Neural Networks - Problem Statement

For Problem 1, implement the following system in Simulink and assume a damping ratio (zeta) of 0.5, a fixed step integration with a time step of 0.01 sec, and a unit step input occurring at 1 second, with a simulation stop time of 15 seconds. Generate a training dataset of (time, input, omega) as inputs and y as the output for values of omega 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, and 3.5 rad/sec.

$$y(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} u(s)$$

Problem 1) Neural Network Estimator – single variable (60 points)

a) Use the nnstart tool Fitting App to fit four, single hidden-layer networks with 10, 15, 20, and 25 nodes in the hidden layer, respectively. Save each as either an m-file function or Simulink block. Generate comparisons of the neural network fits with the training data. Make any observations about the size of the network and the quality of fit it produces.

b) Now test the fit model by testing the neural network output with the true system output for omega values 0.75, 1.25, 1.75, 2.25, 2.75, and 3.25 rad/sec. Make any observations about the size of the network and the quality of fit it produces for this test data or if any frequencies fit better than others.

c) Finally, comment on the overall ability of the neural network to work or not work for this application and the best size for the network, in your opinion.

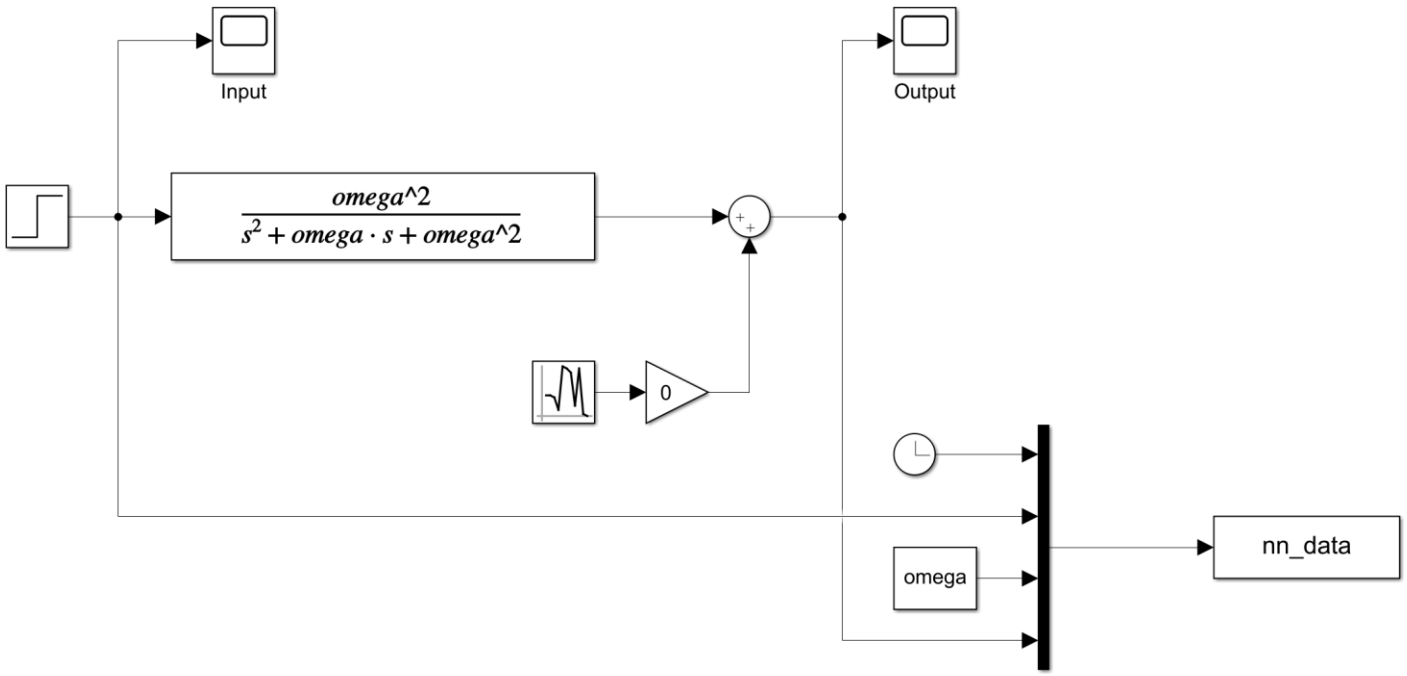
d) Let's see if the network can extrapolate. Set the natural frequency to 0.25 rad/sec, run the system, and compare the output of the true system with the output of your best neural network. Does the network do a good job when operating outside the training limits in this case? Repeat for a natural frequency of 3.75 rad/sec.

Problem 2) Neural Network Estimator – noise effects (40 points)

For Problem 2, repeat the first three steps (a through c but not d) of Problem 1 but add zero mean noise with variance 0.001 to the output data used for training. Comment on the effects of noise on the ability of the network to model these data.

Problem – 1:

a.

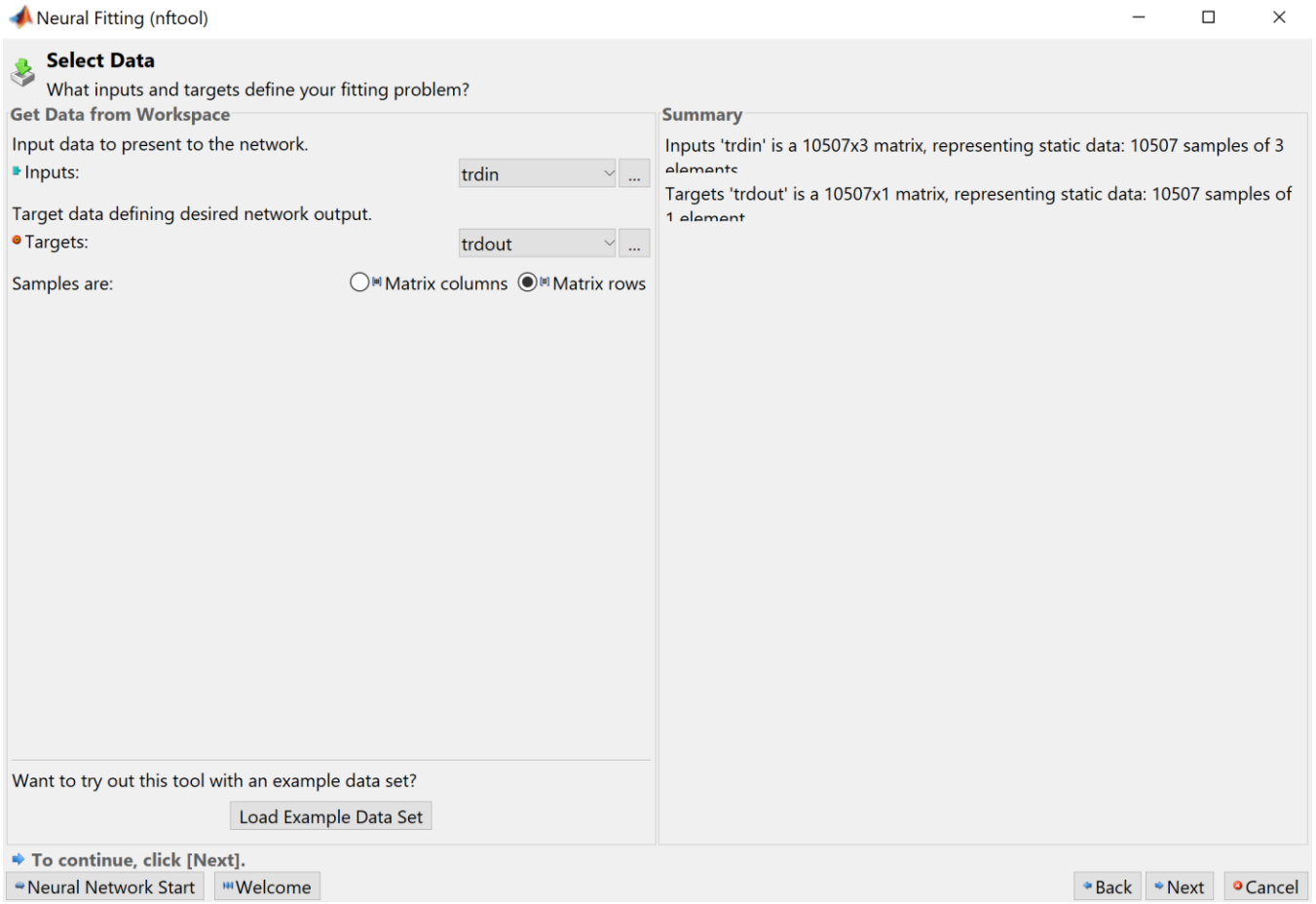


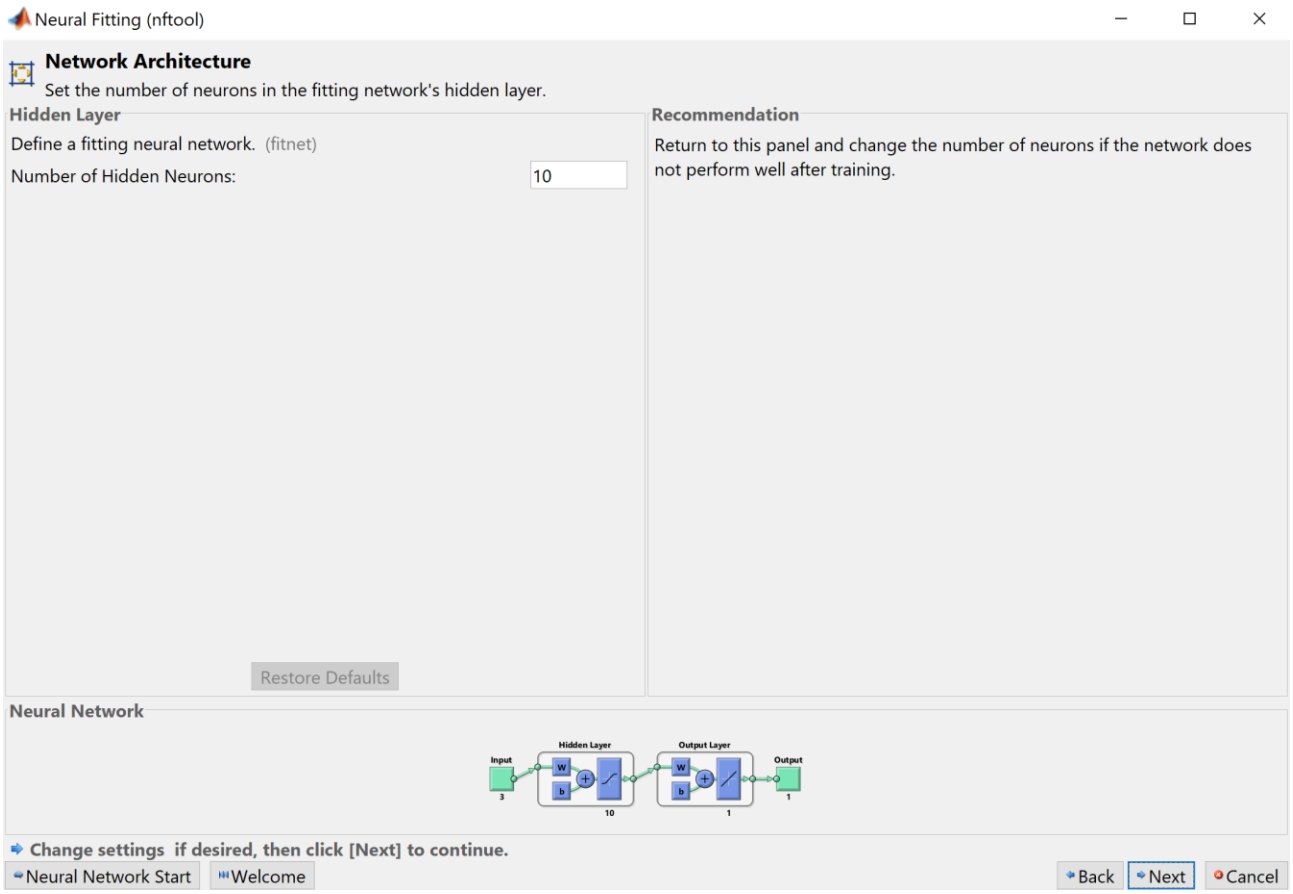
Simulink Diagram to generate training data

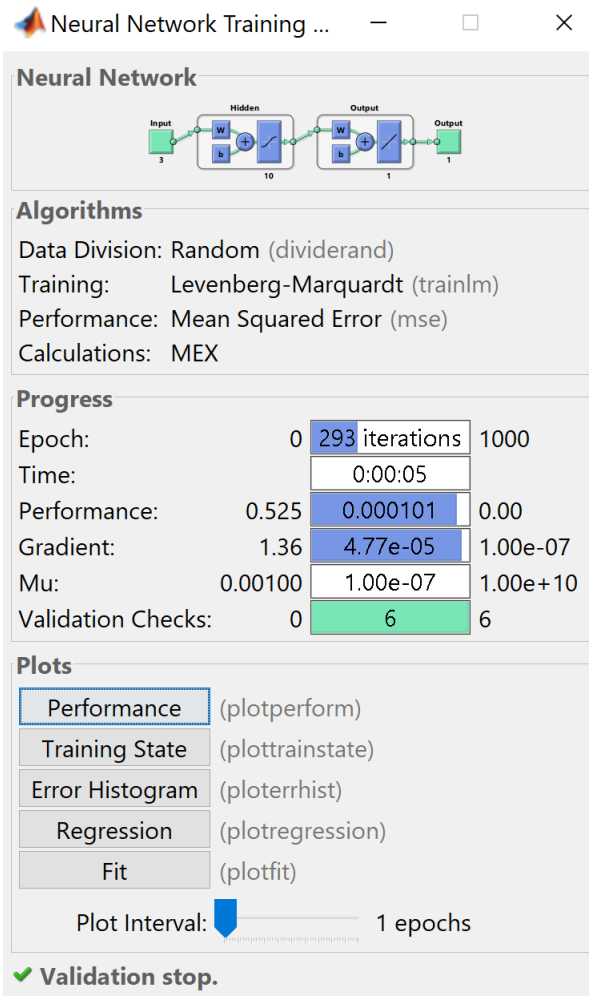
MATLAB Commands to arrange the data

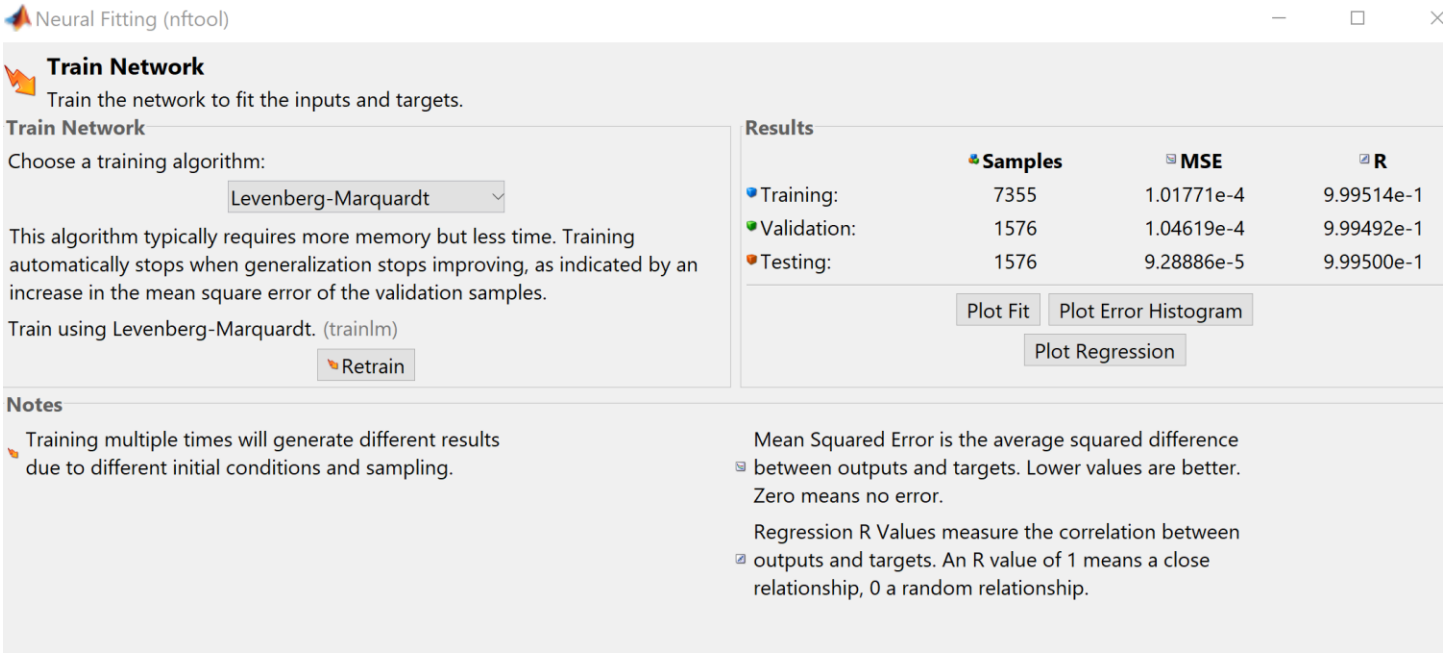
```
omega=0.5 %run the simulation every time a new omega is defined
trdata=nn_data
omega=1
trdata=[trdata;nn_data]
omega=1.5
trdata=[trdata;nn_data]
omega=2
trdata=[trdata;nn_data]
omega=2.5
trdata=[trdata;nn_data]
omega=3
trdata=[trdata;nn_data]
omega=3.5
trdata=[trdata;nn_data]

trdin=trdata(:,1:3)

trdout=trdata(:,4)
```

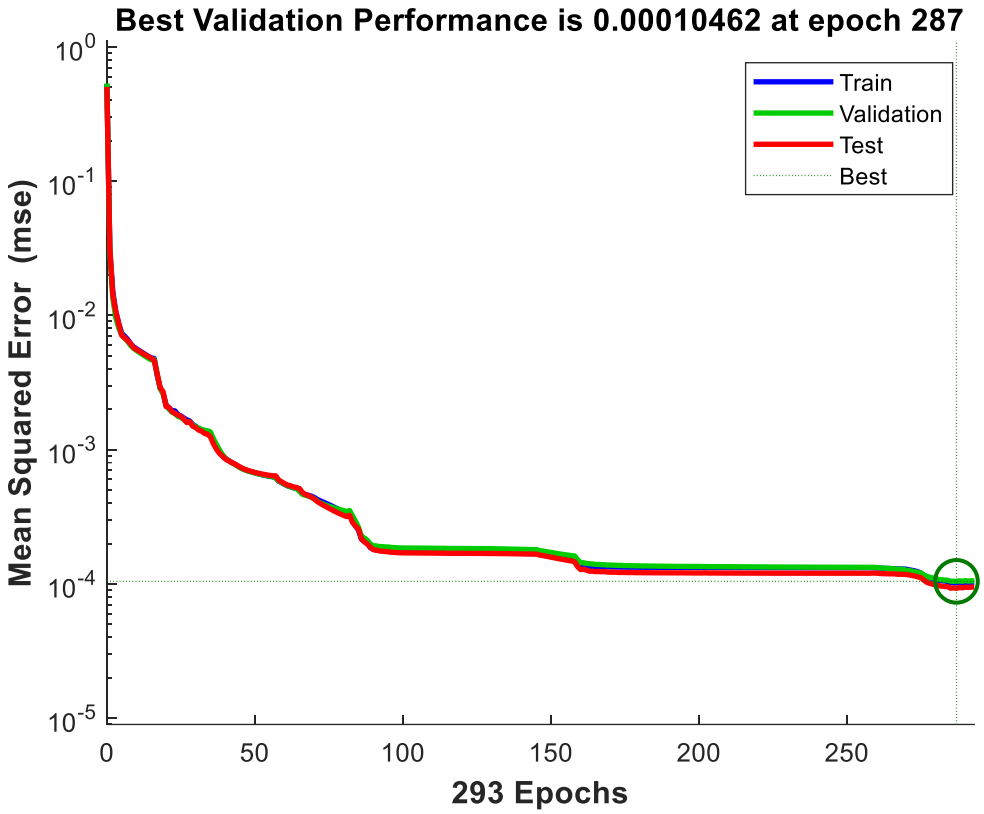
Fitting with 10 neurons



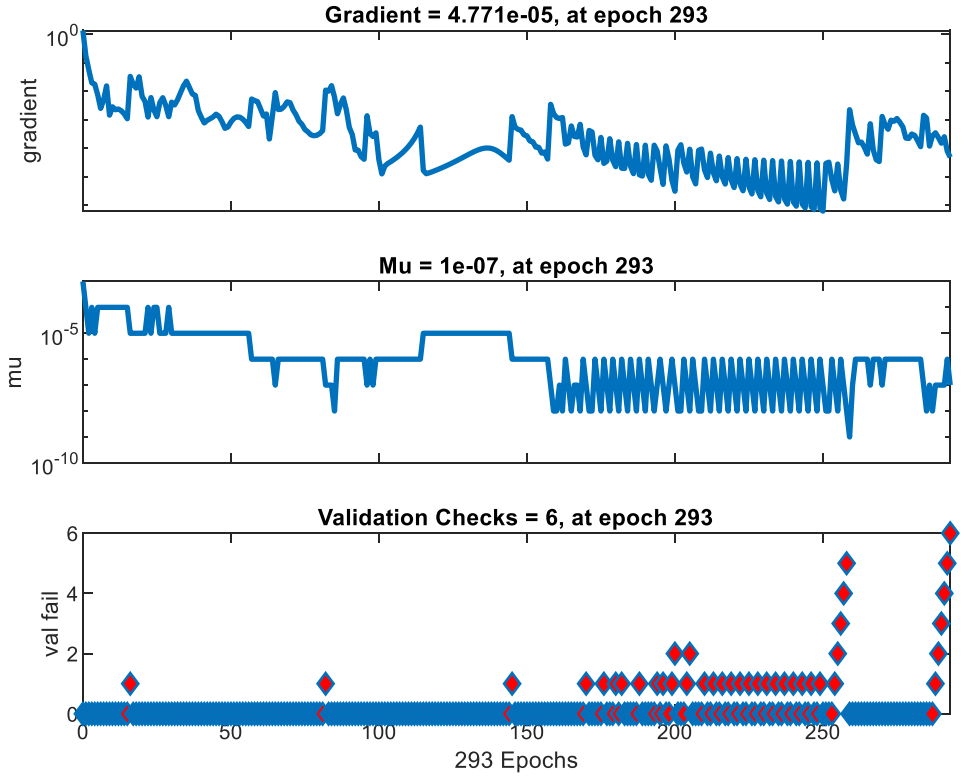


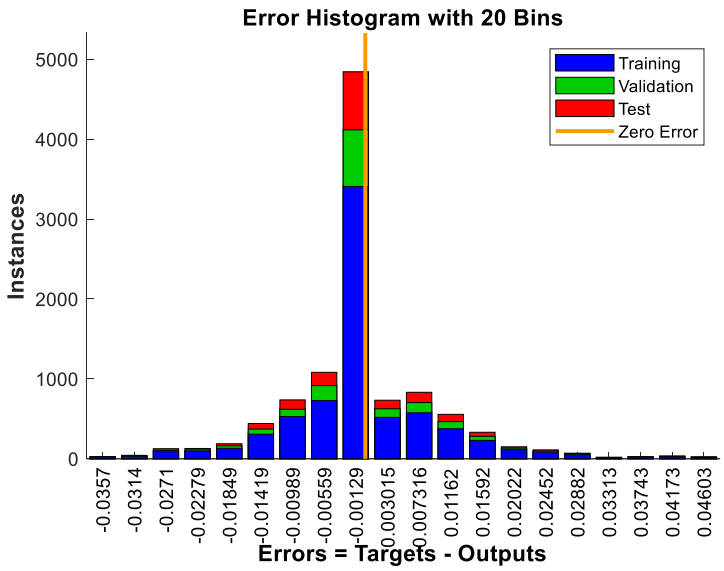

Neural Network Training ...
Neural Network
Input
3
Hidden
W
b
10
Output
W
b
1
Output
1
Algorithms
Data Division: Random (dividerand)
Training: Levenberg-Marquardt (trainlm)
Performance: Mean Squared Error (mse)
Calculations: MEX
Progress
Epoch: 0 293 iterations 1000
Time: 0:00:05
Performance: 0.525 0.000101 0.00
Gradient: 1.36 4.77e-05 1.00e-07
Mu: 0.00100 1.00e-07 1.00e+10
Validation Checks: 0 6 6
Plots
Performance (plotperform)
Training State (plottrainstate)
Error Histogram (ploterrhist)
Regression (plotregression)
Fit (plotfit)
Plot Interval: 1 epochs
Validation stop.

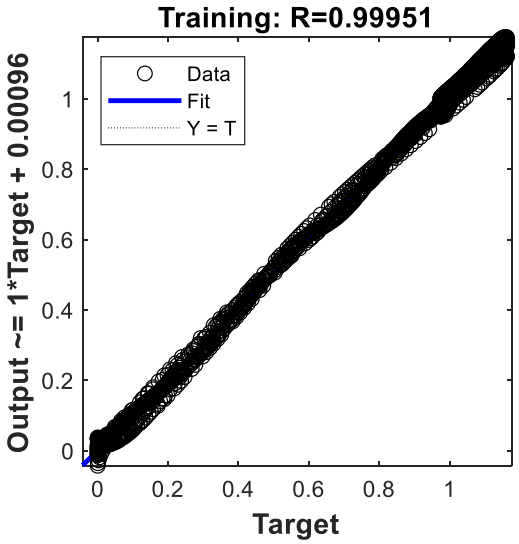

Neural Fitting (nftool)
Train Network
Train the network to fit the inputs and targets.
Train Network
Choose a training algorithm:
Levenberg-Marquardt
This algorithm typically requires more memory but less time. Training automatically stops when generalization stops improving, as indicated by an increase in the mean square error of the validation samples.
Train using Levenberg-Marquardt. (trainlm)
Retrain
Results
Samples MSE R
Training: 7355 1.01771e-4 9.99514e-1
Validation: 1576 1.04619e-4 9.99492e-1
Testing: 1576 9.28886e-5 9.99500e-1
Plot Fit
Plot Error Histogram
Plot Regression
Notes
Training multiple times will generate different results due to different initial conditions and sampling.
Mean Squared Error is the average squared difference between outputs and targets. Lower values are better. Zero means no error.
Regression R Values measure the correlation between outputs and targets. An R value of 1 means a close relationship, 0 a random relationship.

