Sprawozdanie ICS – Mańka Lisek, Lab2

# Feedforwardnet – Dane 1

Dane wykorzystane do trenowania sieci neuronowej są pobrane z Reaction Pendulum. Stanowi to rozszerzenie zadania wykonywanego na Labolatorium Problemowym.

theta = dane\_NN.signals(1).values + 1.58;

time = dane\_NN.time;

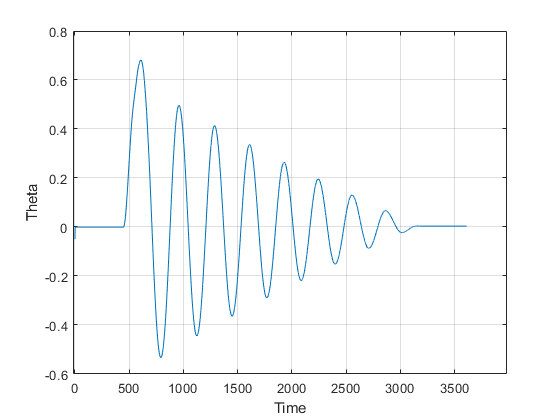
figure

plot(theta)

xlabel('Time')

ylabel('Theta')

grid on



Dane zostały podzielone zgodnie z teorią przedstawioną na labolatoriach. Początkowo sieć neuronowa była trenowana dla 2 warstw.

y\_begin\_sample = 602;

y\_end\_sample = 2875;

training\_data\_input\_1 = theta(y\_begin\_sample-2:y\_end\_sample-2)

training\_data\_input\_1 = 2274×1

0.6777

0.6770

0.6782

0.6797

0.6798

0.6796

0.6807

0.6804

0.6800

0.6808

⋮

training\_data\_input\_2 = theta(y\_begin\_sample-1:y\_end\_sample-1)

training\_data\_output = theta(y\_begin\_sample:y\_end\_sample) ⋮

x\_train = [training\_data\_input\_1'; training\_data\_input\_2']

x\_train = 2×2274

0.6777 0.6770 0.6782 0.6797 0.6798 0.6796 0.6807 ⋯

0.6770 0.6782 0.6797 0.6798 0.6796 0.6807 0.6804

% time = time(600:2873)

y\_train = training\_data\_output'

y\_train = 1×2274

0.6782 0.6797 0.6798 0.6796 0.6807 0.6804 0.6800 ⋯

Następnie została wytrenowana siec neuronowa z użyciem funkcji feedfordwardnet.

net = feedforwardnet(2, 'trainlm');

net = train(net, x\_train, y\_train);

view(net);

y\_net = net(x\_train);

perf = perform(net,y\_train, y\_net)

net = feedforwardnet(2, 'trainlm');

net = train(net, x\_train, y\_train);

y\_net = net(x\_train);

sample\_time = 0.01;

perf = perform(net,y\_train, y\_net)

perf = 3.8247e-06

gensim(net, sample\_time)

ans = 'untitled1'

figure();

hold on

plot(time, y\_train)

plot(time, y\_net)

hold off

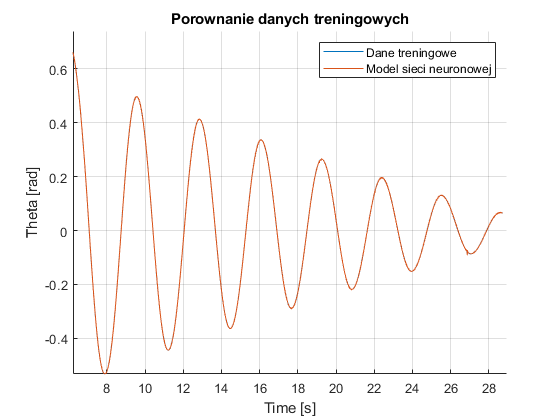
legend("Dane treningowe", 'Model sieci neuronowej')

xlabel('Time [s]')

