MONASH University
Engineering

Assessment cover sheet

Unit and student details					
Unit code	TRC 4901	**Unit title**	Artificial Intelligence		
If this is a group assignment, each student must include their name and ID number and sign the student statement.					
Student ID	29392004	**Surname**	Ranjan	**Given names**	Rivyesch
Student ID		**Surname**		**Given names**	
Student ID		**Surname**		**Given names**	
Student ID		**Surname**		**Given names**	

Assessment details			
Title of assignment /lab	Lab 3 - Object Characterisation and Classification using ANN	**Authorised group assignment**	Yes ☐ No ☑
Lecturer/tutor	Dr. Parasuraman	**Tutorial day and time**	Wednesday 8 am
Due date	05/05/2021	**Date submitted**	03/05/2021
Has any part of this assessment been previously submitted as part of another unit/course?			Yes ☐ No ☑

Student signature	Rivyesch Ranjan	**Date**	03/05/2021
Student signature		**Date**	
Student signature		**Date**	
Student signature		**Date**	

Contents

1.0 Introduction

A Back Propagation Neural Network (BPNN) is designed for object character recognition purposes to classify the iris type of different flowers. In this lab the BPNN is trained with the objective to recognize the type of iris flower based on the four attribute features provided in the training data. The network was trained using a combination of perfect quality inputs and inputs with random noise. Once the network is established, the effect of different training parameters such as number of layers, number of neurons in each layer and training algorithms on the performance are investigated and analysed.

2.0 Methodology

2.1 Dataset

The iris flower dataset is first pre-processed within the Microsoft Excel file itself. To work in Microsoft Excel the dataset file had to be converted from a DATA file to a CSV file where each row consists of four features of the flower and the corresponding type of flower. There are 150 such observations in the dataset file imported, with 50 observations or rows belonging to each type of flower. The dataset consists of data of only three types of iris flowers. The table below shows the data representation in Microsoft Excel where each of the firsts four column corresponds to a particular feature and the last column is the classification of the type of flower.

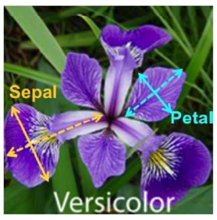
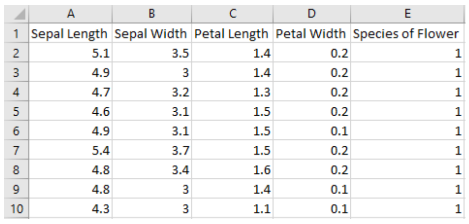


Figure 1: Iris Flower Example



	A	B	C	D	E
1	Sepal Length	Sepal Width	Petal Length	Petal Width	Species of Flower
2	5.1	3.5	1.4	0.2	1
3	4.9	3	1.4	0.2	1
4	4.7	3.2	1.3	0.2	1
5	4.6	3.1	1.5	0.2	1
6	4.9	3.1	1.5	0.1	1
7	5.4	3.7	1.5	0.2	1
8	4.8	3.4	1.6	0.2	1
9	4.8	3	1.4	0.1	1
10	4.3	3	1.1	0.1	1

Figure 2: First 10 rows of Irisdata Dataset

2.2 I/O preparation for training

With the data now imported, the last column is edited so that instead of displaying the names of the flower species it shows numbers that have been pre-allocated to represent each species.

Table 1: Numbers representing each Class of Iris Flowers

Species of Flower	Number Representation
Iris-sentosa	1
Iris-versicolor	2
Iris-virginica	3

Next noise is introduced to the dataset so that the network is trained properly and is more robust. This is done by copying and pasting 21 to 22 rows of data randomly chosen from each category of flower at the bottom of the existing dataset. This creates an additional 65 rows of observation. These newly included 65 rows are then reshuffled so that the order of the observations is completely random and no longer in order of the type of flower. Another way of creating noise is to modify the attributes of the data slightly. This is also done to the newly included 65 rows at random and to a slight extent.

Lastly, the dataset needs to be separated into training data and testing data to validate the accuracy of the trained neural network. A new sheet in the same Excel file is created and named test. Five rows from each category of flower of the first 150 rows are cut and pasted into the test sheet. The observations are taken from the data that has not been modified or altered during the creation of noise. In total there will be 15 observations in the test set while the training set contains a total of 200 observations. This file is then saved and imported into MATLAB so that the training followed by testing of the neural network can commence.

In MATLAB the training data and testing data are imported in cell format. This is converted into an array format using the cell2mat function. Both of these cells are also split up during the data type conversion so that the inputs are saved as one variable and the output is saved as another. This will create two matrices for each of the training and test data. Lastly, both matrices containing the inputs for the training and testing data are transposed so that the number of rows correspond to the number of selected features attributes and each column represents an observation. This is the required format of the inputs that the neural network expects.

3.0 Training Neural Network and Simulating Neural Network

The training process is done in the neural network GUI by initializing the NN toolbox with the command nntool. The input data and the target data are imported from the workspace and belong to the two matrices created from the training data.

The following parameters are varied one at a time while the others are kept fixed:

1. Number of Hidden Layers
2. Number of Neurons in Each Hidden Layer
3. Type of Training Algorithms
4. Transfer Function of Each Layer

Meanwhile the number of epochs is kept constant for every run at 10 epochs.

Firstly, the variation of number of neurons and number of layers is investigated to determine the optimal network architecture that yields the lowest MSE and highest regression. The results can be viewed in section 4.1. Due to time constraints and limited computational resources and memory, the maximum number of hidden layers used was capped at two layers while the maximum number of neurons in each layer was limited to 40 neurons. Using the network that is considered to be the optimal case, the architecture is kept constant while the training algorithm is varied. A total of six training algorithms was tested for the optimal architecture. For the sake of thoroughness, the type of transfer function used in each layer of the network is also varied and then investigated.

Once the network is created and trained it is simulated using the test data to determine the efficiency of training on the network and its overall accuracy and reliability. The results are rounded to obtain integer values before comparing the predicted output and the actual output.

4.0 Results

4.1 Variation of Number of Neurons and Number of Layers

Table 2: Variation of Neurons and Layers

1 hidden layer models						
Case	Layers	No. of Neurons	Iterations	Elapsed Time	MSE	Regression
1	1	10	20	0	4.98E-02	0.99072
	2	1				
2	1	20	11	0	1.18E-03	0.99382
	2	1				

3	1	30	14	0	2.05E-02	0.98055
	2	1				
4	1	40	13	0	3.50E-02	0.99027
	2	1				
2 hidden layer models						
Case	Layers	No. of Neurons	Iterations	Elapsed Time	MSE	Regression
5	1	10	24	0	8.06E-03	0.98344
	2	10				
	3	1				
6	1	20	27	1	3.01E-08	0.98918
	2	20				
	3	1				
7	1	30	33	3	8.59E-03	0.99724
	2	30				
	3	1				
8	1	40	15	5	3.40E-02	0.97801
	2	40				
	3	1				

4.2 Variation of Training Algorithm

Table 3: Variation of Training Algorithms for Case 6

Training Algorithm	Iterations	Elapsed Time	MSE	Regression
BFGS Quasi-Newton	27	1	1.69E-02	0.98371
Resilient Backpropagation	42	0	6.08E-02	0.97755
Fletcher-Powell Conjugate Gradient	36	0	3.80E-02	0.9784
Polak-Ribiére Conjugate Gradient	16	0	3.91E-02	0.97474
Variable Learning Rate Backpropagation	72	0	5.58E-02	0.97247
Levenberg-Marquardt	27	1	3.01E-08	0.98918

4.3 Variation of Transfer Function in Each Layer

Table 4: Variation of Transfer Function for Case 6

Case	Layer	Transfer Function	Iterations	Elapsed Time	MSE	Regression
9	1	PureLin	3	0	4.62E-02	0.96222
	2	PureLin				
	3	PureLin				
10	1	LogSig	24	0	3.33E-01	0.85817
	2	LogSig				
	3	LogSig				
11	1	TanSig	27	1	3.01E-08	0.98918
	2	TanSig				
	3	TanSig				

5.0 Discussion

The main criteria for judging the performance of each individual model created is the Mean Squared Error (MSE) and Regression value. MSE is a measure of how close a fitted line is to data points. The smaller the MSE, the closer the fit is to the data. Regression evaluates the scatter of the data points around the fitted regression line. A higher regression value is desired as it means there is smaller difference between the observed data and the fitted values.

In section 4.1, the study on number of neurons and number of layers provided two important findings. For this specific type of network and dataset, 20 neurons in each layer provided the best results for both one hidden layer models and two hidden layer models. The error MSE was significantly lower compared to when there were fewer or more neurons per layer. The second key finding is that the networks with two hidden layers generally performed better than those with just one hidden layer. The optimal network architecture from this study was found to be the network with two hidden layers and 20 neurons per layer. This corresponds to the network called Case 6.

With the ideal network architecture determined, the next study in section 4.2 aimed to find the best training algorithm that gives the highest training accuracy and lowest possible MSE. The previous study used the Levenberg-Marquardt algorithm which is the pre-set default. For this study an additional five algorithms were compared in addition to the previously trained Case 6 network utilising Levenberg-Marquardt algorithm. The results proved that the Levenberg-Marquardt training algorithm was the best, with an error significantly lower than any of the rest of the compared algorithms. It also had the highest regression showing a good fit and accurate mapping between the inputs and output.

Lastly, the transfer function of each layer is varied. For simplicity's sake and to avoid training too many networks, the transfer function of all the layers would be kept the same. So far, the best model found was Case 6 which uses Levenberg-Marquardt algorithm and for which each layer made use of the hyperbolic tangent sigmoid (TanSig) transfer function. In section 4.3, two new networks with the same architecture as that of Case 6 is trained, except the transfer functions of the first new network was set to all layers using linear (PureLin) transfer function while the second was set to all log-sigmoid (LogSig) transfer function. The two newly created models with different transfer functions performed much worse. The errors significantly increased indicating the PureLin and LogSig transfer function used was unable to correctly map the inputs to the output.

When each of these networks created were simulated with the test data, all but the network Case 1 gave perfect results once rounded. This can be viewed for every network trained in the appendix where the comparison graph between the simulated output and actual test output has been plotted. Whilst this indicates that all other models except Case 1 can produce accurate results once rounded, only network Case 6 is recommended for use in this object recognition and classification application. This is because its comparison plot of the predicted and actual outputs is near perfect even when the predicted outputs have not been rounded.

6.0 Conclusion

The neural network has been created and optimised to accurately recognise and classify the flowers based on data of the four attributes previously mentioned. The introduction of noise to the dataset improved the model's robustness and ability to generalise and then classify. The optimisation process which involved the variation of various parameters highlighted the influence of each parameter and the importance of the optimisation process in achieving an accurate model. Overall, after the various studies conducted the best network for this particular application is a two hidden layer neural network with 20 neurons in each hidden layer using Levenberg-Marquardt training algorithm and hyperbolic tangent sigmoid transfer function in all of its layers.

7.0 References

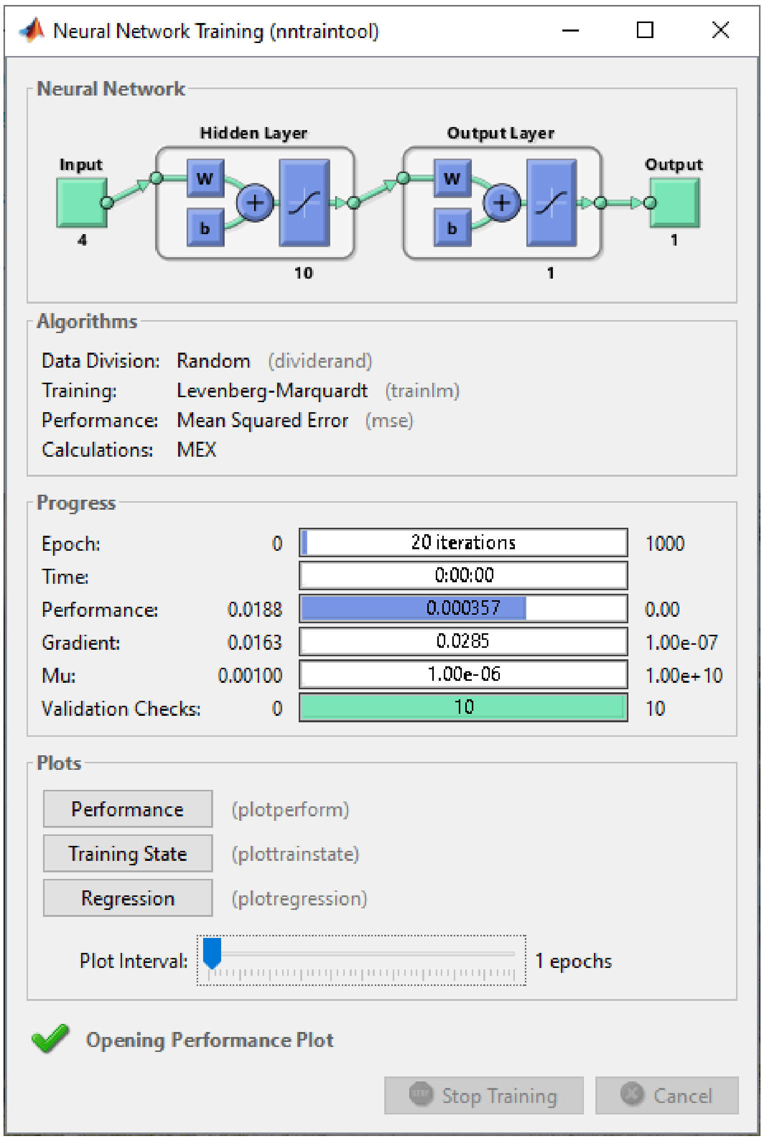
[1] "What are Mean Squared Error and Root Mean Squared Error? - Technical Information Library", *Technical Information Library*, 2021. [Online]. Available: https://www.vernier.com/til/1014. [Accessed: 03- May- 2021].

[2] 2021. [Online]. Available: https://statisticsbyjim.com/regression/interpret-r-squared-regression/. [Accessed: 03- May- 2021].

8.0 Appendix

8.1 Variation of Number of Neurons and Number of Layers

8.1.1 Case 1



Algorithms

Data Division: Random (dividerand)
Training: Levenberg-Marquardt (trainlm)
Performance: Mean Squared Error (mse)
Calculations: MEX

Progress

Epoch:	0	20 iterations	1000
Time:		0:00:00	
Performance:	0.0188	0.000357	0.00
Gradient:	0.0163	0.0285	1.00e-07
Mu:	0.00100	1.00e-06	1.00e+10
Validation Checks:	0	10	10

(plotperform)
(plottrainstate)
(plotregression)
Plot Interval: 1 epochs

Opening Performance Plot

Figure 3: Case 1 Training Network

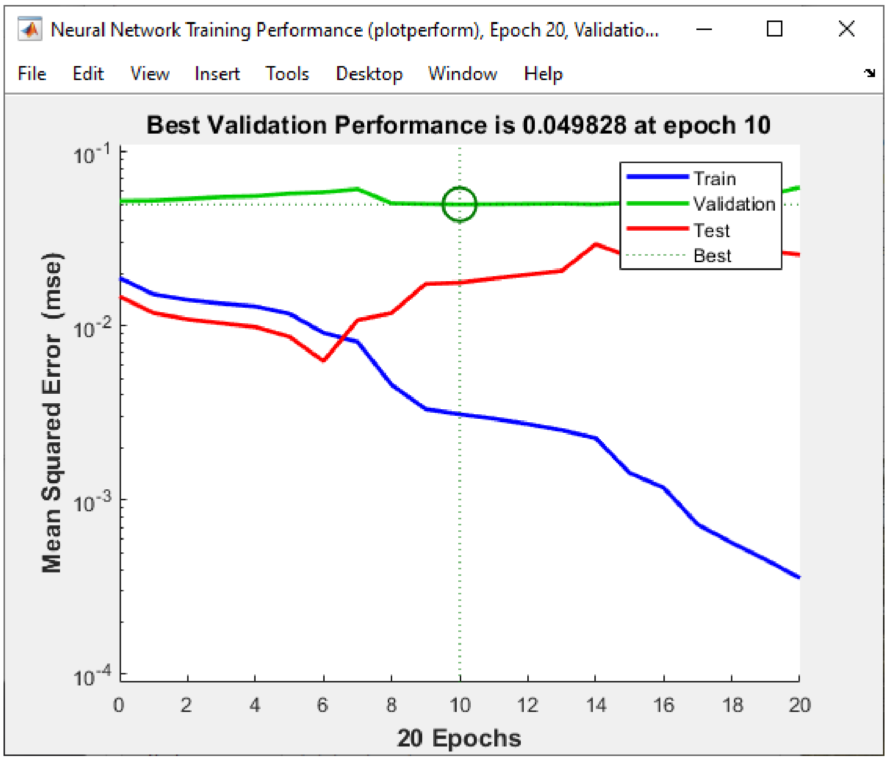


Figure 4: Case 1 Training Performance

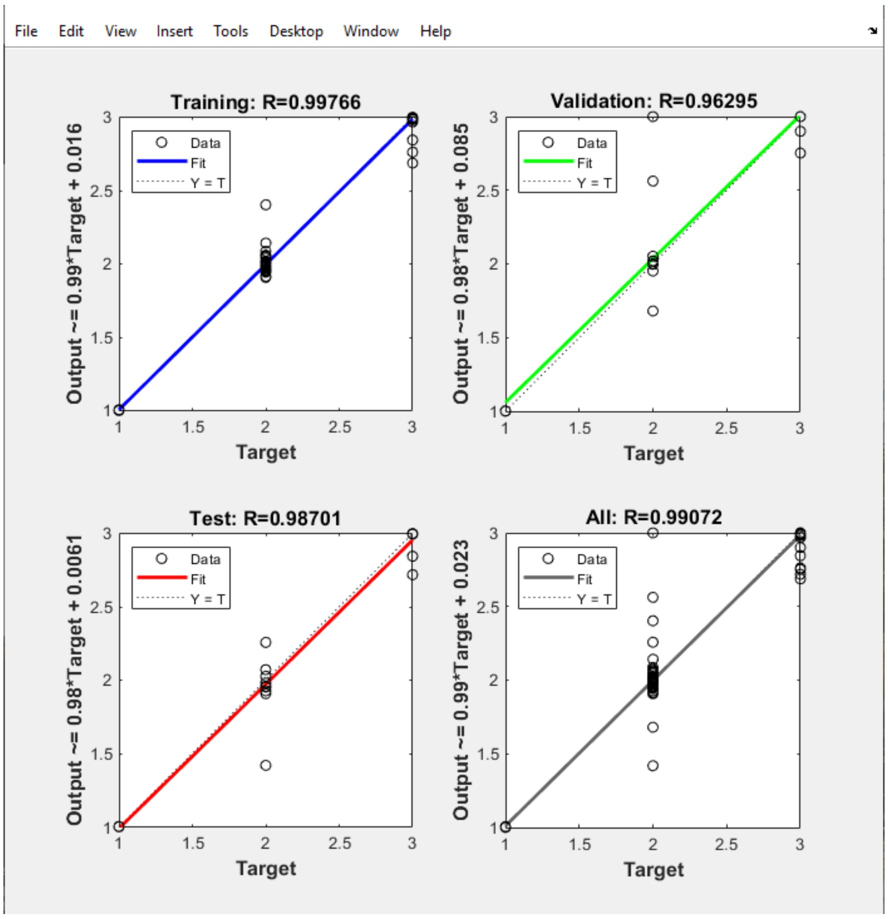


Figure 5: Case 1 Regression

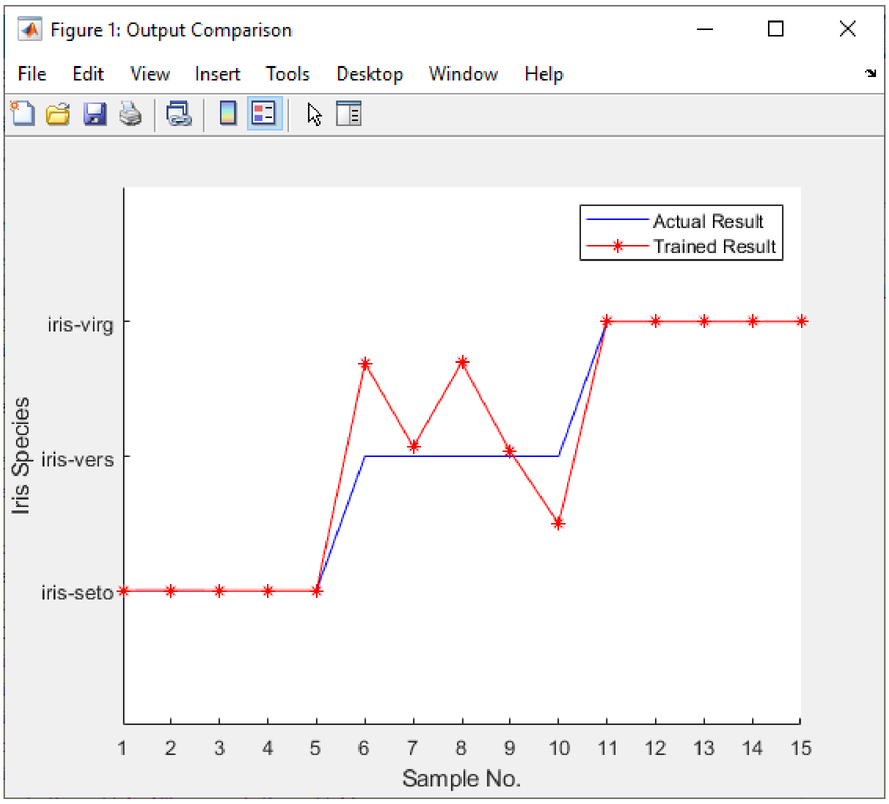


Figure 6: Case 1 Unrounded Simulated Output

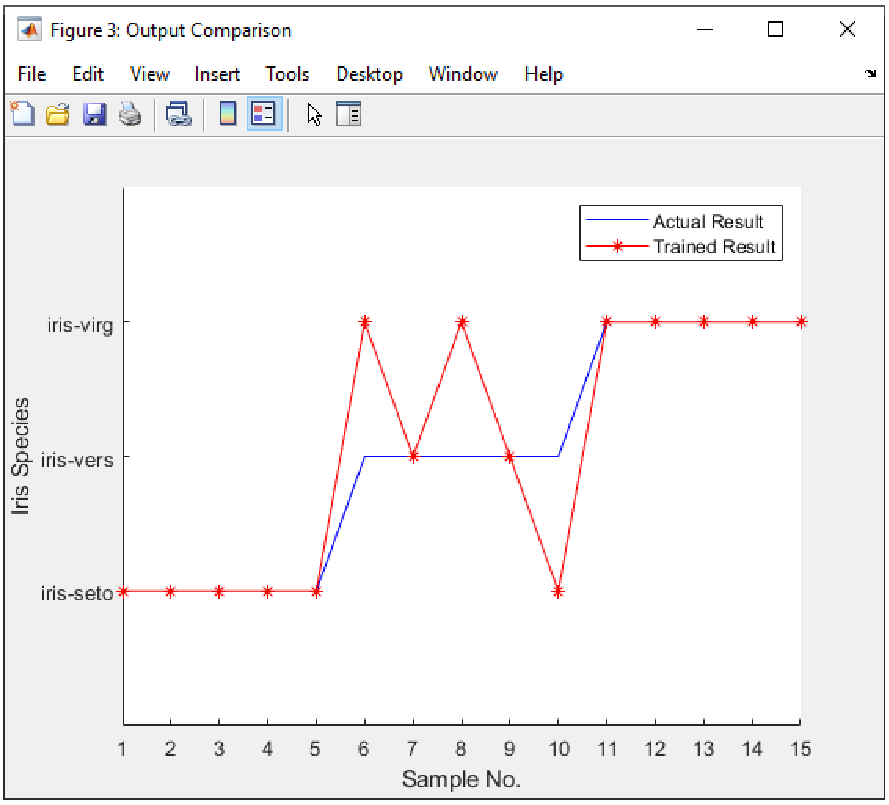
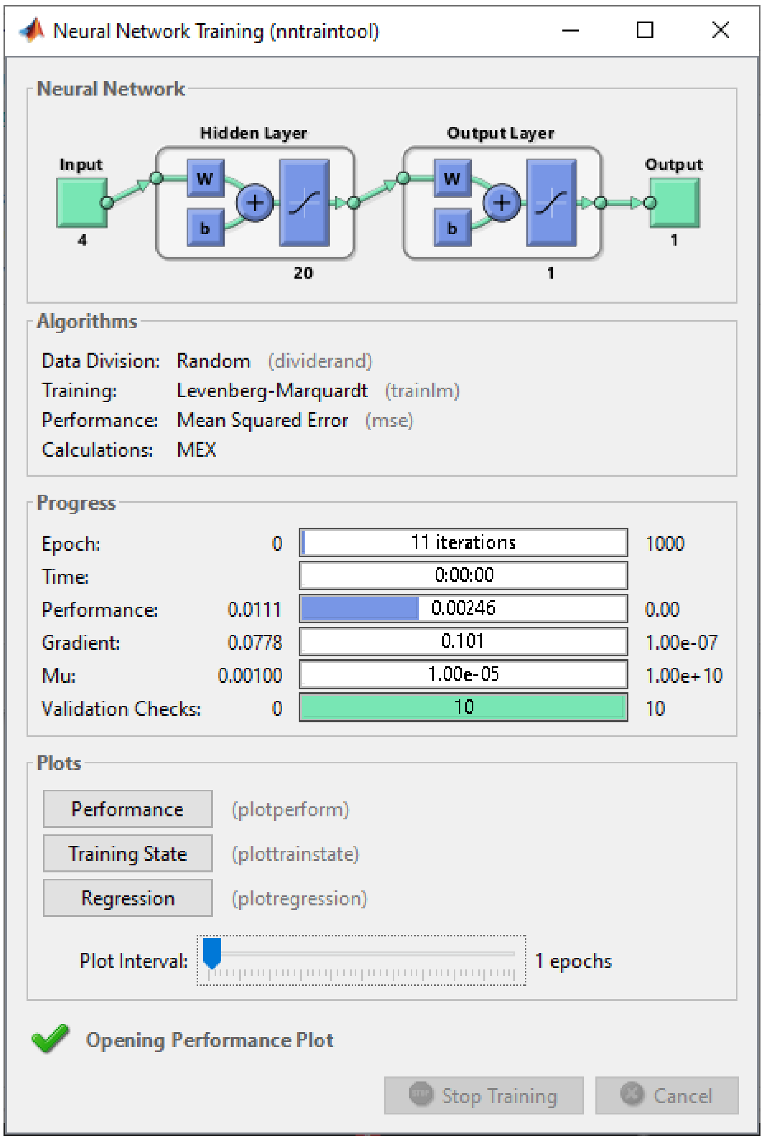


