $u$. Oznacza to, że działanie tak zmodyfikowanego sterownika neuronowego pozostaje w gruncie rzeczy niezmienione (różnica w czasie pomiędzy regresorami nadal wynosi $\Delta t$), zmianie ulega natomiast szybkość jego działania, która dostosowana jest do nowej częstotliwości próbkowania $\Delta\tilde{f}$.

Na rysunku 3.25 przedstawione zostały przykładowe wykresy tłumienia drgań dwóch pierwszych form własnych belki przy wykorzystaniu przedstawionej powyżej metody dla $\Delta f = 100Hz$ oraz $\Delta\tilde{f} = 1000Hz$. Ponadto na przedstawionych wykresach uwzględnione zostało również działanie opisanego wyżej bloku stopniowego dojścia do celu, dla przypadków $d = 1$ oraz $d = 0.1$. Podobnie jak przy sterowaniu pierwszą formą własną układu, zastosowanie wspomnianego bloku wyraźnie ogranicza wartość sygnału



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

sterującego (w tym wypadku napięcia $V_a$) przy nieznacznej zmianie tłumienia drgań rozpatrywanego układu. Z wykresów tych widać, że regulator neuronowy skutecznie tłumi nie tylko pierwszą, ale również drugą formę własną belki.




Rysunek 3.25.: Przemieszczenie końca belki $w_k$ oraz napięcie sterujące $V_a$ dla dwóch pierwszych form własnych drgań przy sterowaniu siecią neuronową typu NARMA-L2 z wykorzystaniem bloku stopniowego dojścia do celu:
a) $d = 1$, b) $d = 0.1$

### Sterowanie przez emulację LQR (wyższe formy)

Ostatnim z prezentowanych przykładów sterowania neuronowego dla przypadku tłumienia drgań belki inteligentnej z elementami piezoelektrycznymi jest sterowanie z wykorzystaniem sieci neuronowej emulującej działanie już istniejącego sterownika klasycznego. W zaprezentowanym przypadku emulator neuronowy uczony jest na bazie próbek pochodzących ze sterownika korzystającego z metody optymalnej regulacji LQR [27]. Zaznaczyć przy tym należy, że metoda może być stosowana wyłącznie dla układów liniowych bądź układów zlinearyzowanych. Podstawą sterowania z wykorzystaniem metody LQR jest minimalizacja wskaźnika jakości $J$ zdefiniowanego w postaci funkcjonału:

$$J = \int_0^\infty (\mathbf{x}^T(t)\mathbf{Q}\mathbf{x}(t) + \mathbf{u}^T(t)\mathbf{R}\mathbf{u}(t))\, dt \tag{3.17}$$

gdzie $\mathbf{x}(t)$ jest wektorem stanu układu, $\mathbf{u}(t)$ wektorem wymuszenia, natomiast macierze $\mathbf{R}$ oraz $\mathbf{Q}$ są dobranymi odgórnie macierzami wag. Odpowiedni dobór macierzy wag determinuje



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

nacisk położony na wektor stanu $\mathbf{x}(t)$ oraz wektor wymuszenia $\mathbf{u}(t)$ w funkcjonale $J$, co z kolei pozwala na modyfikację zachowania sterownika. Na potrzeby wyznaczenia sterowania optymalnego zakłada się, że wektory te powiązane są ze sobą poprzez zależność:

$$\mathbf{u}(t) = \mathbf{G}\mathbf{x}(t) \tag{3.18}$$

gdzie $\mathbf{G}$ definiuje macierz wzmocnienia. Po uwzględnieniu powyższej zależności, równanie stanu oraz wyjścia układu prezentują się w postaci:

$$\begin{cases} \dot{\mathbf{x}}(t) = [\mathbf{A} - \mathbf{B}\mathbf{G}]\,\mathbf{x}(t) \\ \mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) \end{cases} \tag{3.19}$$

Optymalne sterowanie dla tak zdefiniowanego układu liniowego otrzymuje się w przypadku gdy macierz wzmocnienia $\mathbf{G}$ określona jest poprzez zależność:

$$\mathbf{G} = \mathbf{R}^{-1}\mathbf{B}^{\mathbf{T}}\mathbf{K} \tag{3.20}$$

Macierz $\mathbf{K}$ występującą w powyżej zależności, otrzymuje się z kolei jako rozwiązanie tzw. równania Ricattiego:

$$-\mathbf{Q} - \mathbf{A}^{\mathbf{T}}\mathbf{K} - \mathbf{K}\mathbf{A} + \mathbf{K}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^{\mathbf{T}}\mathbf{K} = \mathbf{0} \tag{3.21}$$

Dla przypadku belki inteligentnej macierze wag $\mathbf{Q}$ i $\mathbf{R}$ dobrane zostały w sposób następujący:

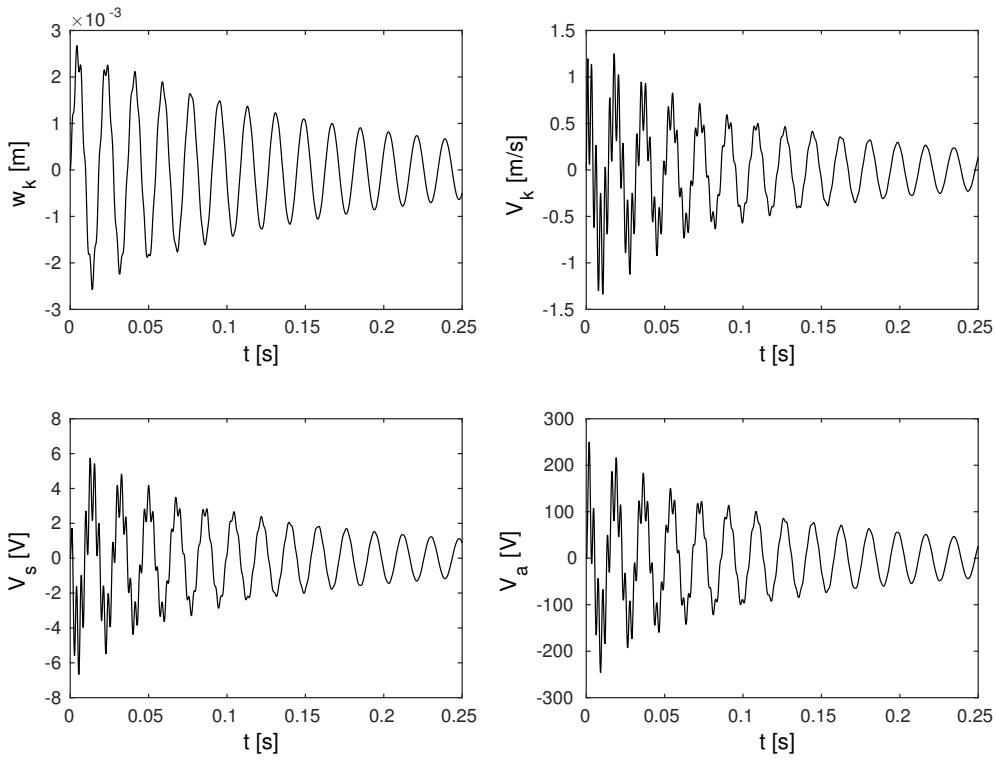
$$\mathbf{Q} = p_Q \begin{bmatrix} [\mathbf{\Omega^2}]_{m\times m} & [\mathbf{0}]_{m\times m} \\ [\mathbf{0}]_{m\times m} & [\mathbf{1}]_{m\times m} \end{bmatrix}_{2m\times 2m} \qquad \mathbf{R} = p_R$$

gdzie $\mathbf{\Omega^2}$ jest, jak opisano wcześniej w rozdziale 2 poświęconym identyfikacji neuronowej macierzą diagonalną częstości własnych układu, natomiast $p_Q$ oraz $p_R$ są parametrami za pomocą których w prosty sposób regulować można w jakiej proporcji wartość wskaźnika jakości $J$ zależna jest od macierzy wag $\mathbf{Q}$ i $\mathbf{R}$.

Dla sterownika LQR, który posłużył później jako wzorzec nauki emulatora neuronowego, przyjęto wartości podanych wyżej parametrów $p_Q = 2.6$, $p_R = 1$, natomiast samo sterowanie odbywało się przy ograniczeniu modelu do dwóch pierwszych form własnych. Dodatkowo w układzie dla obu form własnych uwzględnione zostało tłumienie modalne o wartości $\zeta = 0.005$. Dla tak zdefiniowanego sterownika przeprowadzona została następnie symulacja numeryczna sterowania, przy założonym czasie próbkowania $\Delta t = 0.1$ ms oraz ograniczeniu wartości napięcia sterującego w elemencie wykonawczym do zakresu $[-250, 250]$ V. Jako emulator neuronowy zastosowano standardową sieć dwuwarstwową typu NNARX (15-1), która została nauczona na bazie próbek pobranych w trakcie symulacji.



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ



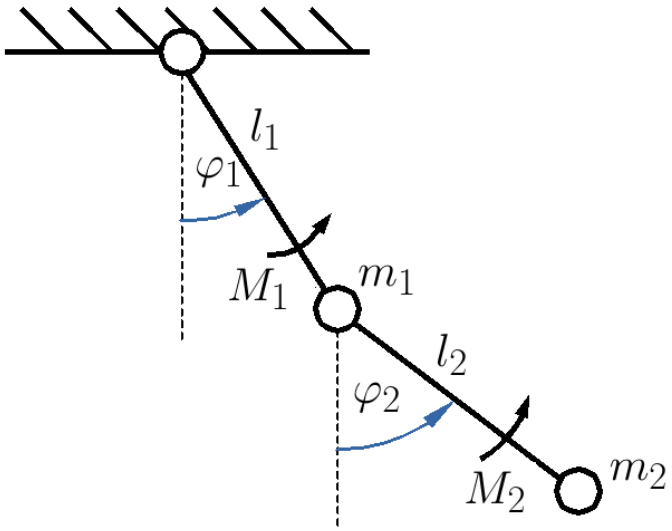

Rysunek 3.26.: Przemieszczenie $w_k$ i prędkość $V_k$ końca belki oraz napięcie wyjściowe $V_s$ i sterujące $V_a$ dla zagadnienia tłumienia dwóch pierwszych form własnych belki za pomocą sieci neuronowej emulującej zachowanie sterownika LQR
Parametry LQR: $p_Q = 2.6$, $p_R = 1$
Parametry sieci: NNARX (15-1), liczba regresorów: $m = 4$, $k = 4$, częstotliwość próbkowania $f_s = 10$ kHz, zakres napięcia sterującego $V_a = [-250, 250]$ V

Na rysunku 3.26 pokazane są wykresy tłumienia drgań belki, przy wykorzystaniu tak nauczonego emulatora neuronowego. Wykresy te obrazują przemieszczenie $w_k$ oraz prędkość $V_k$ końca belki jak również napięcie $V_s$ przyłożone do czujnika piezoelektrycznego oraz napięcie sterujące $V_a$ na aktuatorze. Tak nauczona sieć neuronowa jest w stanie w wierny sposób emulować zachowanie sterownika LQR. O ile w tym przypadku sterowanie klasyczne jest zdecydowanie prostsze w realizacji, o tyle zaprezentowany pomysł uczenia sterownika neuronowego jest ciekawy, w szczególności pod kątem możliwości rozszerzenia go na układy nieliniowe, przykładowo poprzez nauczenie sieci neuronowej emulacji zachowania szeregu sterowników klasycznych działających w kilku odrębnych, lecz sąsiadujących ze sobą zlinearyzowanych obszarach, przy dodatkowym zastosowaniu funkcji przełączających w celu przejścia pomiędzy nimi.



