**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI I INFORMATYKI  
POLITECHNIKA OPOLSKA**

**SPRAWOZDANIE Z PRZEDMIOTU:**

NARZĘDZIA SZTUCZNEJ INTELIGENCJI

**TEMAT:**

Jednokierunkowa sieć neuronowa jako aproksymator  
funkcji – odwrotne zadanie kinematyki

**Ćwiczenie wykonali:**

Paweł Siemiginowski (s101450)  
Paweł Cedzich (s101598)

**Data wykonania:**

07.06.2023 r.

**Data oddania:**

07.06.2023 r.

**Prowadzący:**

dr inż. Alina Kaleta-Jurowska

**Ocena prowadzącego:**

…………………………………..

**Spis treści:**

1. Cel i zakres ćwiczenia.
2. Wstęp teoretyczny.
3. Przebieg ćwiczenia.
   1. Dane wejściowe.
   2. Wykorzystane funkcje.
   3. Kod źródłowy.
   4. Otrzymane wyniki.
4. Wnioski.
5. **Cel i zakres ćwiczenia**

Celem niniejszego ćwiczenia jest zastosowanie jednokierunkowej, dwuwarstwowej sieci neuronowej w celu rozwiązania problemu odwrotnego zadania kinematyki w dziedzinie robotyki. Zadanie polega na wykorzystaniu sieci neuronowej jako systemu sterowania dwuczłonowym manipulatorem planarnym (rys. 1). Manipulator ten składa się z prostych ramion, które poruszają się w płaszczyźnie. W ramach tego ćwiczenia zostanie wykorzystane oprogramowanie Matlab do implementacji i analizy działania sieci neuronowej. Oprogramowanie to zapewnia odpowiednie narzędzia do tworzenia, trenowania i testowania sieci neuronowych.

Obraz zawierający linia, diagram, Wykres, tekst

Opis wygenerowany automatycznie

*Rys. 1. Schemat poglądowy dwuczłonowego manipulatora planarnego.*

1. **Wstęp teoretyczny**

Na podstawie podanych przez operatora docelowych współrzędnych chwytaka (x, y), wykonano neuronowy układ sterujący, który ma za zadanie obliczyć odpowiednie wartości kątów przegubów α i β manipulatora, aby chwytak znalazł się dokładnie w punkcie zadanym przez operatora. Operator ma również możliwość podania sekwencji współrzędnych (x, y), odpowiadającej określonej trajektorii chwytaka – w takim przypadku wykonany układ sterujący generuje odpowiednią sekwencję kątów przegubów (α, β).

W pierwszej części ćwiczenia wykonano samodzielnie dwie funkcje Matlaba, które rozwiązują proste oraz odwrotne zadanie kinematyki. Następnie wykonano przygotowanie danych uczących, które reprezentują współrzędne punktów (x, y) na płaszczyźnie (wzorce wejściowe) oraz odpowiadające im wartości kątów przegubów (α, β) manipulatora (wzorce wyjściowe). Wzorcowe wartości współrzędnych oraz kątów zostały wykonane dla pewnej zadanej trajektorii chwytaka, mieszczącej się wewnątrz obszaru roboczego manipulatora. Kolejnym etapem było utworzenie jednokierunkowej sieci neuronowej o dwóch wejściach i dwóch wyjściach, składającej się z dwóch warstw neuronów.

Następnie przeprowadzono proces uczenia sieci z wykorzystaniem przygotowanych wcześniej wzorców uczących. Po nauczeniu sieci wykonano weryfikację jej działania, podając na jej wejścia zadane wartości współrzędnych (x, y) chwytaka, odpowiadające trajektorii uczącej. Wygenerowane przez sieć wartości kątów (α, β) przegubów zostały skonfrontowane z wartościami wzorcowymi. Wykonano także analizę trajektorii chwytaka, gdyby do sterowania przegubami manipulatora wykorzystane zostały wartości kątów wygenerowane przez sieć neuronową; trajektorię tę również porównano z trajektorią wzorcową.

W ostatnim etapie wykonano badanie czy sieć nauczona na podstawie wzorców wchodzących w skład trajektorii uczącej jest w stanie poprawnie wyznaczać wartości kątów przegubów dla punktów rozmieszczonych na trajektorii innej niż wzorcowa, leżącej w granicach obszaru roboczego.

1. **Przebieg ćwiczenia**
   1. **Dane wejściowe**

W przypadku tego ćwiczenia, dane wejściowe stanowiły wartości stałych potrzebnych do realizacji zadań związanych z kinematyką. Dane, które zostały wykorzystane to:

* Długości przegubów manipulatora
* Wartość punktu początkowego spirali .
* Wzorce 100 danych uczących macierz wyznaczonych na podstawie rozmieszczenia na spirali Archimedesa o równaniu (gdzie – promień wodzący spirali, – kąt, jaki tworzy wektor wodzący spirali z osią ).
* Macierz wartości oczekiwanych kątów – wyznaczonych za pomocą funkcji .
  1. **Wykorzystane funkcje**

**sim** – funkcja służąca do symulacji działania dowolnej sieci neuronowej.

**newff** – inicjalizacja wielowarstwowej sieci jednokierunkowej (początkowe wartości współczynników wagowych – losowe).

**train** – uczenie dowolnej jednokierunkowej sieci neuronowej (także jednokierunkowej sieci wielowarstwowej).

**prostkin** – napisana w trakcie ćwiczenia funkcja wyznaczająca współrzędne chwytaka na podstawie długości przegubów oraz kątów między nimi, a osią – z następujących wzorów:

**odwrkin** – napisana w trakcie ćwiczenia funkcja wyznaczająca wartości kątów przegubów na podstawie następujących wzorów:

gdzie:

**atan2** – dwuargumentowy arcus tangens, wykorzystywany do wyznaczania ćwiartki układu współrzędnych do której należy wynik.

* 1. **Kod źródłowy**
* prostkin.m

function[x, y] = prostkin(alfa, beta, l1, l2)

x = l1 \* cos(alfa) + l2 \* cos(alfa + beta);

y = l1 \* sin(alfa) + l2 \* sin(alfa + beta);

end

* odwrkin.m

function [alfa, beta] = odwrkin(x, y, l1, l2)

c2 = (x.^2 + y.^2 - l1^2 - l2^2) / (2 \* l1 \* l2);

s2 = sqrt(1 - c2.^2);

k1 = l1+(l2 \* c2);

k2 = l2\*s2;

alfa = atan2(y, x) - atan2(k2, k1);

beta = atan2(s2, c2);

end

Powyżej umieszczone zostały dwie funkcje, które zostały zastosowane do wyznaczania punktów chwytaka bądź wartości kątów przegubów manipulatora.

* script1.m (tworzenie sieci)

close all

clear

l1 = 1;

l2 = 1;

xsr = 0.5;

ysr = -0.5;

fi = 0:0.5:49.5;

r = 0.01 \* fi;

[x, y] = pol2cart(fi, r);

x = x+xsr;

y = y+ysr;

figure(1)

plot(x, y, 'r\*-');

xlabel('Współrzędna x')

ylabel('Współrzędna y')

title('Wzorzec spirali')

[alfa, beta] = odwrkin(x, y, l1, l1);

figure(2)

plot(alfa, 'r\*-')

hold on

plot(beta, 'b\*-')

xlabel('Indeks wartości w wektorze')

ylabel('Wartość kąta')

legend({'kąty α', 'kąty ß'}, 'Location', 'east')

title('Porównanie wartości kątów α i ß')

P = [x; y];

T = [alfa; beta];

net = newff(P, T, 2);

net.layers{1}.transferFcn = 'tansig';

net.layers{2}.transferFcn = 'purelin';

L1 = net.IW{1}

L2 = net.LW{2}

b1 = net.b{1}

b2 = net.b{2}

net.trainFcn = 'traingdx';

net.trainParam.goal = 0.0001;

net.trainParam.epochs = 500;

net.trainParam.max\_fail = net.trainParam.epochs;

net = train(net, P, T);

save dane\_ucz net l1 l2 P T

Na początku wykonano zamknięcie wszystkich otwartych okien i wyczyszczenie zmiennych. Następnie zostały określone wartości parametrów manipulatora: - długość pierwszego przegubu oraz - długość drugiego przegubu. Trajektoria manipulatora została wygenerowana w postaci spirali. Kąty zostały określone w zakresie od 0 do 49.5 stopnia z krokiem 0.5, a na ich podstawie obliczono promień . Współrzędne i zostały przekształcone z układu biegunowego na kartezjański i przesunięte o wartości i . Wygenerowano wykres ze współrzędnymi, przedstawiający spiralę wzorcową. Korzystając  
z funkcji odwrotnej kinematyki, obliczono wartości kątów i dla każdego punktu trajektorii. Dla tych wartości wygenerowano wspólny wykres. Utworzono macierze i , które zawierają współrzędne punktów trajektorii  
 i oraz odpowiadające im wartości kątów i .

