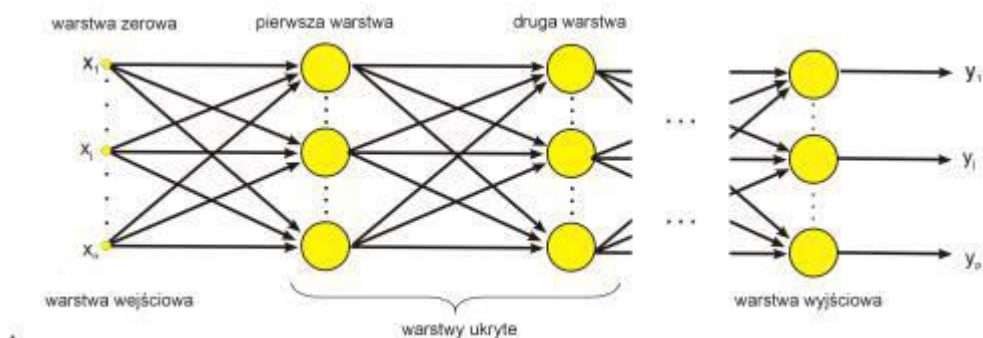


1. Temat ćwiczenia:

Budowa i działanie sieci wielowarstwowej typu feedforward.

2. Wstęp teoretyczny:

Sieci jednokierunkowe wielowarstwowe (feedforward) – sieci neuronowe posiadające prostą, łatwą do opisaną strukturę oraz łatwą do realizacji metodę uczenia. Struktura ta jest zbliżona do budowy mózgu. W sieciach jednokierunkowych można wyróżnić uporządkowane warstwy neuronów (w tym warstwę wejściową i warstwę wyjściową). Liczba neuronów w każdej z warstw może być różna, przy czym w danej warstwie wszystkie neurony mają taką samą funkcję przejścia – neurony z różnych warstw mogą mieć różne funkcje przejścia. Połączenia występują tylko pomiędzy neuronami z sąsiednich warstw, wg zasady „każdy z każdym” i mają one charakter asymetryczny. Sygnały przesyłane są od warstwy wejściowej poprzez warstwy ukryte (jeśli występują) do warstwy wyjściowej (w jednym kierunku). Neurony warstwy wejściowej posiadają tylko jedno wejście i uproszczoną funkcję przejścia (umownie jest to warstwa zerowa sieci). Zadaniem neuronów z warstwy wejściowej jest wstępna obróbka sygnału (np.: normalizacja, kodowanie itp.). Z kolei za przetwarzanie decyzyjne odpowiedzialne są neurony warstw ukrytych i warstwy wyjściowej, a odpowiedź udzielana jest przez neurony warstwy wyjściowej.