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

### 3.2.3. Wahadło podwójne nieliniowe




Rysunek 3.27.: Model wahadła podwójnego

Jako kolejny przykład sterowania neuronowego zostanie rozważony model wahadła podwójnego, będący rozszerzeniem opisanego wcześniej modelu wahadła matematycznego. Wybór modelu wahadła podwójnego podyktowany został jego nieliniowymi właściwościami dynamicznymi. Zachowanie dynamiczne wahadła opisane jest za pomocą układu równań nieliniowych:

$$\begin{cases} (m_1+m_2)l_1^2\ddot{\varphi}_1 + m_2 l_1 l_2 \ddot{\varphi}_2 \cos(\varphi_2-\varphi_1) - m_2 l_1 l_2 \dot{\varphi}_2^2 \sin(\varphi_2-\varphi_1) + \\ \qquad +(m_1+m_2)gl_1\sin\varphi_1 = M_1 \\ m_2 l_2^2 \ddot{\varphi}_2 + m_2 l_1 l_2 \ddot{\varphi}_1 \cos(\varphi_2-\varphi_1) + m_2 l_1 l_2 \dot{\varphi}_1^2 \sin(\varphi_2-\varphi_1) + \\ \qquad +m_2 g l_2 \sin\varphi_2 = M_2 \end{cases} \tag{3.22}$$

które to równania przy założeniu $m = m_1 = m_2$, $l = l_1 = l_2$ upraszczają się do postaci:

$$\begin{cases} 2ml^2\ddot{\varphi}_1 + ml^2\ddot{\varphi}_2\cos(\varphi_2-\varphi_1) - ml^2\dot{\varphi}_2^2\sin(\varphi_2-\varphi_1) + 2mgl\sin\varphi_1 = M_1 \\ ml^2\ddot{\varphi}_2 + ml^2\ddot{\varphi}_1\cos(\varphi_2-\varphi_1) + ml^2\dot{\varphi}_1^2\sin(\varphi_2-\varphi_1) + mgl\sin\varphi_2 = M_2 \end{cases} \tag{3.23}$$

#### Obserwowalność i sterowalność

Wartości definiujące obserwowalność i sterowalność wybranego układu dynamicznego pozwalają na określenie, w jakim stopniu możliwa jest znajomość aktualnego stanu układu jak również do jakiego stopniu możliwym jest jego wysterowanie. Jako że macierze obserwowal-



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

ności $\mathcal{O}$ (3.24) oraz sterowalności $\mathcal{C}$ (3.25) wyznaczane są na podstawie znajomości macierzy stanu $\mathbf{A}$, wejścia $\mathbf{B}$ i wyjścia $\mathbf{C}$ układu, ich analiza przeprowadzona może zostać wyłącznie dla układów liniowych bądź układów po linearyzacji.

$$\mathcal{O} = \begin{bmatrix} \mathbf{C} & \mathbf{CA} & \cdots & \mathbf{CA^{n-1}} \end{bmatrix}^{\mathbf{T}} \tag{3.24}$$

$$\mathcal{C} = \begin{bmatrix} \mathbf{B} & \mathbf{AB} & \cdots & \mathbf{A^{n-1}B} \end{bmatrix} \tag{3.25}$$

Pomimo faktu, iż przedstawiony układ wahadła podwójnego jest układem nieliniowym, wyznaczenie dla niego wartości obserwowalności i sterowalności w zlinearyzowanym zakresie, może pozwolić choćby w częściowym stopniu na określenie możliwości jego wysterowania w pełniejszym nieliniowym zakresie.

Przedstawiony wyżej układ równań nieliniowych dynamiki wahadła podwójnego (3.23), po linearyzacji względem położenia zerowego równowagi sprowadza się do postaci:

$$\begin{cases} 2ml^2\ddot{\varphi}_1 + ml^2\ddot{\varphi}_2 + 2mgl\varphi_1 = M_1 \\ ml^2\ddot{\varphi}_1 + ml^2\ddot{\varphi}_2 + mgl\varphi_2 = M_2 \end{cases} \tag{3.26}$$

bądź do postaci macierzowej:

$$\mathbf{M}\ddot{\boldsymbol{\varphi}} + \mathbf{K}\boldsymbol{\varphi} = \mathbf{u} \tag{3.27}$$

z macierzami $\mathbf{M}$ i $\mathbf{K}$ oraz wektorami $\boldsymbol{\varphi}$ i $\mathbf{u}$ wynoszącymi odpowiednio:

$$\mathbf{M} = \begin{bmatrix} 2ml^2 & ml^2 \\ ml^2 & ml^2 \end{bmatrix}_{2\times 2} \quad \mathbf{K} = \begin{bmatrix} 2mgl & 0 \\ 0 & mgl \end{bmatrix}_{2\times 2}$$

$$\boldsymbol{\varphi} = \begin{bmatrix} \varphi_1 \\ \varphi_2 \end{bmatrix} \quad \mathbf{u} = \begin{bmatrix} M_1 \\ M_2 \end{bmatrix}$$

Równanie 3.27 przekształcić można do postaci:

$$\ddot{\boldsymbol{\varphi}} = -\mathbf{M^{-1}K}\boldsymbol{\varphi} + \mathbf{M^{-1}u} \tag{3.28}$$

z macierzą odwrotną $\mathbf{M^{-1}}$ wynoszącą:

$$\mathbf{M^{-1}} = \frac{1}{ml^2}\begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix}_{2\times 2}$$

Następnie równanie to sprowadzić można do postaci równania stanu i równania wyjścia:

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu} \\ \mathbf{y} = \mathbf{Cu} \end{cases} \tag{3.29}$$



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

$$\mathbf{x} = \begin{bmatrix} \varphi_1 & \varphi_2 & \dot{\varphi_1} & \dot{\varphi_2} \end{bmatrix}^T$$

z macierzami $\mathbf{A}$, $\mathbf{B}$ i $\mathbf{C}$ wynoszącymi dla tak przyjętego wektora stanu $\mathbf{x}$ odpowiednio:

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}_{2\times2} & \mathbf{I}_{2\times2} \\ -\mathbf{M^{-1}K}_{2\times2} & \mathbf{0}_{2\times2} \end{bmatrix}_{4\times4} \quad \mathbf{B} = \begin{bmatrix} \mathbf{0}_{2\times2} \\ \mathbf{M^{-1}}_{2\times2} \end{bmatrix}_{4\times2}$$

$$\mathbf{C} = \begin{bmatrix} \mathbf{I}_{2\times2} & \mathbf{0}_{2\times2} \end{bmatrix}_{2\times4}$$

oraz wynikającym z nich wektorem wyjściowym $\mathbf{y}$:

$$\mathbf{y} = \begin{bmatrix} \varphi_1 & \varphi_2 \end{bmatrix}^T$$

Z definicji układ liniowy jest w pełni obserwowalny i sterowalny wtedy i tylko wtedy gdy odpowiednio rząd macierzy obserwowalności $\mathcal{O}$ i sterowalności $\mathcal{C}$ równe są wymiarowi $n$ wektora stanu układu $\mathbf{x}$.

Dla rozpatrywanego układu wahadła podwójnego macierze $\mathcal{C}$ i $\mathcal{O}$ wynoszą odpowiednio:

$$\mathcal{C} = \begin{bmatrix} \mathbf{0}_{2\times2} & \mathbf{M^{-1}}_{2\times2} & \mathbf{0}_{2\times2} & -\mathbf{M^{-1}KM^{-1}}_{2\times2} \\ \mathbf{M^{-1}}_{2\times2} & \mathbf{0}_{2\times2} & -\mathbf{M^{-1}KM^{-1}}_{2\times2} & \mathbf{0}_{2\times2} \end{bmatrix}_{4\times8}$$

$$\mathcal{O} = \begin{bmatrix} \mathbf{I}_{2\times2} & \mathbf{0}_{2\times2} \\ \mathbf{0}_{2\times2} & \mathbf{I}_{2\times2} \\ -\mathbf{M^{-1}K}_{2\times2} & \mathbf{0}_{2\times2} \\ \mathbf{0}_{2\times2} & -\mathbf{M^{-1}K}_{2\times2} \end{bmatrix}_{8\times4}$$

Z macierzy tych wyraźnie widać, że $r(\mathcal{C}) = r(\mathcal{O}) = n = 4$, a co za tym idzie przedstawiony układ wahadła podwójnego jest w zlinearyzowanym zakresie w pełni obserwowalny i sterowalny.

### Sterowanie typu „inverse"

Dla przypadku sterowania wahadłem podwójnym nieliniowym zdecydowano się na zastosowanie sterownika neuronowego typu „inverse", głównie z uwagi na możliwość bezpośredniego rozszerzenia jego architektury na sterowanie MIMO, podobnie jak zostało to uprzednio pokazane na przykładzie sterowania MISO dla wahadła matematycznego. Dla sterowania typu „inverse" wahadłem podwójnym przetestowane zostały dwa odrębne przypadki. W pierwszym z nich, sieć neuronowa nauczona została w taki sposób, aby wartości momentów sterujących $M_1$ oraz $M_2$ wyznaczane były na bazie pełnego wektora stanu, czyli obu położeń kątowych $\varphi_1$ i $\varphi_2$ oraz prędkości kątowych $\omega_1$ i $\omega_2$. W drugim przypadku, sterowanie neuronowe odbywało się dla okrojonej sieci neuronowej, uczonej tylko i wyłącznie na bazie położeń kątowych ($\varphi_1$, $\varphi_2$) poszczególnych elementów wahadła, z pominięciem odpowiadających im prędkości kątowych ($\omega_1$, $\omega_2$). Na potrzeby niniejszej symulacji przyjęto długość prętów wahadła $l = 0.5$ m



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

oraz wartość mas na ich końcach $m = 0.2$ kg.

- **Sterowanie 4-2** ($\varphi_1$, $\varphi_2$, $\omega_1$, $\omega_2$)-($M_1$, $M_2$)

  Na rysunku 3.28 przedstawione zostały wykresy położeń oraz prędkości kątowych wahadła podwójnego dla przypadku sterowania typu „inverse" z siecią neuronową nauczoną na bazie pełnej informacji na temat rzeczywistego stanu wahadła. Wykresy 3.28 (a), obrazują sterowanie domyślne sieci dla warunków początkowych:

  $$\varphi_1(0) = \pi/2, \quad \varphi_2(0) = -\pi/2, \quad \omega_1(0) = 0, \quad \omega_2(0) = 0$$

  Jak widać z przedstawionych wykresów, nauczona sieć neuronowa jest w stanie w sposób bezpośredni przeprowadzić układ do wartości zerowych dla wszystkich założonych wartości sterowanych (obu położeń oraz prędkości kątowych). Jednakże, z wykresów tych widać również, że sterowanie to odbywa się w sposób nieciągły (co jest szczególnie wyraźne dla wykresów prędkości kątowych $\omega_1$ i $\omega_2$). Potwierdzają to wykresy momentów sterujących dla przypadku domyślnego (Rys. 3.29 a) - podobnie jak dla opisywanych wcześniej modeli, sterowanie tak nauczoną siecią wykazuje tendencję do ekstrapolacji wartości momentów sterujących $M_1$ oraz $M_2$ poza dopuszczalny zakres, a przy uwzględnieniu saturacji w układzie, do skokowej ich zmiany pomiędzy wartościami minimalnymi a maksymalnymi.

  Rozwiązaniem w tym wypadku jest podobnie jak dla opisywanych wcześniej modeli wahadła oraz belki inteligentnej, wprowadzenie do układu dodatkowych modyfikacji, które usprawnią działanie sterownika neuronowego. Wykresy na rysunku 3.28 (b) obrazują sterowania przy tak zmodyfikowanym sterowaniu. Wprowadzone zmiany obejmują:

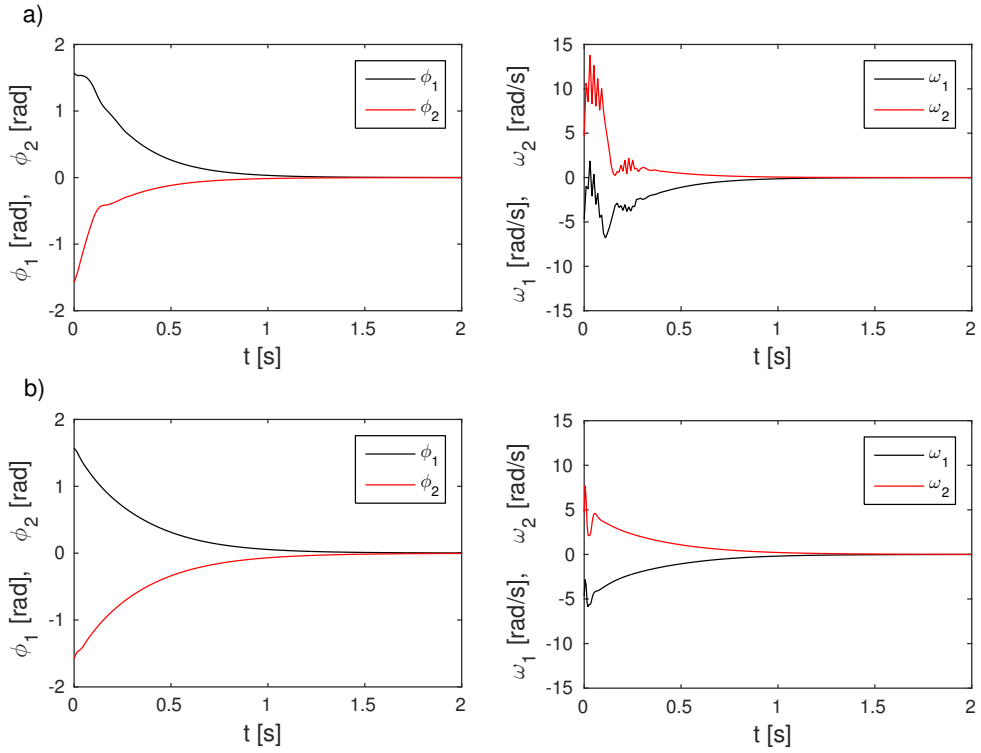
  - zdefiniowanie krzywych przejścia dla poszczególnych składowych sterowanych
  - zredukowanie wartości wzmocnienia - $g = 0.3$
  - dwukrotnie przyspieszenie działania sieci - $r = 2$