Figure 7: Case 1 Rounded Simulated Output

8.1.2 Case 2



Algorithms

Data Division:	Random	(dividerand)
Training:	Levenberg-Marquardt	(trainlm)
Performance:	Mean Squared Error	(mse)
Calculations:	MEX	

Progress

Epoch:	0	11 iterations	1000
Time:		0:00:00	
Performance:	0.0111	0.00246	0.00
Gradient:	0.0778	0.101	1.00e-07
Mu:	0.00100	1.00e-05	1.00e+10
Validation Checks:	0	10	10

Plots

Performance (plotperform)
Training State (plottrainstate)
Regression (plotregression)

Plot Interval: 1 epochs

✓ Opening Performance Plot

Figure 8: Case 2 Training Network

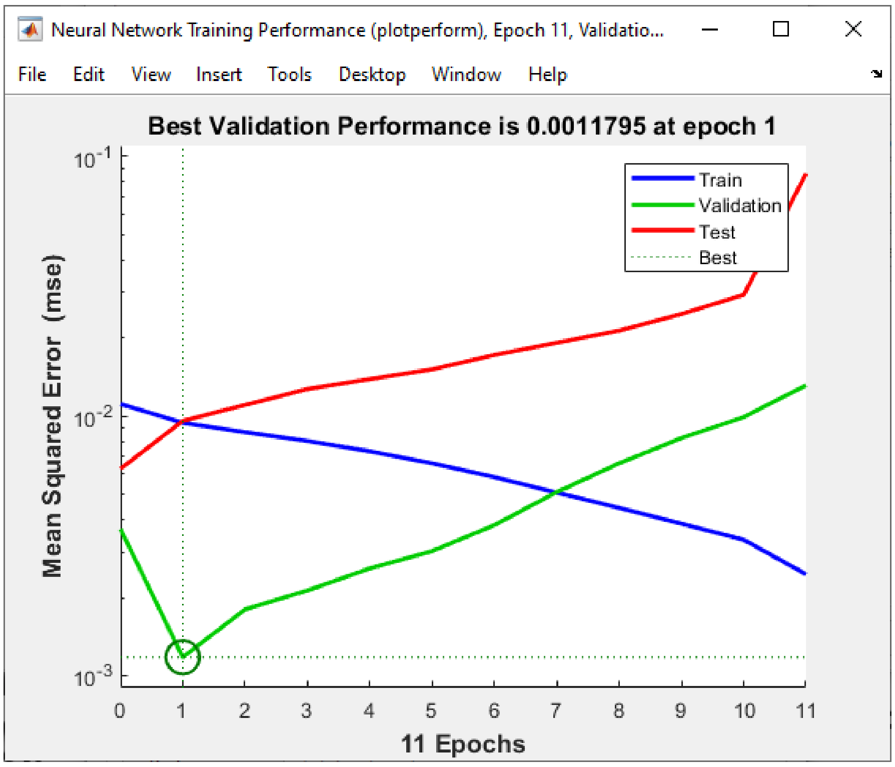


Figure 9: Case 2 Training Performance

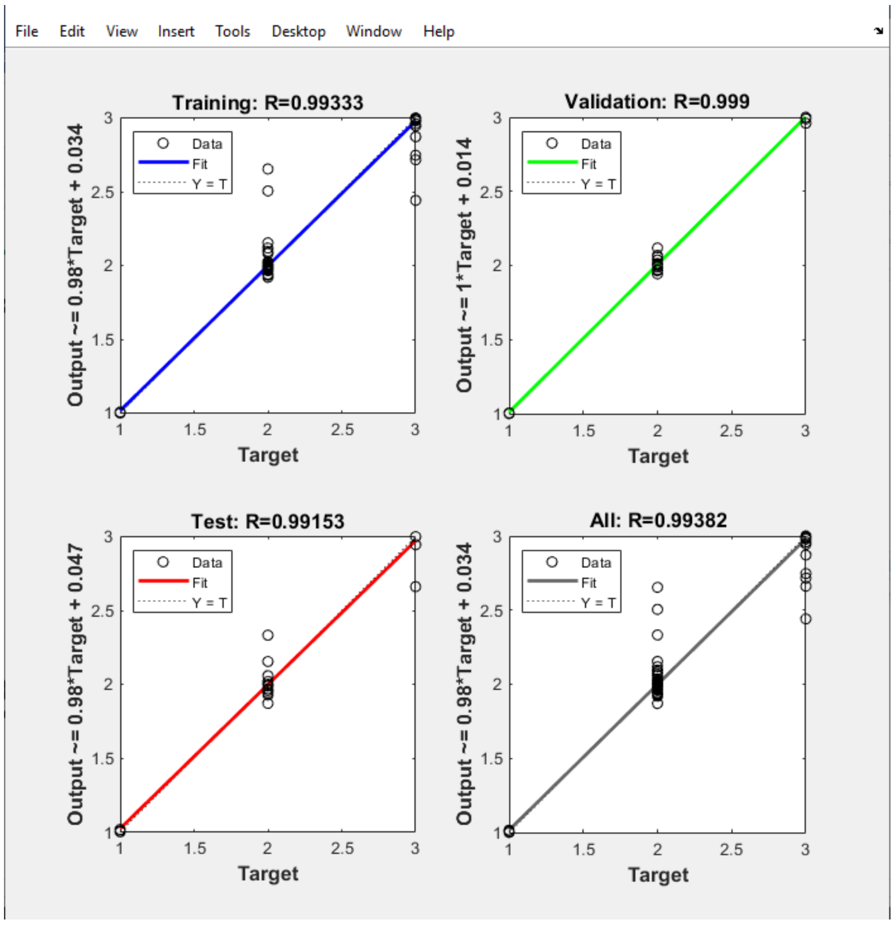


Figure 10: Case 2 Regression

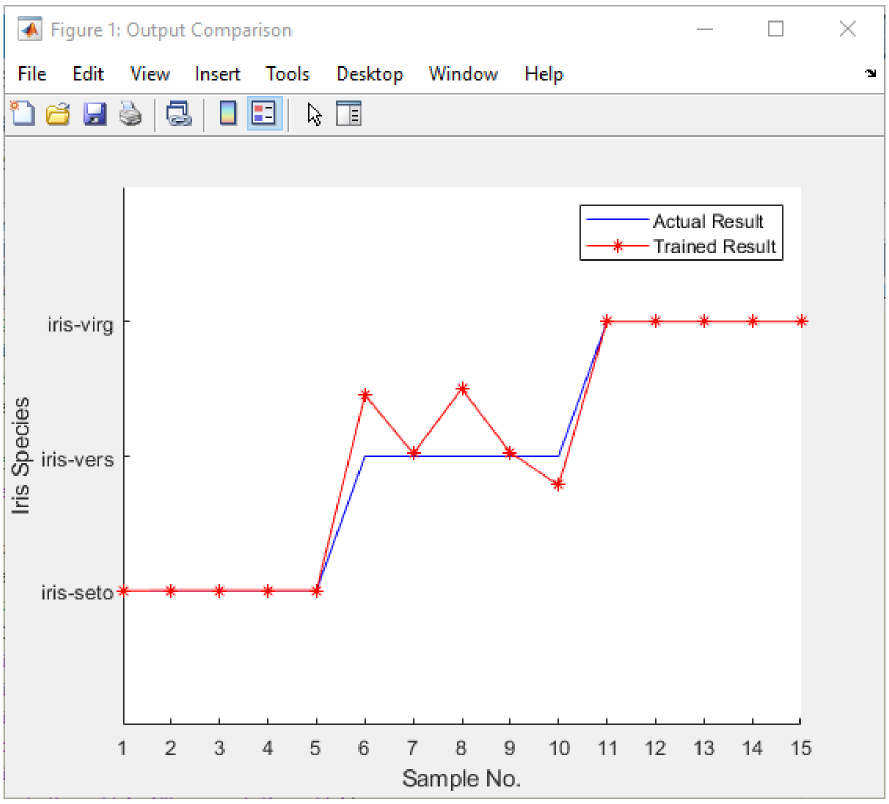


Figure 11: Case 2 Unrounded Simulated Output

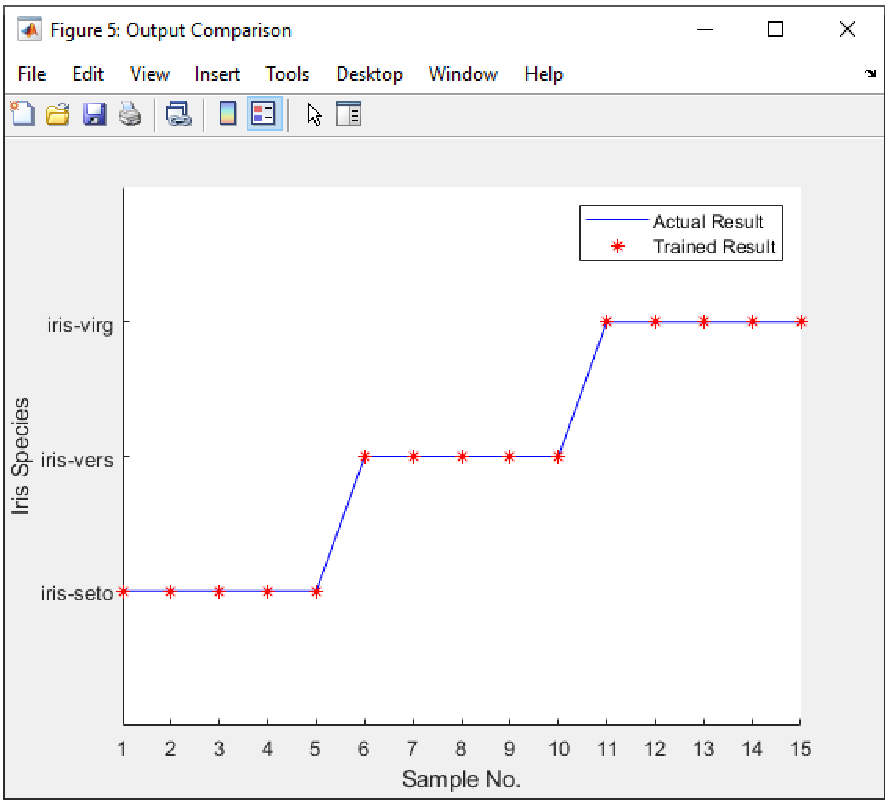
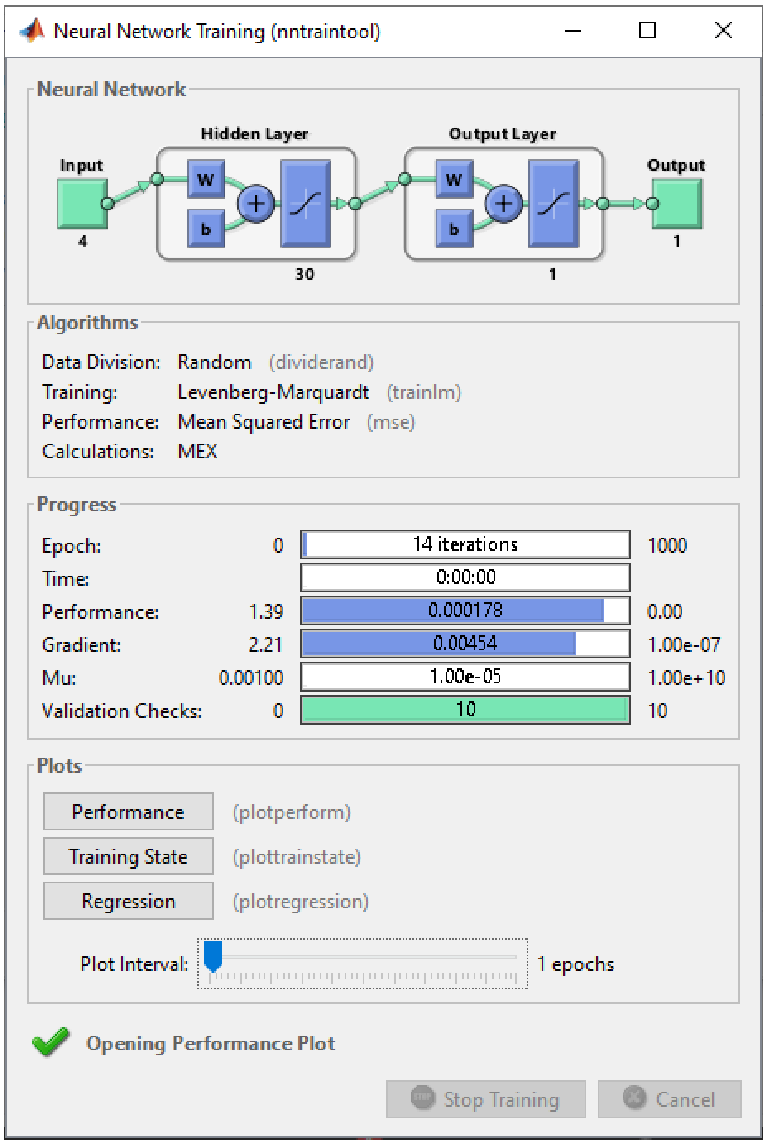


Figure 12: Case 2 Rounded Simulated Output

8.1.3 Case 3



Algorithms

Data Division: Random (dividerand)
Training: Levenberg-Marquardt (trainlm)
Performance: Mean Squared Error (mse)
Calculations: MEX

Progress

```
Epoch:                 0   14 iterations   1000
Time:                      0:00:00
Performance:        1.39   0.000178        0.00
Gradient:           2.21   0.00454         1.00e-07
Mu:              0.00100   1.00e-05        1.00e+10
Validation Checks:     0   10              10
```

Plots

Performance (plotperform)
Training State (plottrainstate)
Regression (plotregression)

Plot Interval: 1 epochs

Opening Performance Plot

Figure 13: Case 3 Training Network

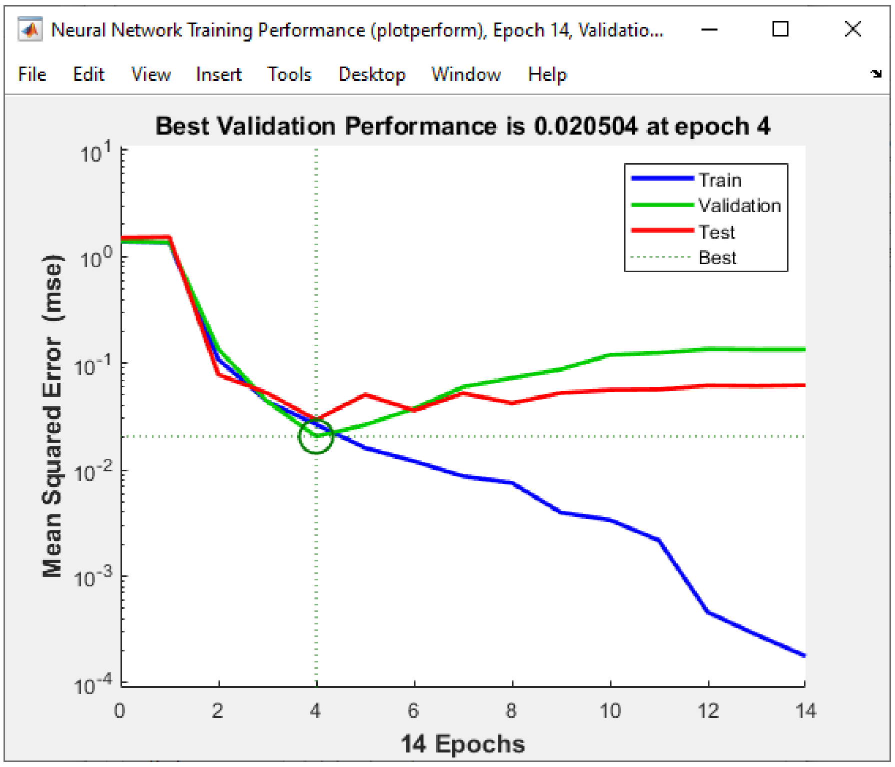


Figure 14: Case 3 Training Performance

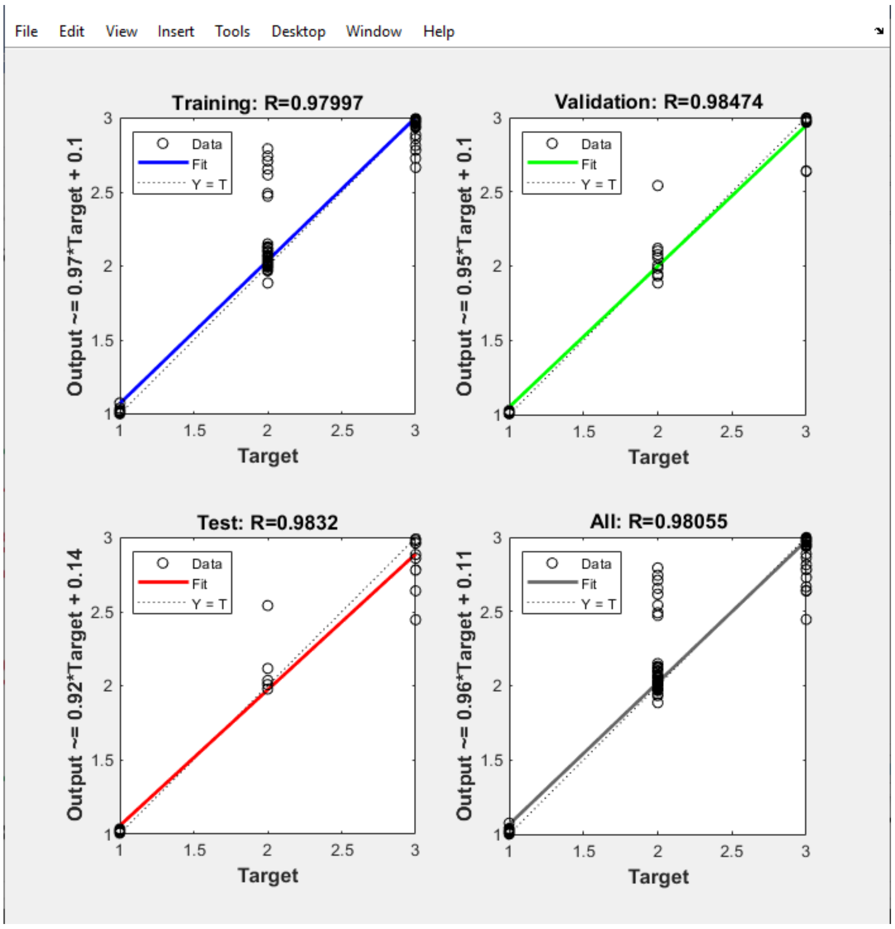


Figure 15: Case 3 Regression

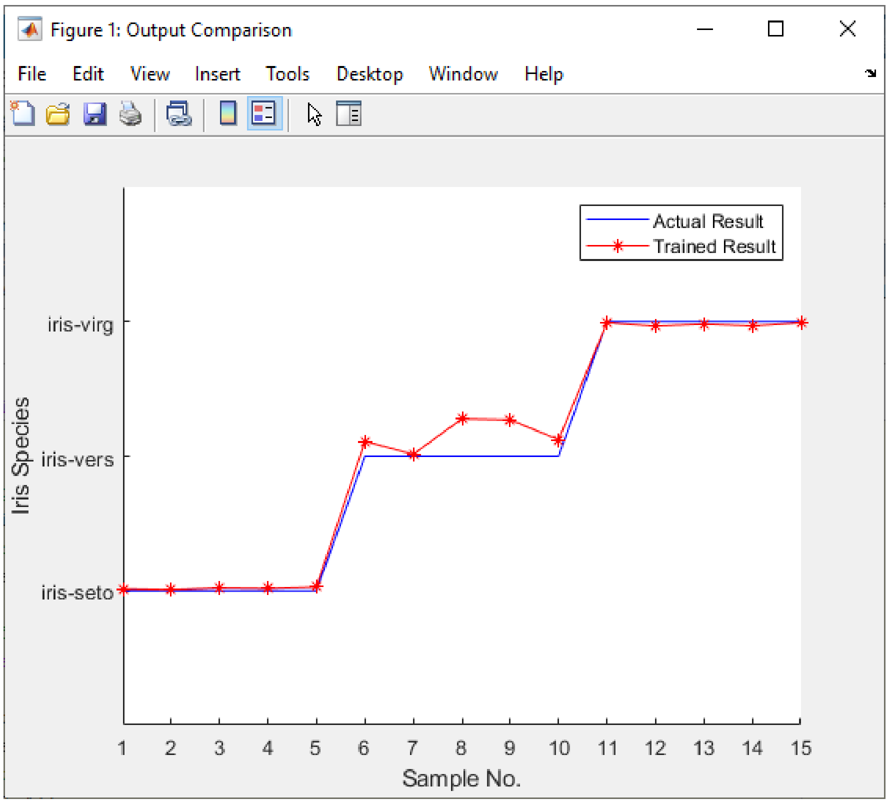


Figure 16: Case 3 Unrounded Simulated Output

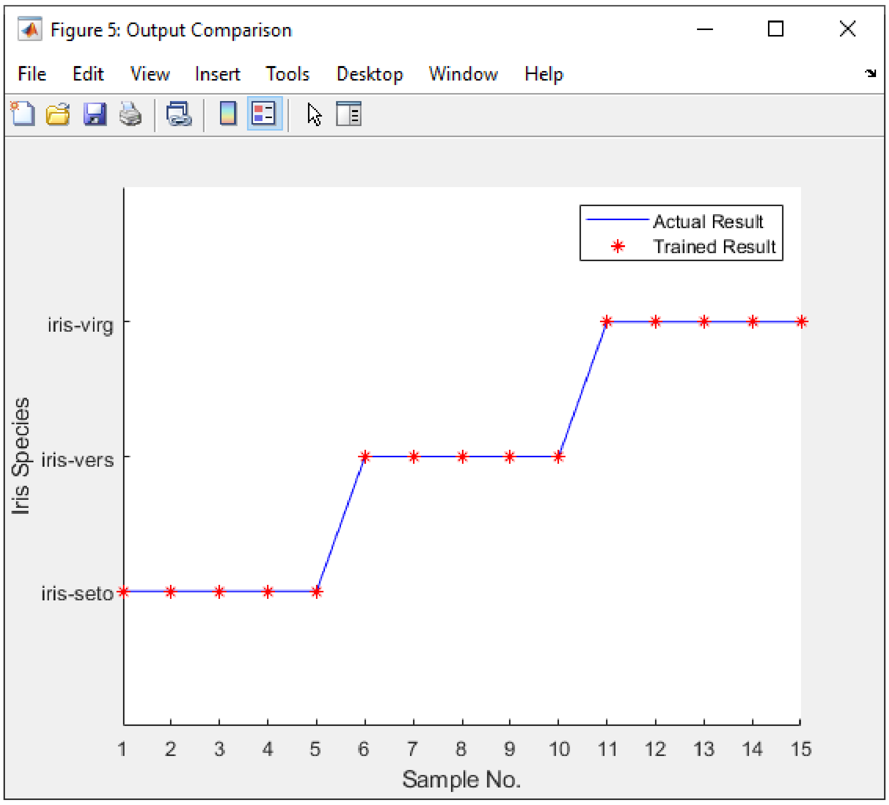
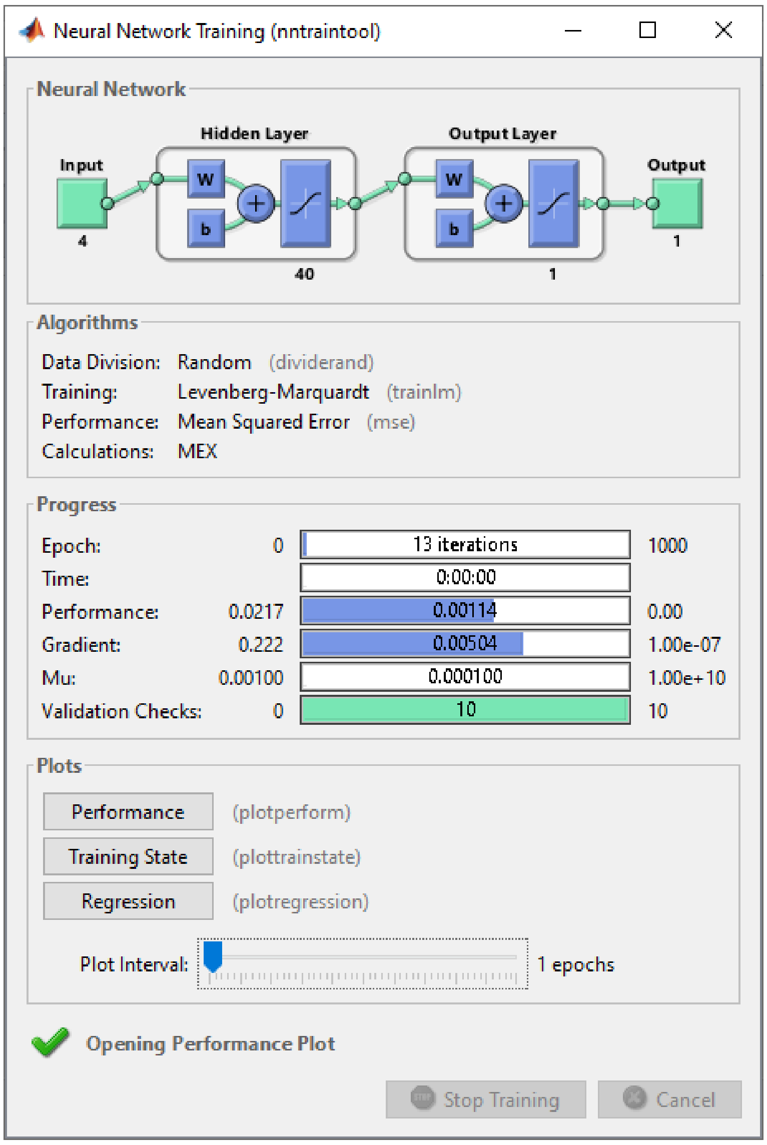