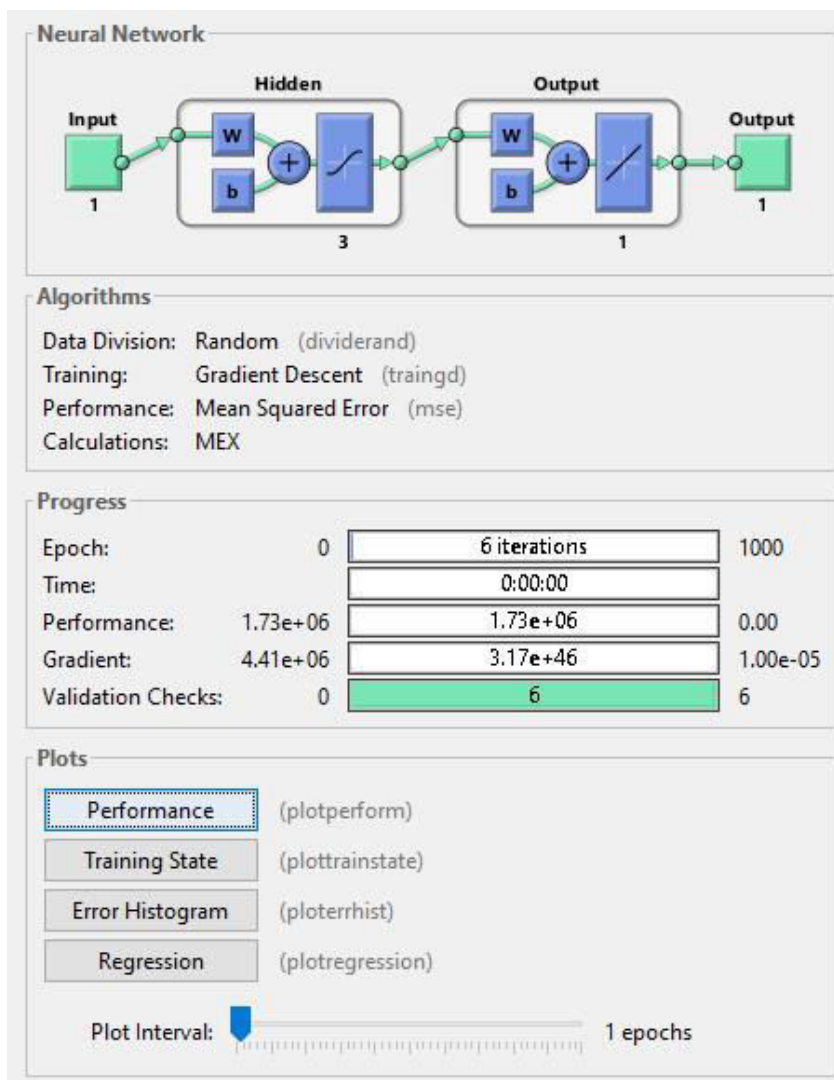


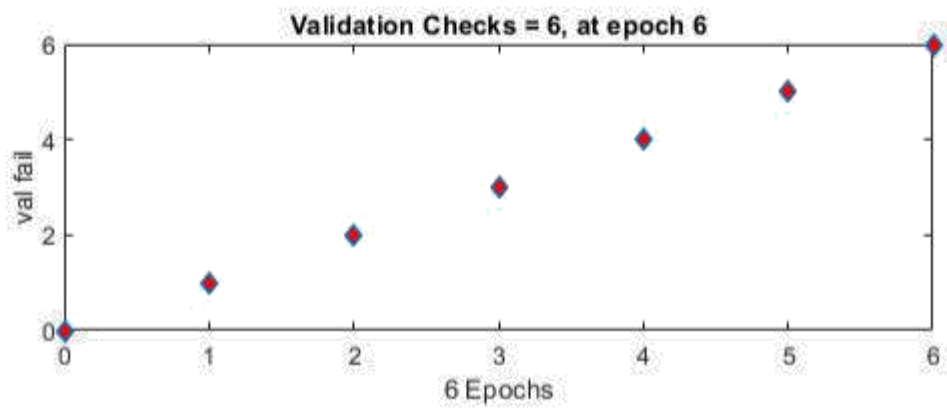
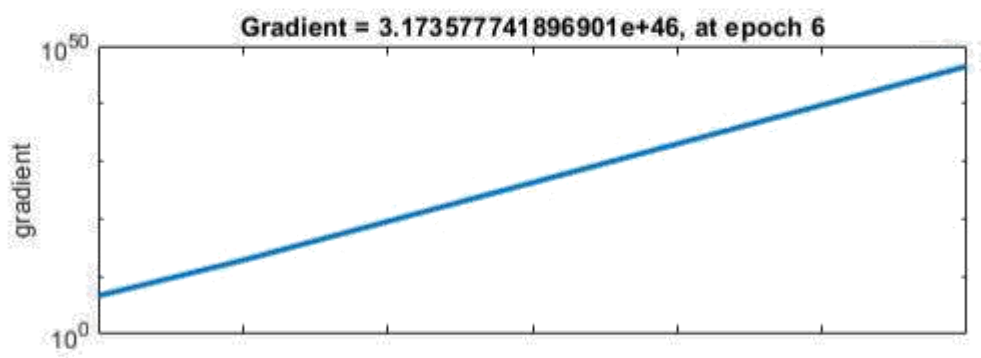