  Krzywe przejścia podobnie jak opisane przy okazji wcześniejszych modeli filtry analogowe II rzędu oraz blok stopniowego dojścia do celu, mają za zadanie zapewnić ciągłą zmianę wartości sterowanych, a co za tym idzie zredukować wartość sygnałów sterujących wyznaczanych przez sieć neuronową. Dla zaprezentowanego przypadku sterowania wahadłem podwójnym, krzywe przejścia dla wartości położeń kątowych $\phi_1$ i $\phi_2$, zdefiniowane są za pomocą funkcji eksponencjalnych w postaci:

  $$\begin{cases} \varphi_{1d} = \varphi_1(0)e^{-w_1 t} \\ \varphi_{2d} = \varphi_2(0)e^{-w_2 t} \end{cases} \tag{3.30}$$



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

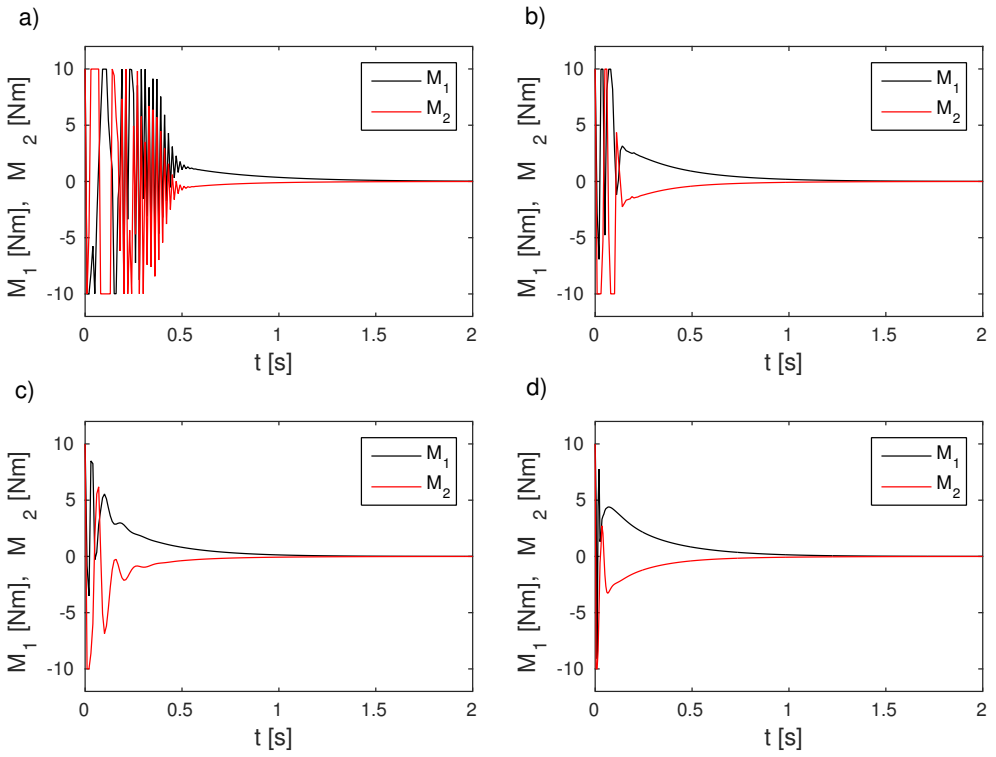



Rysunek 3.28.: Wykresy położeń kątowych $\varphi_1$ i $\varphi_2$ oraz prędkości kątowych $\omega_1$ i $\omega_2$ dla obu elementów wahadła podwójnego dla przypadku sterowania neuronowego typu "inverse"4-2
a) sterowanie domyślne (wzmocnienie $g = 1$)
b) sterowanie zmodyfikowane (zejście po krzywych eksponencjalnych, wzmocnienie $g = 0.3$, przyspieszenie działania sieci $r = 2$)

Dla tak zmodyfikowanego układu, zaobserwować można znaczną poprawę skuteczności sterowania neuronowego, co widoczne jest zarówno na wykresach wartości sterowanych (3.28 b) jak również momentów sterujących wyznaczanych przez sieć neuronową (3.29 d). Na rysunku tym, pokazana jest stopniowa poprawa wartości momentów sterujących ($M_1$, $M_2$) przy wprowadzaniu kolejnych modyfikacji do układu sterowania, poczynając od przypadku sterowania domyślnego, aż do sterowania w pełni zmodyfikowanego.



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ



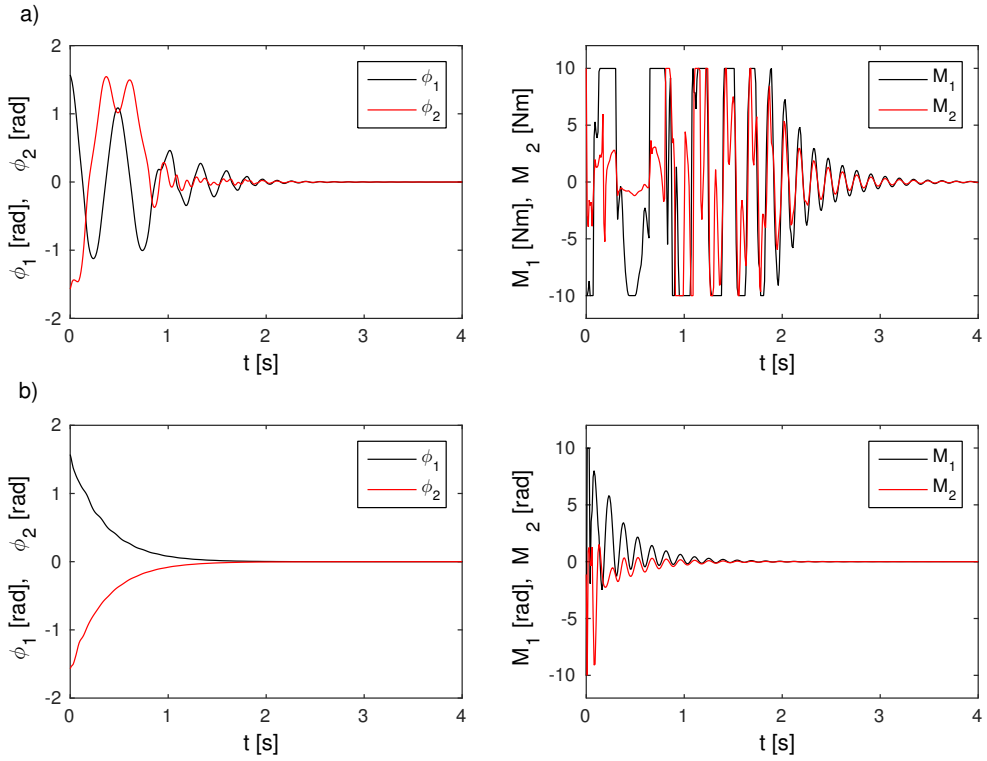

Rysunek 3.29.: Wykresy momentów sterujących $M_1$ oraz $M_2$ dla kolejnych stopni modyfikacji sterowania
a) sterowanie domyślne (wzmocnienie $g = 1$)
b) sterowanie zmodyfikowane (zejście po krzywych eksponencjalnych, wzmocnienie $g = 1$)
c) sterowanie zmodyfikowane (zejście po krzywych eksponencjalnych, wzmocnienie $g = 0.3$)
d) sterowanie zmodyfikowane (zejście po krzywych eksponencjalnych, wzmocnienie $g = 0.3$, przyspieszenie działania sieci $r = 2$)

- **Sterowanie 2-2** ($\varphi_1$, $\varphi_2$)-($M_1$, $M_2$)

  Uczenie sieci neuronowej dla sterowania typu „inverse" 2-2 odbywa się identycznie jak dla sterowania 4-2, z pominięciem na wejściu sieci obu prędkości kątowych $\omega$ wahadła. Jak widać na rysunku 3.30 a, domyślne sterowanie neuronowe wahadła podwójnego dla tak nauczonej sieci jest wolniejsze, natomiast wykresy wartości sterowanych oraz momentów sterujących wykazują znaczne oscylacje w porównaniu z odpowiadającymi im wykresami dla sterowania 4-2. Zastosowanie podobnych modyfikacji jak poprzednio, w znaczy sposób wpływa na poprawę skuteczności sterowania wahadła (3.30 b).



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

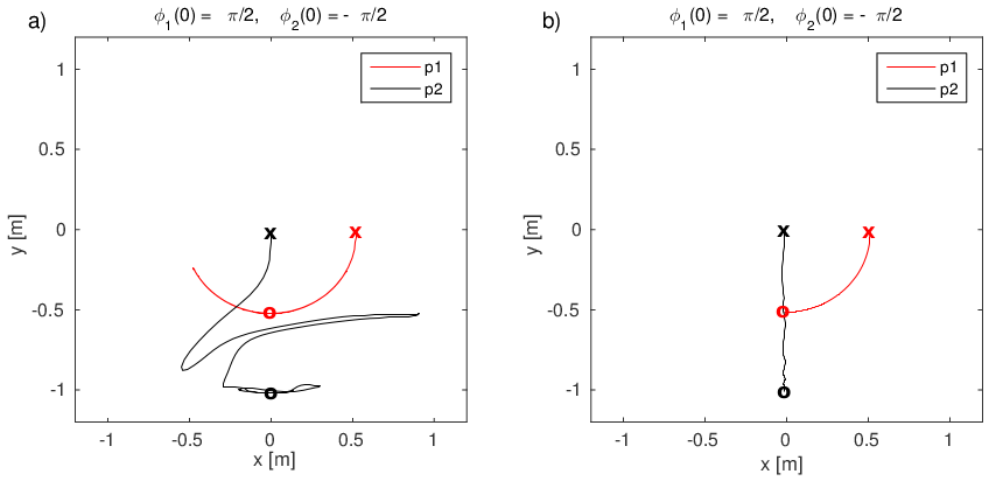



Rysunek 3.30.: Wykresy położeń kątowych $\varphi_1$ i $\varphi_2$ oraz momentów sterujących $M_1$ i $M_2$ dla obu elementów wahadła podwójnego dla przypadku sterowania neuronowego typu "inverse"2-2
a) sterowanie domyślne (wzmocnienie $g = 1$)
b) sterowanie zmodyfikowane (zejście po krzywych eksponencjalnych, wzmocnienie $g = 0.3$, przyspieszenie działania sieci $r = 2$)

Na rysunku 3.31 pokazane są dodatkowo wykresy położeń $p1$ oraz $p2$ obu mas $(m_1, m_2)$ wahadła podwójnego dla pokazanego wyżej domyślnego oraz zmodyfikowanego sterowania 2-2 typu „inverse". Położenie obu mas zdefiniowane jest we współrzędnych kartezjańskich przy dodatkowym założeniu że początek układu współrzędnych (punkt 0) znajduje się w punkcie utwierdzenia wahadła.



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ




Rysunek 3.31.: Krzywe zmiany położenia ($p1$, $p2$) na płaszczyźnie kartezjańskiej odpowiednio masy $m_1$ i $m_2$ wahadła podwójnego od położenia początkowego **x** do położenia końcowego **o** dla przypadku sterowania neuronowego typu "inverse"2-2;
a) sterowanie domyślne (wzmocnienie $g = 1$)
b) sterowanie zmodyfikowane (zejście po krzywych eksponencjalnych, wzmocnienie $g = 0.3$, przyspieszenie działania sieci $r = 2$)

### 3.2.4. Giroskop

Nieliniowy model giroskopu o dwóch stopniach swobody wraz z opisem jego zachowania dynamicznego oraz metod jego neuronowej emulacji, przedstawiony został uprzednio w podrozdziale 2.4.2. Dla tak przyjętego modelu giroskopu, zademonstrowane jest w niniejszym rozdziale sterowanie neuronowe z wykorzystaniem metody odwrotnej („inverse") oraz emulatora sterownika klasycznego LQR, podobnie jak to miało miejsce przy tłumieniu drgań dla modelu belki inteligentnej z piezoelektrykami. W obu przypadkach zadaniem układu było wysterowanie do wartości zerowych obu kątów obrotu ramki (kąta nutacji $\vartheta$ oraz precesji $\psi$) za pomocą odpowiadających im momentów sterujących $M_y$ oraz $M_Z$. Dla obu momentów przyjęto w procesie uczenia sieci, ograniczenie wartości do zakresu $[-1, 1]$ Nm. Dodatkowo dla każdego z rozpatrywanych przypadków sieć uczona była na bazie próbek pobranych z giroskopu przy stałej wartości kątowej wirnika $n = 10$ rad/s.