Figure 17: Case 3 Rounded Simulated Output

8.1.4 Case 4


Neural Network Training (nntraintool)

Neural Network

Hidden Layer
Output Layer
Input
Output
W
b
W
b
4
40
1
1

Algorithms

Data Division:	Random	(dividerand)
Training:	Levenberg-Marquardt	(trainlm)
Performance:	Mean Squared Error	(mse)
Calculations:	MEX	

Progress

Epoch:	0	13 iterations	1000
Time:		0:00:00	
Performance:	0.0217	0.00114	0.00
Gradient:	0.222	0.00504	1.00e-07
Mu:	0.00100	0.000100	1.00e+10
Validation Checks:	0	10	10

Plots

Performance (plotperform)
Training State (plottrainstate)
Regression (plotregression)

Plot Interval: 1 epochs

Opening Performance Plot

Figure 18: Case 4 Training Network

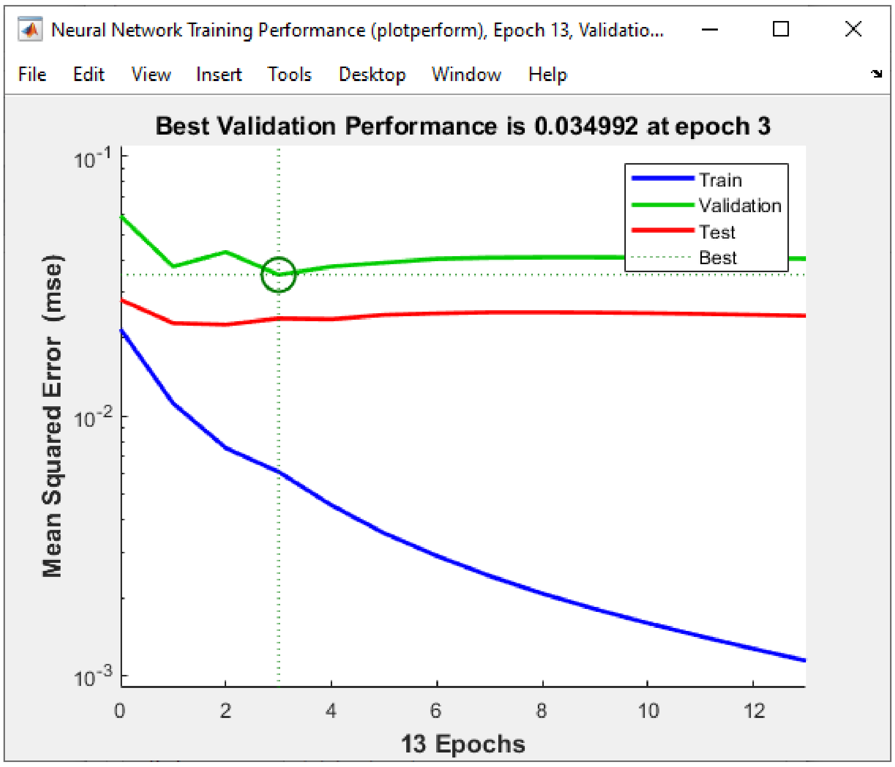


Figure 19: Case 4 Training Performance

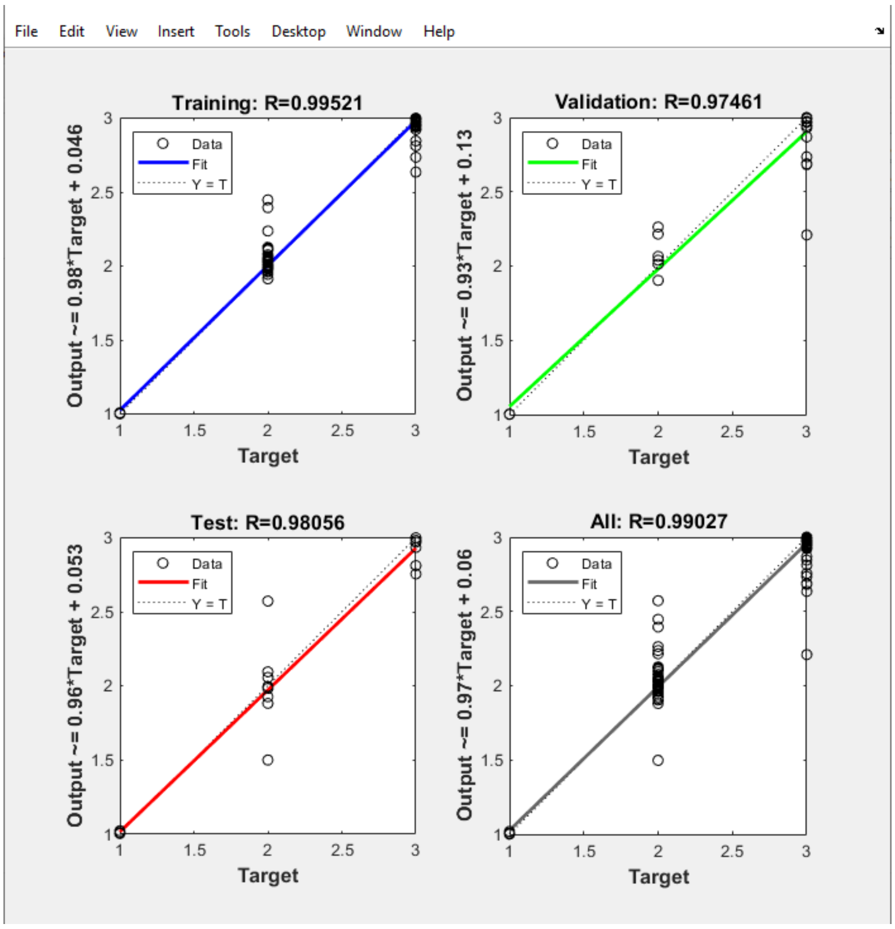


Figure 20: Case 4 Regression

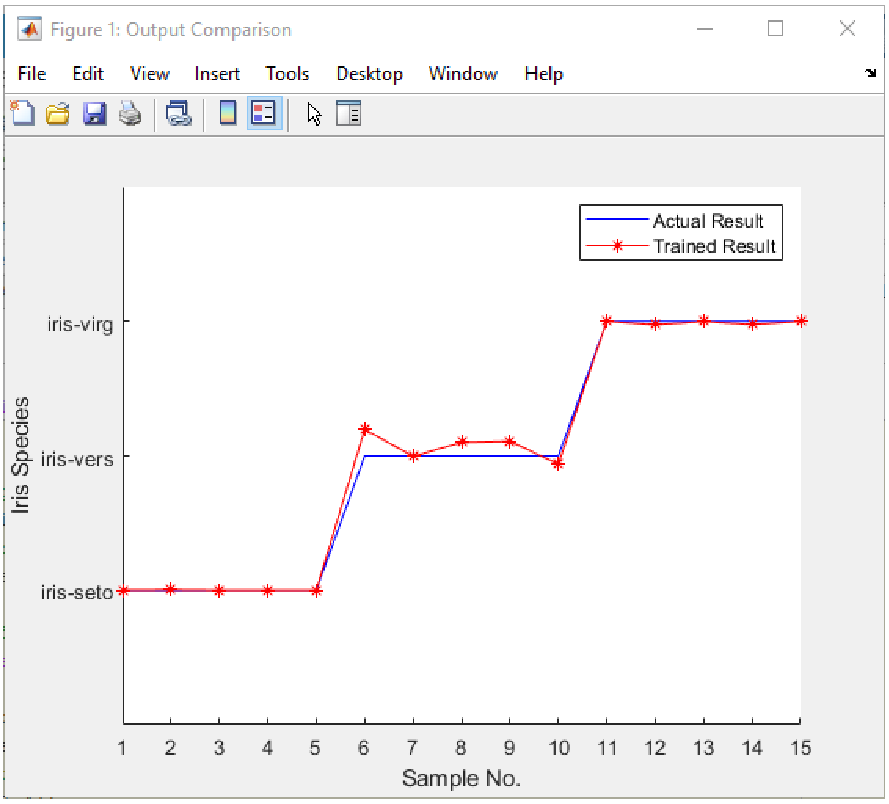


Figure 21: Case 4 Unrounded Simulated Output

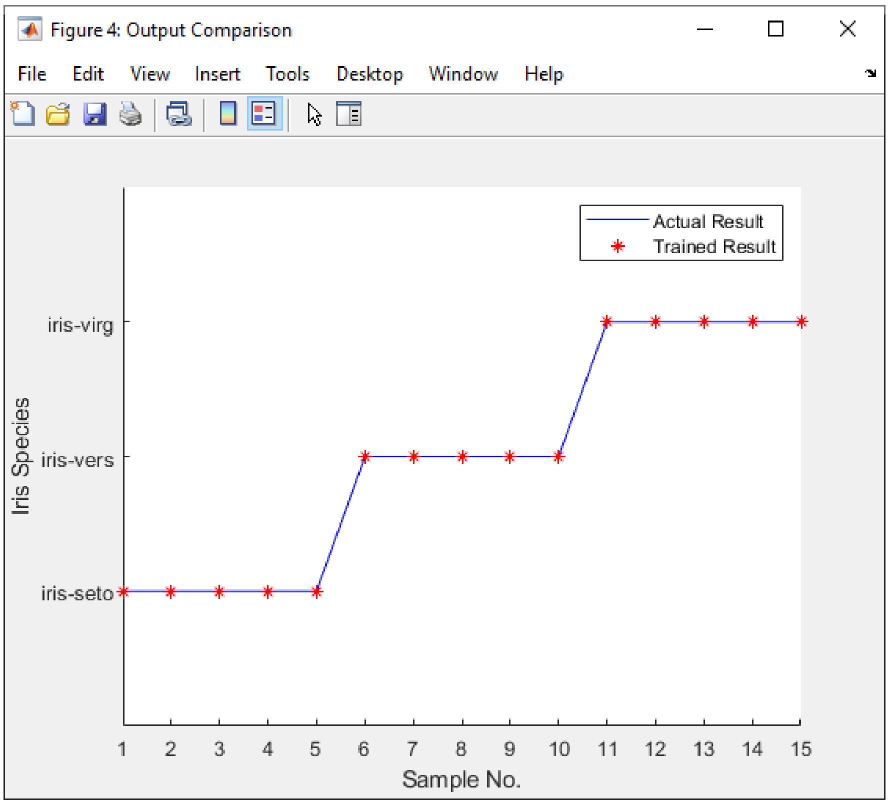
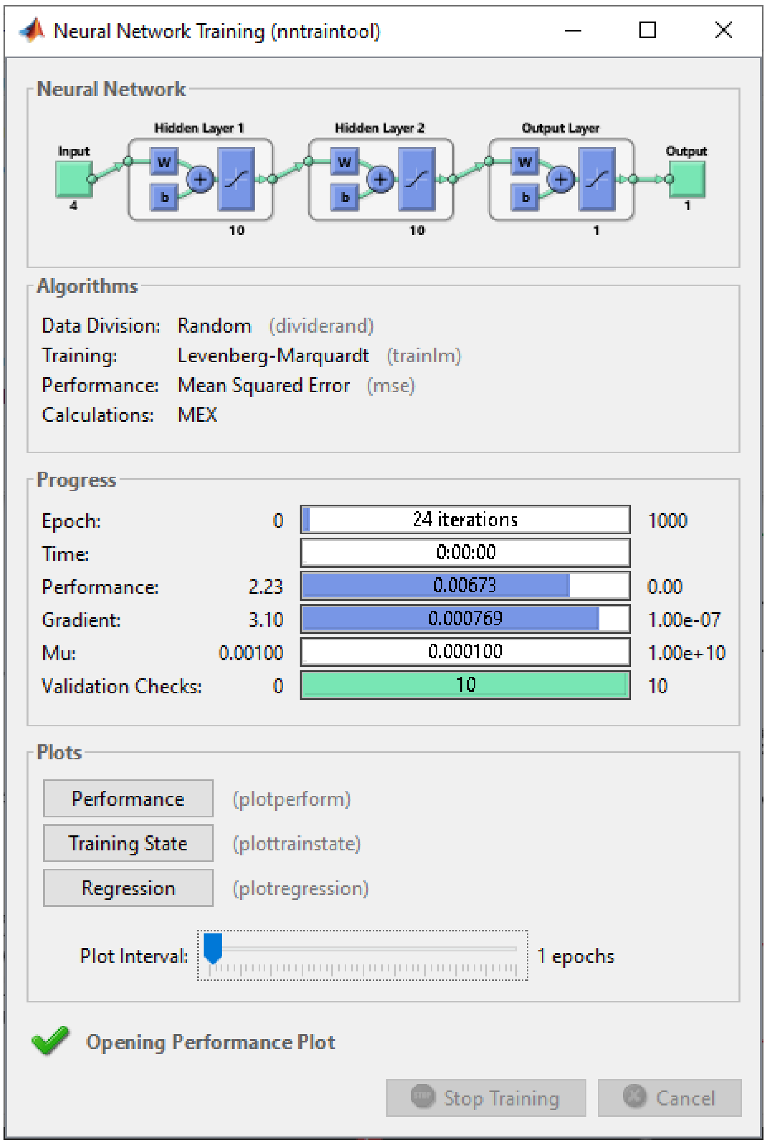


Figure 22: Case 4 Rounded Simulated Output

8.1.5 Case 5



Algorithms

Data Division: Random (dividerand)
Training: Levenberg-Marquardt (trainlm)
Performance: Mean Squared Error (mse)
Calculations: MEX

Progress

Epoch:	0	24 iterations	1000
Time:		0:00:00	
Performance:	2.23	0.00673	0.00
Gradient:	3.10	0.000769	1.00e-07
Mu:	0.00100	0.000100	1.00e+10
Validation Checks:	0	10	10

Opening Performance Plot

Figure 23: Case 5 Training Network

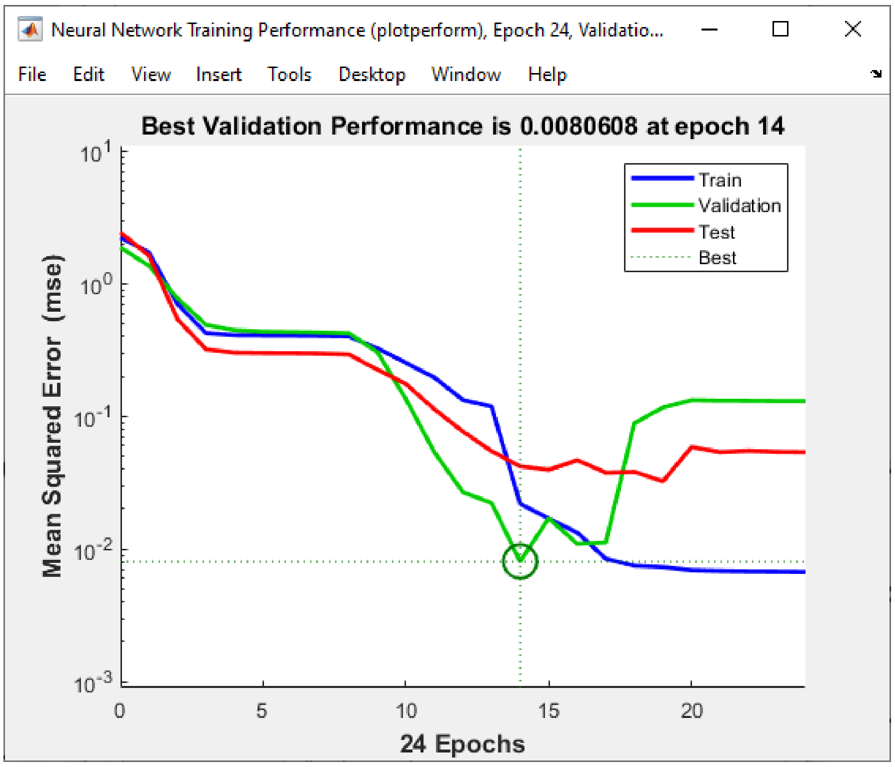


Figure 24: Case 5 Training Performance

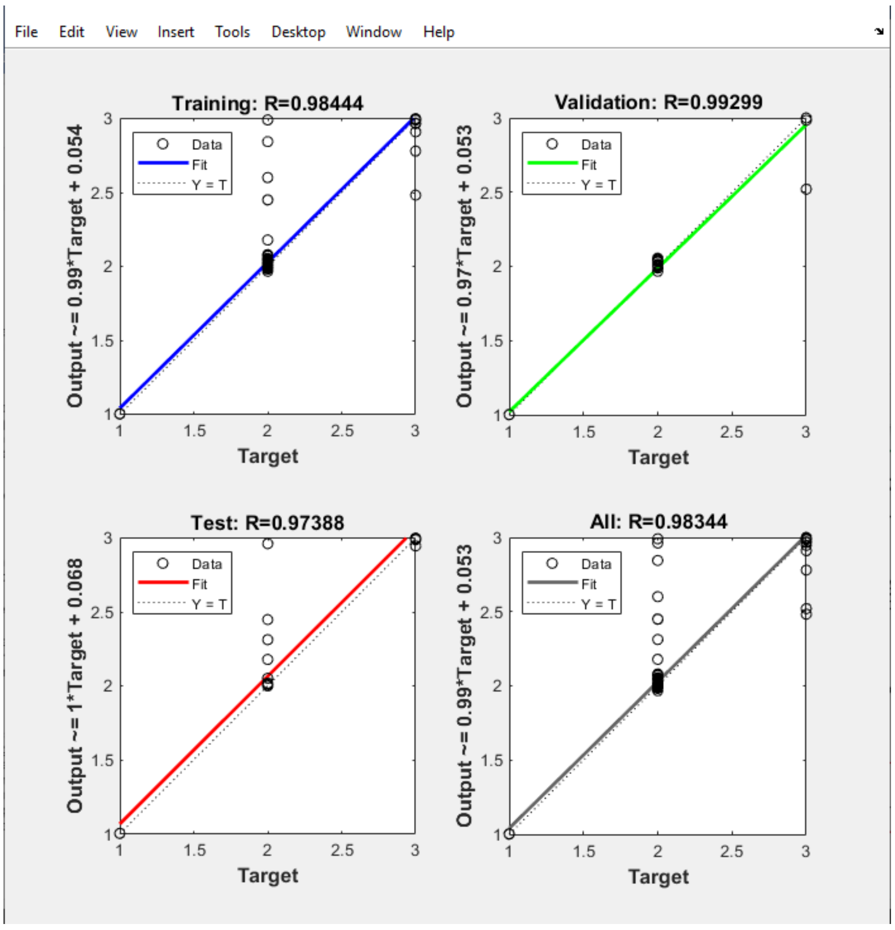


Figure 25: Case 5 Regression

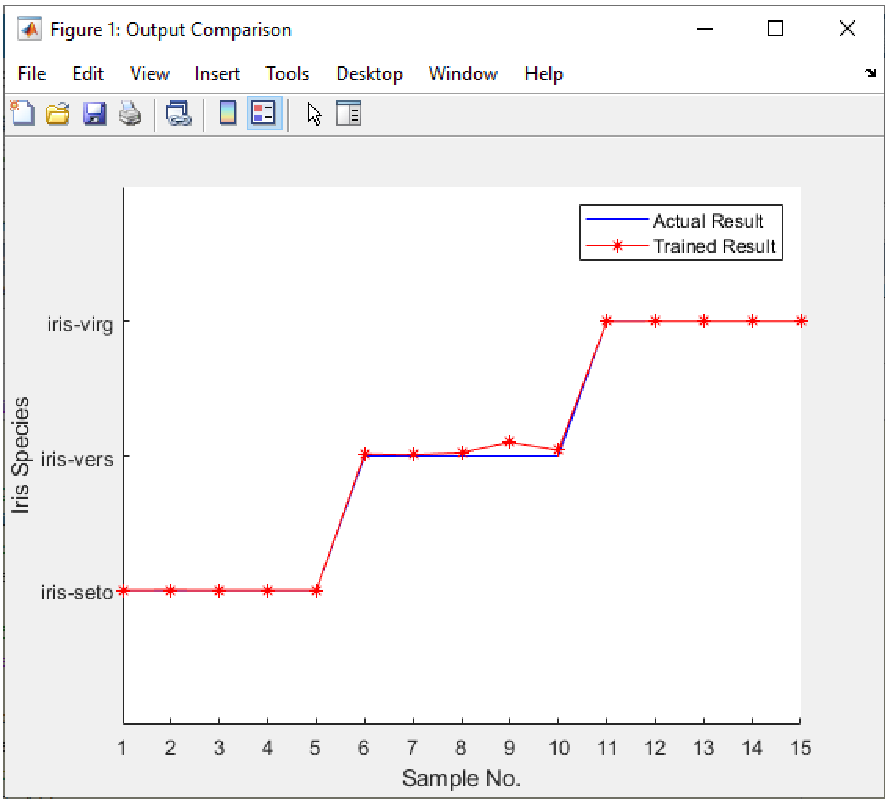


Figure 26: Case 5 Unrounded Simulated Output

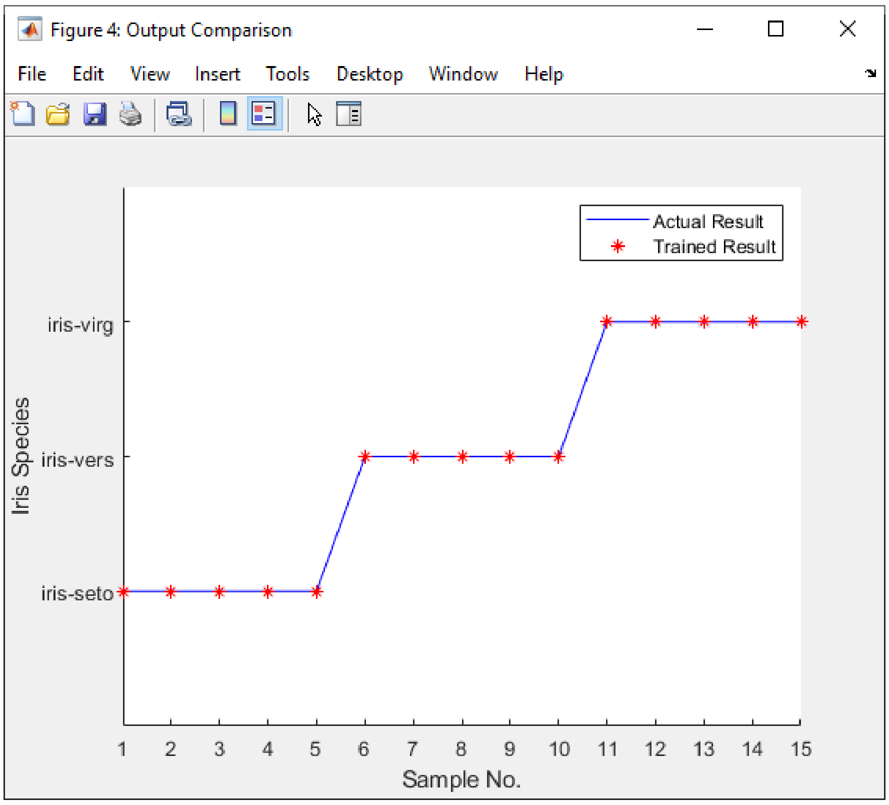
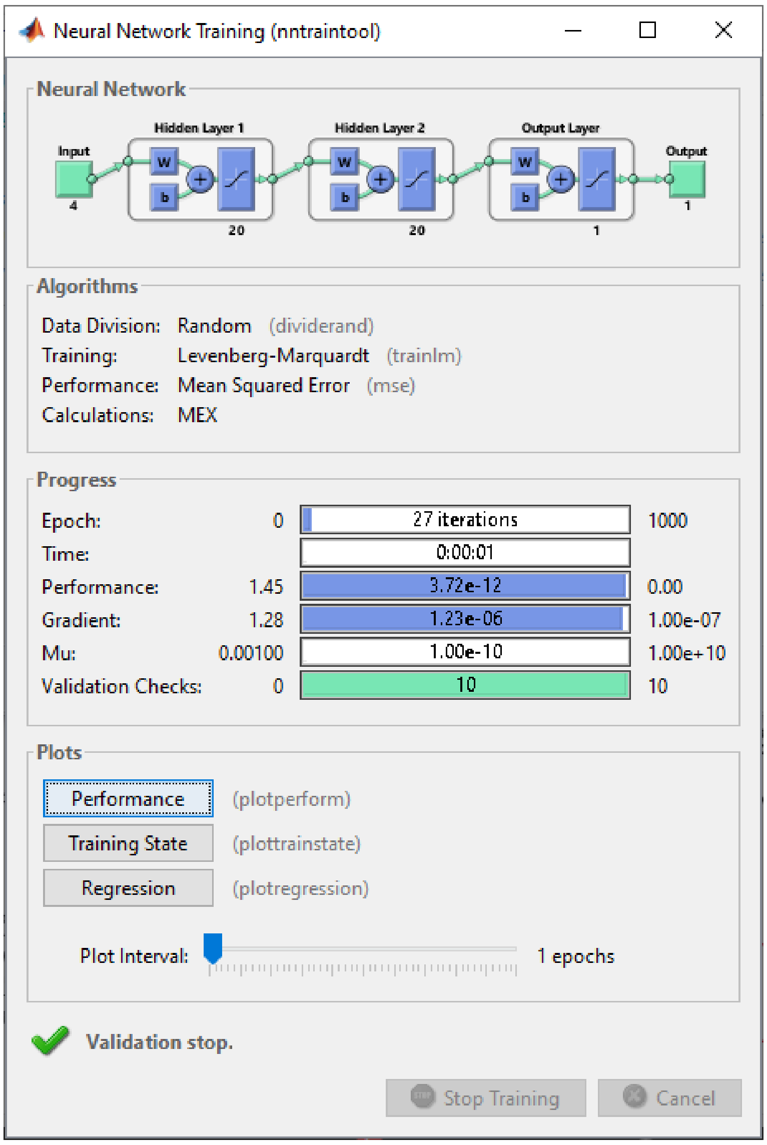


Figure 27: Case 5 Rounded Simulated Output

8.1.6 Case 6



Algorithms

Data Division:	Random	(dividerand)
Training:	Levenberg-Marquardt	(trainlm)
Performance:	Mean Squared Error	(mse)
Calculations:	MEX	

Progress

Epoch:	0	27 iterations	1000
Time:		0:00:01	
Performance:	1.45	3.72e-12	0.00
Gradient:	1.28	1.23e-06	1.00e-07
Mu:	0.00100	1.00e-10	1.00e+10
Validation Checks:	0	10	10

Plots

Performance (plotperform)
Training State (plottrainstate)
Regression (plotregression)

Plot Interval: 1 epochs

Validation stop.