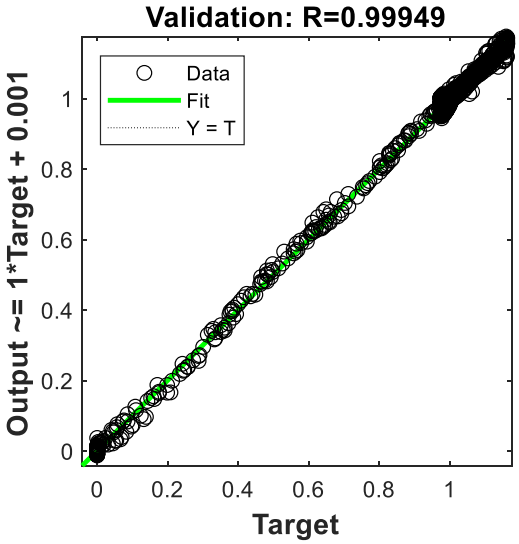


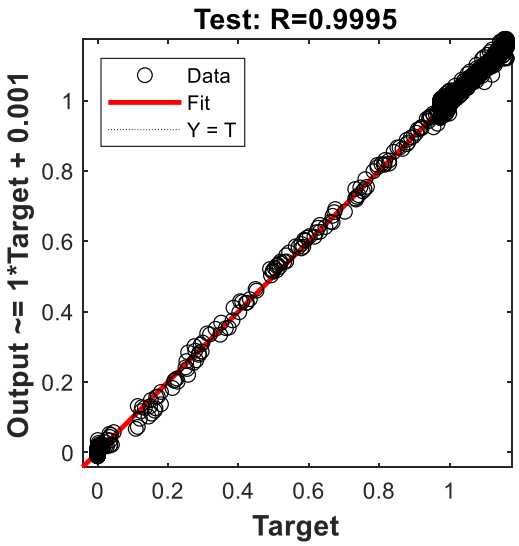
Training State

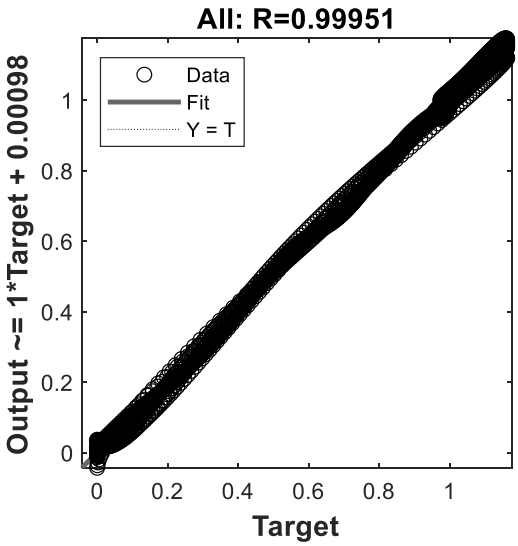


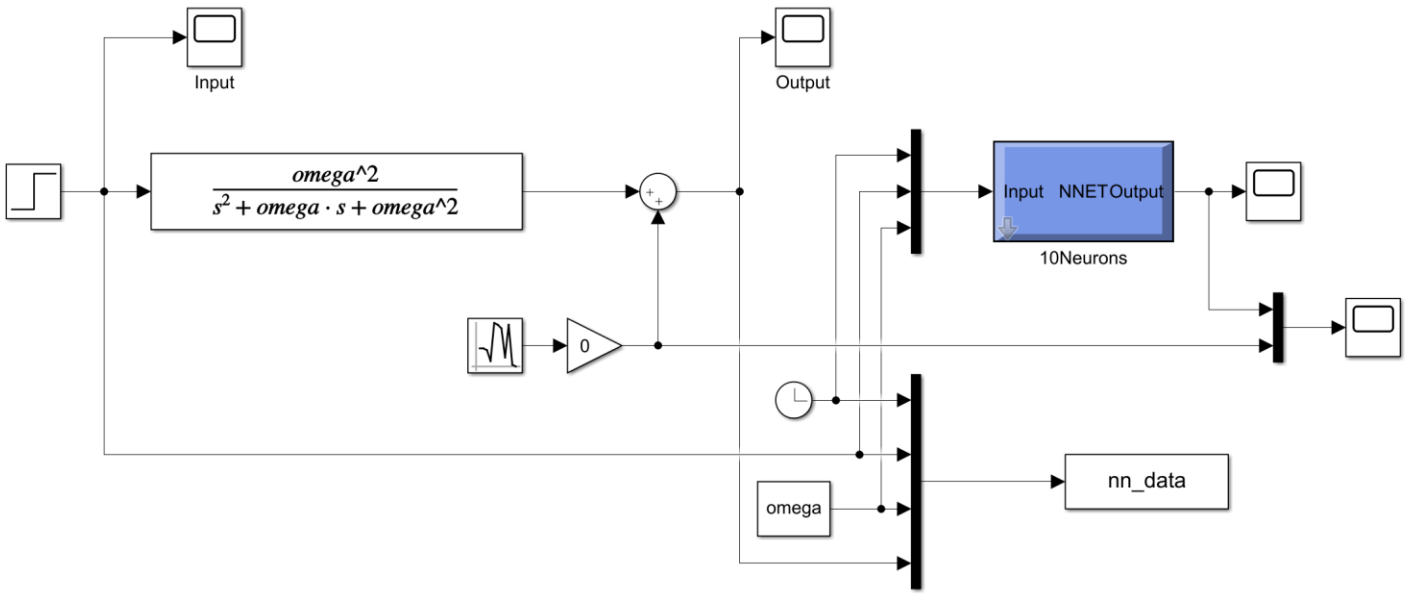

Error Histogram with 20 Bins
Training
Validation
Test
Zero Error
Instances
5000
4000
3000
2000
1000
0
-0.0357
-0.0314
-0.0271
-0.02279
-0.01849
-0.01419
-0.00989
-0.00559
-0.00129
0.003015
0.007316
0.01162
0.01592
0.02022
0.02452
0.02882
0.03313
0.03743
0.04173
0.04603
Errors = Targets - Outputs


Training: R=0.99951
Data
Fit
Y = T
Output ~= 1*Target + 0.00096
Target


Validation: R=0.99949
Data
Fit
Y = T
Output ~= 1*Target + 0.001
Target

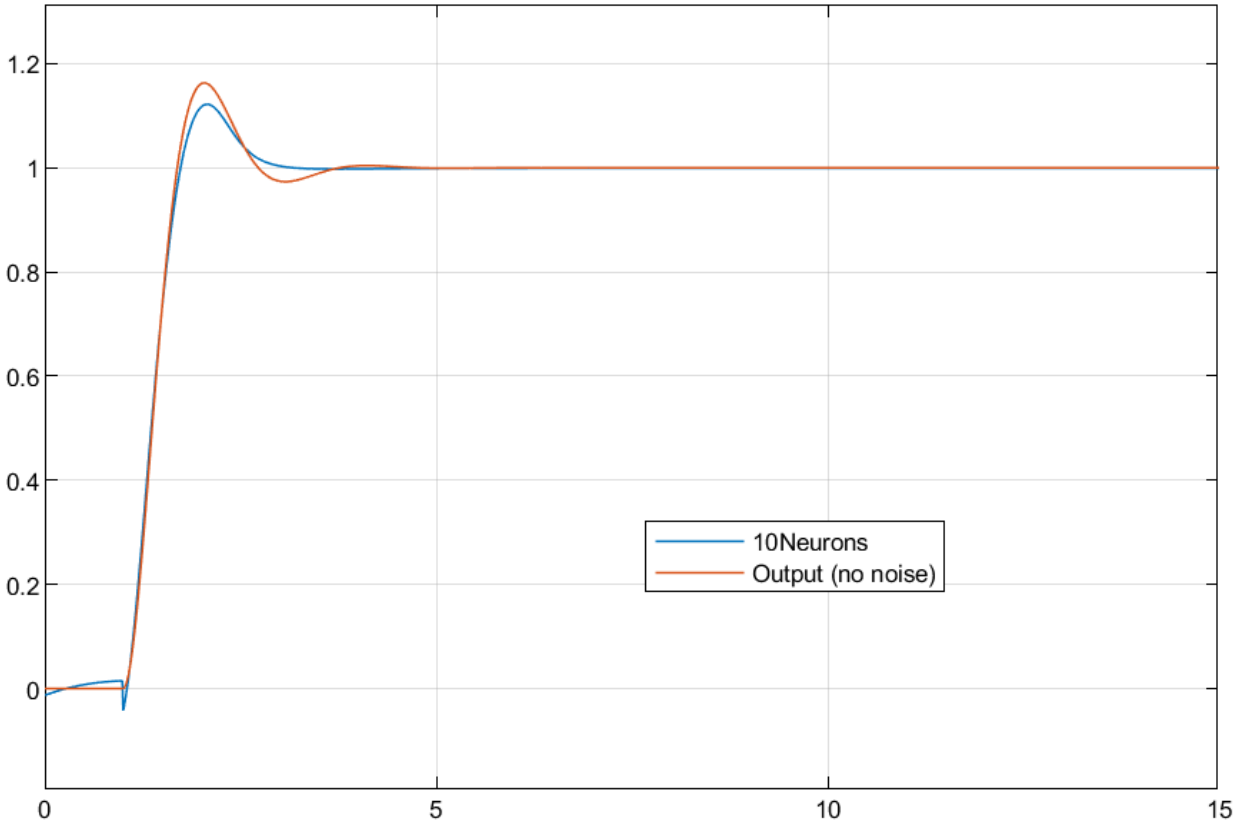

Test: R=0.9995
Data
Fit
Y = T
Output ~= 1*Target + 0.001
Target


All: R=0.99951
Data
Fit
Y = T
Output ~= 1*Target + 0.00098
Target

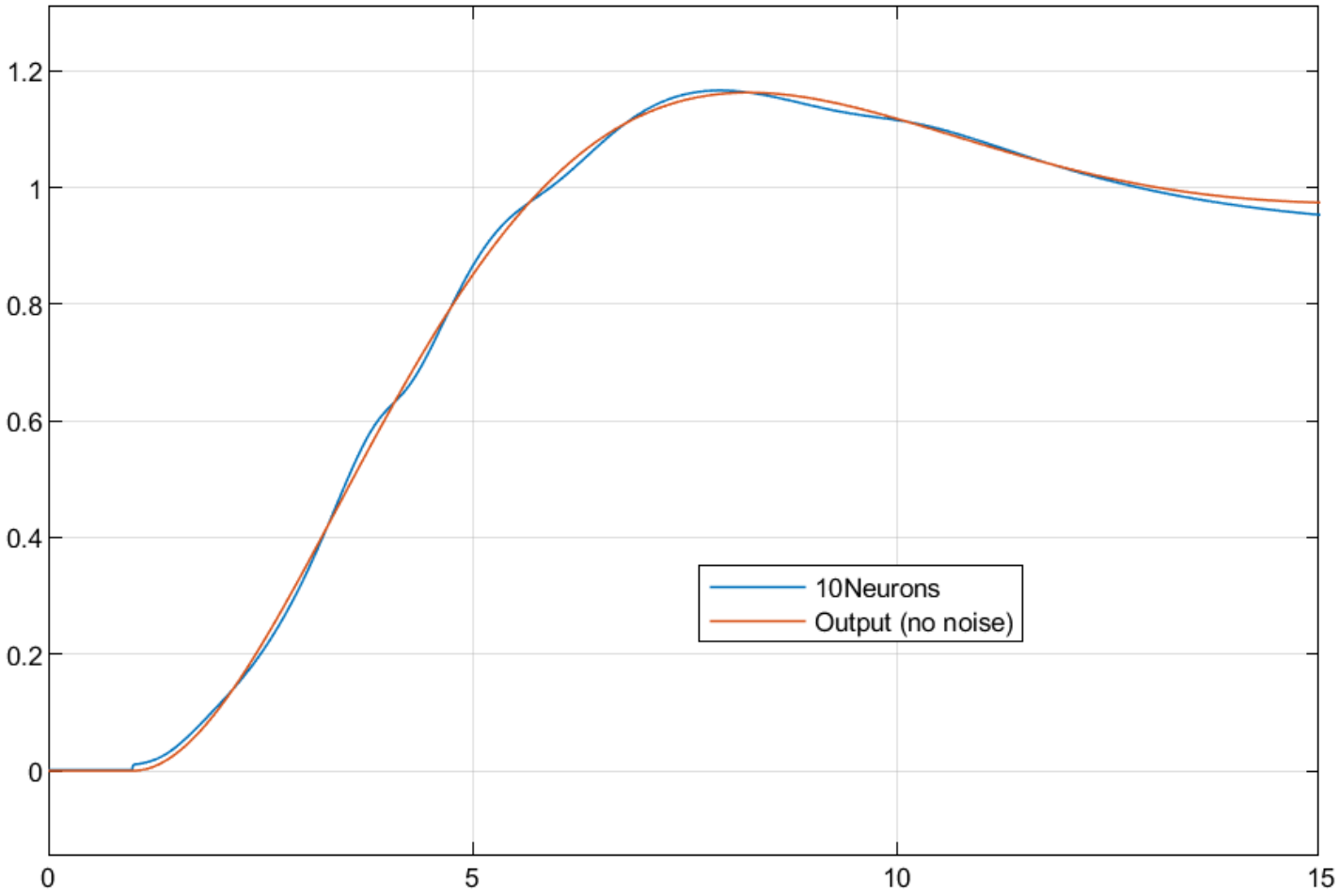


Simulink Diagram

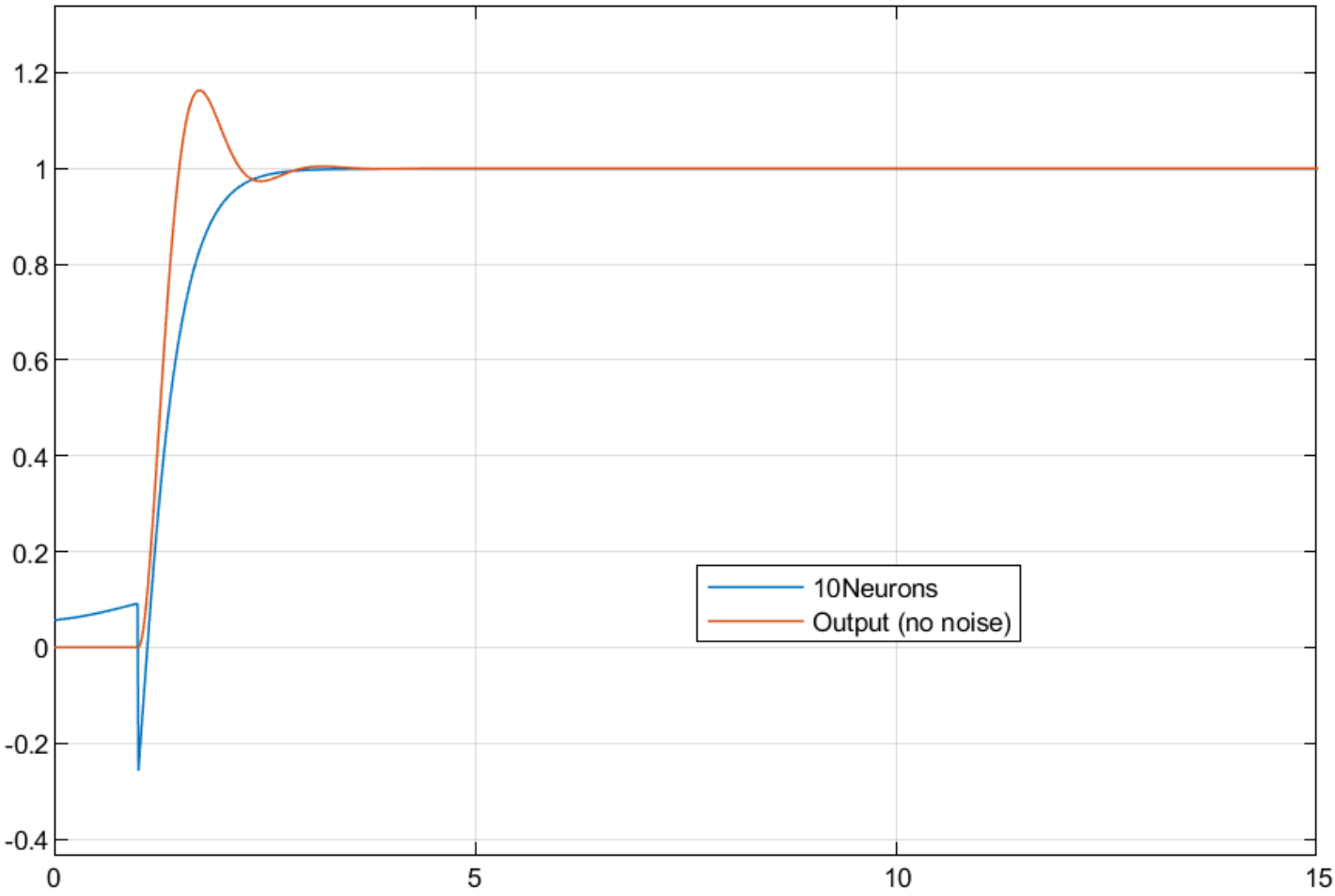
Output comparison



It is observed that the neural network approximation is slightly different at the start. The value of omega is at the last defined value of 3.5. On running the simulation again with omega = 0.5, the following graph is obtained

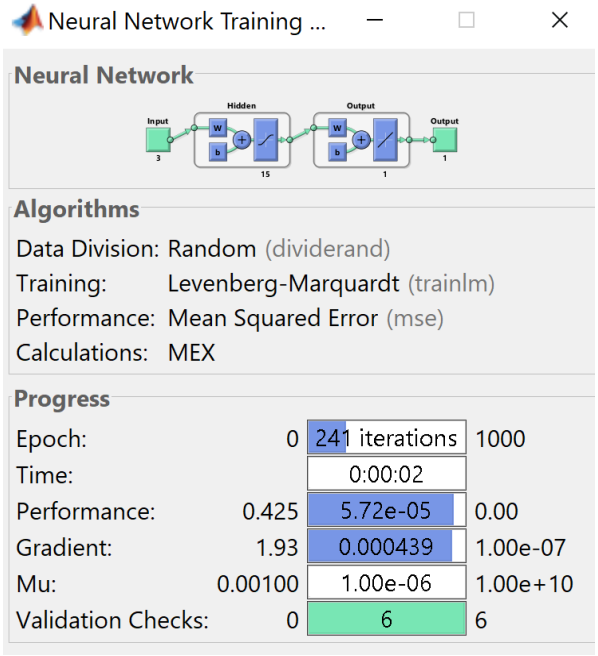


Although the mismatch is still present, we can see that the approximation is very good. Now checking the algorithm for extrapolation, we define omega as 5 which is far from all training data.



It is observed that the transition state outputs are significantly different although they converge in around 3 seconds which is also the settling time for the system.

Training with 15 neurons



Neural Fitting (nftool)

Train Network

Train the network to fit the inputs and targets.

Train Network

Choose a training algorithm:

Levenberg-Marquardt

This algorithm typically requires more memory but less time. Training automatically stops when generalization stops improving, as indicated by an increase in the mean square error of the validation samples.

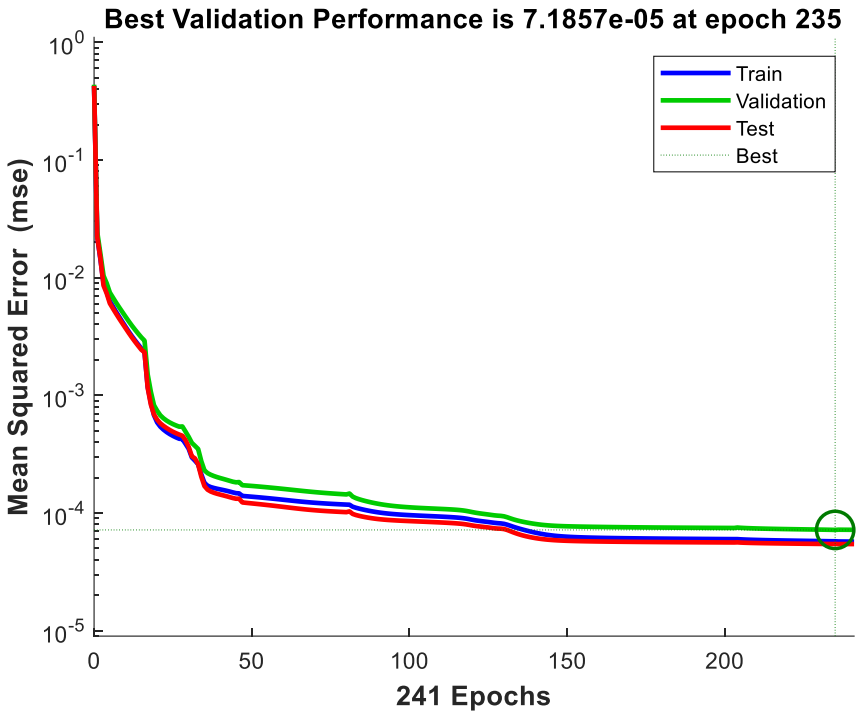
Train using Levenberg-Marquardt. (trainlm)

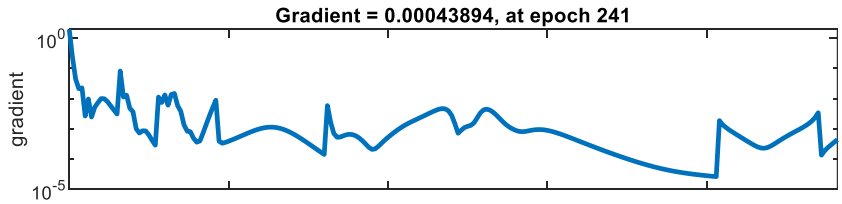
Retrain

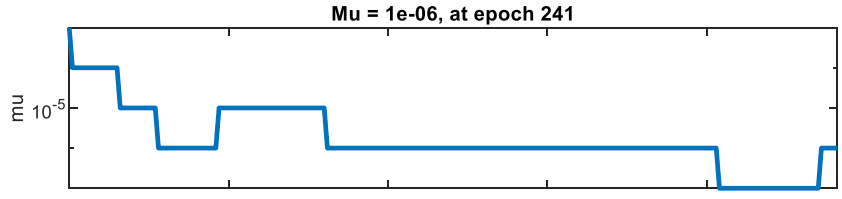
Results

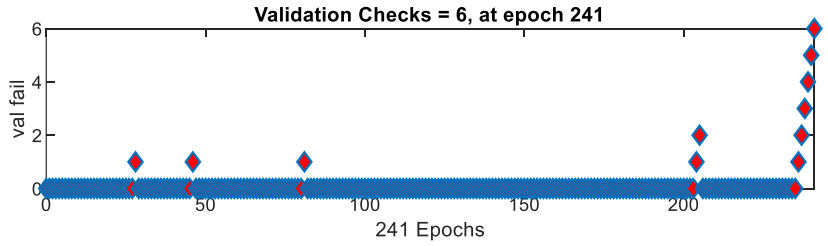
	Samples	MSE	R
Training:	7355	5.75615e-5	9.99717e-1
Validation:	1576	7.18570e-5	9.99684e-1
Testing:	1576	5.46131e-5	9.99721e-1

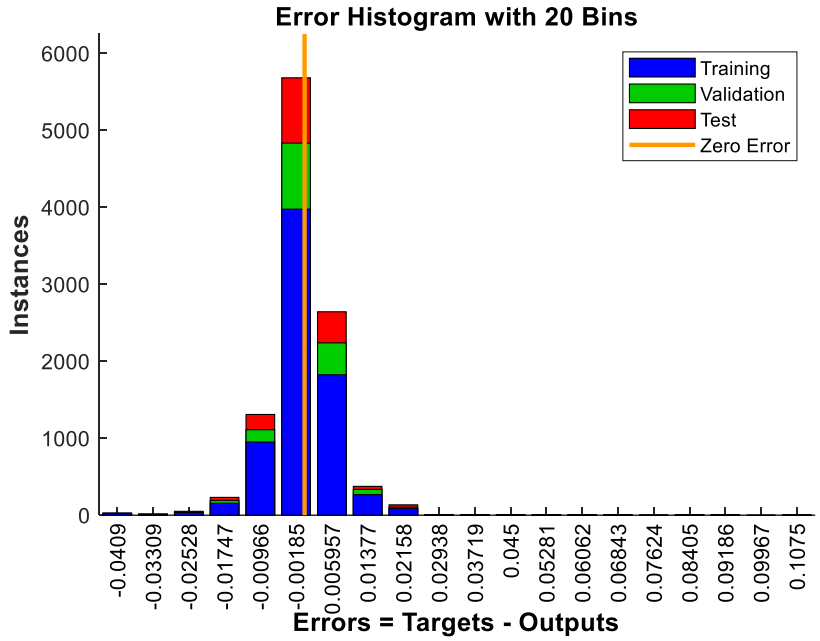
Plot Fit | Plot Error Histogram | Plot Regression

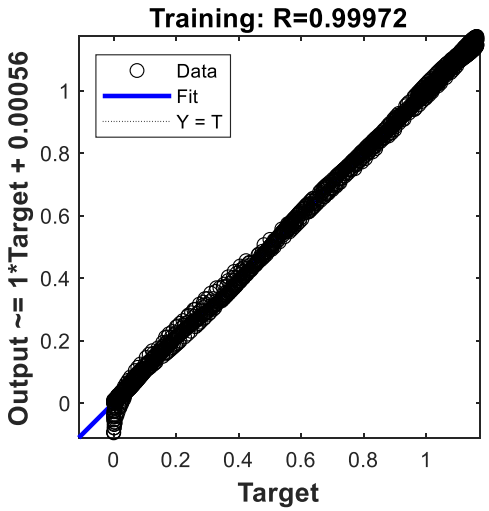


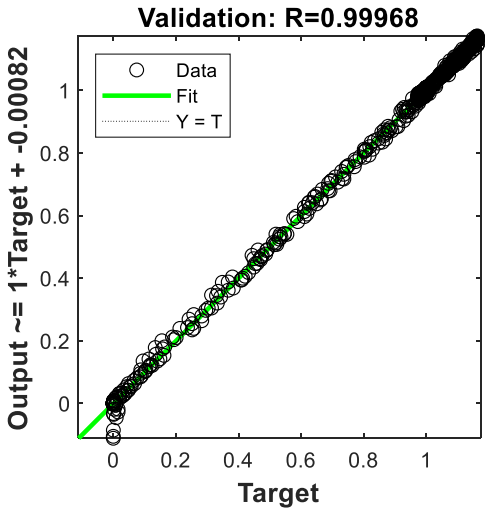


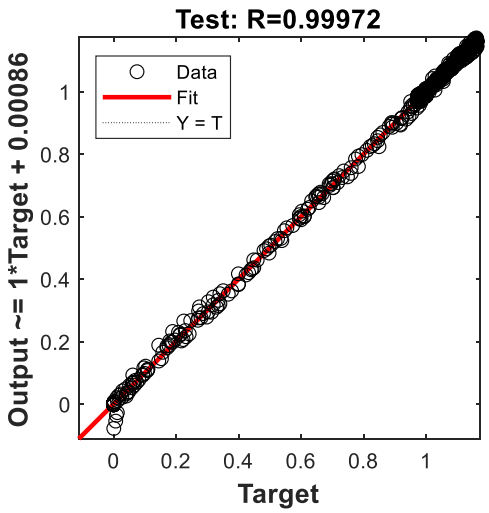


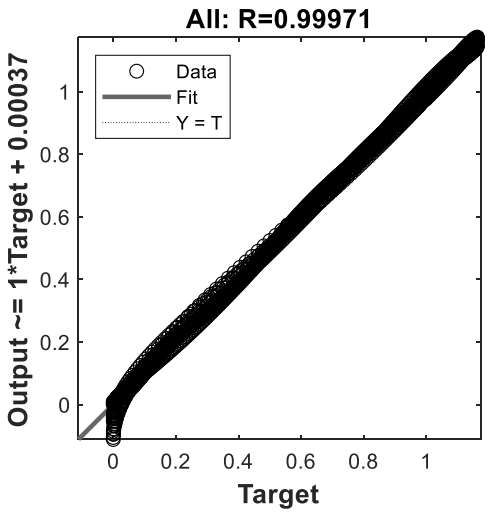


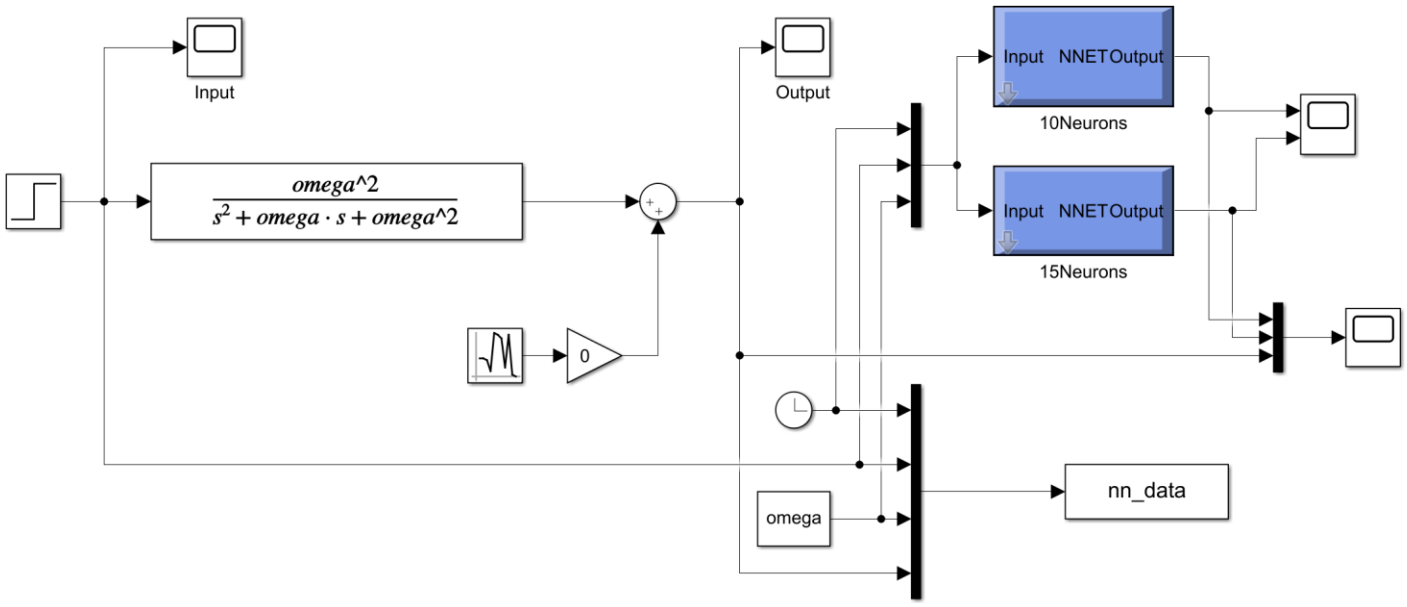

Error Histogram with 20 Bins
Training
Validation
Test
Zero Error
Instances
6000
5000
4000
3000
2000
1000
0
-0.0409
-0.03309
-0.02528
-0.01747
-0.00966
-0.00185
0.005957
0.01377
0.02158
0.02938
0.03719
0.045
0.05281
0.06062
0.06843
0.07624
0.08405
0.09186
0.09967
0.1075
Errors = Targets - Outputs