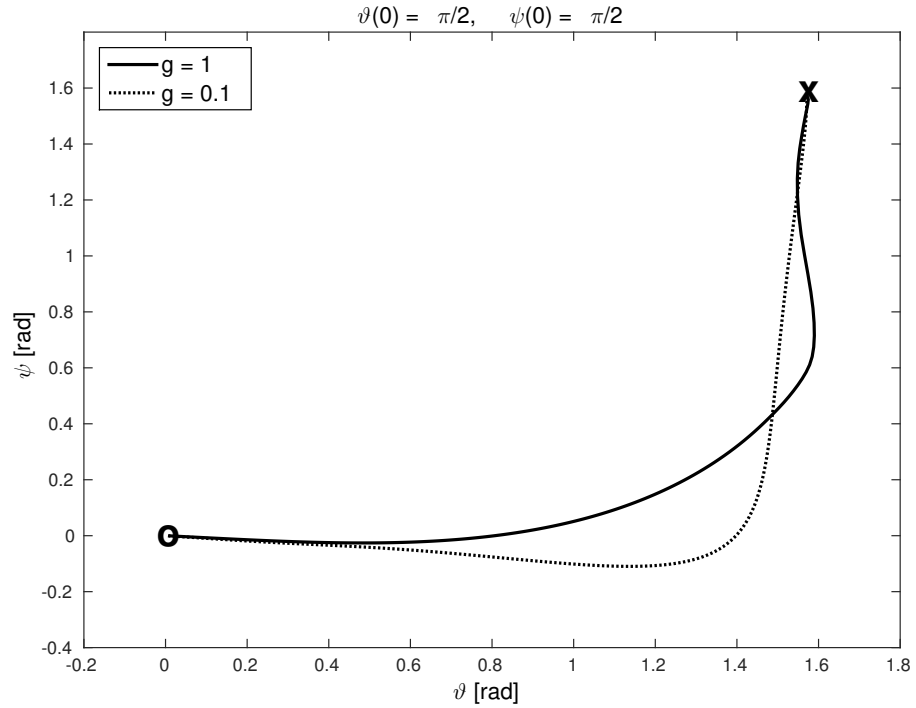
#### Sterowanie typu „inverse"

Na rysunku 3.32 pokazane są krzywe odpowiadające sterowaniu kątami precesji $\psi$ oraz nutacji $\vartheta$ z użyciem właściwie nauczonego sterownika neuronowego 2-2 typu „inverse". W celu sprawdzenia poprawności działania sterownika w zakresie nieliniowym giroskopu jako wartość początkową kątów $\vartheta$ i $\psi$ przyjęto wartość $\pi/2$. Różnica pomiędzy krzywymi na rysunku 3.32 polega na zastosowaniu różnej wartości wzmocnienia $g$ (odpowiednio 1 oraz 0.1). Odpowiadające im wykresy momentów sterujących $M_y$ oraz $M_Z$ przedstawione są na rysunku 3.33. Jak widać na przedstawionych wykresach, wysterowanie układu dla obu wartości wzmocnie-



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

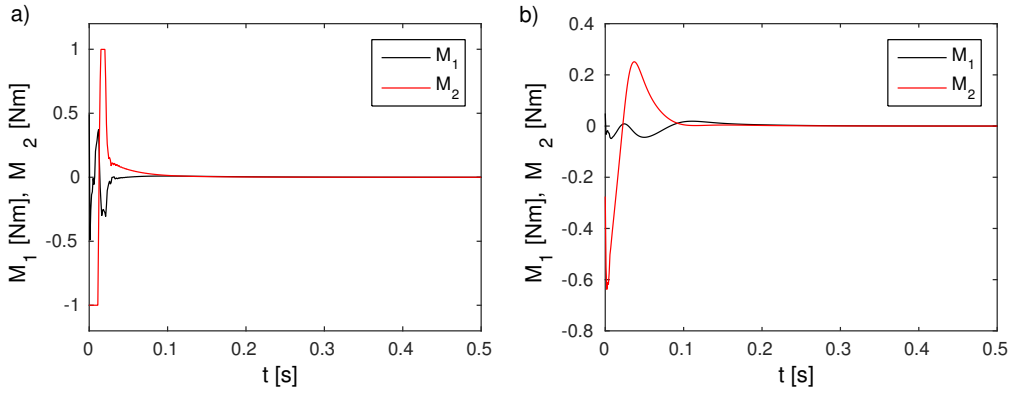
nia $g$ odbywa się w czasie poniżej 0.2 s. Dodatkowo, jak zostało to już również pokazane dla wcześniejszych modeli, zmniejszenie wartości wzmocnienia $g$ umożliwia uniknięcie zmiany skokowej sygnałów sterujących, przy nieznacznym wydłużeniu czasu potrzebnego na wysterowanie układu. Co jednak najważniejsze, badania symulacyjne wskazują, że zastosowana sieć neuronowa jest w stanie w skuteczny sposób sterować układem zarówno w zakresie liniowym jak i nieliniowym.




Rysunek 3.32.: Krzywe sterowania kątami precesji $\psi$ oraz nutacji $\vartheta$ giroskopu dla przypadku sterowania neuronowego typu „inverse" z uwzględnieniem modyfikacji wartości wzmocnienia $g$



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ



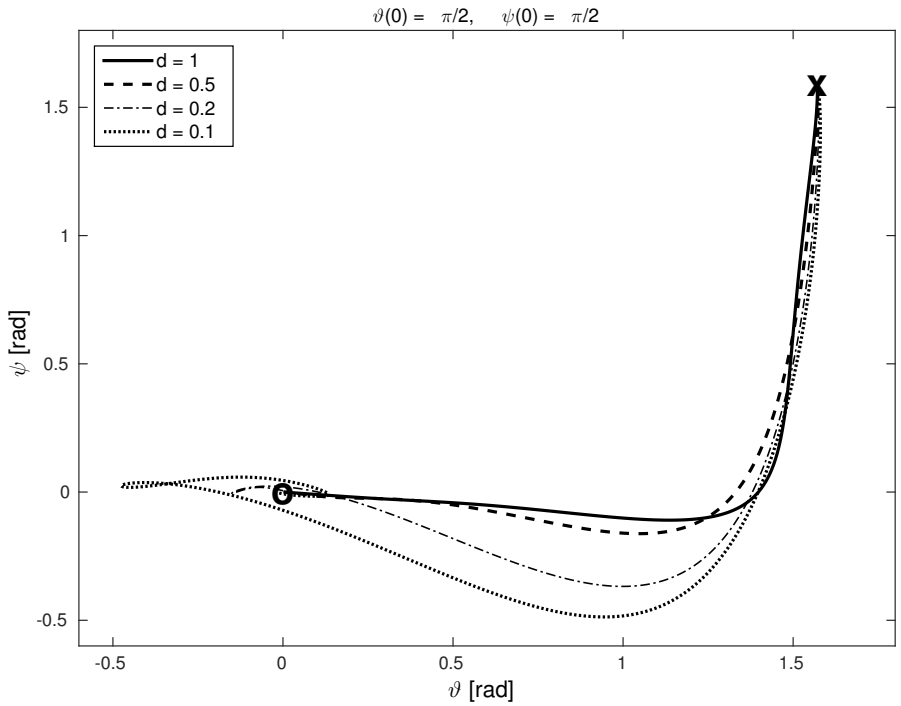

Rysunek 3.33.: Krzywe momentów sterujących $M_y$ i $M_z$ dla przypadku sterowania neuronowego typu „inverse" z uwzględnieniem modyfikacji wartości wzmocnienia $g$
a) $g = 1$
b) $g = 0.1$

Dla tak nauczonego sterownika neuronowego przeprowadzona została dodatkowo symulacja sterowania dla przypadku stałej wartości wzmocnienia $g = 0.1$, lecz z uwzględnieniem bloku stopniowego dojścia do celu, przy kilku różnych wartościach parametru $d$ $(1, 0.5, 0.2$ oraz $0.1)$. Krzywe kątów precesji $\psi$ i nutacji $\vartheta$ pokazane są na rysunku 3.34. Zmiana wspomnianego parametru $d$ wpływa na zmniejszenie wartości momentów sterujących wymaganych do wysterowania układu, jednak kosztem wydłużenia czasu na to potrzebnego. Można to wyraźnie zaobserwować na wykresach momentów sterujących $M_y$ oraz $M_z$ pokazanych na rysunku 3.35.

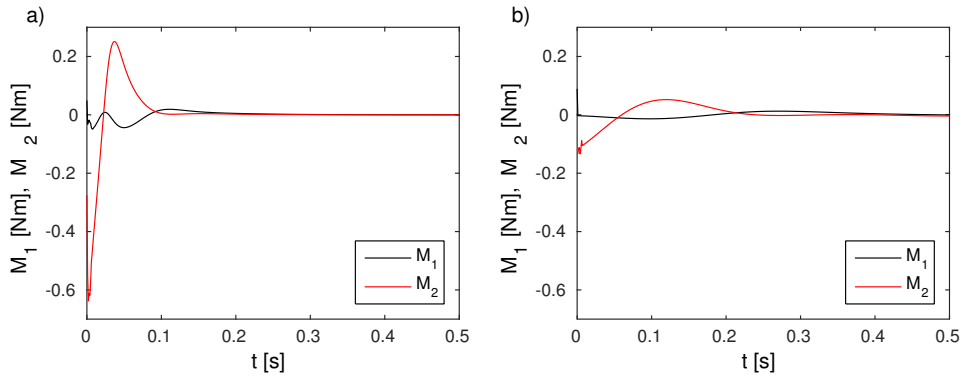
Ostatnim z rozpatrywanych przypadków dla zastosowanego sterowania typu „inverse" było sprawdzenie w jakim stopniu sieć neuronowa jest w stanie sterować kątami giroskopu przy wyższych wartościach prędkości kątowej $n$ wirnika. Ponieważ oryginalnie nauka sieci neuronowej odbywała się na bazie symulacji przy częstości kątowej wirnika $n = 10$ rad/s, a co za tym idzie przy niższej częstotliwości próbkowania, w celu rozszerzenia jej działania na wyższe częstości koniecznym było przyspieszenie działania sieci podobnie jak zostało to zaprezentowane dla przypadku belki inteligentnej.



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ



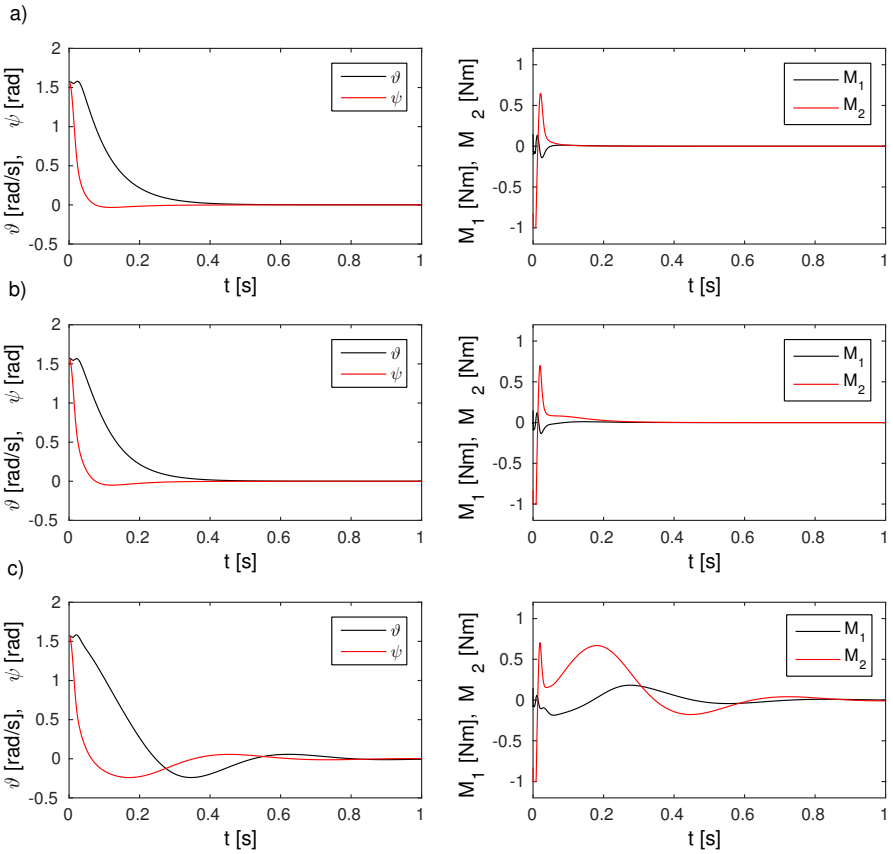

Rysunek 3.34.: Krzywe sterowania kątami precesji $\psi$ oraz nutacji $\vartheta$ giroskopu dla przypadku sterowania neuronowego typu „inverse" dla wzmocnienia $g = 0.1$ przy różnych wartościach parametru stopniowego dojścia do celu $d$




