Politechnika Śląska **Gliwice**

Wydział Automatyki Elektroniki i Informatyki **Rok Akademicki 2010/2011**

Kierunek: Automatyka i Robotyka sem. 7 **Semestr zimowy**

Metody sztucznej inteligencji

Laboratorium

Ćw. 5: Sieci neuronowe jednokierunkowe

Wykonali:

Agnieszka Antczak
Leszek Owczarek

Grupa 3

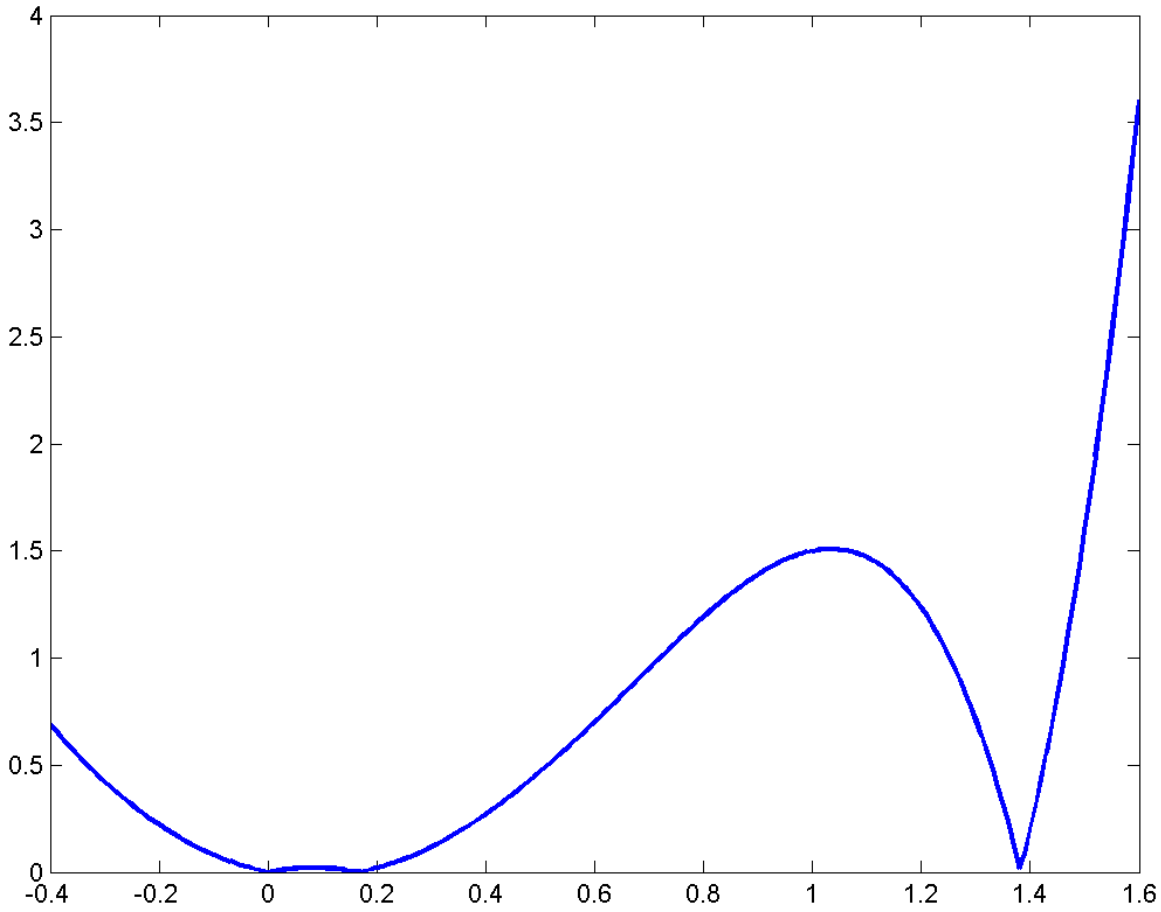
Data odbycia ćwiczenia:

15.12.2010

I. Przygotowanie danych

1. Funkcja nieliniowa

Do testowania sieci neuronowych jednokierunkowych wybrana została następująca funkcja:
$f(x) = \left|x^5 - 3x^2 + \frac{1}{2}x\right|$
Funkcja ta posiada 5 ekstremów lokalnych w przedziale $[-0.4,1.6]$, co widać na poniższym wykresie.



Rysunek 1: wykres funkcji w badanym przedziale

2. Zbiór uczący i testowy

Zbiór uczący składał się z punktów należących do wykresu funkcji f w przedziale $[-0.4,1.6]$, oddalonych od siebie na osi odciętych co 0.1. Zbiór uczący składał się więc z 21 punktów.

Zbiór testowy był analogiczny do zbioru testowego z tą różnicą, że punkty na osi odciętych są od siebie oddalone co 0.001. Zbiór uczący składał się więc z 2001 punktów.

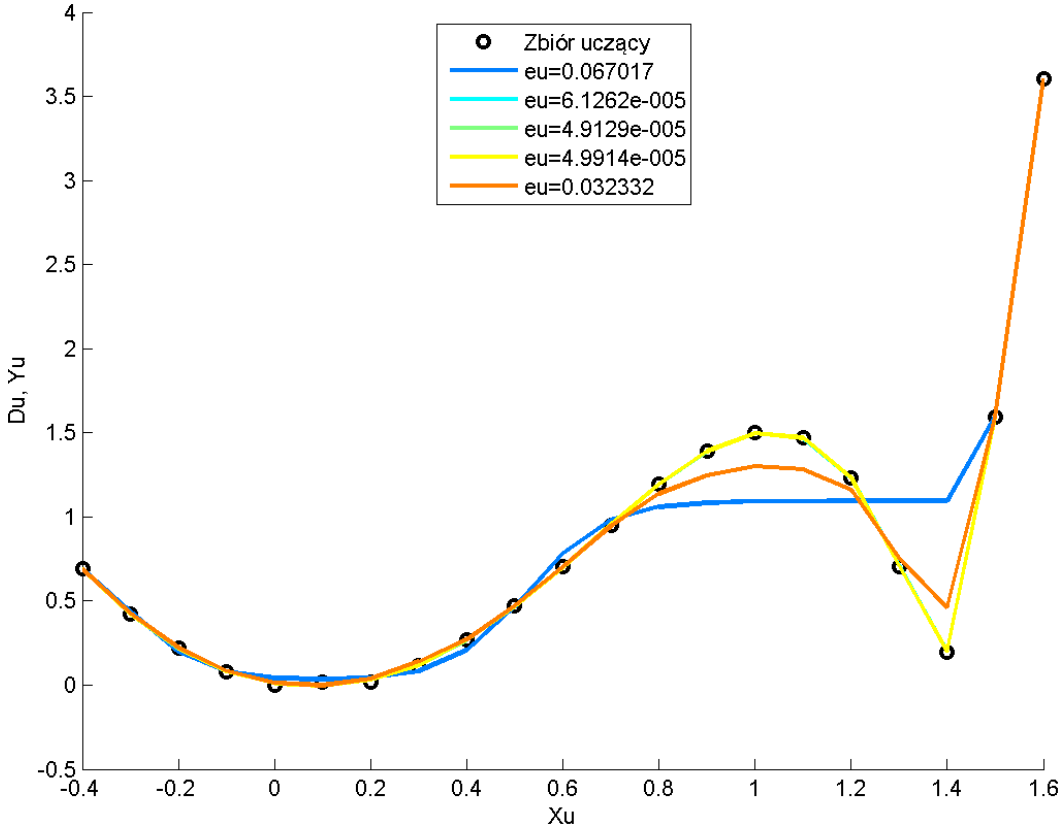
3. Utworzenie sieci

Po wygenerowaniu zbiorów uczącego i testowego utworzona została wielowarstwowa sieć neuronowa, której zadaniem była aproksymacja nieliniowej funkcji f na podstawie zbioru uczącego. Parametrami tej sieci są:
N – wektor zawierający liczby neuronów w poszczególnych warstwach sieci; domyślnie: $[10 \quad 1]$,
F – wektor komórkowy zawierający funkcje aktywacji poszczególnych warstw; domyślnie: $\{'tansig' \quad 'purelin'\}$,
M – ciąg tekstowy określający metodę uczenia sieci; domyślnie: $'trainlm'$,
e – liczba określająca maksymalną ilość epok; domyślnie: 100.

