

Sztuczna inteligencja

Temat: Testowanie algorytmów BP

Remigiusz Bekierz

Rafał Popiołek

Krzysztof Koźlik

1. Wprowadzenie

Działanie sieci neuronowej oparte jest na nauce opartej na konkretnych przykładach. Naszym zadaniem jest pokazanie rozwiązanych przykładów. Następnie tok uczenia zaczyna od przypisania losowych lub równych wag dla każdego z wejścia. Wagi zostają zmieniane do momentu uzyskania poprawnej odpowiedzi.

Algorytm propagacji wstecznej polega na doborze odpowiednich wag przy zastosowaniu gradientowych metod optymalizacji. Algorytm początkowo wyznacza wagi sieci w sposób losowy. Następnie podawany jest na wejściu pierwszy wektor uczący, liczy sygnały wyjściowe sieci dla poszczególnych neuronów. Zostaje policzony błąd na podstawie różnicy odpowiedzi sieci a sygnałem zadany. Obliczona zostaje wartość funkcji energetycznej, sprawdzane są kryteria zakończenia obliczeń. Głównym kryterium jest wartość funkcji energetycznej, w przypadku otrzymania wartości 0 dalsze liczenie nie przyniesie pożądanych skutków. Jeśli wartość funkcji energii jest różna od 0 to stosowana jest propagacja wsteczna. Obliczane zostają nowe wartości wag oraz modyfikacja nowymi wartościami dla neuronów warstwy wyjściowej. Kolejnym krokiem jest liczenie błędu dla każdego z neuronów warstwy ukrytej. Tak jak w przypadku wcześniejszym dla warstwy wyjściowej obliczane są wartości wag oraz modyfikacja dla każdego z neuronów, tym razem dla warstwy ukrytej. Obliczony zostanie błąd oraz nowa wartość neuronów dla warstwy ukrytej, przypisanie nowych wartości. Ta operacja będzie iterowana, aż zostaną zmodyfikowane wszystkie warstwy ukryte i dotrze do warstwy pierwszej.

Epoka jest to jeden cykl uczenia. Przedstawia wszystkie próbki uczone w sieci neuronowej. Wagi są modyfikowane jednokrotnie.

Trening może zostać zakończony, gdy zostanie osiągnięta maksymalna liczba epoko, wydajność jest zminimalizowana do celu lub zostanie przekroczony czas.

Opis algorytmów

- **Propagacja wsteczna regularyzacji Bayesowskiej** powstała w celu minimalizacji liniowej kombinacji kwadratów błędów i wag. Wykorzystuje swoje możliwości do modyfikacji kombinacji liniowej, robi to w celu uzyskania dobrych wyników generalizacji na końcu szkolenia. Propagacja wsteczna wykorzystuje do obliczania macierz Jacobiego, zbudowany jest z pochodnych cząstkowych rzędu pierwszego dla składowych funkcji rzeczywistych. Jego działanie opisano na wzorze:

$$jj = jX * jX$$

$$je = jX * E$$

$$dX = -(jj + I * \mu) \setminus je$$

- **Propagacja wsteczna Levenberga-Marquardta** została zaprojektowana tak, by zbliżyć się do szybkości uczenia rzędu drugiego, ale by konieczne nie było obliczania macierzy Hessego. Gdy funkcja wydajności ma postać sumy kwadratów, wówczas macierz Hessego można przybliżyć jako:

$$H = J^T J$$

a gradient można obliczyć jako:

$$g = J^T e$$

- **Propagacja wsteczna gradientu koniugatu skalowanego** służy do obliczania pochodnych wydajności w odniesieniu do zmiennych wagi i odchylenia. Przeskalowany algorytm gradientu sprzężony jest na podstawie sprzężonych kierunkach.

2. Opis problemu

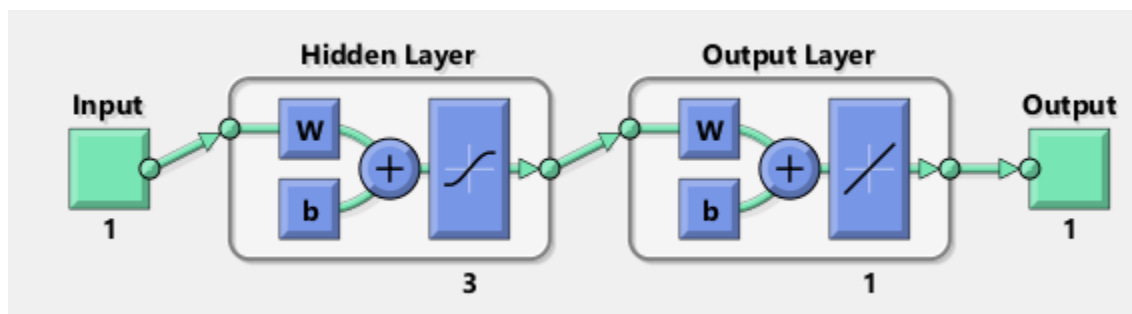
Podczas analizy przykładowych danych podawanych na wejściu (X) zostaną przyporządkowane odpowiadające wielkości wyjściowe (Y). Po poprawnym lub błędnym dobraniu danych system zwraca komunikat z informacją. Na podstawie otrzymanej wiadomości możemy przeanalizować poszczególne wyniki. Ważną informacją jest

również odpowiednia liczba neuronów w warstwie ukrytej. W dalszej części sprawozdania zostanie szczegółowo opisany ten problem.

Na zajęciach korzystano ze środowiska „Matlab”, które nie posiada interpretacji. W celu zrozumienia działania przedstawiono przykładową interpretację:

Zadaniem sieci modelowej jest modelowanie prędkości sportowca biegnącego na określonym dystansie. Na sygnale wejściowym zostanie podany czas liczony w sekundach po jakim zawodnik pokona dystans. Czas jest mierzony na mecie. Natomiast na wyjściu podano dystans liczony w metrach, który sportowiec pokonał. Zadaniem sieci modelowej jest obliczanie wartości prędkości dla poszczególnych par wartości.

3. Schemat o opis sieci neuronowej



Rysunek 1 Przedstawia schemat sieci neuronowej.

Analizowana sieć jest dwuwarstwowa oraz jednokierunkowa. Sieć neuronowa wyposażona jest w sigmoidalną funkcję aktywacji dla warstwy ukrytej i liniową dla warstwy wyjściowej. Powyższy schemat sieci neuronowej składa się z warstwy wyjściowej i wejściowej, każda z warstw posiada jeden neuron. W warstwie ukrytej sytuacja wygląda zupełnie inaczej, w ramach ćwiczeń zmieniano wartości neuronów na 3 lub 10 dla każdego ze sposobów uczenia. W przedstawionym schemacie wartość neuronów wynosi 3.

4. Wyniki stymulacji

Propagacja wsteczna Levenberga-Marquardta – próba 1 (3 neurony w warstwie ukrytej)

NetworkTraining

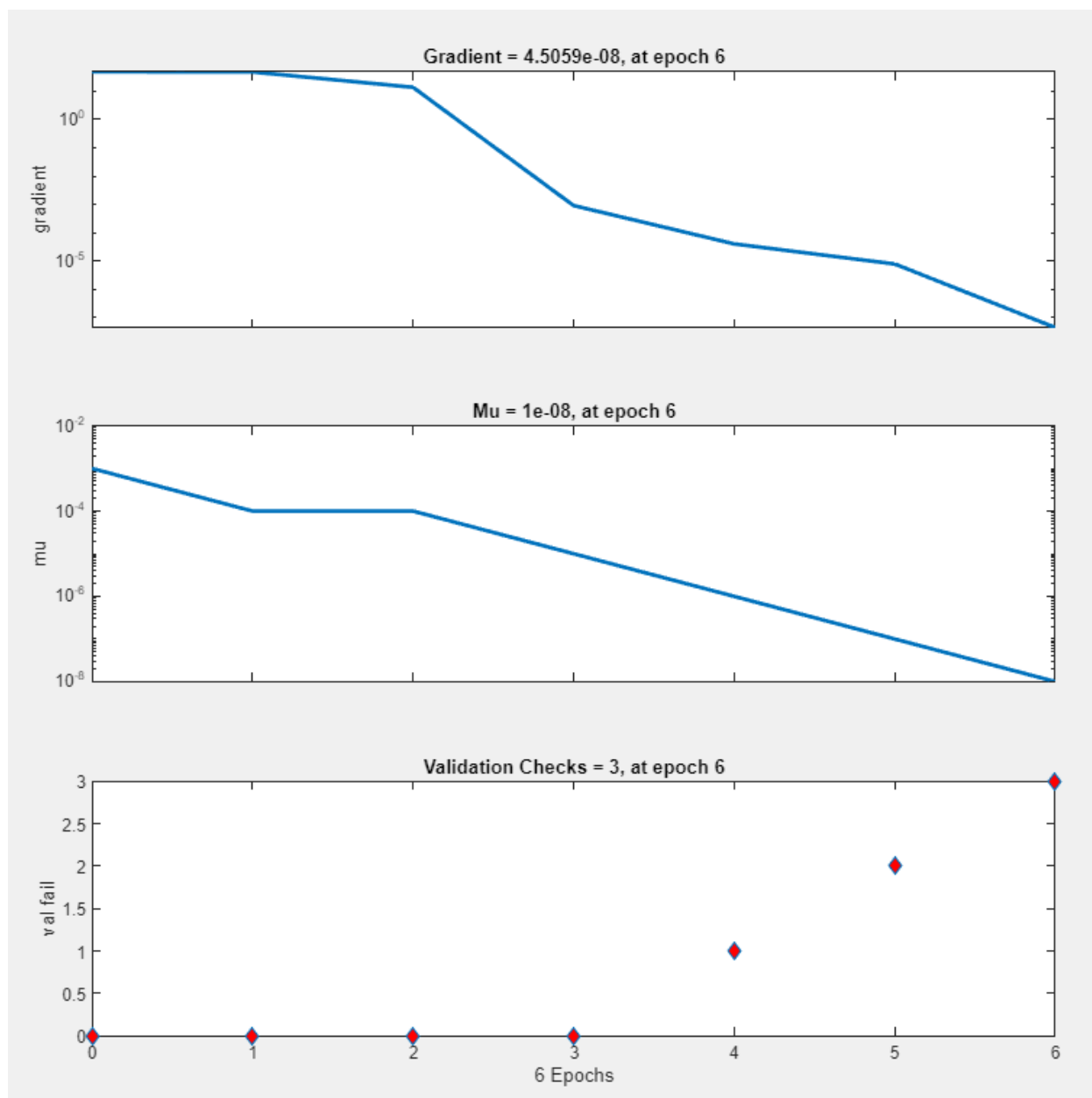
Training Results

Training finished: Reached minimum gradient ✓

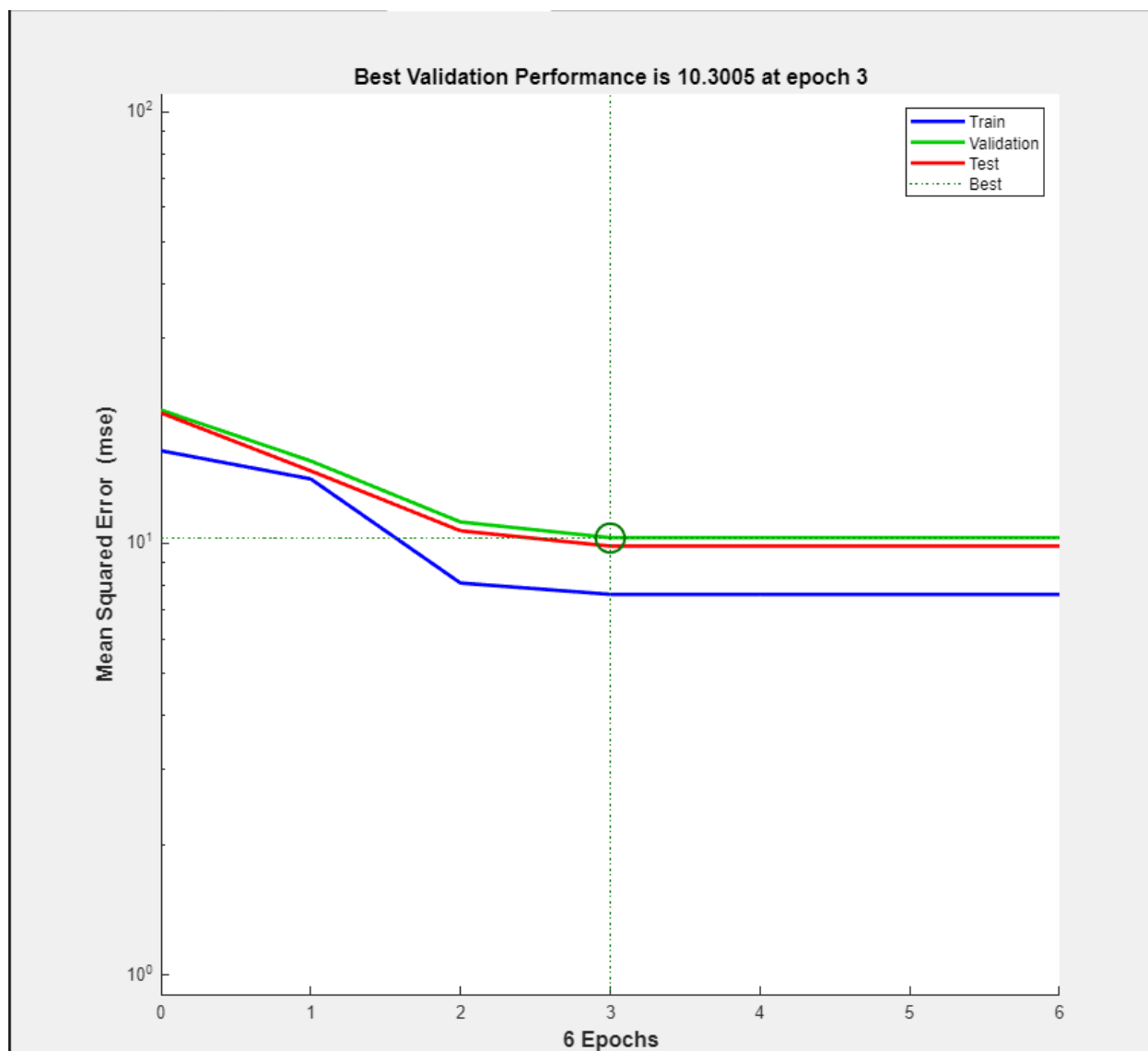
Training Progress

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	6	1000
Elapsed Time	-	00:00:00	-
Performance	16.4	7.61	0
Gradient	48.7	4.51e-08	1e-07
Mu	0.001	1e-08	1e+10
Validation Checks	0	3	6

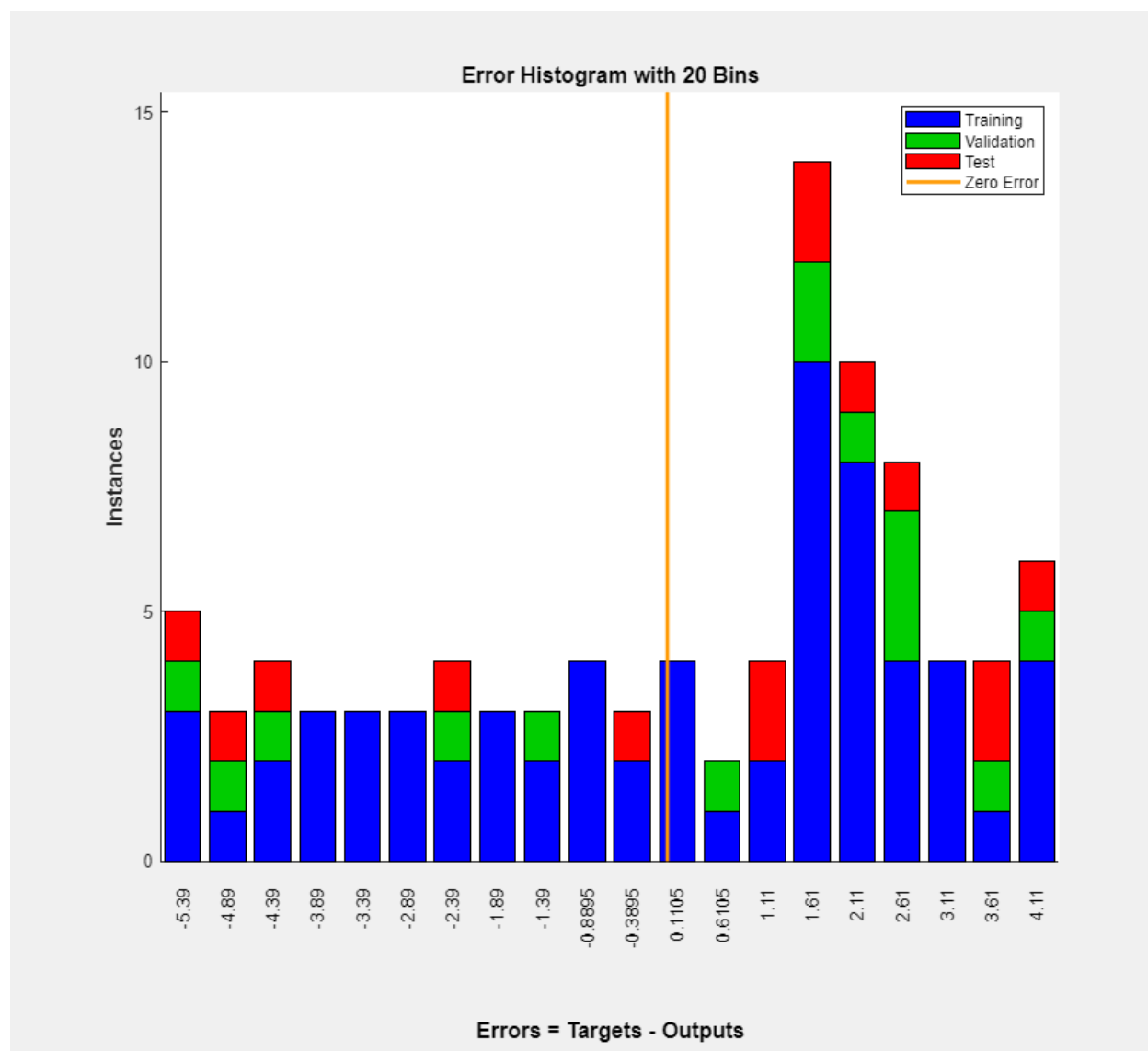
1 Uczenie się sieci



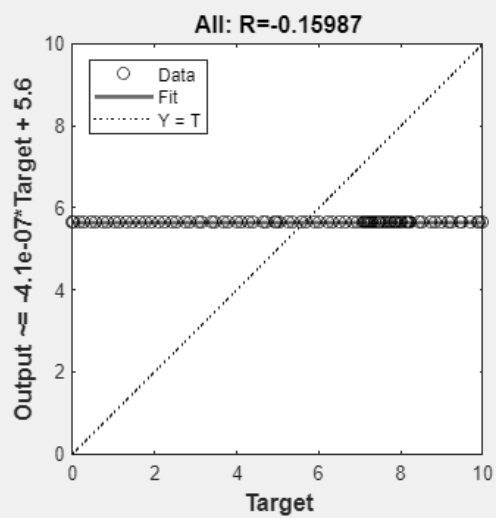
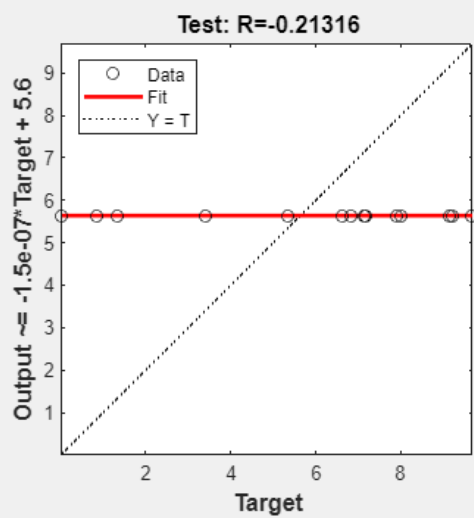
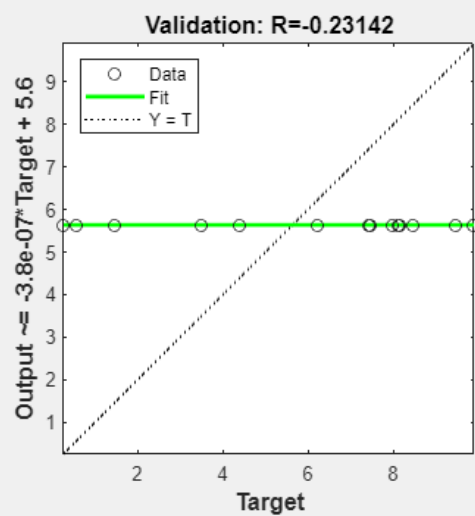
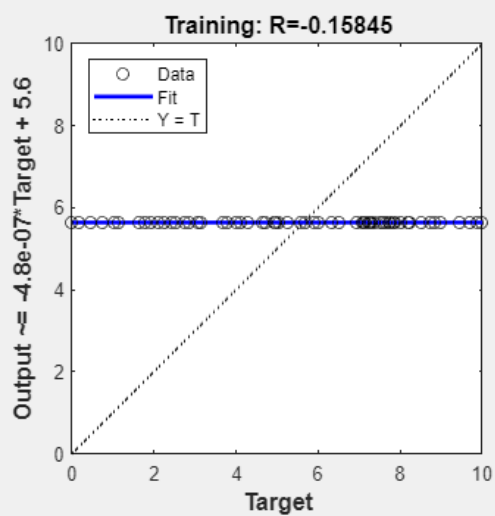
2 Wykres stanu szkolenia