Rysunek 3.35.: Krzywe momentów sterujących $M_y$ i $M_z$ dla przypadku sterowania neuronowego typu „inverse" dla wartości wzmocnienia $g = 0.1$ z uwzględnieniem modyfikacji parametru stopniowego dojścia do celu $d$
a) $d = 1$
b) $d = 0.1$



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ




Rysunek 3.36.: Wartości położeń kątowych $\vartheta$ oraz $\psi$ giroskopu o dwóch stopniach swobody oraz momentów sterujących $M_y$ i $M_z$ dla sterowania neuronowego typu „inverse" dla różnych wartości prędkości kątowej $n$ oraz przyspieszenia działania sieci $r$:
a) $n = 10$ rad/s, $r = 1$,
b) $n = 100$ rad/s, $r = 10$,
c) $n = 1000$ rad/s, $r = 100$
Parametry sterowania: $g = 0.3$, $d = 1$

Na wykresach 3.36 pokazane zostało działanie sterownika neuronowego dla wartości prędkości kątowej $n$ wynoszącej kolejno 10, 100 oraz 1000 rad/s, oraz dla odpowiadającej im kolejno wartości przyspieszenia działania sieci $r$ równej 1, 10 oraz 100. Jak widać z wykresów, sieć była w stanie dostosować się do zwiększonej prędkości kątowej wirnika oraz pomyślnie wysterować wartościami kątów precesji $\psi$ oraz nutacji $\vartheta$ od ich wartości początkowych będących w zakresie nieliniowym, aż do zerowych wartości końcowych. Wraz ze wzrostem prędkości kątowej $n$ wydłuża się również czas potrzeby na wysterowanie układu jak również poziom wymaganych do tego momentów sterujących $M_y$ oraz $M_Z$. Wynika to jednak z zachowania dynamicznego samego giroskopu - im większa jest prędkość wirowania wirnika, tym trudniej jest sterować kątami precesji $\psi$ oraz nutacji $\vartheta$ jego ramek.



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

### Sterowanie przez emulację LQR

Drugą z metod sterowania neuronowego, która była przetestowana dla modelu giroskopu, jest podobnie jak dla przypadku belki inteligentnej, metoda sterowania poprzez emulację zachowania działającego sterownika typu LQR. Jako, że zastosowany tam zredukowany model belki wspornikowej był modelem liniowym, symulacja sterowania typu LQR przeprowadzona została w sposób bezpośredni z wykorzystaniem dostępnych macierzy stanu $\mathbf{A_m}$, $\mathbf{B_m}$ i $\mathbf{C_m}$ oraz założonych macierzy $\mathbf{Q}$ i $\mathbf{R}$. Ponieważ przyjęty model giroskopu jest modelem nieliniowym opisanym za pomocą układu równań 2.18, sterowanie typu LQR w jego przypadku możliwe jest wyłącznie w ograniczonym zakresie dla modelu zlinearyzowanego. Dynamika giroskopu po linearyzacji względem położenia zerowego obu kątów $(\vartheta, \psi)$ opisana jest za pomocą układu równań:

$$\begin{cases} M_y = I_y\ddot{\vartheta} + h\dot{\psi} \\ M_z = I_Z\ddot{\psi} - h\dot{\vartheta} \end{cases} \tag{3.31}$$

gdzie $I_Z = I_{oz} + I_z$. Układ równań 3.31 przekształcić można następnie do postaci równania stanu oraz równania wyjścia:

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \end{cases} \tag{3.32}$$

z wektorami stanu $\mathbf{x}$, wymuszenia $\mathbf{u}$ oraz wyjścia $\mathbf{y}$ w postaci:

$$\mathbf{x} = \begin{Bmatrix} \vartheta \\ \psi \\ \dot{\vartheta} \\ \dot{\psi} \end{Bmatrix} \quad \mathbf{u} = \begin{Bmatrix} M_y \\ M_z \end{Bmatrix} \quad \mathbf{y} = \begin{Bmatrix} \vartheta \\ \psi \end{Bmatrix}$$

oraz odpowiednio wyprowadzonymi macierzami stanu $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ i $\mathbf{D}$:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -h/I_y \\ 0 & 0 & -h/I_Z & 0 \end{bmatrix}_{4\times4} \quad \mathbf{B} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1/I_y & 0 \\ 0 & 1/I_Z \end{bmatrix}_{4\times2}$$

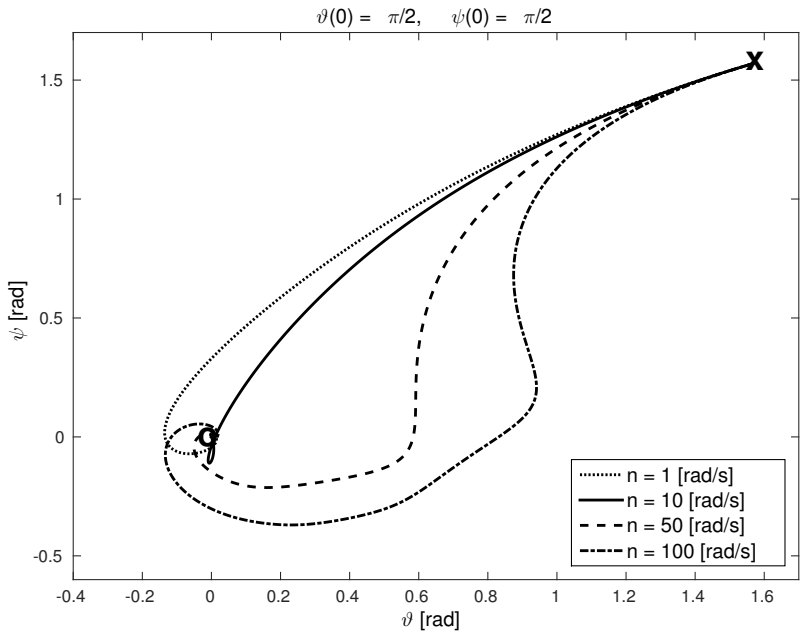
$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}_{2\times4} \quad \mathbf{D} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}_{2\times2}$$

Dla tak określonego modelu giroskopu, wyprowadzone zostało następnie sterowanie LQR (korzystając z opisanej uprzednio procedury), które z kolei posłużyło jako wzorzec późniejszej nauki emulatora neuronowego. Dla próbek uczących narzucono ograniczenie zakresu kątowego $\vartheta = [-0.3, 0.3]$ rad oraz $\psi = [-0.3, 0.3]$ rad, przy założeniu prędkości wirnika równej $n = 10$ rad/s. Na rysunku 3.37 pokazane zostały krzywe dojścia do położenia zerowego obu kątów



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

giroskopu, dla sterowania tak nauczonym emulatorem neuronowym, przy założeniu różnych wartości prędkości kątowej wirnika $n$.

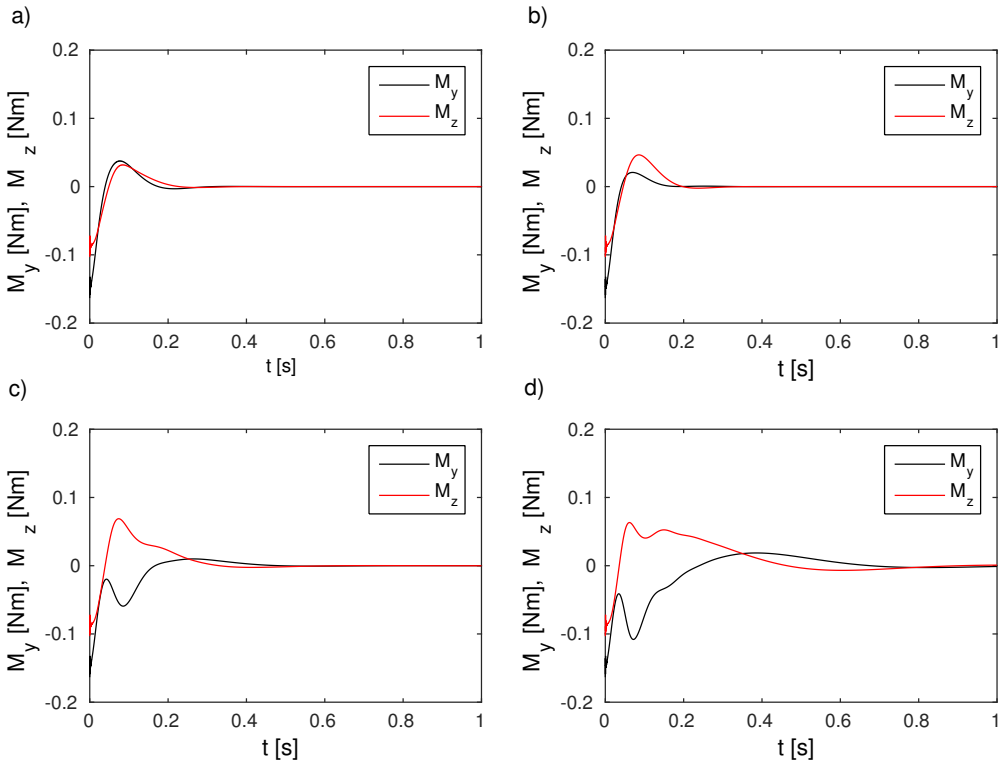



Rysunek 3.37.: Krzywe sterowania kątami precesji $\vartheta$ oraz nutacji $\psi$ giroskopu dla przypadku sterowania neuronowego za pomocą emulatora neuronowego LQR przy różnych wartościach prędkości kątowej wirnika $n$

Jak widać na rysunku, sieć jest w stanie skutecznie sterować kątami $\vartheta$ i $\phi$ oraz w skończonym czasie zapewnić przejście od położenia początkowego $[\vartheta_0, \phi_0]$ do położenia końcowego (zerowego). Pomimo faktu, że sieć neuronowa nauczona została sterowania na bazie układu giroskopu o stałej prędkości kątowej wirnika $n = 10$ rad/s, sieć jest w stanie działać poprawnie również dla innych prędkości wirowania $n$. Podobnie jak dla sterowania typu „inverse" odbywa się to jednak kosztem dłuższego czasu potrzebnego na wysterowanie układu oraz wyższych wartości momentów sterujących $M_y$ i $M_z$. Na rysunku 3.38 pokazane zostały wykresy momentów sterujących dla poszczególnych krzywych z rysunku 3.37. Warto przy tej okazji podkreślić, że uczenie sieci odbywało się na bazie próbek wyłącznie ze zlinearyzowanego zakresu, właściwe sterowanie neuronowe wybiega natomiast znacznie poza ten zakres.



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ




Rysunek 3.38.: Wykresy momentów sterujących $M_y$ oraz $M_z$ dla przypadku sterowania neuronowego położeniem kątowym giroskopu za pomocą emulatora neuronowego LQR przy różnych wartościach prędkości kątowej wirnika $n$:
a) $n = 1$ rad/s,
b) $n = 10$ rad/s,
c) $n = 50$ rad/s,
d) $n = 100$ rad/s



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

## 3.3. Zagadnienia uzupełniające

### 3.3.1. Realizacja praktyczna

Jak zostało to uprzednio zaznaczone w rozdziale stanowiącym wprowadzenie do niniejszej pracy, jej zasadniczym celem nie było wykonywanie pomiarów na rzeczywistych obiektach, tudzież konstrukcja fizycznego układu sterowania, za cel ten postawiono natomiast przetestowanie w sposób symulacyjny szeregu metod sterowania pod kątem perspektywy zastosowania ich na użytek aktywnego tłumienia drgań układów ciągłych oraz dyskretnych.

Jednakże, niewłaściwym byłoby przedstawienie wspomnianego zagadnienia wyłącznie z perspektywy numerycznej oraz symulacyjnej, bez zaznaczenia chociażby w pewnym stopniu możliwości dalszej realizacji praktycznej oraz bez odwołania się do już istniejących układów realizujących podobnego typu sterowanie. Jest to kluczowe, zważywszy na fakt, iż w przeciągu ostatnich piętnastu lat zainteresowanie zagadnieniem sztucznych sieci neuronowych znacznie wzrosło, a co za tym idzie pojawiła się znaczna liczba publikacji dotyczących praktycznych zastosowań sterowania neuronowego.