Training: R=0.99972
Data
Fit
Y = T
Output ~= 1*Target + 0.00056
Target


Validation: R=0.99968
Data
Fit
Y = T
Output ~= 1*Target + -0.00082
Target

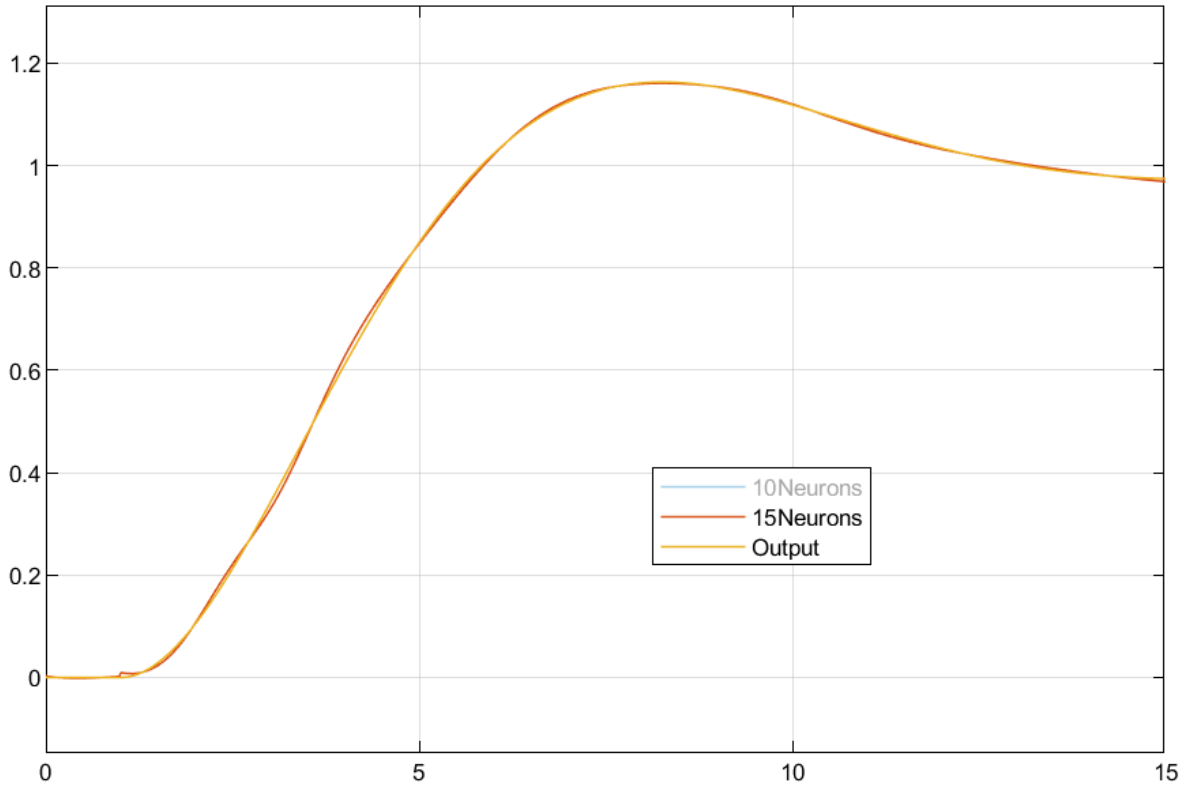

Test: R=0.99972
Data
Fit
Y = T
Output ~= 1*Target + 0.00086
Target


All: R=0.99971
Data
Fit
Y = T
Output ~= 1*Target + 0.00037
Target

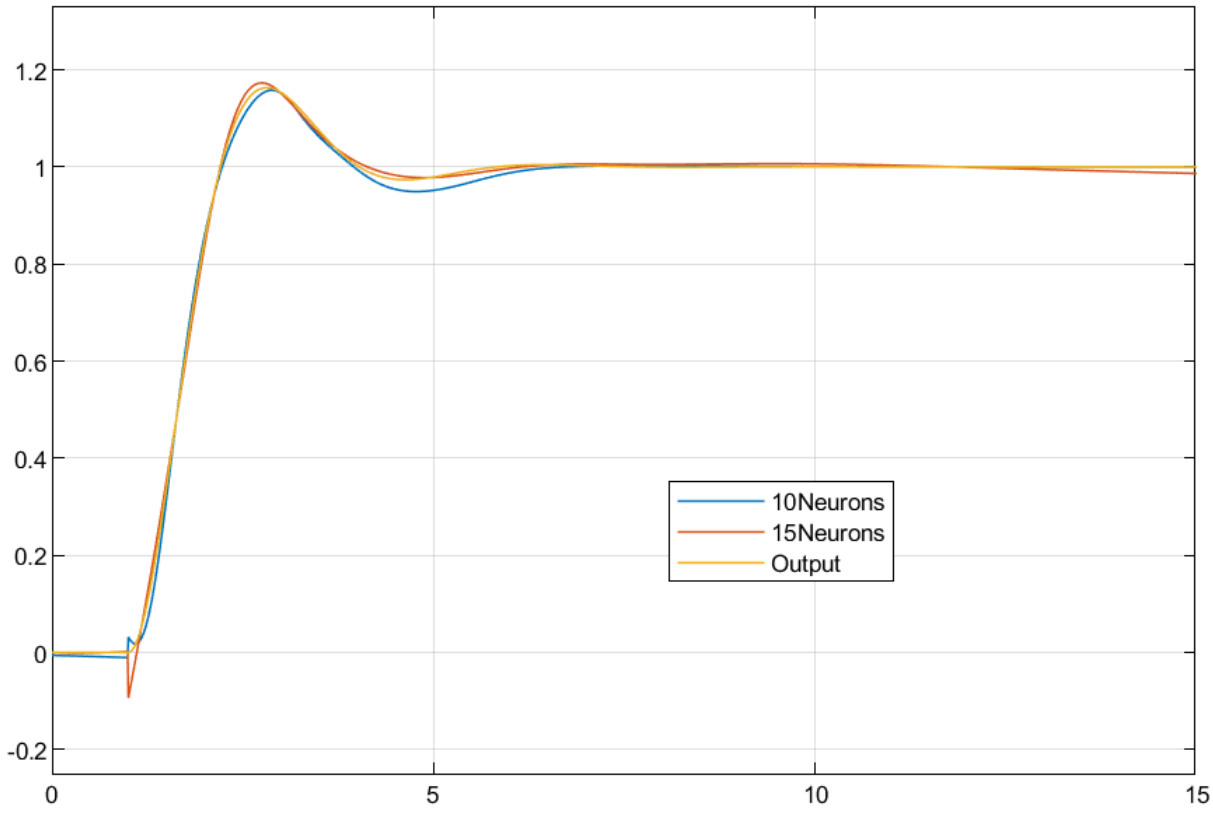


Simulink Diagram

Output Comparison

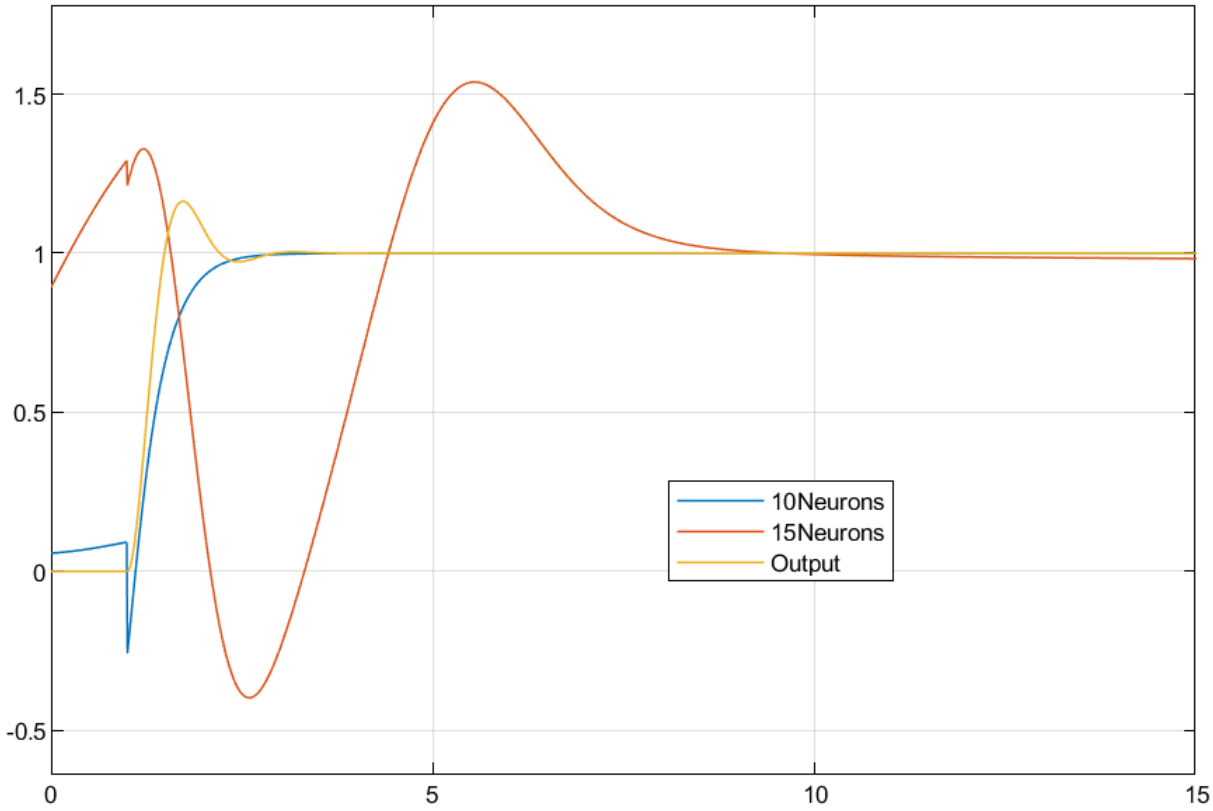


Omega = 0.5



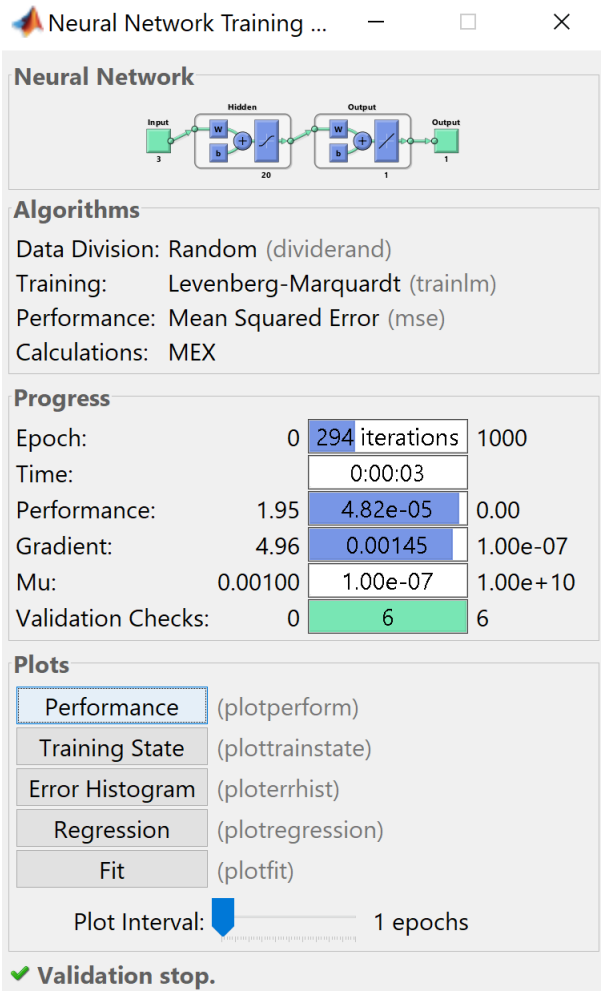
Plot above Omega = 2

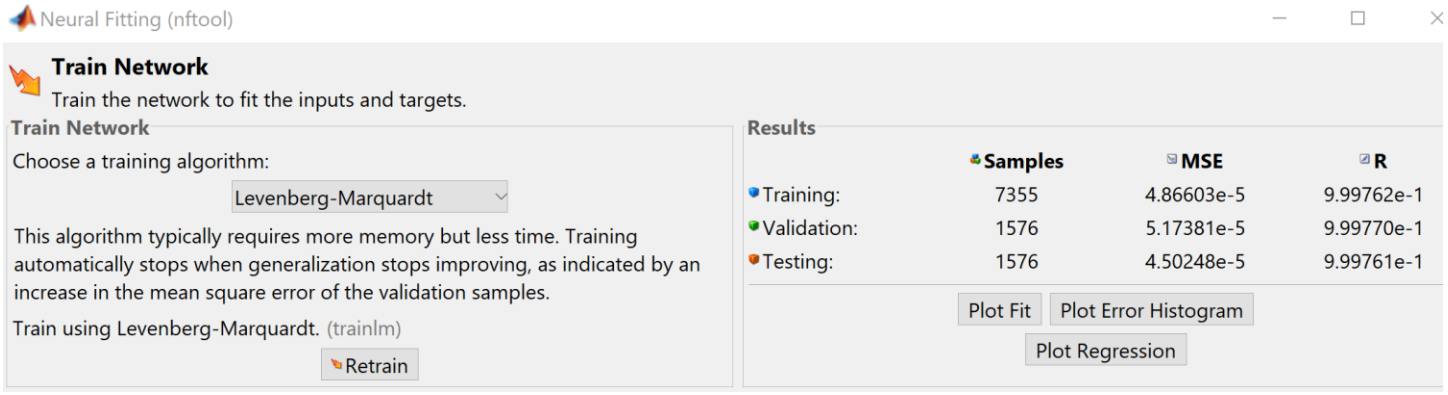
Plot below Omega = 5

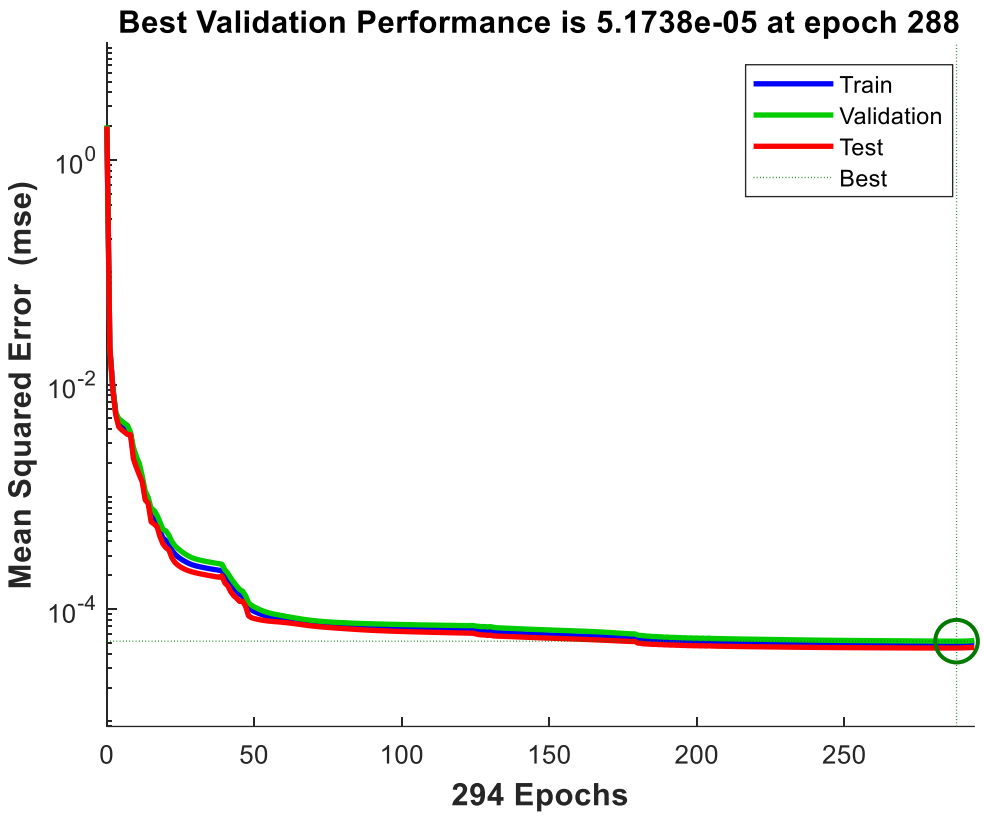


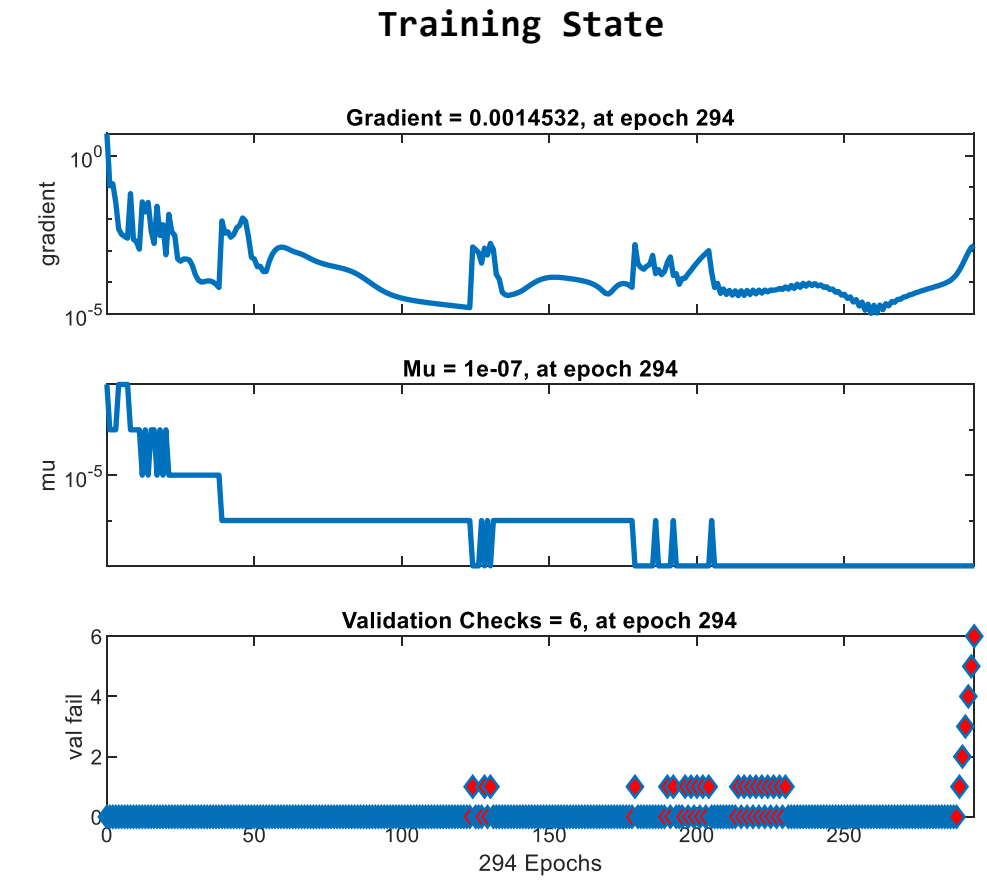
Initial mismatches have increased and the neural network trained with 15 neurons is less efficient at extrapolating omega values as compared to the previous one with 10 neurons.

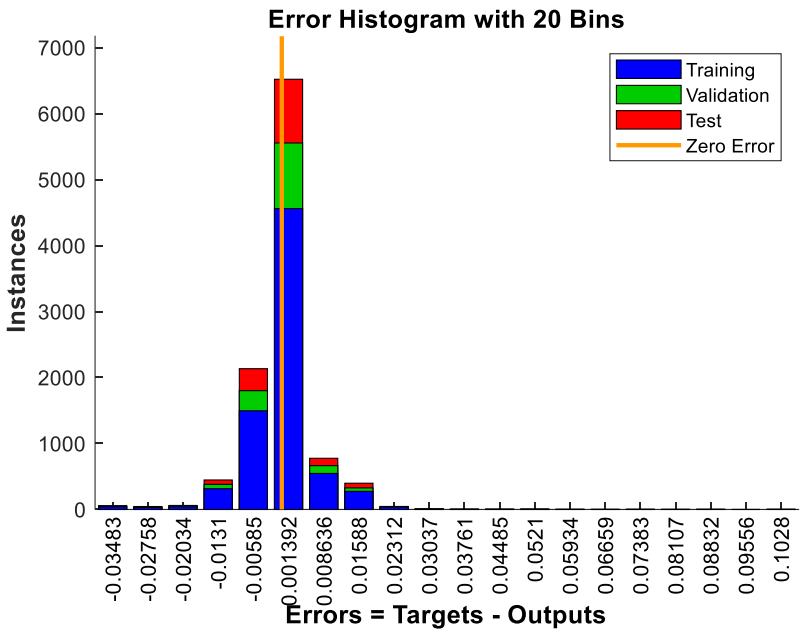
Training with 20 neurons

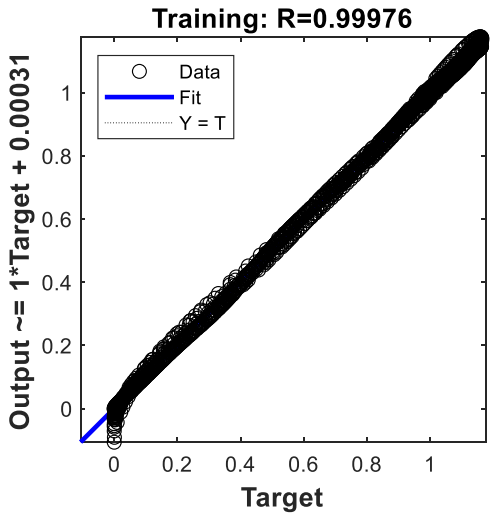


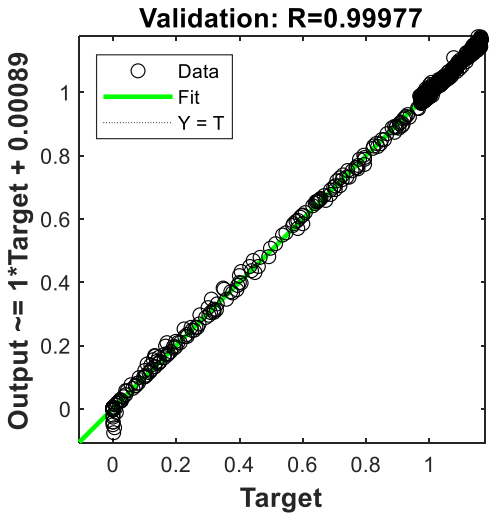


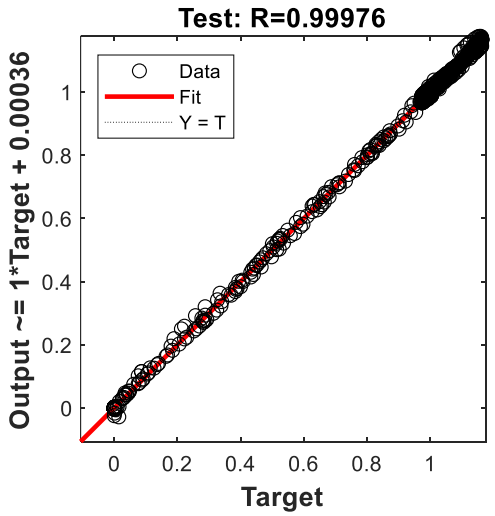

Best Validation Performance is 5.1738e-05 at epoch 288
Train
Validation
Test
Best
Mean Squared Error (mse)
10^{0}
10^{-2}
10^{-4}
0
50
100
150
200
250
294 Epochs

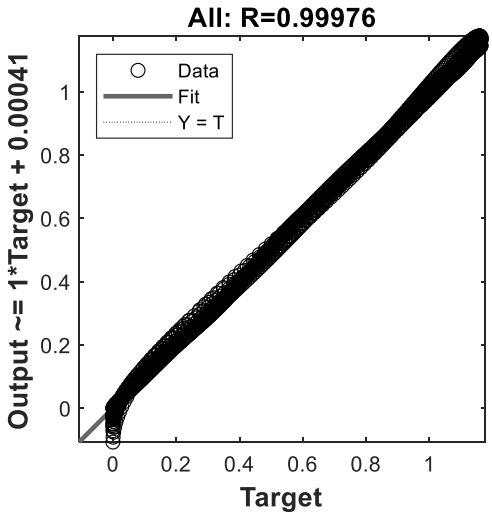

Training State
Gradient = 0.0014532, at epoch 294
gradient
10^{0}
10^{-5}
Mu = 1e-07, at epoch 294
mu
10^{-5}
Validation Checks = 6, at epoch 294
val fail
6
4
2
0
0
50
100
150
200
250
294 Epochs


Error Histogram with 20 Bins
Training
Validation
Test
Zero Error
Instances
7000
6000
5000
4000
3000
2000
1000
0
-0.03483
-0.02758
-0.02034
-0.0131
-0.00585
0.001392
0.008636
0.01588
0.02312
0.03037
0.03761
0.04485
0.0521
0.05934
0.06659
0.07383
0.08107
0.08832
0.09556
0.1028
Errors = Targets - Outputs


Training: R=0.99976
Data
Fit
Y = T
Output ~= 1*Target + 0.00031
Target
0
0.2
0.4
0.6
0.8
1


Validation: R=0.99977
Data
Fit
Y = T
Output ~= 1*Target + 0.00089
Target
0
0.2
0.4
0.6
0.8
1


Test: R=0.99976
Data
Fit
Y = T
Output ~= 1*Target + 0.00036
Target
0
0.2
0.4
0.6
0.8
1


All: R=0.99976
Data
Fit
Y = T
Output ~= 1*Target + 0.00041
Target
0
0.2
0.4
0.6
0.8
1

Training with 25 neurons

Neural Network Training ...

Neural Network

Algorithms

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

Progress

Epoch:	0	529 iterations	1000
Time:		0:00:08	
Performance:	4.55	2.06e-05	0.00
Gradient:	6.29	2.50e-05	1.00e-07
Mu:	0.00100	1.00e-07	1.00e+10
Validation Checks:	0	6	6

Plots

Performance (plotperform)
Training State (plottrainstate)
Error Histogram (ploterrhist)
Regression (plotregression)
Fit (plotfit)

Plot Interval: 1 epochs

Neural Fitting (nftool)

Train Network

Train the network to fit the inputs and targets.

Train Network

Choose a training algorithm:

Levenberg-Marquardt

This algorithm typically requires more memory but less time. Training automatically stops when generalization stops improving, as indicated by an increase in the mean square error of the validation samples.

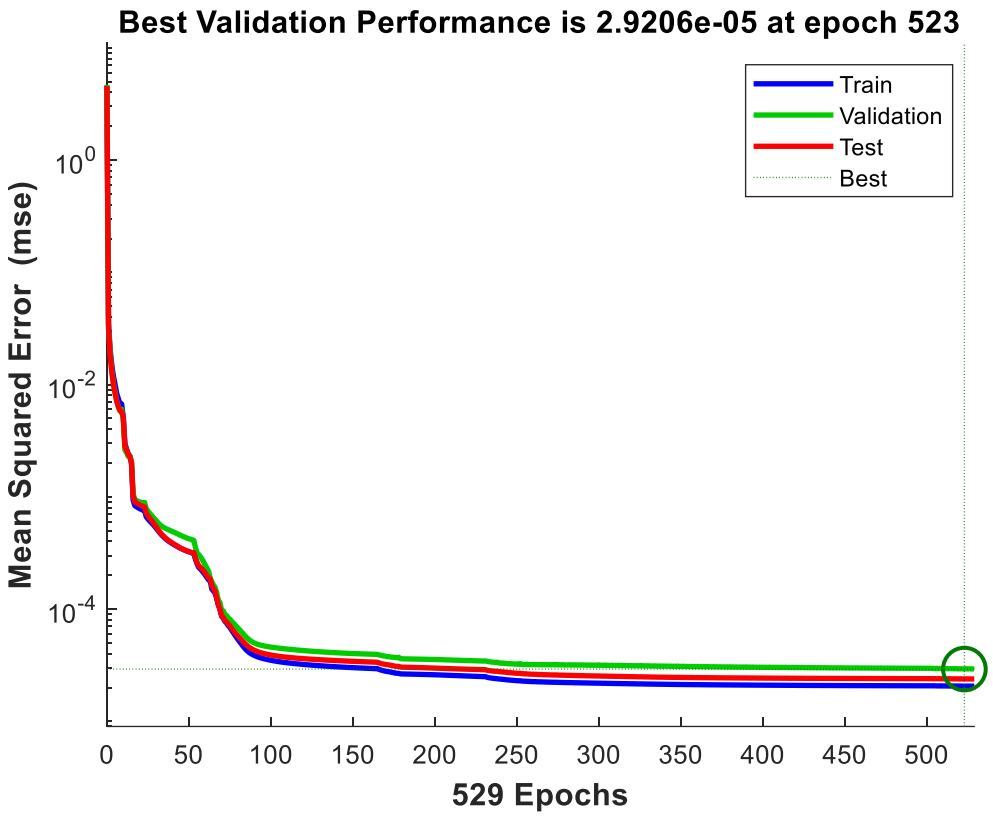
Train using Levenberg-Marquardt. (trainlm)

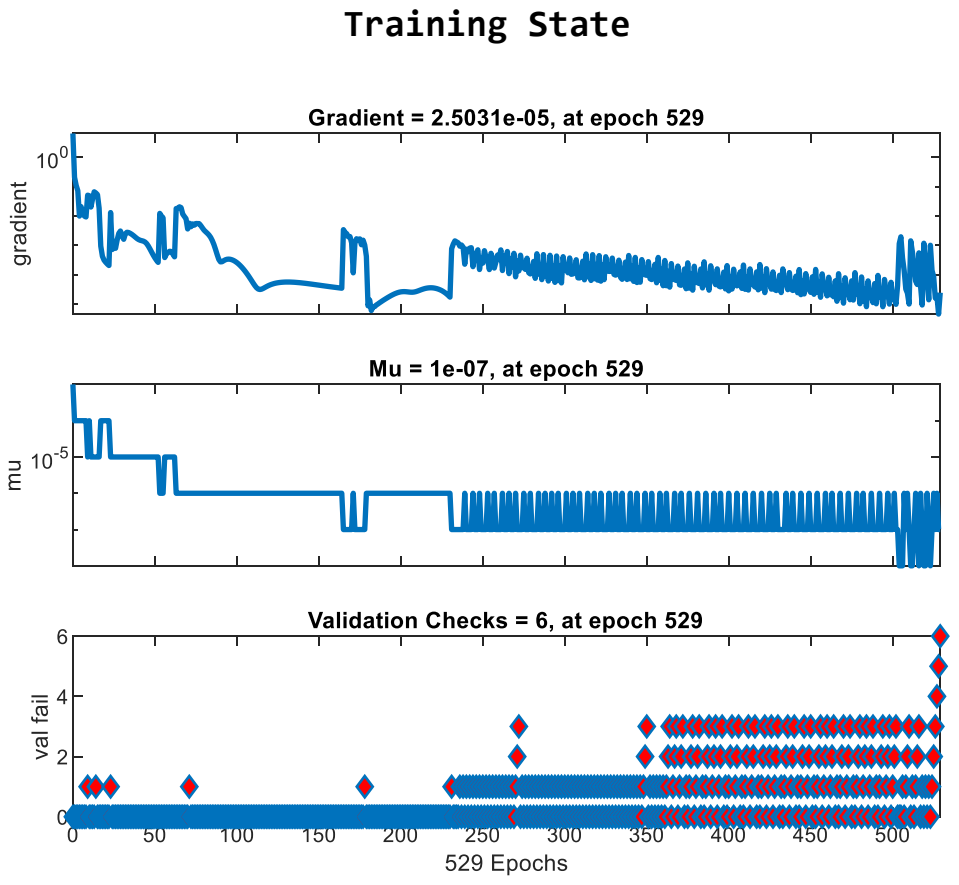
Retrain

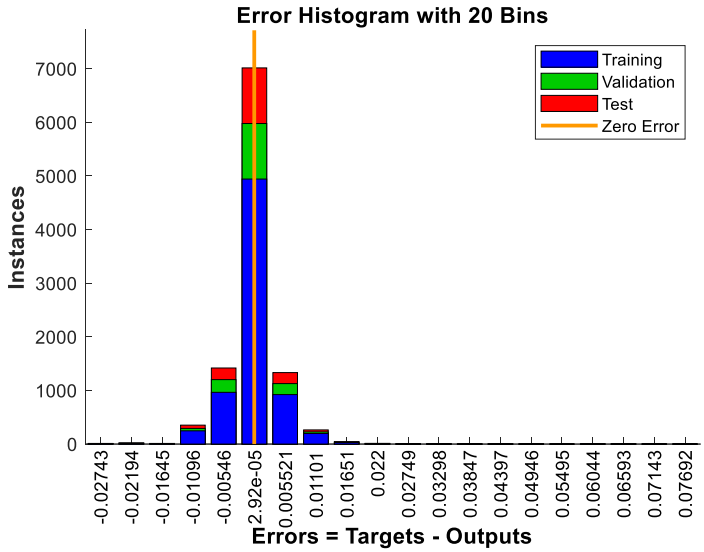
Results

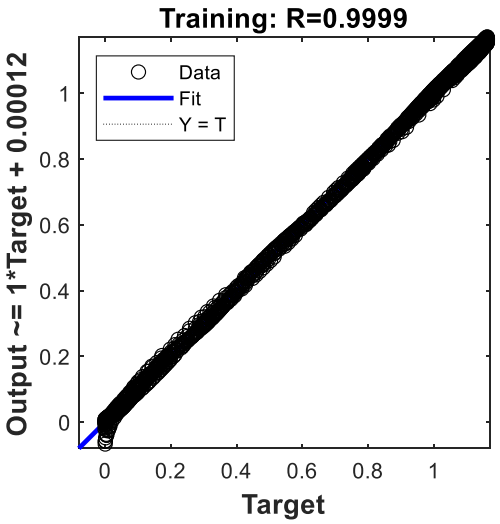
	Samples	MSE	R
Training:	7355	2.05856e-5	9.99900e-1
Validation:	1576	2.92058e-5	9.99860e-1
Testing:	1576	2.38380e-5	9.99878e-1

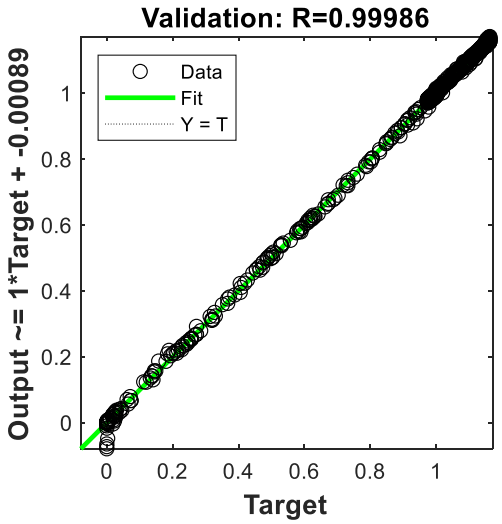
Plot Fit | Plot Error Histogram | Plot Regression