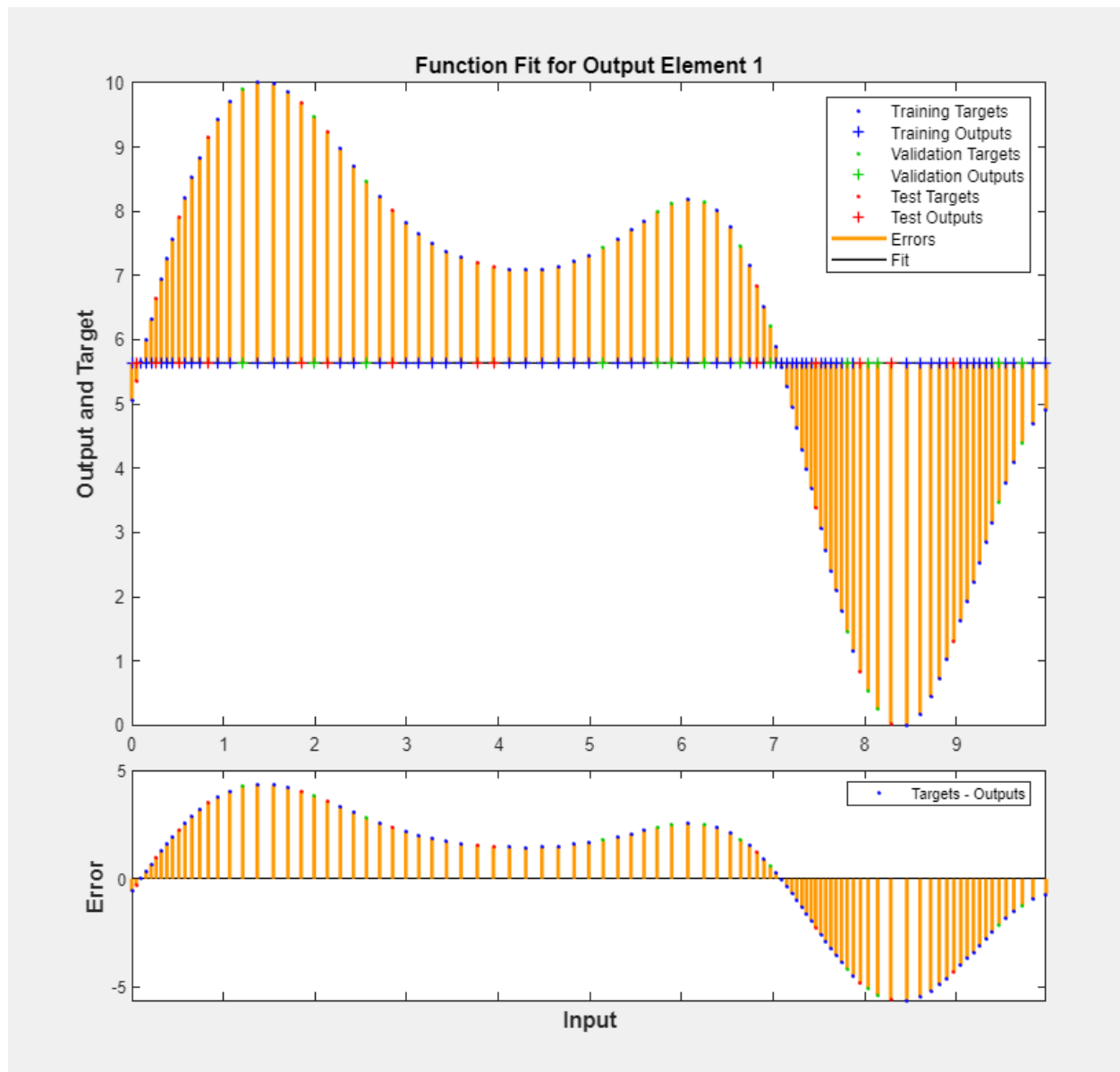
3 Błąd średnio kwadratowy



4 Histogram błędów




5 Wykres regresji



6 Dopasowanie funkcji dla pojedynczego elementu

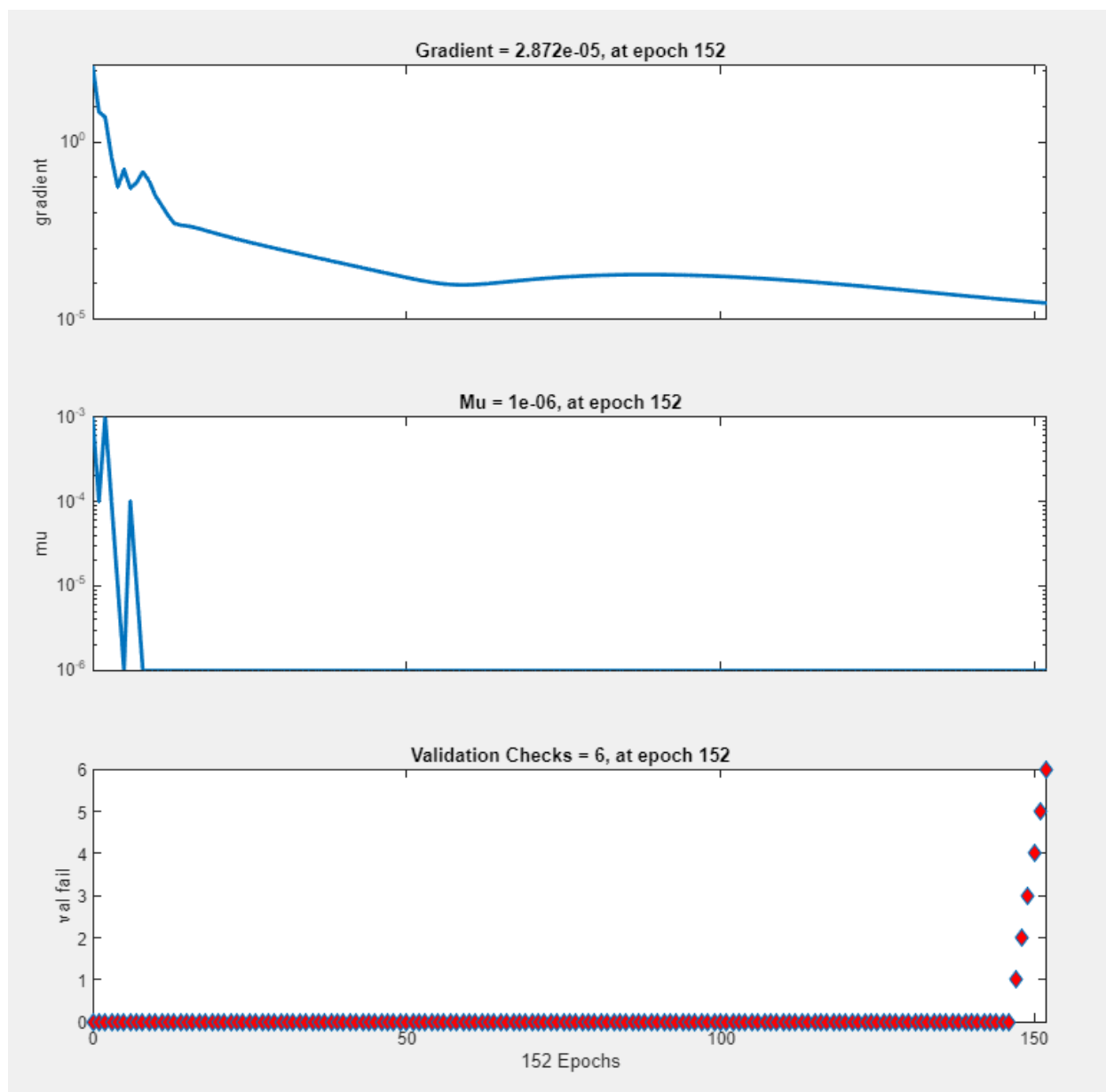
Propagacja wsteczna Levenberga-Marquardta – próba 2 (10 neurony w warstwie ukrytej)

Training Results

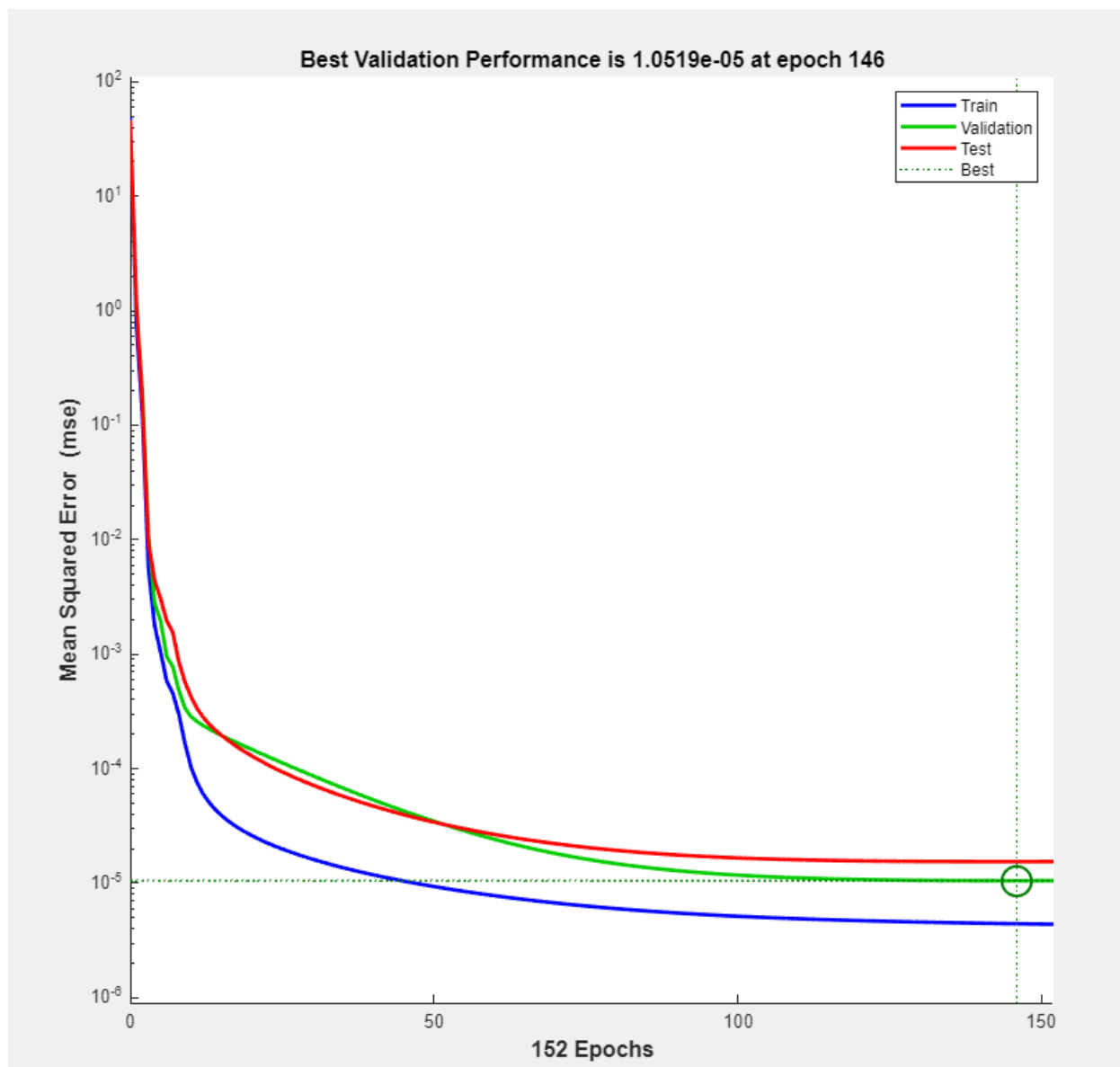
Training finished: Met validation criterion 

Training Progress

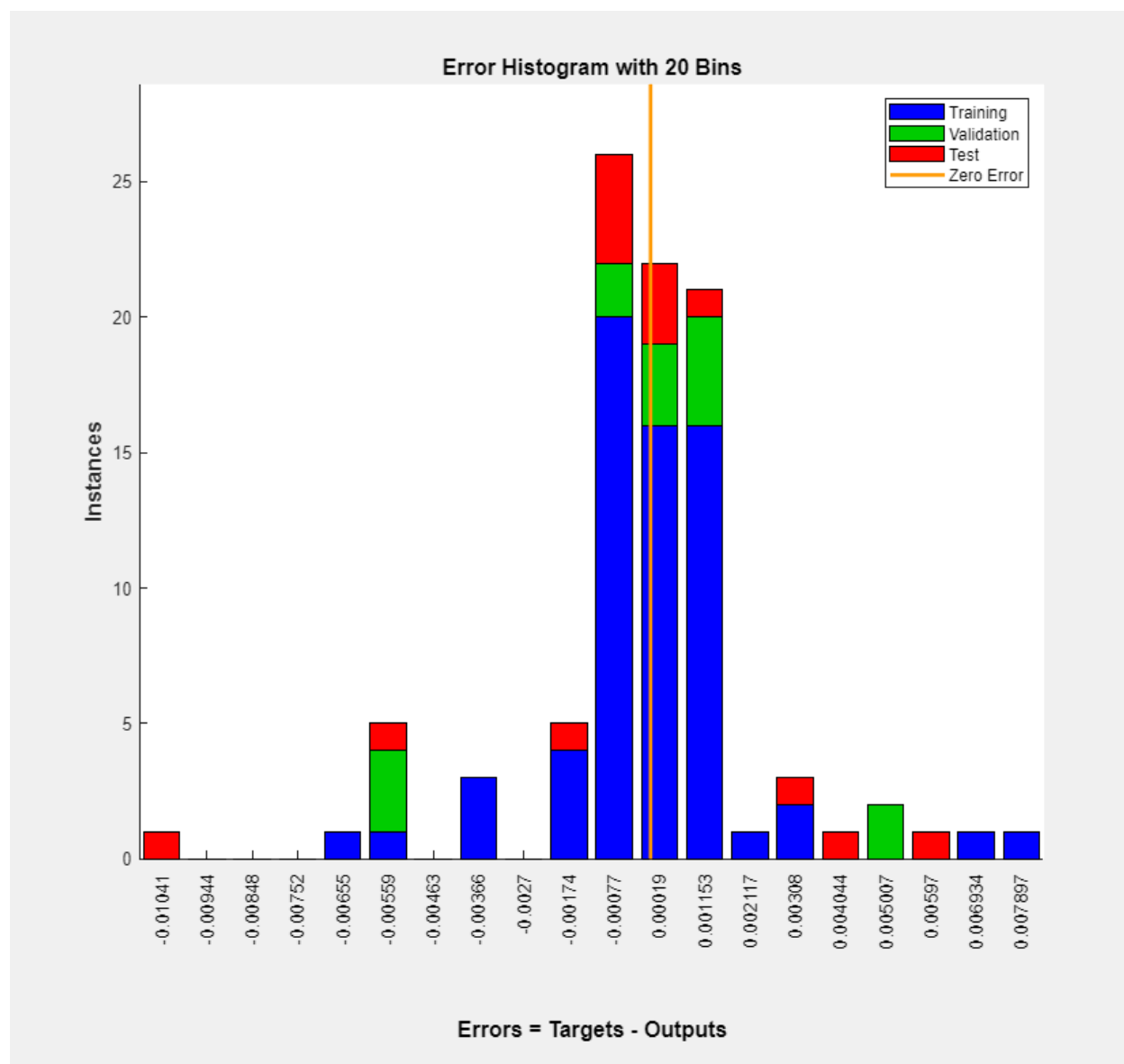
Unit	Initial Value	Stopped Value	Target Value
Epoch	0	152	1000
Elapsed Time	-	00:00:01	-
Performance	48.8	4.38e-06	0
Gradient	141	2.87e-05	1e-07
Mu	0.001	1e-06	1e+10
Validation Checks	0	6	6



