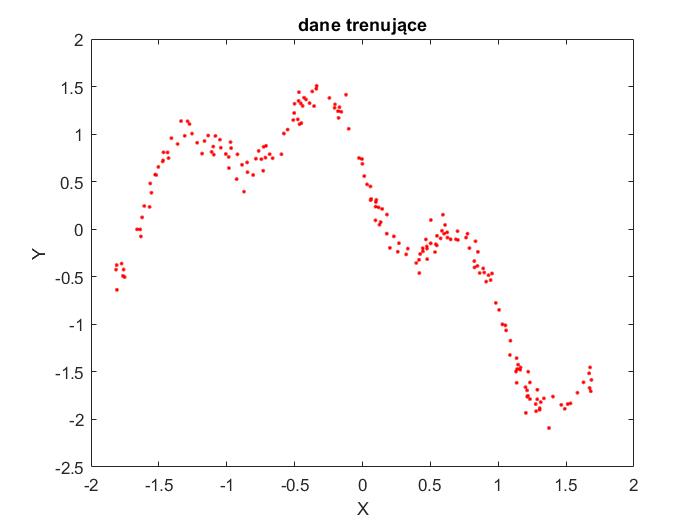
# 1 Cel projektu

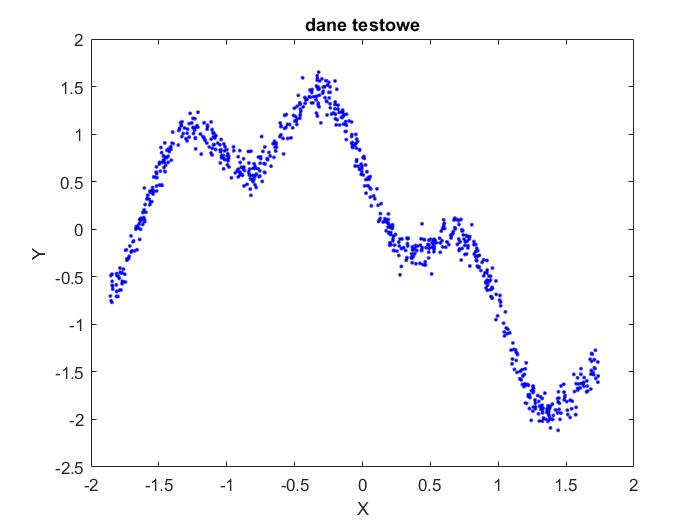
Celem projektu jest aproksymacja jednowymiarowej funkcji na podstawie dostarczonych danych przy użyciu sieci neuronowej. Zaprojektowana sieć neuronowa powinna być optymalna dla zadanych danych, czyli minimalizować błąd średniokwadratowy funkcji przy jednoczesnej dobrej generalizacji.

# 2 Opis i wizualizacja danych

Po załadowaniu i uporządkowaniu danych otrzymano dwie macierze – dataTrain, która składa się z 200 par (x,y), gdzie x to wartość wejściowa, a y to wartość wyjściowa oraz dataTest, która zawiera 800 par (x,y). Wszystkie dane zostały znormalizowane i scentrowane na podstawie parametrów otrzymanych z funkcji zscore(), gdzie jako argument podano macierz dataTrain. Dane po normalizacji zaprezentowano na wykresach poniżej:



Rysunek 1. Wykres znormalizowanych i wycentrowanych danych trenujących.



Rysunek 2. Wykres znormalizowanych i scentrowanych danych testujących.

# 3 Algorytm uczenia

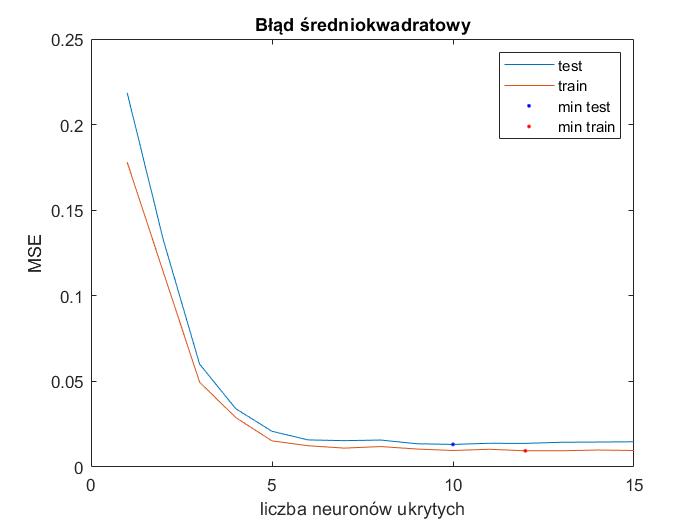
Projekt zrealizowano w środowisku Matlab R2018b. W celu aproksymacji jednowymiarowej funkcji stworzono sieć neuronową składającą się z jednego neuronu wejściowego, jednej warstwy neuronów ukrytych o funkcji aktywacji tangens hiperboliczny (tansig) oraz jednego neuronu wyjściowego o liniowej funkcji aktywacji (purelin). Do każdego neuronu dołączone zostało dodatkowe wejście bias. Wszystkie początkowe wagi sieci oraz wartości wejściowe bias były wartościami losowymi z przedziału (-0.15,0.15) otrzymanymi za pomocą funkcji init(). Dla każdej iteracji ustawiono identyczne warunki początkowe za pomocą funkcji rng(). Uczenie sieci odbywa się w dwóch etapach. Pierwszy etap składa się maksymalnie ze 100 iteracji przy użyciu algorytmu wstecznej propagacji błędu (traingd) i zostaje przerwany gdy błąd średniokwadratowy będzie mniejszy lub równy 0,001. Kolejne 200 epok wykorzystuje algorytm Levenberga-Marquadta (trainlm) i uczenie zostaje przerwane jeśli błąd średniokwadratowy będzie mniejszy bądź równy 0,001.