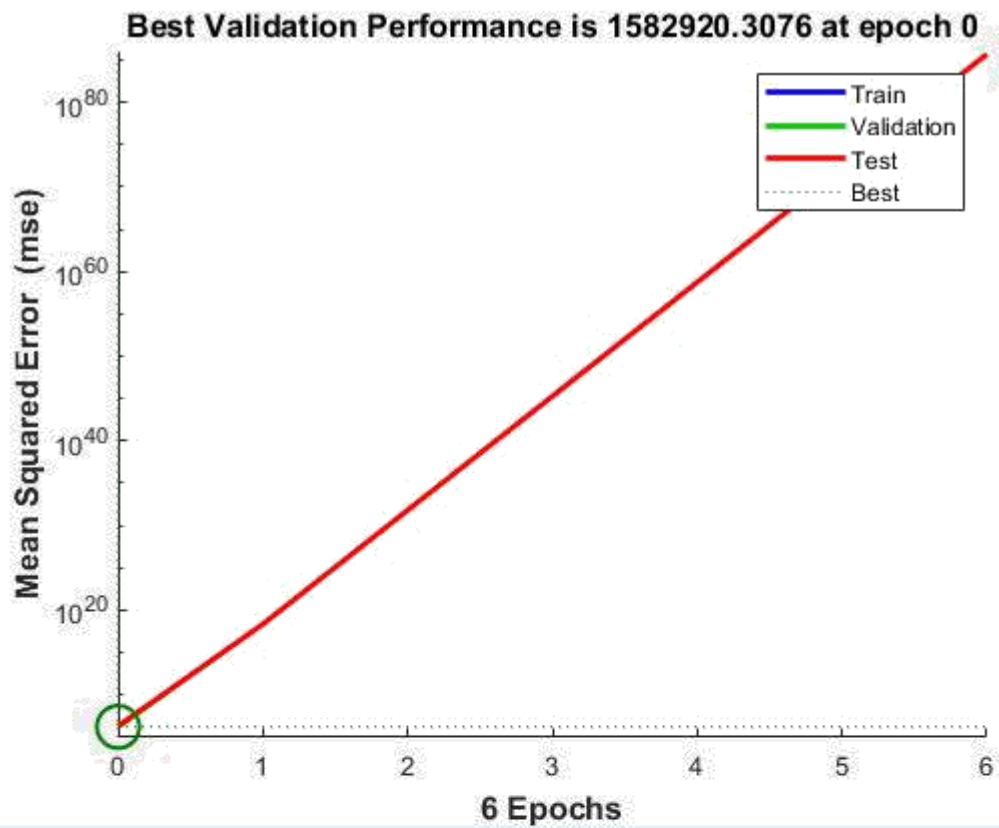
3. Opis użytych funkcji i metod zaczerpniętych z pakietu MATLAB:

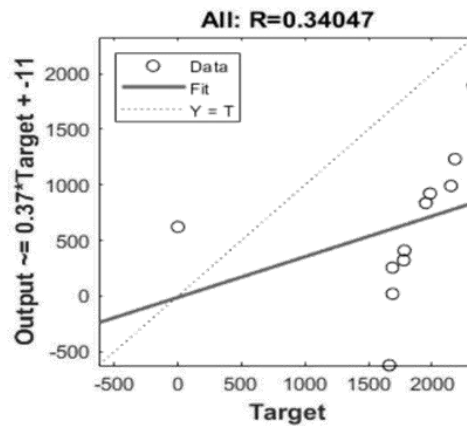
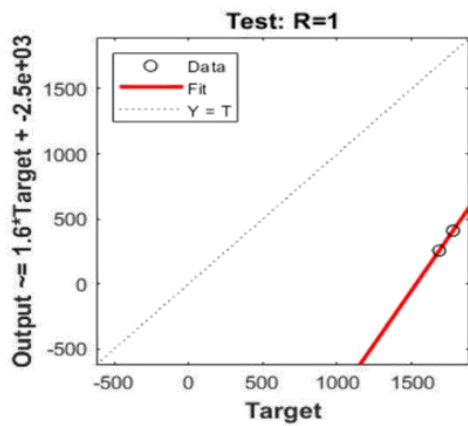
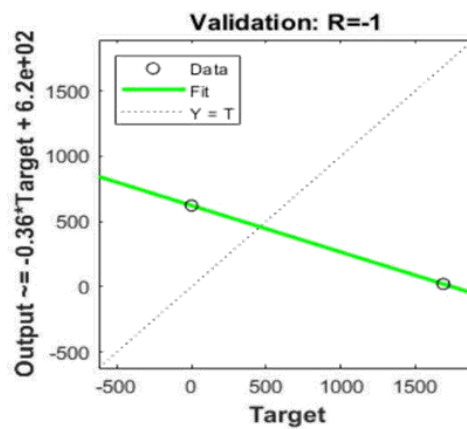
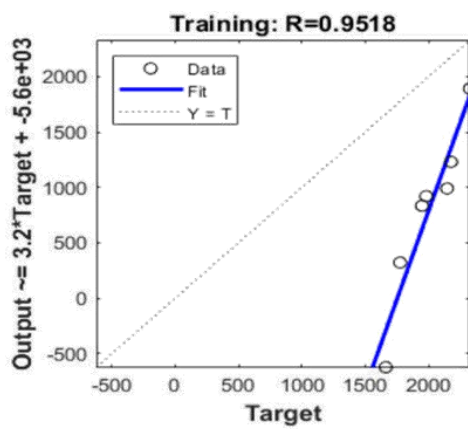
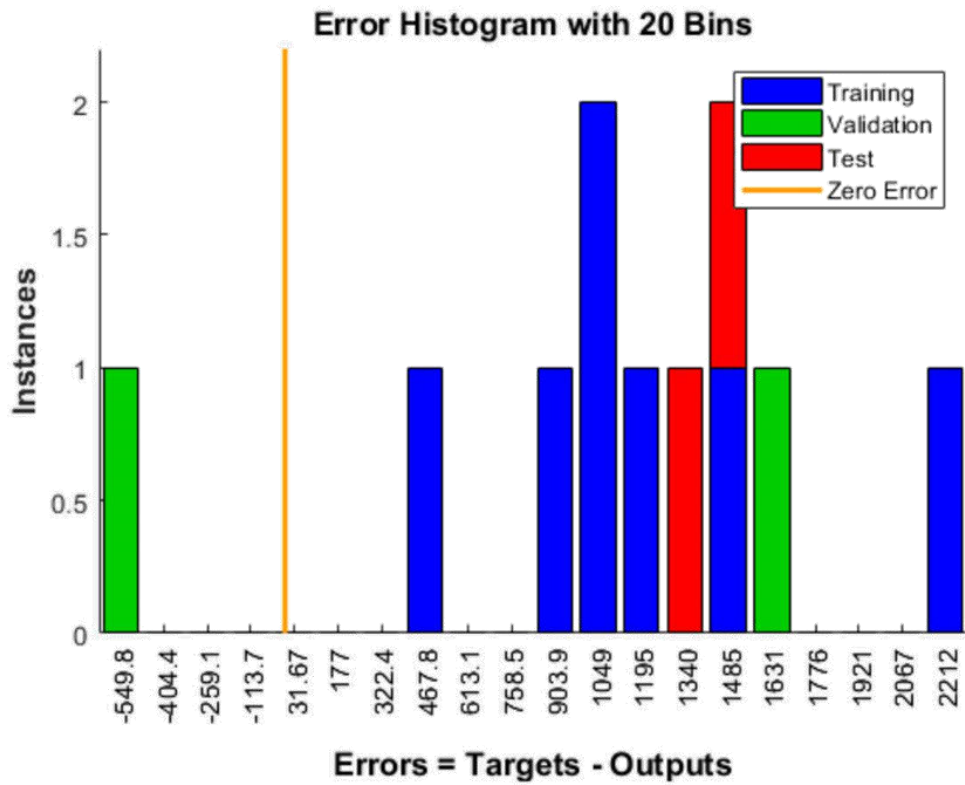
- wejście – tablica danych wejściowych z przedziału od -2 do 2, z krokiem zmieniania co 0.4
- wyjście – tablica o takiej samej liczbie elementów jak wejście, której elementami są wyniki uzyskane dzięki napisanej funkcji Rastigin3D dla liczb z tablicy wejścia
- testowe – tablica pomocnicza uzupełniona zerami
- feedforwardnet() – funkcja służąca do tworzenia sieci z ukrytymi warstwami, ilość wszystkich warstw wskazana jest jako wartość argumentu
- trainFcn – algorytm służący do przeprowadzania wstecznej propagacji
- trainParam – pod tą zmienną przechowywane są wszelkie parametry potrzebne do wykonania zadania; w naszym przypadku są to współczynnik uczenia oraz bezwładność
- train() – przeprowadzenie procesu uczenia sieci z wykorzystaniem danych wejściowych i wyjściowych
- RastriginTest3D() – funkcja napisana w osobnym pliku, dla której należało wygenerować dane uczące i testujące