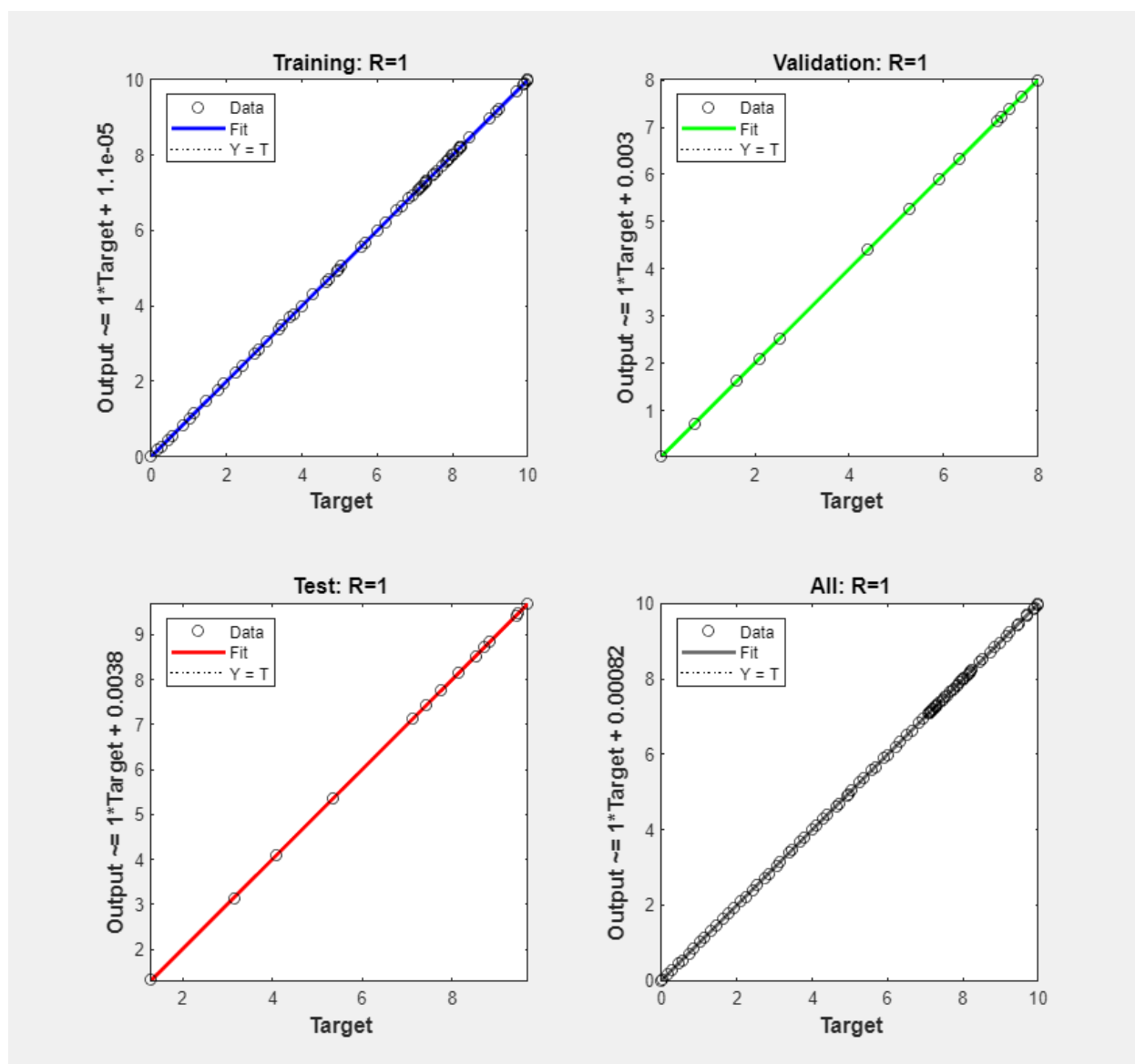
8 Wykres stanu szkolenia



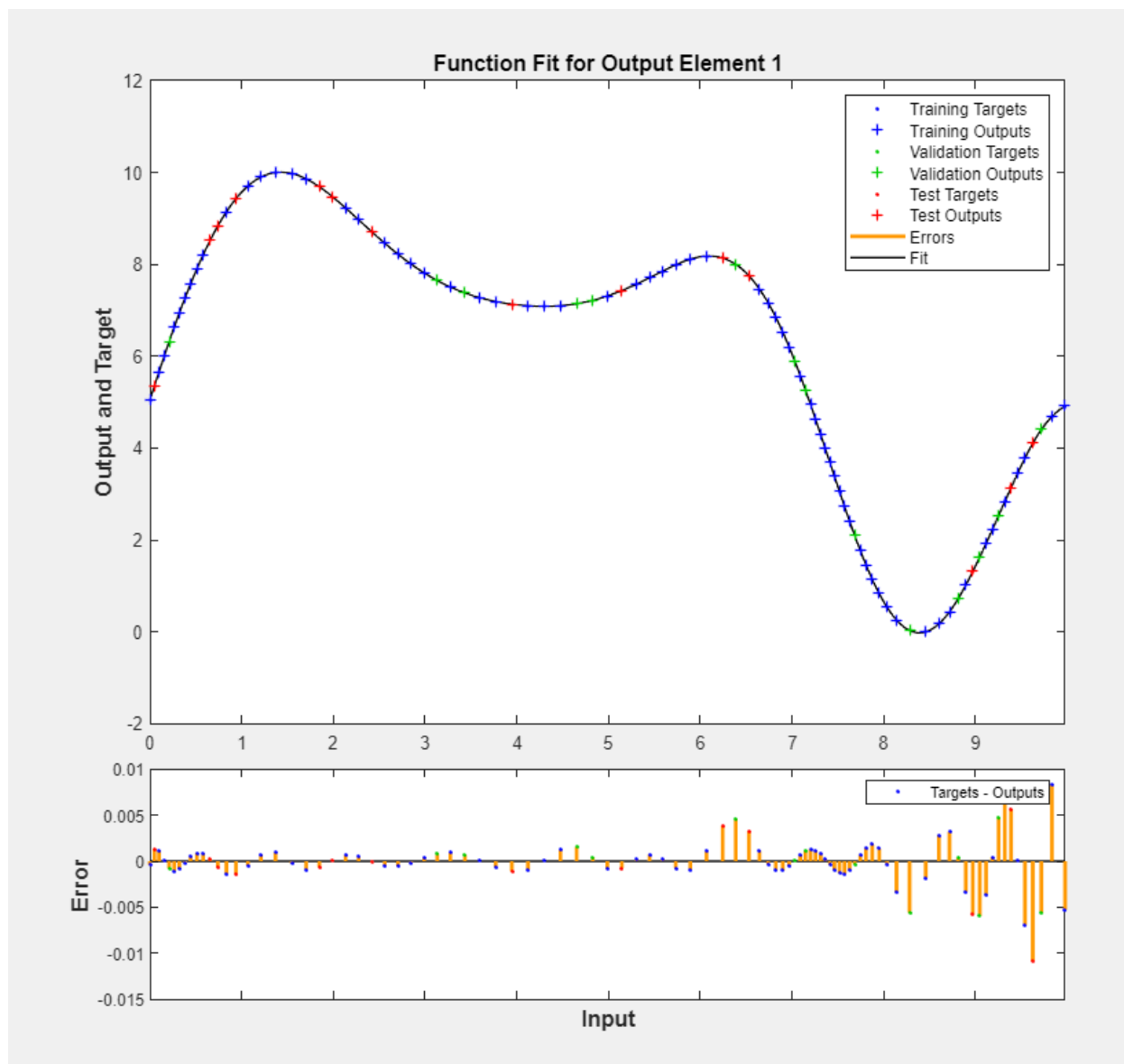
9 Błąd średnio kwadratowy



10 Histogram błędów




11 Wykres regresji



12 Dopasowanie funkcji dla pojedynczego elementu

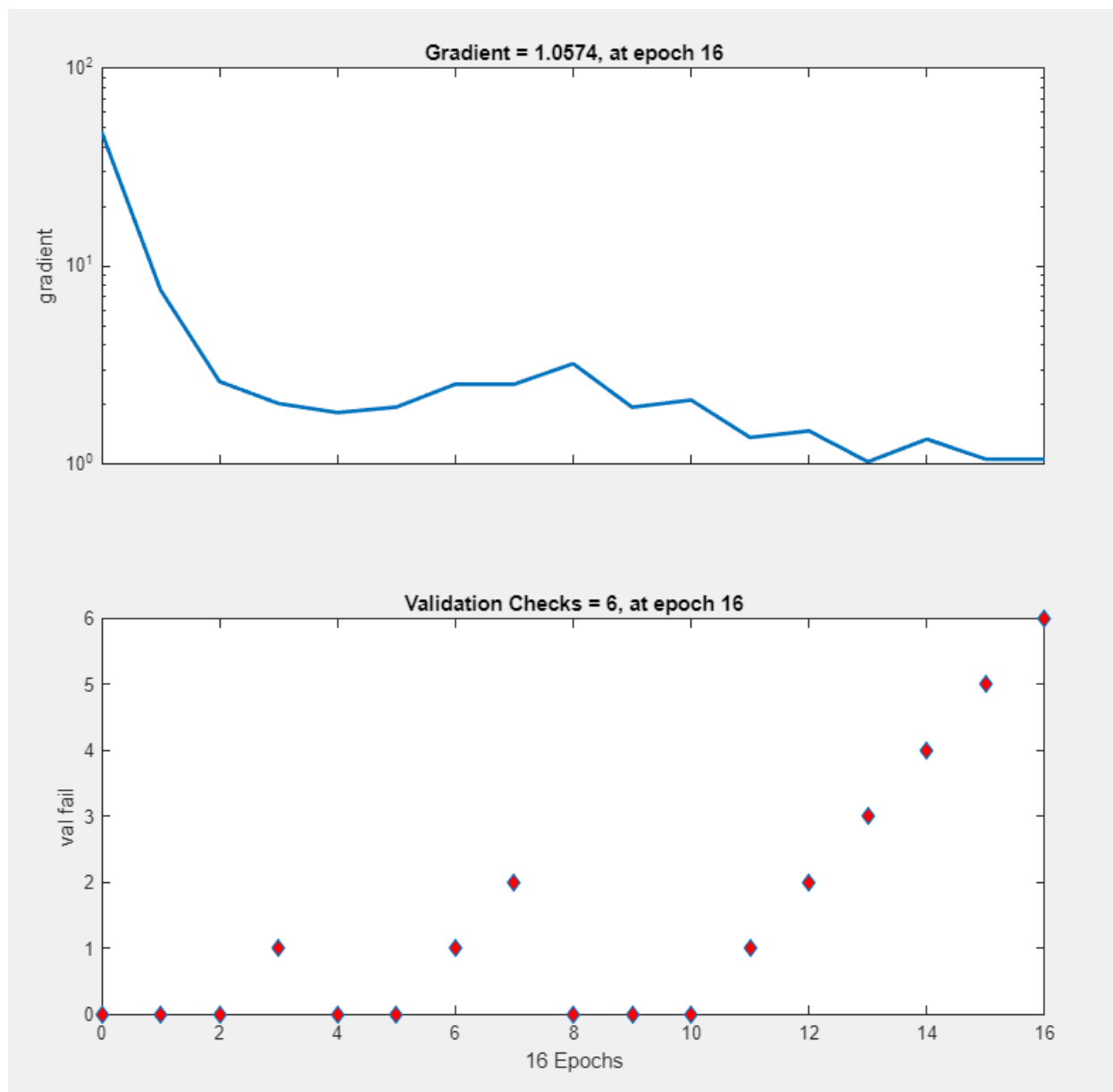
Propagacja wsteczna gradientu koniugatu skalowanego – próba 1 (3 neurony w warstwie ukrytej).

Training Results

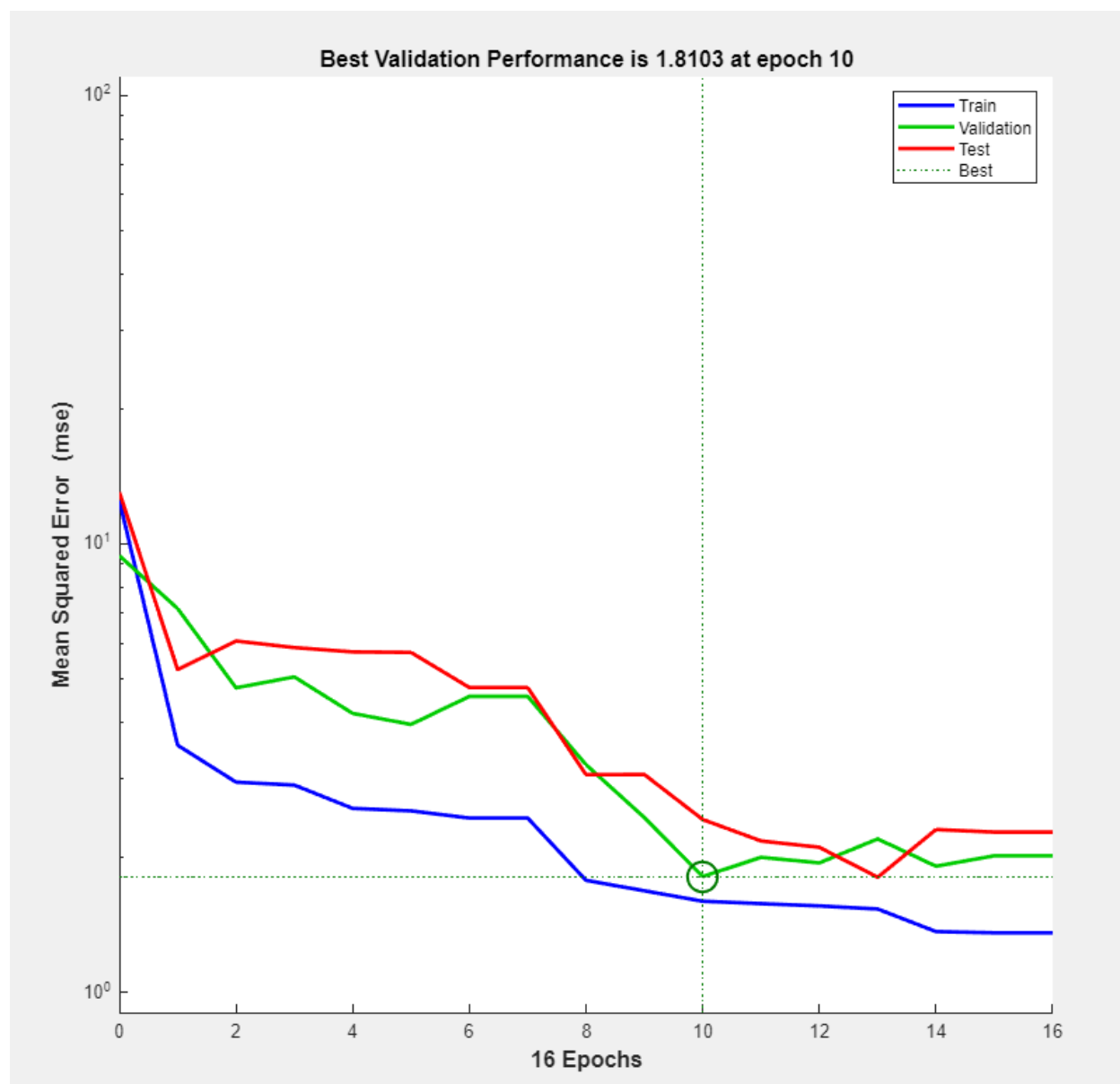
Training finished: Met validation criterion 

Training Progress

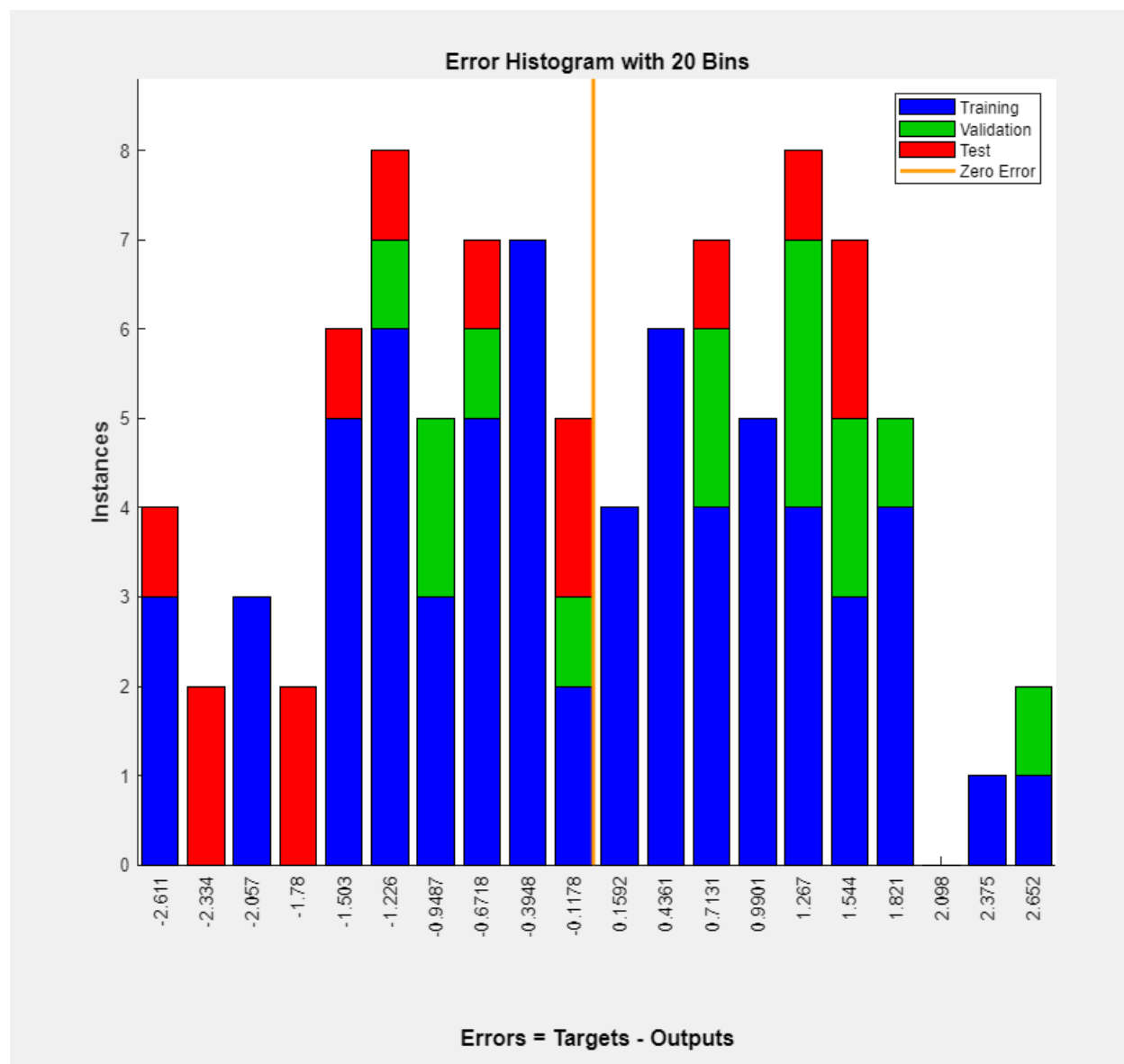
Unit	Initial Value	Stopped Value	Target Value	
Epoch	0	16	1000	▲
Elapsed Time	-	00:00:00	-	
Performance	12.5	1.36	0	
Gradient	47.8	1.06	1e-06	
Validation Checks	0	6	6	▼



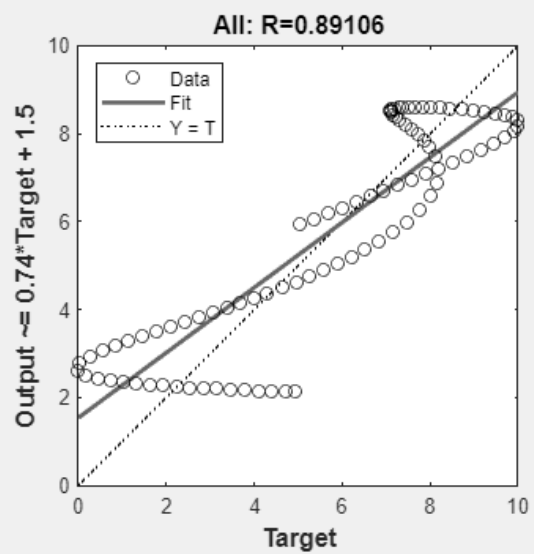
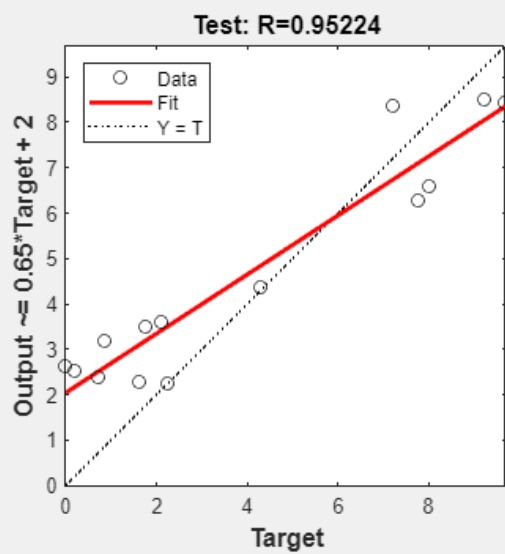
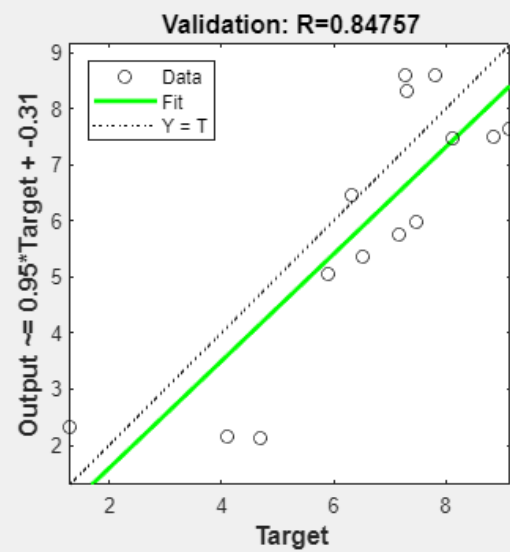
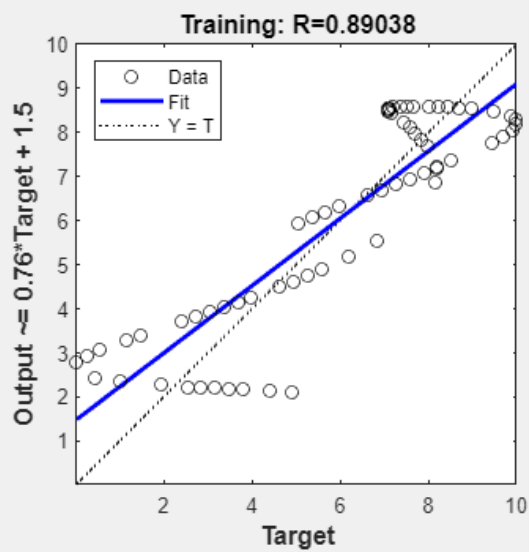
14 Wykres stanu szkolenia