Sieć neuronowa została inicjalizowana za pomocą funkcji , gdzie macierze i zostały przekazane jako dane wejściowe, a liczba neuronów w jednej warstwie ukrytej została ustawiona na początku na dwie. Funkcje aktywacji dla poszczególnych warstw sieci zostały zdefiniowane za pomocą . Parametry treningu sieci, takie jak funkcja treningu (), cel treningu (), liczba epok () i maksymalna liczba niepowodzeń (), zostały ustawione przed samym treningiem sieci.

Sieć neuronowa została wytrenowana za pomocą funkcji ,  
z wykorzystaniem macierzy i jako danych treningowych. Na koniec sieć neuronowa oraz wartości oraz zostały zapisane do pliku

* script2.m (generowanie spirali za pomocą utworzonej sieci)

close all

clear

load dane\_ucz.mat

Y = sim(net, P);

alfa = T(1,:);

beta = T(2,:);

alfas = Y(1,:);

betas = Y(2,:);

E = [alfa - alfas; beta - betas]

SSE = (1 / (size(T, 1) \* size(T, 2))) \* sumsqr(E)

disp(['Sumaryczny błąd średniokwadratowy (SSE): ' num2str(SSE)]);

figure

plot(alfa, 'ro-')

hold on

plot(alfas, '+-', 'Color', 'red')

plot(beta, 'bo-')

plot(betas, '+-', 'Color', 'blue')

xlabel('Indeks wartości w wektorze')

ylabel('Wartość kąta')

title('Porównanie wartości kątów α i ß')

legend({'Wartości kątów α (wzorzec)', 'Wartości kątów α (sieć)', 'Wartości kątów ß (wzorzec)', 'Wartości kątów ß (sieć)'}, 'Location','east')

figure

plot(E(1,:), 'r')

hold on

plot(E(2,:), 'b')

xlabel('Indeks wartości w wektorze')

ylabel('Błąd aproksymacji')

title('Wykres błędów aproksymacji wartości kątów α i ß')

legend({'Błąd aproksymacji α', 'Błąd aproksymacji ß'}, 'Location', 'north')

[xs, ys] = prostkin(alfas, betas, l1, l2);

figure

x = P(1,:);

y = P(2,:);

plot(x, y, 'r\*-')

hold on

plot(xs, ys, 'b\*-')

xlabel('Współrzędna x')

ylabel('Współrzędna y')

title('Porównanie trajektorii wzorcowej i wygenerowanej')

legend({'Trajektoria wzorcowa', 'Trajektoria wygenerowana przez sieć neuronową'}, 'Location', 'southeast')

W drugim skrypcie na początku wykonano zamknięcie wszystkich otwartych okien i wyczyszczenie zmiennych. Wczytano dane uczące z pliku . Wykorzystując sieć neuronową, obliczono wartości wyjściowe dla danych wejściowych i przypisano je do zmiennej . Przygotowano zmienne , i , które przechowują odpowiednio wartości kątów i dla wzorców oraz wartości wygenerowanych przez sieć. Obliczono błędy aproksymacji () jako różnicę pomiędzy wartościami kątów wzorcowych,   
a wygenerowanych przez sieć. Obliczono sumaryczny błąd średniokwadratowy () na podstawie błędów aproksymacji. Wygenerowano wykresy porównujące wartości kątów i wzorcowych oraz wygenerowanych przez sieć. Wykresy te przedstawiają wartości kątów i , oraz wartości kątów i . Wygenerowano również wykresy przedstawiające błędy aproksymacji dla wartości kątów i . Na podstawie tych kątów stworzono trajektorię manipulatora dla sieci neuronowej. Wygenerowana trajektoria została porównana z trajektorią wzorcową poprzez nałożenie na siebie obu wartości na wykresie.

* script3.m (sprawdzanie jakości działania sieci dla trajektorii testowej)

close all

clear

load dane\_ucz.mat

xab = [0:0.05:1];

yab = [0:-0.05:-1];

xbc = ones(1, 21);

ybc = [-1:0.05:0];

xt = [xab xbc];

yt = [yab ybc];

% rozszerzenie o punkt D

xcd = [1:-0.05:0]

ycd = [0:-0.05:-1];

xda = zeros(1, 21);

yda = [-1:0.05:0];

xt = [xab xbc xcd xda];

yt = [yab ybc ycd yda];

figure

plot(xt,yt,'ro-')

PT = [xt; yt];

YT = sim(net, PT);

alfast = YT(1,:);

betast = YT(2,:);

[alfat, betat] = odwrkin(xt, yt, l1, l2);

figure

plot(alfat,'ro-')

hold on

plot(alfast,'b\*-')

plot(betat,'go-')

plot(betast,'m\*-')

xlabel('Indeks próbki')

ylabel('Wartość kąta')

title('Porównanie wartości kątów α i ß dla trajektorii testowej')

legend({'Wartości kątów α (wzorzec)', 'Wartości kątów α (sieć)', 'Wartości kątów ß (wzorzec)', 'Wartości kątów ß (sieć)'}, 'Location', 'east')

[xts, yts] = prostkin(alfast, betast, l1, l2);

figure(1)

hold on

plot(xts, yts, 'bo-')

xlabel('Współrzędna x')

ylabel('Współrzędna y')

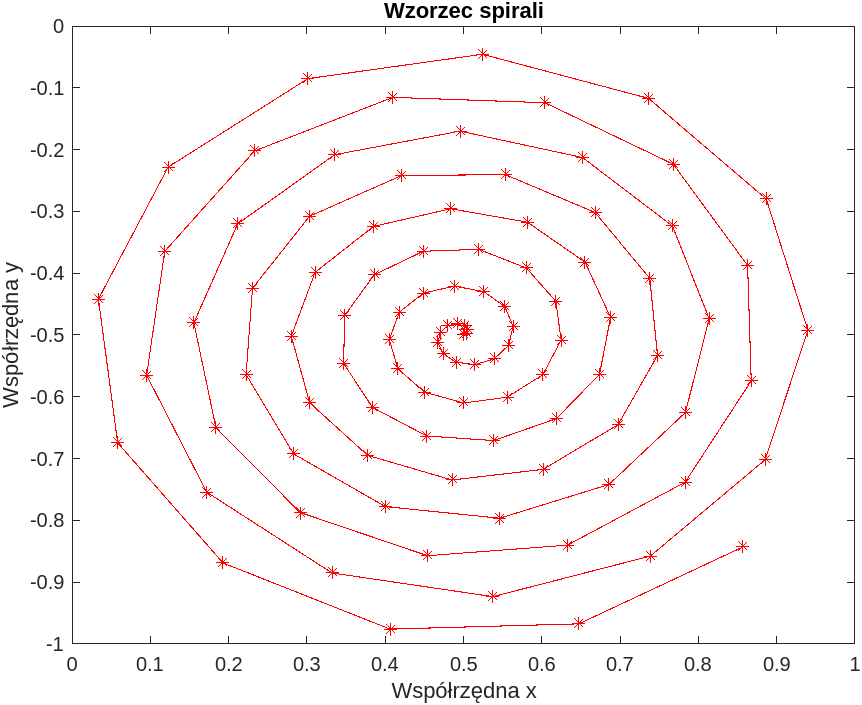
title('Porównanie trajektorii wzorcowej i wygenerowanej przez sieć')

legend({'Trajektoria wzorcowa', 'Trajektoria wygenerowana przez sieć neuronową'}, 'Location', 'north')

W tej części kodu sprawdzono działanie sieci neuronowej dla nowej trajektorii manipulatora, rozpoczynając od zamknięcia wszystkich otwartych okien  
i wyczyszczenia zmiennych. Następnie wczytano dane uczące z pliku . Na początku, zdefiniowano wektory łączące kolejne punkty łącząc je po kolei – z , z itd. Następnie, połączono wszystkie utworzone punkty , , , , tworząc trajektorię testową. Wartości i trajektorii testowej zostały przypisane do zmiennych i .