4. Zrzuty ekranu przedstawiające przykładowe rezultaty po uruchomieniu programu:

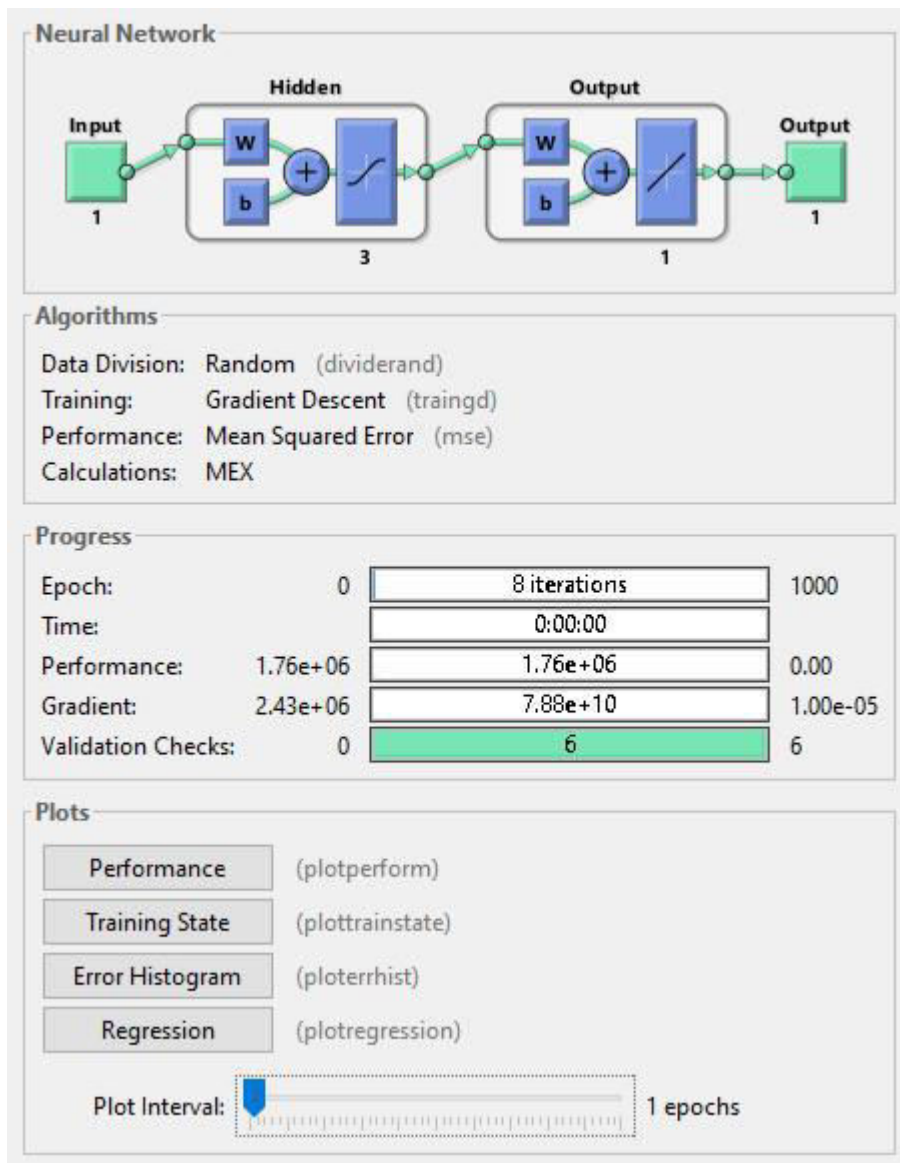
wersja dla parametrów : współczynnik uczenia = 0.5, bezwładność = 0.5