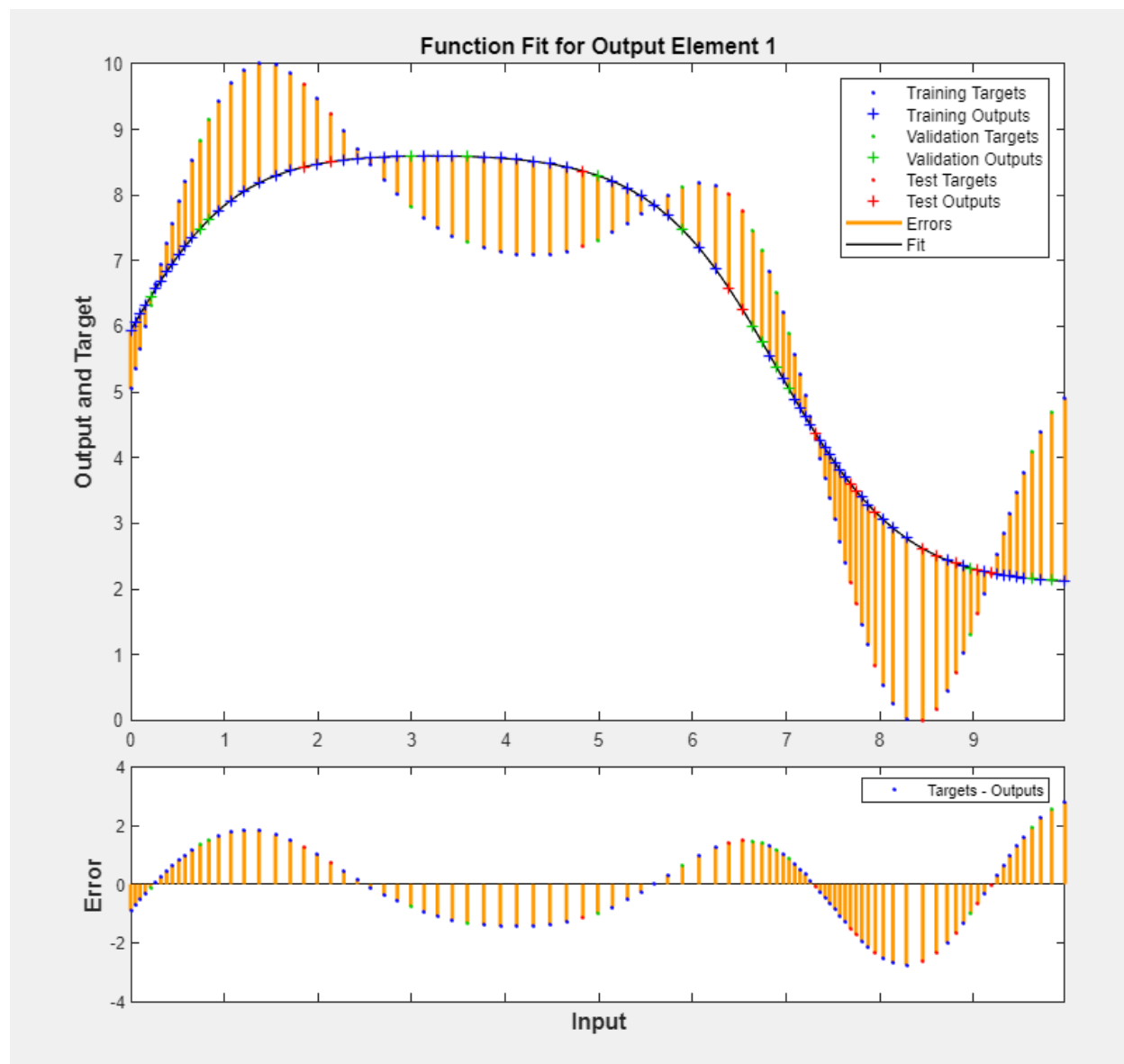
15 Błąd średnio kwadratowy



16 Histogram błędów




17 Wykres regresji




18 Dopasowanie funkcji dla pojedynczego elementu

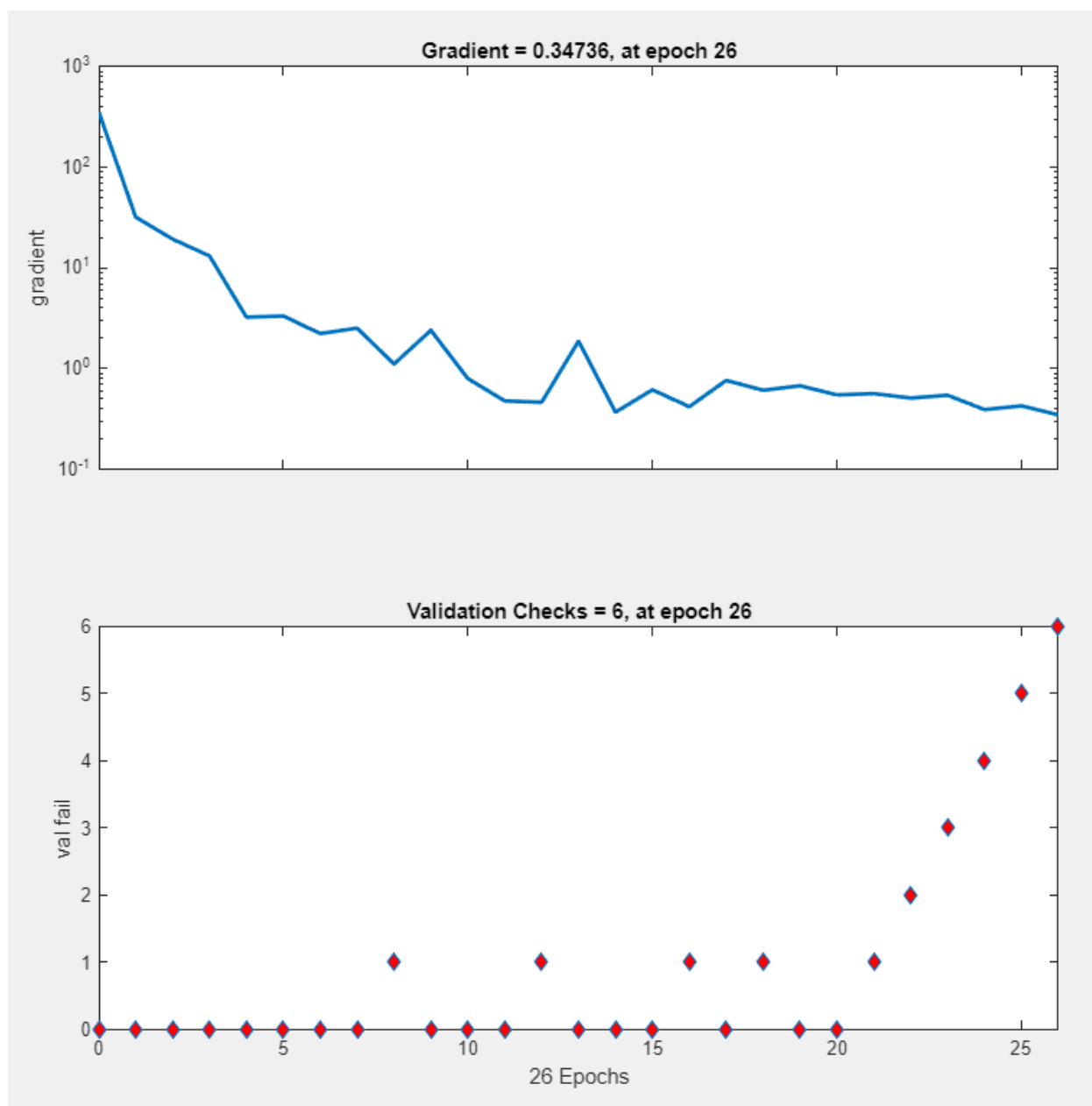
Propagacja wsteczna gradientu koniugatu skalowanego – próba 2 (10 neurony w warstwie ukrytej).

Training Results

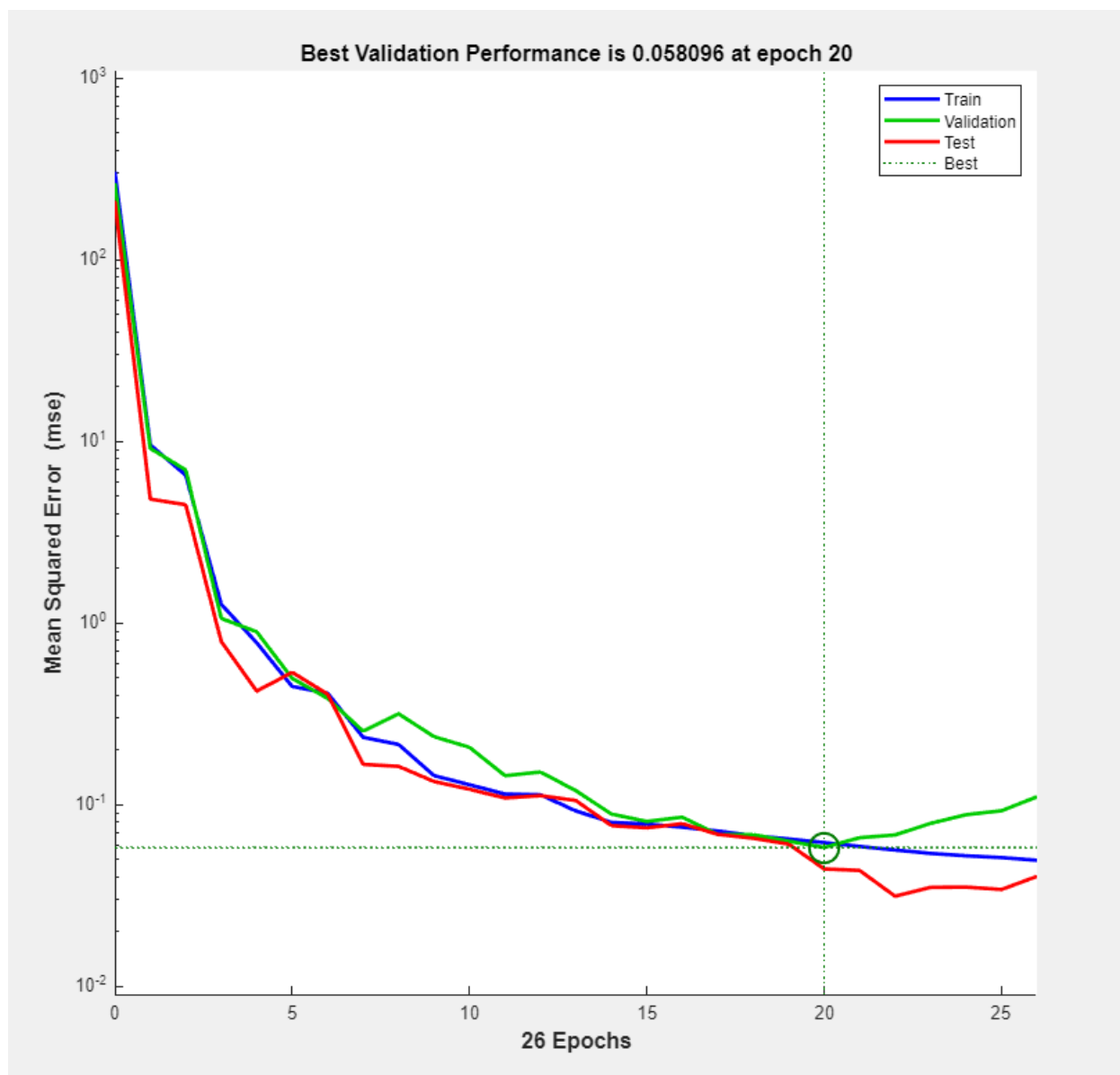
Training finished: Met validation criterion 

Training Progress

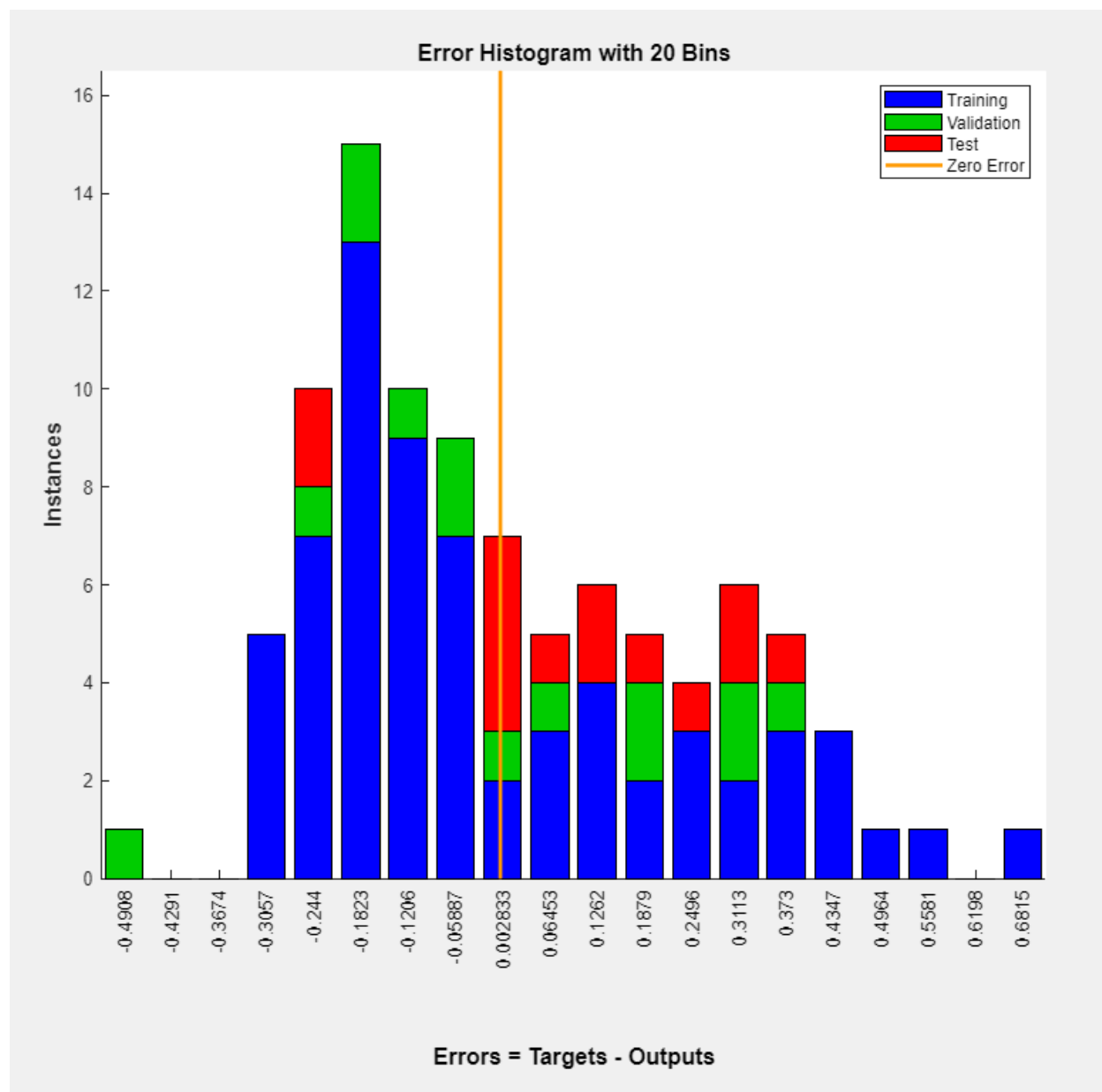
Unit	Initial Value	Stopped Value	Target Value	
Epoch	0	26	1000	
Elapsed Time	-	00:00:00	-	
Performance	306	0.0493	0	
Gradient	355	0.347	1e-06	
Validation Checks	0	6	6	



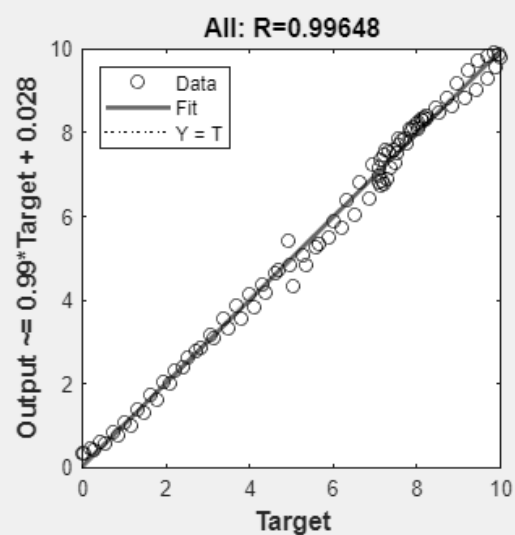
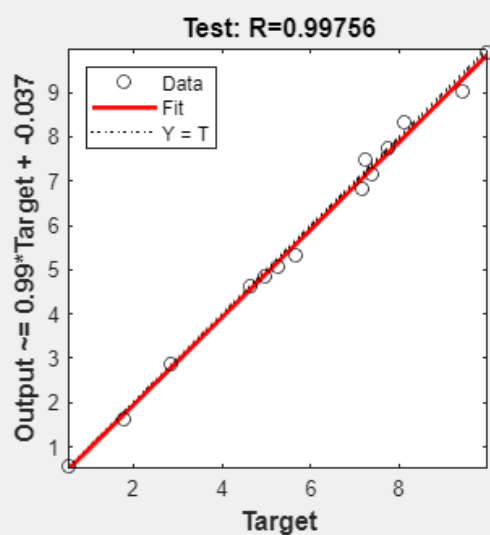
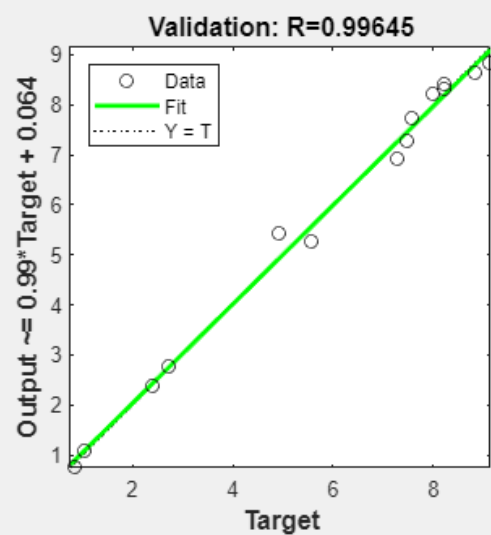
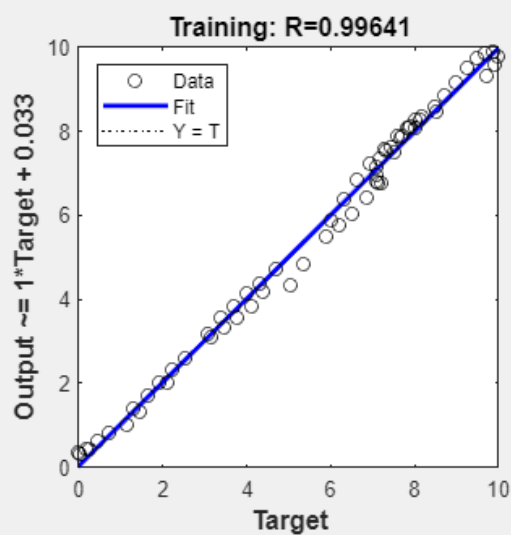
20 Wykres stanu szkolenia

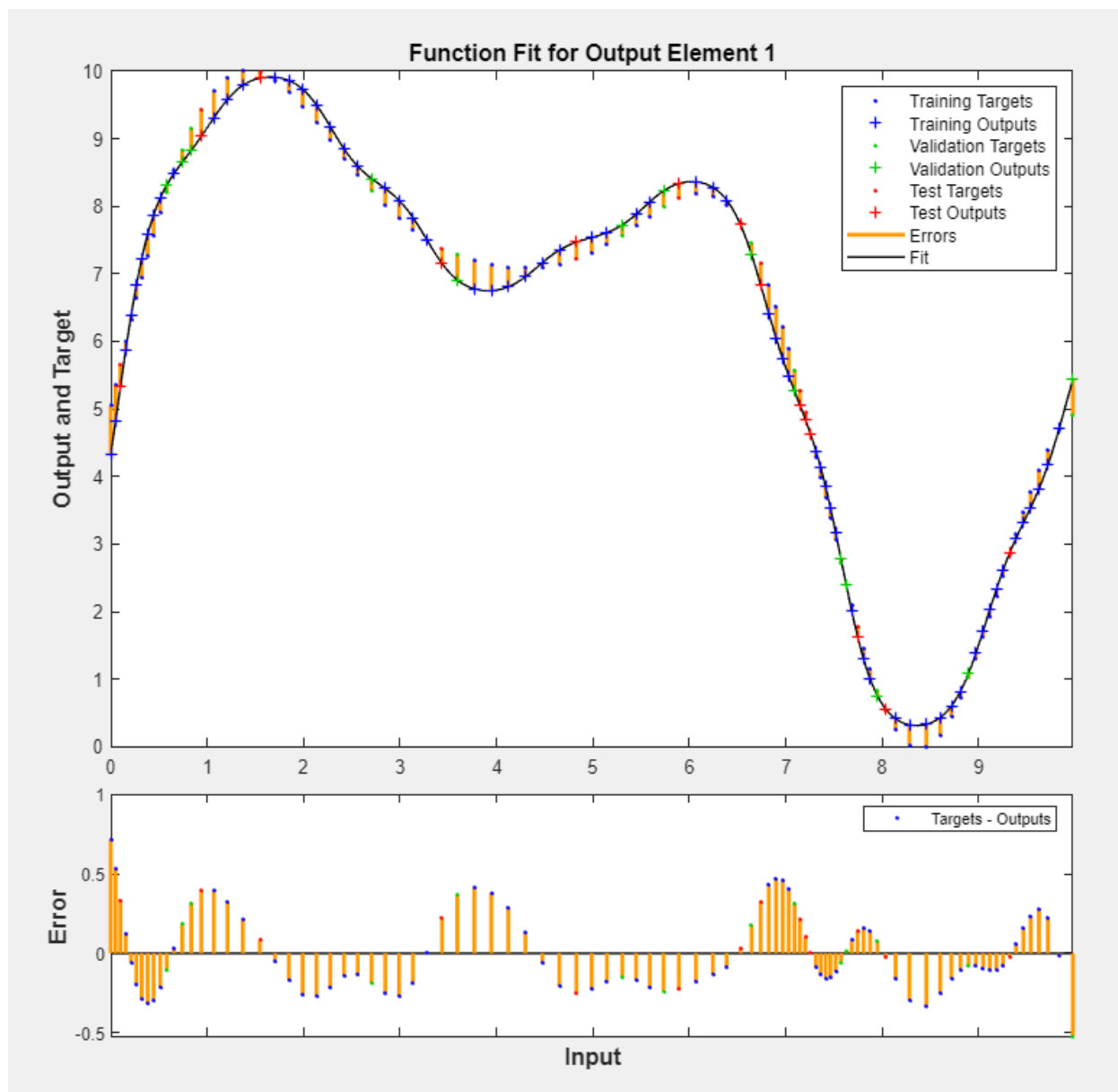


21 Błąd średnio kwadratowy



22 Histogram błędów





24 Dopasowanie funkcji dla pojedynczego elementu

Propagacja wsteczna regularyzacji Bayesowskiej – próba 1 (3 neurony w warstwie ukrytej)