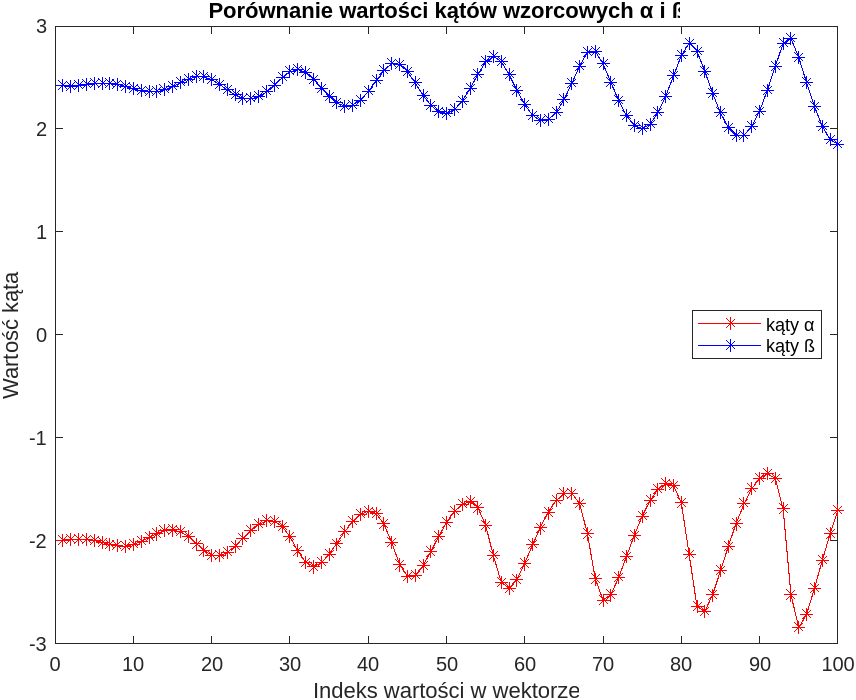
Dokonano symulacji sieci neuronowej, przekazując trajektorię testową jako dane wejściowe do sieci. Otrzymano wyniki kątów i . Obliczono różnicę pomiędzy wartościami i dla trajektorii wzorcowej a wartościami i dla trajektorii generowanej przez sieć neuronową. Następnie utworzono wykresy porównujące wartości kątów i dla trajektorii wzorcowej i generowanej przez sieć. Wykresy te zostały podpisane i opatrzone legendą. Na koniec, wygenerowano wykres porównujący trajektorię wzorcową  
i trajektorię wygenerowaną przez sieć neuronową.

* 1. **Otrzymane wyniki.**



*Rys. 2. Wykres przedstawiający wzorzec spirali.*

W ramach przeprowadzonego ćwiczenia podstawową sprawą było utworzenie sieci neuronowej, której zadaniem będzie operowanie chwytakiem manipulatora. Z pomocą podstawowych właściwości kinematyki sieć miała zautomatyzować rysowanie wzoru spirali (przedstawionej na rys. 2).



*Rys. 2. Porównanie wartości kątów wzorcowych oraz , będących kątami między  
przegubami manipulatora a osią układu kartezjańskiego.*

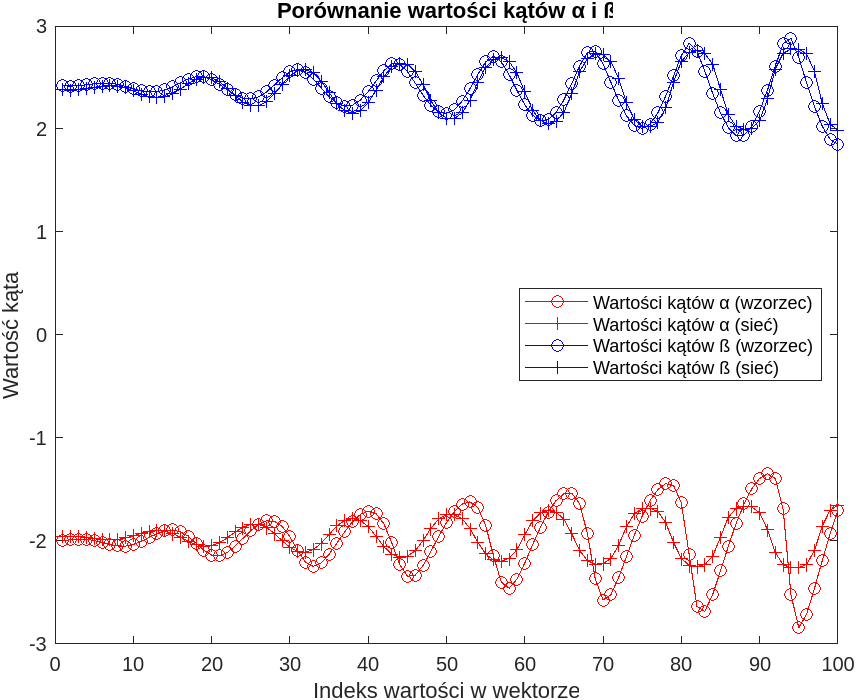
Działanie jednokierunkowej sieci neuronowej zostało przetestowane dla trzech ilości neuronów w jednej warstwie ukrytej. Wyznaczone zostały dla nich wartości sumaryczne błędów średniokwadratowych (tabela 1) oraz porównane zostały różnice w uzyskanych trajektoriach pomiędzy oryginalną spiralą - uzyskaną za pomocą równań kinematyki - a spiralą wyrysowaną przez sieć neuronową. Przyjęto dodatkowo następujące parametry treningowe sieci: cel – , epoki - oraz błąd maksymalny równy ilości epok.