# 4 Raport z wykonania zadań projektu i wybór optymalnego modelu

W celu dobrania liczby neuronów w warstwie ukrytej 50 razy sprawdzano działanie sieci o liczbie neuronów ukrytych od 1 do 15, policzono przy tym średni błąd średniokwadratowy dla zbioru uczącego i testowego oraz za wyświetlano wykres dźwigni od danych trenujących. Dla każdej iteracji ustawiono takie same warunki początkowe za pomocą funkcji rng().

Na tej podstawie sporządzono wykresy średniego błędu średniokwadratowego dla zbioru uczącego i testowego z 10 prób dla liczby neuronów ukrytych od 1 do 15 i zaznaczono minimum.

## 4.1 Wyznaczanie liczby neuronów ukrytych na postawie błędy średniokwadratowego

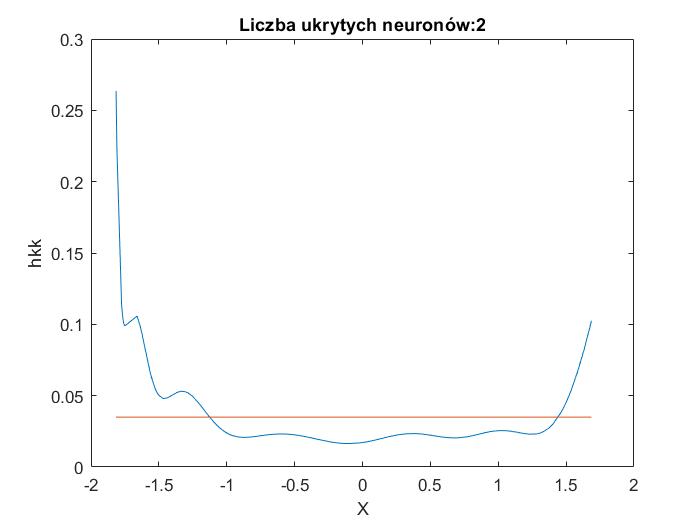
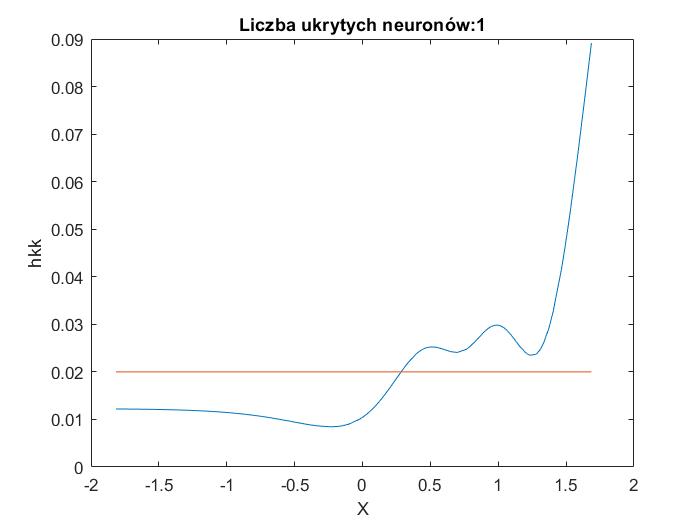