Network

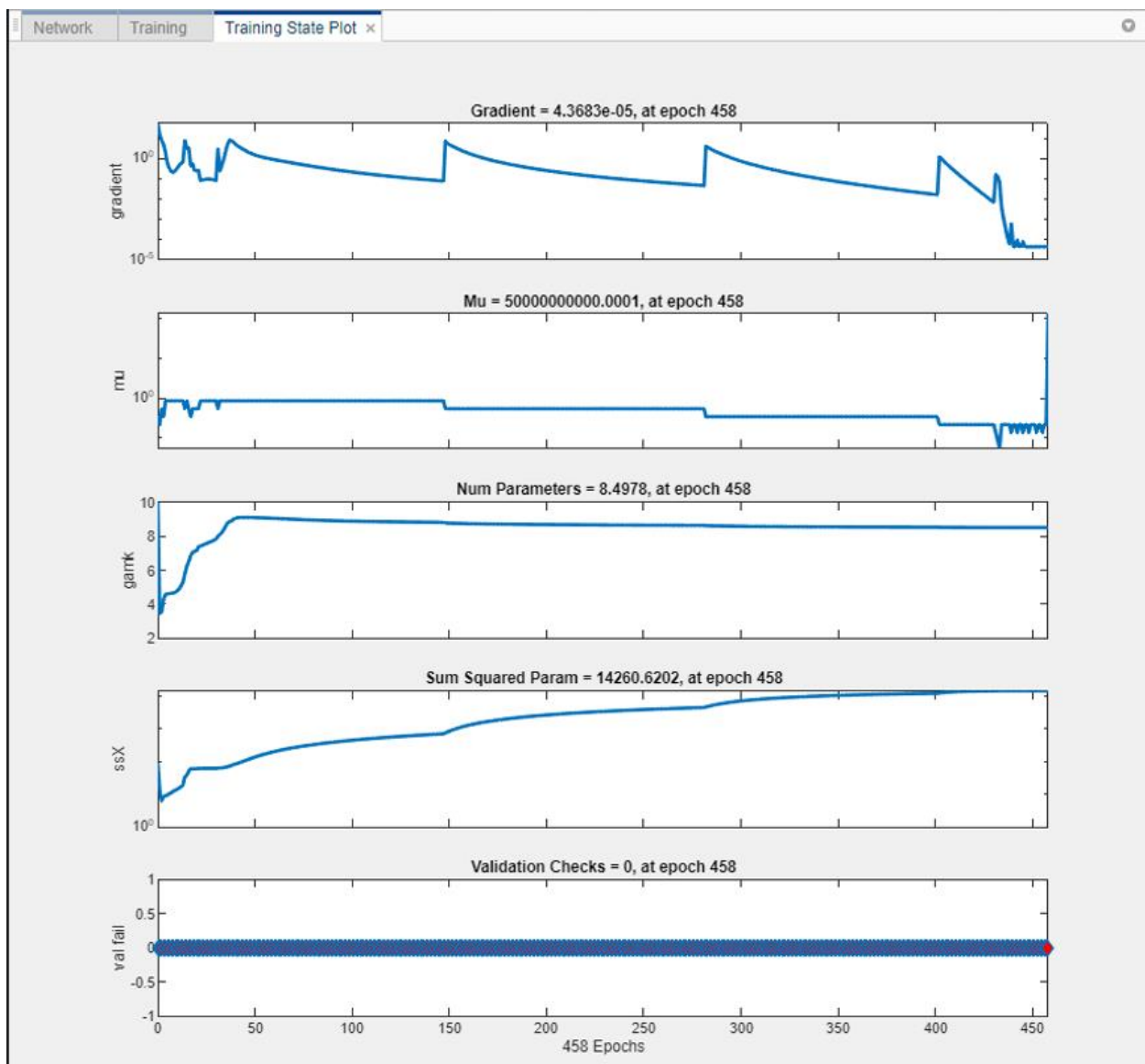
Training

Training Results

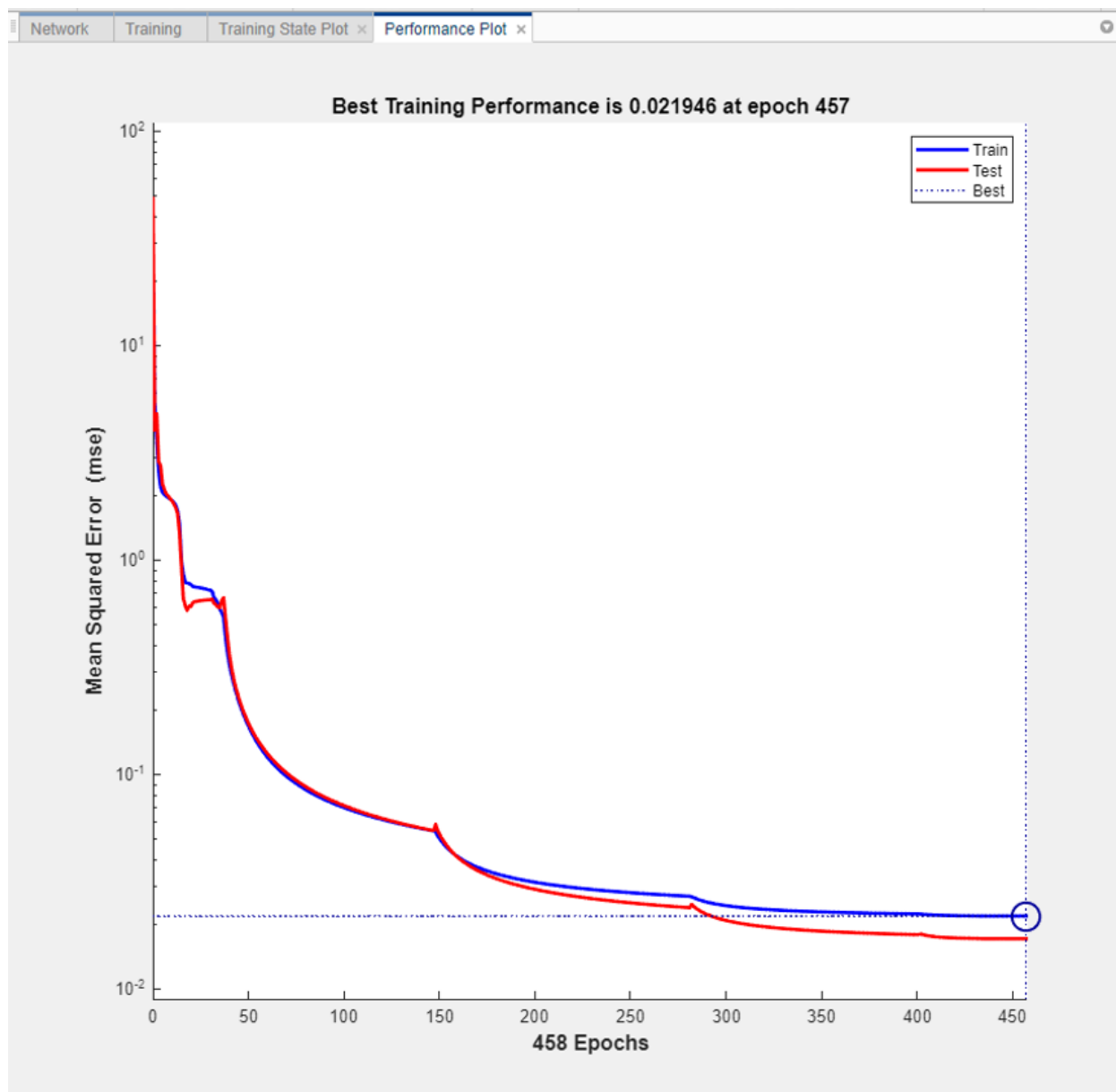
Training finished: Reached maximum mu

Training Progress

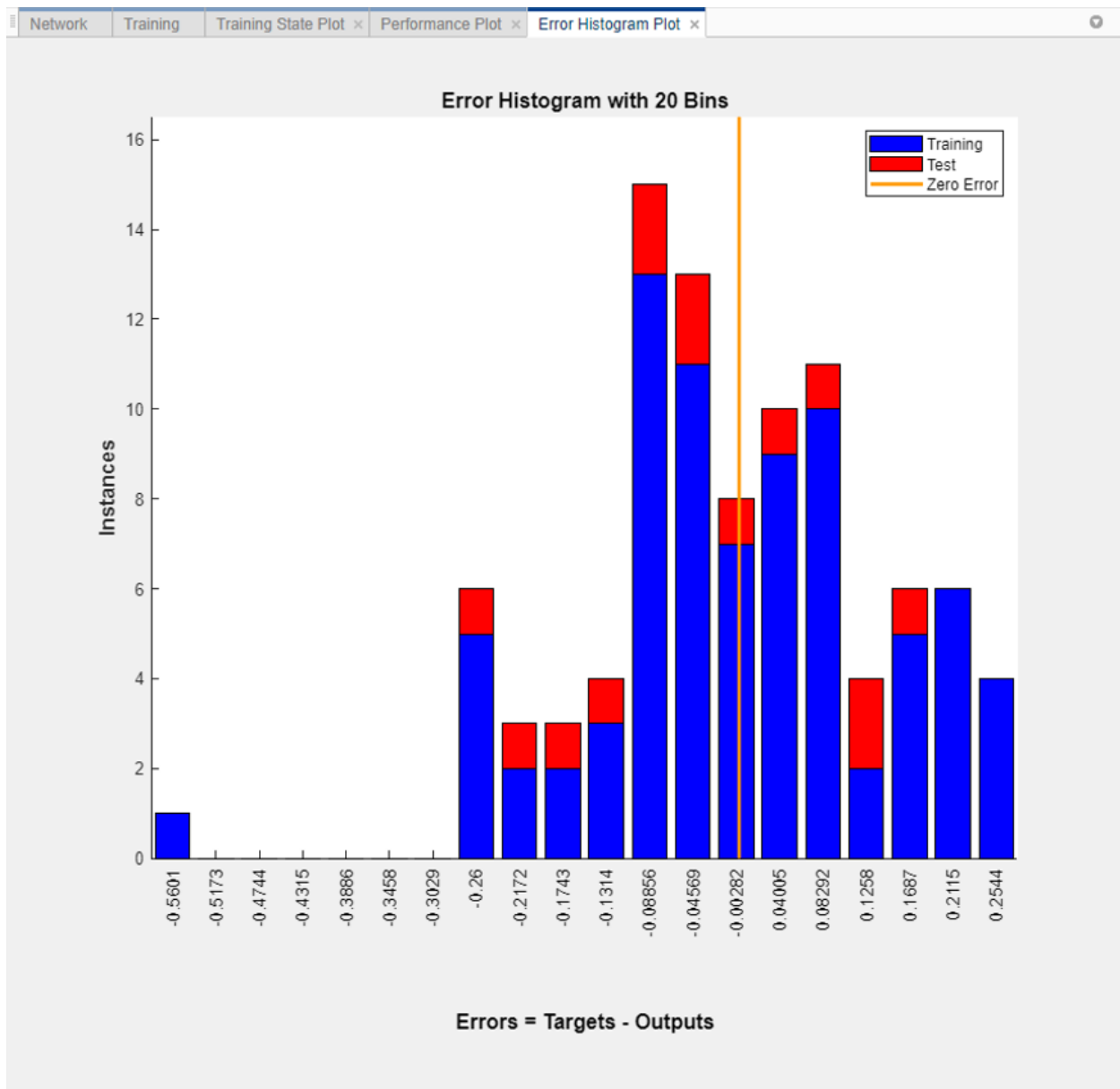
Unit	Initial Value	Stopped Value	Target Value
Epoch	0	458	1000
Elapsed Time	-	00:00:03	-
Performance	40.5	0.0219	0
Gradient	55.8	4.37e-05	1e-07
Mu	0.005	5e+10	1e+10
Effective # Param	10	8.5	0
Sum Squared Param	89	1.43e+04	0

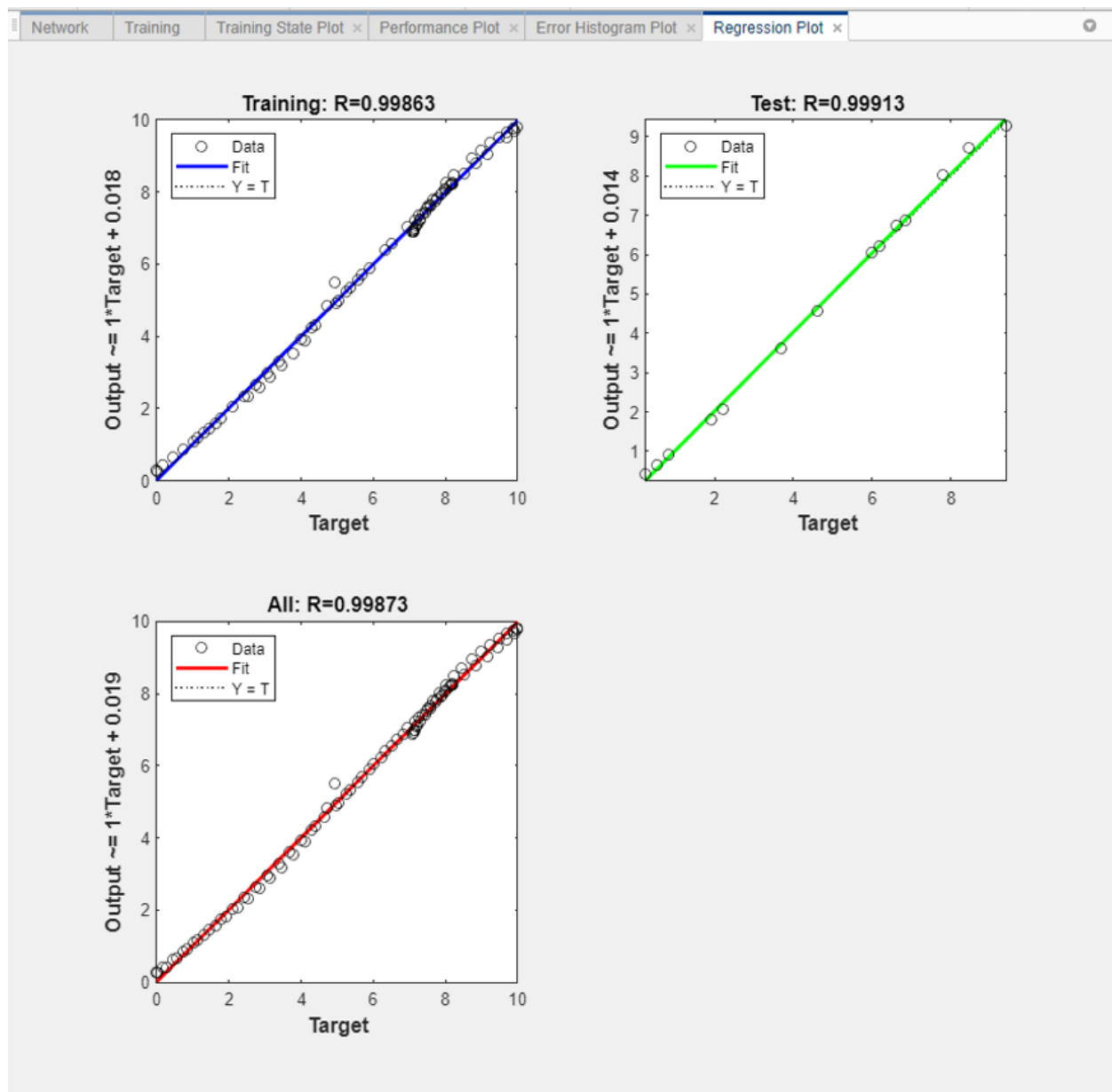


26 Wykres stanu szkolenia

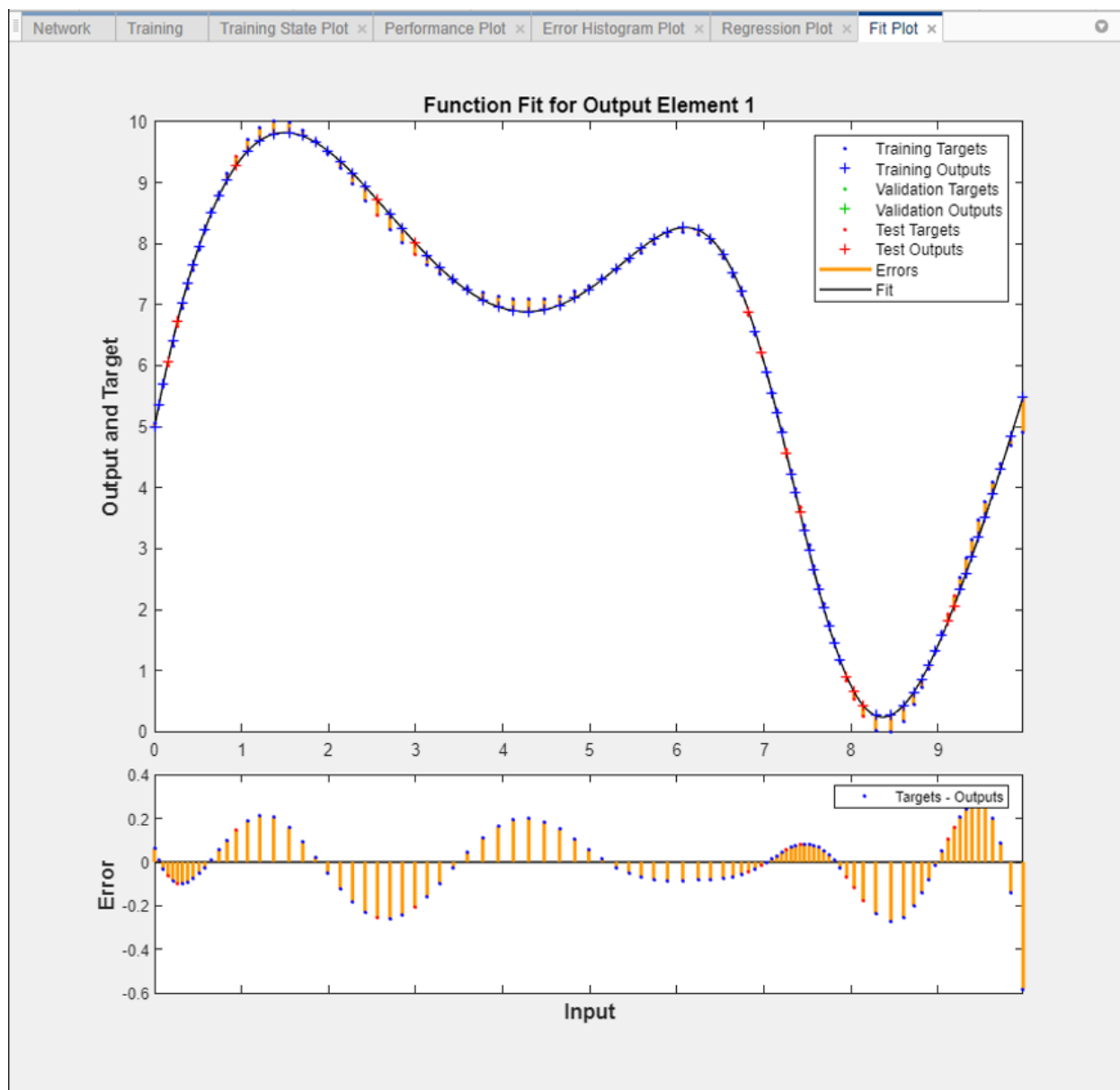


27 Błąd średnio kwadratowy





29 Wykres regresji




30 Dopasowanie funkcji dla pojedynczego elementu

Propagacja wsteczna regularyzacji Bayesowskiej – próba 2 (10 neuronów w warstwie ukrytej)