*Tabela 1 – porównanie SSE i wydajności sieci w zależności od ilości neuronów.*

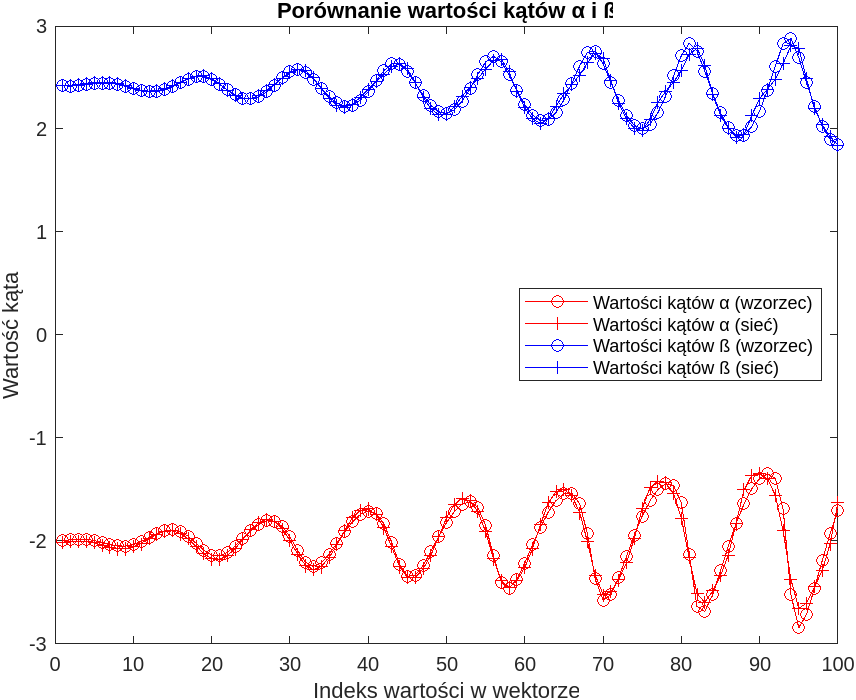
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Liczba neuronów sieci | | | |
|  |  |  | **10\*** |
| **SSE** |  |  |  | 0.002785 |
| **NAJLEPSZA WYDAJNOŚĆ**  **TRENINGOWA** | Przy 148 epokach | Przy 500 epokach | Przy 491 epokach | 0.0041512  Przy 489 epokach |

**\* - zadanie dodatkowe**

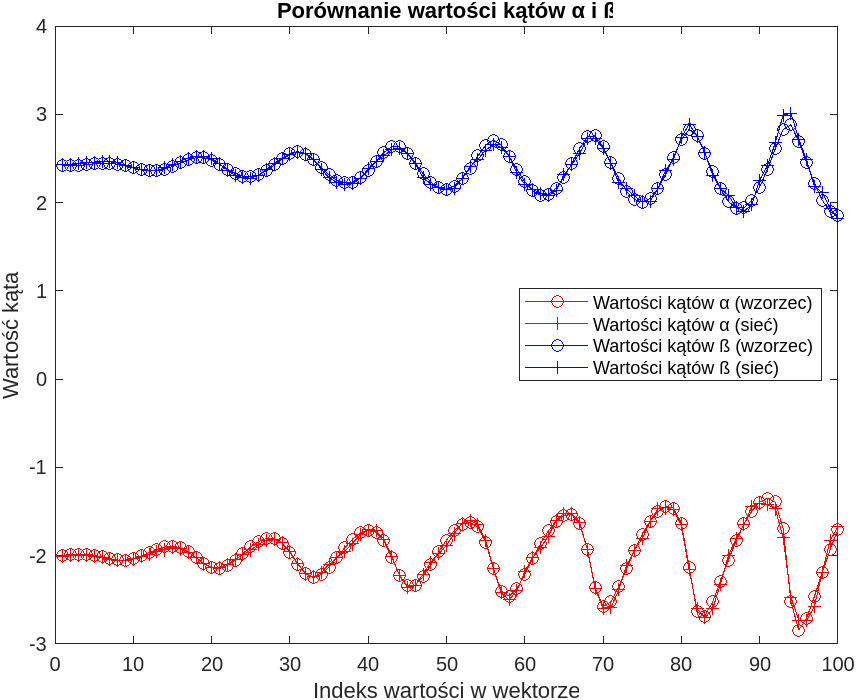
Otrzymano następujące wartości kątów przy pomocy sieci neuronowych:



*Rys. 3. Porównanie wartości kątów wzorcowych oraz uzyskanych przez sieć ( i dla 1 neuronu.*

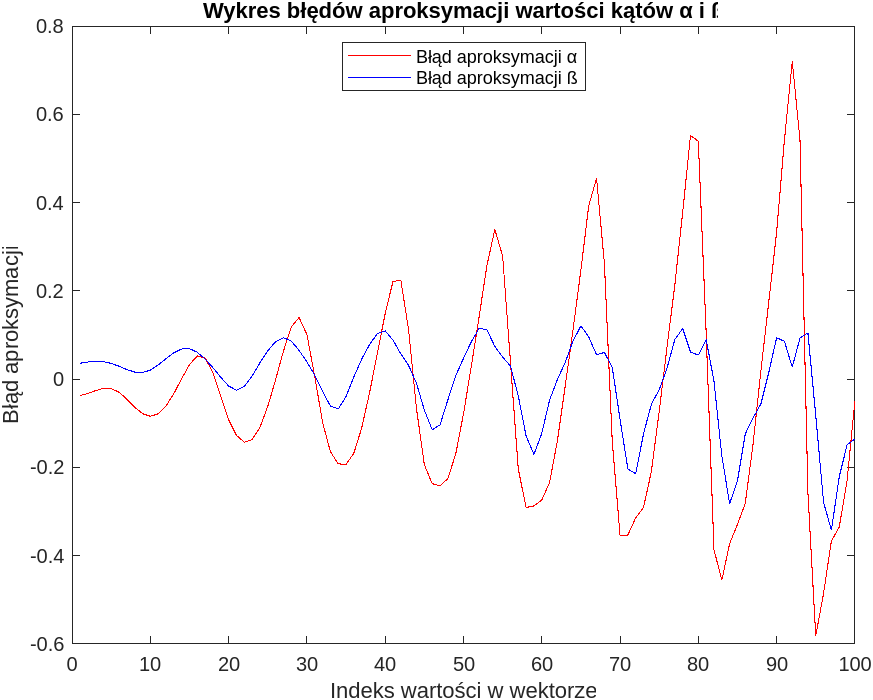


*Rys. 4. Porównanie wartości kątów wzorcowych oraz uzyskanych przez sieć ( i dla 2 neuronów.*

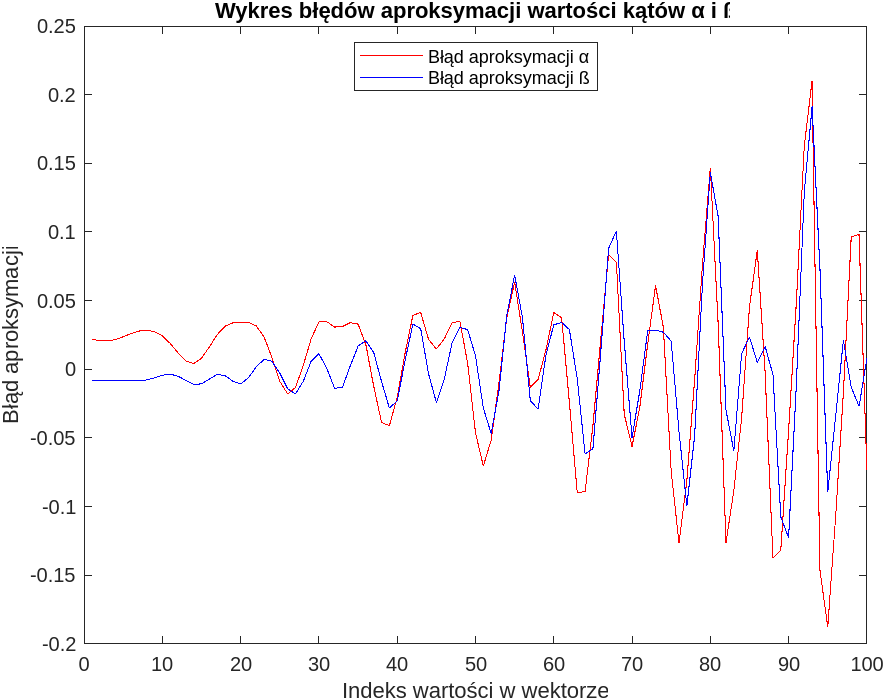


*Rys. 5. Porównanie wartości kątów wzorcowych oraz uzyskanych przez sieć ( i dla 5 neuronów.*