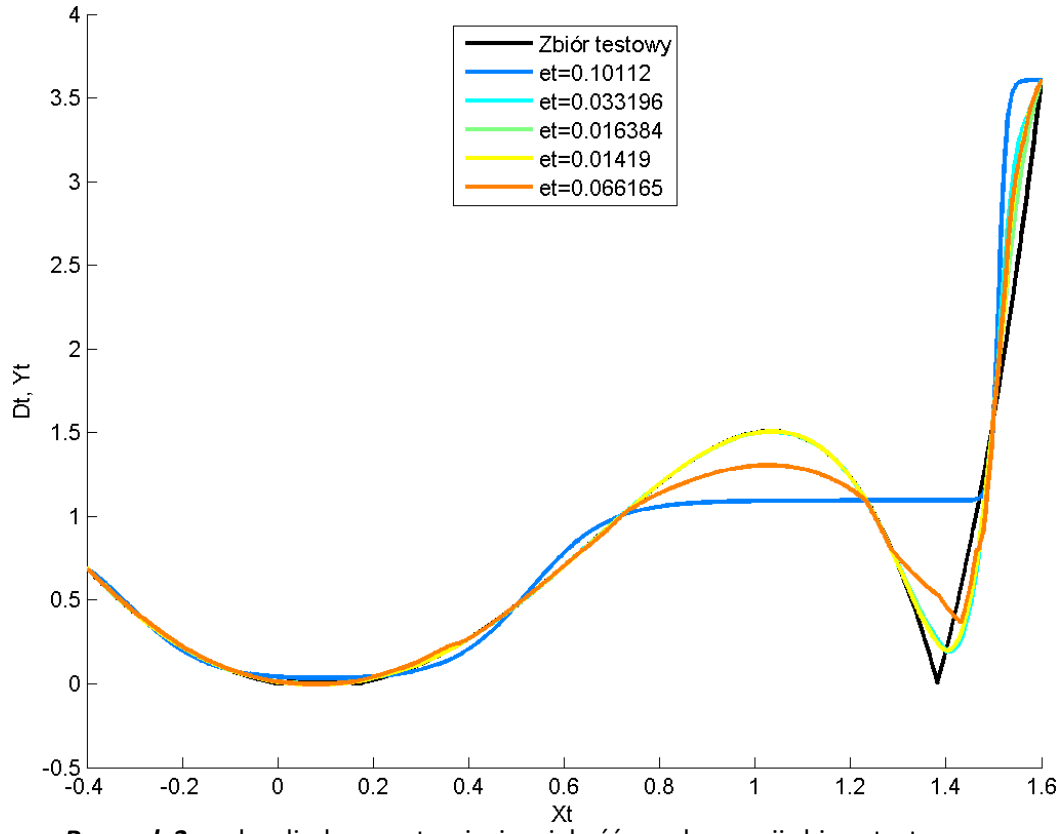
II. Wpływ parametrów na jakość aproksymacji

1. Liczba warstw sieci

Liczba ta zmieniana była kolejno od 2 do 6 warstw. Kolejność ta odpowiada kolorom linii i wartościom błędu średniokwadratowego umieszczonych w legendzie. Ostatnią warstwę stanowił 1 neuron, dla którego postać funkcji aktywacji to $'purelin'$, natomiast pozostałe warstwy zawierały 3 neurony i funkcję $'tansig'$.



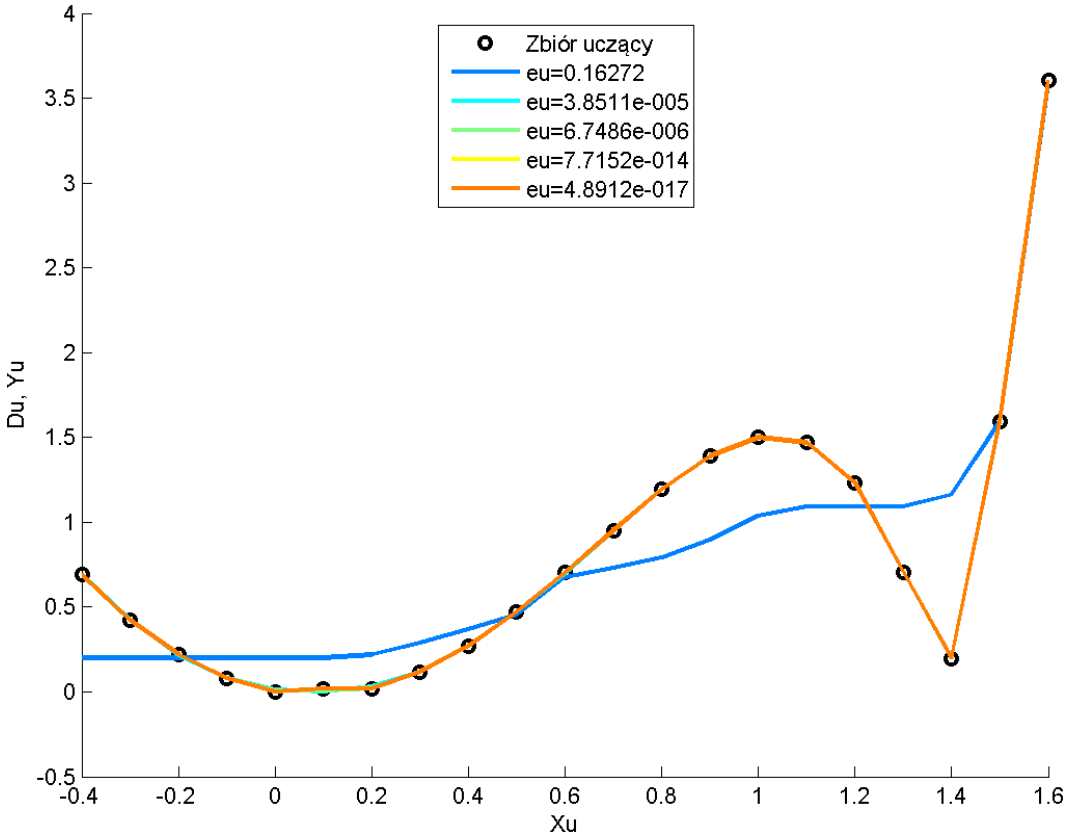
Rysunek 2: wpływ liczby warstw sieci na jakość aproksymacji zbioru uczącego