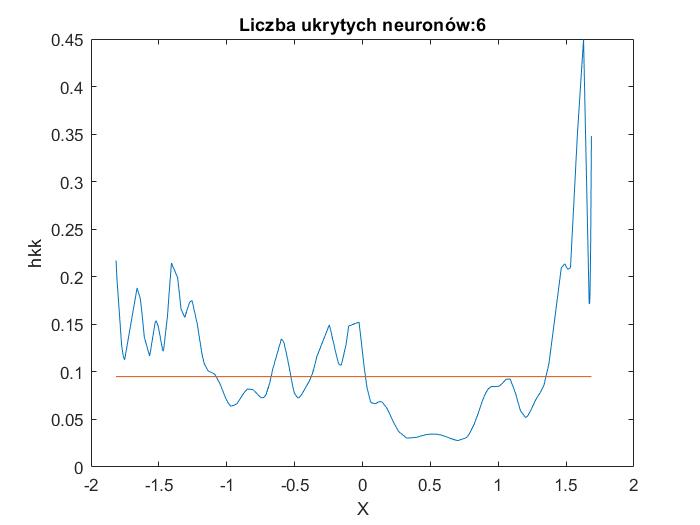
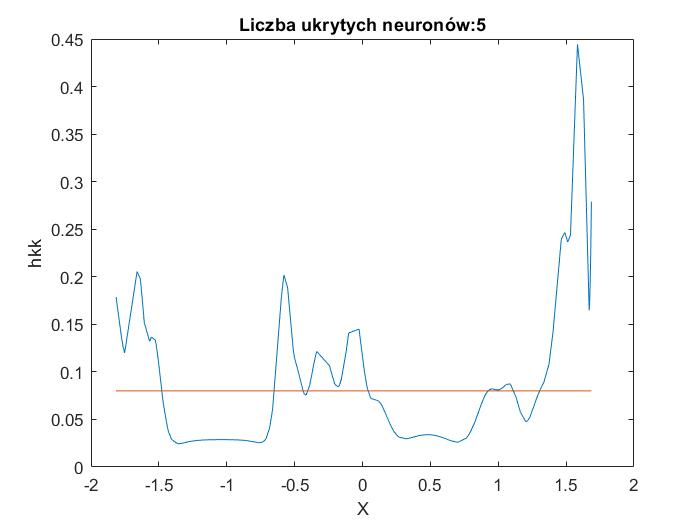
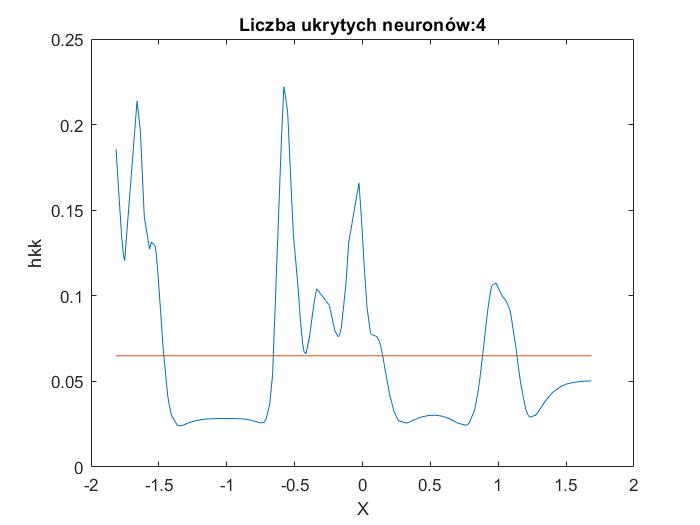
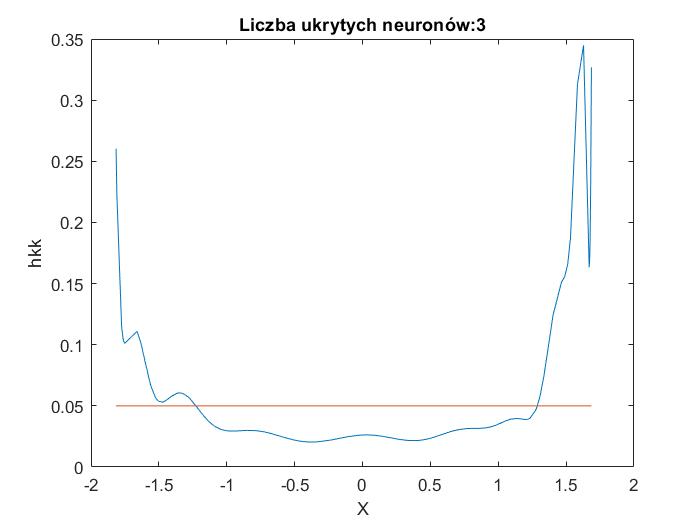
Rysunek 3. Wykres średniego błędy średniokwadratowego od liczby neuronów w warstwie ukrytej.