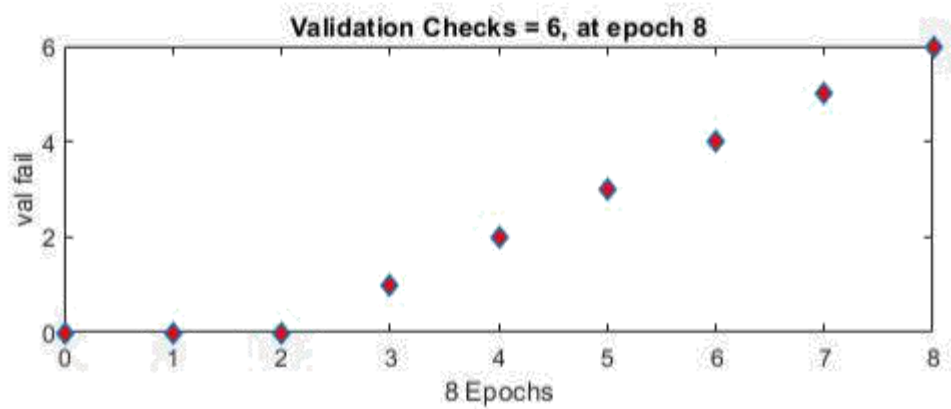
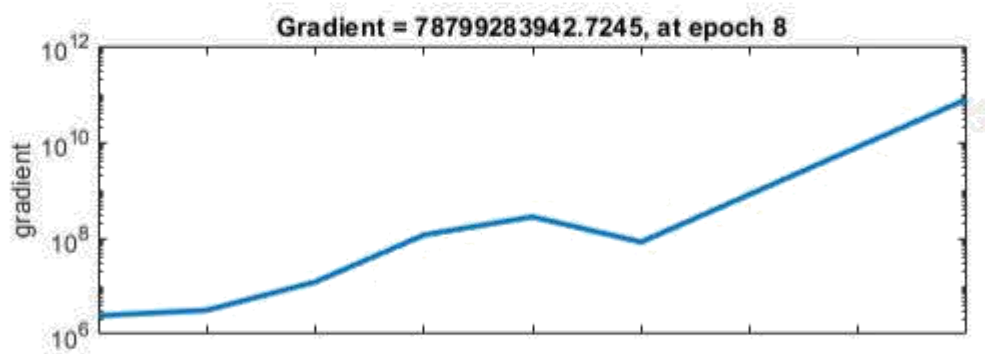
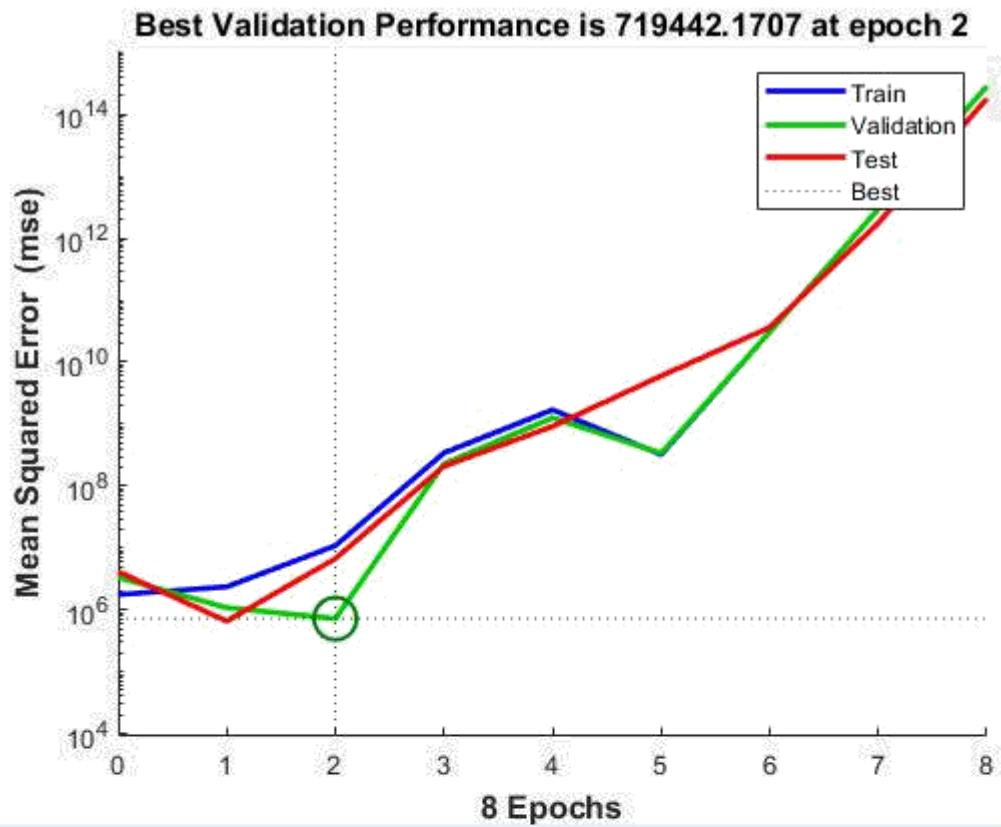