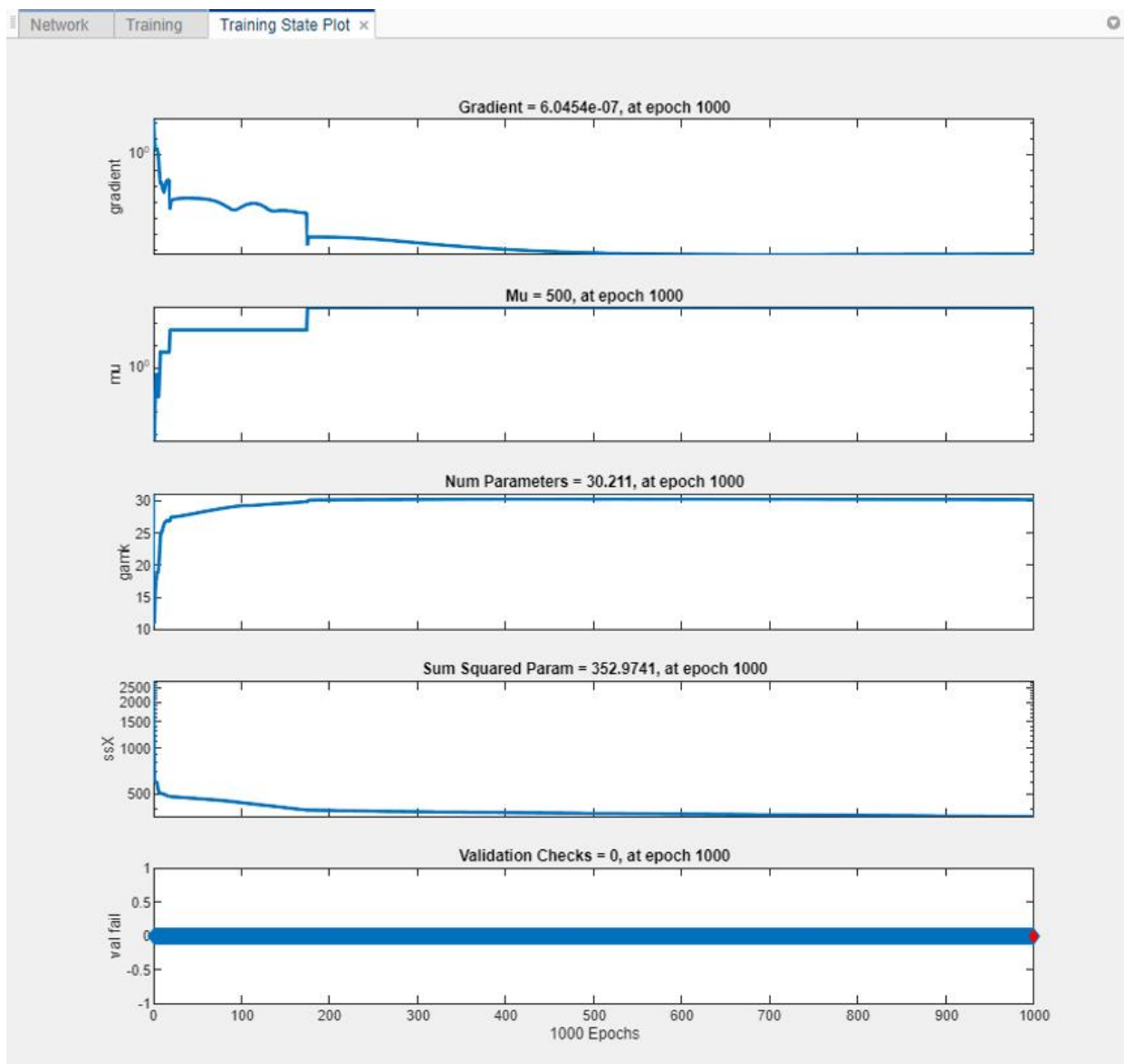
NetworkTraining

Training Results

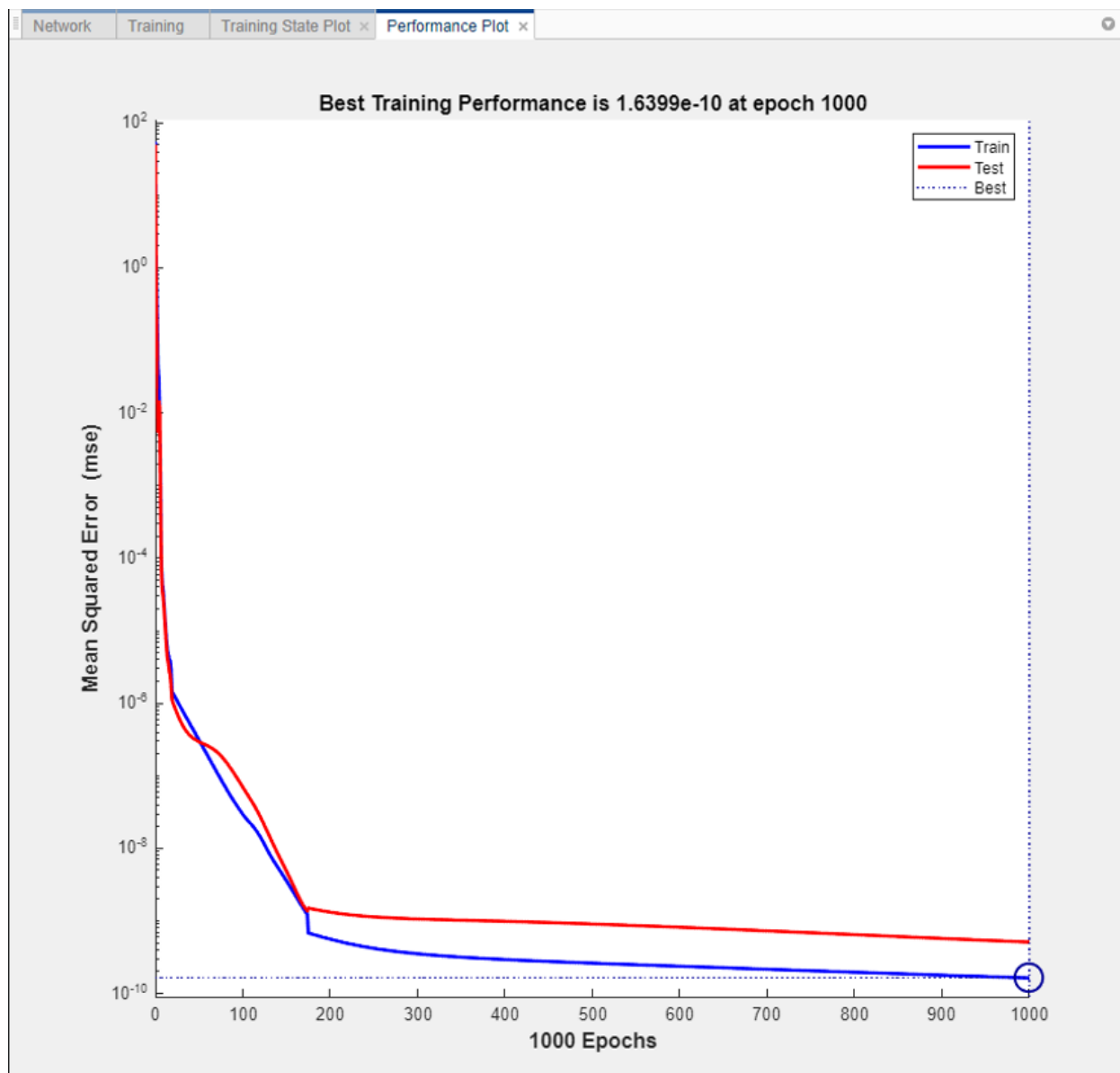
Training finished: Reached maximum number of epochs 

Training Progress

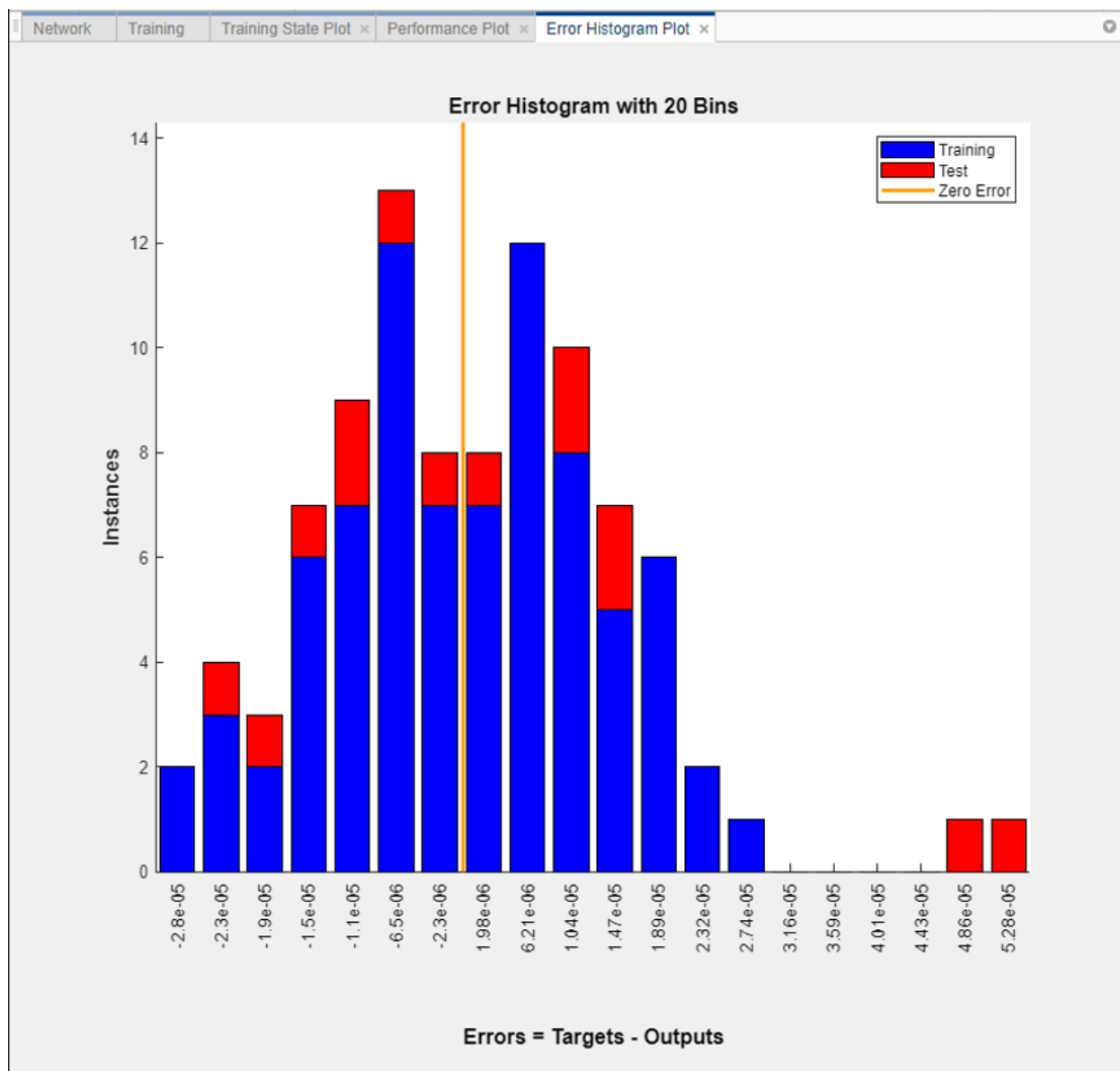
Unit	Initial Value	Stopped Value	Target Value
Epoch	0	1000	1000
Elapsed Time	-	00:00:06	-
Performance	54.9	1.64e-10	0
Gradient	157	6.05e-07	1e-07
Mu	0.005	500	1e+10
Effective # Param	31	30.2	0
Sum Squared Param	2.76e+03	353	0