Figure 28: Case 6 Training Network

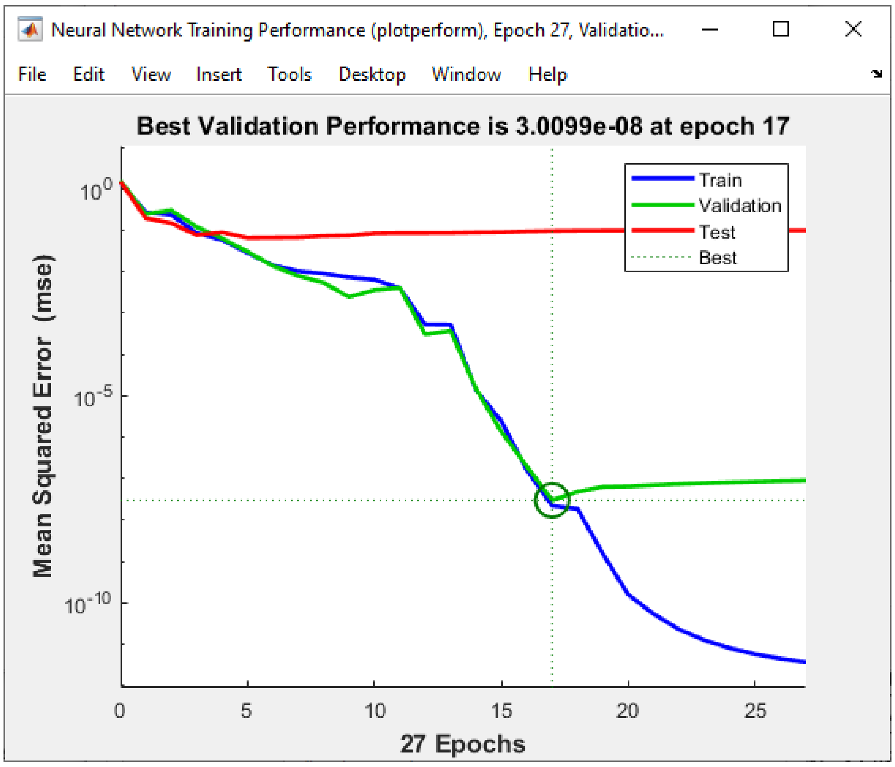


Figure 29: Case 6 Training Performance

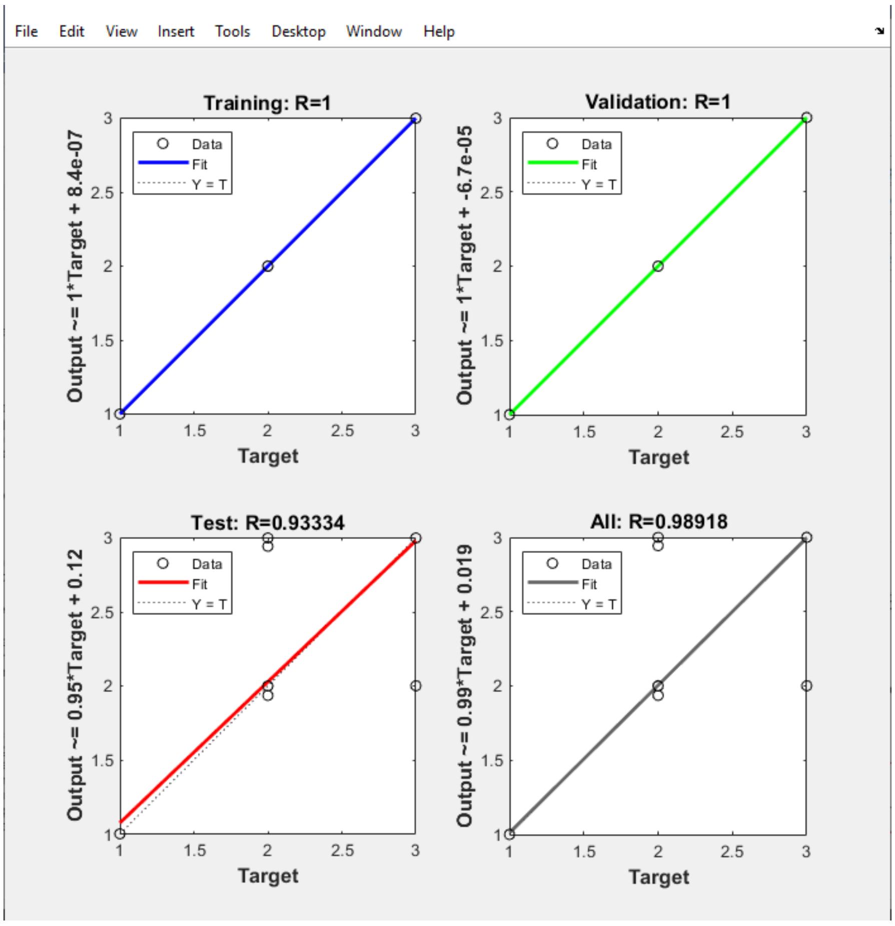


Figure 30: Case 6 Regression

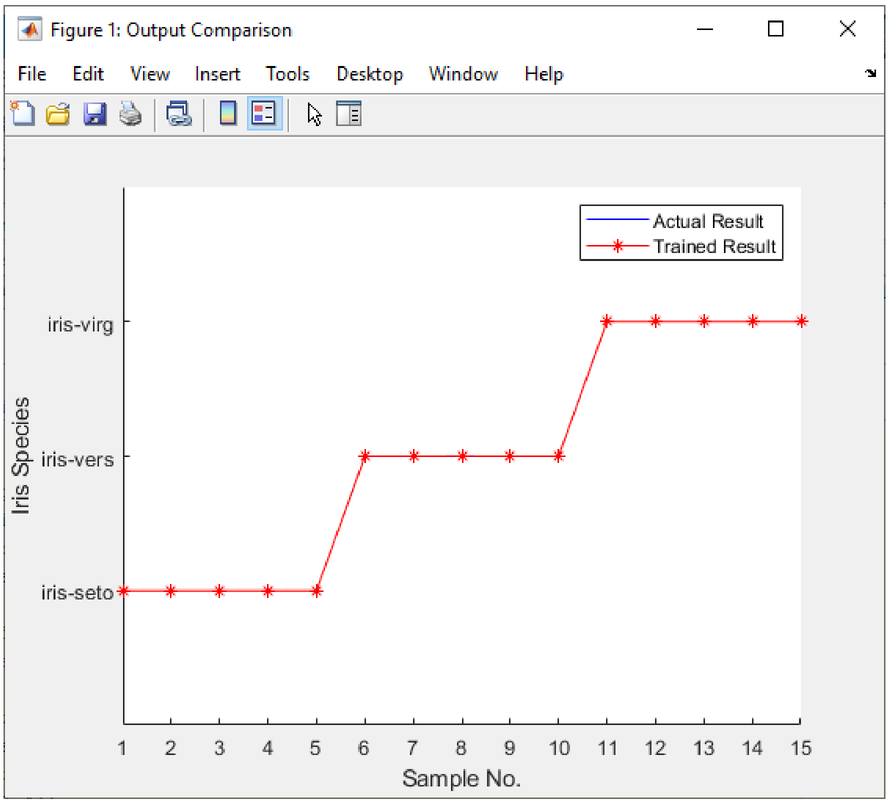


Figure 31: Case 6 Unrounded Simulated Output

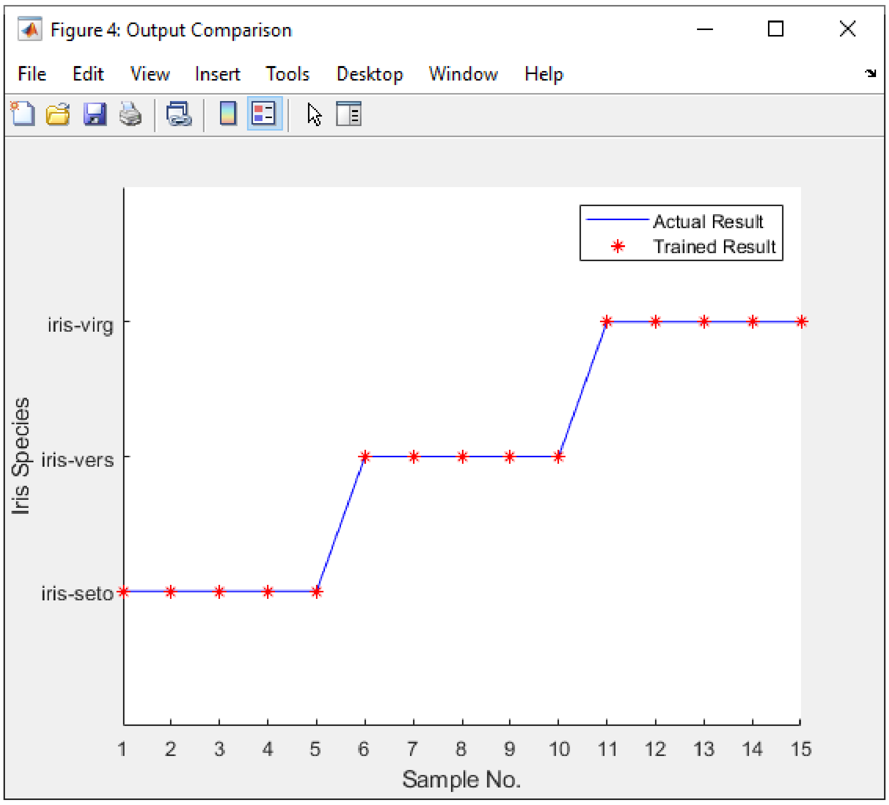
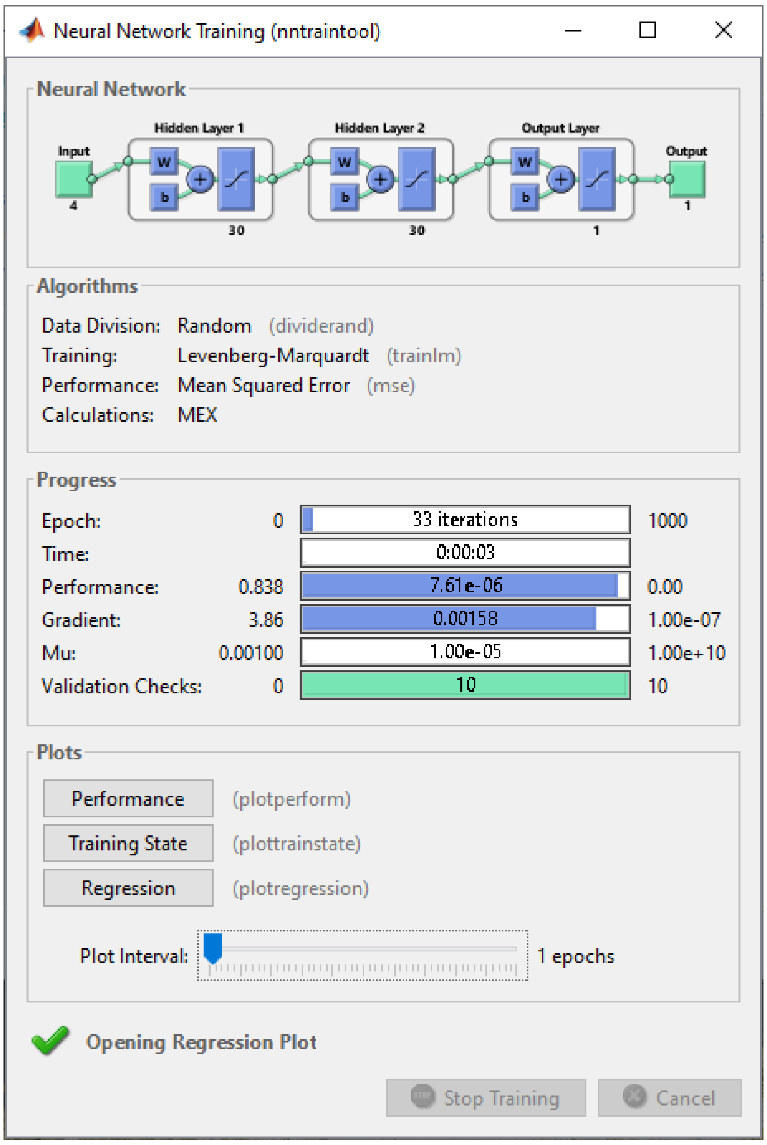


Figure 32: Case 6 Rounded Simulated Output

8.1.7 Case 7



Algorithms

Data Division: Random (dividerand)
Training: Levenberg-Marquardt (trainlm)
Performance: Mean Squared Error (mse)
Calculations: MEX

Progress

Epoch:	0	33 iterations	1000
Time:		0:00:03	
Performance:	0.838	7.61e-06	0.00
Gradient:	3.86	0.00158	1.00e-07
Mu:	0.00100	1.00e-05	1.00e+10
Validation Checks:	0	10	10

Performance (plotperform)
Training State (plottrainstate)
Regression (plotregression)

Plot Interval: 1 epochs

✓ Opening Regression Plot

Figure 33: Case 7 Training Network

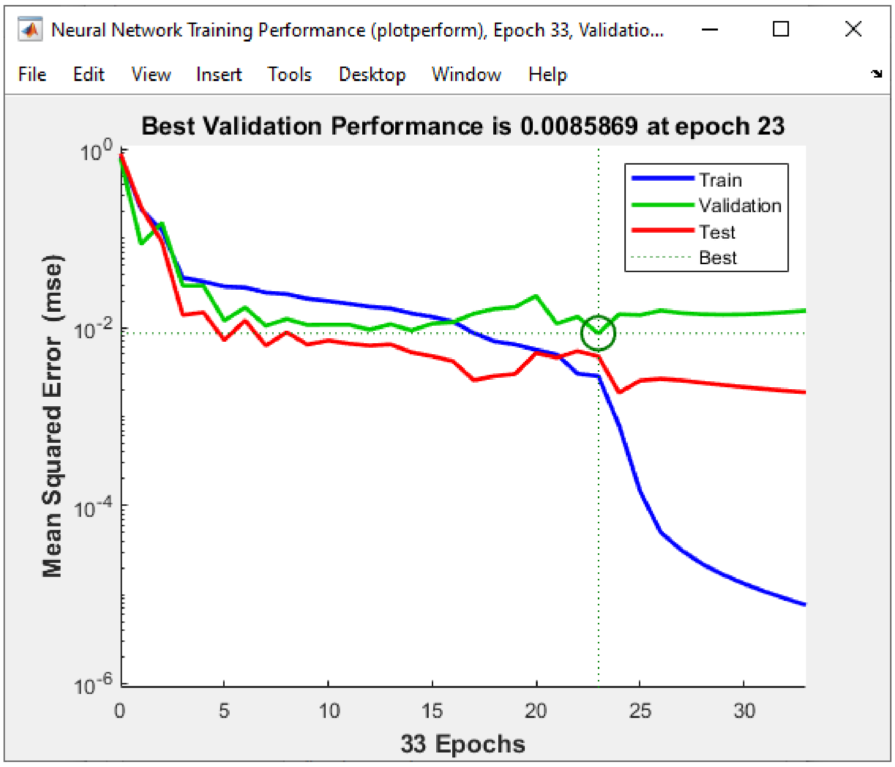


Figure 34: Case 7 Training Performance

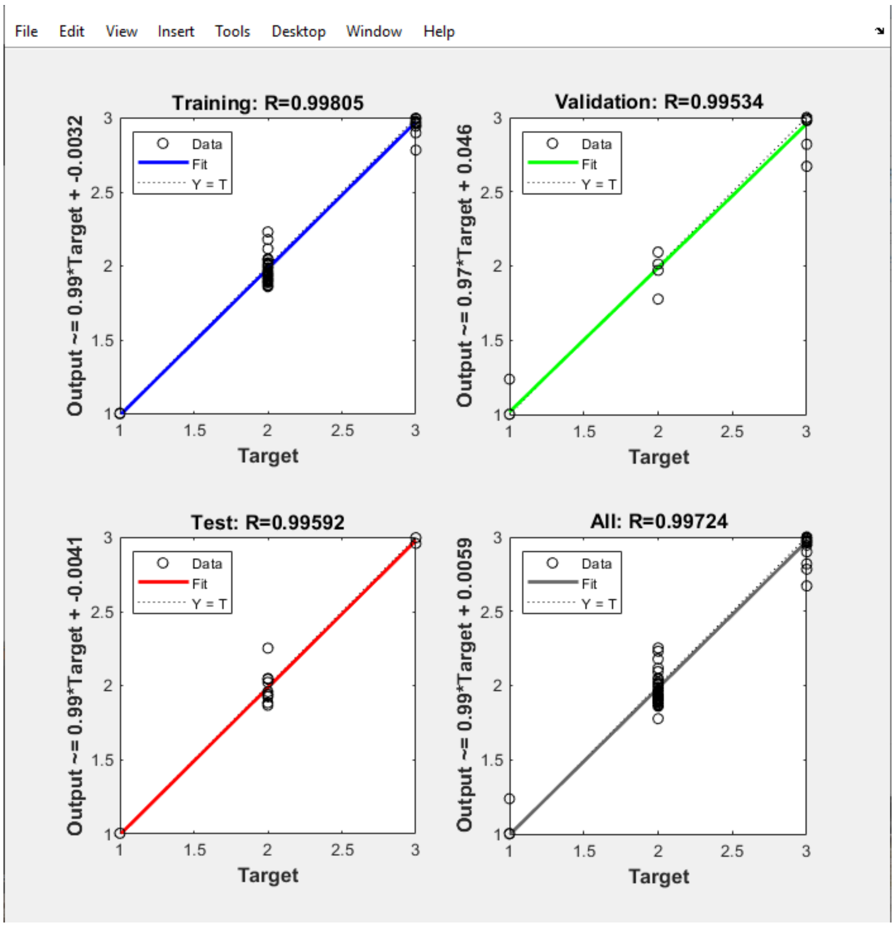


Figure 35: Case 7 Regression

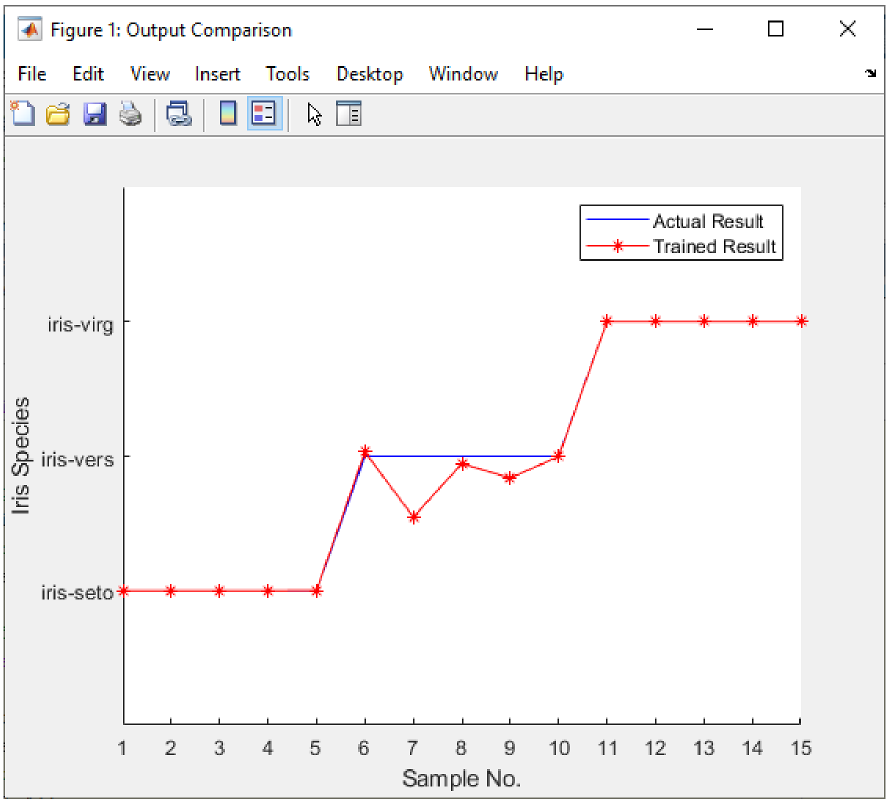


Figure 36: Case 7 Unrounded Simulated Output

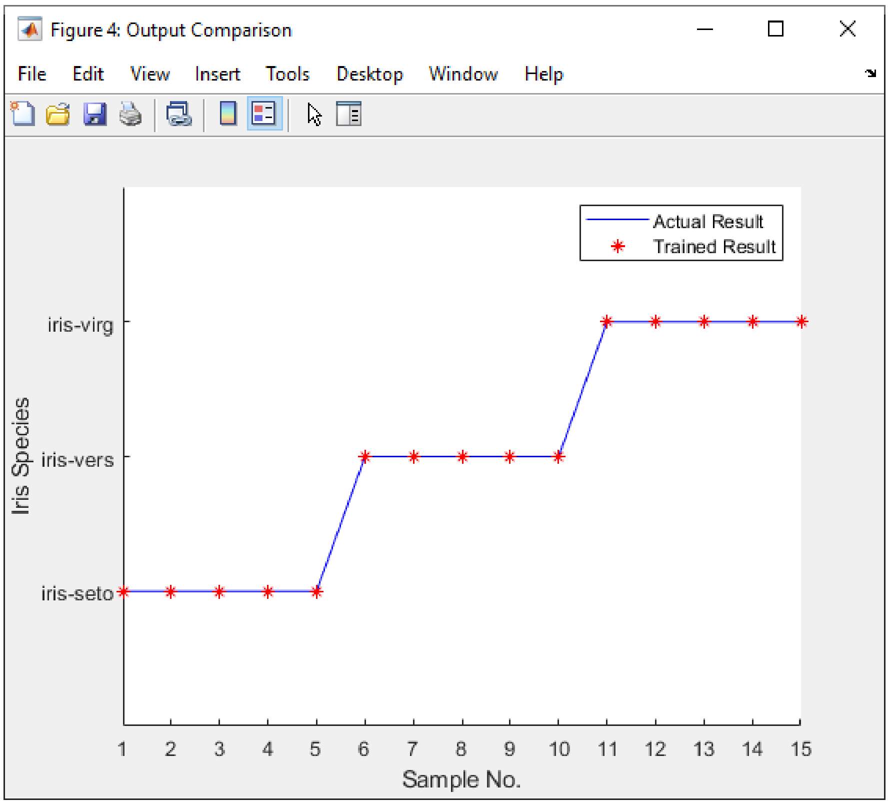
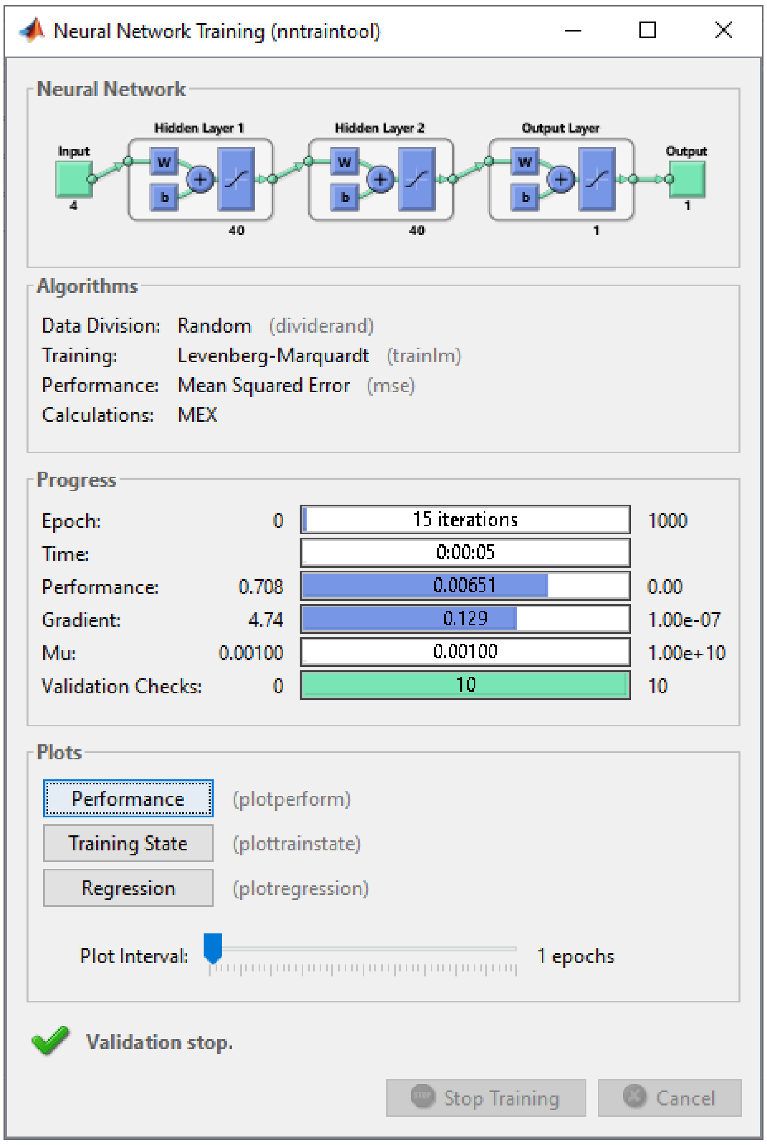


Figure 37: Case 7 Rounded Simulated Output

8.1.8 Case 8



Algorithms

Data Division:	Random (dividerand)
Training:	Levenberg-Marquardt (trainlm)
Performance:	Mean Squared Error (mse)
Calculations:	MEX

Progress

Epoch:	0	15 iterations	1000
Time:		0:00:05	
Performance:	0.708	0.00651	0.00
Gradient:	4.74	0.129	1.00e-07
Mu:	0.00100	0.00100	1.00e+10
Validation Checks:	0	10	10

Plots

Performance	(plotperform)
Training State	(plottrainstate)
Regression	(plotregression)

Plot Interval: 1 epochs

✓ Validation stop.