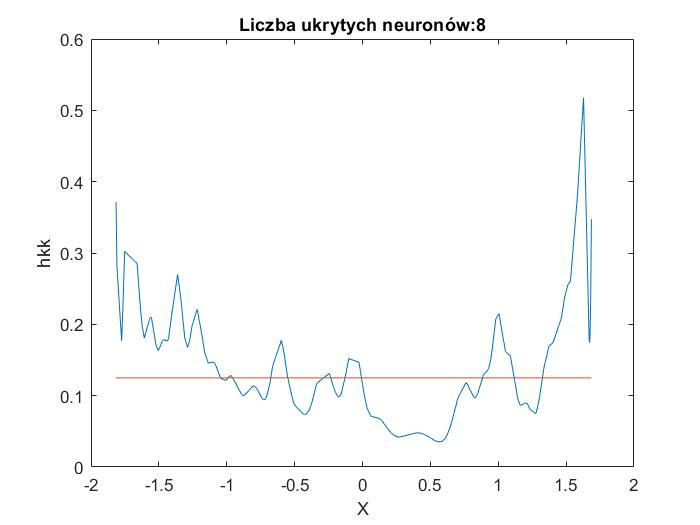
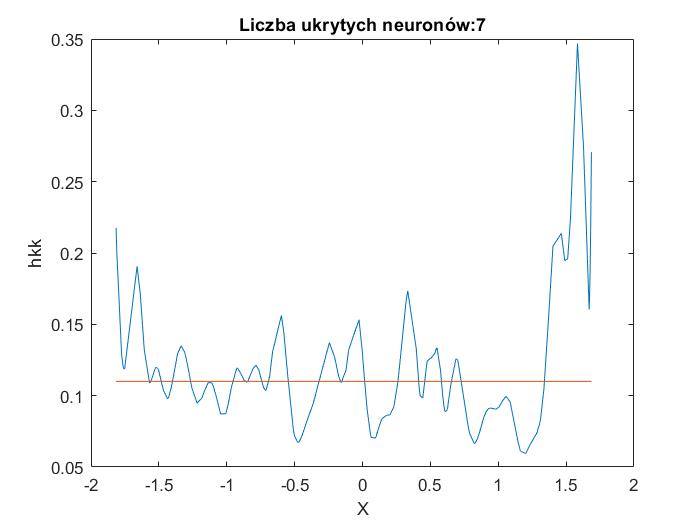
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Liczba neuronów w warstwie ukrytej | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| MSE Test | 0,177993 | 0,113719 | 0,049554 | 0,028826 | 0,015262 | 0,012417 | 0,011035 | 0,011941 |
| MSE Train | 0,218556 | 6,624392 | 3,004828 | 1,701205 | 1,04299 | 0,79094 | 0,76964 | 0,78737 |
| Liczba neuronów w warstwie ukrytej | 9 | 10 | 11 | 12 | 13 | 14 | 15 |  |
| MSE Test | 0,010496 | 0,009653 | 0,010361 | 0,009465 | 0,009482 | 0,009916 | 0,00963 |  |
| MSE Train | 0,013582 | 0,01319 | 0,013844 | 0,013818 | 0,014445 | 0,014574 | 0,014736 |  |

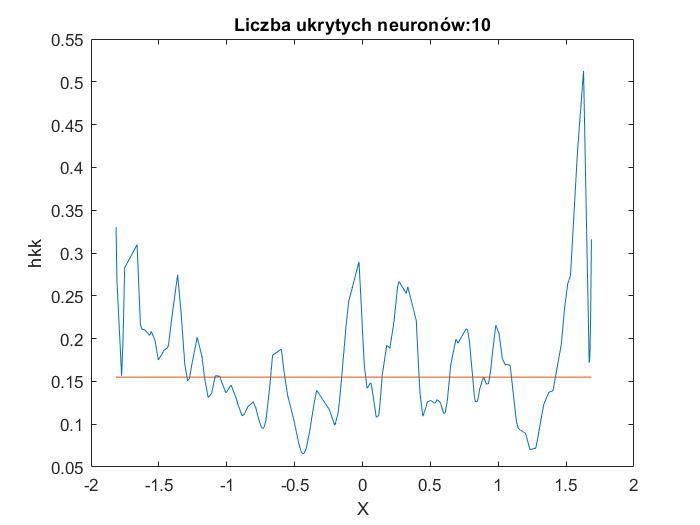
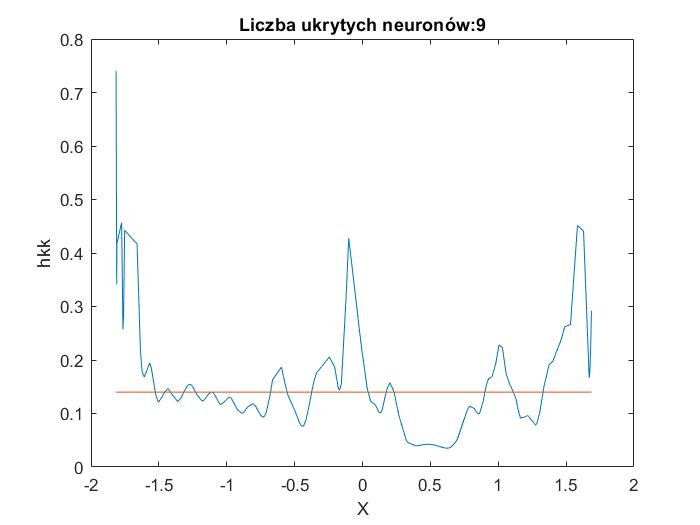
Z wykresu widać, że błąd średniokwadratowy znacznie maleje, jeśli liczba neuronów ukrytych jest większa równa 5. Najmniejszy błąd średniokwadratowy dla danych testujących jest dla 9 neuronów w warstwie ukrytej, natomiast dla zbioru uczącego najmniejszy błąd występuje przy 15 neuronach ukrytych.