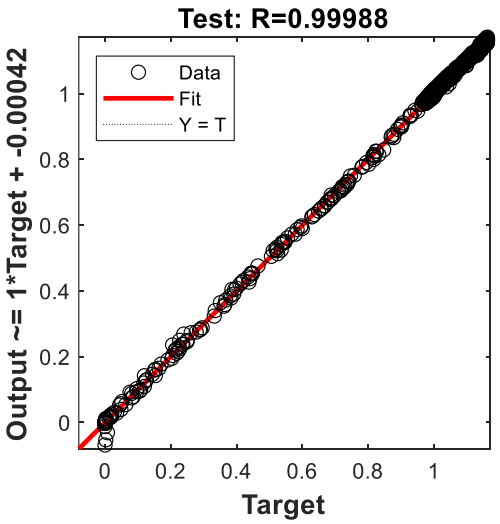

Best Validation Performance is 2.9206e-05 at epoch 523
Train
Validation
Test
Best
Mean Squared Error (mse)
10^{0}
10^{-2}
10^{-4}
0
50
100
150
200
250
300
350
400
450
500
529 Epochs

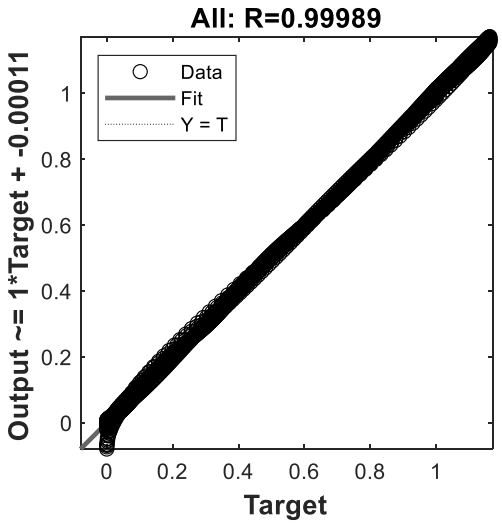

Training State
Gradient = 2.5031e-05, at epoch 529
gradient
10^{0}
Mu = 1e-07, at epoch 529
mu
10^{-5}
Validation Checks = 6, at epoch 529
val fail
6
4
2
0
0
50
100
150
200
250
300
350
400
450
500
529 Epochs


Error Histogram with 20 Bins
Training
Validation
Test
Zero Error
Instances
7000
6000
5000
4000
3000
2000
1000
0
-0.02743
-0.02194
-0.01645
-0.01096
-0.00546
2.92e-05
0.005521
0.01101
0.01651
0.022
0.02749
0.03298
0.03847
0.04397
0.04946
0.05495
0.06044
0.06593
0.07143
0.07692
Errors = Targets - Outputs

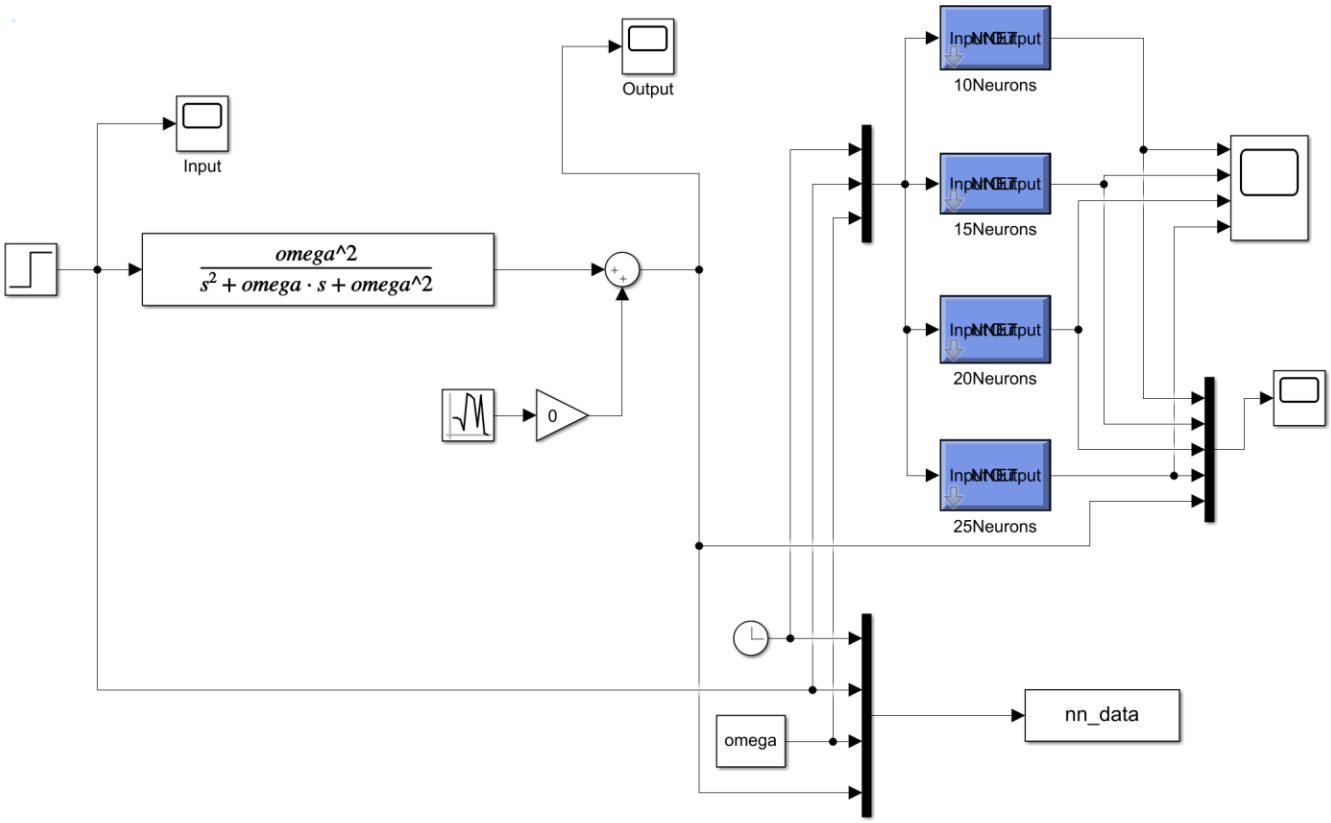

Training: R=0.9999
Data
Fit
Y = T
Output ~= 1*Target + 0.00012
Target


Validation: R=0.99986
Data
Fit
Y = T
Output ~= 1*Target + -0.00089
Target

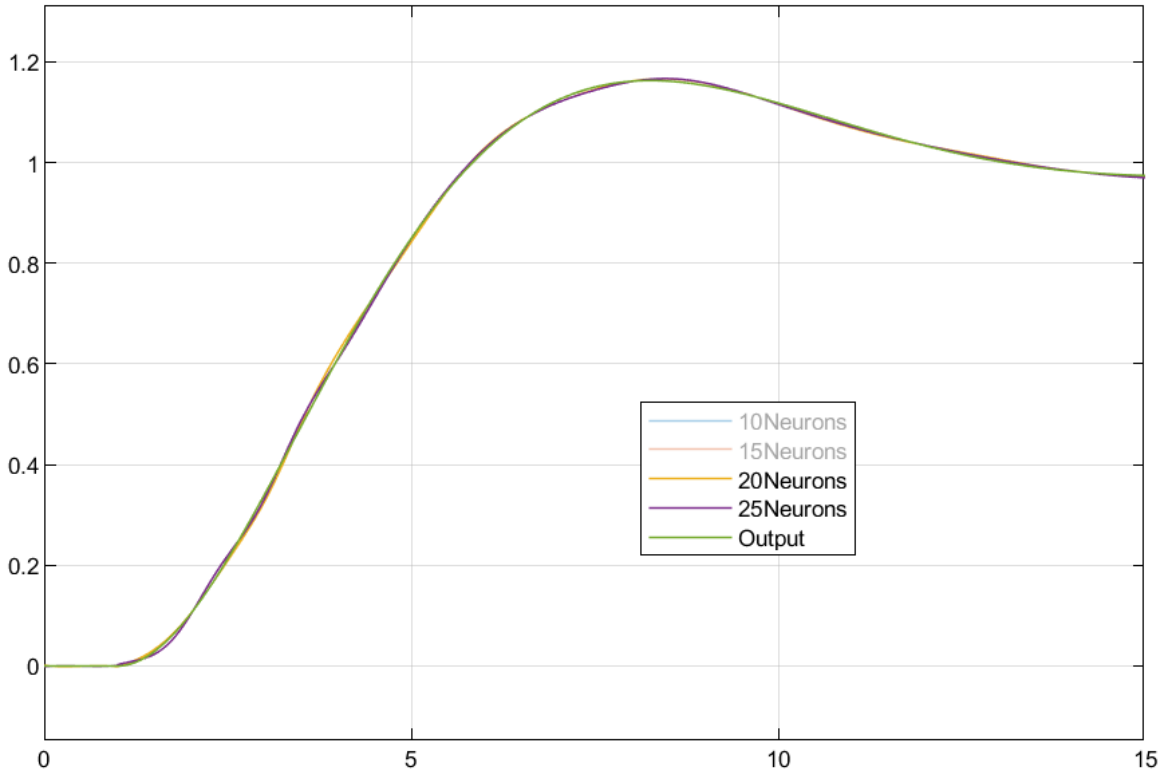

Test: R=0.99988
Data
Fit
Y = T
Output ~= 1*Target + -0.00042
Target


All: R=0.99989
Data
Fit
Y = T
Output ~= 1*Target + -0.00011
Target

Output Comparison

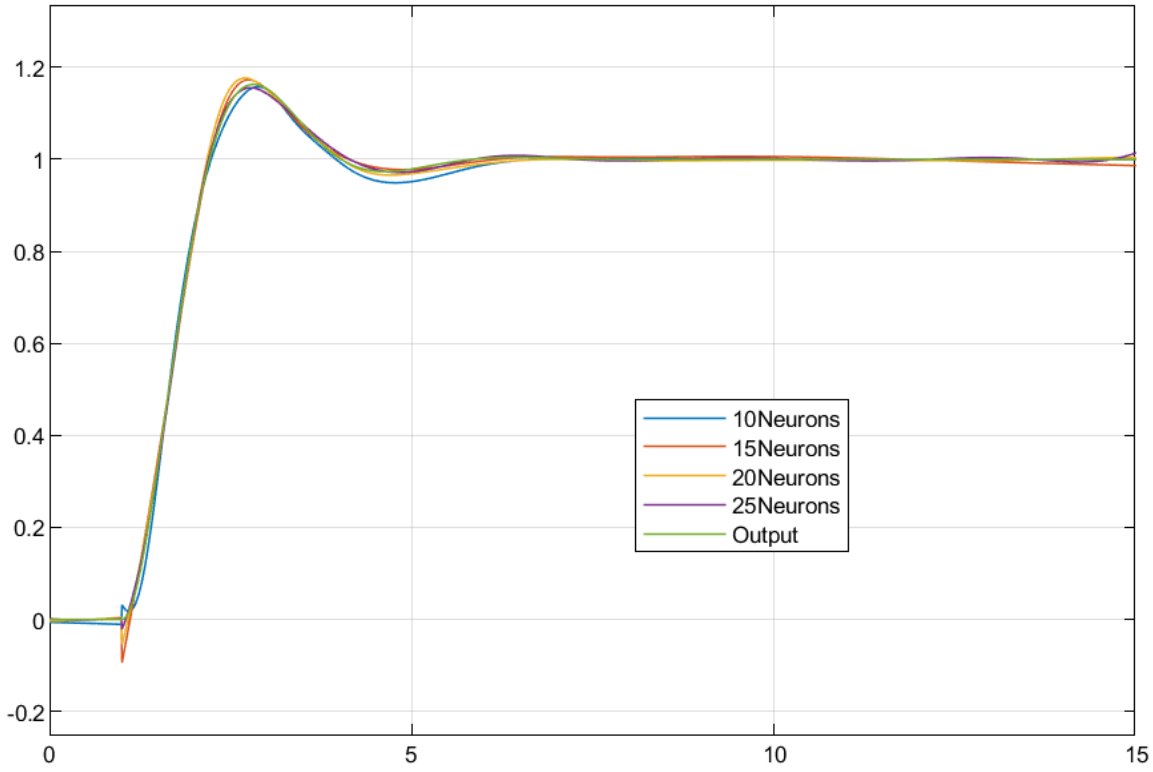


Simulink Diagram



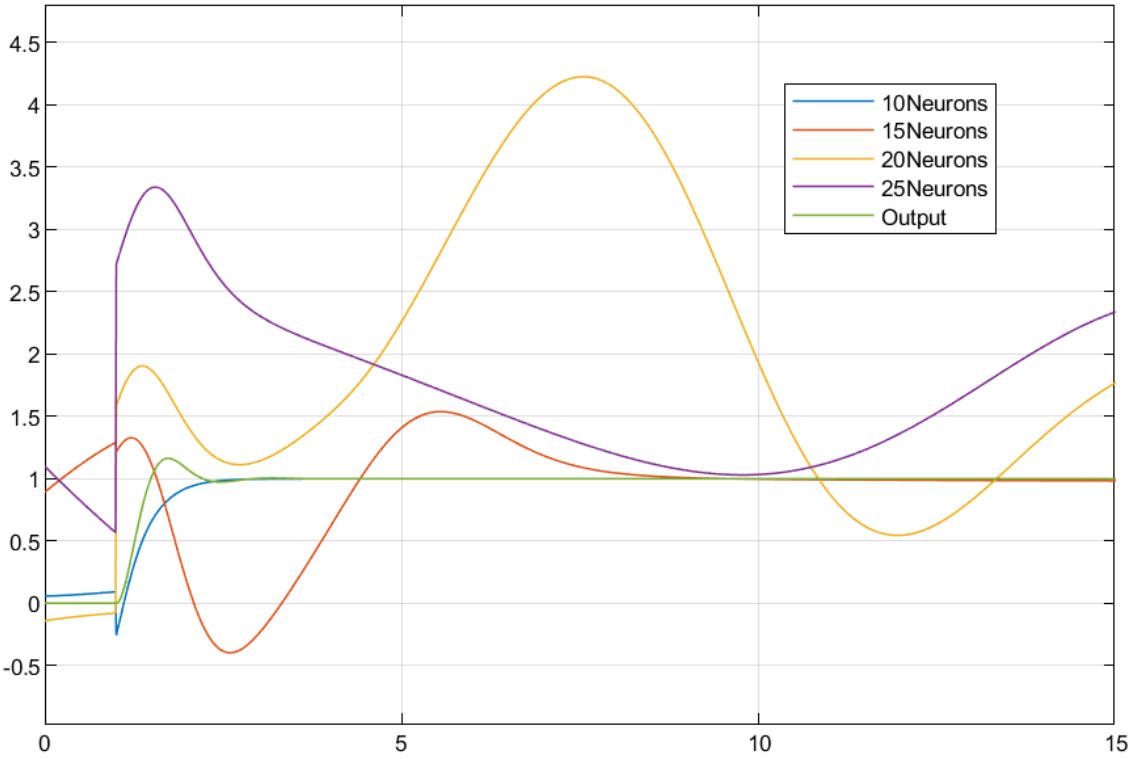
Plot for omega = 0.5

It is observed that as the number of neurons is increased to 20 and 25, the fit becomes better with reduced mismatch in the initial stages for omega = 0.5.



Plot above omega =2

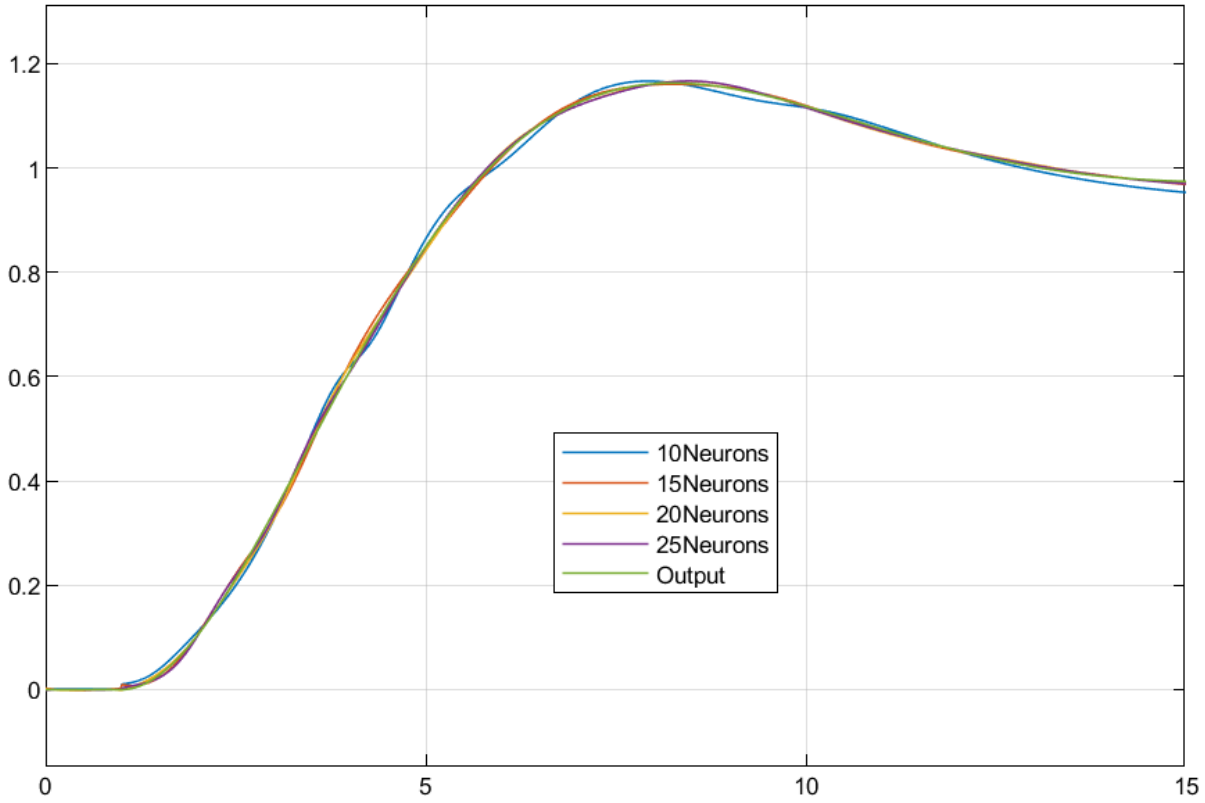
Plot below, omega =5



More neurons are good for fitting if much extrapolation is not required. However, since more neurons accommodate more information, they might fit unnecessarily complex functions.

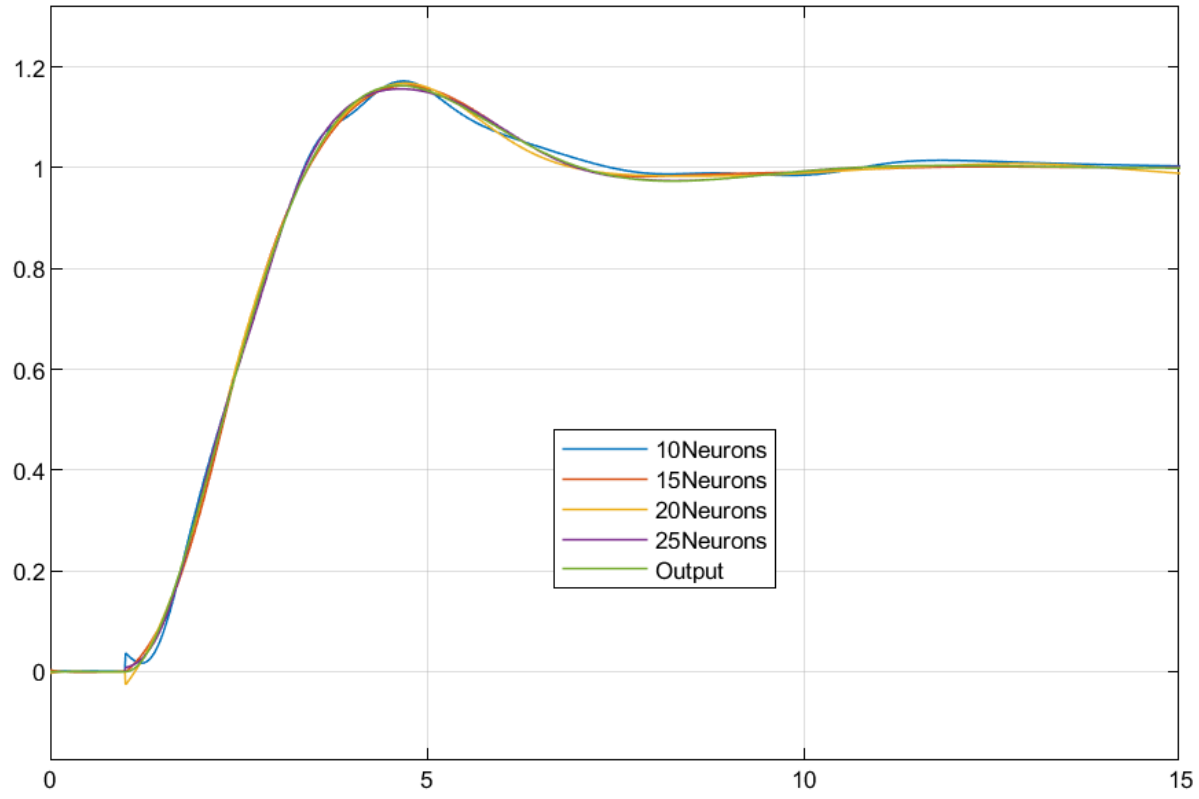
Extended Comparison

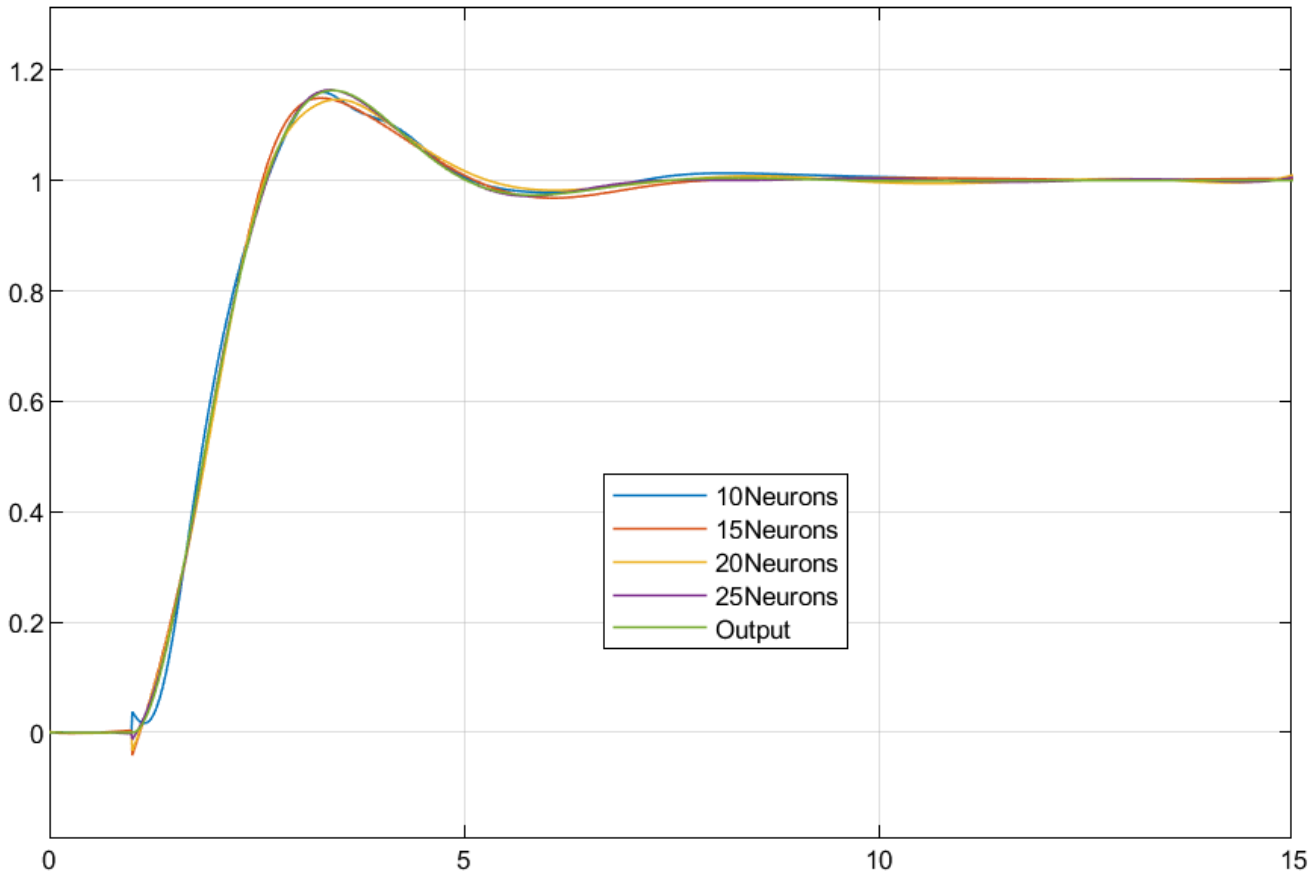
This part merely compares all the fits at all the given omega values for training.