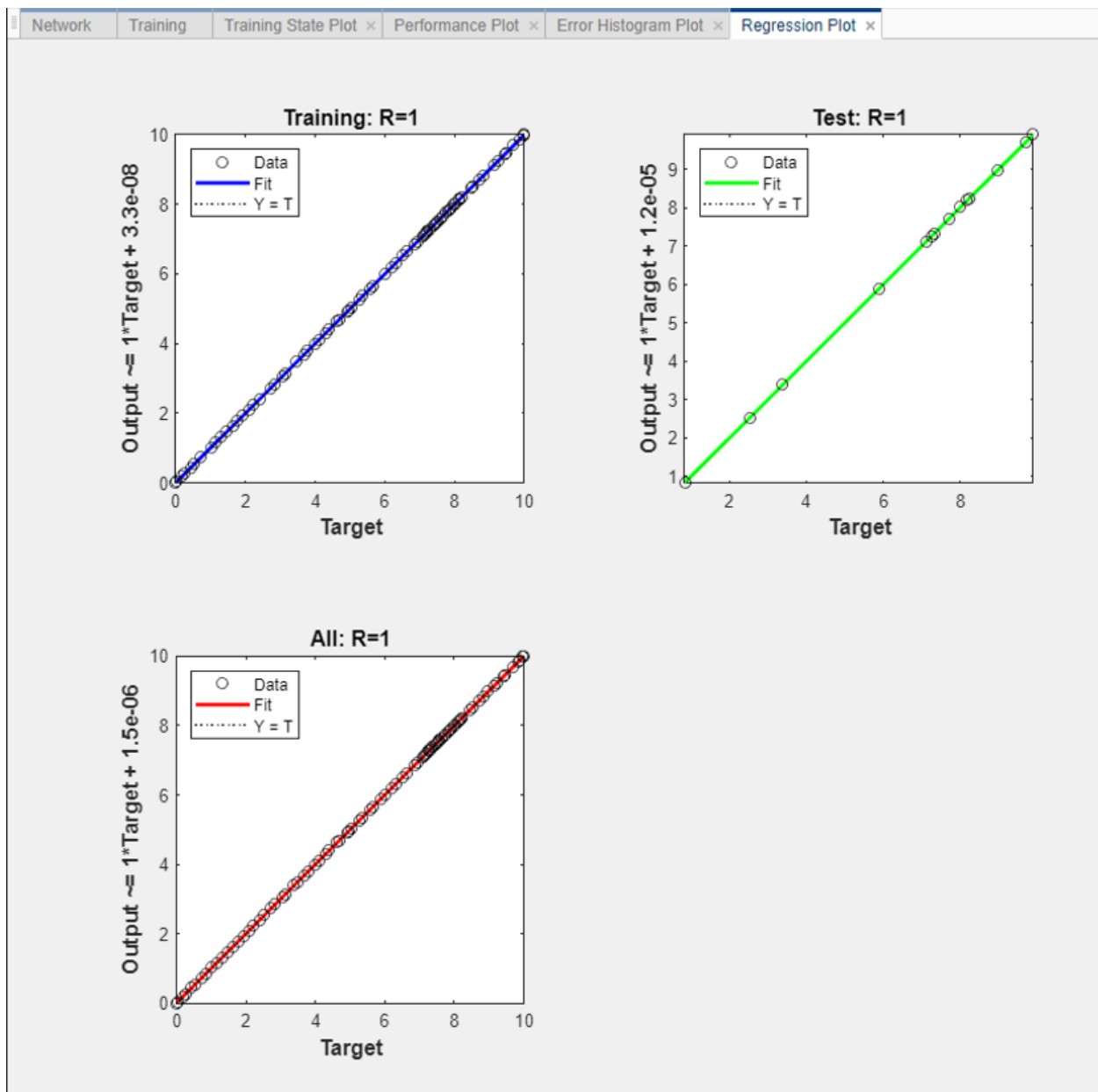
32 Wykres stanu szkolenia



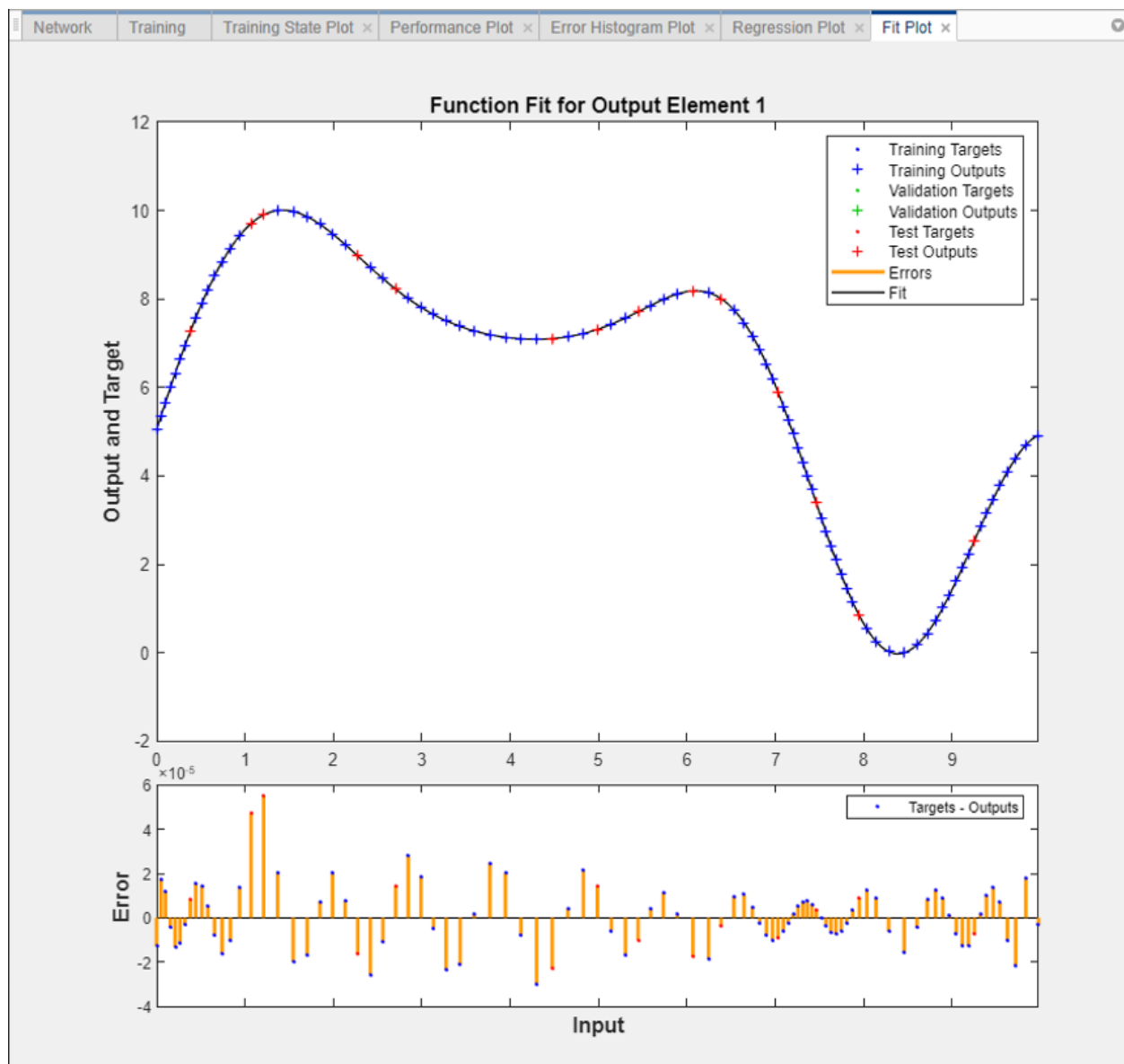
33 Błąd średnio kwadratowy



34 Histogram błędów

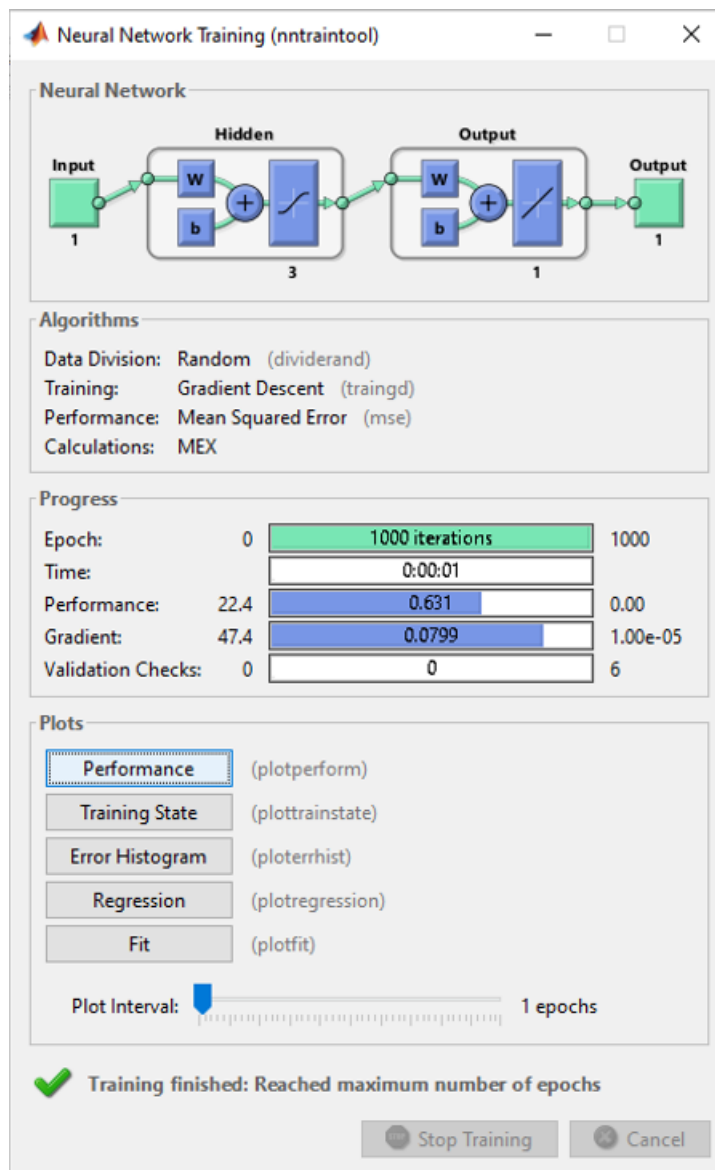


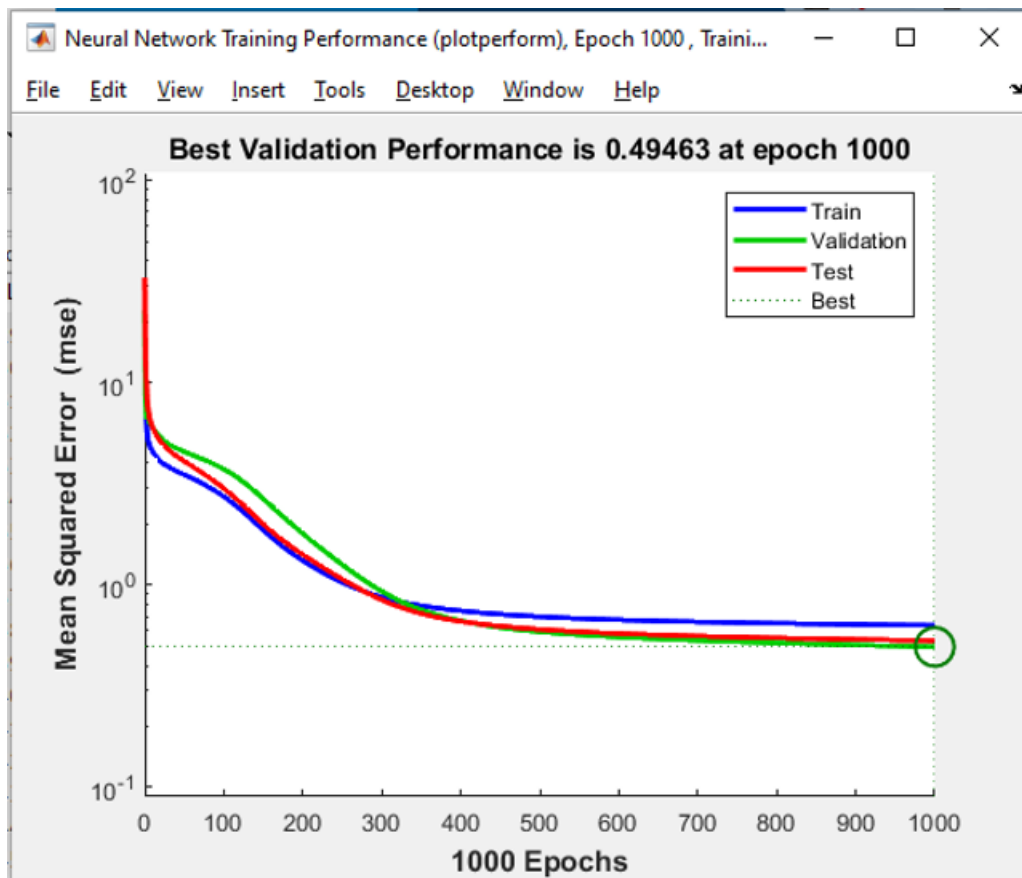
35 Wykres regresji



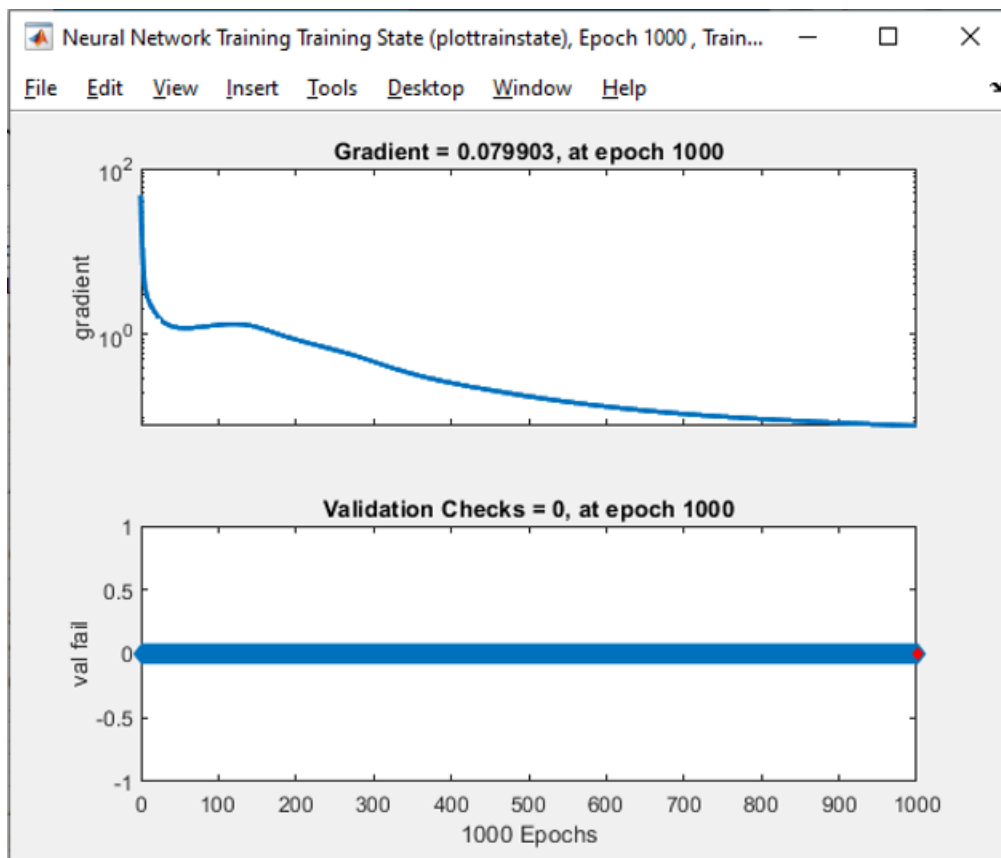
36 Dopasowanie funkcji dla pojedynczego elementu

Zejsćie gradientowe – próba 1 (3 neurony w warstwie ukrytej)