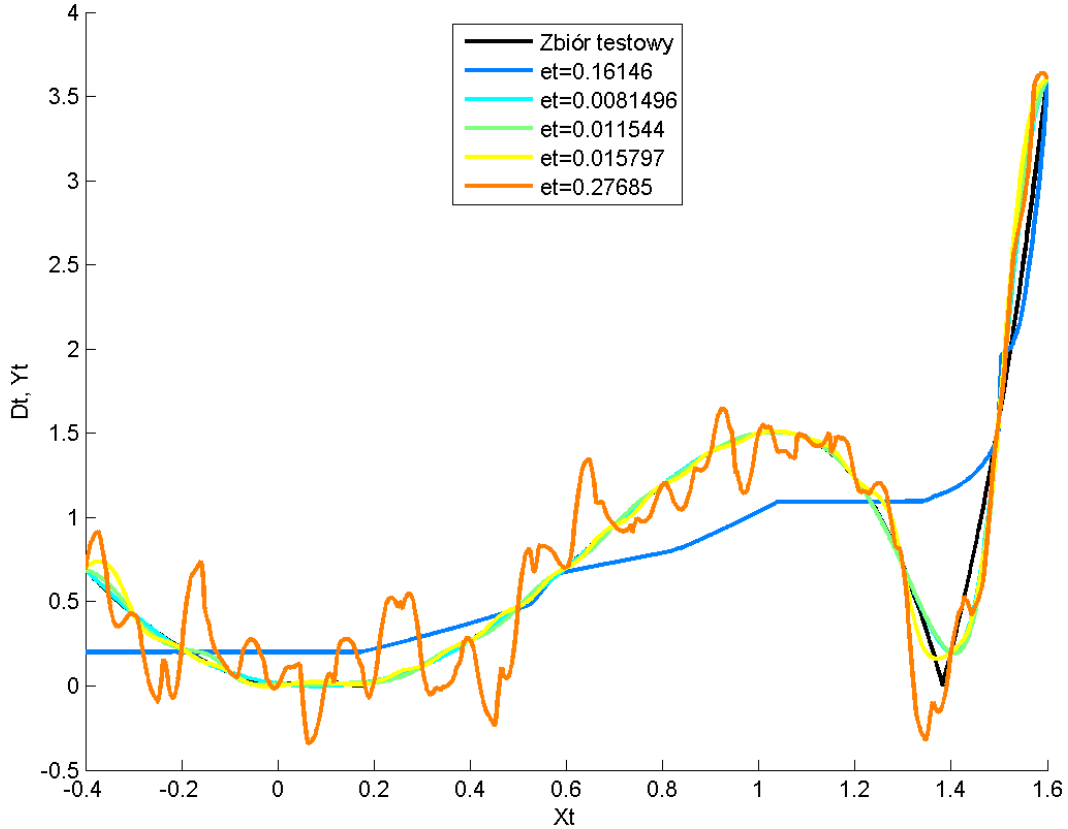
Rysunek 3: wpływ liczby warstw sieci na jakość aproksymacji zbioru testowego

2. Liczba neuronów w warstwach

Zmieniana była liczba neuronów tylko w pierwszej warstwie sieci. Liczba ta wynosiła kolejno: 2, 5, 10, 20 i 50. W drugiej warstwie w każdym przypadku znajdował się jeden neuron.



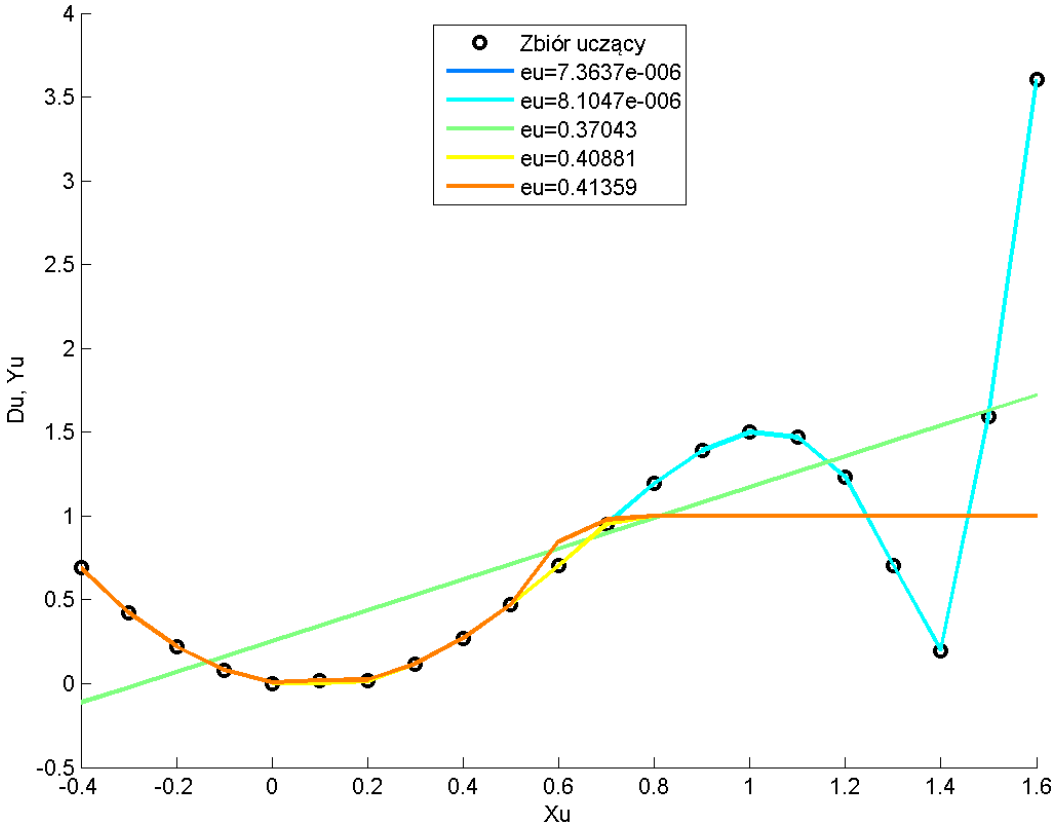
Rysunek 4: wpływ liczby neuronów w warstwach na jakość aproksymacji zbioru uczącego