Plot above, Omega = 0.5

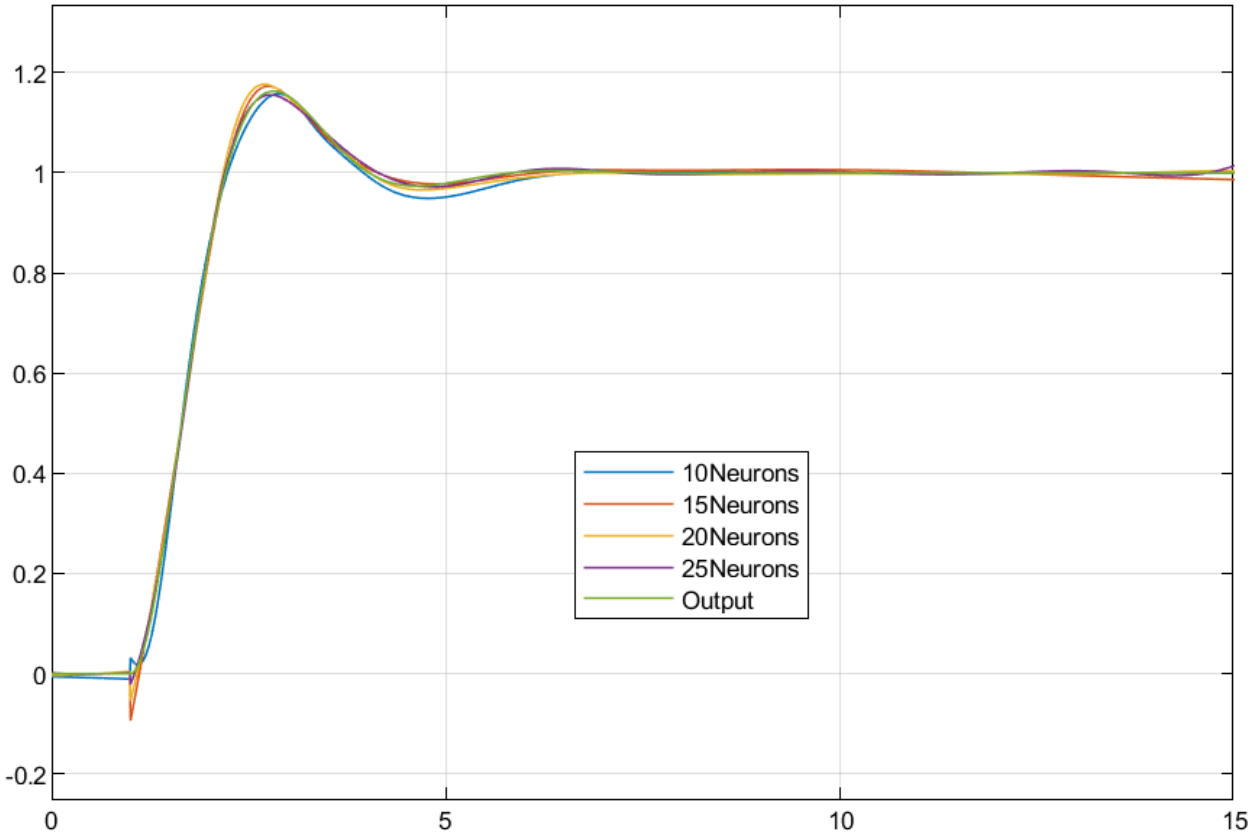
Plot below, Omega = 1

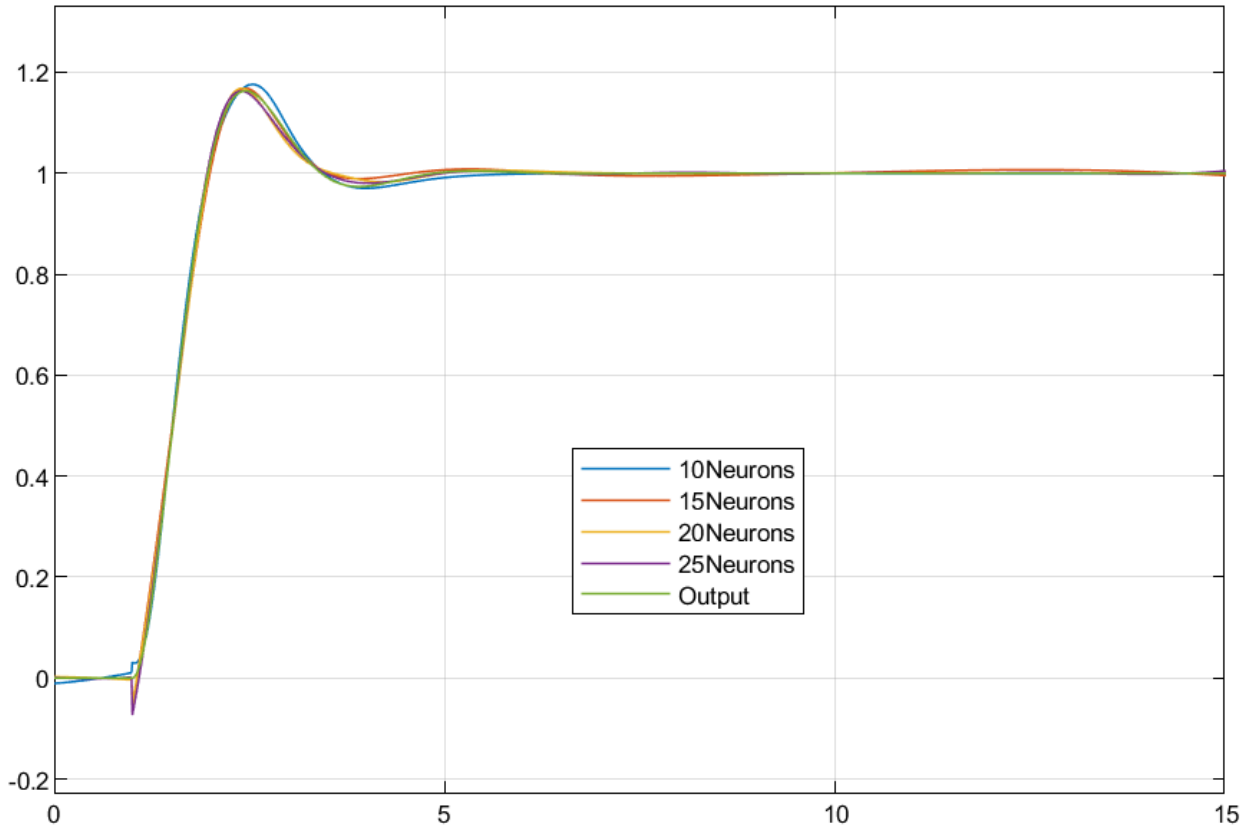




Plot above, omega = 1.5

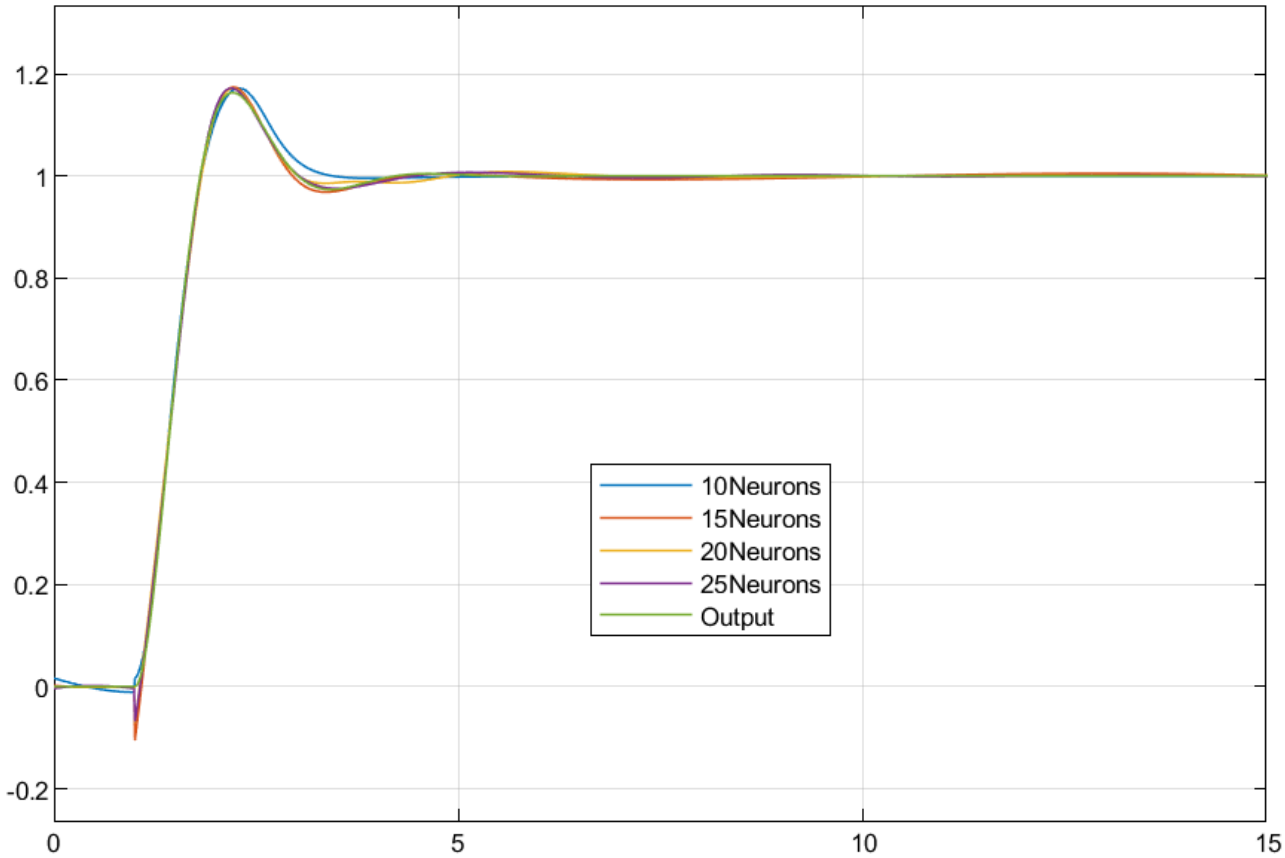
Plot below, omega = 2

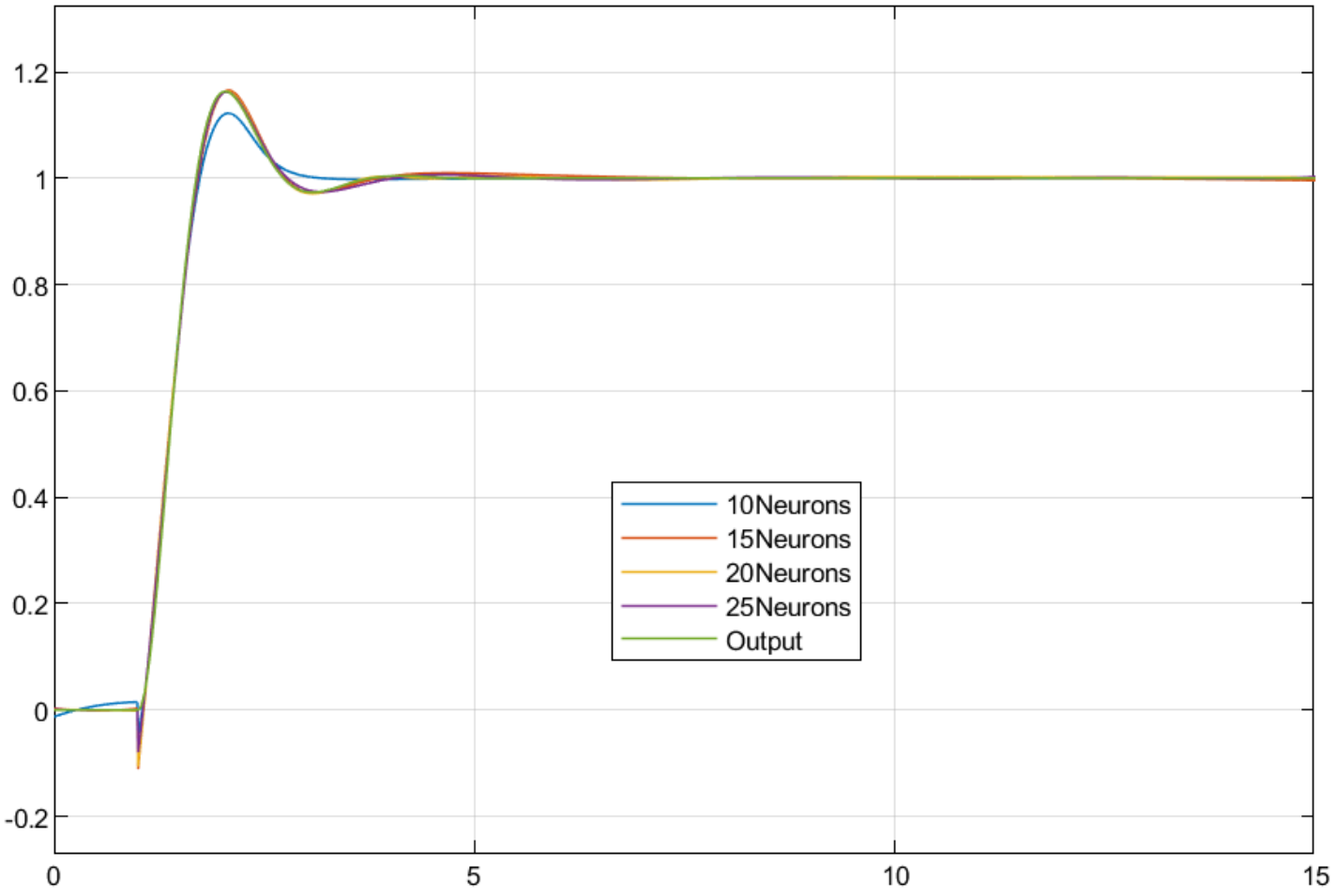




Plot above, omega = 2.5

Plot below, omega =3



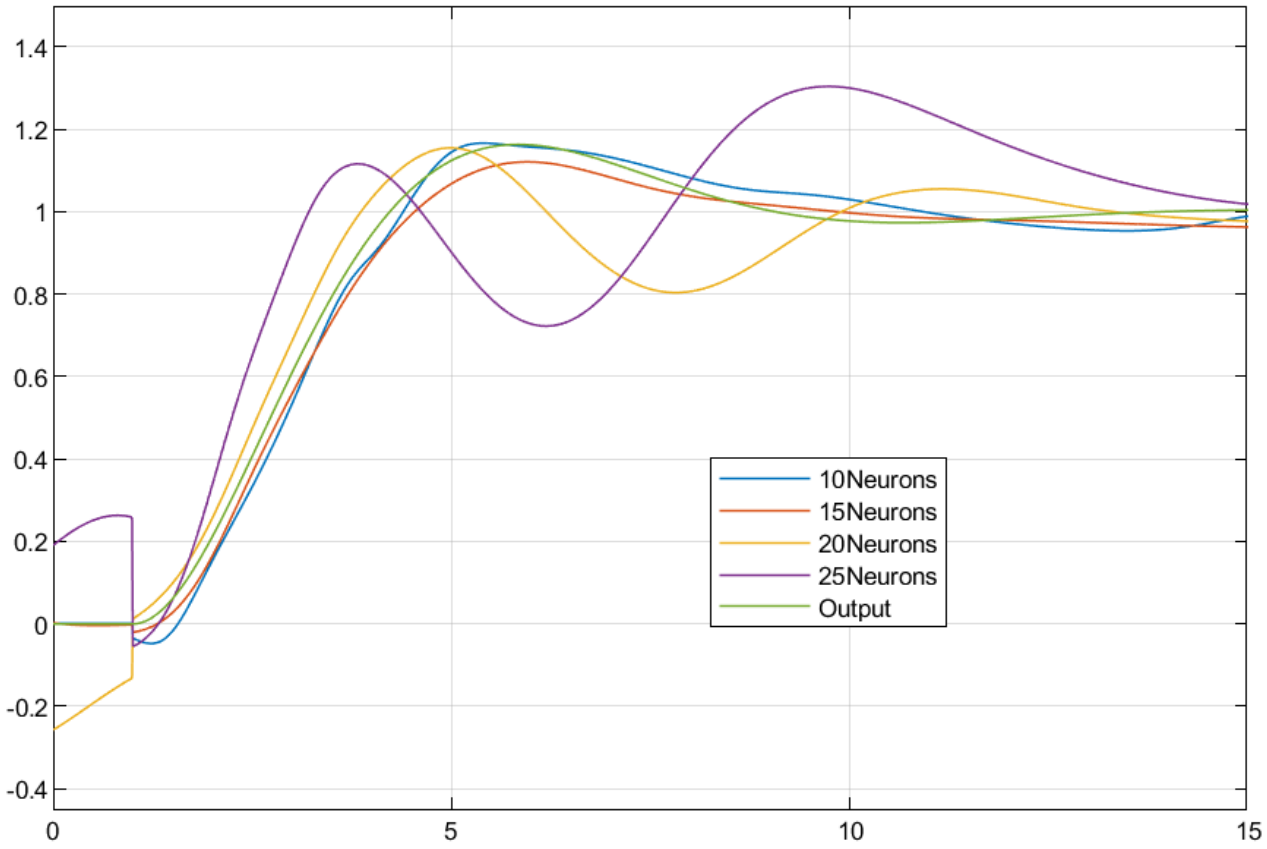


Omega = 3.5

Conclusion – From the training and validation data, it is apparent that more neurons produce a better fit. This is supported by the best performance for validation data and the general performance (in the Neural Network Training window) metrics.

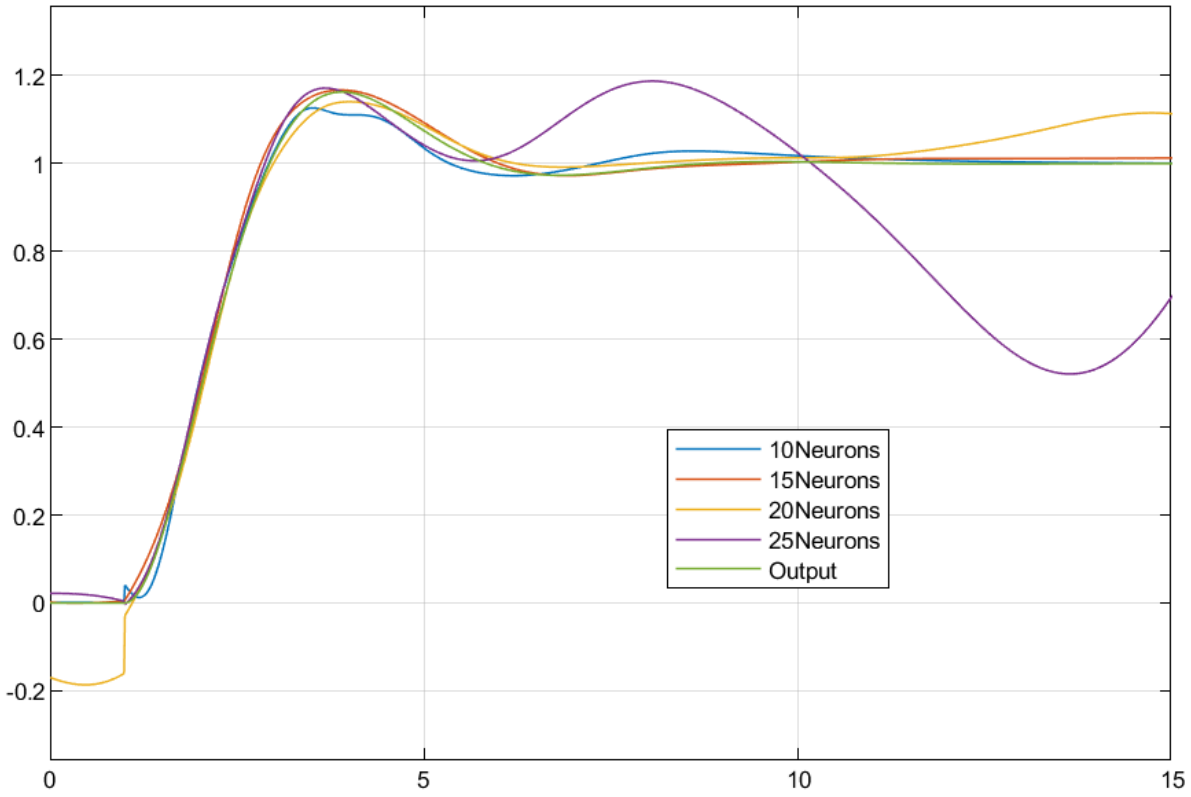
Other than that, the output is slightly erroneous in the transition state but is very good once steady state is reached for almost all neural networks. However, it is notable that the neural network with 10 Neurons goes up just before the step input whereas others dip downwards. The dipping amplitude decreases with the increase of neurons.

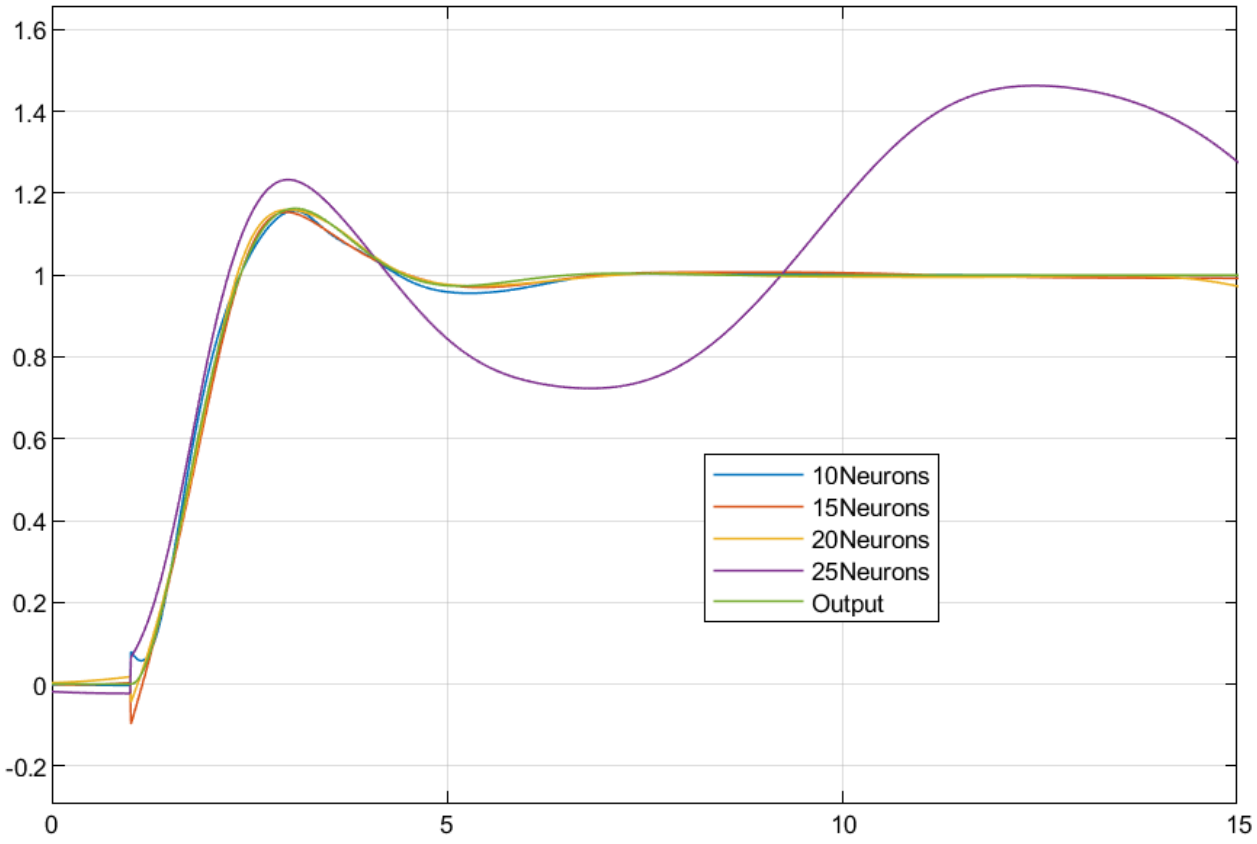
b. Testing extrapolation



Plot above, omega =0.75

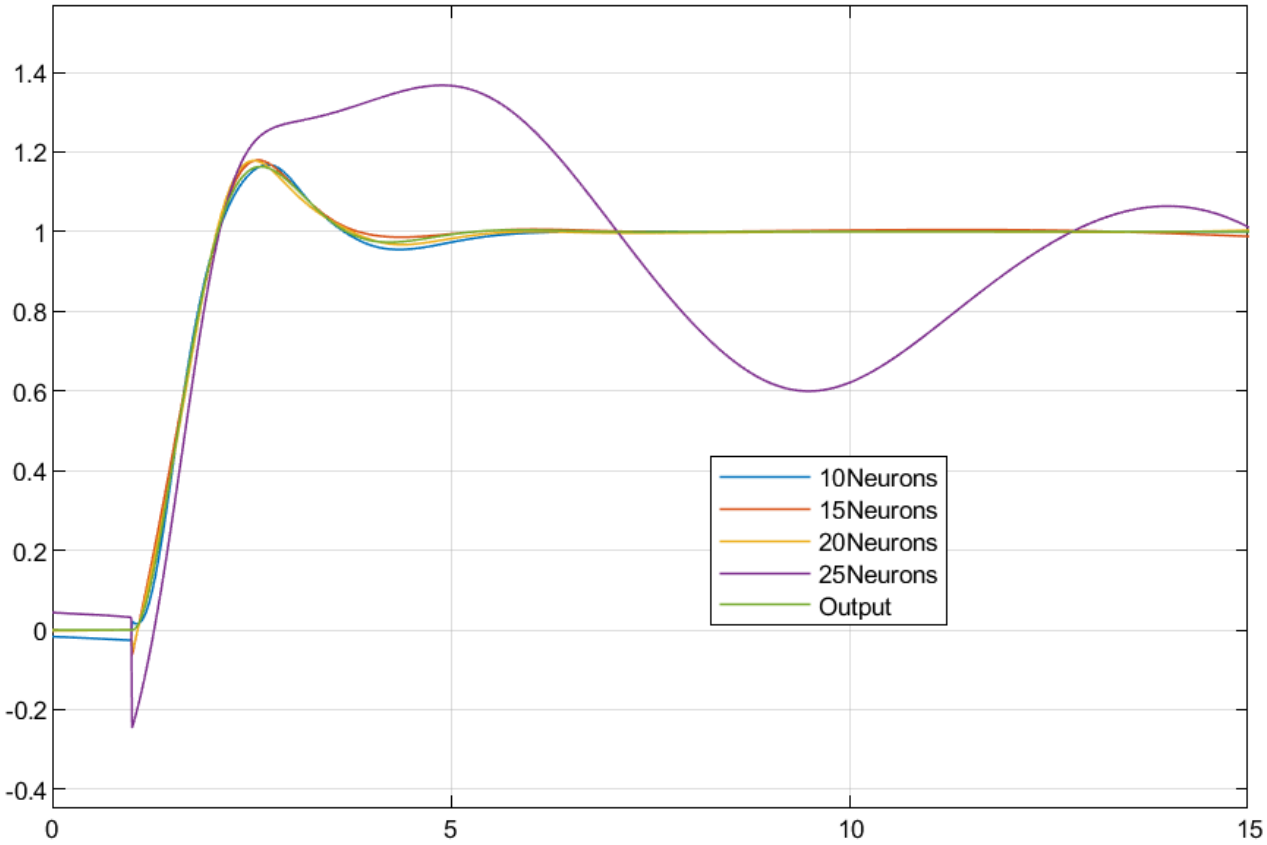
Plot below, omega=1.25

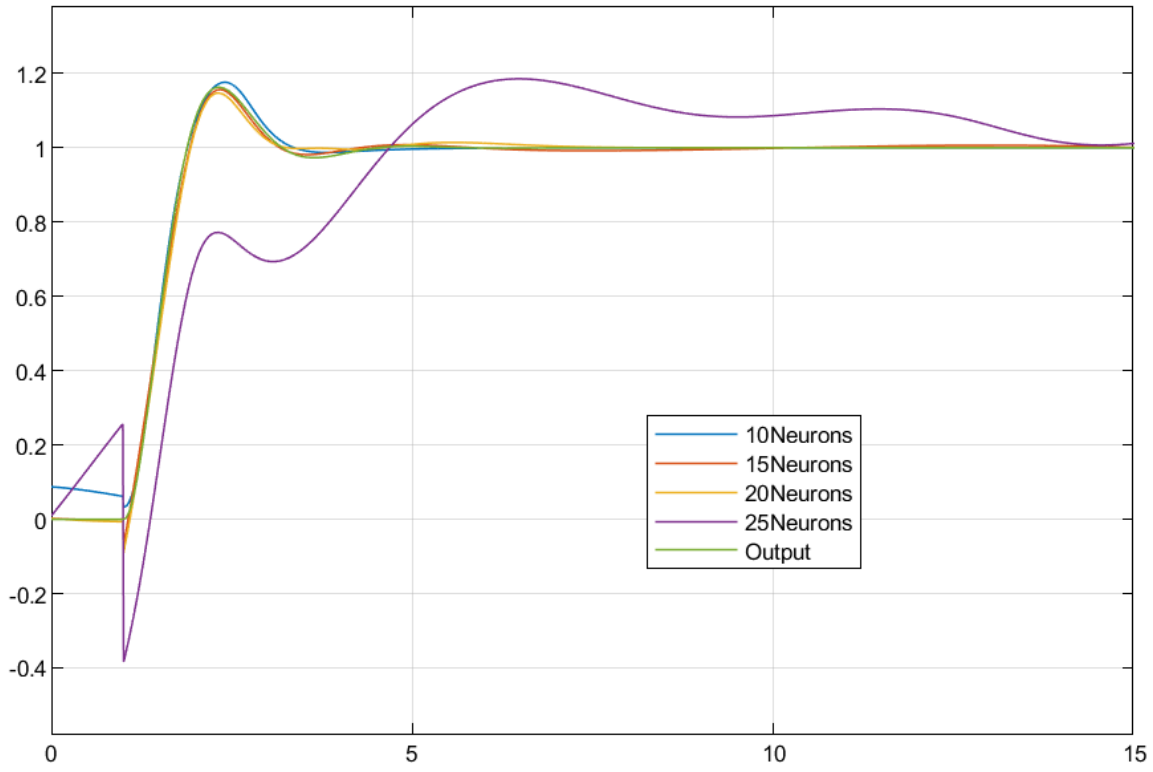




Plot above, omega = 1.75

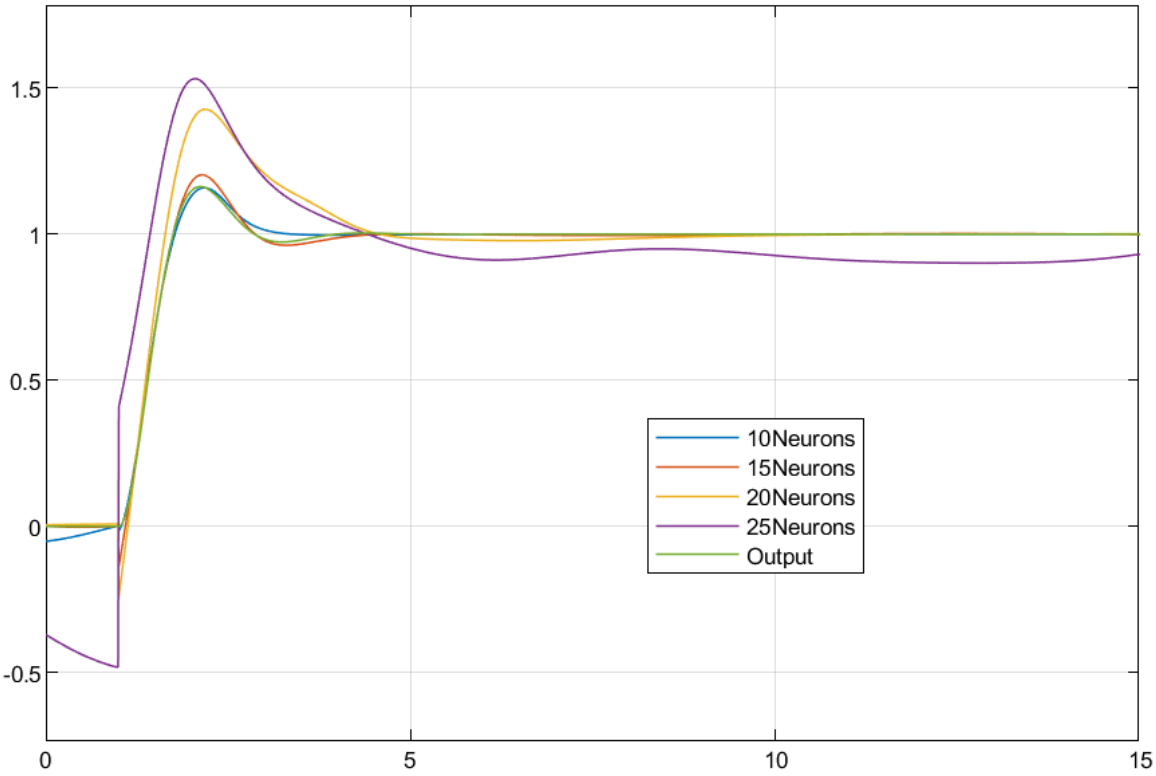
Plot below, omega = 2.25





Plot above, omega = 2.75

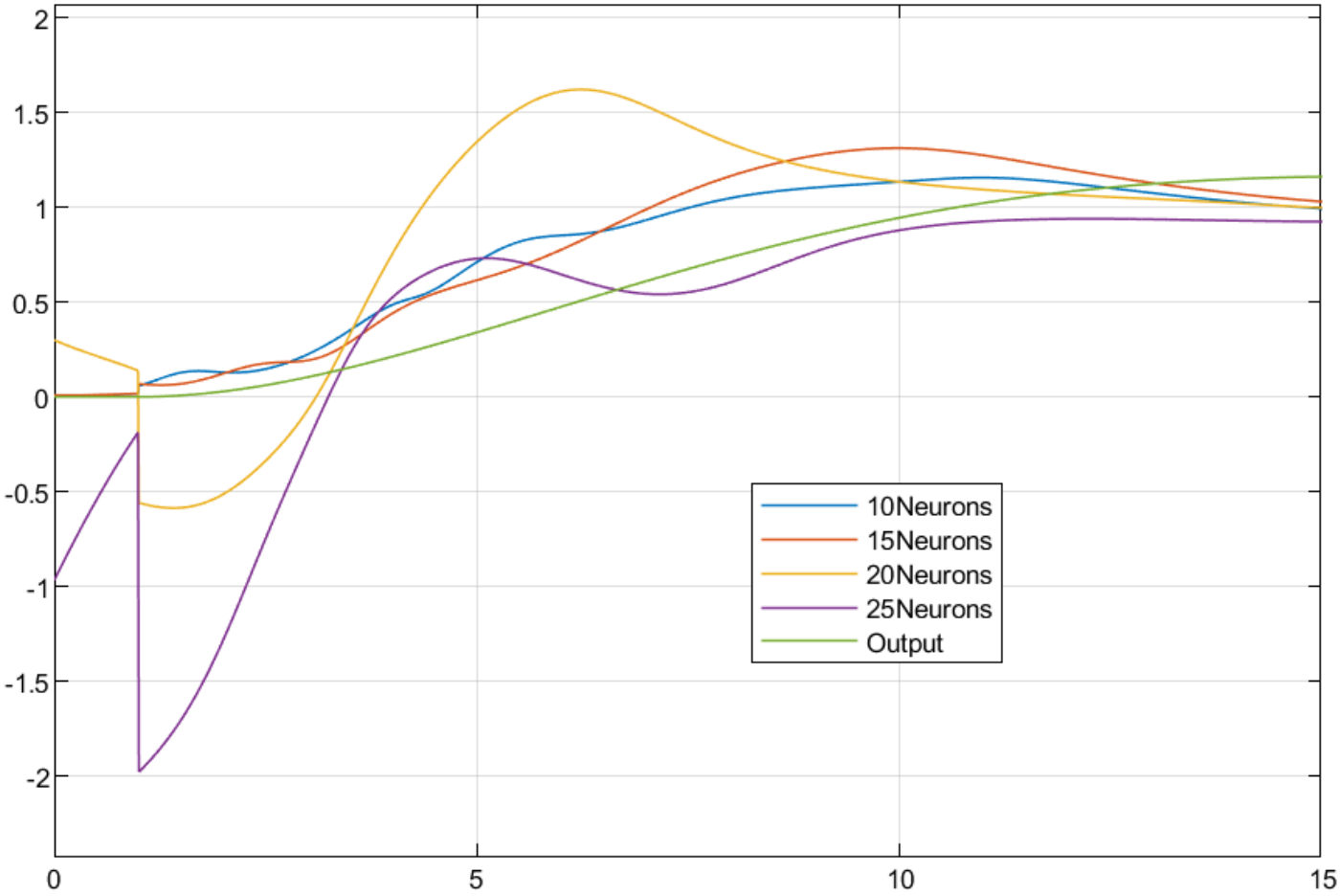
Plot below, omega = 3.25



Conclusion – It is observed that smaller frequencies are tougher to interpolate as compared to larger frequencies. The neural network with 10 neurons does the best job at interpolation as it provides the nearest output at omega = 0.75. More number of neurons may cause overfitting which is why more neurons lead to less efficient interpolation.