Figure 38: Case 8 Training Network

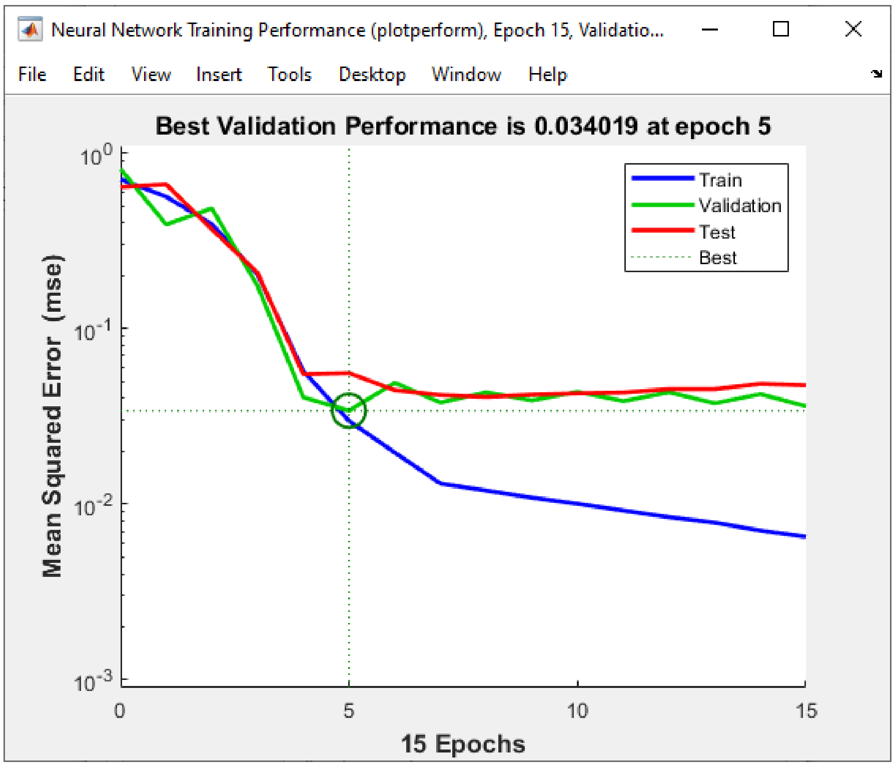


Figure 39: Case 8 Training Performance

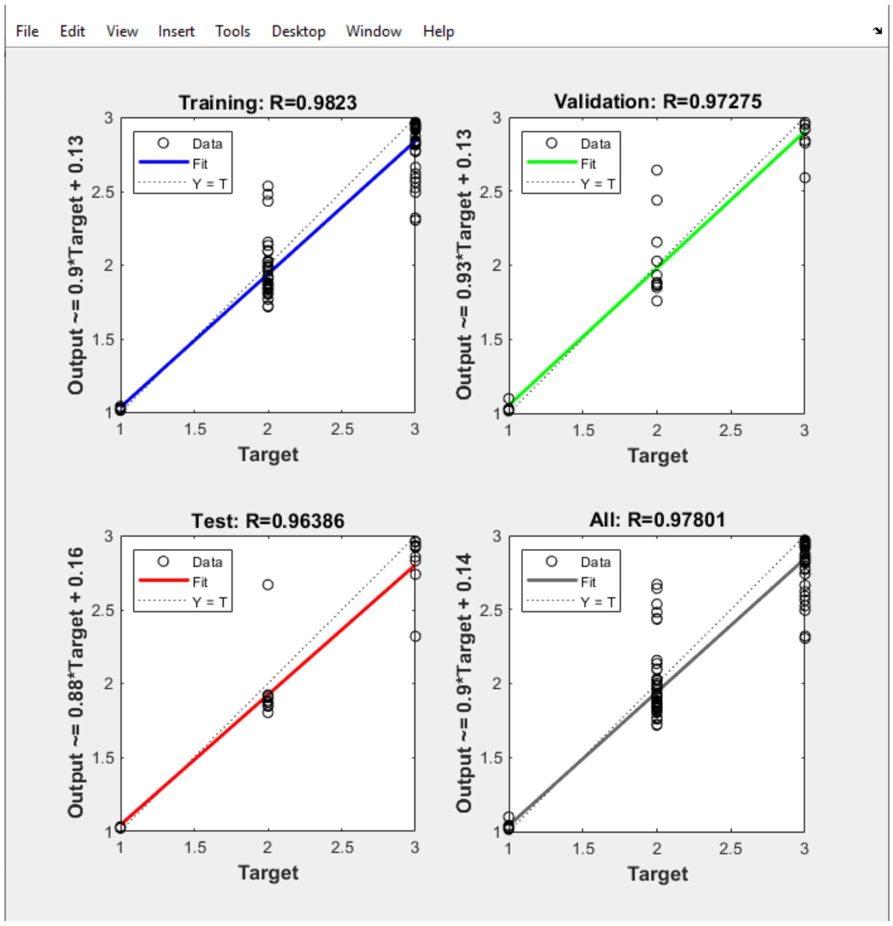


Figure 40: Case 8 Regression

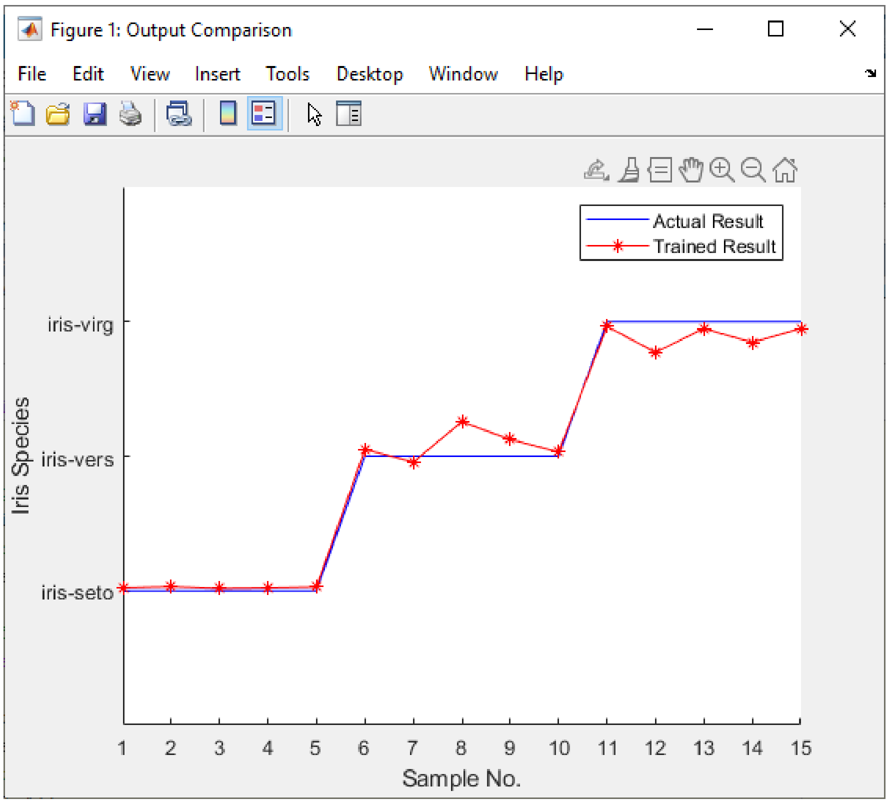


Figure 41: Case 8 Unrounded Simulated Output

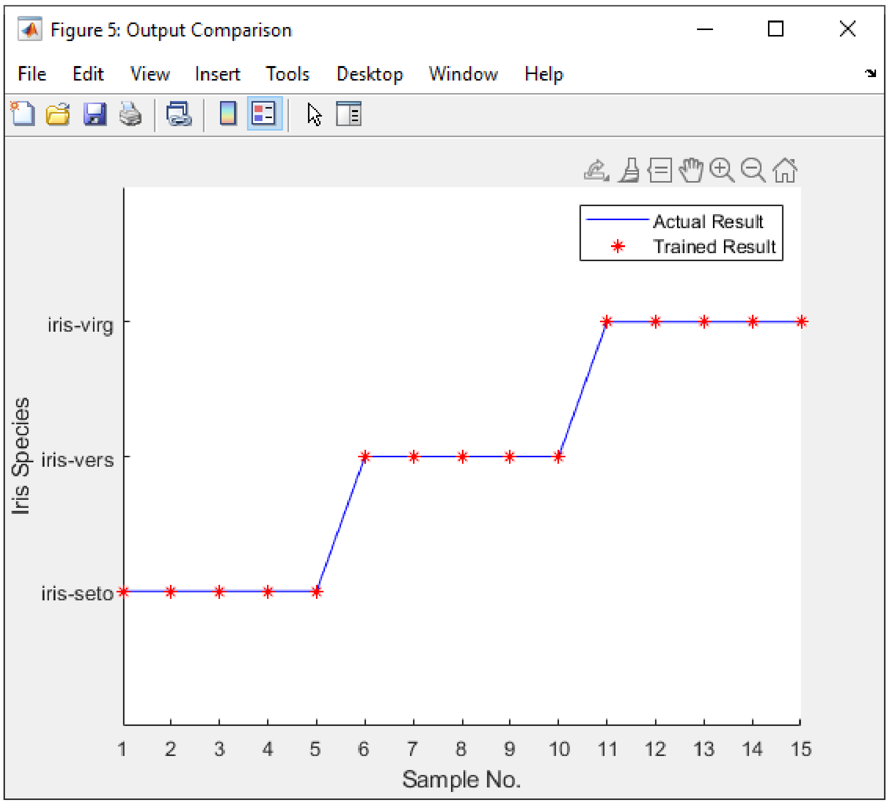
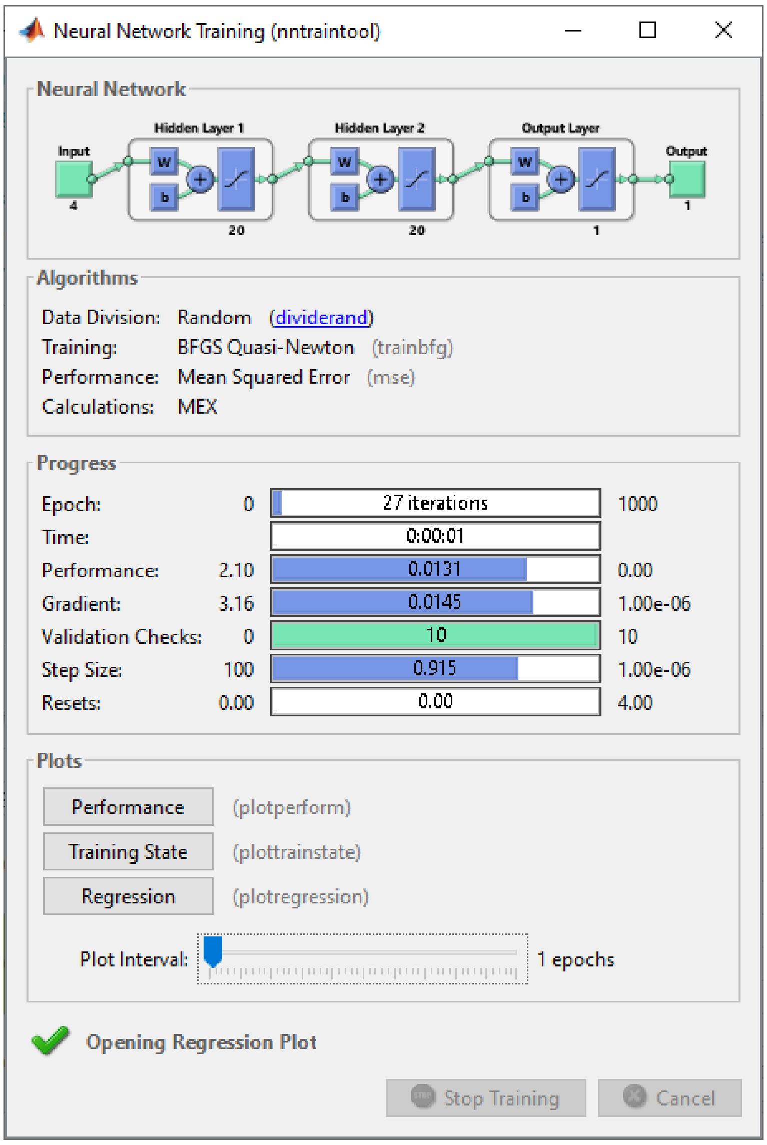


Figure 42: Case 8 Rounded Simulated Output

8.2 Variation of Training Algorithm

8.2.1 BFGS Quasi-Newton



Algorithms

Data Division: Random (dividerand)
Training: BFGS Quasi-Newton (trainbfg)
Performance: Mean Squared Error (mse)
Calculations: MEX

Progress

Epoch:	0	27 iterations	1000
Time:		0:00:01	
Performance:	2.10	0.0131	0.00
Gradient:	3.16	0.0145	1.00e-06
Validation Checks:	0	10	10
Step Size:	100	0.915	1.00e-06
Resets:	0.00	0.00	4.00

Plots

Performance (plotperform)
Training State (plottrainstate)
Regression (plotregression)

Plot Interval: 1 epochs

Opening Regression Plot

Figure 43: BFG Training Network

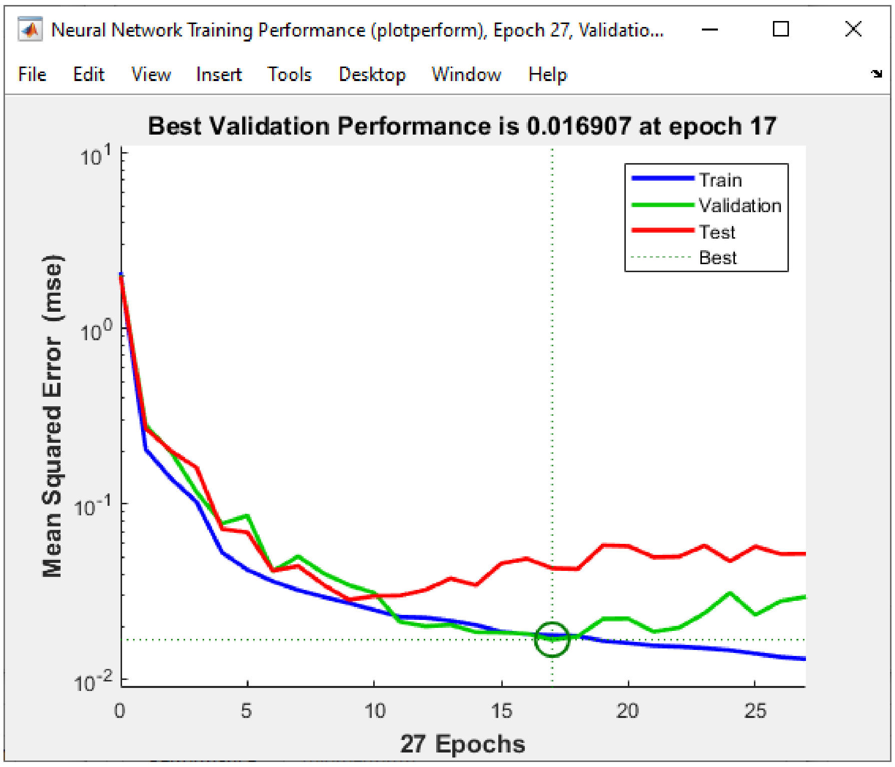


Figure 44: BFG Training Performance

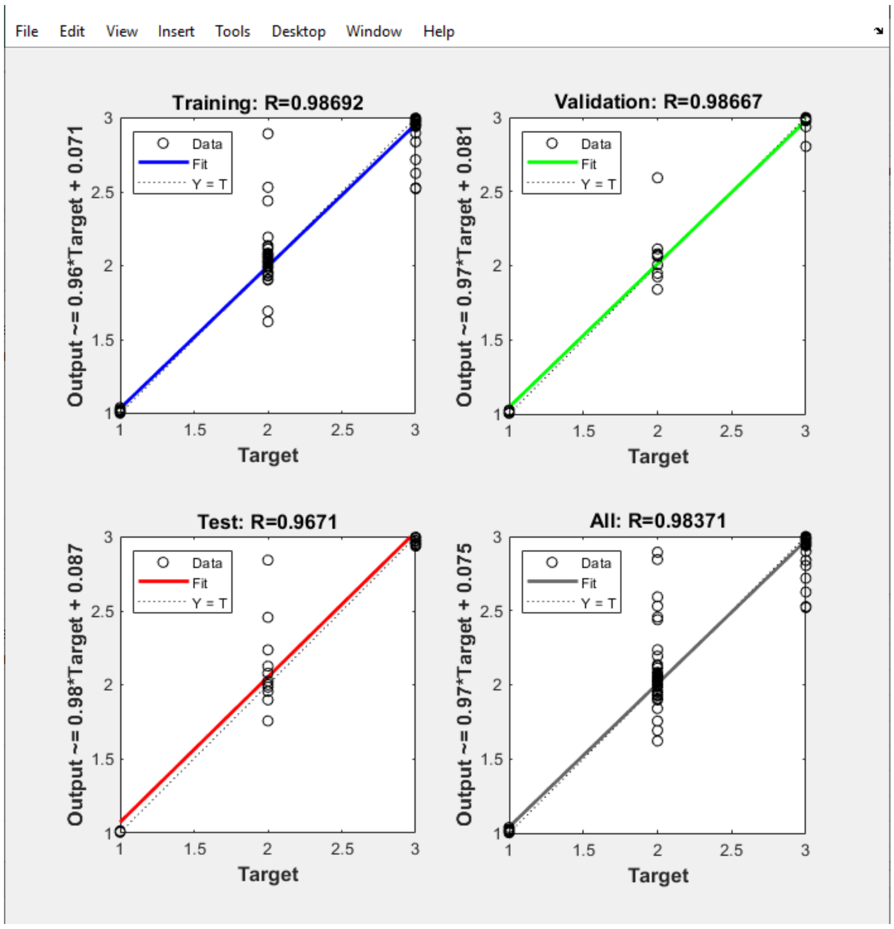


Figure 45: BFG Regression

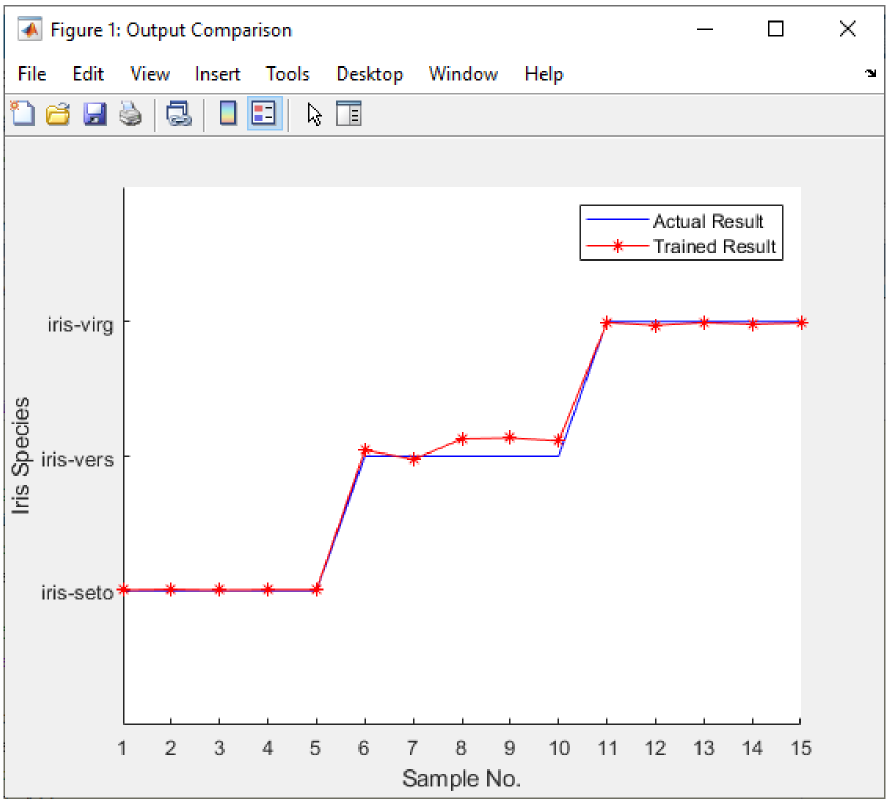


Figure 46: BFG Unrounded Simulated Output

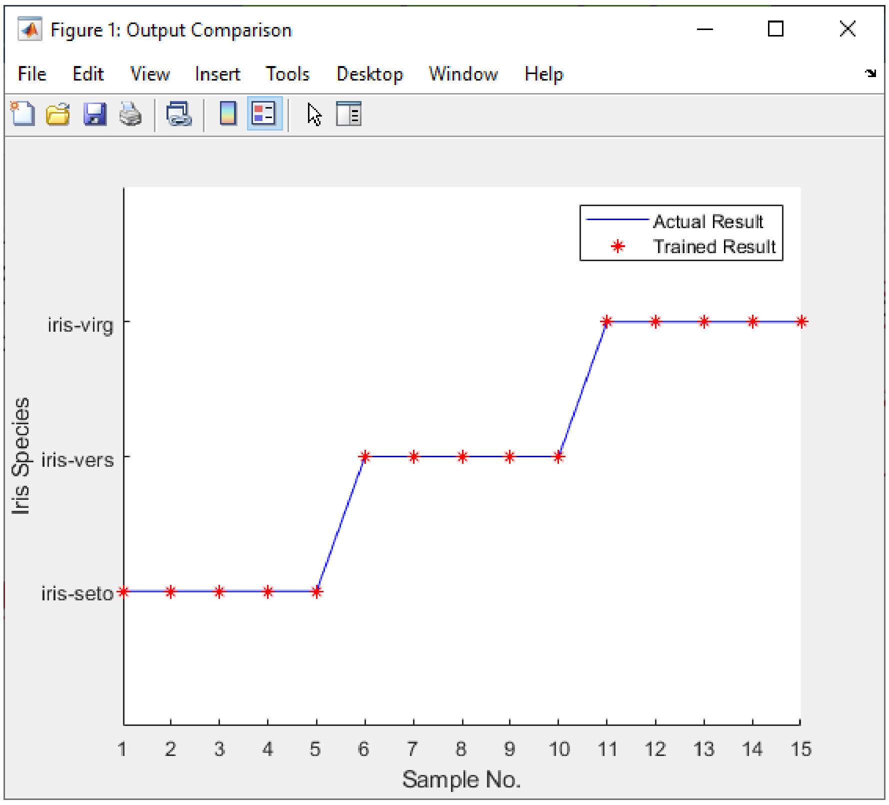
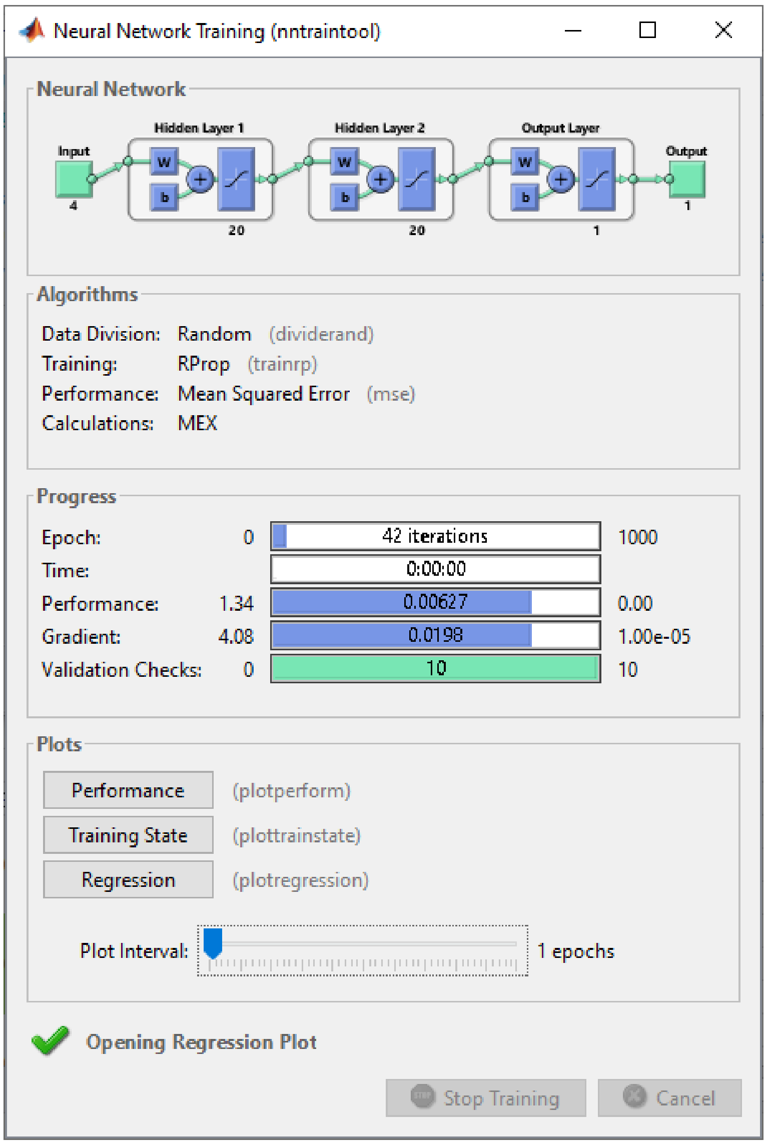