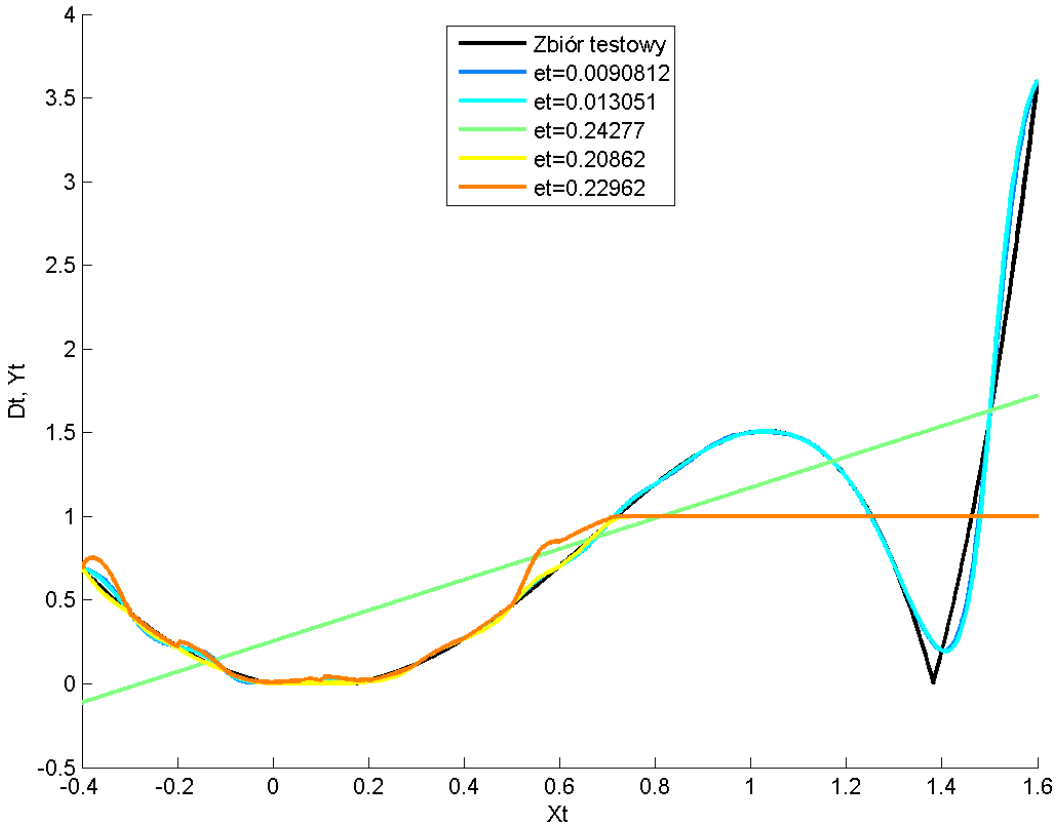
Rysunek 5: wpływ liczby neuronów w warstwach na jakość aproksymacji zbioru testowego

3. Postać funkcji aktywacji

Wpływ funkcji aktywacji badany był przy domyślnej liczbie warstw i neuronów. Przetestowane zostały postacie: $\{'tansig' \ 'purelin'\}$, $\{'logsig' \ 'purelin'\}$, $\{'purelin' \ 'purelin'\}$, $\{'tansig' \ 'logsig'\}$ oraz $\{'tansig' \ 'tansig'\}$.



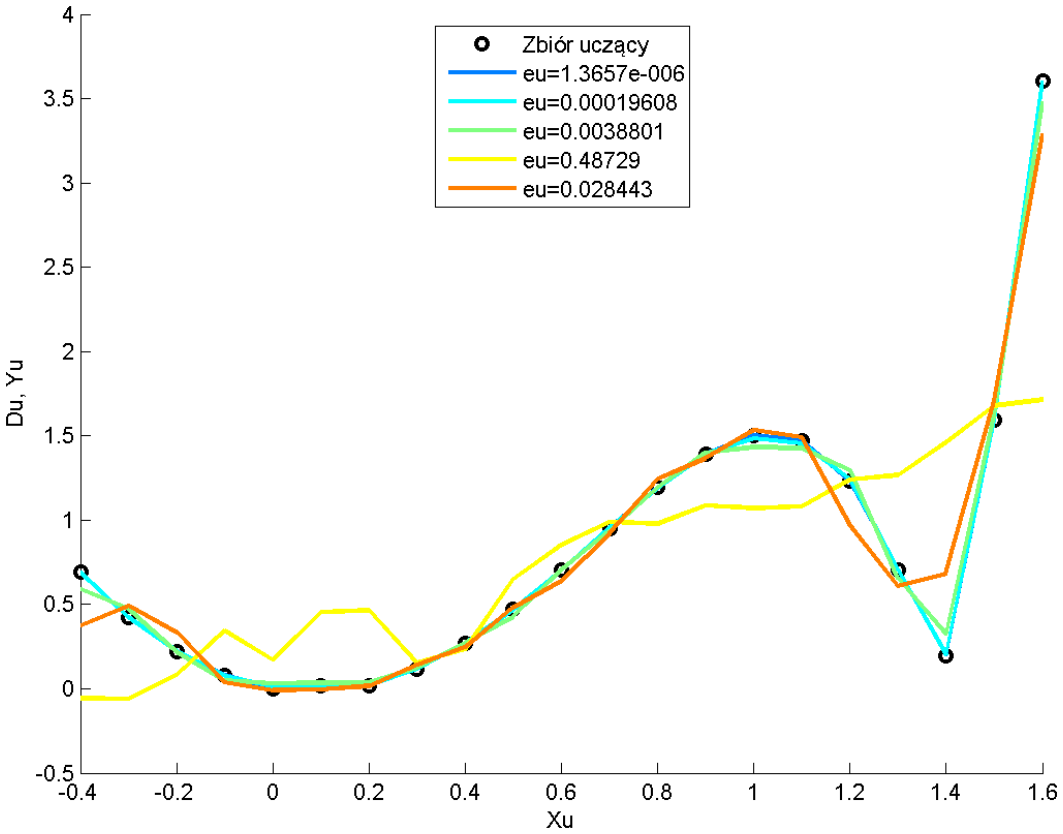
Rysunek 6: wpływ postaci funkcji aktywacji na jakość aproksymacji zbioru uczącego



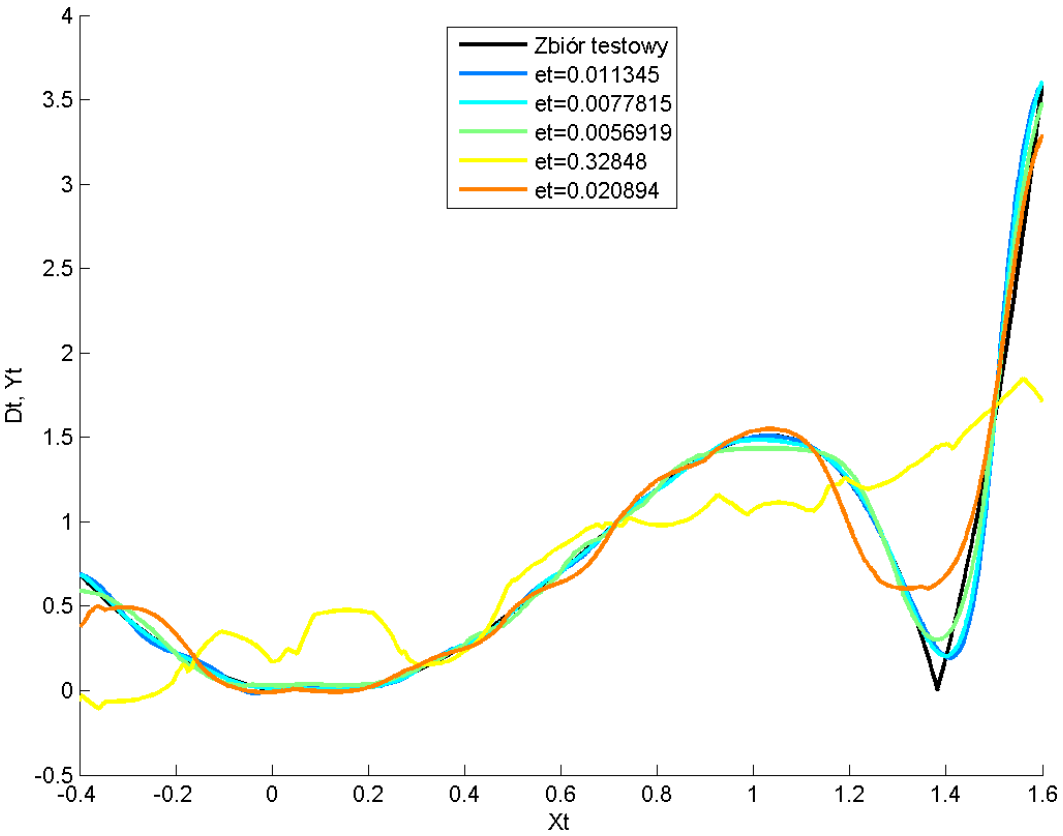
Rysunek 7: wpływ postaci funkcji aktywacji na jakość aproksymacji zbioru testowego