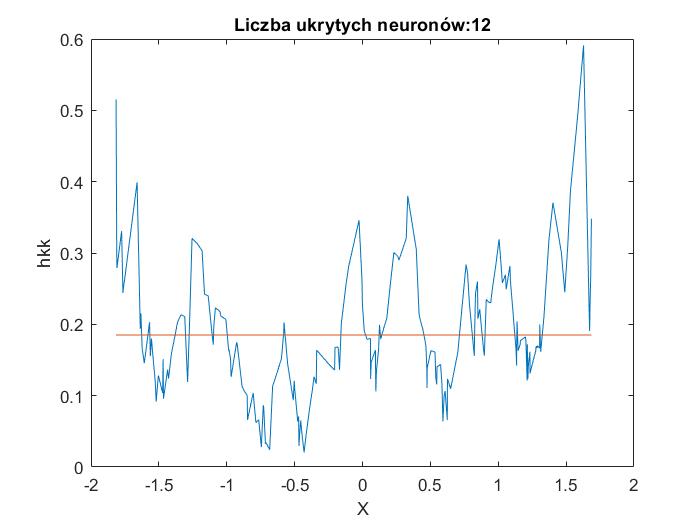
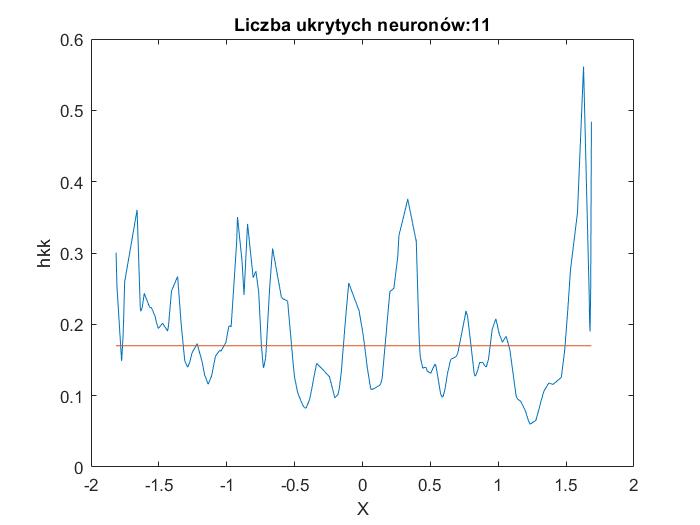
## 4.2 Wyznaczanie liczby neuronów ukrytych przy użyciu metody virtual leave-one-out

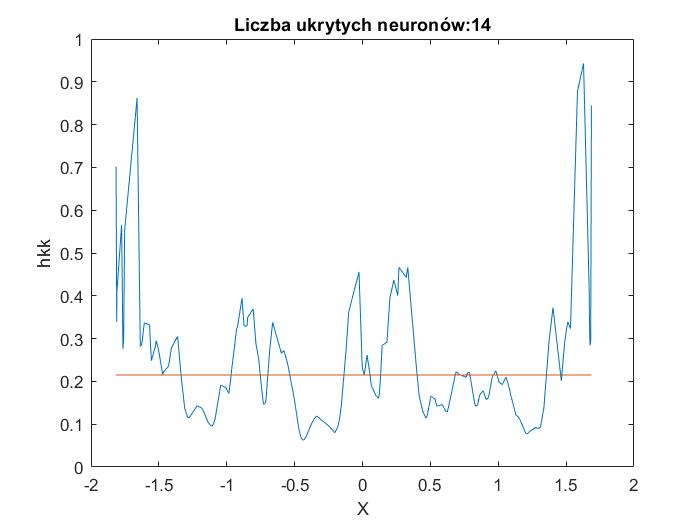
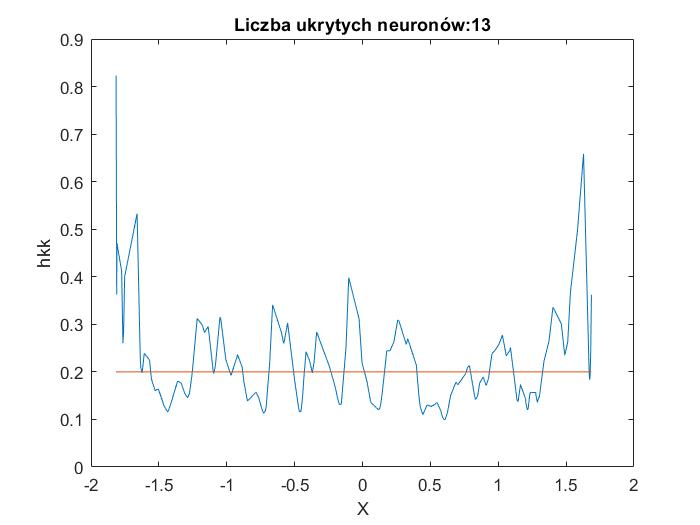