ylabel('Theta [rad]')

title('Porownanie danych treningowych')

grid on



err = immse(y\_train, y\_net)

err = 3.8455e-06

err = y\_net-y\_train

err = 1×2274

-0.0018 -0.0005 0.0013 0.0003 -0.0013 0.0013 0.0002 ⋯

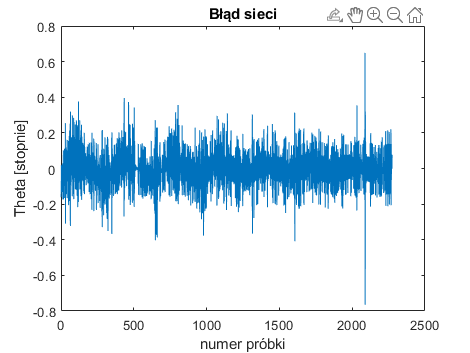
err\_degree = err \* 180/pi

err\_degree = 1×2274

-0.1040 -0.0299 0.0722 0.0167 -0.0750 0.0744 0.0108 ⋯

figure;

plot(err)



layers = 1:4;

layers\_test = zeros(length(layers), length(training\_data\_output));

for i=layers

net = feedforwardnet(i, 'trainlm');

net = train(net, x\_train, y\_train);

y\_net = net(x\_train);

layers\_test(i, :) = y\_net;

perf = perform(net,y\_train, y\_net)

end

perf = 3.8506e-06

perf = 3.8419e-06

perf = 3.8511e-06

perf = 3.8247e-06

err = layers\_test - y\_net;

hold on

plot(err(1, :))

plot(err(2, :))

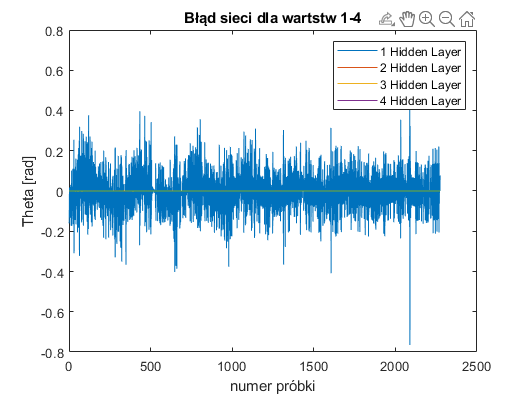
plot(err(3, :))

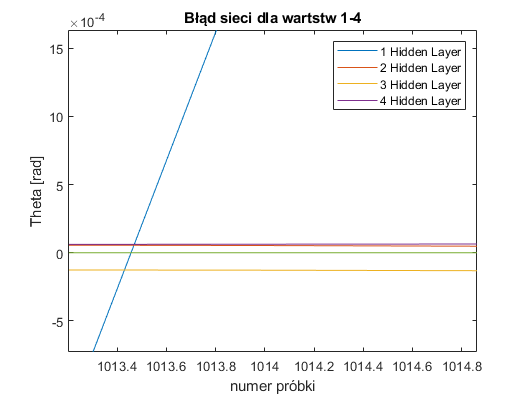
plot(err(4, :))

hold off

legend('1 Hidden Layer', '2 Hidden Layer', '3 Hidden Layer', '4 Hidden Layer')