4. Wybór metody uczenia

Przetestowanych zostało 5 metod uczenia sieci: $'trainlm'$, $'trainbfg'$, $'trainrp'$, $'traingd'$ oraz $'traingdx'$.



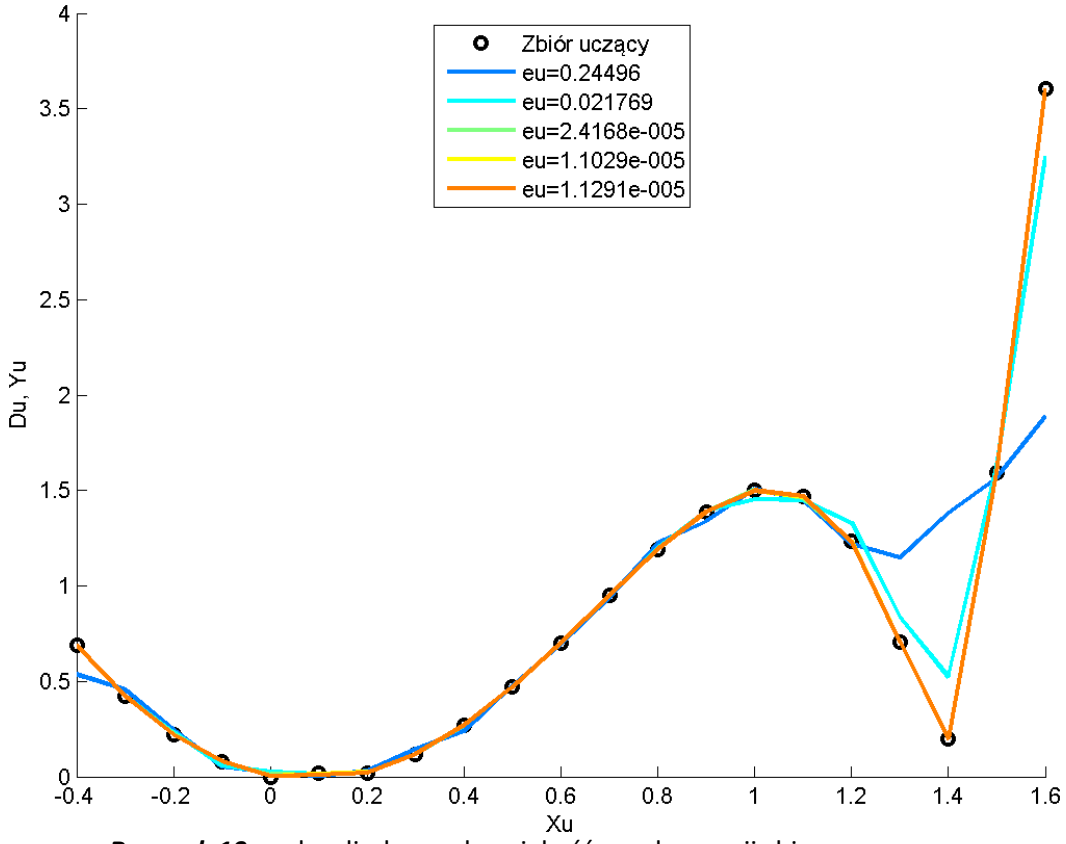
Rysunek 8: wpływ wyboru metody uczenia na jakość aproksymacji zbioru uczącego



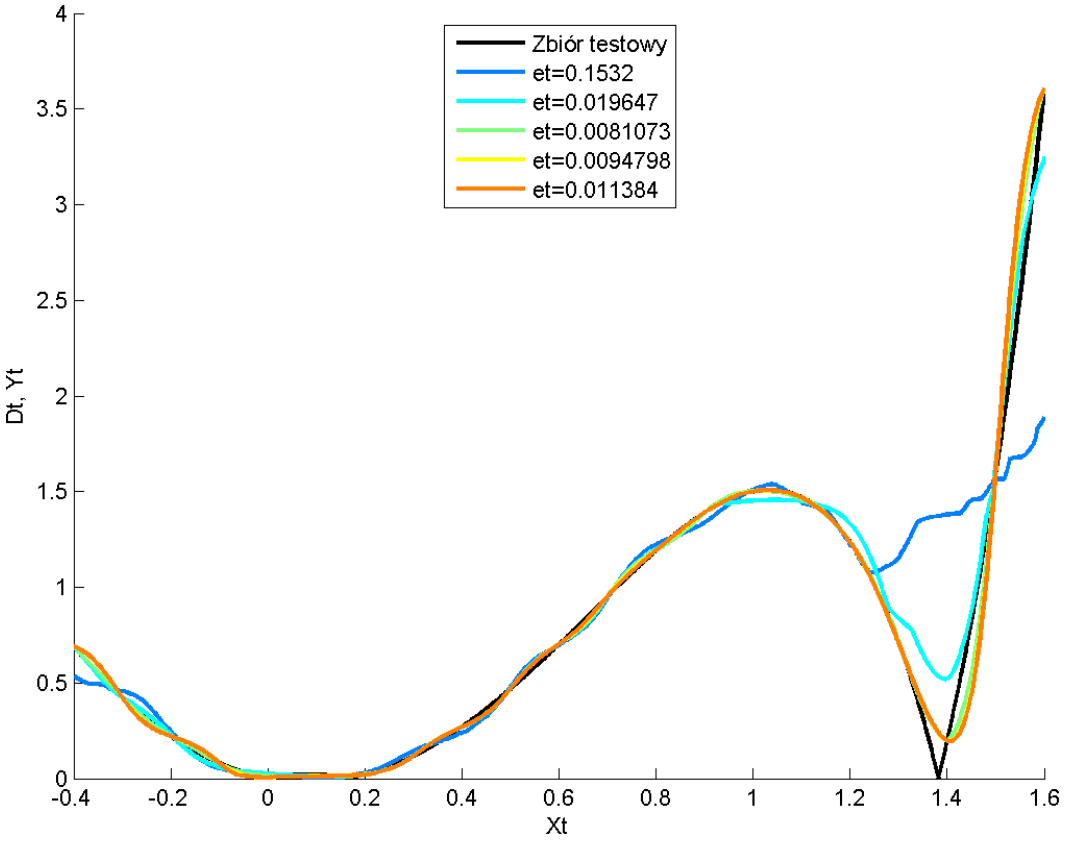
Rysunek 9: wpływ wyboru metody uczenia na jakość aproksymacji zbioru testowego

5. Liczba epok

Liczba epok zmieniana była podobnie jak liczba neuronów, a więc kolejno: 2, 5, 10, 20 i 50.