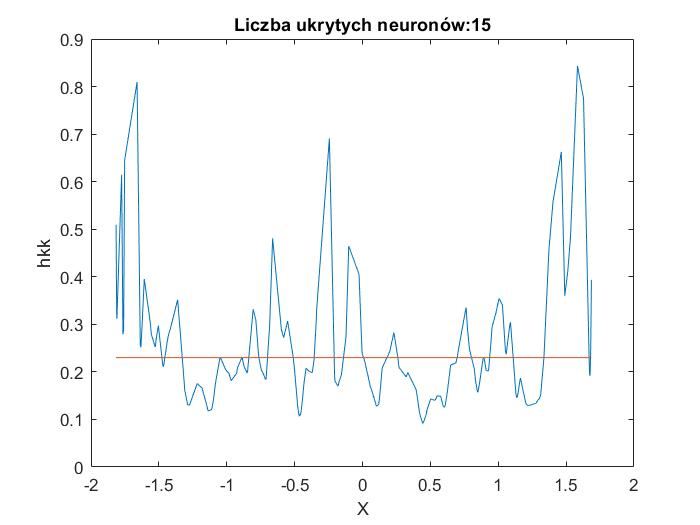






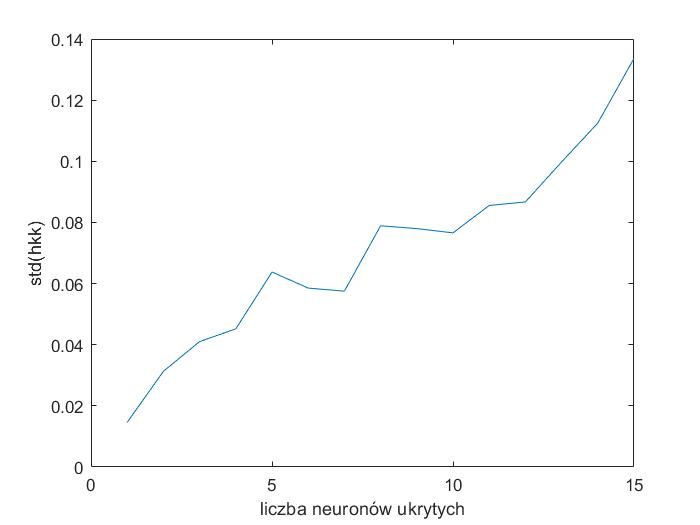






Rysunek 4. Przykładowe wykresy dźwigni od X danych trenujących.

Na podstawie wyników błędu średniokwadratowego stwierdzono, że liczba neuronów w warstwie ukrytej musi być większa od 5. Aby wybrać dokładną liczbę neuronów skorzystano z metody virtual leave one out. Policzono średnie odchylenie standardowe dźwigni hkk dla każdej z liczby neuronów w warstwie ukrytej.



Rysunek 5. Średnie odchylenie standardowe dźwigni.

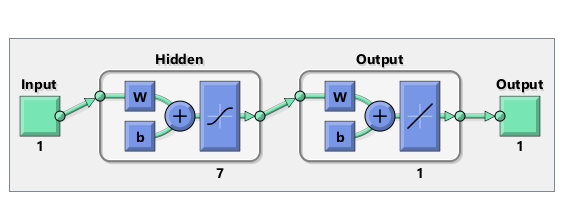
Do obliczenia Jakobianu wykorzystano funkcję fpderiv(). Dla liczby neuronów w warstwie ukrytej od 1 do 15 dokonano wizualnej oceny jakości modelu. Dźwignie hkk pokazują wyraźnie, że jeżeli neuronów ukrytych jest 13 lub więcej następuje nadmierne dopasowanie, ponieważ niektóre dźwignie są   
bliskie 1.