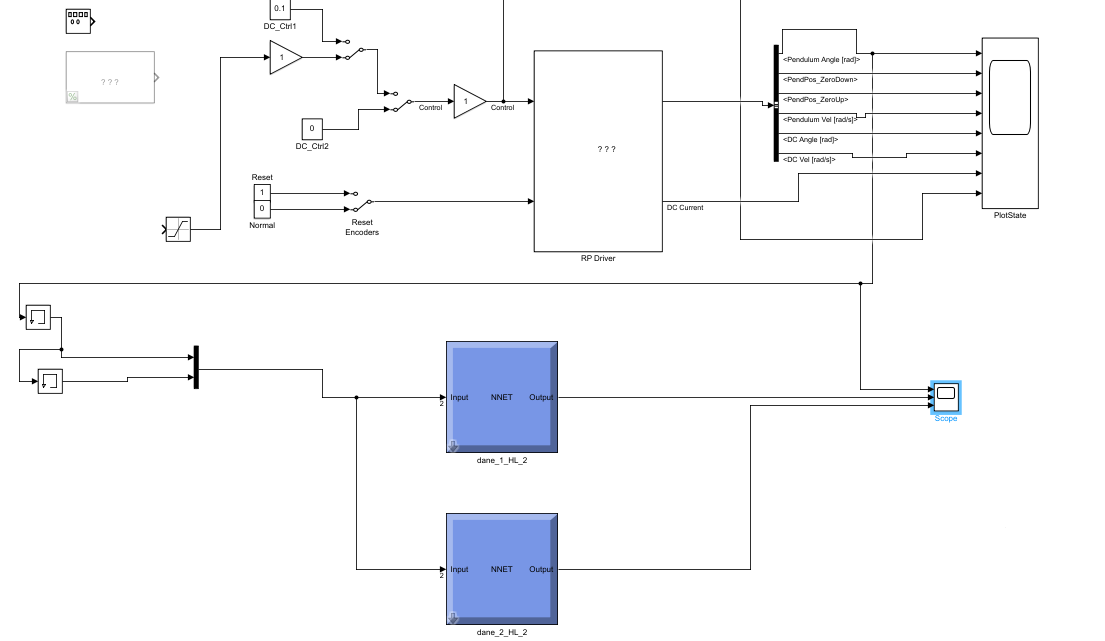
# Analiza błędów sieci

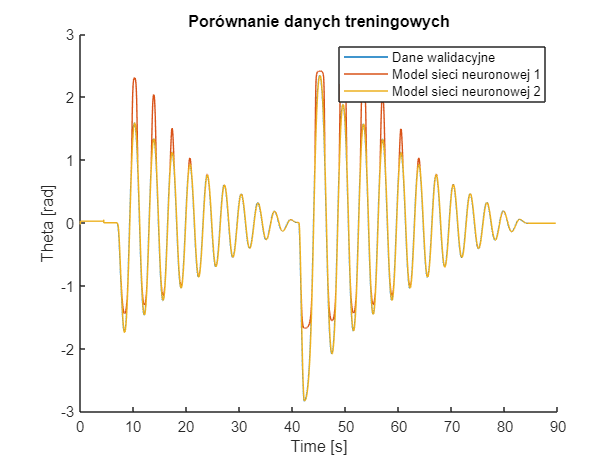




# Porównanie dla danych o większym zakresie ruchu wahadła.

Rysunek poniżej przedstawia walidację dwóch sieci neuronowych sprawdzonych na labolatoriach. Druga sieć neuronowa jest przetrenowana na większej ilości danych.





# Narxnet

Narxnet w środowisku Matlab to zaawansowane narzędzie, które profesjonalnie implementuje opisane procedury. Aby skorzystać z tej funkcji w pełni efektywnie, konieczne jest przekształcenie formatu danych na typ 'cell'. Następnie należy dostarczyć jako wejście zmienną theta, reprezentującą wynikową zmienną sieci. Kluczowym elementem jest precyzyjne określenie parametrów działania sieci, takich jak opóźnienia w danych, które mają być zaimplementowane w celu generowania danych treningowych wewnątrz sieci.

Narxnet w Matlab to narzędzie o dużej elastyczności, umożliwiające skuteczne tworzenie modeli neuronowych. Dostosowywanie opóźnień danych pozwala na dokładne dostosowanie sieci do specyfiki danego zadania. Ten proces staje się szczególnie kluczowy w przypadku danych zawierających złożone wzorce czasowe. Eksperymentowanie z różnymi konfiguracjami staje się niezbędne w celu osiągnięcia optymalnej wydajności modelu.