Jednym z ciekawych praktycznych zastosowań sterowania neuronowego, przynajmniej z perspektywy niniejszej pracy, jest przykład sterowania położeniem kątowym manipulatora o jednym stopniu swobody zaprezentowany w publikacji [55]. Przykład ten jest o tyle ciekawy, że zaprezentowany w publikacji układ manipulatora odpowiada w gruncie rzeczy modelowi wahadła nieliniowego opisywanemu szczegółowo w podrozdziałach 2.4.1 oraz 3.2.1. Ponadto do sterowania neuronowego położeniem kątowym manipulatora zastosowano zmodyfikowany algorytm typu „NARMA-L2" [56], natomiast samo uczenie sieci neuronowej odbywało się z wykorzystaniem algorytmu optymalizacji Lavenberga-Margquardta.

Najważniejszym wnioskiem jaki może zostać wyciągnięty z pracy [55], jest stwierdzenie, że sztuczna sieć neuronowa (w tym przypadku sieć typu „NARMA-L2") jest w stanie sterować układem dynamicznym takim jak wspomniane wahadło bądź manipulator o jednym stopniu swobody, również w zakresie nieliniowym. Na wykresach rzeczywistego sterowania zaprezentowanych we wspomnianej pracy, widać stopniową zmianę położenia kątowego manipulatora od położenia początkowego (dolnego) do położenia końcowego (górnego). Niestety na wykresach tych widać również wyraźnie tak zwany efekt „chatteringu" czyli oscylacyjnej zmiany położenia w otoczeniu docelowego położenia końcowego. Być może zmiana architektury bądź parametrów uczenia sieci neuronowej albo rzeczywista implementacja niektórych z modyfikacji zaprezentowanych w niniejszej pracy doktorskiej (wolnego dojścia do celu, ograniczonego wzmocnienia) umożliwiłyby zniwelowanie wspomnianego efektu, a co za tym idzie bardziej płynne i dokładniejsze sterowanie.

Niestety w pracy [55], pominięte zostały wykresy obrazujące wartości sygnału sterującego (napięcia, momentu sterującego), brak również jakiejkolwiek informacji na temat parametrów sterowanego układu (masy, długości) oraz własności samej sieci neuronowej (liczby neuronów w warstwach sieci, liczby regresorów). Podobnie, pomimo faktu iż rozpatrywany układ manipulatora nie jest układem złożonym, w pracy nie zawarte zostały żadne równania, które



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

opisywałyby jego dynamikę. Nie jest przez to możliwe odtworzenie w sposób symulacyjny rezultatów otrzymanych w trakcie przeprowadzonych doświadczeń, co najwyżej możliwym jest porównanie jakościowe zachowania układów o porównywalnej strukturze i zachowaniach dynamicznych. Należy jednak zaznaczyć, iż pominięcie równań dynamiki układu w pracy dotyczącej praktycznego zastosowania sztucznych sieci neuronowych w sterowaniu układami dynamicznymi, nie jest odosobnionym przypadkiem. Jest to najprawdopodobniej wynikiem własności „black-box" samych sztucznych sieci neuronowych, na potrzeby uczenia których nie jest wymaganym znajomość nierzadko stosunkowo złożonej dynamiki symulowanego bądź sterowanego układu, wystarczająca jest wyłącznie znajomość wygenerowanego sygnału wymuszenia oraz otrzymanych w bezpośrednim wyniku eksperymentu próbek uczących opisujących stan i zachowanie układu.

Układy eksperymentalne bardzo podobne do wymienionego wyżej manipulatora o jednym stopniu swobody opisane zostały również w pracy [57]. W pierwszym z przeprowadzonych tam eksperymentów, sterowanie neuronowe stosowane jest do regulacji położenia układu połączonych dysków, w drugim podobnego typu sterowanie wykorzystane jest na użytek zmiany położenia dolnego ramienia robota CRS A55. Podobnie jak i w poprzedniej pracy, w obu przypadkach sterowanie układami odbywa się z wykorzystaniem metody typu „NARMA-L2", natomiast algorytmem służący do minimalizacji błędu średnio-kwadratowego jest algorytm Lavenberga-Margquardta. Niestety, pomimo tego, że w pracy opisane są dokładnie parametry i architektura zastosowanej sieci, próbki uczące pobrane są bezpośrednio z rzeczywistych fizycznych modeli, natomiast równanie dynamiki podane jest wyłącznie dla układu dysków, na dodatek wyłącznie w postaci ogólnej bez zdefiniowania obecnej w nim funkcji $f(\dot{y}(t))$, ani bez informacji na temat wartości wielkości w nim występujących (masy $m$, długości $l$). Co za tym idzie, podobnie jak poprzednio, bez znajomości tych informacji nie jest możliwym przeprowadzenie porównawczej symulacji numerycznej sterowania dla zaprezentowanych układów.

Pomimo tego, warto zwrócić uwagę na fakt, iż coraz więcej publikacji dotyczących zagadnienia sterowania z wykorzystaniem sztucznych sieci neuronowych, pokazuje podobnie jak dwie wymienione wyżej publikacje sposób praktycznego ich zastosowania do rzeczywistych istniejących obiektów, przykładowo sterowania położeniem autonomicznego drona z wykorzystaniem metody typu „Inverse" [58], aktywnej redukcji drgań płyt inteligentnych [59] oraz wielu innych przykładów zarówno regulacji jak i sterowania układami dynamicznymi [60], [61], [62], [63], [64].

### 3.3.2. Analiza stabilności

Ważnym zagadnieniem, które musi zostać koniecznie poruszone w przypadku jakiegokolwiek sterowania, nie tylko sterowania neuronowego jest zagadnienie stabilności układu sterowanego. Jest ono o tylko kluczowe, że determinuje czy dany układ sterowania jest w praktyce realizowalny, a jeżeli tak to jakie warunki muszą zostać spełnione aby tak było. Dla układów liniowych bądź zlinearyzowanych zagadnienie to jest stosunkowo dobrze znane i możliwym jest określenie stabilności danego układu korzystając z szeregu standardowych metod (Hurwitza,



BPK BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

Nyquista) [41], [65], [66]. Podobnie dla układów nieliniowych istnieją metody umożliwiające zbadanie stabilności m.in. metoda bezpośrednia Lyapunowa [39], [65], [67], [68].

Dla przypadku sterowania neuronowego zagadnienie stabilności staje się dodatkowo bardziej złożone. Wymienić można kilka powodów takiego stanu rzeczy. Po pierwsze - jak to już uprzednio wspomniano, sieć neuronowa w gruncie rzeczy bazuje na zasadzie „czarnej skrzynki" („black-box"), a zatem określenie stabilności na bazie macierzy wag oraz często nieliniowych funkcji aktywacji obecnych wewnątrz struktury analizowanego sterownika neuronowego stanowi skomplikowane zadanie. Po drugie - sam obiekt, który poddawany jest sterowaniu może być obiektem wykazującym zachowania nieliniowe, a co za tym idzie złożoność analizy dotyczącej sterowania takim obiektem dodatkowo wzrasta. Wreszcie, jak zostało to zaznaczone wcześniej w szczególności w rozdziale poświęconym sterowaniu, poprawne działanie sterownika neuronowego uzależnione jest od szeregu różnego rodzaju parametrów i zmiennych poczynając od wybranej architektury sieci, poprzez zastosowanie dodatkowego wzmocnienia (bądź atenuacji), na odpowiednim doborze próbek uczących kończąc. Każdy z tych parametrów może w ostatecznym rozrachunku w mniejszym bądź większym stopniu determinować czy układ będzie stabilny czy też nie.

Podobne rozważania na temat stabilności dla zagadnienia sterowania neuronowego przedstawione zostały w pracy [10]. W pracy tej podkreślono, że o ile stabilność rozumiana w sposób asymptotyczny jest kluczowa z perspektywy sterowania układem dynamicznym, o tyle niekiedy (w tym dla sterowania neuronowego) niezbędnym jest zapewnienie stabilności BIBO („bounded-in bounded-out"), co oznacza że w przypadku ograniczenie narzucone na wartości wejściowe układu implikuje ograniczenie również wartości wyjściowych. Właściwa definicja stabilności BIBO stwierdza, że rozwiązanie $y(t) = 0$ jest stabilne w sensie BIBO wtedy i tylko wtedy gdy istnieją stałe $c_u$ oraz $c_y$ spełniające implikację:

$$||u(t)|| < c_u \Rightarrow ||y(t)|| < c_y$$

gdzie $u(t)$ oraz $y(t)$ są odpowiednio wielkością wejściową i wyjściową układu.

Bazując na przedstawionej powyżej definicji, w pracy tej podkreślono, że dla przypadku gdy wszystkie z funkcji aktywacji obecnych w sieci neuronowej są ograniczone (tangensoidalne, sigmoidalne), bądź gdy nie istnieje takie połączenie pomiędzy wejściem $u(t)$ i wyjściem $y(t)$ układu które zawierałoby tylko i wyłącznie funkcje nieograniczone (liniowe), wówczas sieć neuronowa, a co za tym idzie sterowanie neuronowe będzie stabilne w sensie BIBO.

Dodatkowo w pracy tej zaznaczono, że o ile w przypadku układów stacjonarnych liniowych zarówno stabilność asymptotyczna oraz BIBO są osiągalne w przypadku gdy wartości własne układu zawierają się w obszarze stabilności (okręgu jednostkowym w przestrzeni $z$), natomiast osiągnięcie jej dla układów niestacjonarnych nieliniowych stanowi bardziej złożony problem, zależny od wielu czynników i parametrach. W niektórych przypadkach przy założeniu, że zmiana układu następuje w sposób powolny możliwe jest określenie obszaru stabilności układu bazując na wartościach własnych wyznaczonych dla układu po linearyzacji.



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

# 4. Podsumowanie i wnioski

W pracy przedstawione zostały podstawowe zagadnienia dotyczące oceny możliwości zastosowania algorytmów sterowania bazujących na sztucznych sieciach neuronowych do tłumienia drgań układów dyskretnych i ciągłych, zarówno w zakresie liniowym jak i nieliniowym. Charakter pracy był w głównej mierze obliczeniowy i symulacyjny, z modelami dobranymi w taki sposób by uwypuklić możliwe zalety oraz wady sterowania z wykorzystaniem sztucznych sieci neuronowych, jak również możliwe trudności związane z procesem ich uczenia.

W rozdziale poświęconym identyfikacji neuronowej opisane zostały kolejno architektura oraz modele dynamicznych sztucznych sieci neuronowych (DSSN), wybrane modele emulatorów neuronowych oraz algorytmy uczenia sieci neuronowych z dodatkowym wyprowadzeniem zasady uczenia dla algorytmu gradientowego największego spadku. W rozdziale tym zaprezentowano ponadto emulację neuronową na przykładzie wybranych modeli dynamicznych: wahadła matematycznego, giroskopu oraz belki inteligentnej z elementami piezoelektryczymi.

Główna część pracy poświęcona została sterowaniu neuronowemu, w szczególności pod kątem zastosowania go do aktywnego tłumienia drgań układów oscylacyjych. Opierając się na literaturze dotyczącej zagadnienia sztucznych sieci neuronowych jak również informacji zawartej w rozdziale poświęconym identyfikacji neuronowej, zaproponowano oraz przedstawiono szereg metod sterowania neuronowego m.in. metody sterowania przez inwersję oraz metody typu „NARMA-L2'. Metody te oraz skuteczność ich działania przetestowana została na wybranych modelach układów dynamicznych. Wśród nich znalazły się między innymi układ wahadła podwójnego nieliniowego, giroskopu oraz belki inteligentnej.

Ponadto w pracy przetestowano w jaki sposób dobór odpowiedniej architektury sieci, próbek uczących oraz algorytmu uczenia wpływa na efektywność i szybkość działania sterownika neuronowego. W pracy zaproponowano również szereg oryginalnych modyfikacji oraz zmian do układu sterowania, które umożliwiły poprawę skuteczności sterowania bez konieczności ingerencji w strukturę czy macierze wag nauczonej sieci.