Rysunek 10: wpływ liczby epok na jakość aproksymacji zbioru uczącego

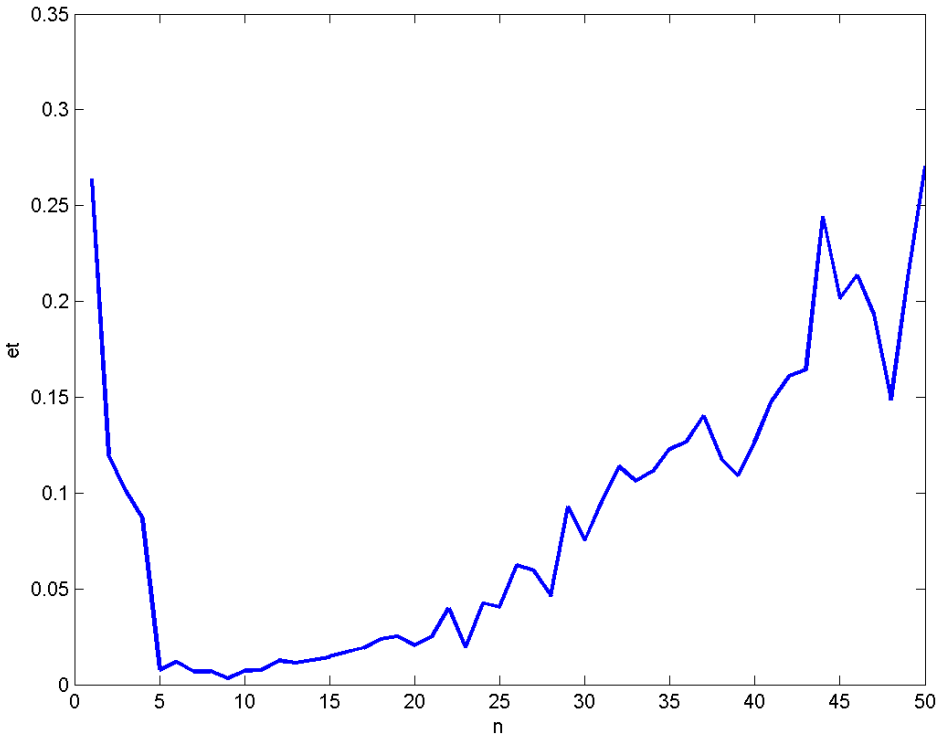


Rysunek 11: wpływ liczby epok na jakość aproksymacji zbioru testowego

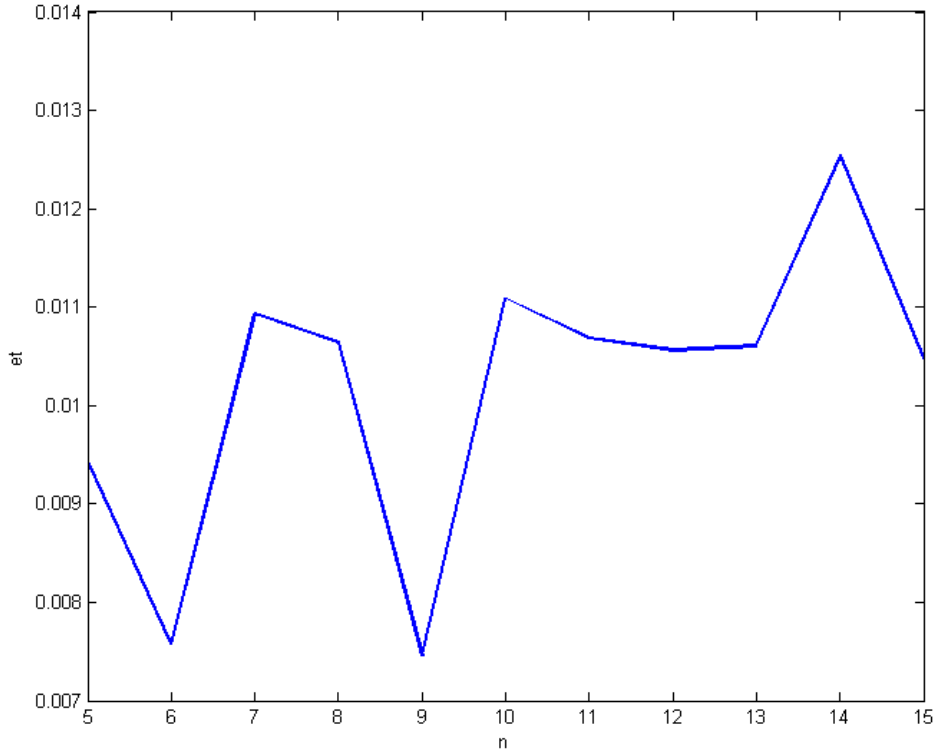
III. Wpływ liczby neuronów na jakość aproksymacji

1. Zbiór testowy

Zbadany został dokładny wpływ liczby neuronów w pierwszej warstwie sieci na jakość aproksymacji zbioru testowego. Testowanie odbyło się podobnie jak w punkcie II. 2. z tą różnicą, że liczba neuronów zmieniała się od 1 do 50 krokiem 1, powtarzając każdorazowo proces uczenia trzykrotnie i obliczając medianę z tych trzech wyników. Wskaźnikiem jakości był, podobnie jak wcześniej, błąd średniokwadratowy aproksymacji. Aby łatwiej znaleźć minimum tego wykresu, powtórzono operację dla liczby neuronów od 5 do 15, powtarzając proces uczenia trzydziestokrotnie.