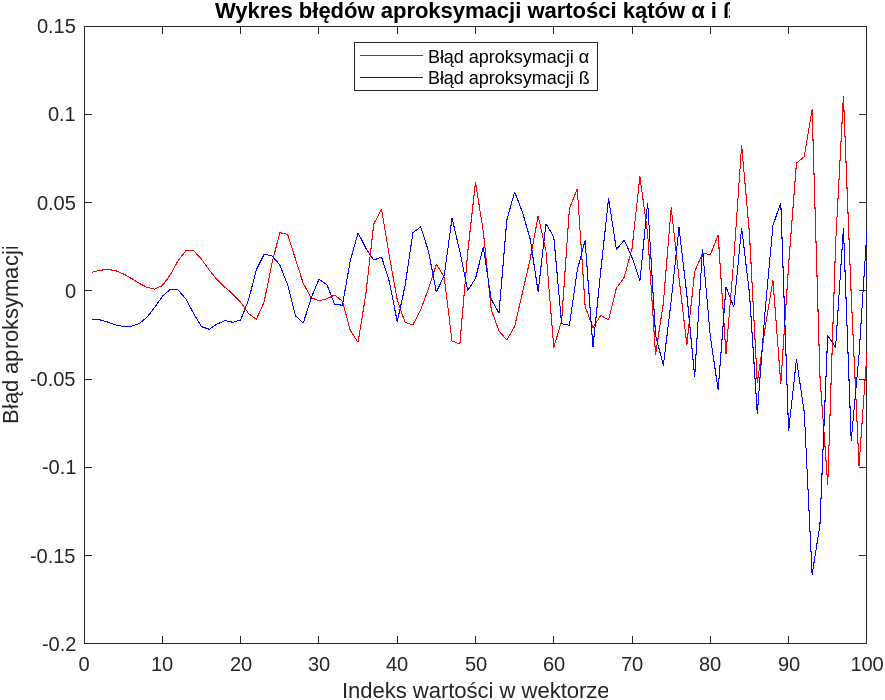
Uzyskane błędy pomiędzy wartościami wzorcowymi, a uzyskanymi przy pomocy sieci neuronowej zostały obliczone i naniesione na wykresy, rozróżniając przy tym błędy dla kątów i .



*Rys. 6. Wykresy błędów aproksymacji wartości kątów i dla sieci z 1 neuronem.*

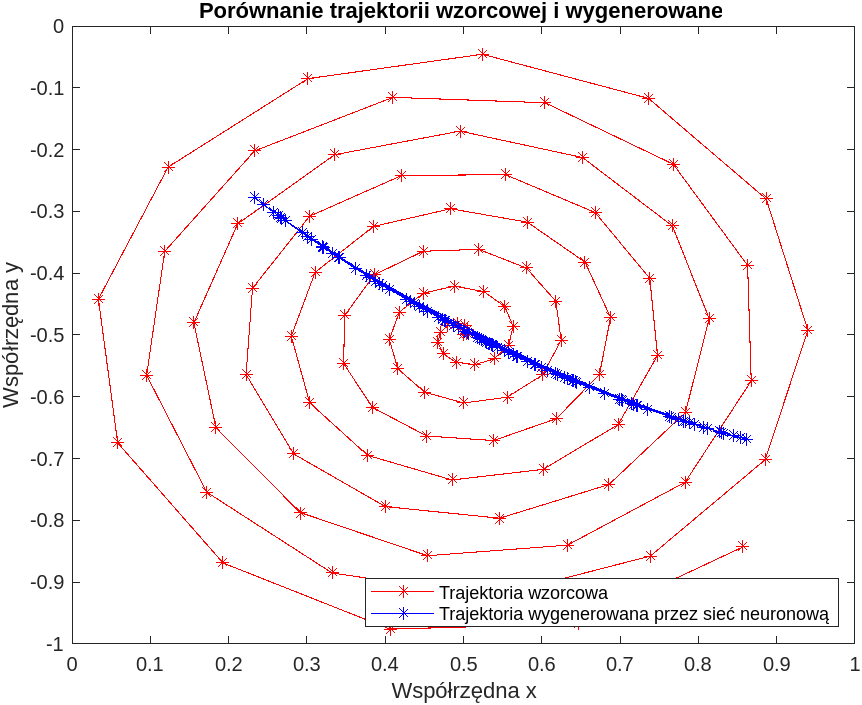


*Rys. 7. Wykresy błędów aproksymacji wartości kątów i dla sieci z 2 neuronami.*

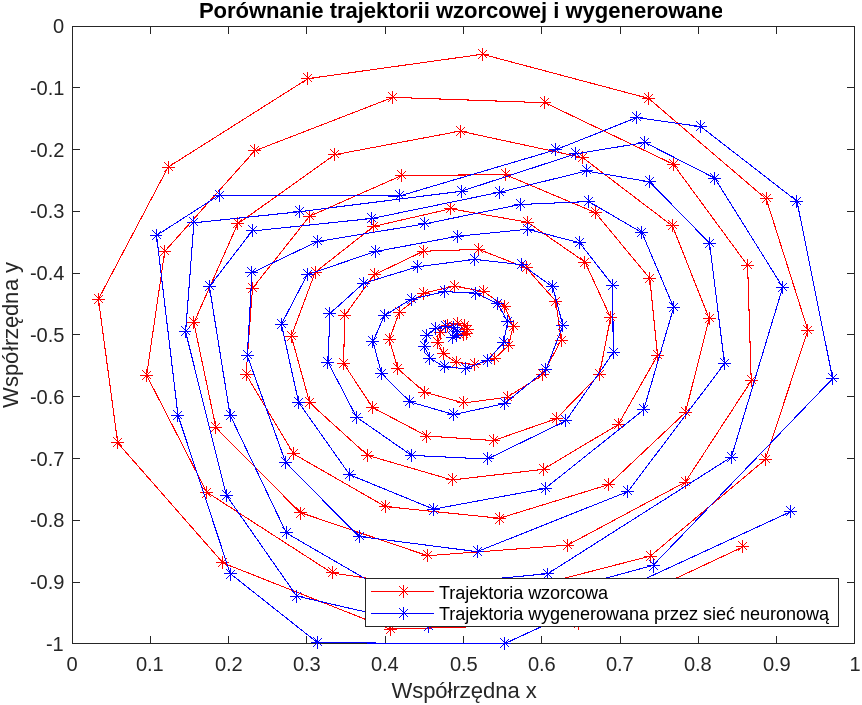


*Rys. 8. Wykresy błędów aproksymacji wartości kątów i dla sieci z 5 neuronami.*

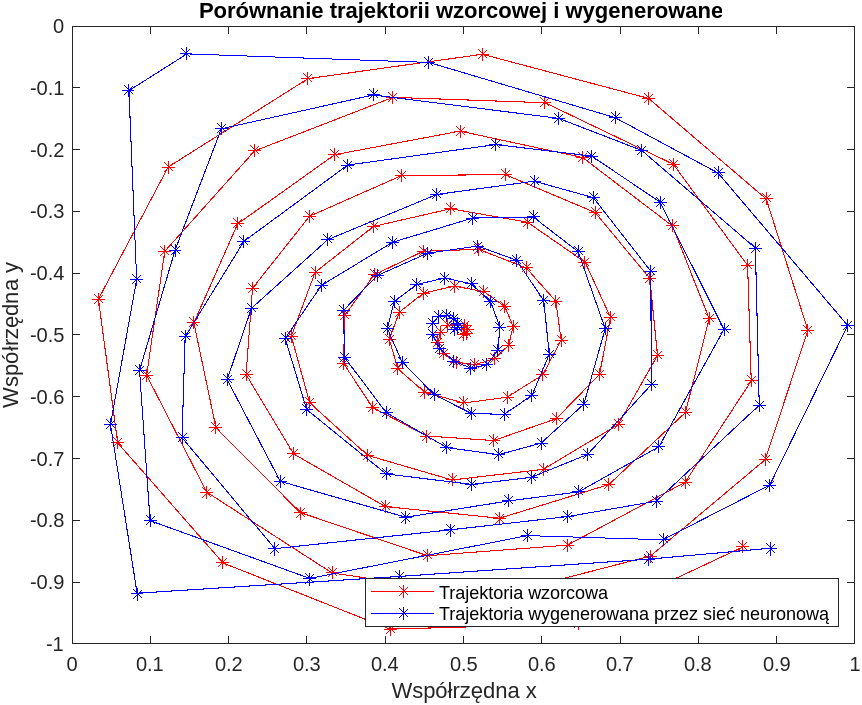
Najważniejszym jednak było utworzenie wykresów przedstawiających naniesione punkty – wyznaczone przez sieć – w porównaniu ze spiralą wzorcową. Jest na nich najbardziej widoczne jak bardzo różni się trajektoria obiektu obliczonego kinematycznie oraz przez sztuczną inteligencję. Przy zastosowaniu jednego neuronu widoczny jest problem z wygenerowaniem spirali, powstało coś podobnego do krzywej.