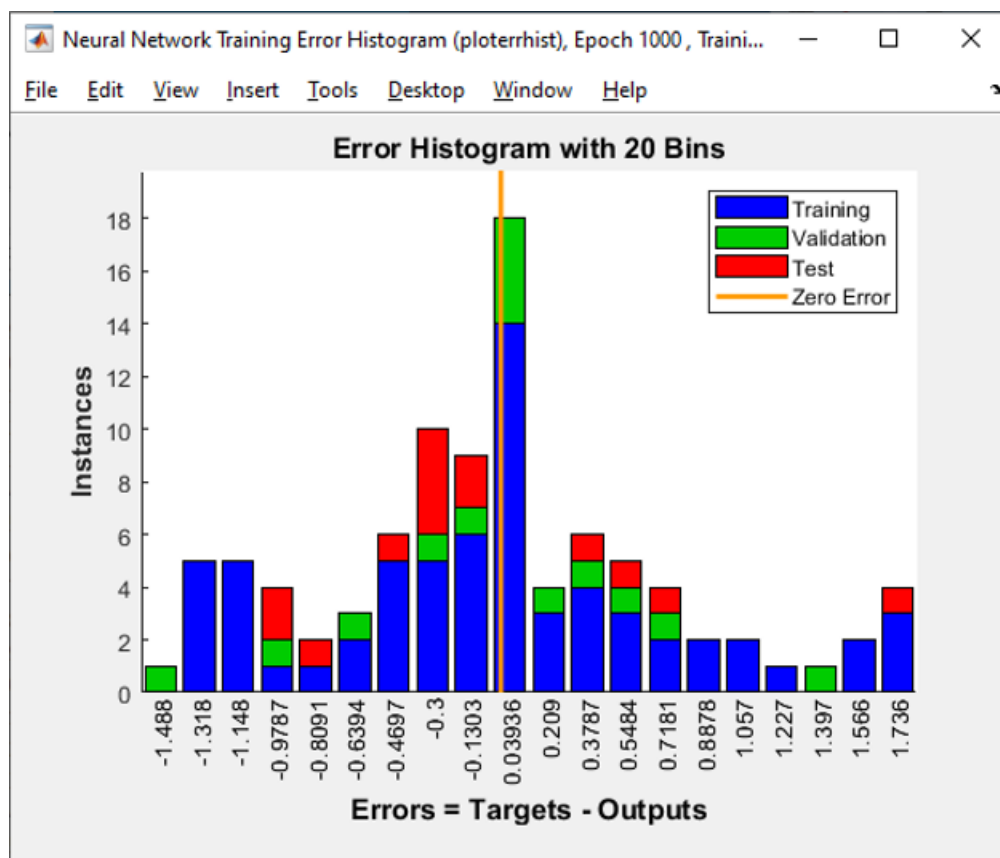




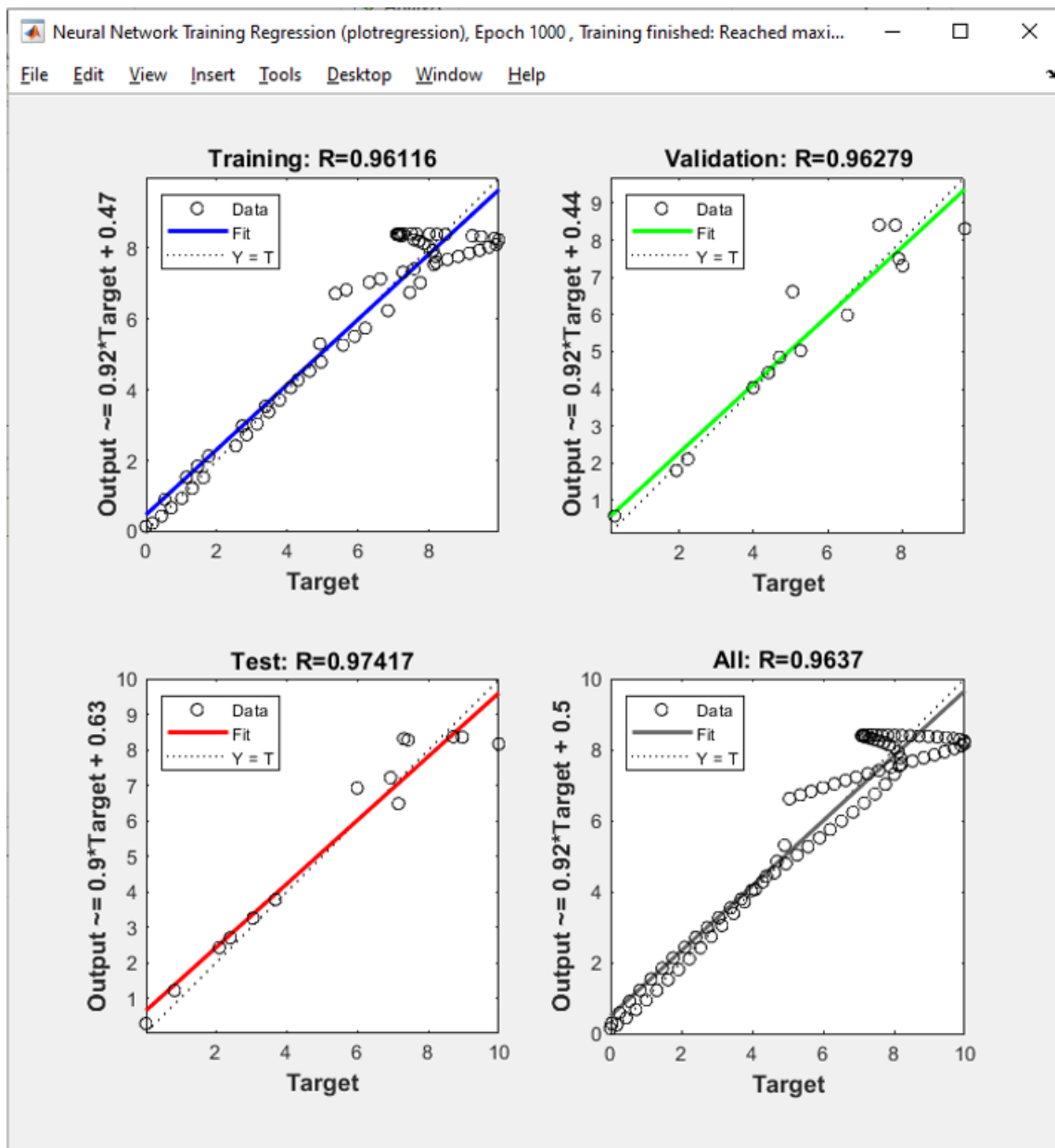
38 Błąd średnio kwadratowy



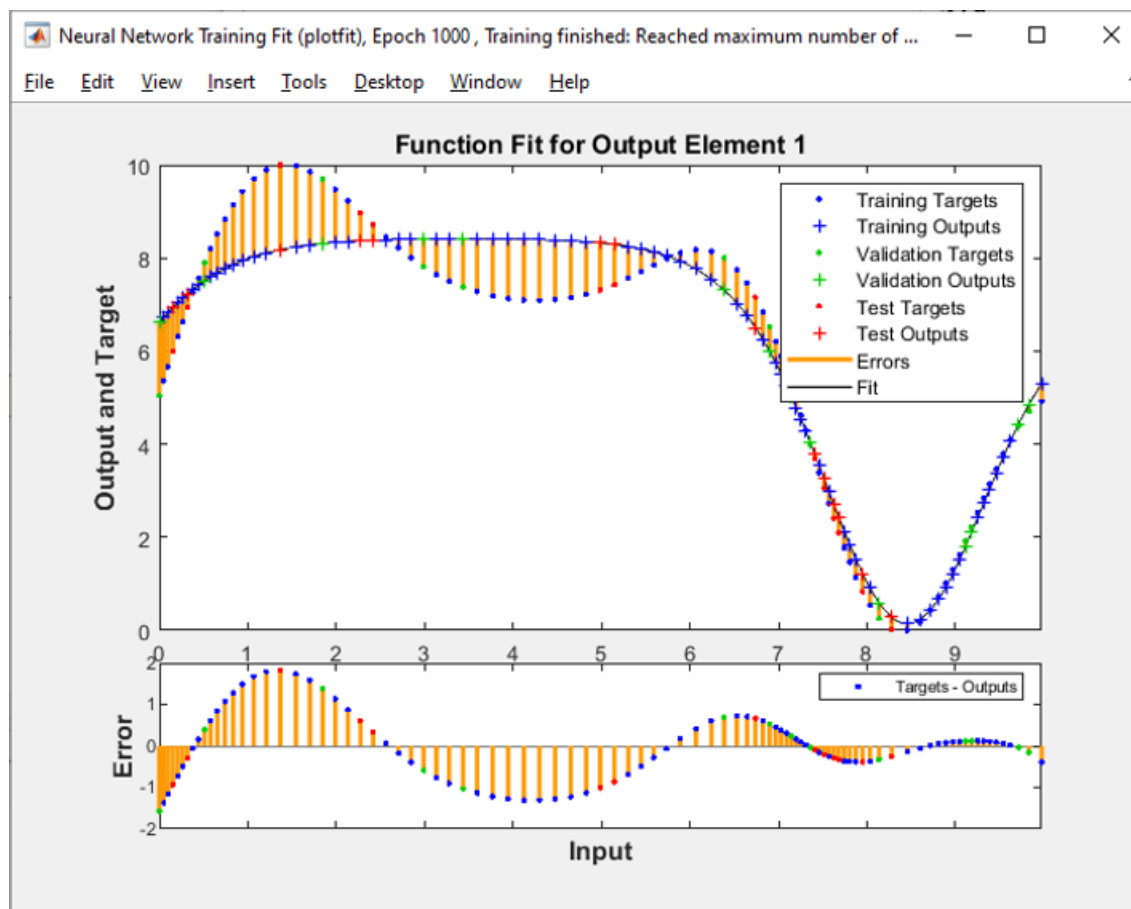
39 Wykres stanu szkolenia



40 Histogram błędów

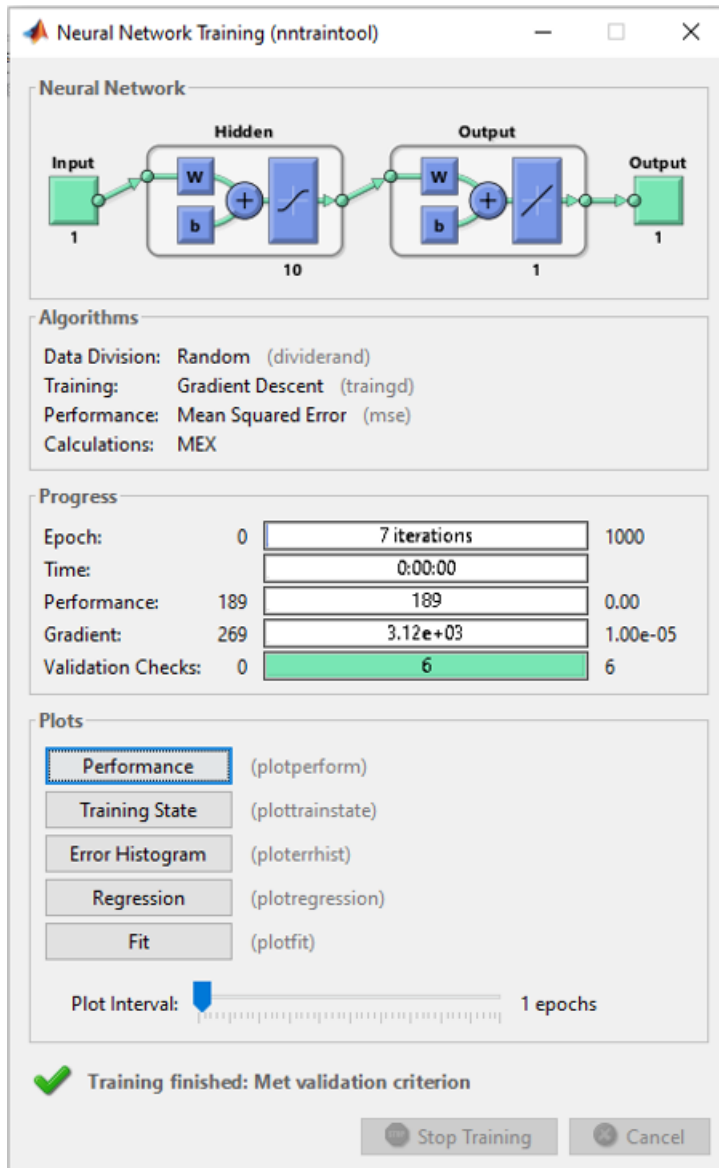


41 Wykres regresji

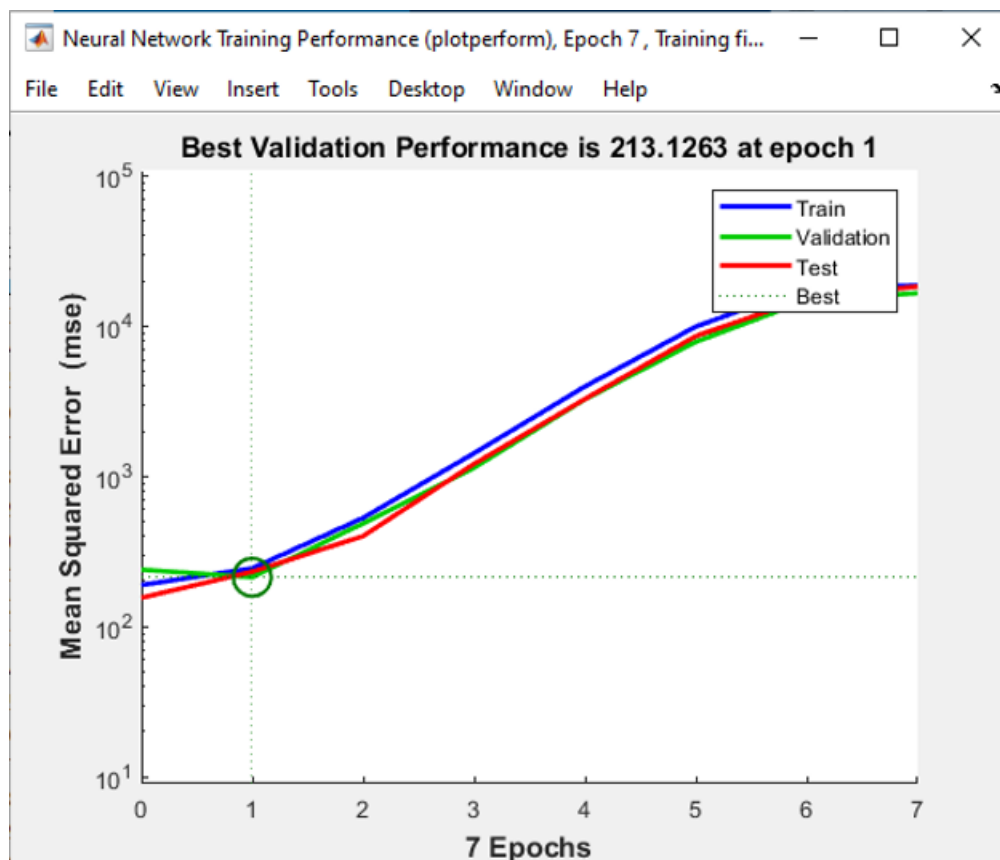


42 Dopasowanie funkcji dla pojedynczego elementu

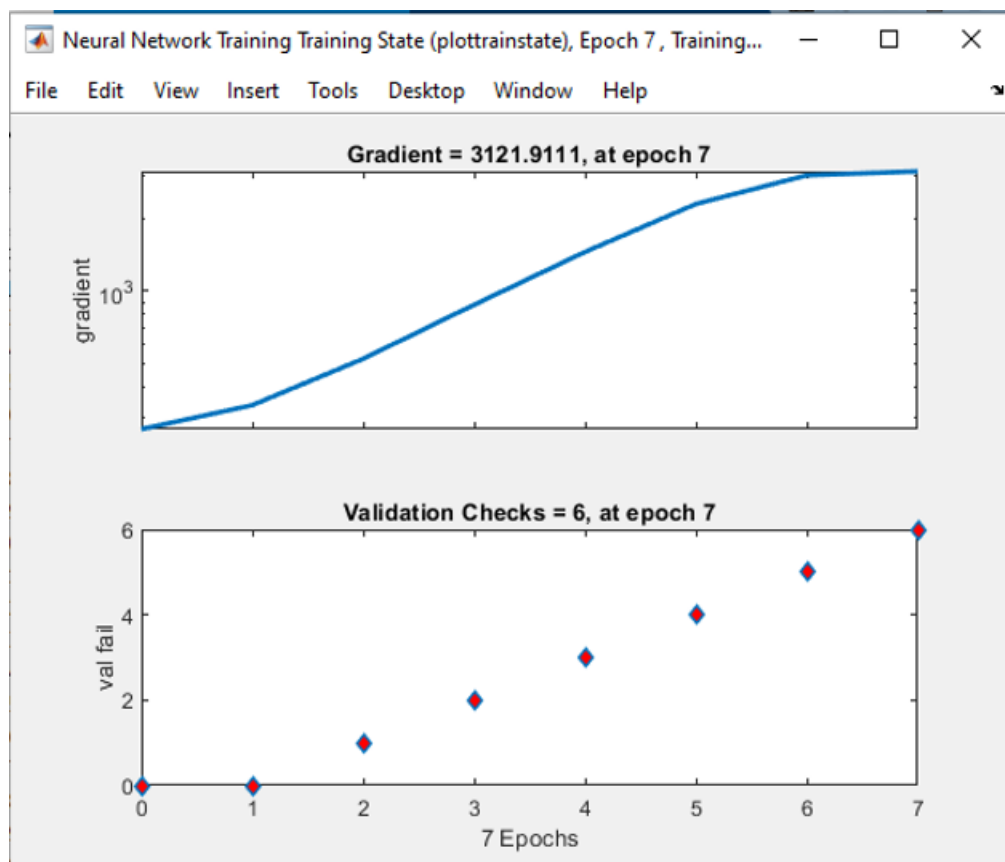
Zejście gradientowe – próba 2 (10 neuronów w warstwie ukrytej)



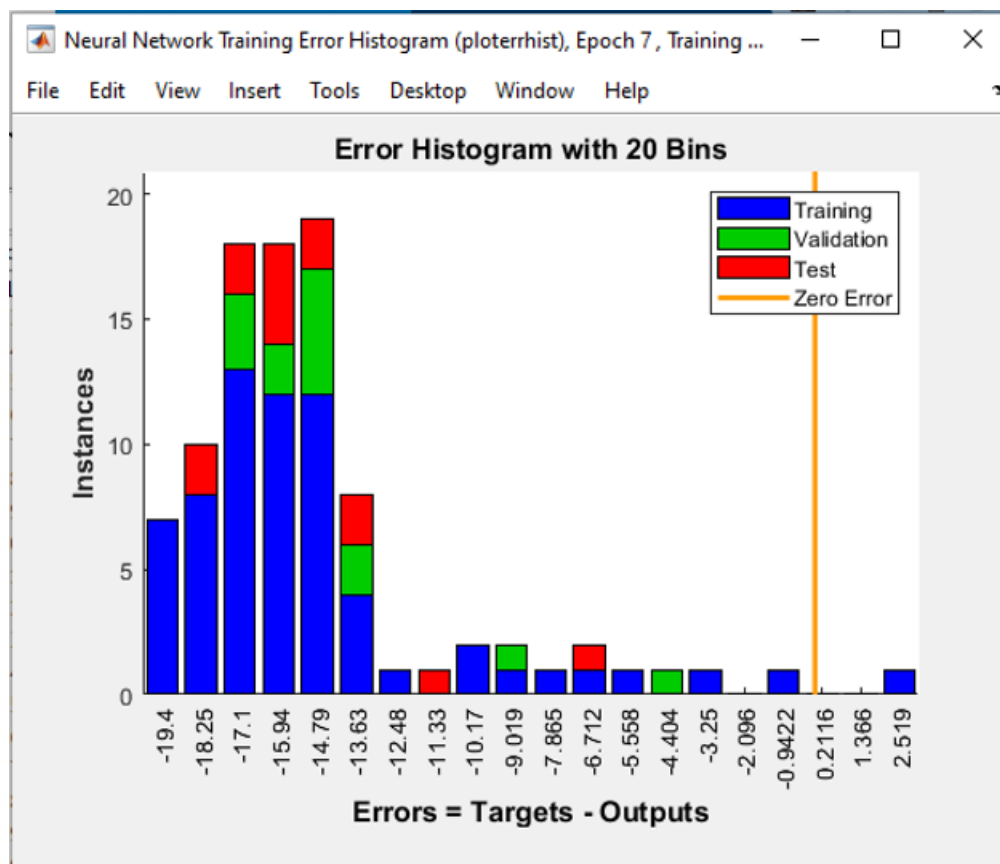
43 Uczenie się sieci



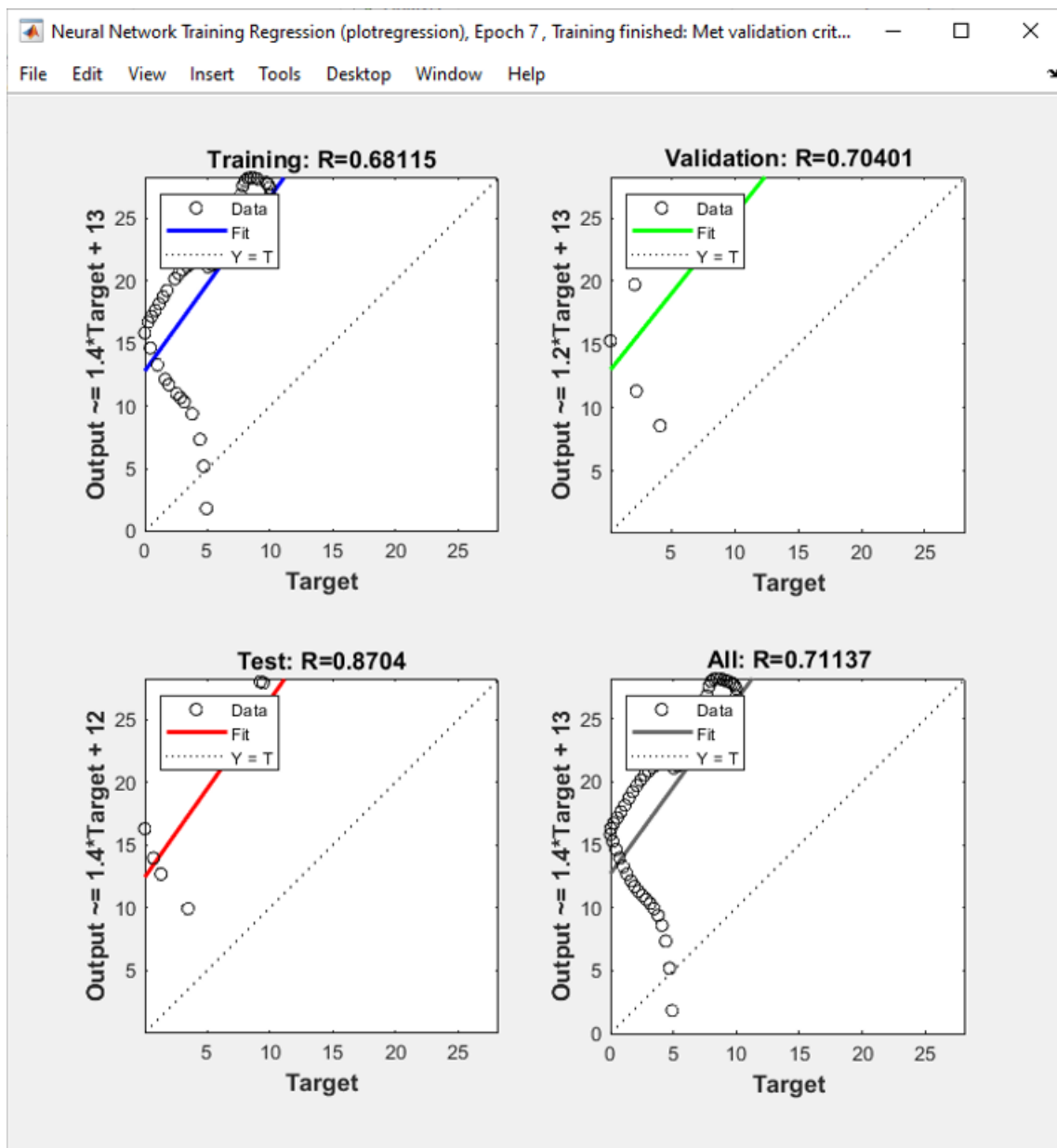
44 Błąd średnio kwadratowy



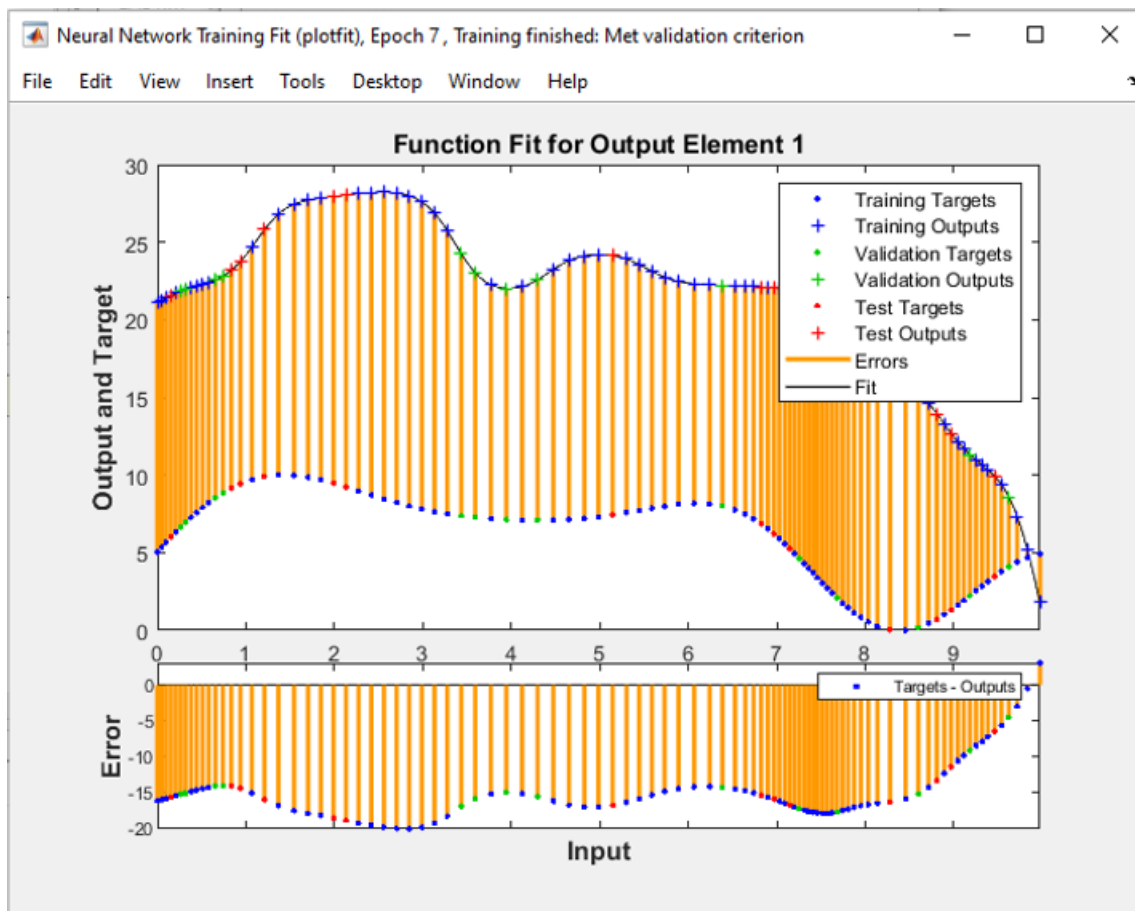
45 Wykres stanu szkolenia



46 Histogram błędów



47 Wykres regresji



48 Dopasowanie funkcji dla pojedynczego elementu

5. Analiza wyników

Lp.	Liczba neuronów w warstwie ukrytej	Epoka	Błąd średniokwadratowy	Gradient	Liczba potwierdzonych walidacji
1	3	6	10,3	4,51e-08	3
	10	152	1.05e-01	2,87e-05	6
2	3	16	18,103	1,36	6
	10	26	0,058096	0,347	6
3	3	458	0,021946	5.00e+10	0
	10	1000	1.64e-06	6.05e-07	0
4	3	1000	0,49463	0,0799	0
	10	7	2,131,263	3,12e+03	6

Legenda:

1. Propagacja wsteczna Levenberga-Marquardta
2. Propagacja wsteczna gradientu koniugatu skalowanego
3. Propagacja wsteczna regularyzacji Bayesowskiej
4. Zejście gradientowe (Gradient Descent)

Opis algorytmów względem liczby porządkowej:

Poniższa analiza została przeprowadzona tylko i wyłącznie dla wartości, które posiadały 10 neuronów w warstwie ukrytej. Niestety dla 3 neuronów analiza jest zbędna, algorytmy robią bardzo duże błędy, lecz zdarzają się również wyjątki od tej reguły. Przykładowo algorytm propagacji wstecznej Levenberga-Marquardta.

1. Proces został zakończony w 152 epoki, powodem przerwania operacji uczenia było osiągnięcie wyniku 6 potwierdzonych walidacji. Błąd średniokwadratowy uzyskał wartość 1,05e-01, gradient był równy 2,87e-05. Wynik błędu nie przekracza wartości 0,015. Co dało bardzo precyzyjne wyniki. Dla 3 neuronów w warstwie ukrytej pod koniec nauczania wartość dążyła do zera, co daje bardzo mały błąd. Same nauczanie trwało tylko 6 iteracji.
2. Działanie zostało zakończone w 26 epoki, z powodu osiągnięcia wartości 6 dla potwierdzonych walidacji. Błąd średniokwadratowy jest równy 0,058096 wartość gradientu osiągnęła 0,347. Na charakterystyce wartość błędu w większości nie przekracza 0,5, tylko na początkowym etapie wartość wyniosła około 0,7.

3. Proces został zakończony po osiągnięciu minimalnej granicy gradientu. Cała operacja została iterowana 1000. Błąd średniokwadratowy wyniósł wartość $1,64e-06$, natomiast gradient $6,05e-07$. Sam błąd jest bardzo korzystny. Algorytm spełnia swoje zastosowanie i choć nauka przebiega dłużej, można uzyskać lepszą precyzję.
4. Algorytm Gradient Descent osiągnął lepsze wyniki gdy w warstwie ukrytej były 3 neurony. Dla 10 neuronów algorytm generował dużo większe błędy niż w przypadku dla 3 neuronów. Dla 3 neuronów działanie zostało zakończone w epoce 1000, z powodu osiągnięcia maksymalnej liczby iteracji. Błąd średniokwadratowy jest równy 0,49463 wartość gradientu osiągnęła 0,0799. Na charakterystyce wartość błędu nie przekracza 2 w dalszej fazie wartość ta dąży do 0.

6. Wnioski

Po przetestowaniu 4 wybranych algorytmów dostępnych w środowisku „MatLab” sieci neuronowych stwierdzono, że część algorytmów spełnia swoje zadanie i można osiągnąć bardzo małe błędy czasami nie przekraczające wartości 0,015 dla 10 neuronów w warstwie ukrytej. Zdarzały się algorytmy z błędem przekraczającym 20% zadanej wartości. Jest to niedopuszczalne by zastosować te algorytmy przykładowo do nauczania obliczania wartości rezystancji, bądź jak w naszym przykładzie obliczeniu prędkości zawodnika. Takie błędy nie pozwoliły, by wybrać ewentualnego zwycięzcy.

Algorytmy z 3 neuronami w warstwie ukrytej całkowicie oblały test. Nie wykonały by ewentualnych obliczeń z wysoką dokładnością. Większość algorytmów popełniała błędy na poziomie 50% zadanej wartości. Takich wyników nie można brać pod uwagę. Wyjątkiem od tej reguły jest algorytm Gradient Descent. W jego przypadku błąd w wersji z 3 neuronami w warstwie ukrytej wynosi 0,49463.

Idealne dopasowanie ilości neuronów w warstwie ukrytej ma bardzo dużą wagę na poprawność działania algorytmu oraz pozwala maksymalnie zniwelować błąd podczas nauki.

Na podstawie analizowanych algorytmów najlepszy okazał się algorytm propagacji wstecznej Levenberga-Marquardta, który może pochwalić się wynikiem błędu na poziomie 0,01. Po nauczaniu sieci neuronowej z wykorzystaniem owego algorytmu można bez problemu obliczać prędkość poruszającego się zawodnika.