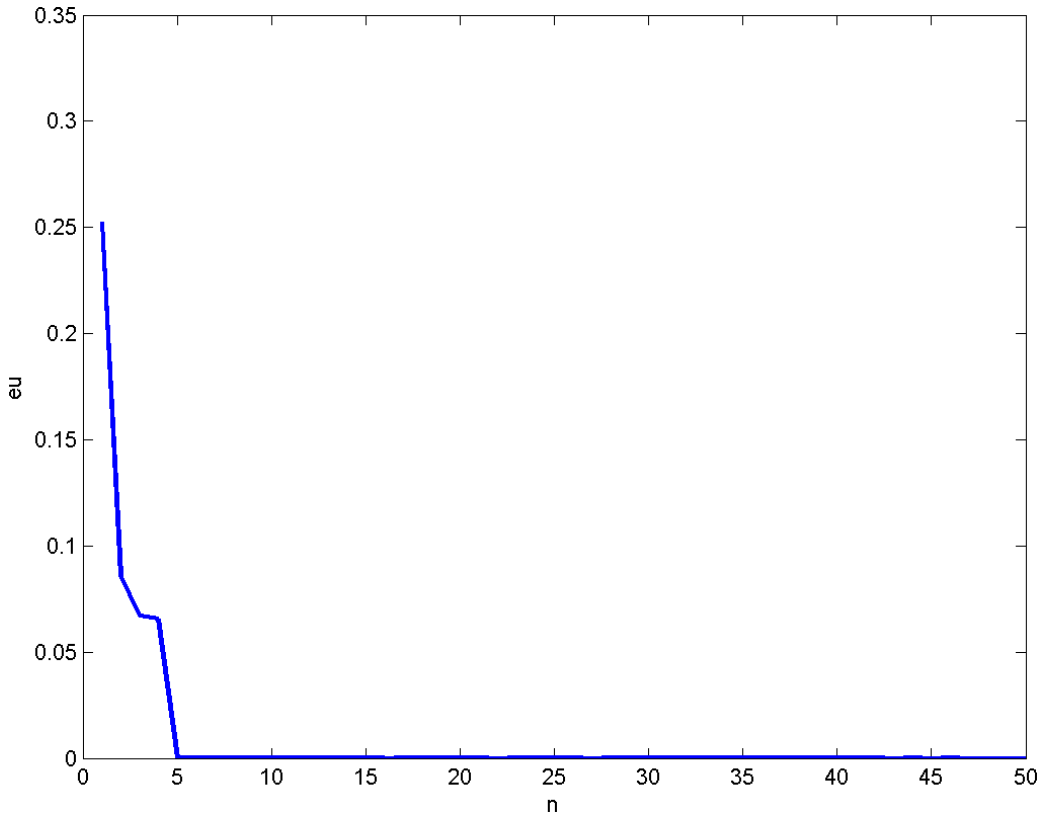
Rysunek 12: wpływ liczby neuronów na jakość aproksymacji zbioru testowego – szeroki przedział



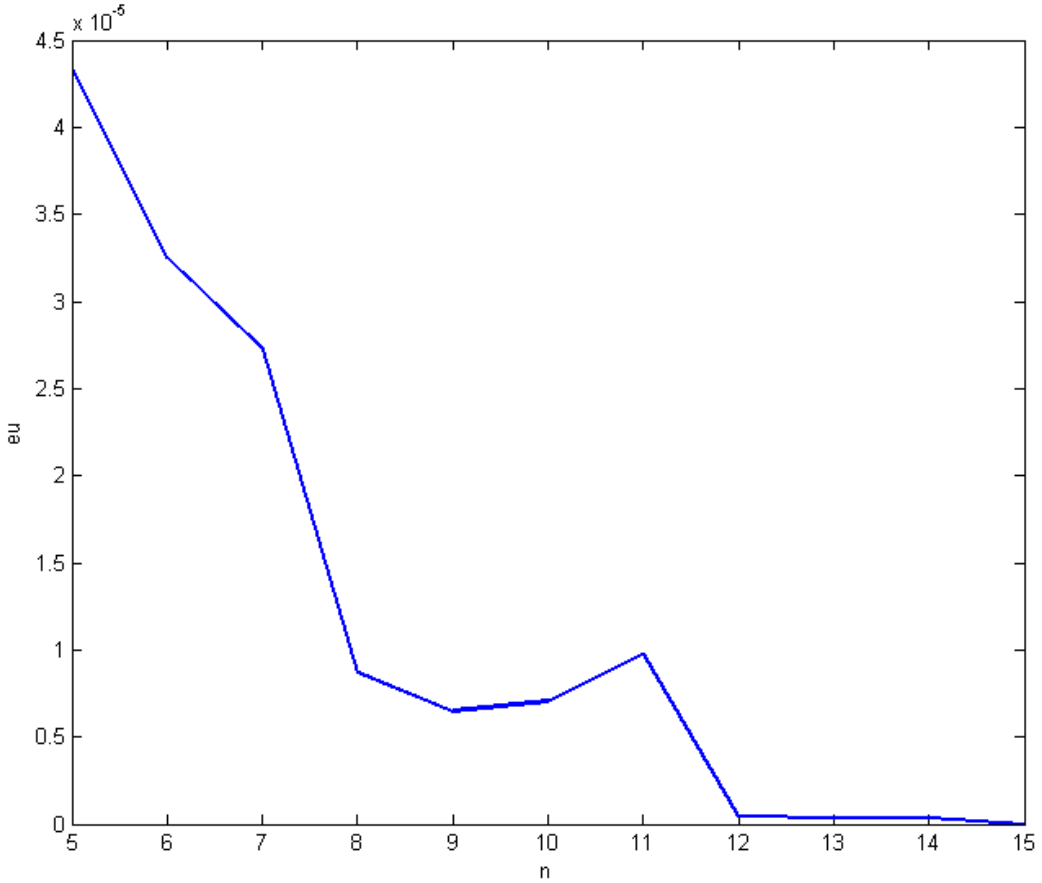
Rysunek 13: wpływ liczby neuronów na jakość aproksymacji zbioru testowego – wąski przedział

2. Zbiór uczący

Wpływ liczby neuronów na aproksymację zbioru uczącego został zbadany w sposób analogiczny, jak w przypadku zbioru testowego.



Rysunek 14: wpływ liczby neuronów na jakość aproksymacji zbioru uczącego – szeroki przedział



Rysunek 15: wpływ liczby neuronów na jakość aproksymacji zbioru uczącego – wąski przedział

IV. Spostrzeżenia i wnioski

Dobór liczby neuronów sieci ma bardzo istotny wpływ na jakość aproksymacji. W przypadku zbioru uczącego zwiększanie liczby neuronów wpływa korzystnie na przebieg wykresu (rys. 4) i minimalizuje błąd średniokwadratowy (rys. 14 i 15). W przypadku zbioru testowego sprawa się komplikuje. Zbyt mała liczba neuronów powoduje „niedouczenie" (rys. 5, kolor niebieski), a zbyt duża „przeuczenie" (rys. 5, kolor pomarańczowy) sieci. Z rys. 5 można by wnioskować, że dla badanej przez nas funkcji (rys. 1) i sieci najbardziej optymalna liczba neuronów będzie bliska liczbie 5, gdyż poniżej i powyżej tej wartości błąd średniokwadratowy rośnie.

Dokładne przebadanie zależności jakości aproksymacji od liczby neuronów (rys. 12 i 13) wskazuje jednak, że jest inaczej. Widać na niej, że dla badanej przez nas funkcji (rys. 1) i sieci najbardziej optymalną liczbą neuronów w pierwszej warstwie jest 9. Wykres ten potwierdza, że rzeczywiście błąd dla 5 neuronów jest mniejszy niż dla n=10 oraz że w pobliżu n=5 znajduje się minimum, jednakże jest to minimum lokalne. Pobieżne zbadanie wpływu liczby neuronów na jakość aproksymacji (rys. 5) nie pozwala więc znaleźć najbardziej optymalnej liczby n. Można też zauważyć, że minima lokalne w przedziale liczby neuronów od 5 do 15 występują dokładnie co trzy neurony, a więc dla n=6, 9, 12 i 15.

Wpływ liczby warstw sieci jest podobny. Istnieje więc pewna liczba warstw, dla której błąd średniokwadratowy dla zbioru testowego jest minimalny. W naszym przypadku sieć najlepiej aproksymowała zbiór testowy przy 5 warstwach (rys. 3, kolor żółty). W tym przypadku, w odróżnieniu od wpływu liczby neuronów, dobór liczby warstw sieci ma także znaczenie przy zbiorze uczącym. Sieć najlepiej nauczyła się tego zbioru dla 4 warstw, poniżej i powyżej tej wartości błąd średniokwadratowy rósł (rys. 2).