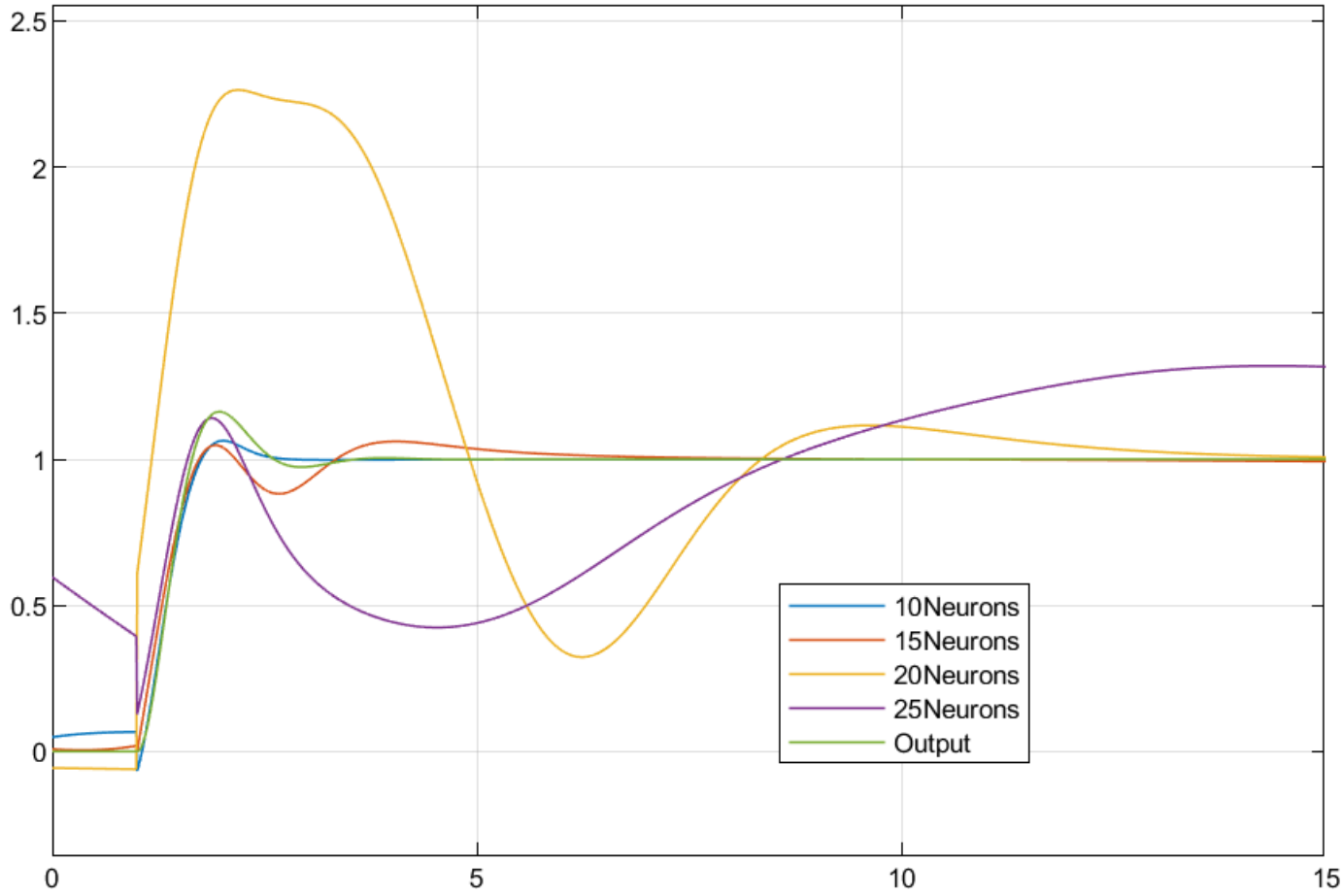
C. The neural network with 10 neurons does a good job of interpolating frequency at lower as well as higher frequencies. Thus, **neural networks with around 10 neurons** will work effectively for this system if they are trained using inputs and outputs over all the operating frequencies. If the operating frequency doesn't change or stays constant at regions used for training data then more neurons may be good. Note that more neurons are good at fitting but they are not so good at interpolation for this system. Also, none of the neural networks is really close to the system so they might not be usable if the acceptable errors in output are of the magnitude 10^-3.

d. Plots for extrapolation



Plot above, omega = 0.25

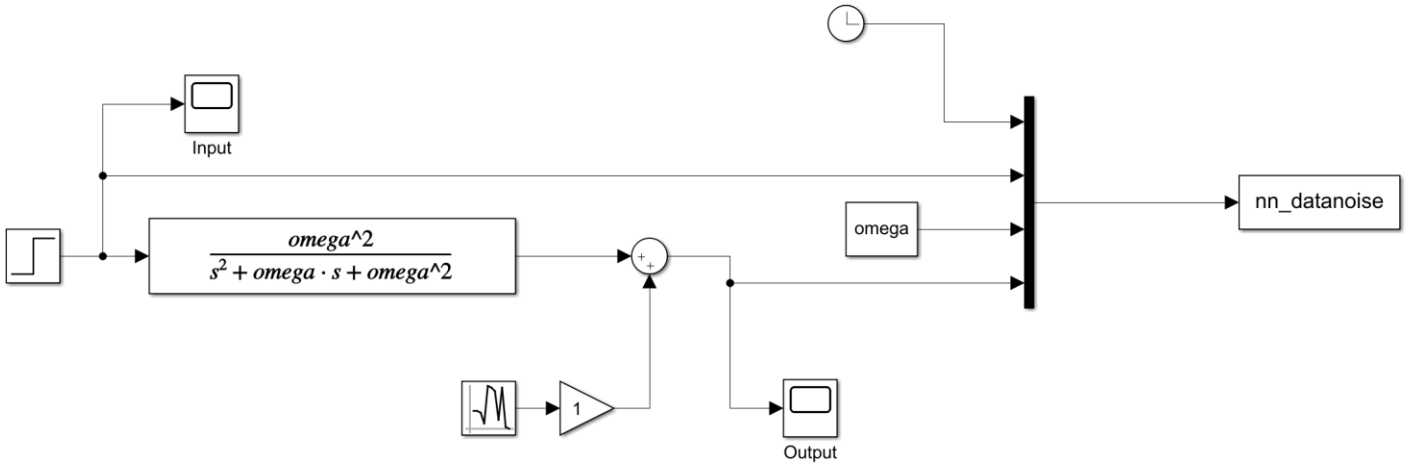
Plot below, omega = 3.75



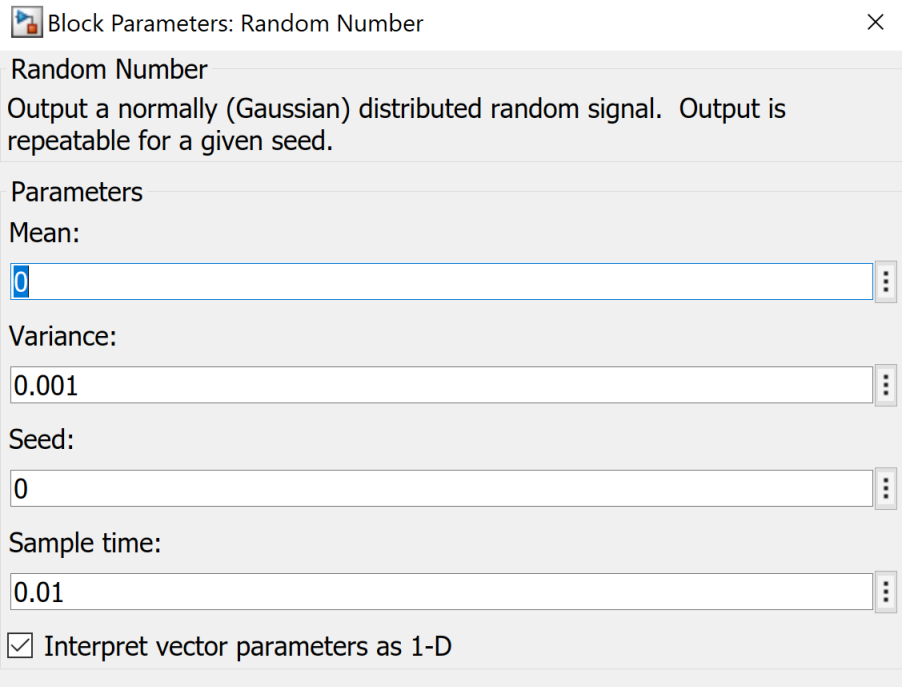
Conclusion – It is observed that for both frequencies, the neural network with 10 neurons is the closest to the actual output. While it doesn't do a very good job when operating outside the training limits, it is still better than the rest.

Problem – 2:

a.



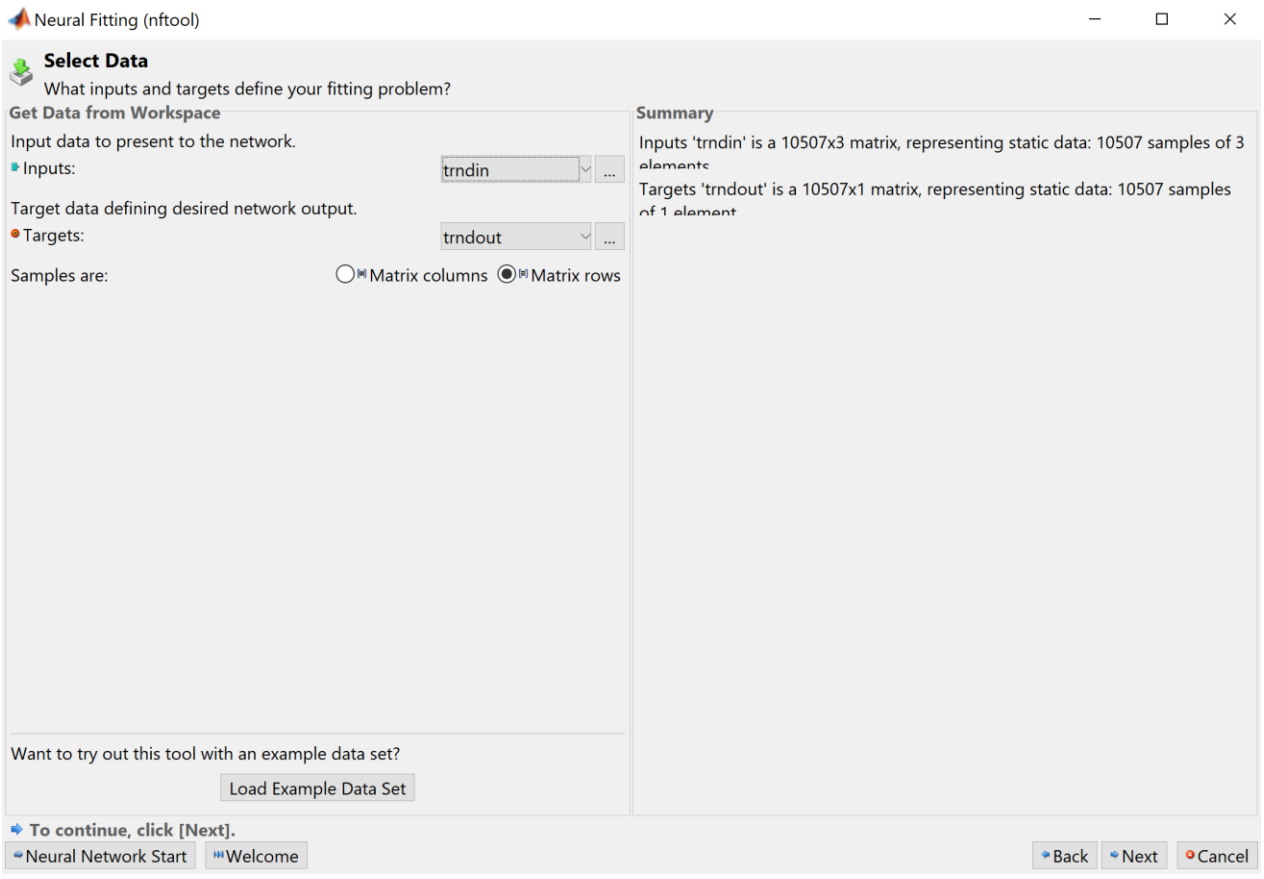
Simulink Diagram

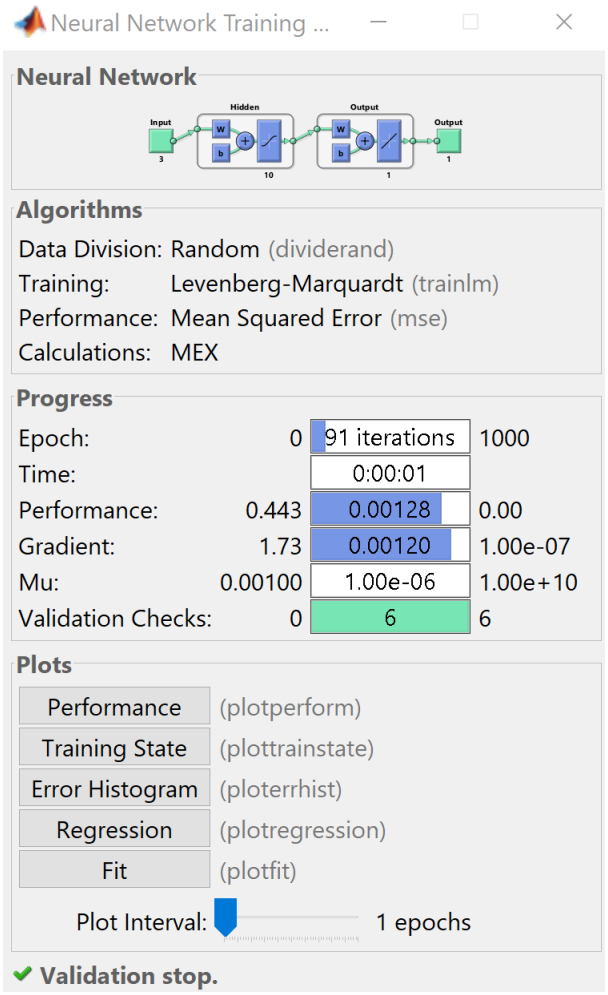


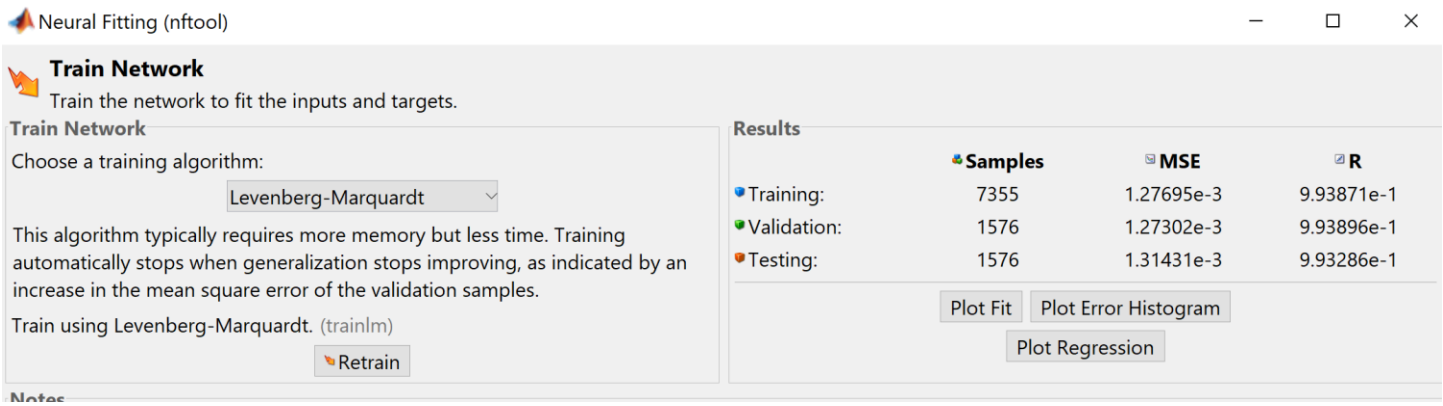
Noise Parameters

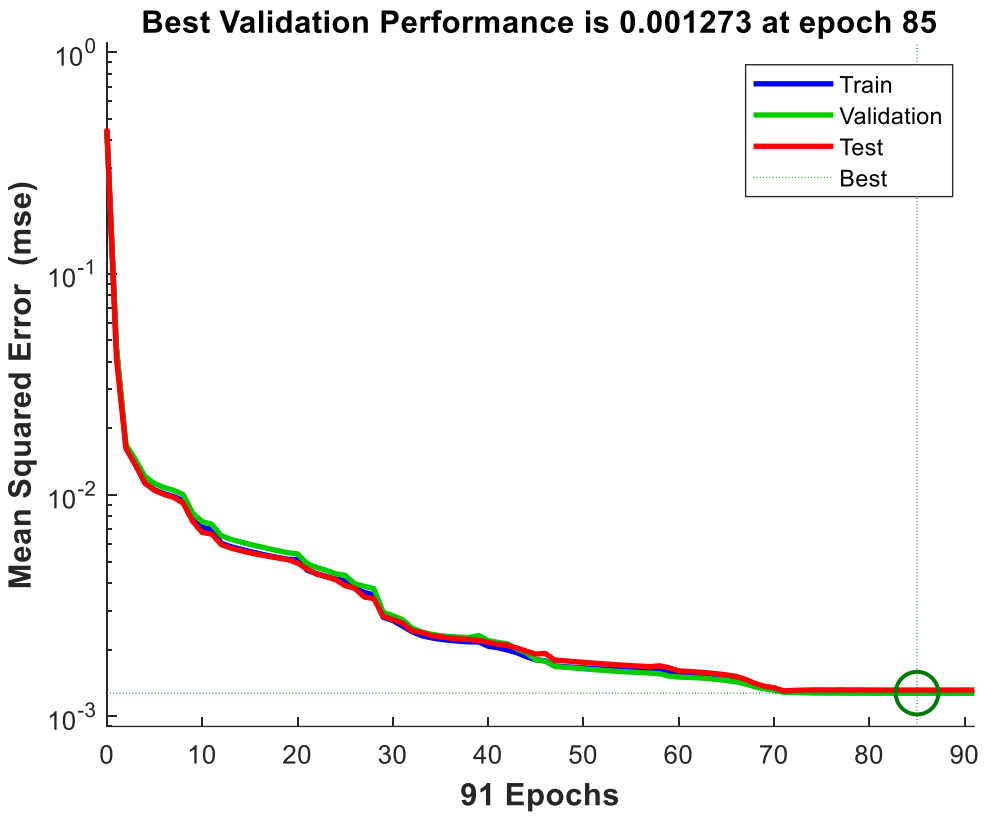
MATLAB Commands to arrange the data

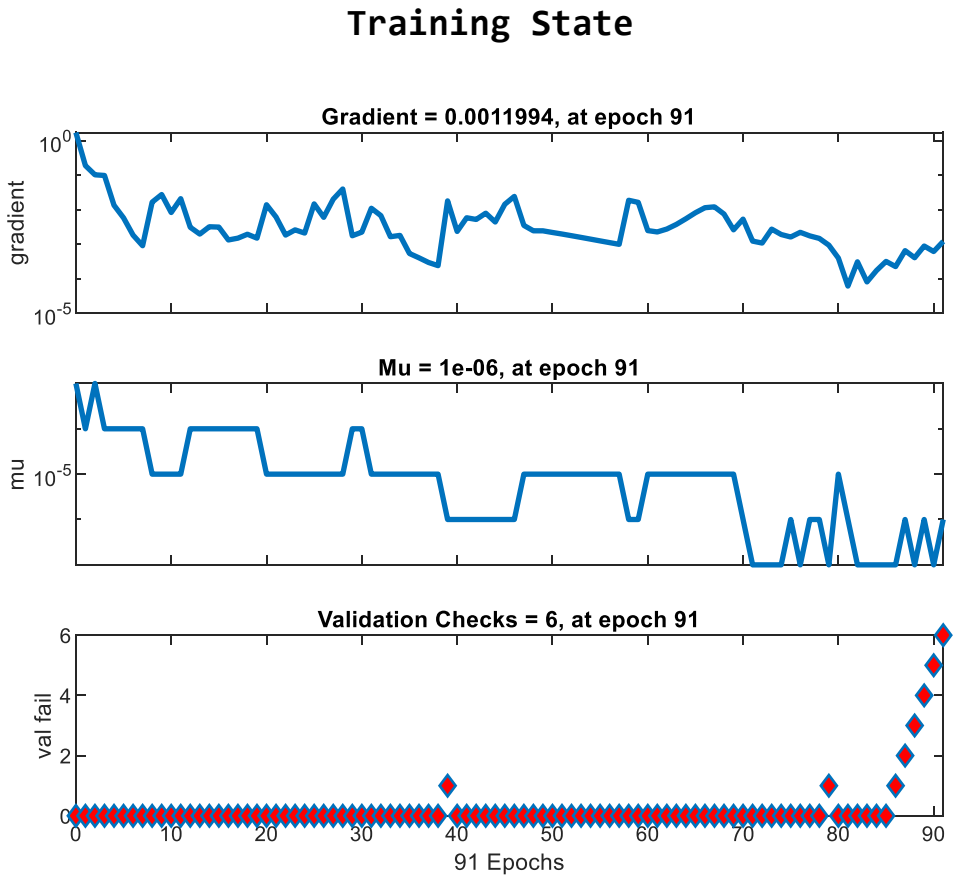
```
omega=0.5 %run simulation after every new omega value

trndata=nn_datanoise

omega=1

trndata=[trndata;nn_datanoise]

omega=1.5

trndata=[trndata;nn_datanoise]

omega=2

trndata=[trndata;nn_datanoise]

omega=2.5

trndata=[trndata;nn_datanoise]

omega=3

trndata=[trndata;nn_datanoise]

omega=3.5

trndata=[trndata;nn_datanoise]

trndin=trndata(:,1:3)

trndout=trndata(:,4)
```

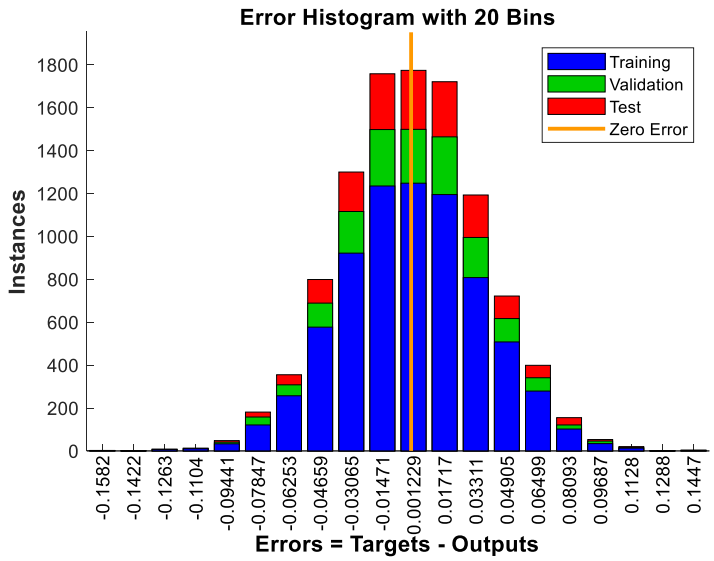


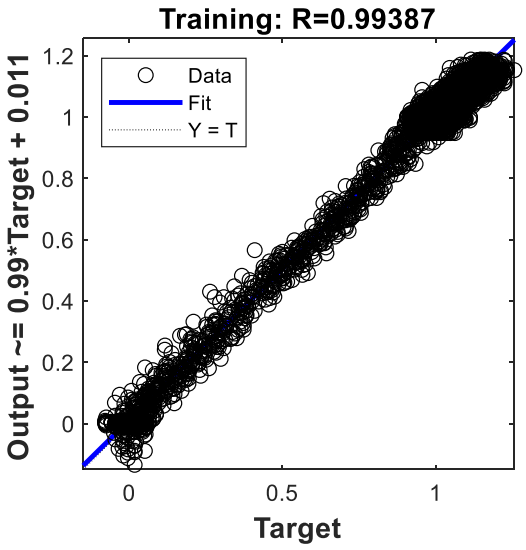

Neural Network Training ...
Neural Network
Input
3
Hidden
W
b
10
Output
W
b
1
Output
1
Algorithms
Data Division: Random (dividerand)
Training: Levenberg-Marquardt (trainlm)
Performance: Mean Squared Error (mse)
Calculations: MEX
Progress
Epoch: 0 91 iterations 1000
Time: 0:00:01
Performance: 0.443 0.00128 0.00
Gradient: 1.73 0.00120 1.00e-07
Mu: 0.00100 1.00e-06 1.00e+10
Validation Checks: 0 6 6
Plots
Performance (plotperform)
Training State (plottrainstate)
Error Histogram (ploterrhist)
Regression (plotregression)
Fit (plotfit)
Plot Interval: 1 epochs
Validation stop.

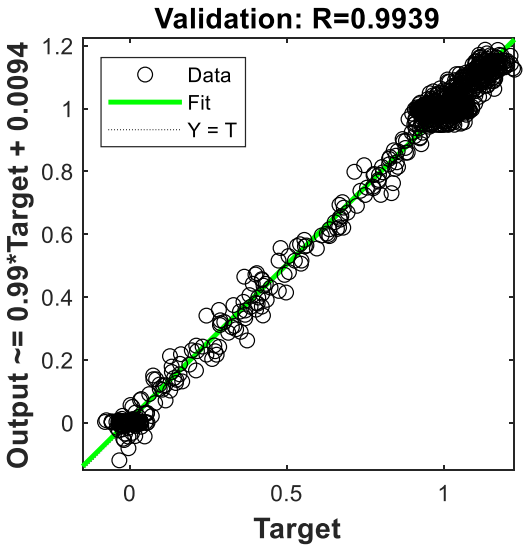

Neural Fitting (nftool)
Train Network
Train the network to fit the inputs and targets.
Train Network
Choose a training algorithm:
Levenberg-Marquardt
This algorithm typically requires more memory but less time. Training automatically stops when generalization stops improving, as indicated by an increase in the mean square error of the validation samples.
Train using Levenberg-Marquardt. (trainlm)
Retrain
Results
Samples MSE R
Training: 7355 1.27695e-3 9.93871e-1
Validation: 1576 1.27302e-3 9.93896e-1
Testing: 1576 1.31431e-3 9.93286e-1
Plot Fit
Plot Error Histogram
Plot Regression
Notes

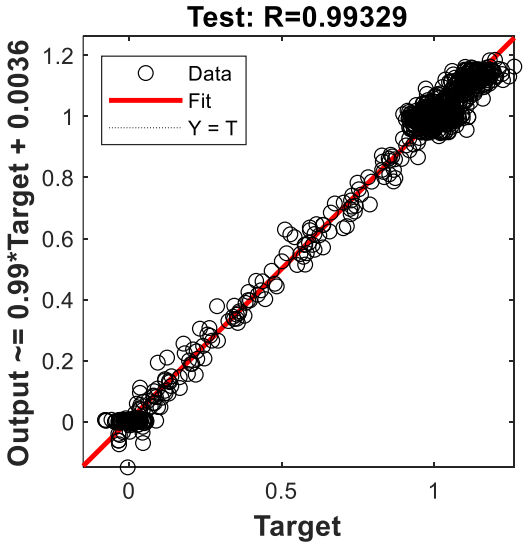

Best Validation Performance is 0.001273 at epoch 85
Train
Validation
Test
Best
Mean Squared Error (mse)
91 Epochs

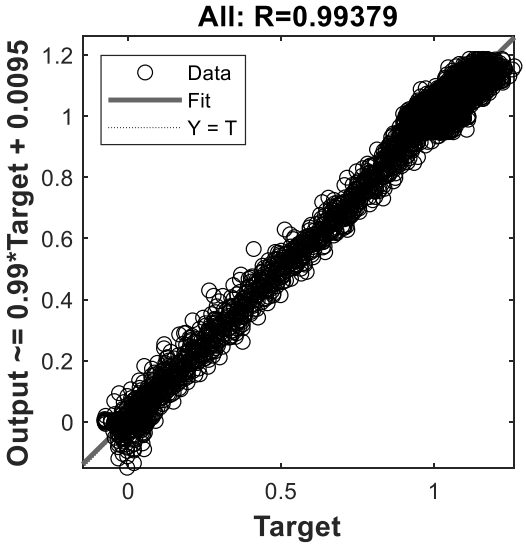

Training State
Gradient = 0.0011994, at epoch 91
gradient
Mu = 1e-06, at epoch 91
mu
Validation Checks = 6, at epoch 91
val fail
91 Epochs