Figure 47: BFG Rounded Simulated Output

8.2.2 Resilient Backpropagation



Algorithms

Data Division: Random (dividerand)
Training: RProp (trainrp)
Performance: Mean Squared Error (mse)
Calculations: MEX

Progress

Epoch:	0	42 iterations	1000
Time:		0:00:00	
Performance:	1.34	0.00627	0.00
Gradient:	4.08	0.0198	1.00e-05
Validation Checks:	0	10	10

(plotperform)
(plottrainstate)
(plotregression)

Plot Interval: 1 epochs

✓ Opening Regression Plot

Figure 48: RP Training Network

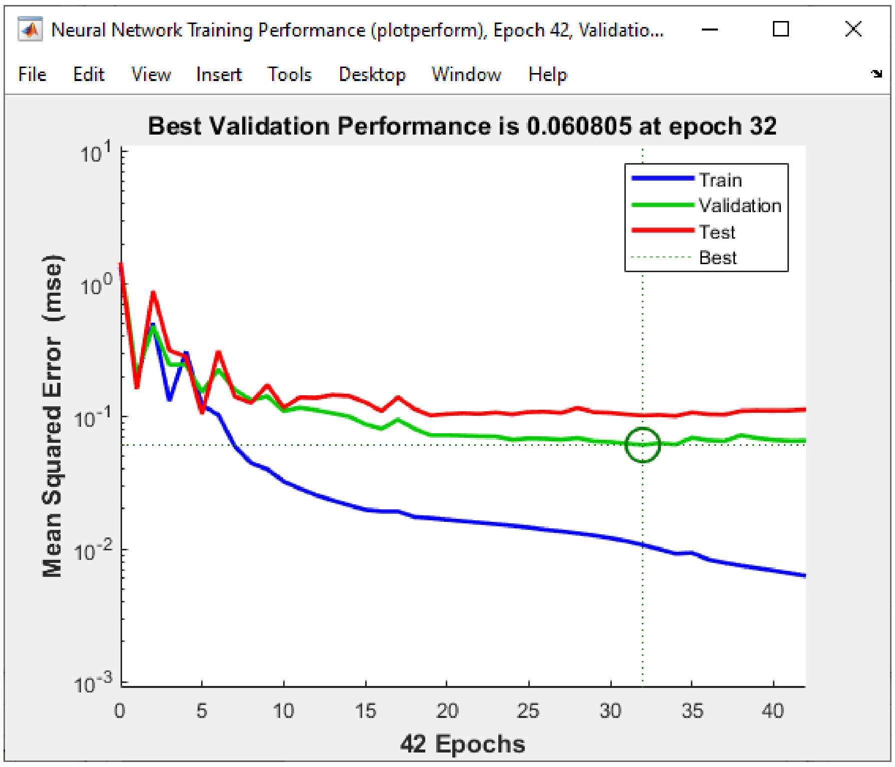


Figure 49: RP Training Network

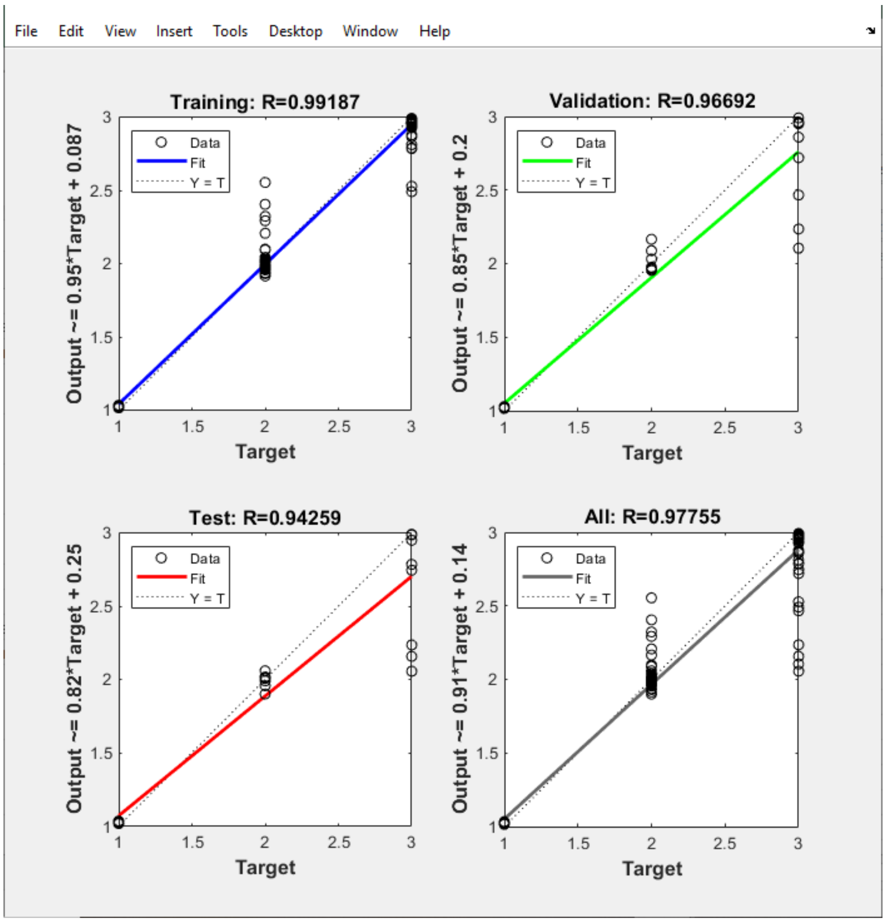


Figure 50: RP Regression

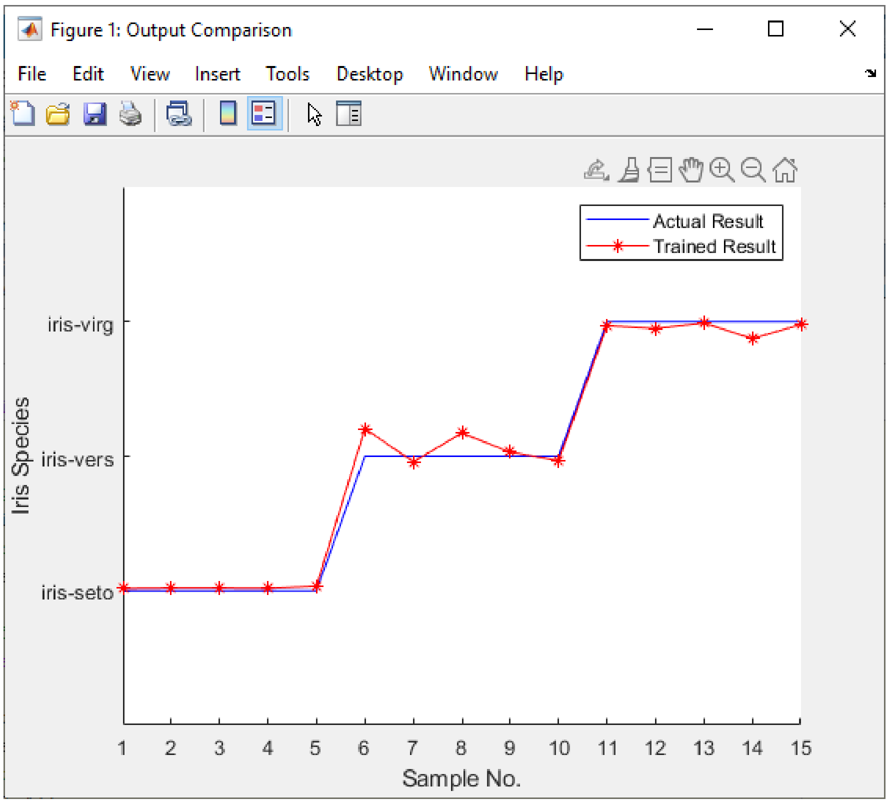


Figure 51: RP Unrounded Simulated Output

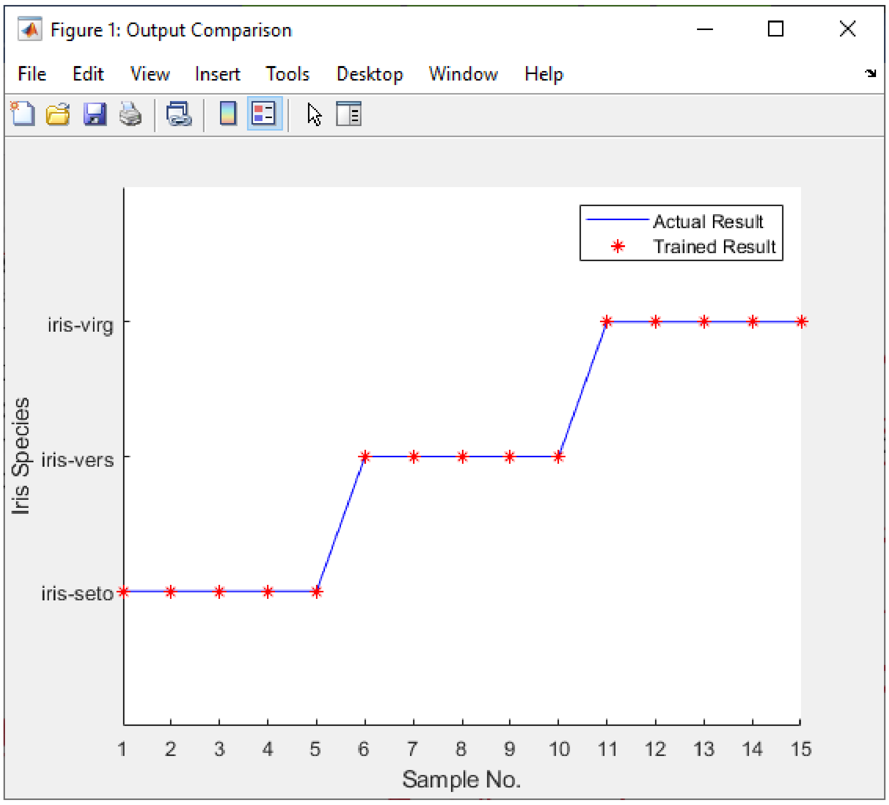
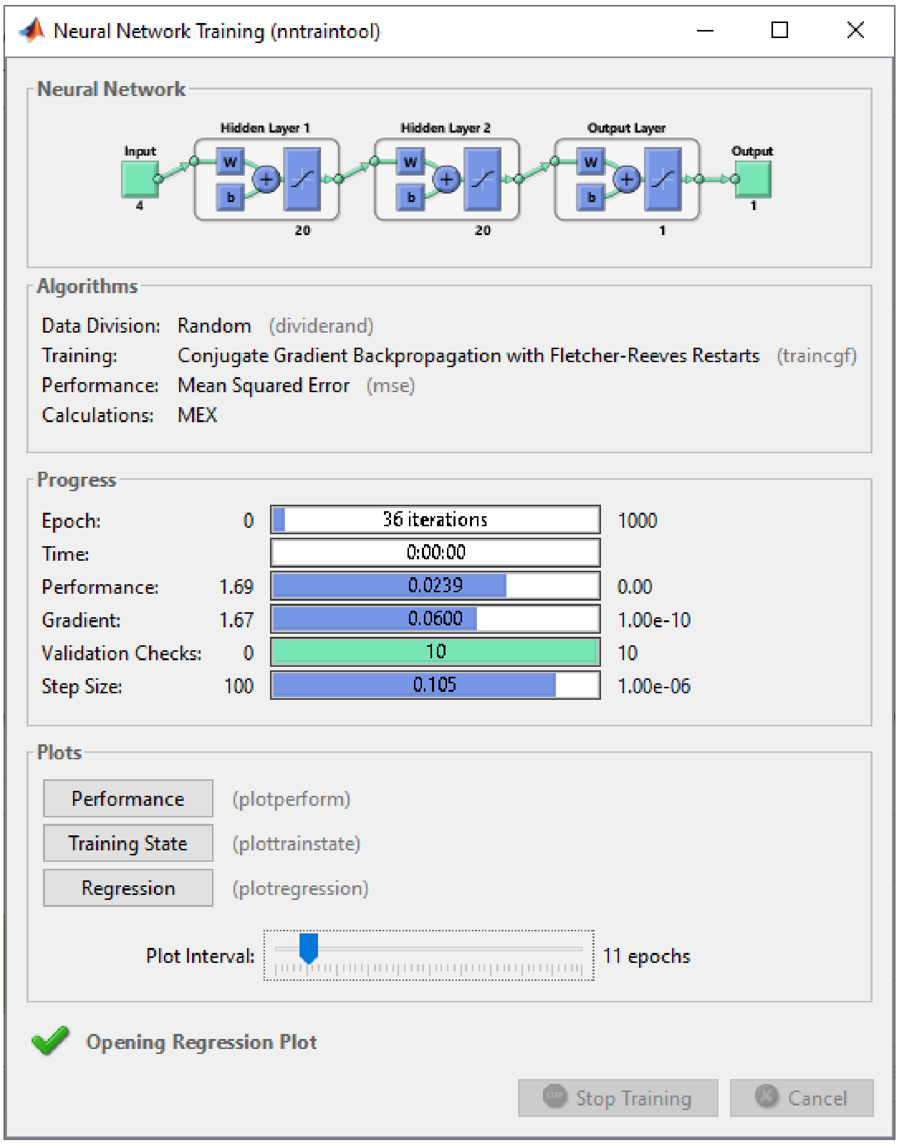


Figure 52: RP Rounded Simulated Output

8.2.3 Fletcher-Powell Conjugate Gradient



Algorithms

Data Division: Random (dividerand)
Training: Conjugate Gradient Backpropagation with Fletcher-Reeves Restarts (traincgf)
Performance: Mean Squared Error (mse)
Calculations: MEX

Progress

Epoch:	0	36 iterations	1000
Time:		0:00:00	
Performance:	1.69	0.0239	0.00
Gradient:	1.67	0.0600	1.00e-10
Validation Checks:	0	10	10
Step Size:	100	0.105	1.00e-06

Plots

Performance (plotperform)
Training State (plottrainstate)
Regression (plotregression)

Plot Interval: 11 epochs

Opening Regression Plot

Figure 53: CGF Training Network

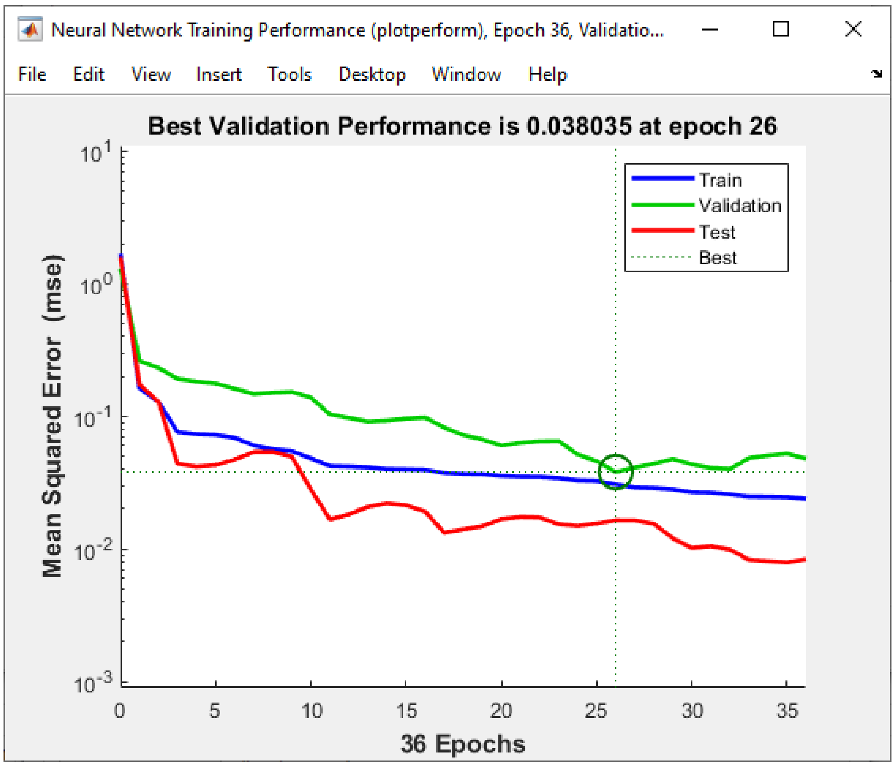


Figure 54: CGF Training Performance

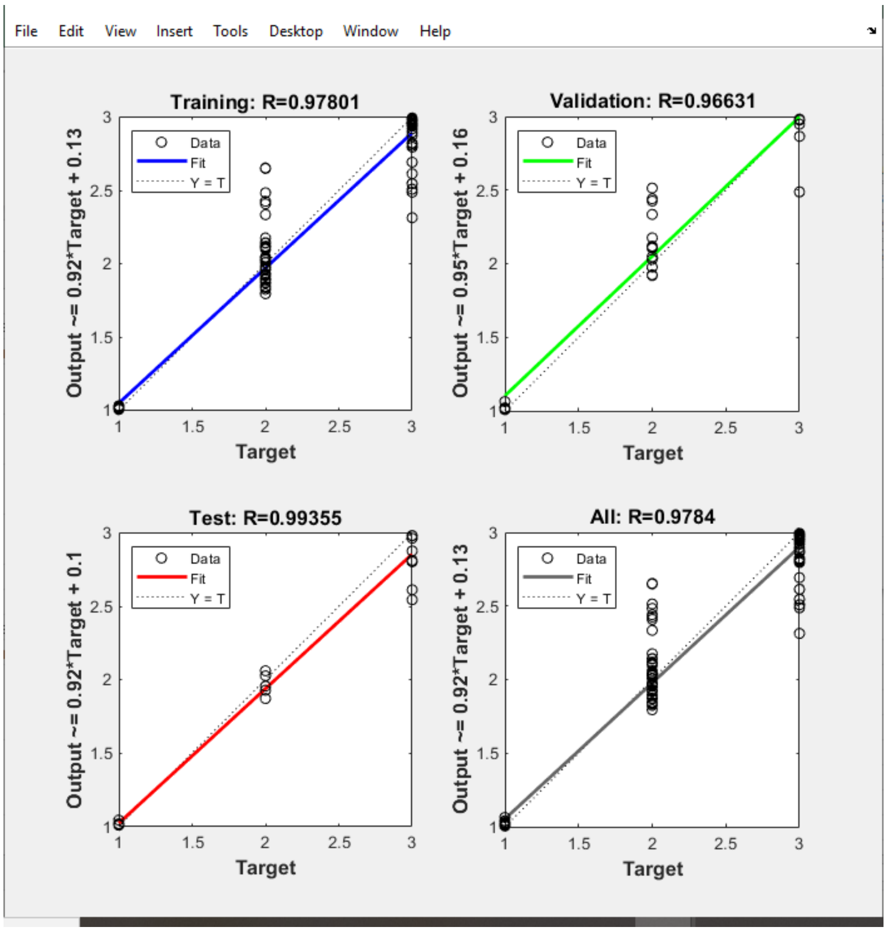


Figure 55: CGF Regression

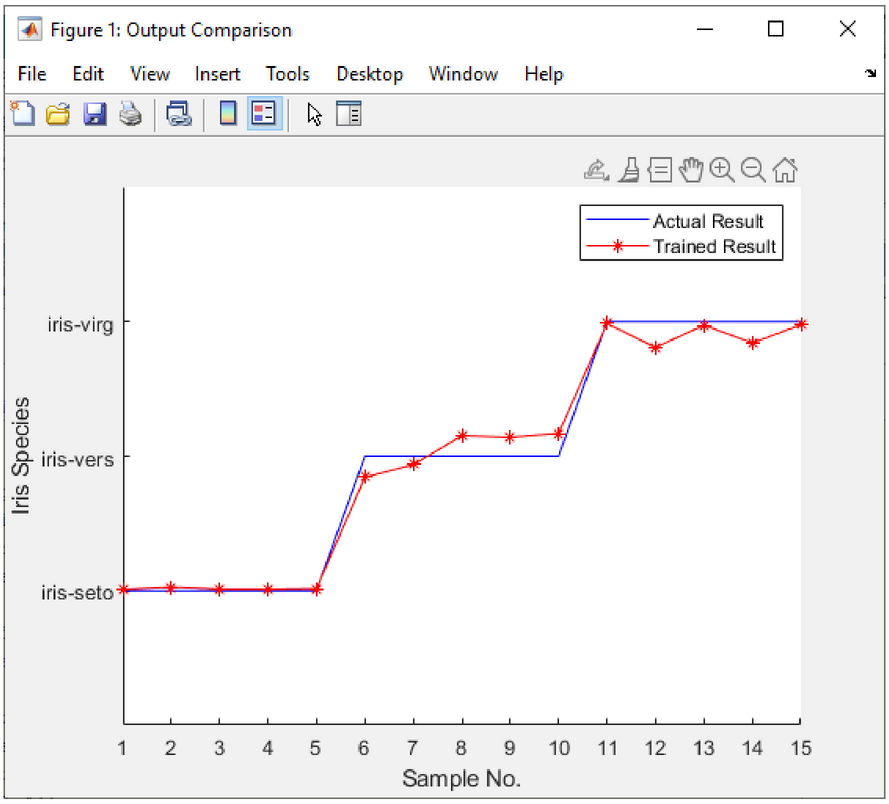


Figure 56: CGF Unrounded Simulated Output

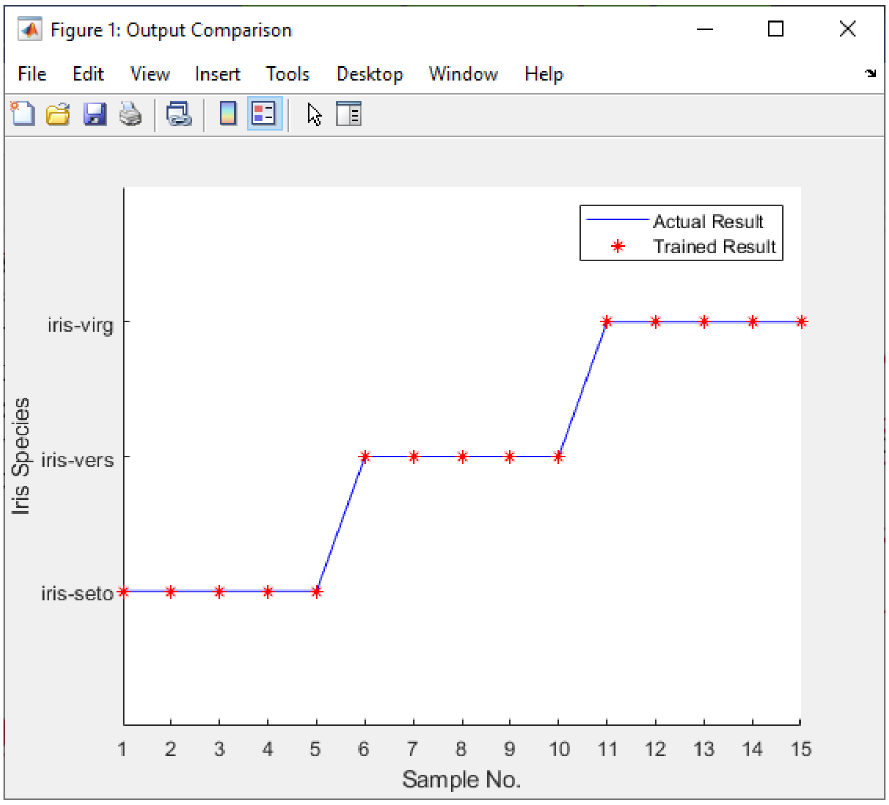
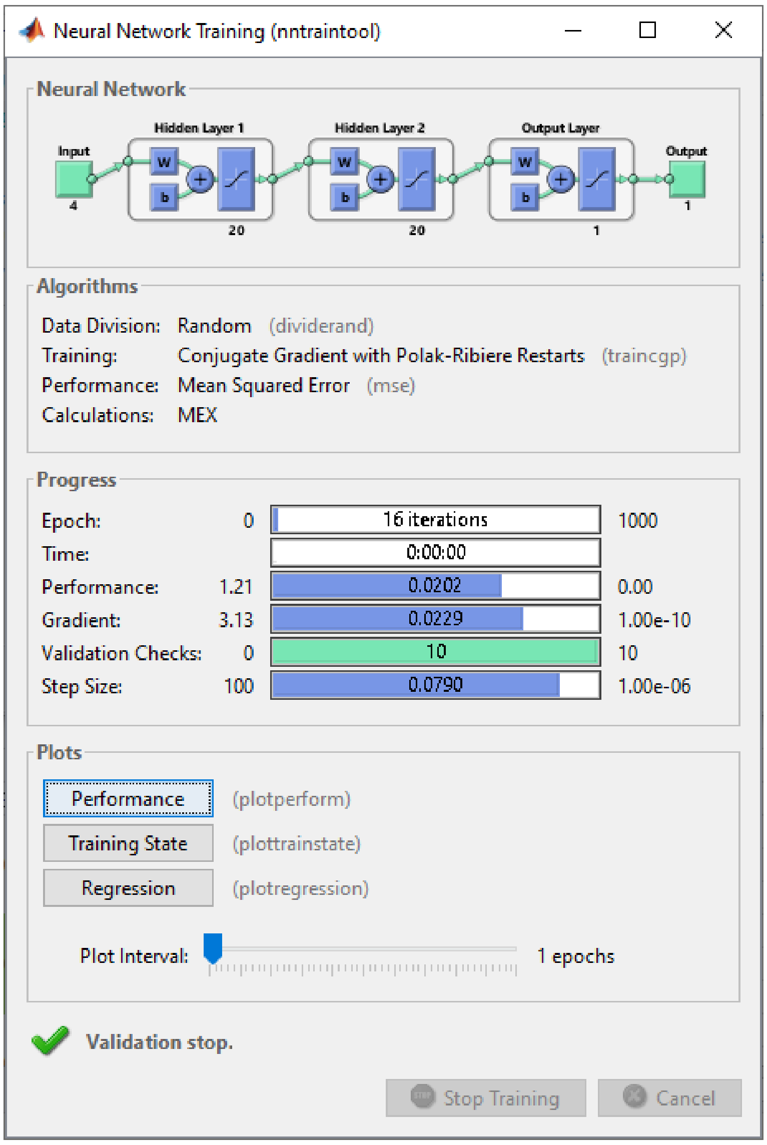


Figure 57: CGF Rounded Simulated Output

8.2.4 Polak-Ribiére Conjugate Gradient



Algorithms

Data Division: Random (dividerand)
Training: Conjugate Gradient with Polak-Ribiere Restarts (traincgp)
Performance: Mean Squared Error (mse)
Calculations: MEX

Progress

Epoch:	0	16 iterations	1000
Time:		0:00:00	
Performance:	1.21	0.0202	0.00
Gradient:	3.13	0.0229	1.00e-10
Validation Checks:	0	10	10
Step Size:	100	0.0790	1.00e-06

(plotperform)
(plottrainstate)
(plotregression)

Plot Interval: 1 epochs

Validation stop.