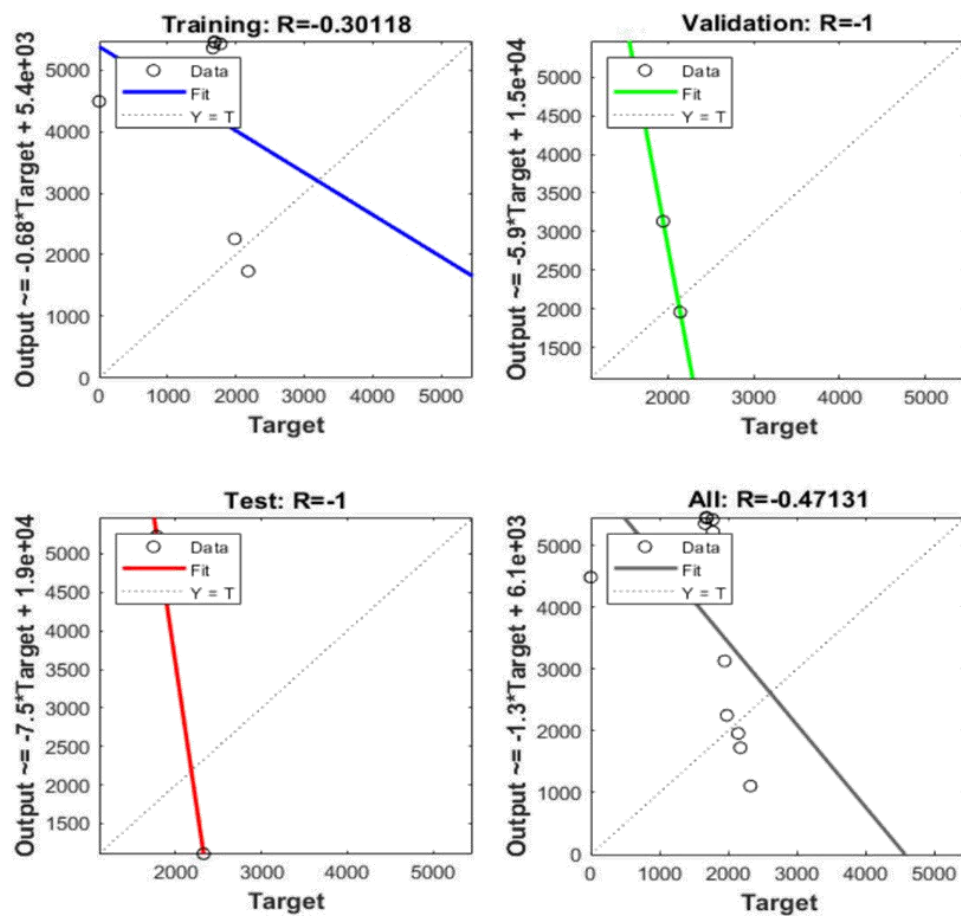
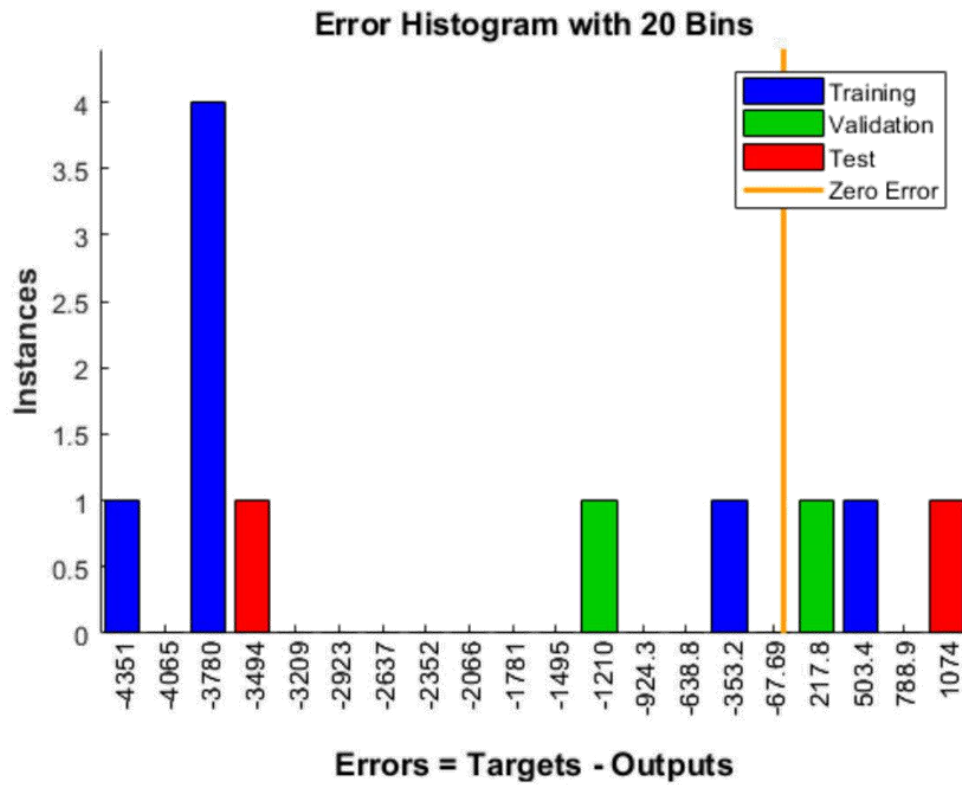