## 4.3 Ostateczny wybór liczby neuronów w warstwie ukrytej.

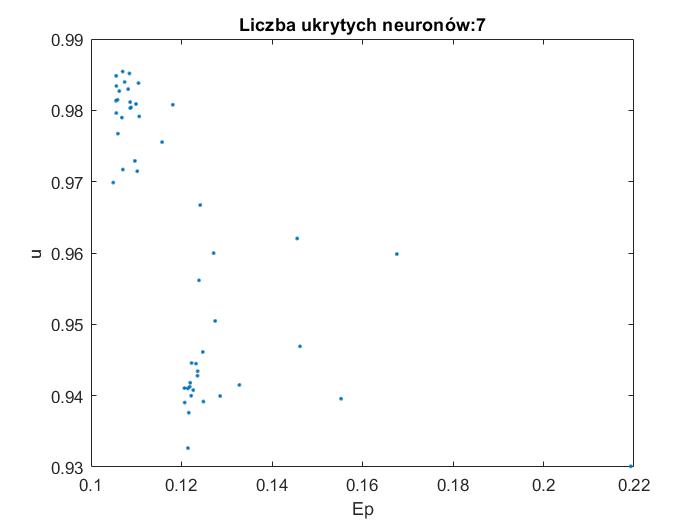
Porównując wyniki otrzymane z obliczania błędu średniokwadratowego oraz wykresy dźwigni uznano, że 7 neuronów w warstwie ukrytej będzie optymalnym wyborem, który zapewni małą złożoność obliczeniową modelu, mały błąd otrzymanych za pomocą sieci wartości wyjściowych oraz nie będzie zachodziło zjawisko overfitting’u, dzięki stosunkowo niedużemu odchyleniu standardowemu dźwigni.

# 5 Parametry optymalnej sieci neuronowej

Wykonano 50 symulacji sieci neuronowej o 7 neuronach ukrytych. Dla każdej sieci policzono Jakobian i w każdym przypadku był on pełnego rzędu. Obliczono wartości Ep oraz µ. Najmniejszy uzyskany błąd Ep wynosił 0.1049 przy jedoczesnej wartości µ=0.9699.

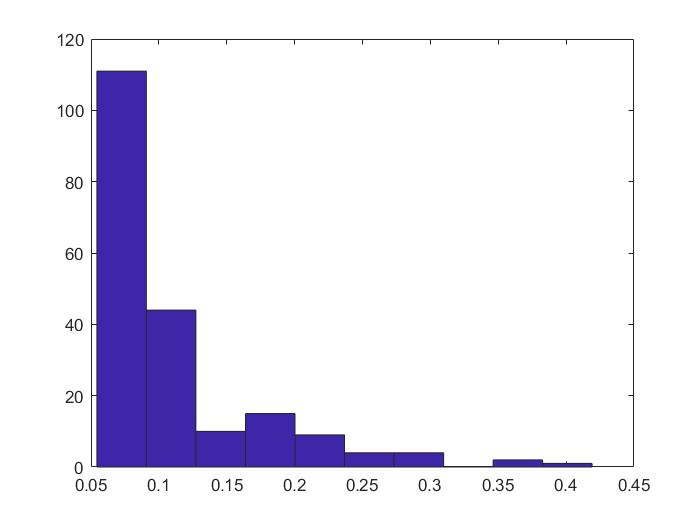


Rysunek 6. Schemat sieci neuronowej.



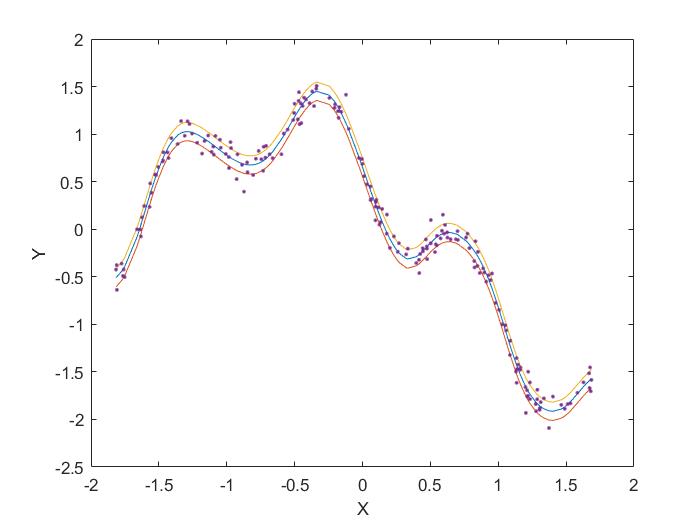
Rysunek 7. Wykres µ od Ep.

Dla najlepszej sieci neuronowej sporządzono również histogram dźwigni.