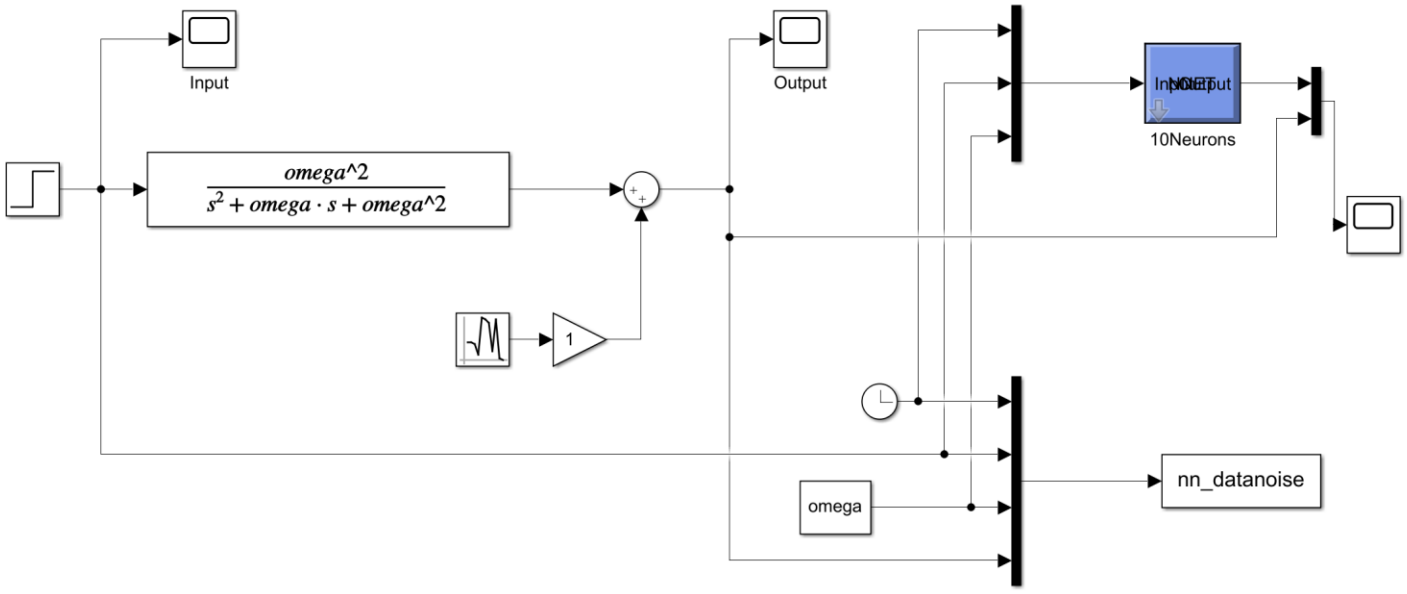

Error Histogram with 20 Bins
Training
Validation
Test
Zero Error
Instances
1800
1600
1400
1200
1000
800
600
400
200
0
-0.1582
-0.1422
-0.1263
-0.1104
-0.09441
-0.07847
-0.06253
-0.04659
-0.03065
-0.01471
0.001229
0.01717
0.03311
0.04905
0.06499
0.08093
0.09687
0.1128
0.1288
0.1447
Errors = Targets - Outputs

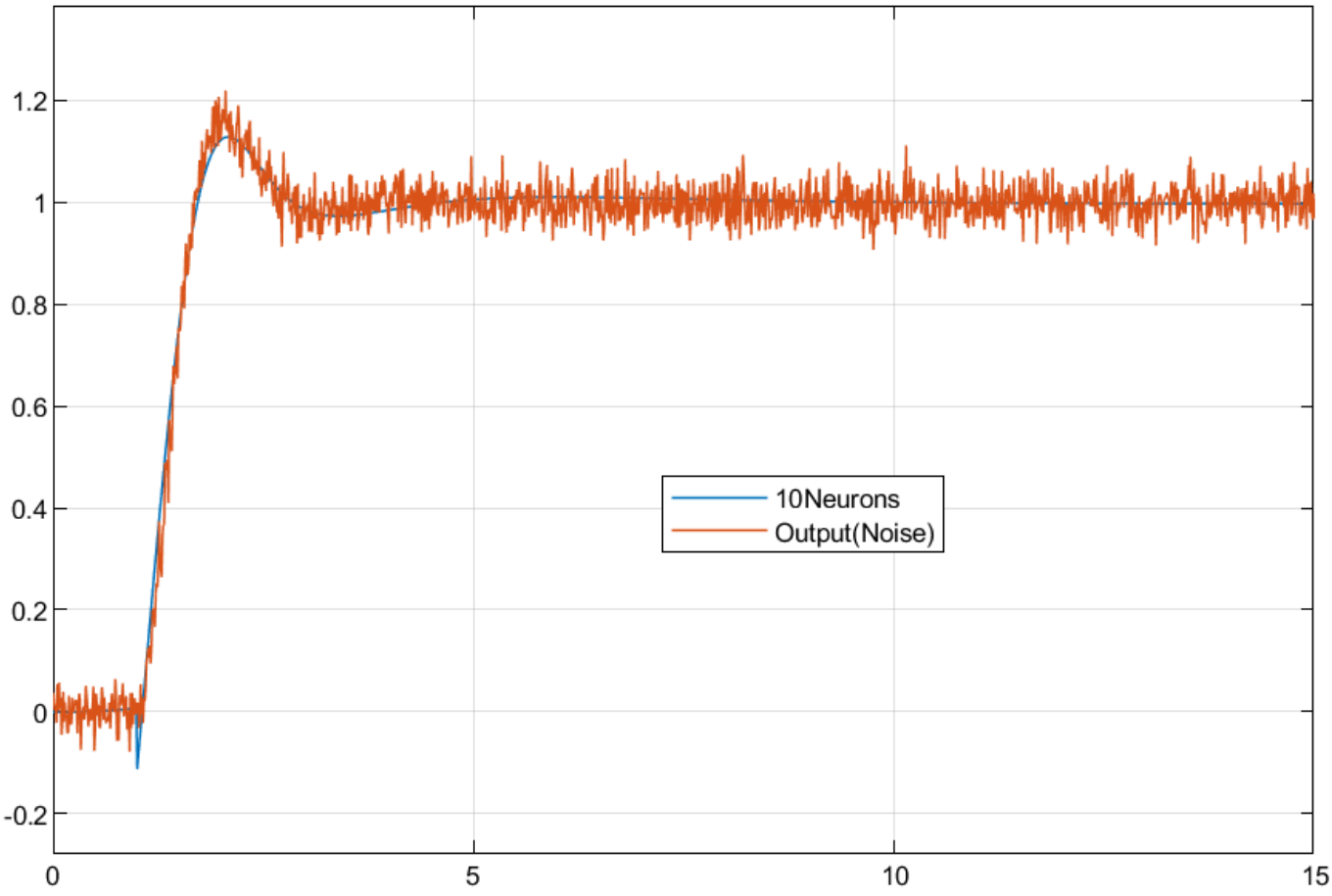

Training: R=0.99387
Data
Fit
Y = T
Output ~= 0.99*Target + 0.011
Target


Validation: R=0.9939
Data
Fit
Y = T
Output ~= 0.99*Target + 0.0094
Target

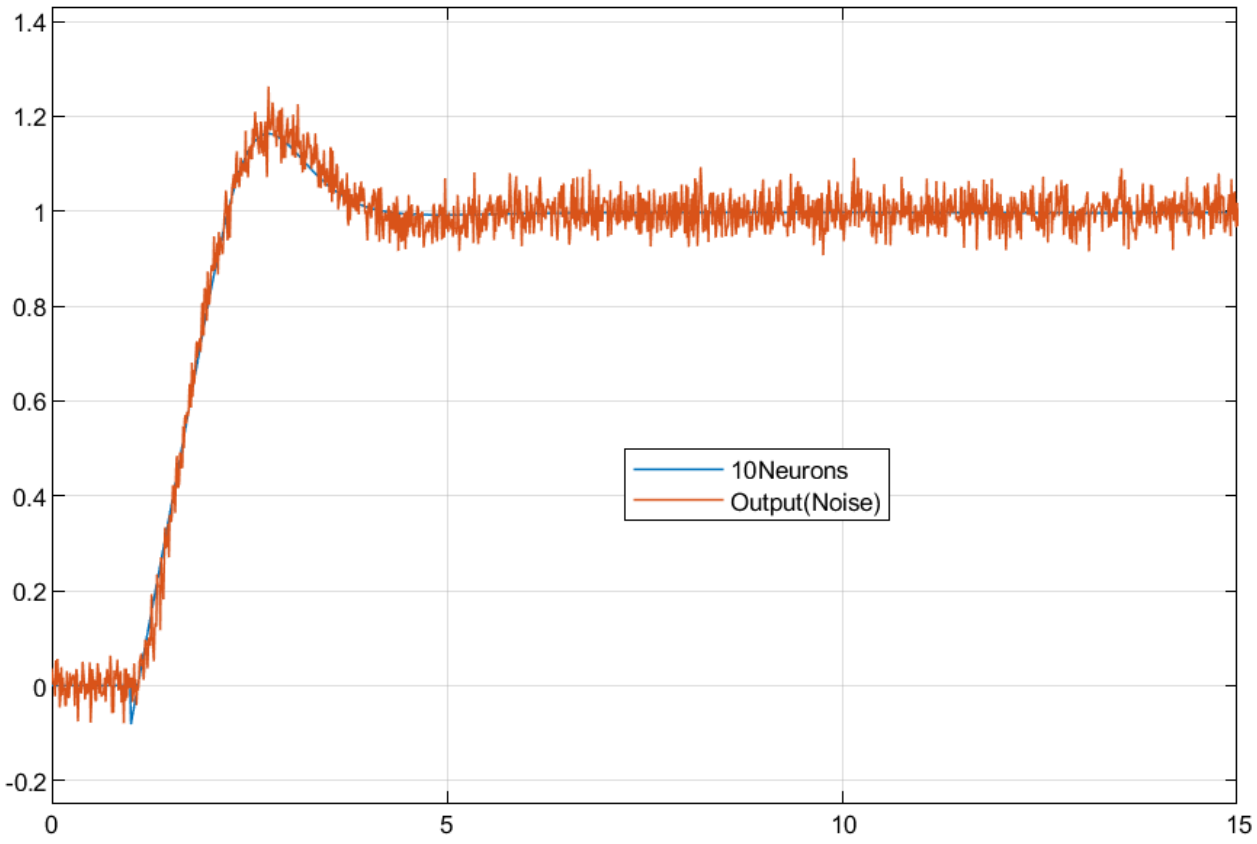

Test: R=0.99329
Data
Fit
Y = T
Output ~= 0.99*Target + 0.0036
Target


All: R=0.99379
Data
Fit
Y = T
Output ~= 0.99*Target + 0.0095
Target


Input
Output
omega^2
s^2 + omega · s + omega^2
1
10Neurons
omega
nn_datanoise

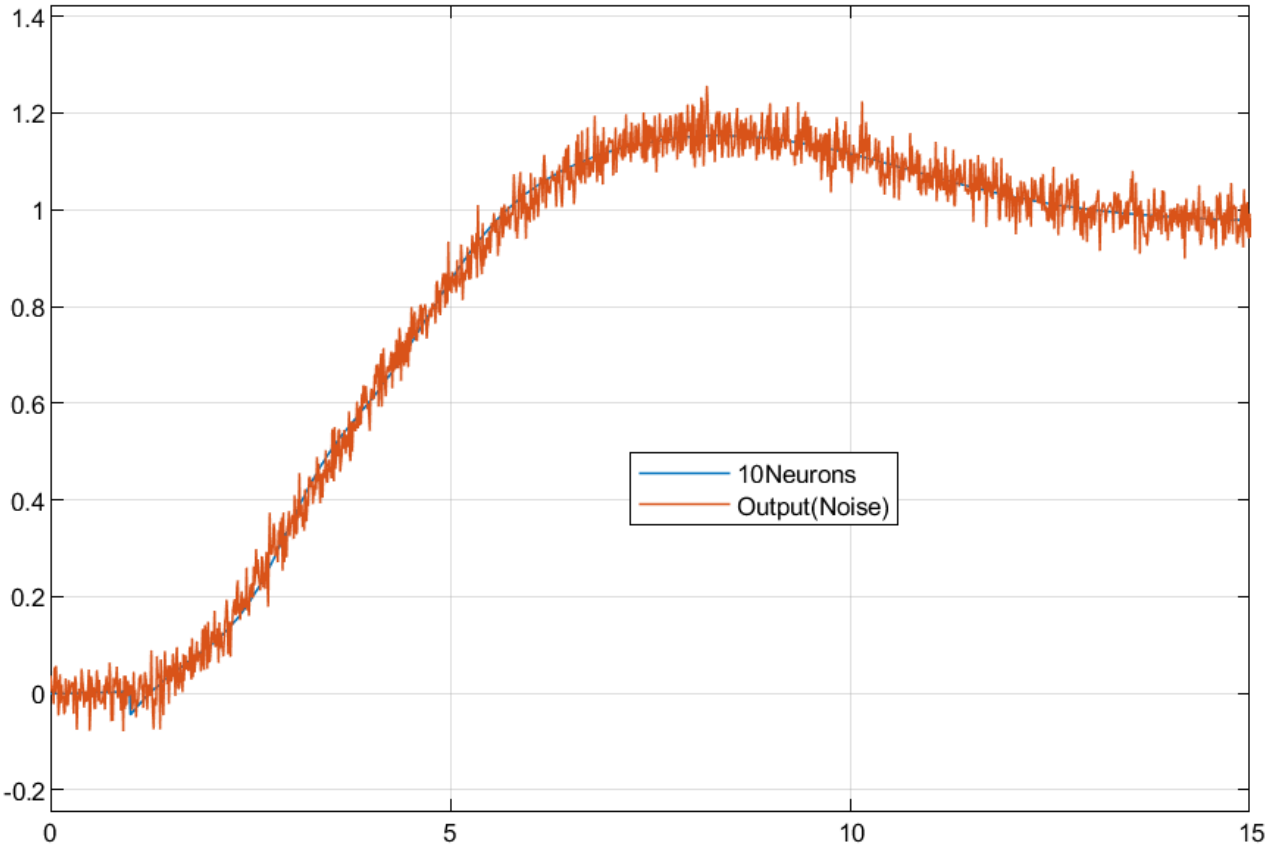

1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
5
10
15
10Neurons
Output(Noise)

Omega = 3.5



Plot above, omega = 2

Plot below, omega = 0.5



Training with 15 Neurons

Neural Network Training ...

Neural Network

Algorithms

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

Progress

Epoch:	0	137 iterations	1000
Time:		0:00:01	
Performance:	1.57	0.00124	0.00
Gradient:	2.55	0.000471	1.00e-07
Mu:	0.00100	1.00e-07	1.00e+10
Validation Checks:	0	6	6

Plots

Performance (plotperform)
Training State (plottrainstate)
Error Histogram (ploterrhist)
Regression (plotregression)
Fit (plotfit)

Plot Interval: 1 epochs

Neural Fitting (nftool)

Train Network

Train the network to fit the inputs and targets.

Train Network

Choose a training algorithm:

Levenberg-Marquardt

This algorithm typically requires more memory but less time. Training automatically stops when generalization stops improving, as indicated by an increase in the mean square error of the validation samples.

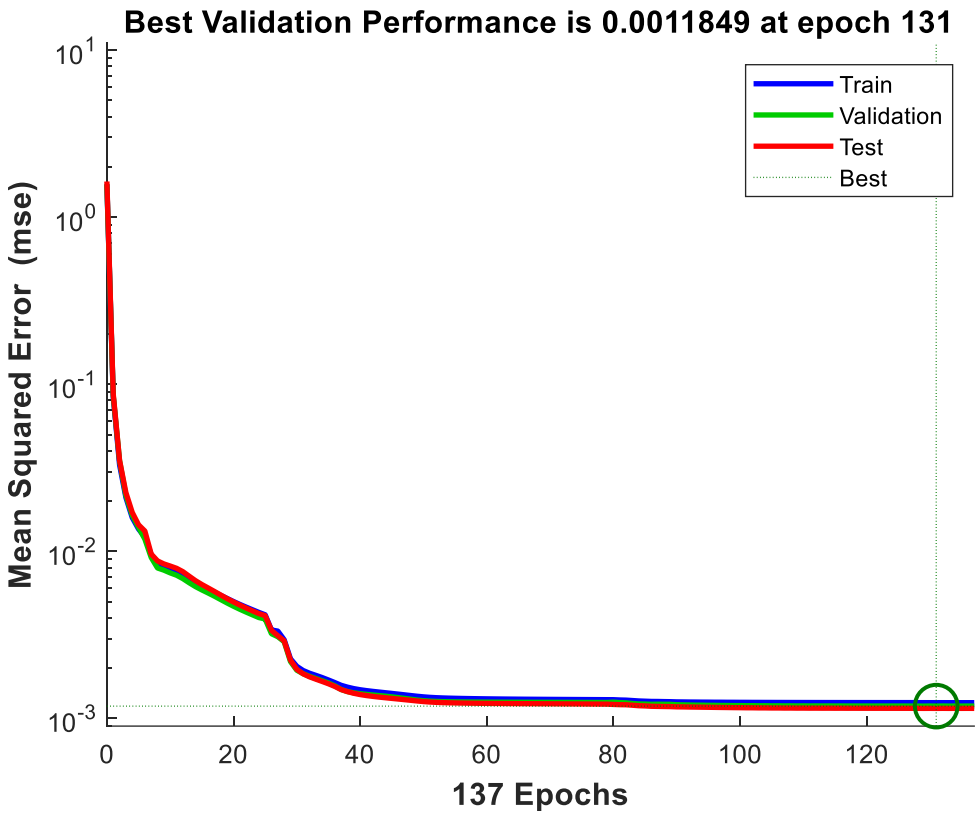
Train using Levenberg-Marquardt. (trainlm)

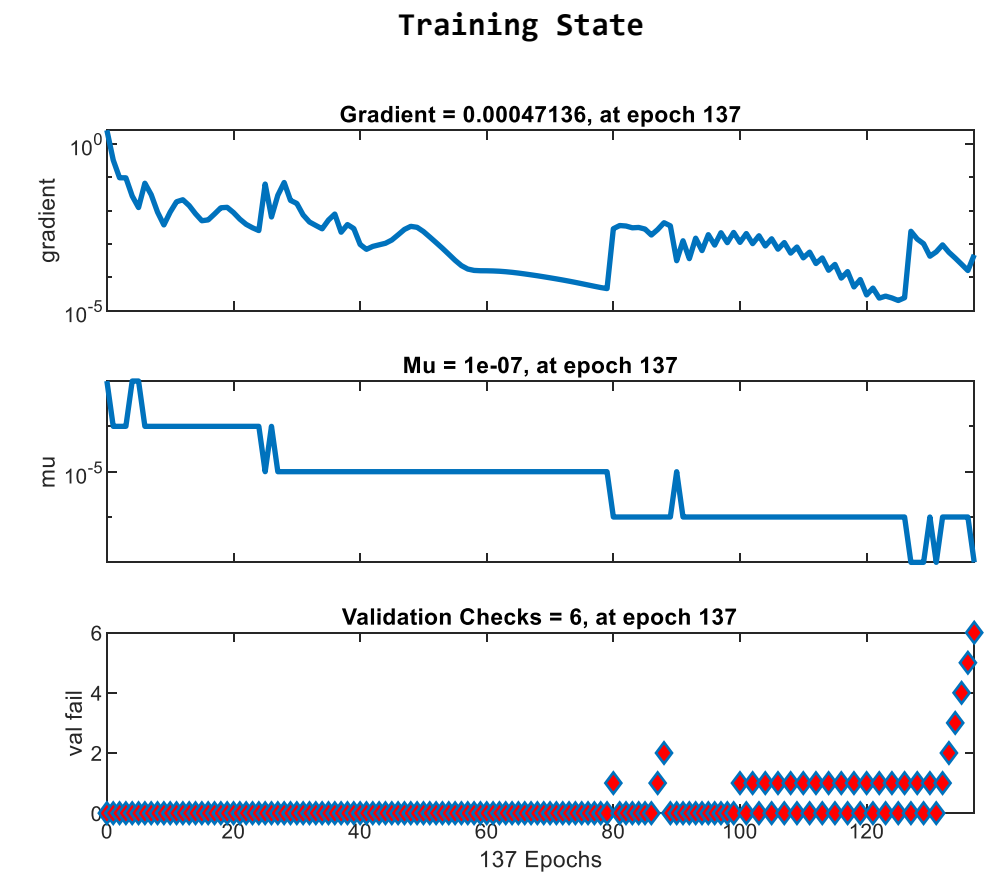
Retrain

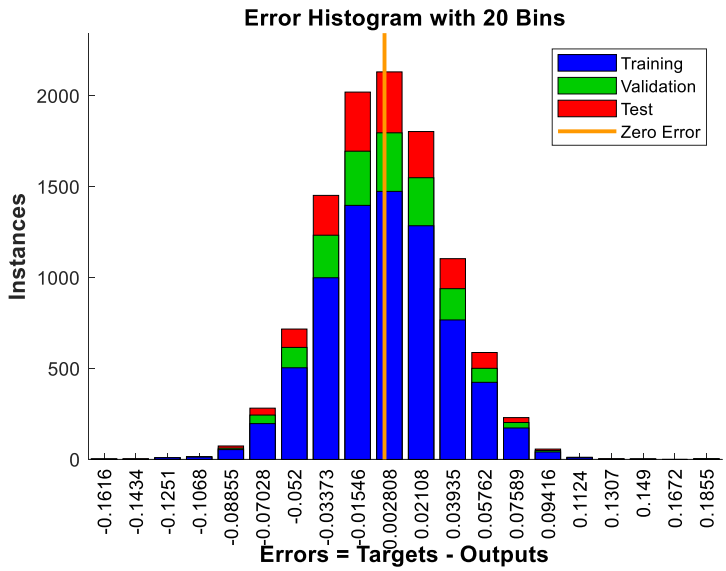
Results

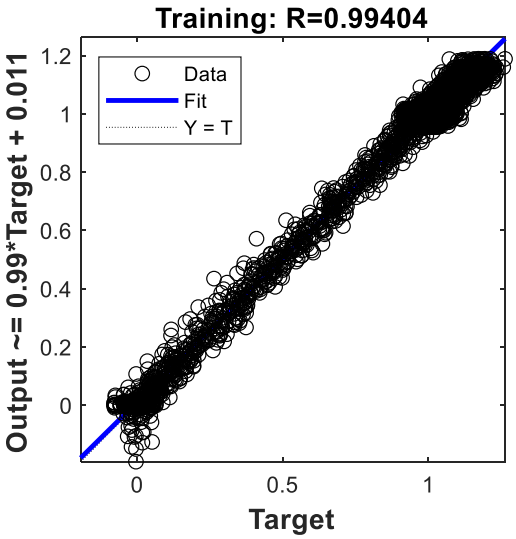
	Samples	MSE	R
Training:	7355	1.24325e-3	9.94040e-1
Validation:	1576	1.18485e-3	9.93999e-1
Testing:	1576	1.14847e-3	9.94390e-1

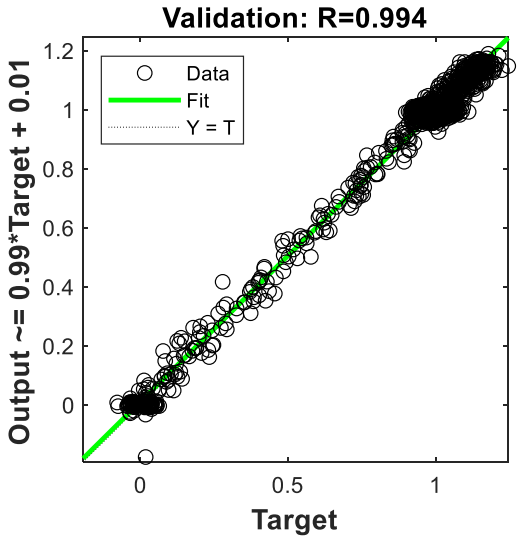
Plot Fit | Plot Error Histogram | Plot Regression

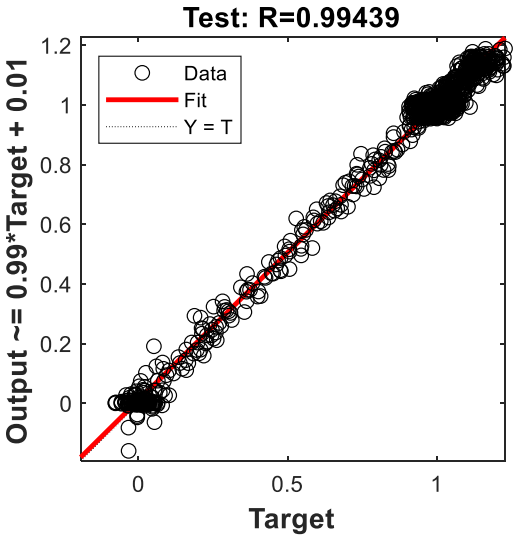

Best Validation Performance is 0.0011849 at epoch 131
Train
Validation
Test
Best
Mean Squared Error (mse)
137 Epochs

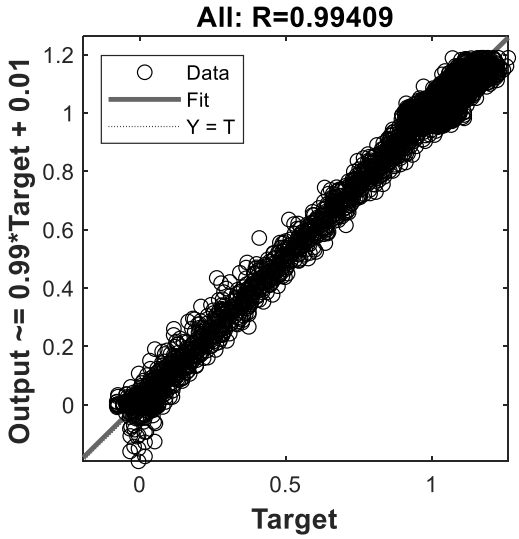

Training State
Gradient = 0.00047136, at epoch 137
gradient
Mu = 1e-07, at epoch 137
mu
Validation Checks = 6, at epoch 137
val fail
137 Epochs


Error Histogram with 20 Bins
Training
Validation
Test
Zero Error
Instances
2000
1500
1000
500
0
-0.1616
-0.1434
-0.1251
-0.1068
-0.08855
-0.07028
-0.052
-0.03373
-0.01546
0.002808
0.02108
0.03935
0.05762
0.07589
0.09416
0.1124
0.1307
0.149
0.1672
0.1855
Errors = Targets - Outputs


Training: R=0.99404
Data
Fit
Y = T
Output ~= 0.99*Target + 0.011
Target


Validation: R=0.994
Data
Fit
Y = T
Output ~= 0.99*Target + 0.01
Target


Test: R=0.99439
Data
Fit
Y = T
Output ~= 0.99*Target + 0.01
Target


All: R=0.99409
Data
Fit
Y = T
Output ~= 0.99*Target + 0.01
Target

Training with 20 neurons

Neural Network Training ...

Neural Network

Algorithms

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

Progress

Epoch:	0	137 iterations	1000
Time:		0:00:01	
Performance:	1.61	0.00117	0.00
Gradient:	5.20	0.000549	1.00e-07
Mu:	0.00100	1.00e-06	1.00e+10
Validation Checks:	0	6	6

Plots

Performance (plotperform)
Training State (plottrainstate)
Error Histogram (ploterrhist)
Regression (plotregression)
Fit (plotfit)

Plot Interval: 1 epochs

Neural Fitting (nftool)

Train Network

Train the network to fit the inputs and targets.

Train Network

Choose a training algorithm:

Levenberg-Marquardt

This algorithm typically requires more memory but less time. Training automatically stops when generalization stops improving, as indicated by an increase in the mean square error of the validation samples.