Na bazie obliczeń oraz symulacji przeprowadzonych na potrzeby prezentowanej pracy można wyciągnąć następujące wnioski:

- dynamiczne sztuczne sieci neuronowe są w stanie w sposób skuteczny emulować zachowanie oraz dynamikę układów mechaniczych pod warunkiem przeprowadzenia poprawnego procesu ich uczenia; zakres działania sieci ograniczony jest odgórnie przez zakres próbek przyjętych do uczenia sieci

- podstawowa zaleta sterowania neuronowego, niezależnie od zastosowanej metody bądź



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

algorytmu, przejawia się w możliwości nauczenia sieci sterowania nie tylko w zakresie liniowym bądź dla modelu zlinearyzowanego, lecz również w zakresie nieliniowym

- zastosowanie sztucznej sieci neuronowej w roli emulatora bądź sterownika układu dynamicznego wymusza na użytkowniku konieczność zdefiniowania szeregu kluczowych parametrów opisujących sieć m.in. architektury sieci, ilości regresorów oraz ilości neuronów w poszczególnych jej warstwach; podobna procedura dotycząca doboru parametrów musi również zostać przeprowadzona dla próbek uczących, w celu zdefiniowania zakresu uczenia oraz częstotliwości próbkowania, a co za tym idzie częstotliwości działania samego sterownika

- sterownik neuronowy, który nauczony został na bazie próbek pobranych z układu dynamicznego o konkretnych ściśle zadanych parametrach obiektu, wykazuje do pewnego stopnia odporność na zmianę tychże parametrów w trakcie jego pracy; sieć taka jest w dalszym ciągu w stanie tłumić drgania rozpatrywanego układu, związane jest to jednak ze zmniejszoną efektywnością jej działania

- częściową zaletą zastosowania sztucznych sieci neuronowych jest możliwość całkowitego pominięcia procesu przygotowania modelu matematycznego sterowanego układu pod warunkiem, że układ taki jest fizycznie dostępny; jako że sieć neuronowa działa na zasadzie „czarnej skrzynki" („black-box") do nauczenia sieci wymagane jest wtedy jedynie zdobycie odpowiedniej ilości próbek uczących sygnału sterującego oraz wielkości sterowanej: pomimo faktu, iż w pracy skorzystano z próbek testowych uzyskanych z modeli matematycznych rozpatrywanych układów, próbki te można by równie dobrze uzyskać z eksperymentu przeprowadzonego na rzeczywistym obiekcie dynamicznym

- z uwagi na złożoną budowę sztucznych sieci neuronowych oraz wymienione wyżej uczenie na zasadzie „czarnej skrzynki", układ sterowania neuronowego różni się znacznie od bazującego na modelu matematycznym układu sterowania klasycznego; w wyniku tego analiza wpływu oraz zachowania algorytmów sterowania neuronowego (przykładowo badanie stabilności układu) jest znacznie bardziej skomplikowana, jako że w tym wypadku nie jest możliwym skorzystanie ze standardowo stosowanych do tego celu metod

Podsumowując przedstawione powyżej wnioski można zatem stwierdzić, że tłumienie drgań układów dynamicznych przy wykorzystaniu algorytmów sterowania neuronowego jest możliwe, co więcej nie ogranicza się ono wyłącznie do zakresu liniowego, lecz przy odpowiednim doborze architektury sieci oraz próbek uczących obejmować ono może również zakres nieliniowy. Realizacja tak działającego układu sterowania wymaga jednakże znacznie większego nakładu pracy oraz przygotowań, niż dla sterowania klasycznego.

Z uwagi na ograniczony zakres pracy przy jednocześnie obszernym zakresie zagadnienia sztucznych sieci neuronowych oraz stosunkowo nowoczesnej problematyce zastosowania ich do sterowania zachowaniem układów dynamicznych, nie było niestety wykonalne przeanalizowanie wszystkich możliwych opcji i rozwiązań. W dalszej perspektywie planowane jest



BPK BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

jednakże przetestowanie zaproponowanych w pracy algorytmów neuronowych na rzeczywistym modelach dynamicznych z wykorzystaniem właściwych układów sterowania.

Ponadto rozważane jest przez autora ewentualne rozszerzenie analizy oraz badań dotyczących aktywnego neuronowego tłumienia drgań na bardziej złożone układy oraz algorytmy sterowania m.in. sterowniki neuronowe z ciągłym douczaniem (w trybie „on-line"), sieci rekurencyjne oraz sterowanie neuronowo-rozmyte.



BPK BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

# A. Dodatek

## Algorytm gradientowy największego spadku

W algorytmie największego spadku („steepest descent"), równanie błędu średniokwadratowego $\hat{E}$, definuje się dla ostatniego wzorca uczącego [12]:

$$\hat{E} = \mathbf{e}(k)^T \mathbf{e}(k) = [\mathbf{y}(k) - \hat{\mathbf{y}}(k)]^T [\mathbf{y}(k) - \hat{\mathbf{y}}(k)] \tag{A.1}$$

gdzie $k$ odpowiada ostatniemu krokowi iteracji algorytmu.

Elementy macierzy wag $\mathbf{W}$ oraz wektorów odchylenia $\mathbf{b}$ dla iteracji $(k+1)$ modyfikowane są na bazie ich wartości z iteracji poprzedniej $(k)$ zgodnie z wzorami:

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha G(w_{i,j}^m) = w_{i,j}^m(k) - \alpha \frac{\partial \hat{E}}{\partial w_{i,j}^m} \tag{A.2}$$

$$b_i^m(k+1) = b_i^m(k) - \alpha G(b_i^m) = b_i^m(k) - \alpha \frac{\partial \hat{E}}{\partial b_i^m} \tag{A.3}$$

w których $G(w_{i,j}^m)$ oraz $G(b_i^m)$ są gradientem zmiany odpowiednio dla elementu $(i,j)$ $m$-tej macierzy wag $W$ oraz elementu $i$ $m$-tego wektora odchyleń $b$, natomiast $\alpha$ jest krokiem uczenia (zmiany) algorytmu. Zbieżność rozwiązania dla podanego algorytmu zależna jest w głównej mierze od doboru kroku uczenia $\alpha$, przy czym zaznaczyć należy, iż parametr ten w trakcie procesu uczenia pozostaje niezmienny.

Wartości gradientu z równań A.2 i A.3, rozłożyć można względem i-tej wartości wektora wyjściowego $\varphi$ $m$-tej warstwy sieci, na iloraz dwóch pochodnych cząstkowych:

$$\frac{\partial \hat{E}}{\partial w_{i,j}^m} = \frac{\partial \hat{E}}{\partial \varphi_i^m} \times \frac{\partial \varphi_i^m}{\partial w_{i,j}^m} \tag{A.4}$$

$$\frac{\partial \hat{E}}{\partial b_i^m} = \frac{\partial \hat{E}}{\partial \varphi_i^m} \times \frac{\partial \varphi_i^m}{\partial b_i^m} \tag{A.5}$$

aby następnie korzystając z wartości $\varphi_i^m$ zdefiniowanej w postaci:

$$\varphi_i^m = \sum_{j=1}^{q} w_{i,j}^m \hat{y}_j^{m-1} + b_i^m \tag{A.6}$$

otrzymać w sposób bezpośredni wartości pochodnych cząstkowych:



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

$$\frac{\partial \varphi_i^m}{\partial w_{i,j}^m} = \hat{y}_j^{m-1}, \quad \frac{\partial \varphi_i^m}{\partial b_i^m} = 1 \tag{A.7}$$

Definiując następnie czułość $s_i^m$ jako:

$$s_i^m = \frac{\partial \hat{E}}{\partial \varphi_i^m} \tag{A.8}$$

pochodne cząstkowe A.4 oraz A.5 przekształcą się do postaci:

$$\frac{\partial \hat{E}}{\partial w_{i,j}^m} = s_i^m \hat{y}_j^{m-1} \tag{A.9}$$

$$\frac{\partial \hat{E}}{\partial b_i^m} = s_i^m \tag{A.10}$$

Podstawiając A.9 oraz A.10 odpowiednio do równań A.2 i A.3 otrzymamy ostatecznie zdefiniowane równania zmiany odpowiednio dla elementu $(i, j)$ $m$-tej macierzy wag $W$ oraz elementu $i$ $m$-tego wektora odchyleń $b$

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha s_i^m \hat{y}_j^{m-1} \tag{A.11}$$

$$b_i^m(k+1) = b_i^m(k) - \alpha s_i^m \tag{A.12}$$

Powyższe równania zapisane w postaci macierzowej prezentują się następująco:

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha \mathbf{s}^m (\hat{\mathbf{y}}^{m-1})^T \tag{A.13}$$

$$\mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha \mathbf{s}^m \tag{A.14}$$

Brakującym i niezbędnym elementem w przedstawionych powyżej równaniach, jest wektor czułości $m$-tej warstwy $\mathbf{s}^m$, który koniecznie musi zostać wyznaczony dla każdej z $M$ warstw sieci. Wektory czułości otrzymywane są na zasadzie indukcji - w pierwszej kolejności wyznacza się wektor $\mathbf{s}^M$ dla wyjściowej $M$-tej warstwy, a następnie korzystając z niego wyznacza się wektor $\mathbf{s}^{M-1}$ dla warstwy poprzedniej i tak kolejno aż do wektora $\mathbf{s}^1$.

Korzystając z podanej wyżej definicji czułości A.8, $i$-tą czułość $M$-tej (czyli ostatniej) warstwy sieci rozbić można na dwie pochodne cząstkowe w następujący sposób:

$$s_i^M = \frac{\partial \hat{E}}{\partial \varphi_i^M} = \frac{\partial \hat{E}}{\partial \hat{y}_i^M} \times \frac{\partial \hat{y}_i^M}{\partial \varphi_i^M} \tag{A.15}$$

Podane pochodne cząstkowe wyznaczyć można dla M-tej warstwy w sposób bezpośredni, wykorzystując przedstawione uprzednio zależności A.1 oraz 2.1



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

$$\frac{\partial \hat{E}}{\partial \hat{y}_i^M} = -2(y_i - \hat{y}_i) \tag{A.16}$$

$$\frac{\partial \hat{y}_i^M}{\partial \varphi_i^M} = f'_M(\varphi_i^M) \tag{A.17}$$

Podstawiając A.16 oraz A.17 do A.15 otrzymujemy:

$$s_i^M = -2f'_M(\varphi_i^M)(y_i - \hat{y}_i) \tag{A.18}$$

Podobnie sprawa wygląda w przypadku $j$-tej wartości dla warstwy $M-1$, którą można rozbić na cztery pochodne cząstkowe, w następujący sposób:

$$s_j^{M-1} = \frac{\partial \hat{E}}{\partial \hat{y}_i^M} \times \frac{\partial \hat{y}_i^M}{\partial \varphi_i^M} \times \frac{\partial \varphi_i^M}{\partial \hat{y}_j^{M-1}} \times \frac{\partial \hat{y}_j^{M-1}}{\partial \varphi_j^{M-1}} \tag{A.19}$$

dla której:

$$\frac{\partial \varphi_i^M}{\partial \hat{y}_j^{M-1}} = w_{i,j}^M \tag{A.20}$$

$$\frac{\partial \hat{y}_j^{M-1}}{\partial \varphi_j^{M-1}} = f'_{M-1}(\varphi_j^{M-1}) \tag{A.21}$$

Podstawiając następnie A.20, A.21 oraz A.18 do zależności A.19, otrzymujemy:

$$s_j^{M-1} = s_i^M w_{i,j}^M f'_{M-1}(\varphi_j^{M-1}) \tag{A.22}$$

którą następnie można rozszerzyć do bardziej ogólnej postaci, obejmującej wszystkie oprócz ostatniej $M$-tej warstwy:

$$s_j^m = s_i^{m+1} w_{i,j}^{m+1} f'_m(\varphi_j^{M-1}), \quad m = M-1, \cdots, 1 \tag{A.23}$$

Następnie, dokonując złożenia rozwiązania A.18 i A.23 po wszystkich wartościach, otrzymujemy odpowiednio wektor czułości dla warstwy $M$-tej, oraz dla wszystkich pozostałych warstw sieci:

$$\mathbf{s}^M = -2\mathbf{F'}_M(\varphi^M)(\mathbf{y} - \hat{\mathbf{y}}) \tag{A.24}$$

$$(\mathbf{s}^m)^T = (\mathbf{s}^{m+1})^T \mathbf{W}^{m+1} \mathbf{F'}_m(\varphi^m), \quad m = M-1, \cdots, 1 \tag{A.25}$$

gdzie macierz $\mathbf{F'}_m(\varphi^m)$ zawierającą pochodne wszystkich funkcji aktywacji $m$-tej warstwy, zdefiniowana jest w postaci diagonalnej:



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

$$\mathbf{F'}_m(\varphi^m) = \begin{bmatrix} f'_m(\varphi_1^m) & 0 & \dots & 0 \\ 0 & f'_m(\varphi_2^m) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & f'_m(\varphi_q^m) \end{bmatrix} \tag{A.26}$$

Pamiętając o tym, że dla macierzy diagonalnej $\mathbf{F'}_M(\varphi^M)$ prawdziwa jest zależność:

$$(\mathbf{F'}_M(\varphi^M))^T = \mathbf{F'}_M(\varphi^M) \tag{A.27}$$

wektor (A.25) przekształcić można do postaci podobnej do wektora (A.24):

$$\mathbf{s}^m = (\mathbf{W}^{m+1})^T \mathbf{F'}_m(\varphi^m)\mathbf{s}^{m+1}, \quad m = M-1, \cdots, 1 \tag{A.28}$$

Warto przy tej okazji wspomnieć, że odpowiedni dobór funkcji aktywacji $f(\varphi)$, umożliwia w prosty sposób wyznaczenie ich pochodnych $f'(\varphi)$, w celu ich zastosowania w podanych powyżej wzorach dla wektorów czułości $\mathbf{s}^m$. Dla funkcji aktywacji

$$f_1(\varphi) = \varphi \tag{A.29}$$

$$f_2(\varphi) = \frac{1}{1+e^{\varphi}} \tag{A.30}$$

$$f_3(\varphi) = \mathrm{tgh}(\varphi) = \frac{e^{2\varphi}-1}{e^{2\varphi}+1} \tag{A.31}$$

pochodne otrzymuje się odpowiednio w formie:

$$f'_1(\varphi) = 1 \tag{A.32}$$

$$f'_2(\varphi) = \frac{-e^{\varphi}}{(1+e^{\varphi})^2} \tag{A.33}$$

$$f'_3(\varphi) = \frac{4e^{2\varphi}}{(1+e^{2\varphi})^2} \tag{A.34}$$

Ostatnią z wymaganych informacji niezbędnych przy implementacji podanego algorytmu, jest zdefiniowanie warunków przy których algorytm powinien się zatrzymać. Najczęściej stosowanymi kryteriami stopu dla algorytmów uczenia sieci neuronowych są:

- warunek maksymalnej liczby epok uczących
- warunek osiągnięcia założonej wartości błędu $\hat{E} < E_{min}$
- warunek stopnia przeuczenia (wyznaczony korzystając z próbek walidacyjnych)



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

# Literatura


[1] J.M. Benyus. *Biomimicry: Innovation Inspired by Nature.* HarperCollins, 1997.