- wersja dla parametrów : współczynnik uczenia = 0.000001, bezwładność = 0.1







5. Wyniki:

- dla pierwszego przypadku:

1.6633e+03

1.6880e+03

1.6867e+03

1.7764e+03

1.7800e+03

1.9495e+03

1.9825e+03

2.1477e+03

2.1791e+03

2.3262e+03

- dla drugiego przypadku:

1.6633e+03

1.6880e+03

1.6867e+03

1.7764e+03

1.7800e+03

1.9495e+03

1.9825e+03

2.1477e+03

2.1791e+03

2.3262e+03

6. Wnioski:

w ramach projektu udało mi się zrealizować wszystkie wymagane kroki
sieciami wielowarstwowymi łatwo jest sterować, zmieniając poszczególne parametry, takie jak bezwładność czy współczynnik uczenia
tempo uczenia sieci jest zmienne, z reguły im większy współczynnik uczenia, tym sieć uczy się szybciej
w takiej sytuacji jednak generowana jest większa ilość błędów, gdyż tracona jest precyzja
bardzo małe wartości współczynnika uczenia wpływają nie tylko na spowolnienie uczenia, ale też na zwiększenie liczby epok
w ogólnym wniosku można stwierdzić, że algorytm z propagacją danych znacznie przyspiesza proces uczenia, jednak wykryć można więcej błędów
co ciekawe, dla wartości 0 funkcja Rastrigin3D przyporządkowała również 0

7. Listing całego kodu programu wykonanego w MATLABie:

- funkcja RastrignTest3D:

```
function fx = RastrignTest3D(x)
    if x == 0
        fx = 0;
    else
        A = 10;
        n = 100;
        x1 = x;
        dx = (5.12-x)/n;

        fx = A * n;

        for i = 1:n
            x = x1 + (i * dx);
            fx = fx + (x^2) - (A * cos(2 * pi * x));
        end
        disp(fx)
    end
end
```

- główny program:

```
close all; clear all; clc;

wejście = [-2 -1.6 -1.2 -0.8 -0.4 0 0.4 0.8 1.2 1.6 2];
wyjście = [1.6633e+03 1.6880e+03 1.6867e+03 1.7764e+03 1.7800e+03 0.0 1.9495e+03 1.9825e+03 2.1477e+03 2.1791e+03 2.3262e+03];
testowe = zeros(1);
net = feedforwardnet(3);
net.trainFcn = 'traingd';
net.trainParam.lr = 0.000001;
net.trainParam.mc = 0.1;
net = train(net, wejście, wyjście);
efekty = zeros(size(net));
for i = 1:11
    testowe(i) = RastrignTest3D(wejście(i));
    efekty(i) = sim(net, wejście(i));
end
```