%wyjscie jest wejściem

x\_train = training\_data\_output'

x\_train = 1×4101

0.1725 0.2099 0.2472 0.2846 0.3220 0.3588 0.3958 ⋯

inputSeries = num2cell(x\_train)

inputSeries = 1×4101 cell

|  | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **⋯** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | 0.1725 | 0.2099 | 0.2472 | 0.2846 | 0.322 | 0.3588 | 0.3958 |  |

targetSeries = num2cell(x\_train)

targetSeries = 1×4101 cell

|  | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **⋯** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | 0.1725 | 0.2099 | 0.2472 | 0.2846 | 0.322 | 0.3588 | 0.3958 |  |

net = narxnet(1:2, 1, 2);

[Xs,Xi,Ai,Ts] = preparets(net,targetSeries,{},inputSeries)

net = train(net,Xs,Ts,Xi,Ai);



[y\_net,Xf,Af] = net(Xs,Xi,Ai);

y\_net = cell2mat(y\_net)

y\_net = 1×4099

0.2471 0.2845 0.3218 0.3592 0.3953 0.4327 0.4685 ⋯

figure();

hold on

plot(training\_data\_output)

plot(y\_net)

hold off

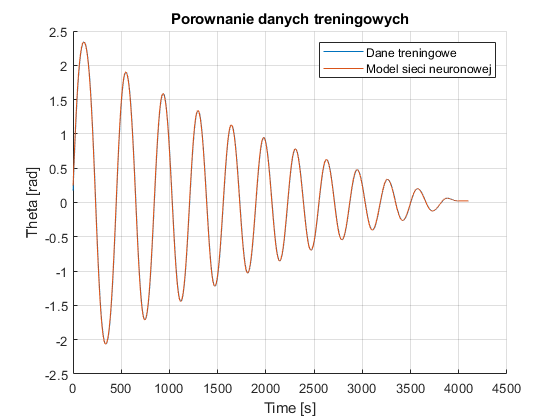
legend("Dane treningowe", 'Model sieci neuronowej')

xlabel('Time [s]')

ylabel('Theta [rad]')

title('Porownanie danych treningowych')

grid on



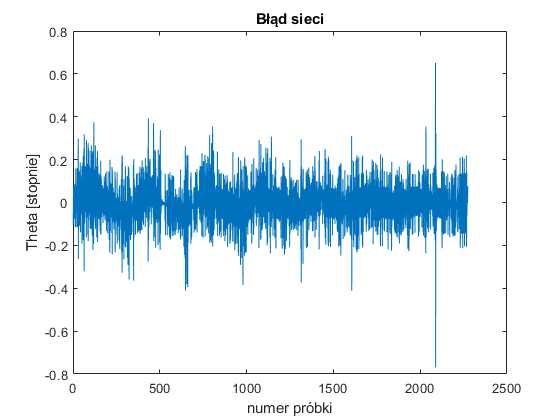
figure;

plot(err\_degree)

xlabel('numer próbki')

ylabel('Theta [stopnie]')

title('Błąd sieci')



## Wnioski i spostrzeżenia

Dla sieci feedforward dane zostały dopasowane z błędem (pomiędzy próbkami) poniżej 0.005 radiana dla 1 warstwowej sieci neuronowej.

Dla większej liczby warstw od 2 – 4 błąd jest na poziomie e-5.

Problemem okazał się fakt, iż sieć została przetrenowana dla danych w zakresie -0.6 do 0.6 radianów. Po zaaplikowaniu danych z zakresu -3.14 do 3.14 radianów okazało się, że błąd dla wartości spoza danych treningowych osiąga wartości 1 radiana (pomiędzy próbkami). Z tego powodu przetrenowano sieć na nowych danych, w pełnym zakresie wahadła, co poskutkowało znaczącą poprawą.

Dzięki użyciu bloczka ‘gensim’ i ‘scope’ możliwa była obserwacja działania modelu w czasie rzeczywistym.