Przebadanie różnych postaci funkcji aktywacji (rys. 6 i 7) pozwala nam stwierdzić, że jedynymi rozsądnymi kombinacjami dla dwóch warstw są: $\{'tansig' \quad 'purelin'\}$ oraz $\{'logsig' \quad 'purelin'\}$. Pozostałe przebadane opcje dały rezultaty dalekie od optymalnych.

Zbadanych zostało także 5 metod uczenia sieci (rys. 8 i 9). Metoda $'trainlm'$ jest domyślną i najszybszą metodą uczenia. Metoda $'trainbfg'$ jest wolniejsza, ale daje lepsze rezultaty, podobnie jak $'trainrp'$, która jest najwolniejsza i najdokładniejsza z całej tej trójki. Metoda gradientowa $'traingd'$ dała słabe wyniki, ale już jej rozwinięcie o metodę momentum i uczenie adaptacyjne, a więc metoda $'traingdx'$, pozwoliła na nauczenie sieci w sposób zbliżony dokładnością do metody $'trainlm'$.

Ostatnim z przebadanych parametrów była maksymalna liczba epok. Zwiększanie tej liczby powodowało lepszą aproksymację zbioru uczącego (rys. 10 i 11), ale tylko do pewnej wartości, która w naszym przypadku wynosiła 10. Zwiększanie tej liczby nie wpływało już dalej na poprawę jakości aproksymacji obydwu zbiorów, ponieważ zadana liczba epok (iteracji) bardzo często nie była w ogóle osiągana – proces był przerywany z powodu osiągnięcia maksymalnej lub minimalnej wartości innych wskaźników około dziesiątej iteracji. Odpowiedni dobór tej liczby w naszym przypadku sprowadza się więc tylko do przekroczenia pewnego minimalnego progu. Liczba epok może mieć większe znaczenie w zadaniach, w których inne wskaźniki nie osiągają swego ekstremum, a tym samym nie przerywają procesu uczenia.

V. Listing

```
clear all; close all; clc;

% Nasza funkcja nieliniowa
f=@(x) abs(x.^5-3*x.^2+0.5*x);
figure;
X=-0.4:0.01:1.6;
Y=f(X);
plot(X,Y,'LineWidth',2);
saveas(gcf,'f.png');
close(gcf);

% Zadanie 1
N={ [10 1];
    [10 1];
    [10 1];
    [10 1];
    [10 1]
  };
F={ {'tansig','purelin'};
    {'tansig','purelin'};
    {'tansig','purelin'};
    {'tansig','purelin'};
    {'tansig','purelin'}
  };
M={ 'trainlm';
    'trainlm';
    'trainlm';
    'trainlm';
    'trainlm';
  };
e={ 2;
    5;
    10;
    20;
    50;
  };
Zad1(f,N,F,M,e,10);

% Zadanie 2
nopt = Zad2(f,[1 50],F,M,e,3);
nopt = Zad2(f,[5 15],F,M,e,30);
```

```
function Zad1(f,N,F,M,e,p)

    % Generowanie zbioru uczącego
    Xu=-0.4:0.1:1.6;
    Du=f(Xu);

    % Generowanie zbioru testowego
    Xt=-0.4:0.001:1.6;
    Dt=f(Xt);

    % Prealokacja macierzy
    n=length(N);
    Yu=zeros(p,length(Xu),n);
    Yt=zeros(p,length(Xt),n);
    Eu=zeros(p,n);
    Et=zeros(p,n);
    EU=cell(1,n);
    ET=cell(1,n);

    % Różne wartości parametrów
    for j=1:n

        % Wiele powtórzeń procesu uczenia
        for i=1:p

            % Utworzenie sieci neuronowej
            S=newff([-0.4 1.6],N{j},F{j},M{j});

            % Uczenie sieci neuronowej
            S.trainParam.showWindow=0;
            S.trainParam.epochs=e{j};
            S=train(S,Xu,Du);

            % Aproksymacja zbioru uczącego
            Yu(i,:,j)=sim(S,Xu);
            Eu(i,j)=mse(Du-Yu(i,:,j));

            % Aproksymacja zbioru testowego
            Yt(i,:,j)=sim(S,Xt);
            Et(i,j)=mse(Dt-Yt(i,:,j));

        end

    end

    % Mediana aproksymacji i błędu
    Yu=squeeze(median(Yu));
    Yt=squeeze(median(Yt));
    Eu=median(Eu);
    Et=median(Et);

    % Wykresy końcowe
    K=jet(n);
    figure(1); hold on;
    plot(Xu,Du,'ko','LineWidth',2);
    for i=1:n
        plot(Xu,Yu(:,i)','Color',K(i,:),'LineWidth',2);
        EU{i}=['eu=' num2str(Eu(i))];
    end
    EU(2:n+1)=EU;
    EU{1}='Zbiór uczący';
    xlabel('Xu');
    ylabel('Du, Yu');
    legend(EU,'Location','North');
    figure(2); hold on;
    plot(Xt,Dt,'k','LineWidth',2);
    for i=1:n
        plot(Xt,Yt(:,i)','Color',K(i,:),'LineWidth',2);
        ET{i}=['et=' num2str(Et(i))];
    end
    ET(2:n+1)=ET;
    ET{1}='Zbiór testowy';
    xlabel('Xt');
    ylabel('Dt, Yt');
    legend(ET,'Location','North');

    % Zapisanie wykresów
    saveas(1,'Zad1\eu.png');
    saveas(2,'Zad1\et.png');
    close all;

end
```

```
function nopt = Zad2(f,np,F,M,e,p)

    % Generowanie zbioru uczącego
    Xu=-0.4:0.1:1.6;
    Du=f(Xu);

    % Generowanie zbioru testowego
    Xt=-0.4:0.001:1.6;
    Dt=f(Xt);

    % Prealokacja macierzy
    d=diff(np)+1;
    Eu=zeros(p,1);
    Et=zeros(p,1);
    eu=zeros(d,1);
    et=zeros(d,1);

    % Różna liczba neuronów
    for n=1:d

        % Wiele powtórzeń procesu uczenia
        for i=1:p

            % Utworzenie sieci neuronowej
            S=newff([-0.4 1.6],[n+np(1)-1 1],F,M);

            % Uczenie sieci neuronowej
            S.trainParam.showWindow=0;
            S.trainParam.epochs=e;
            S=train(S,Xu,Du);

            % Aproksymacja zbioru uczącego
            Yu=sim(S,Xu);
            Eu(i)=mse(Du-Yu);

            % Aproksymacja zbioru testowego
            Yt=sim(S,Xt);
            Et(i)=mse(Dt-Yt);

        end

        % Mediana błędu
        eu(n)=median(Eu);
        et(n)=median(Et);

    end

    % Wyznaczenie optymalnej liczby neuronów
    nopt=find(et==min(et))+np(1)-1;

    % Wykresy końcowe
    figure(1);
    plot(np(1):np(2),eu,'LineWidth',2);
    xlabel('n');
    ylabel('eu');
    figure(2);
    plot(np(1):np(2),et,'LineWidth',2);
    xlabel('n');
    ylabel('et');

    % Zapisanie wykresów
    saveas(1,'Zad2\eu.png');
    saveas(2,'Zad2\et.png');
    close all;

end
```