*Rys. 9. Porównanie spirali wzorcowej ze spiralą narysowaną przez sieć  
neuronową – przy 1 neuronie.*

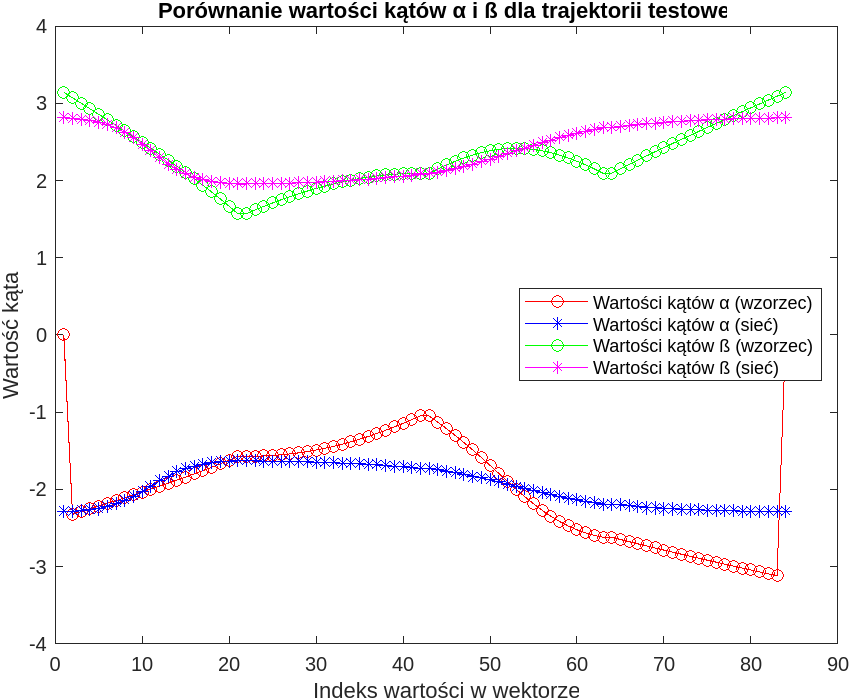


*Rys. 10. Porównanie spirali wzorcowej ze spiralą narysowaną przez sieć  
neuronową – przy 2 neuronach.*

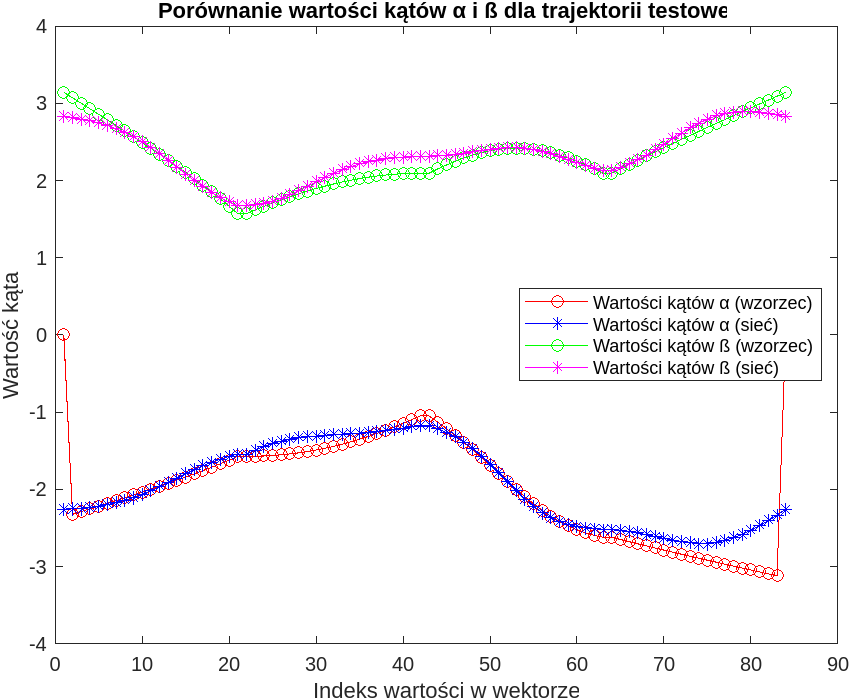
**

*Rys. 11. Porównanie spirali wzorcowej ze spiralą narysowaną przez sieć  
neuronową – przy 5 neuronach.*

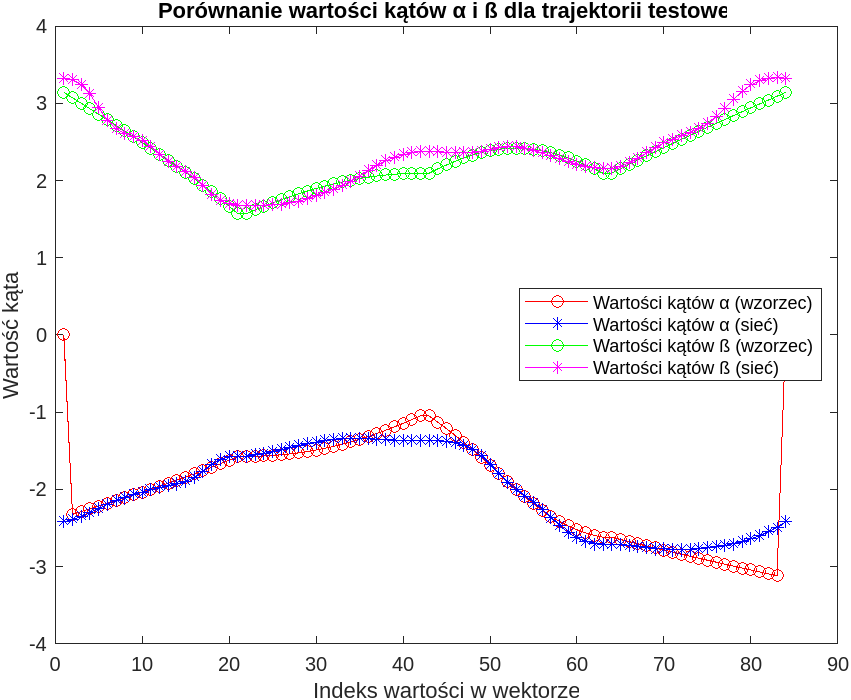
Kolejnym krokiem ćwiczenia było utworzenie nowej trajektorii oraz przetestowanie dla niej wcześniej wyuczonej na spirali sieci. Przyjęto wzór przypominający klepsydrę, na podstawie czterech punktów oraz, z których utworzono wektory łączące. Następnie przetestowano sieć do odtworzenia wykonanej trajektorii wzorcowej.