[2] Y. Bar-Cohen. *Biomimetics: Nature-Based Innovation.* Biomimetics series. CRC Press, 2011.

[3] K. M. Passino. *Biomimicry for Optimization, Control, and Automation.* Springer London, 2005.

[4] W. S. McCulloch, Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

[5] D. O. Hebb. *The Organization of Behavior.* Wiley, New York, 1949.

[6] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386, 1958.

[7] M. Minsky, S. Papert. *Perceptrons : An Introduction to Computational Geometry.* The MIT Press, Cambridge (Mass.), London, 1969.

[8] M. Flasiński. *Wstęp do sztucznej inteligencji.* Wydawnictwo Naukowe PWN, 2011.

[9] L. Rutkowski. *Metody i techniki sztucznej inteligencji: inteligencja obliczeniowa.* Wydawnictwo Naukowe PWN, 2005.

[10] M. Nørgaard, M. Ravn, N. K. Poulsen, L. K. Hansen. *Neural Networks for Modelling and Control of Dynamical Systems.* Springer, 2003.

[11] O. Omidvar, D. L. Elliot, redaktorzy. *Neural Systems for Control.* Academic Press, 1997.

[12] M. T. Hagan, H. B. Demuth. Neural networks for control. In *Proceedings of the 1999 American Control Conference*, pages 1642–1656, San Diego, 1999.

[13] K .S. Narendra, K. Parthsarathy. Identification and control of dynamical system using neural networks. *IEENN*, 1(1):4–27, 1990.

[14] S. Omatu, M. Yoshioka. Self-tuning neuro-PID control and applications. In *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, volume 3, pages 1985–1989. IEEE, 1997.




BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

[15] S. Omatu, T. Fujinaka, M. Yoshioka. Neuro-PID control for inverted single and double pendulums. In *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, volume 4, pages 2685–2690. IEEE, 2000.

[16] T.U. D. C. Thanh, K. K. Ahn. Nonlinear PID control to improve the control performance of 2 axes pneumatic artificial muscle manipulator using neural network. *Mechatronics*, 16(9):577–587, 2006.

[17] J. Łęski. *Systemy neuronowo-rozmyte*. Wydawnictwa Naukowo-Techniczne, 2008.

[18] D. Nauck, F. Klawonn, R. Kruse. *Foundations of Neuro-Fuzzy Systems*. John Wiley & Sons, Inc., 1997.

[19] P. Koehn. *Combining Genetic Algorithms and Neural Networks: The Encoding Problem*. Citeseer, 1994.

[20] R. De Nardi, J. Togelius, O. E. Holland, S. M. Lucas. Evolution of neural networks for helicopter control: Why modularity matters. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 1799–1806. IEEE, 2006.

[21] S.-B. Choi. *Vibration Control*. John Wiley & Sons, Inc., 2002.

[22] E. F. Crawley, J. De Luis. Use of piezoelectric actuators as elements of intelligent structures. *AIAA Journal*, 25(10):1373–1385, 1987.

[23] E. F. Crawley, K. B. Lazarus. Induced strain actuation of isotropic and anisotropic plates. *AIAA Journal*, 29(6):944–951, 1991.

[24] E. K. Dimitriadis, C. R. Fuller, C. A. Rogers. Piezoelectric actuators for distributed vibration excitation of thin plates. *Journal of Vibration and Acoustics*, 113(1):100–107, 1991.

[25] C. R. Fuller. *Sound Control with Smart Skins*. John Wiley & Sons, Inc., 2002.

[26] S. Hurlebaus, L. Gaul. Smart structure dynamics. *Mechanical Systems and Signal Processing*, 20(2):255–281, 2006.

[27] A. Preumont. *Vibration Control of Active Structures*. Kluwer Acedemic Publisher, 1997.

[28] C.-K. Lee, F. C. Moon. Modal sensors/actuators. *Journal of Applied Mechanics*, 57(2):434–441, 1990.

[29] A. Tylikowski. Stabilization of beam parametric vibrations. *Journal of Theoretical and Applied Mechanics*, 31(3):657–670, 1993.

[30] V. Giurgiutiu. *Structural Health Monitoring with Piezoelectric Wafer Active Sensors*. Academic Press, 2007.



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

[31] C.-K. Soh, Y. Yang, S. Bhalla. *Smart Materials in Structural Health Monitoring, Control and Biomechanics*. Springer Science & Business Media, 2012.

[32] S.O. R. Moheimani, A. J Fleming. *Piezoelectric Transducers for Vibration Control and Damping*. Springer Science & Business Media, 2006.

[33] M. R. Kermani, M. Moallem, R. V. Patel. *Applied Vibration Suppression Using Piezoelectric Materials*. Nova Publishers, 2008.

[34] S.-B. Choi, Y.-M. Han. *Piezoelectric Actuators: Control Applications of Smart Materials*. CRC Press, 2010.

[35] A. Erturk, D. J. Inman. *Piezoelectric Energy Harvesting*. John Wiley & Sons, 2011.

[36] D. W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[37] J. Korbicz, A. Obuchowicz, D. Uciński. *Sztuczne sieci neuronowe*. Akademicka Oficyna Wydawnicza PLJ, 1994.

[38] J. Żurada, M. Barski, Jędruch W. *Sztuczne sieci neuronowe. Podstawy teorii i zastosowania*. Wydawnictwo PWN, Warszawa, 1996.

[39] J. Nizioł. *Podstawy drgań w maszynach*. Politechnika Krakowska, Kraków, 1989.

[40] Ł. Łacny. Modelling of the dynamics of a gyroscope using artificial neural networks. *Journal of Theoretical and Applied Mechanics*, 50(3):85–97, 2012.

[41] R. H. Cannon jr. *Dynamika układów fizycznych*. WNT, 1973.

[42] Z. Koruba. *Dynamika i sterowanie giroskopem na pokładzie obiektu latającego*. Wydawnictwo Politechniki Świętokrzyskiej, Kielce, 2001.

[43] Z. Koruba. *Elementy teorii i zastosowań giroskopu sterowanego*. Wydawnictwo Politechniki Świętokrzyskiej, Kielce, 2001.

[44] J. Nizioł, redaktor. *Dynamika układów mechanicznych*, volume II. IPPT, Warszawa, 2005.

[45] Z. Koruba, J. Osiecki. Redukcja kątowych drgań obiektu za pomocą giroskopu o trzech stopniach swobody. In *III Szkoła Metody Aktywne Redukcji Drgań i Hałasu*, pages 121–127, 1997.

[46] E. Crawley, J.D. Luis. Use of piezoelectric acutator as elements of intelligent structures. *AIAA Journal*, 25:1373–1385, 1987.

[47] P. Cupiał. *Coupled Electromechanical Vibration Problems for Piezoelectric Distributed-Parameter Systems*. Politechnika Krakowska, Kraków, 2008.



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

[48] L. Meirovitch. *Elements of Vibration Analysis.* McGraw-Hill, New York, 2nd edition, 1986.

[49] C. P. Smyser, K. Chandrashekhara. Robust vibration control of composite beams using piezoelectric devices and neural networks. *Smart Materials and Structures*, 6(2), 1997.

[50] M. T. Valoor, K. Chandrashekhara, S. Agarwal. Self-adaptive vibration control of smart composite beams using recurrent neural architecture. *International Journal of Solid and Structures*, 38:7857–7874, 2001.

[51] Z. Qiu, X. Zhang, C. Ye. Vibration suppresion of a flexible piezoelectric beam using bp neural network controller. *Acta Mechanika Solida Sinica*, 25:417–428, 2012.

[52] J. Žilková, J. Timko, P. Girovský. Nonlinear system control using neural networks. *Acta Polytechnica Hungarica*, 3(4):85–94, 2006.

[53] *Mathworks MATLAB Neural Network ToolBox Documentation.* http://uk.mathworks.com/help/nnet/index.html.

[54] P. Cupiał, Ł. Łacny. Neural network control design consideration for the active damping of a smart beam. *Journal of Theoretical and Applied Mechanics*, 2014.

[55] S. S. Mokri, H. Husain, W. Martono, A. Shafie. Real time implementation of narma-l2 control of a single link manipulator. *American Journal of Applied Sciences*, 5(12):1642–1649, 2008.

[56] A. Pukrittayakamee, O. De Jesús, M. T Hagan. Smoothing the control action for narma-l2 controllers. In *Circuits and Systems, 2002. MWSCAS-2002. The 2002 45th Midwest Symposium on*, volume 3, pages III–37. IEEE, 2002.

[57] K. Srakaew, J. Kananai, R. Chancharoen. Trajectory control of a nonlinear dynamical system using narma l2 neurocontroller. *Journal of Computer and Information Technology*, 1(1):11–17, 2011.

[58] M. T. Frye, R. S. Provence. Direct inverse control using an artificial neural network for the autonomous hover of a helicopter. In *Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on*, pages 4121–4122. IEEE, 2014.

[59] D. Chhabra, S. Kumar, et al. Active vibration control of the smart plate using artificial neural network controller. *Advances in Aerospace Engineering*, 2015, 2015.

[60] A. Draeger, S. Engell, H. Ranke. Model predictive control using neural networks. *Control Systems, IEEE*, 15(5):61–66, 1995.

[61] M. Sekiguchi, T. Sugasaka, S. Nagata. Control of a multi variable system by a neural network. In *Proc. IEEE Int. Conf. Robotics and Automation*, volume 3, pages 2644–2649, 1991.



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ

[62] S.-S. Kim, T. I. Kim, K. S. Jang, S. Jung, S. S. Kim. Control experiment of a wheeled drive mobile pendulum using neural network. In *Industrial Electronics Society, 2004. IECON 2004. 30th Annual Conference of IEEE*, volume 3, pages 2234–2239. IEEE, 2004.

[63] F. Chetouane, S. Darenfed. Neural network narma control of a gyroscopic inverted pendulum. *Engineering Letters*, 16(3):274–279, 2008.

[64] C. M. Gustavo Luiz, R. L. T. de Abreu, F. R. José. A neural network-based direct inverse control for active control of vibrations of mechanical systems. In *Proceedings of the VI Brazilian Symposium on Neural Networks (SBRN'00)*, page 107, 2000.

[65] G. F. Franklin, J. D. Powell, A. Emami-Naeini. *Feedback Control of Dynamics Systems*. Prentice Hall, 1994.

[66] T. Kaczorek, A. Dzieliński, W. Dąbrowski, R. Łopatka. *Podstawy teorii sterowania*. Wydawnictwo WNT, 2014.

[67] J. P. LaSalle, S. Lefschetz. *Stability by Liapunov's Direct Method: with Applications*, volume 4. Academic Press New York, 1961.

[68] J.-J. E. Slotine, W. Li, et al. *Applied Nonlinear Control*, volume 199. Prentice-Hall Englewood Cliffs, NJ, 1991.



BIBLIOTEKA CYFROWA POLITECHNIKI KRAKOWSKIEJ