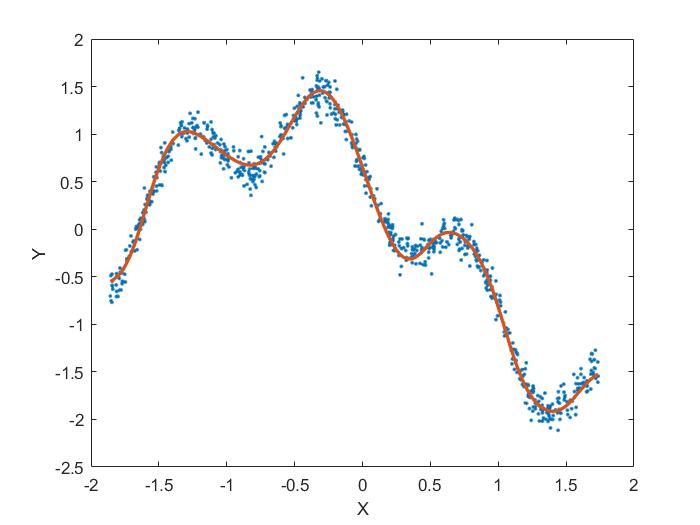
Rysunek 8. Histogram dźwigni dla najlepszej sieci neuronowej.

Wykonano również wykres aproksymacji dla zbioru uczącego wraz z zaznaczonymi przedziałami ufności na poziomie 95%.



Rysunek 9. Wykres aproksymacji dla zbioru uczącego wraz z przedziałami ufności.

Dodatkowo przeprowadzono ocenę działania sieci na zbiorze testowym i policzono błąd średniokwadratowy, który był równy 0.0119.

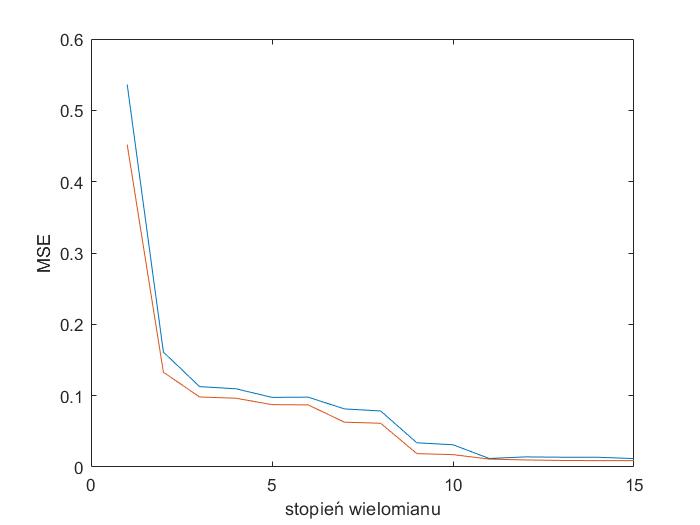


Rysunek 10. Wyniki aproksymacji dla zbioru testowego.

|  |
| --- |
| Wagi wejściowe |
| -10,1246 |
| 7,1973 |
| -11,6114 |
| -8,8003 |
| -10,5605 |
| 9,8506 |
| 10,0269 |
| bias wejściowy |
| 9,1208 |
| -4,5295 |
| 3,6105 |
| 0,4471 |
| -2,9271 |
| 5,516 |
| 8,8298 |
| wagi połączeń wewnątrz sieci |
| -0,15886 |
| -0,609 |
| -0,12692 |
| 0,530197 |
| -0,2517 |
| -0,11611 |
| 0,473562 |
| bias n. wyjściowego |
| -0,4543 |

# 6 Porównanie działania sieci neuronowej i regresji wielomianowej

Przy użyciu funkcji polyfit() dokonano dopasowania danych uczących do wielomianów różnych rzędów od 1 do 15. Dla każdego wielomianu policzono błąd średniokwadratowy zarówno dla danych uczących jak i testujących. Dokonano również wizualizacji wyników wielomianu dla danych testowych.



Rysunek 11. Zależność błędu średniokwadratowego od stopnia wielomianu.

W przypadku regresji wielomianowej wynik błędu średniokwadratowego podobny do uzyskanego przy pomocy sieci neuronowej można uzyskać dla wielomianów przynajmniej 11 stopnia. Jednakże porównując wykresy otrzymane przy użyciu wielomianu i sieci neuronowej można zauważyć, że w przypadku wielomianu dla skrajnych danych następuje znaczna rozbieżność w stosunku do wartości rzeczywistych.

# 7 Wnioski.

Sieć neuronowa o 7 neuronach w warstwie ukrytej jest w stanie bardzo dobrze aproksymować zaszumioną funkcję nieliniową. Dzięki obserwacji zmiany błędu średniokwadratowego oraz dźwigni i odchylenia standardowego dźwigni w zależności od liczby neuronów w warstwie ukrytej można wybrać optymalną liczbę neuronów ukrytych, która pozwoli na bardzo dobrą aproksymację danych. Sieć neuronowa o 7 neuronach w warstwie zdaje się skuteczniej aproksymować tego typu dane niż dopasowanie wielomianowe.