*Rys. 12. Porównanie wartości kątów i wzorcowych i uzyskanych dla N=1.*

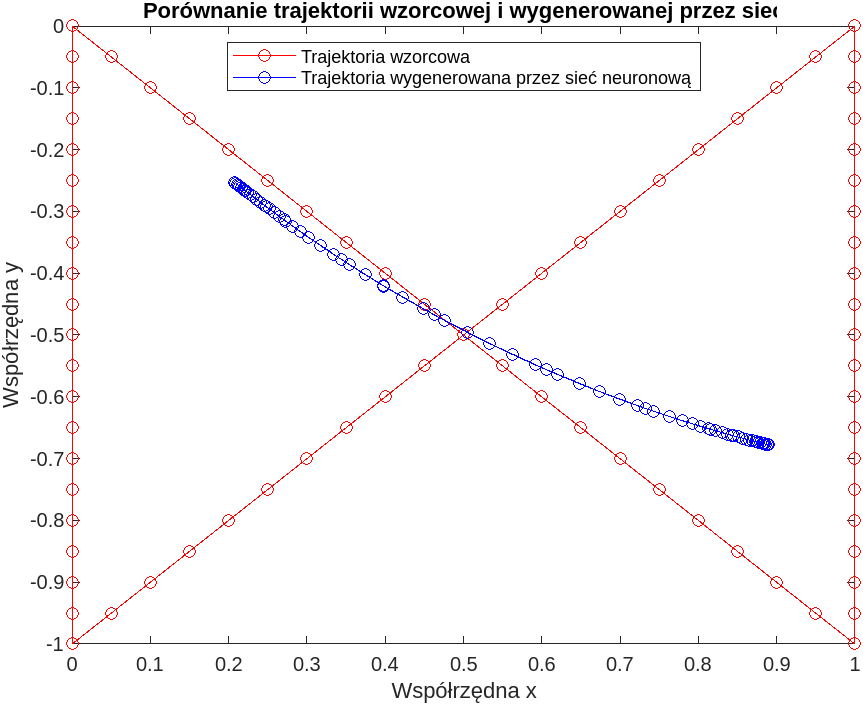


*Rys. 13. Porównanie wartości kątów i wzorcowych i uzyskanych dla N=2.*

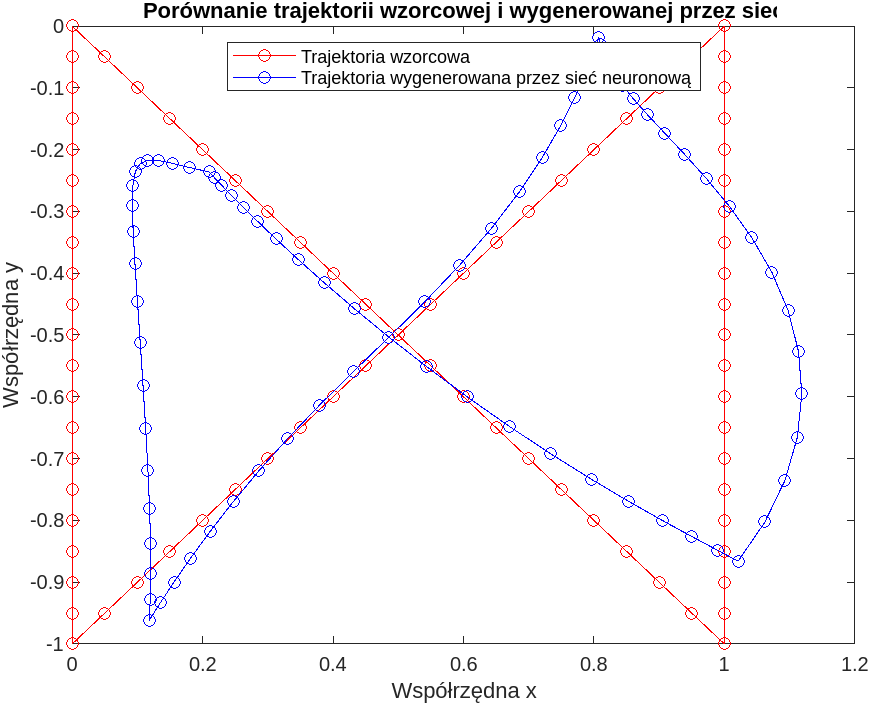


*Rys. 14. Porównanie wartości kątów i wzorcowych i uzyskanych dla N=5.*

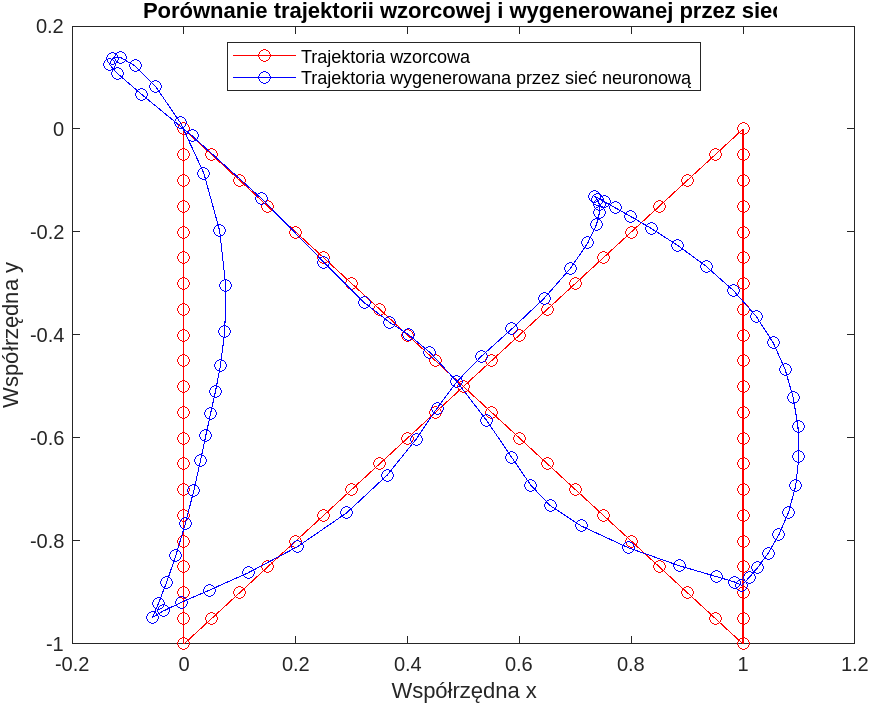
Uzyskano następujące trajektorie przypominające klepsydrę (podobnie, jak w przypadku spirali dla jednego neuronu sieć nie była w stanie poprawnie wygenerować wzoru):



*Rys. 15. Porównanie klepsydry wzorcowej z narysowaną przez sieć  
neuronową – przy 1 neuronie.*



*Rys. 16. Porównanie klepsydry wzorcowej z narysowaną przez sieć  
neuronową – przy 2 neuronach.*

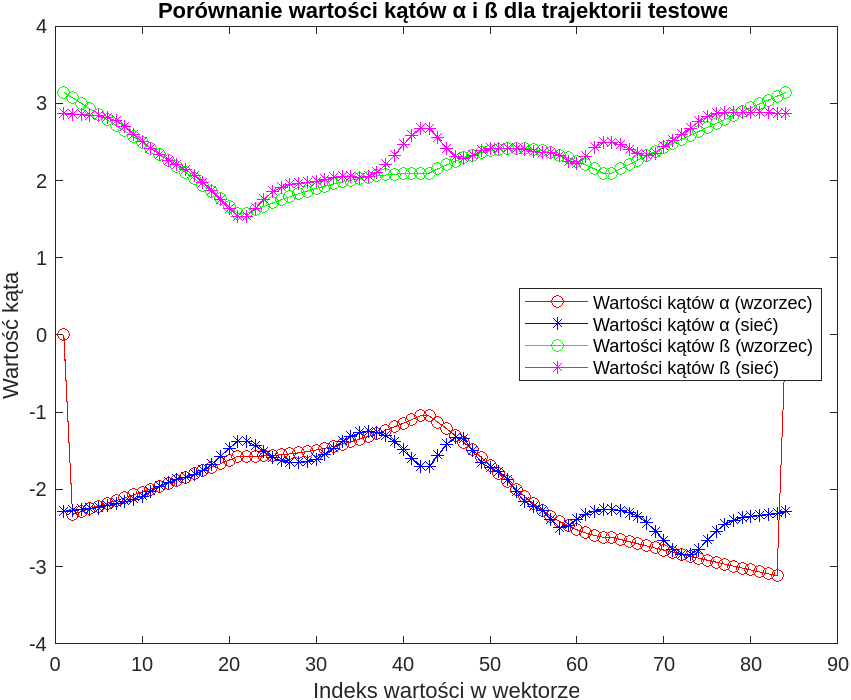
**

*Rys. 17. Porównanie klepsydry wzorcowej z narysowaną przez sieć  
neuronową – przy 5 neuronach.*