Figure 58: CGP Training Network

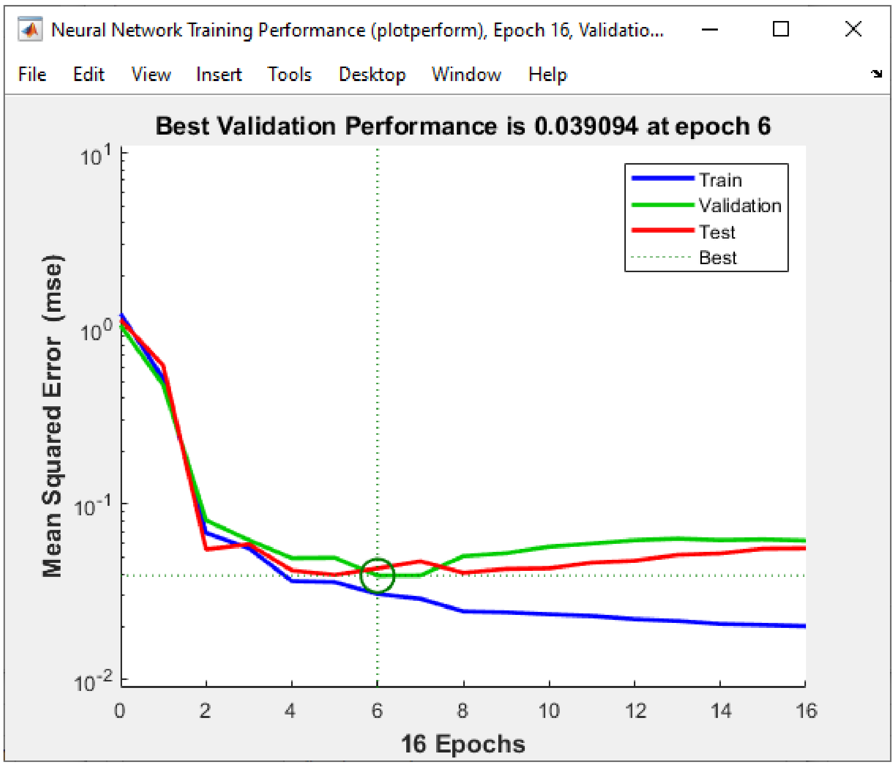


Figure 59: CGP Training Performance

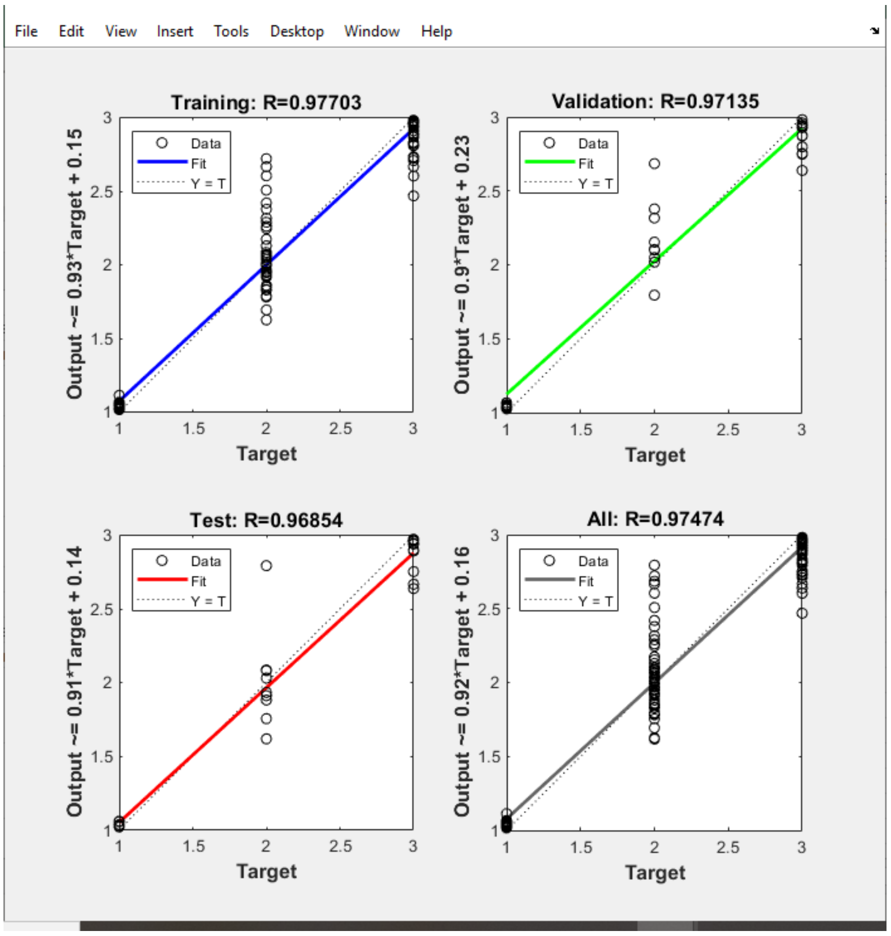


Figure 60: CGP Regression

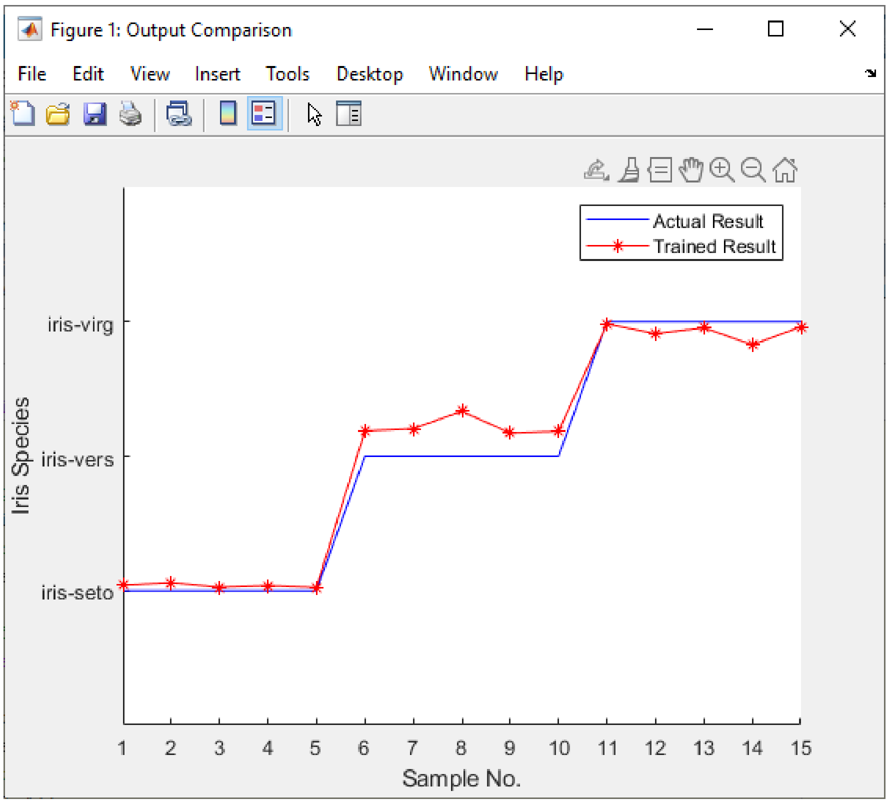


Figure 61: CGP Unrounded Simulated Output

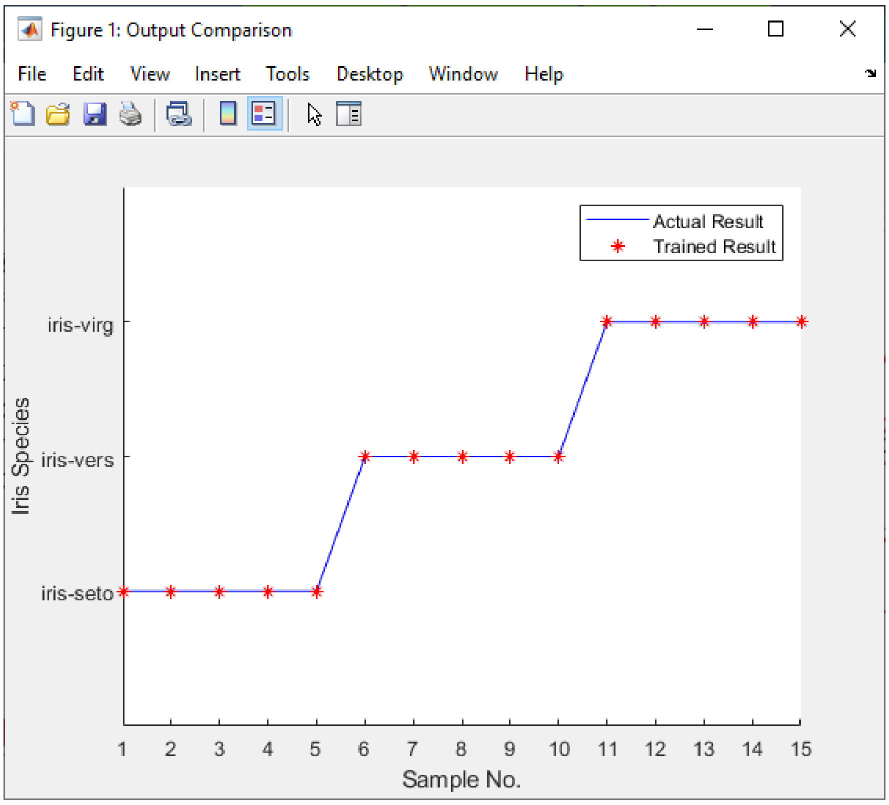
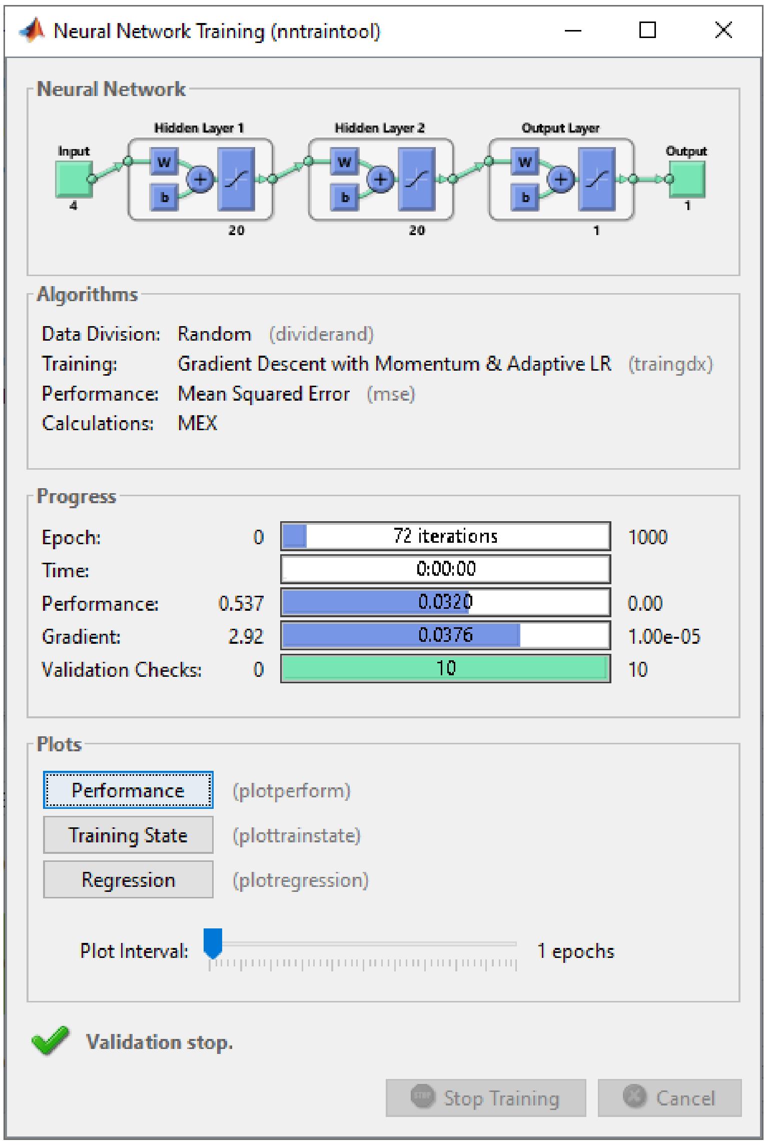


Figure 62: CGP Rounded Simulated Output

8.2.5 Variable Learning Rate Backpropagation


Neural Network Training (nntraintool)

Neural Network

Algorithms

Data Division: Random (dividerand)
Training: Gradient Descent with Momentum & Adaptive LR (traingdx)
Performance: Mean Squared Error (mse)
Calculations: MEX

Progress

```
Epoch:                0      72 iterations     1000
Time:                        0:00:00
Performance:      0.537      0.0320            0.00
Gradient:          2.92      0.0376            1.00e-05
Validation Checks:    0      10                10
```

Plots

Performance (plotperform)
Training State (plottrainstate)
Regression (plotregression)

Plot Interval: 1 epochs

Validation stop.

Stop Training Cancel

Figure 63: GDX Training Network

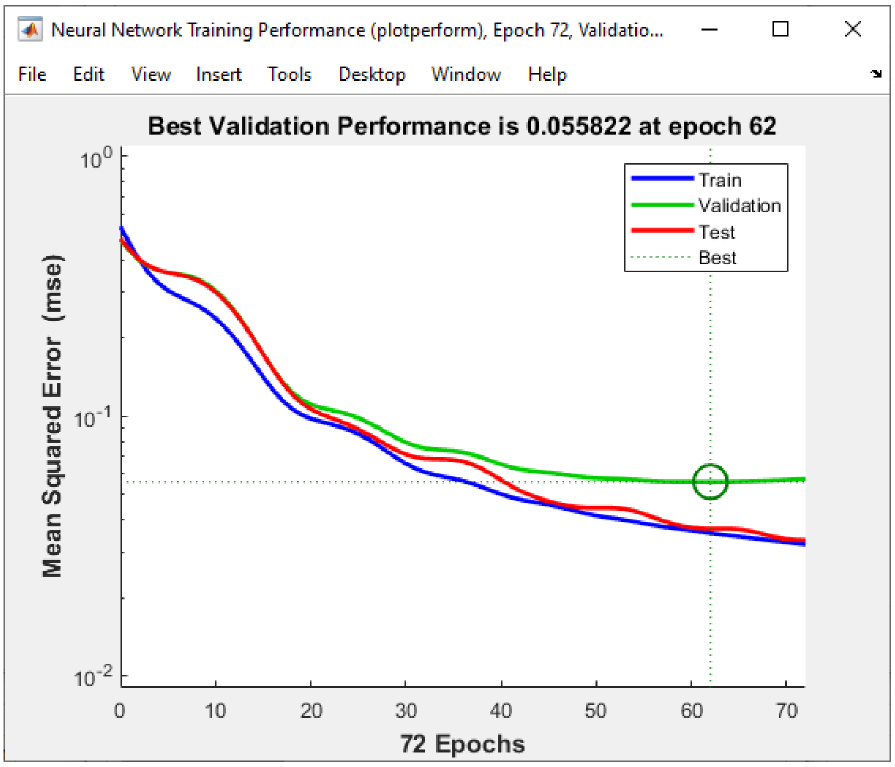


Figure 64: GDX Training Performance

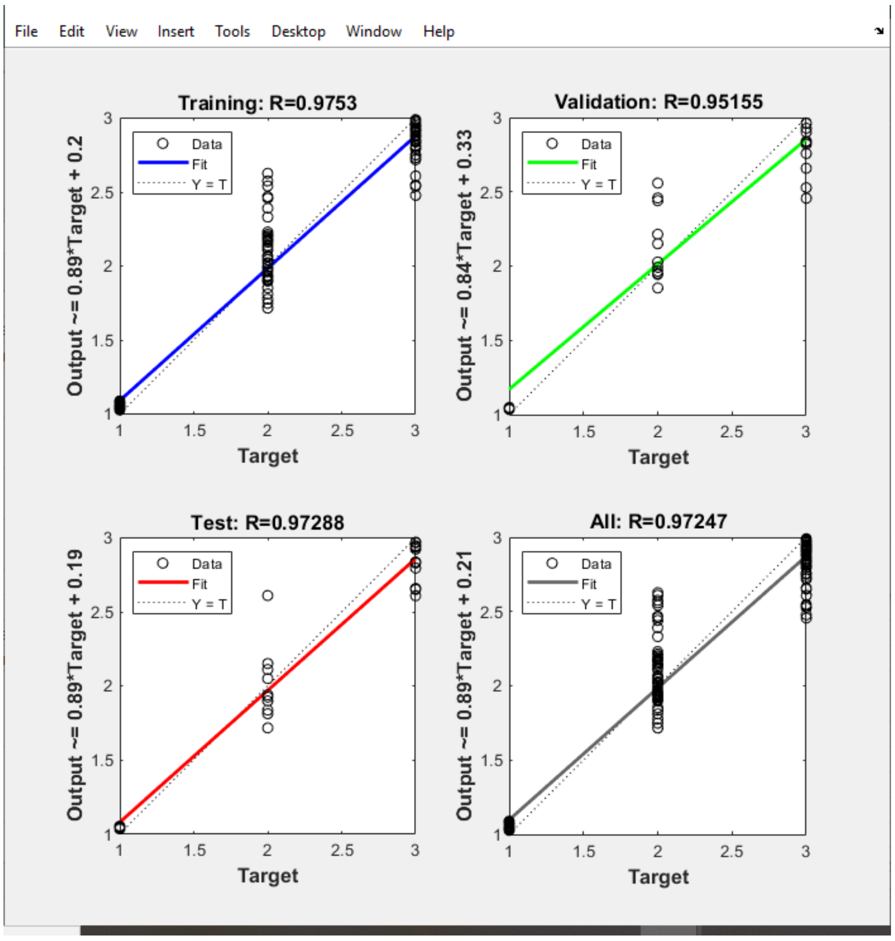


Figure 65: GDX Regression

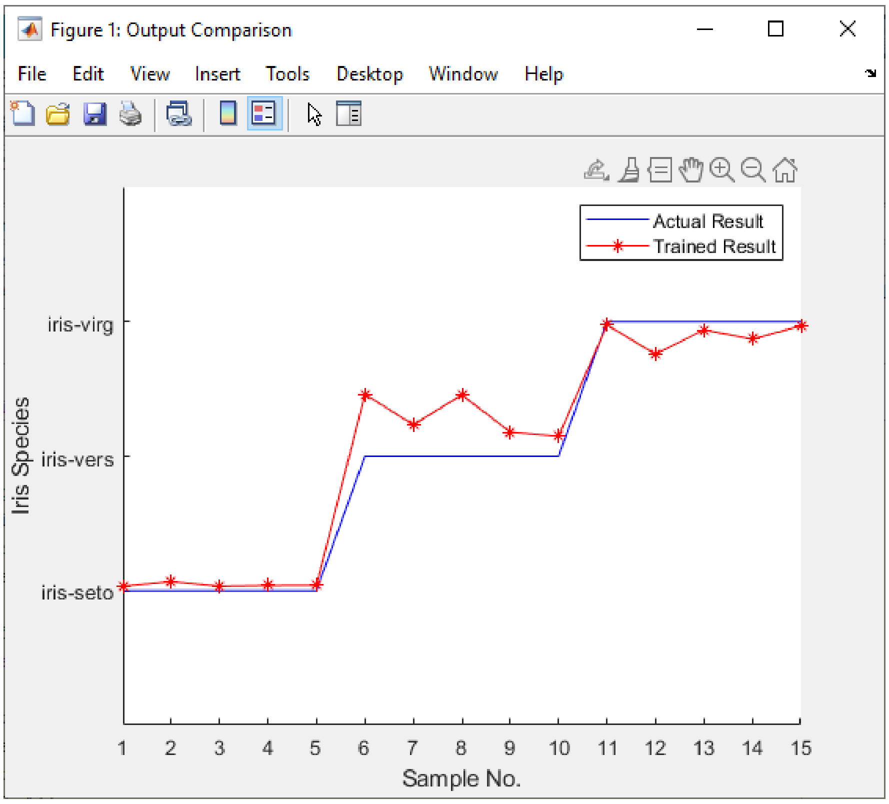


Figure 66: GDX Unrounded Simulated Output

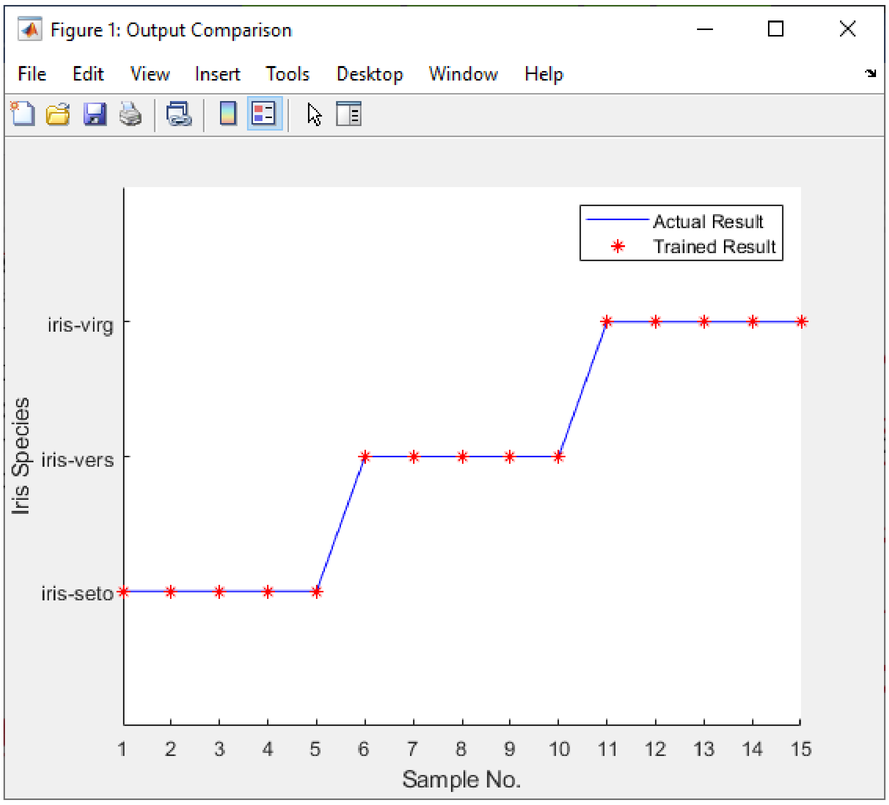
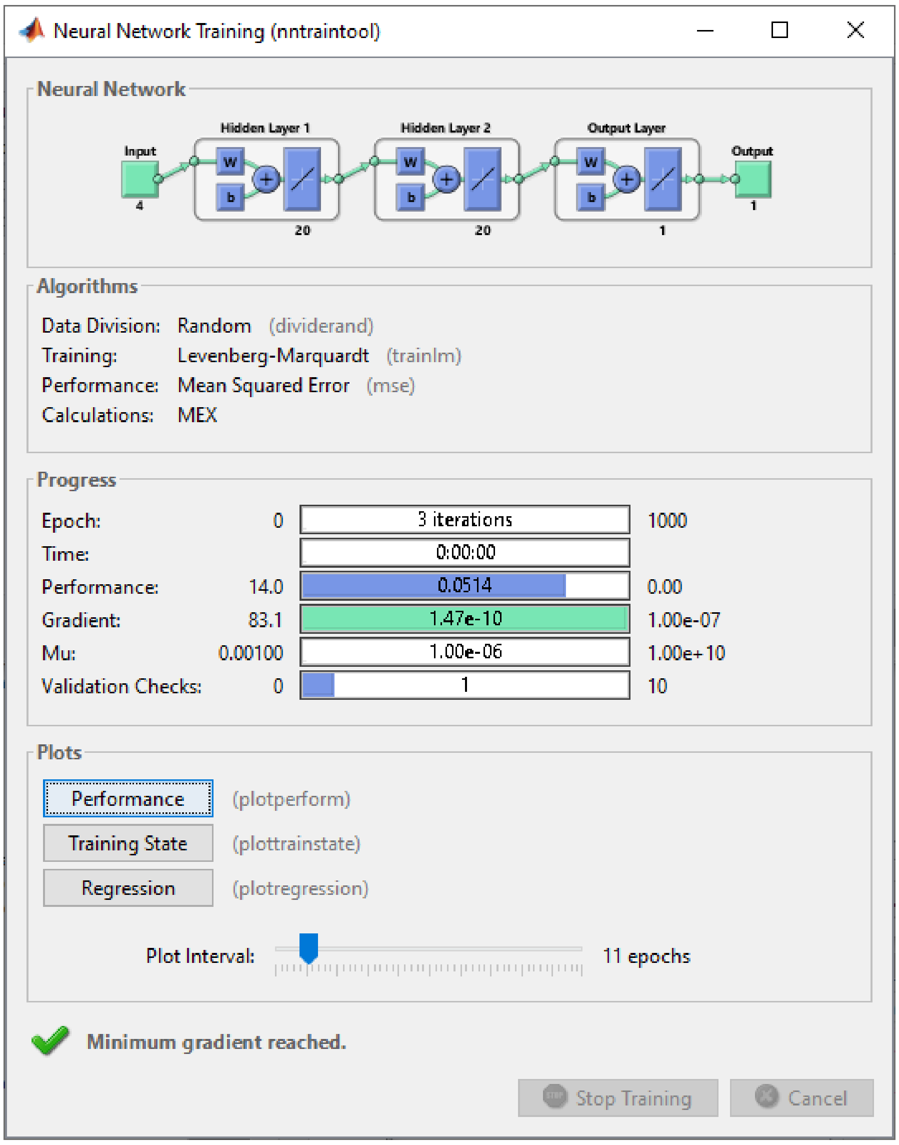


Figure 67: GDX Rounded Simulated Output

8.3 Variation of Transfer Function

8.3.1 Case 9


Neural Network Training (nntraintool)

Neural Network

Algorithms

Data Division: Random (dividerand)
Training: Levenberg-Marquardt (trainlm)
Performance: Mean Squared Error (mse)
Calculations: MEX

Progress

Epoch:	0	3 iterations	1000
Time:		0:00:00	
Performance:	14.0	0.0514	0.00
Gradient:	83.1	1.47e-10	1.00e-07
Mu:	0.00100	1.00e-06	1.00e+10
Validation Checks:	0	1	10

Plots

Performance (plotperform)
Training State (plottrainstate)
Regression (plotregression)

Plot Interval: 11 epochs

Minimum gradient reached.