* **ZADANIE DODATKOWE**

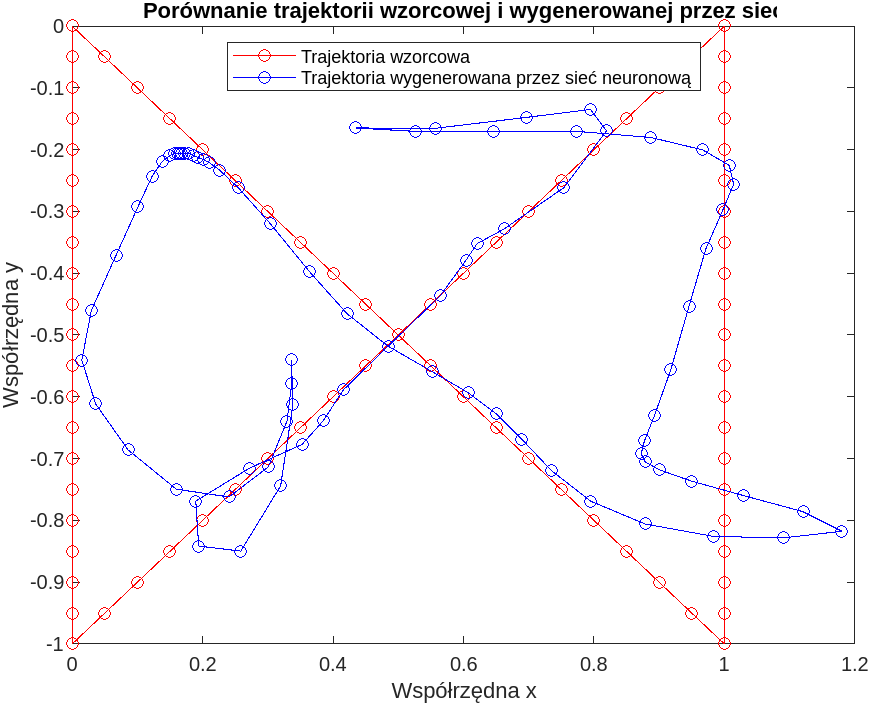
W ramach zadania dodatkowego zademonstrowane zostanie zjawisko nadparametryzacji (z ang. overfitting) polegające na niekorzystnym wpływie na właściwości uogólniania sieci. Zadanie zostanie przedstawione na przykładzie trajektorii testowej – klepsydrze. Zgodnie z tabelą 1, zauważalny jest nagły wzrost sumarycznego błędu średniokwadratowego w przypadku nagłego zwiększenia ilości neuronów sieci. Występuje wtedy właśnie, wyżej wspomniane zjawisko nadparametryzacji – sieć zaczyna zamiast minimalizować błędy względem wzorca, drastycznie je maksymalizować.

Tak też, dla pięciu neuronów SSE wynosił w przybliżeniu i był  
w tendencji malejącej. Natomiast po podwojeniu ilości neuronów wartość błędu wzrosła aż do , co stanowi praktycznie dwukrotność.



*Rys. 18. Porównanie wartości kątów i wzorcowych i uzyskanych dla N=10.*

Na wykresie (rys. 18) zauważalne jest pojawianie się dużych błędów przy nagłych zmianach wartości kątów oraz manipulatora. Dla tej ilości neuronów, przy prostych odcinkach rysowania, bez konieczności stosowania dużej różnicy w kącie między dwoma punktami – błędy są minimalne, a rosną przy nagłych zmianach kierunku wektora.



*Rys. 19. Porównanie klepsydry wzorcowej z narysowaną przez sieć  
neuronową – przy 10 neuronach.*

Na rysunku 19 natomiast widoczne jest pojawianie się dużych rozbieżności w kształcie „klepsydry”. Pojawiają się nadmiarowe, ostre zmiany kierunku przy punktach oraz .

1. **Wnioski.**

W trakcie przeprowadzonego ćwiczenia dotyczącego jednokierunkowej sieci neuronowej jako aproksymatora funkcji w odwrotnym zadaniu kinematyki, zrealizowano trzy skrypty w oprogramowaniu Matlab. Głównym celem tych skryptów było stworzenie jednokierunkowej sieci neuronowej, która mogłaby zostać wykorzystana do sterowania manipulatorem dwuczłonowym. Sieć neuronowa miała za zadanie odtworzyć wzorcową trajektorię spirali, uzyskaną przy użyciu kinematyki manipulatora. Ponadto, przeprowadzono testy sieci na innej trajektorii wzorcowej, która miała kształt klepsydry. Dodatkowo, podjęto zadanie dodatkowe, które miało na celu zbadanie zjawiska nadparametryzacji, prowadzącego do nieprawidłowego działania jednokierunkowej sieci neuronowej.

Przeprowadzone eksperymenty dostarczyły istotnych wniosków dotyczących zastosowania jednokierunkowej sieci neuronowej jako aproksymatora. Po przeprowadzeniu procesu uczenia sieci z wykorzystaniem wzorców uczących, sieć była w stanie poprawnie odwzorować trajektorię spirali oraz klepsydry na podstawie zadanych wartości współrzędnych chwytaka. Porównanie wygenerowanych przez sieć wartości kątów przegubów z wartościami wzorcowymi wykazało zadowalające rezultaty, potwierdzając skuteczność sieci w odtwarzaniu ruchów manipulatora na podstawie danych wejściowych.

Jednakże, przetestowanie sieci ze zjawiskiem nadparametryzacji - na trajektorii wzorcowej o kształcie klepsydry, wykazało, że jednokierunkowa sieć neuronowa nie była w stanie dokładnie odwzorować ruchów manipulatora na podstawie nowych danych wejściowych. Wygenerowane przez sieć wartości kątów przegubów nie odpowiadały wartościom wzorcowym, co oznaczało, że sieć nie potrafiła dobrze uogólnić swojej wiedzy na nowe przypadki.

Wniosek wynikający z powyższych obserwacji jest taki, że jednokierunkowa sieć neuronowa może być skutecznym aproksymatorem funkcji w odwrotnym zadaniu kinematyki dla znanych wzorców uczących. Jeśli jednak sieć ma być wykorzystana do obsługi trajektorii, które wychodzą poza zakres wzorców uczących, należy zwrócić uwagę na problem nadparametryzacji. Zjawisko to może prowadzić do nieprawidłowych wyników i konieczne może być zastosowanie innych wartości trenujących sieć lub innych metod uczenia maszynowego.

Podsumowując, ćwiczenie dotyczące jednokierunkowej sieci neuronowej jako aproksymatora funkcji w odwrotnym zadaniu kinematyki dostarczyło istotnych wniosków. Sieć neuronowa okazała się skutecznym narzędziem w odwzorowywaniu ruchów manipulatora na podstawie znanych wzorców uczących. Jednakże, zjawisko nadparametryzacji stanowiło wyzwanie przy próbie obsługi trajektorii poza zakresem wzorców uczących. Aby osiągnąć lepsze rezultaty, należy dokładnie dobierać właściwości sieci dla każdego zadania stawianego przed nią.