Figure 68: Case 9 Training Network

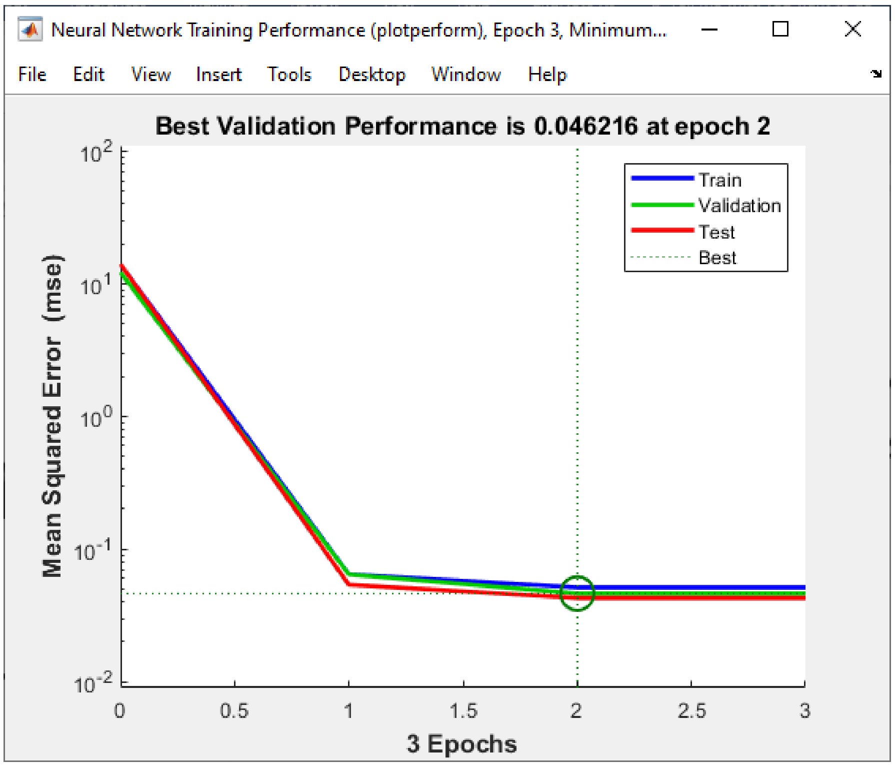


Figure 69: Case 9 Training Performance

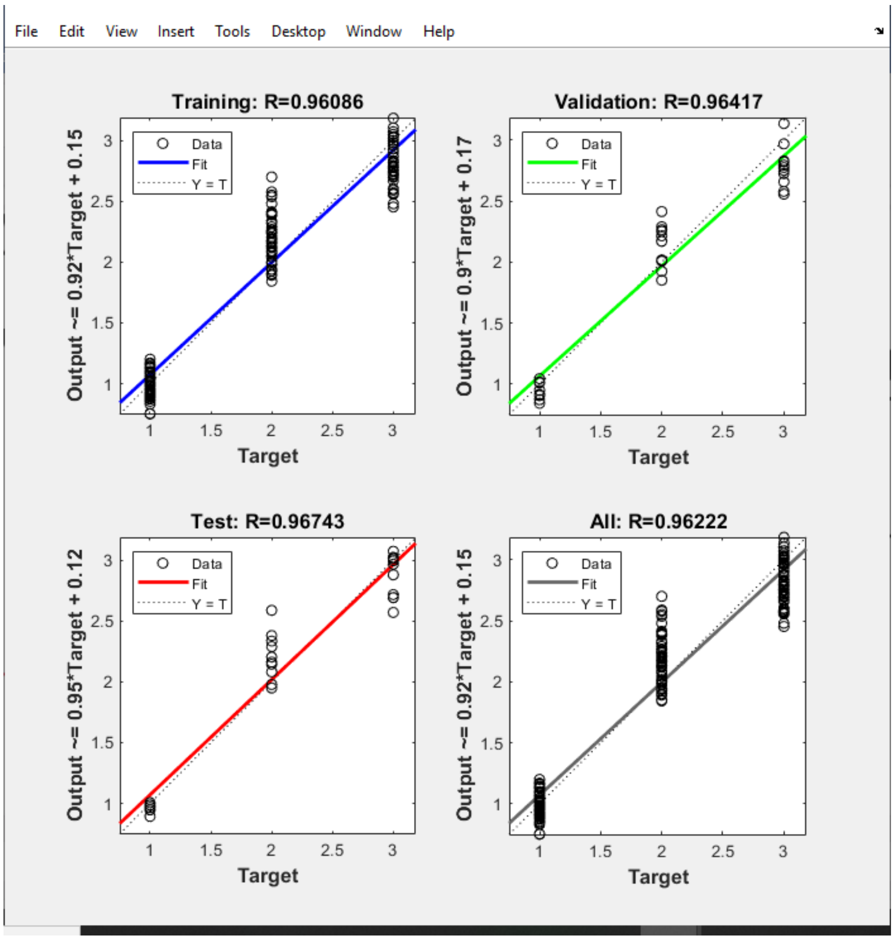


Figure 70: Case 9 Regression

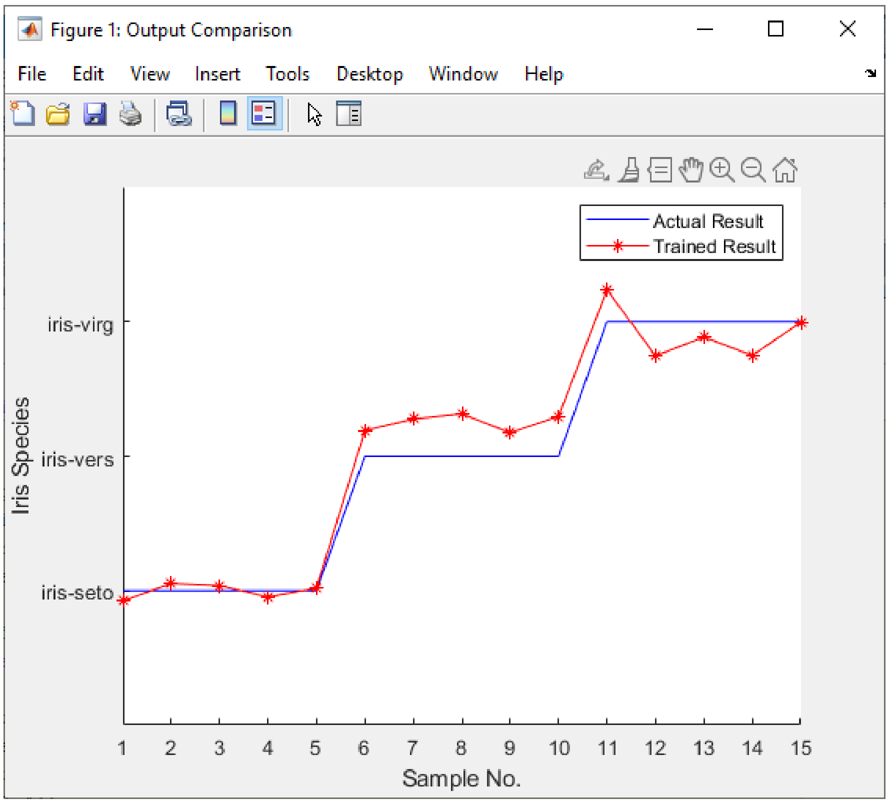


Figure 71: Case 9 Unrounded Simulated Output

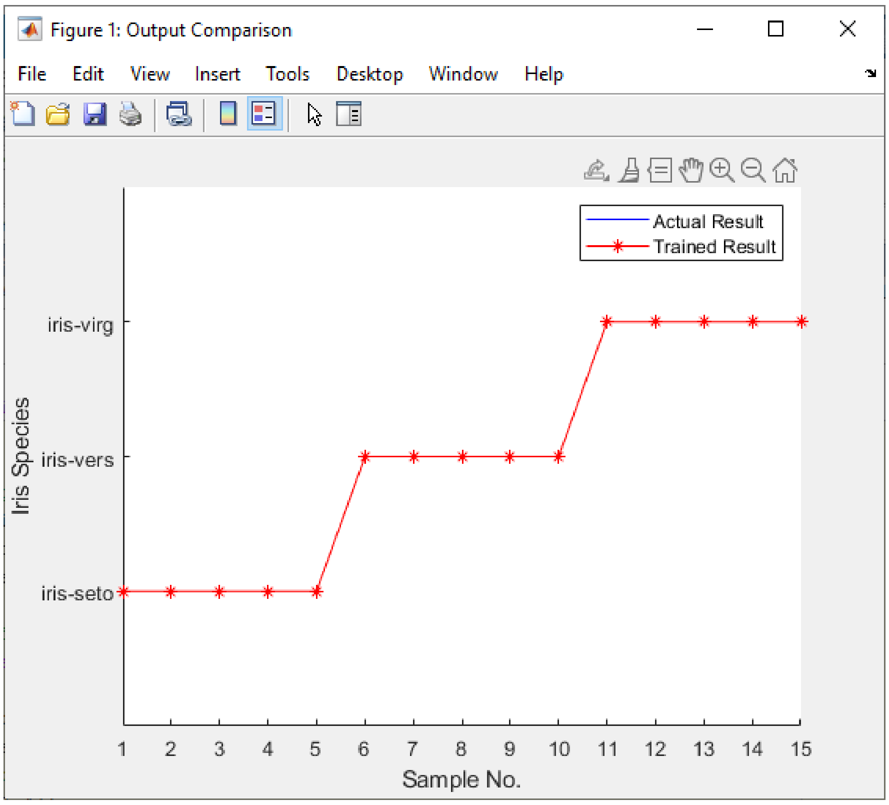
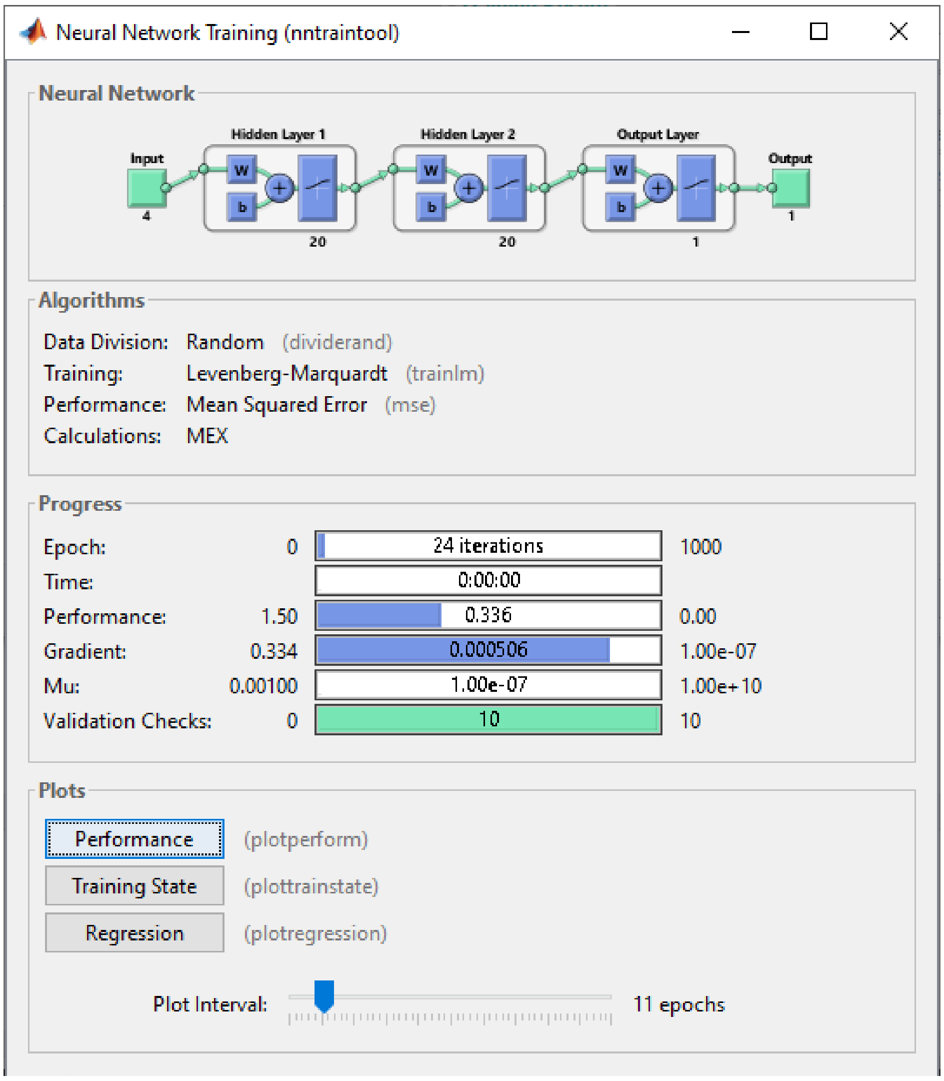


Figure 72: Case 9 Rounded Simulated Output

8.3.2 Case 10


Neural Network Training (nntraintool)

Neural Network

Input
4
Hidden Layer 1
20
Hidden Layer 2
20
Output Layer
1
Output
1

Algorithms

Data Division: Random (dividerand)
Training: Levenberg-Marquardt (trainlm)
Performance: Mean Squared Error (mse)
Calculations: MEX

Progress

```
Epoch:                 0   24 iterations   1000
Time:                      0:00:00
Performance:        1.50   0.336           0.00
Gradient:          0.334   0.000506        1.00e-07
Mu:              0.00100   1.00e-07        1.00e+10
Validation Checks:     0   10              10
```

Plots

Performance (plotperform)
Training State (plottrainstate)
Regression (plotregression)

Plot Interval: 11 epochs

Figure 73: Case 10 Training Network

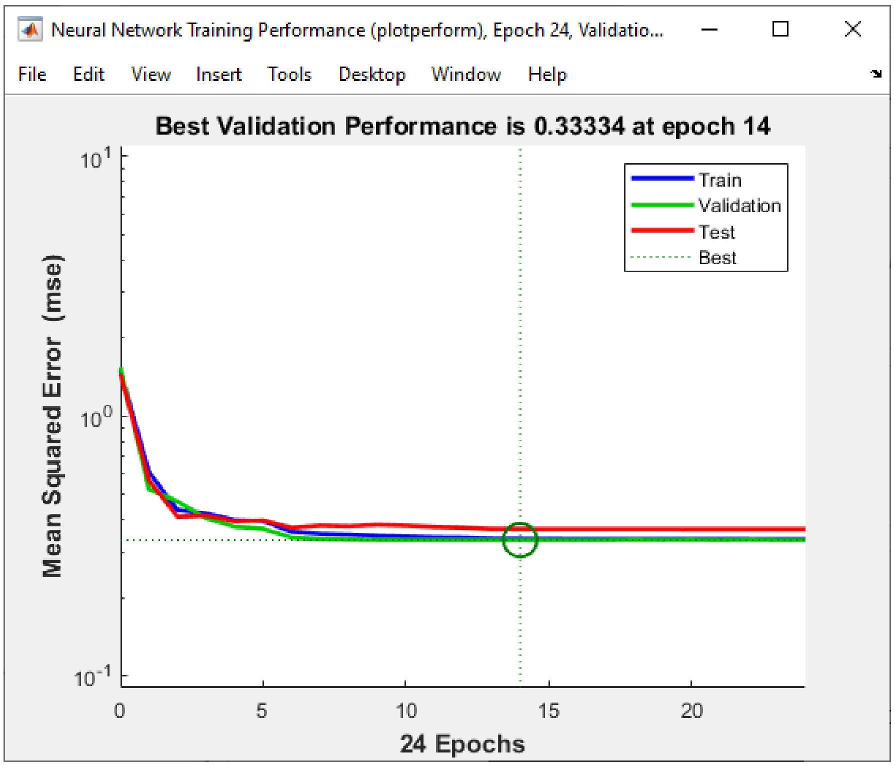


Figure 74: Case 10 Training Performance

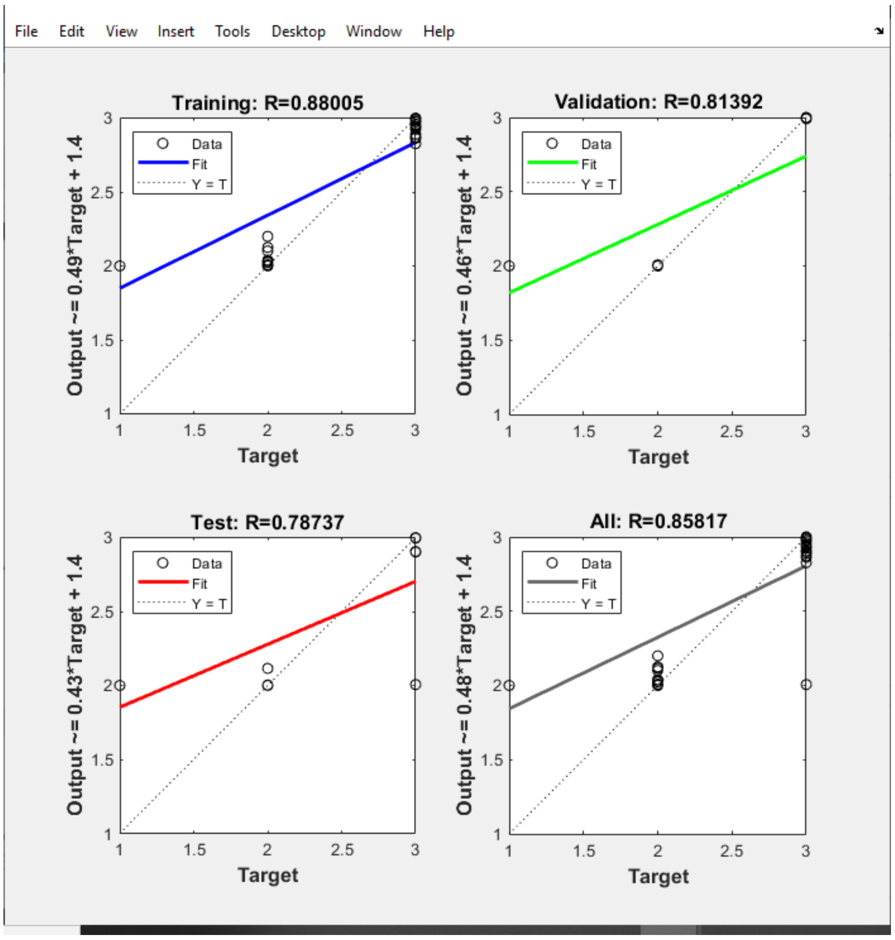


Figure 75: Case 10 Regression

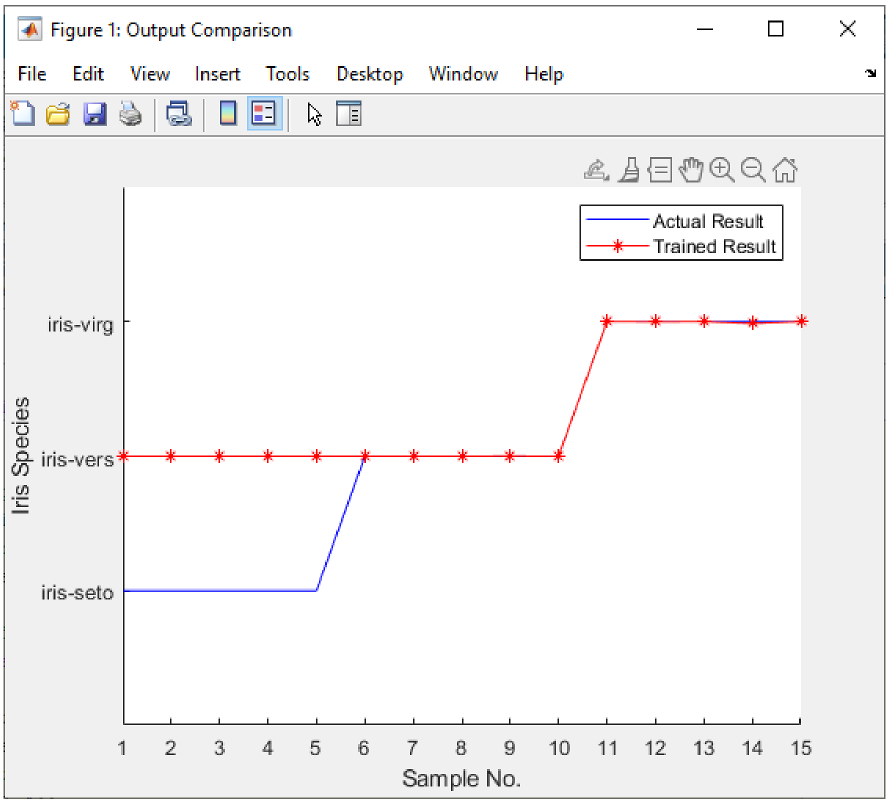


Figure 76: Case 10 Unrounded Simulated Output

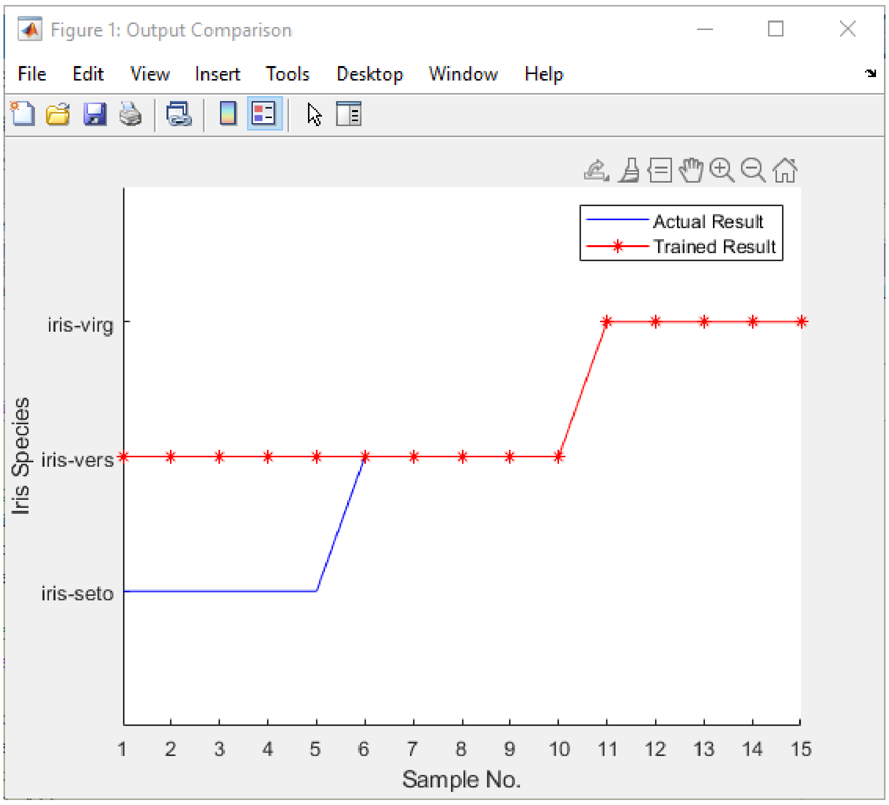


Figure 77: Case 10 Rounded Simulated Output