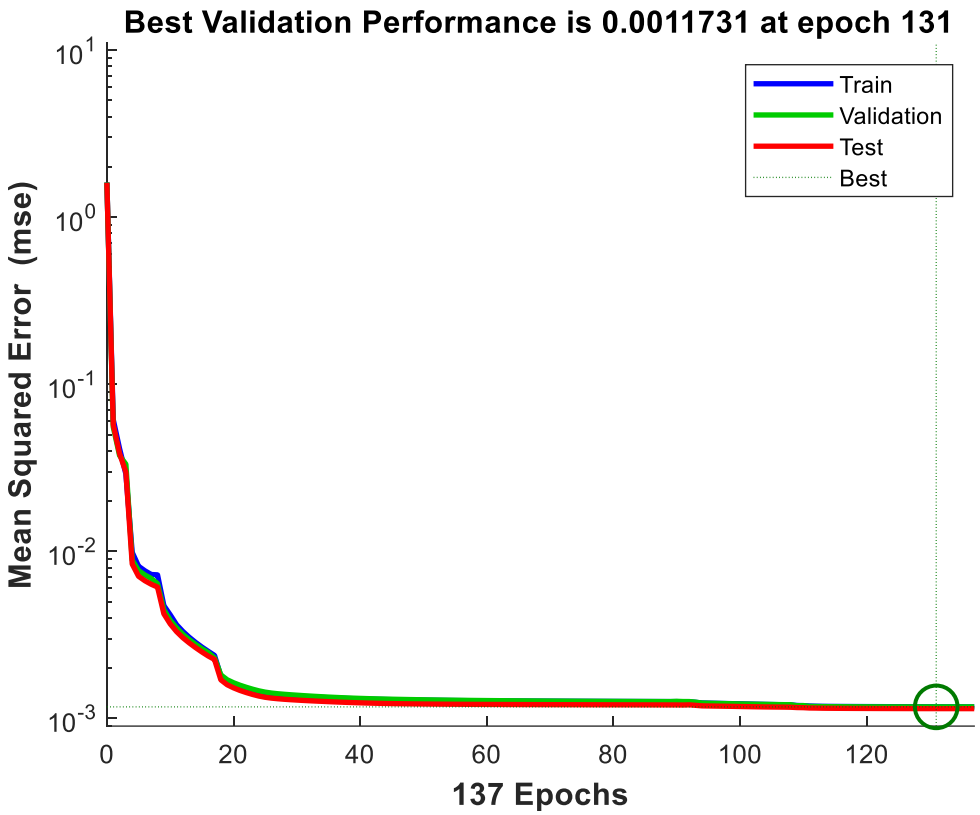
Train using Levenberg-Marquardt. (trainlm)

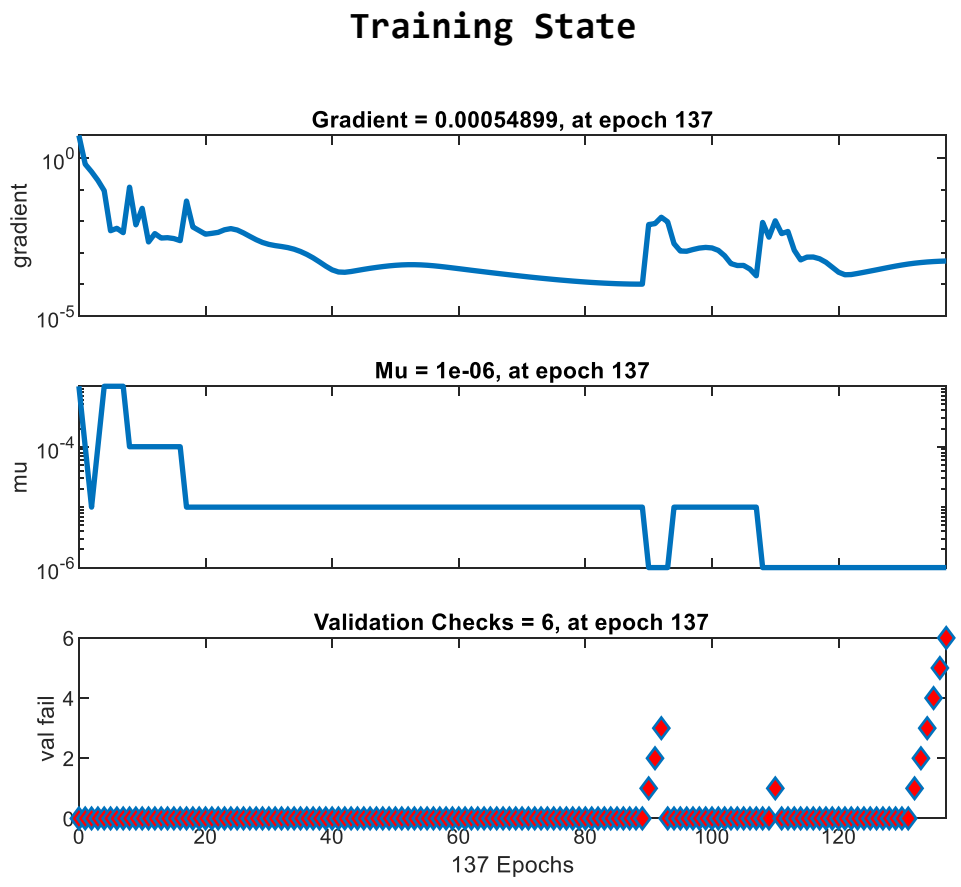
Retrain

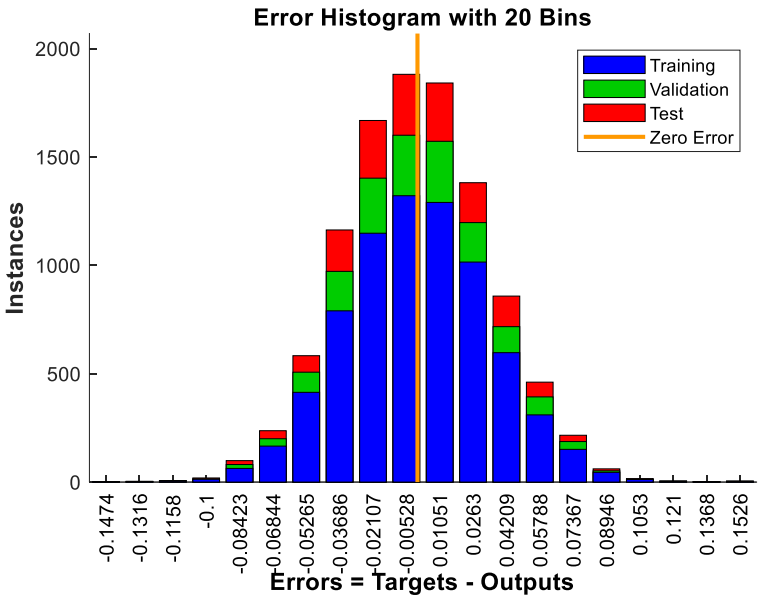
Results

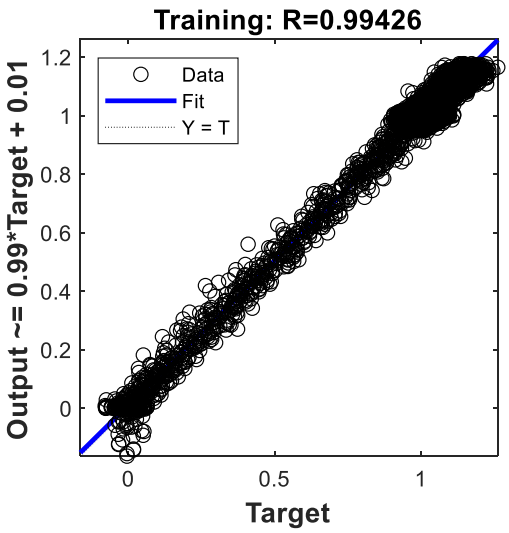
	Samples	MSE	R
Training:	7355	1.17418e-3	9.94256e-1
Validation:	1576	1.17307e-3	9.94551e-1
Testing:	1576	1.14481e-3	9.94454e-1

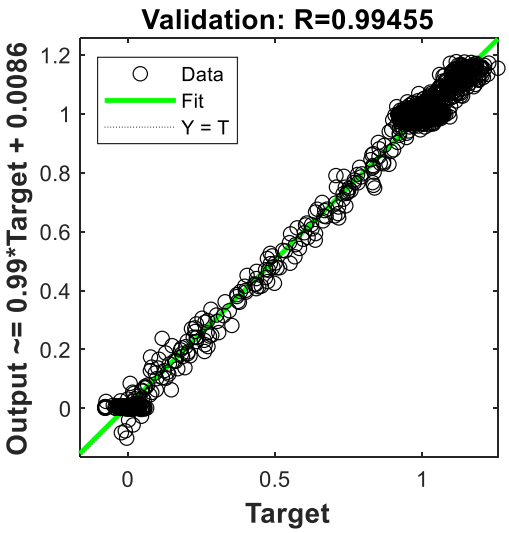
Plot Fit | Plot Error Histogram | Plot Regression

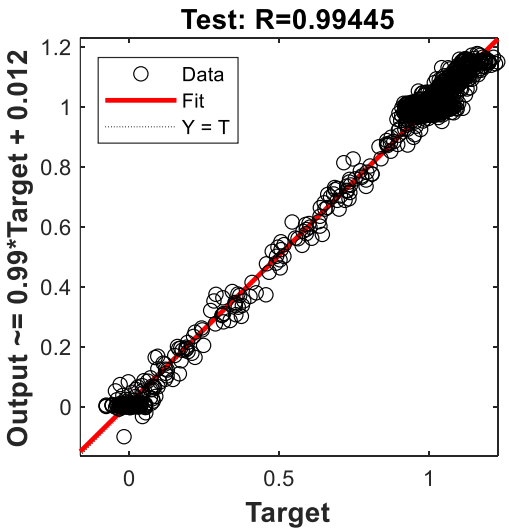

Best Validation Performance is 0.0011731 at epoch 131
Train
Validation
Test
Best
Mean Squared Error (mse)
137 Epochs

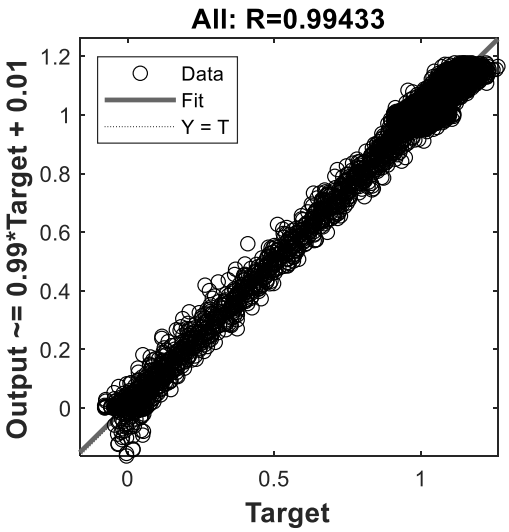

Training State
Gradient = 0.00054899, at epoch 137
gradient
Mu = 1e-06, at epoch 137
mu
Validation Checks = 6, at epoch 137
val fail
137 Epochs


Error Histogram with 20 Bins
Training
Validation
Test
Zero Error
Instances
2000
1500
1000
500
0
-0.1474
-0.1316
-0.1158
-0.1
-0.08423
-0.06844
-0.05265
-0.03686
-0.02107
-0.00528
0.01051
0.0263
0.04209
0.05788
0.07367
0.08946
0.1053
0.121
0.1368
0.1526
Errors = Targets - Outputs

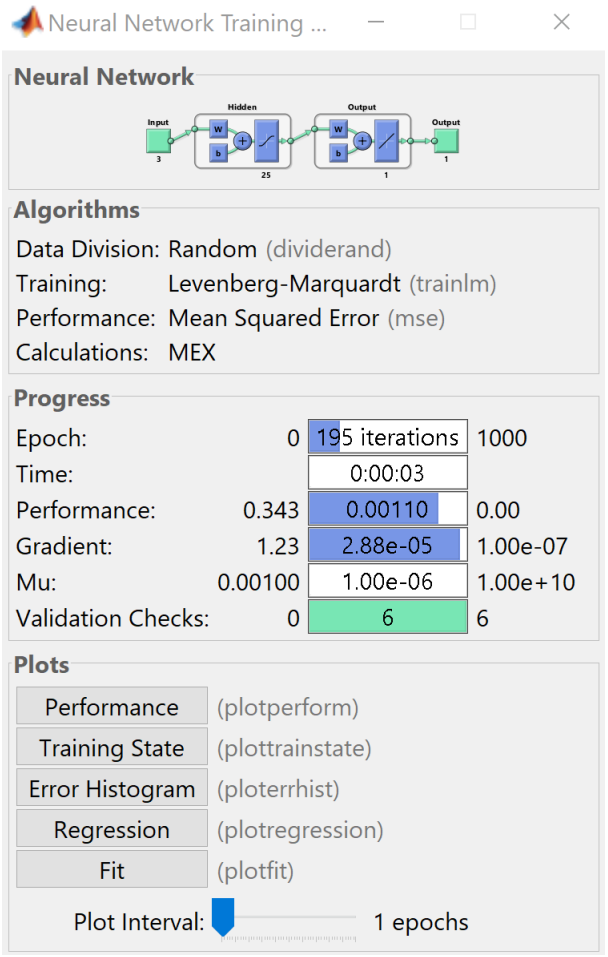

Training: R=0.99426
Data
Fit
Y = T
Output ~= 0.99*Target + 0.01
Target

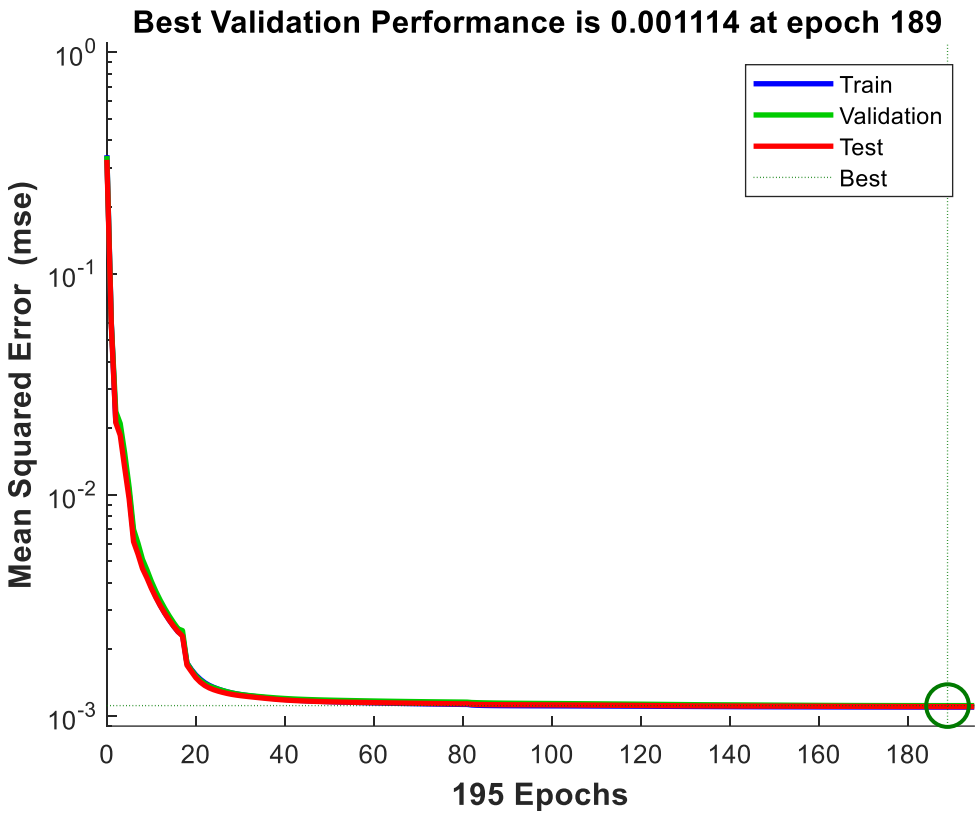

Validation: R=0.99455
Data
Fit
Y = T
Output ~= 0.99*Target + 0.0086
Target

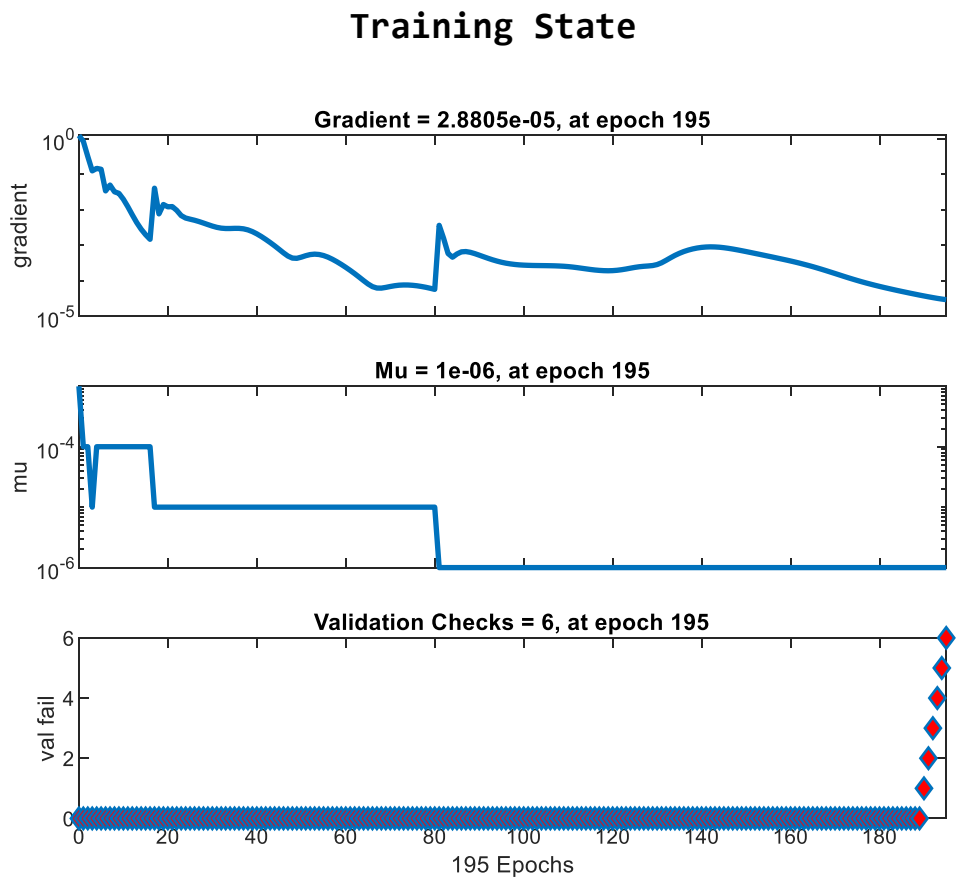

Test: R=0.99445
Data
Fit
Y = T
Output ~= 0.99*Target + 0.012
Target

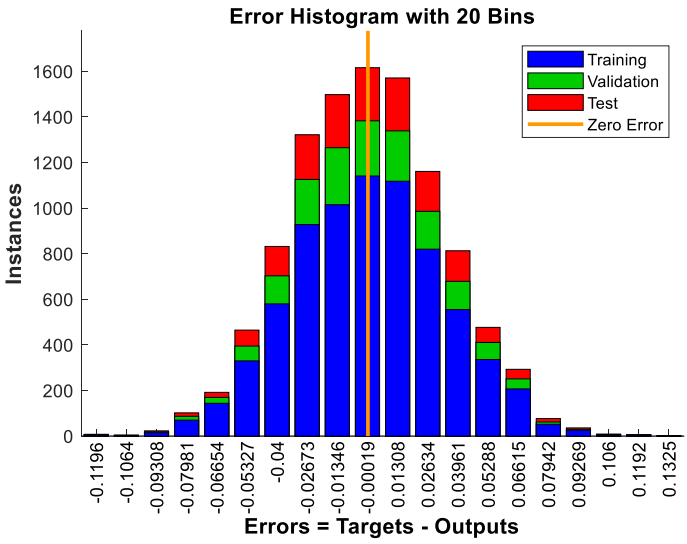

All: R=0.99433
Data
Fit
Y = T
Output ~= 0.99*Target + 0.01
Target

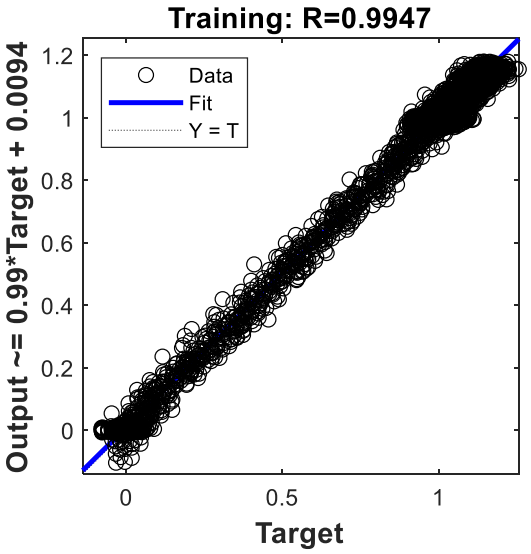
Training with 25 neurons

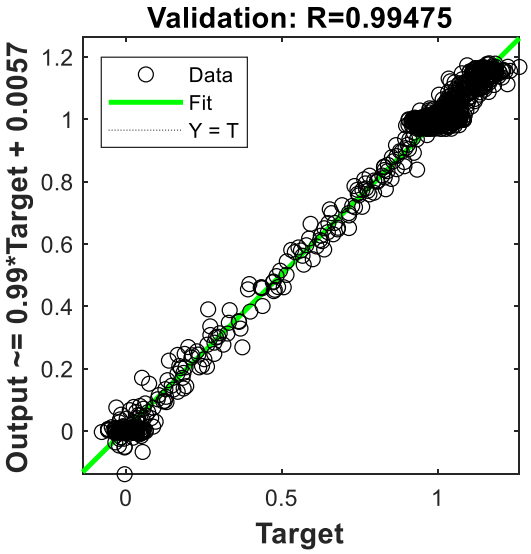


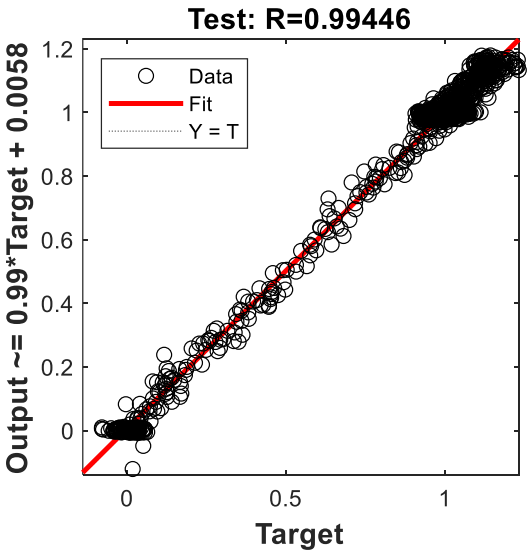

Best Validation Performance is 0.001114 at epoch 189
Train
Validation
Test
Best
Mean Squared Error (mse)
195 Epochs

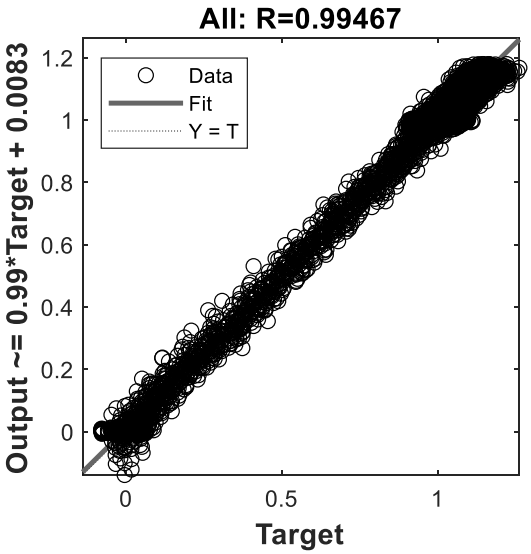

Training State
Gradient = 2.8805e-05, at epoch 195
gradient
Mu = 1e-06, at epoch 195
mu
Validation Checks = 6, at epoch 195
val fail
195 Epochs


Error Histogram with 20 Bins
Training
Validation
Test
Zero Error
Instances
1600
1400
1200
1000
800
600
400
200
0
-0.1196
-0.1064
-0.09308
-0.07981
-0.06654
-0.05327
-0.04
-0.02673
-0.01346
-0.00019
0.01308
0.02634
0.03961
0.05288
0.06615
0.07942
0.09269
0.106
0.1192
0.1325
Errors = Targets - Outputs

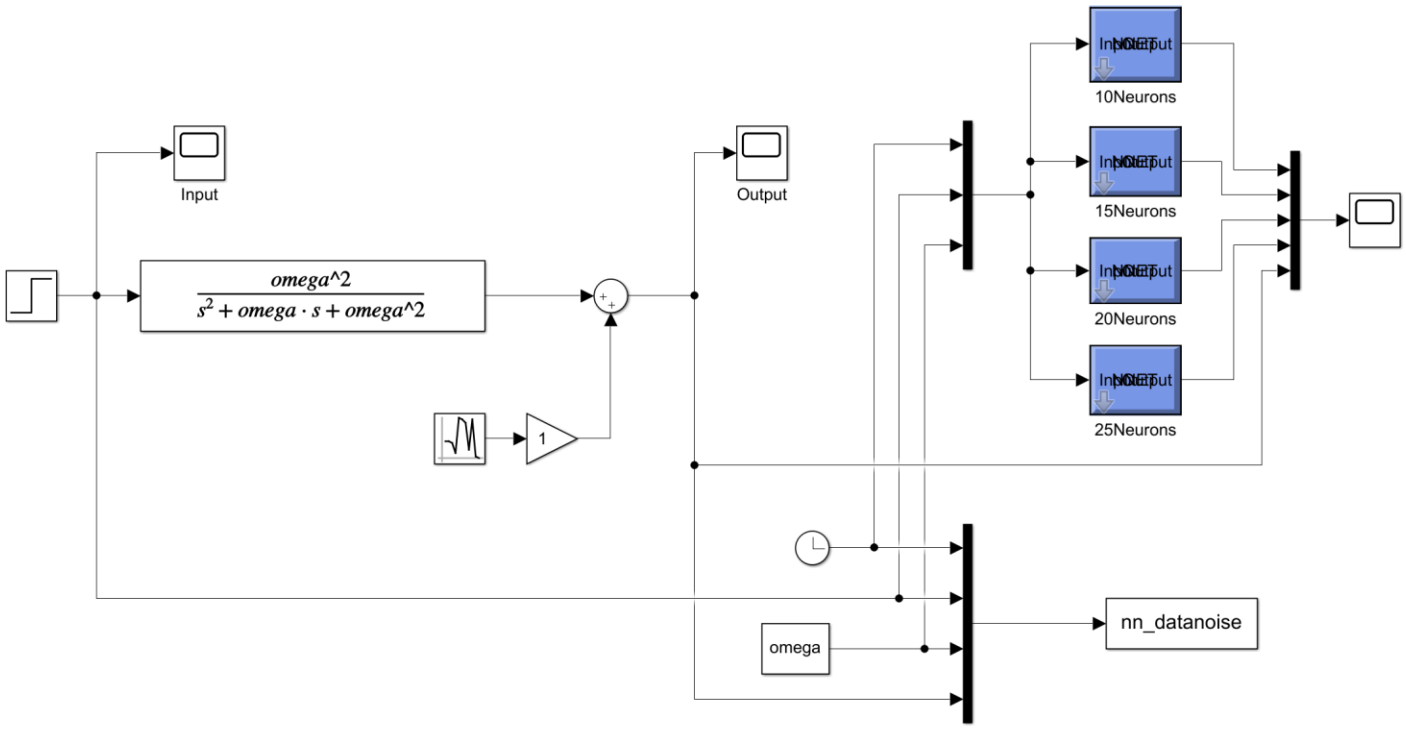

Training: R=0.9947
Data
Fit
Y = T
Output ~= 0.99*Target + 0.0094
Target


Validation: R=0.99475
Data
Fit
Y = T
Output ~= 0.99*Target + 0.0057
Target

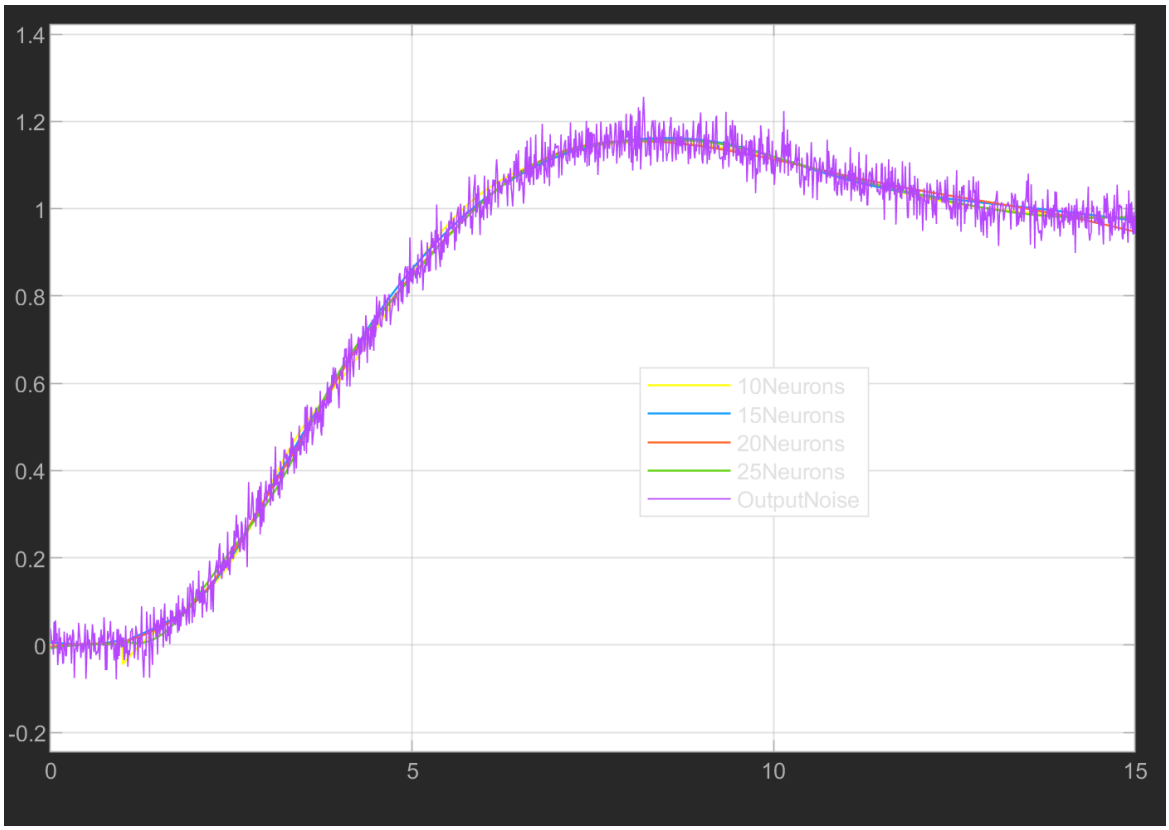

Test: R=0.99446
Data
Fit
Y = T
Output ~= 0.99*Target + 0.0058
Target


All: R=0.99467
Data
Fit
Y = T
Output ~= 0.99*Target + 0.0083
Target

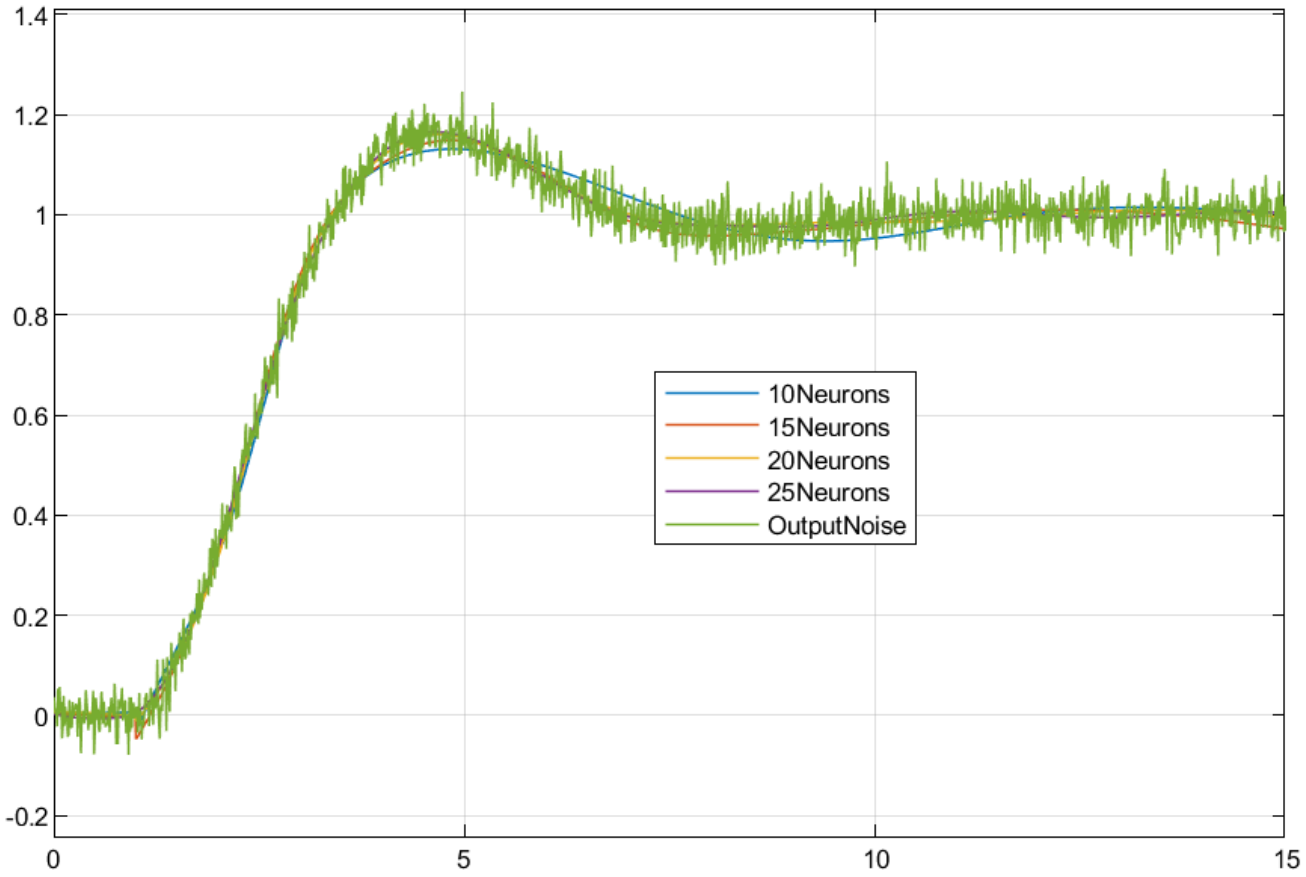
Output Comparison



Simulink Diagram

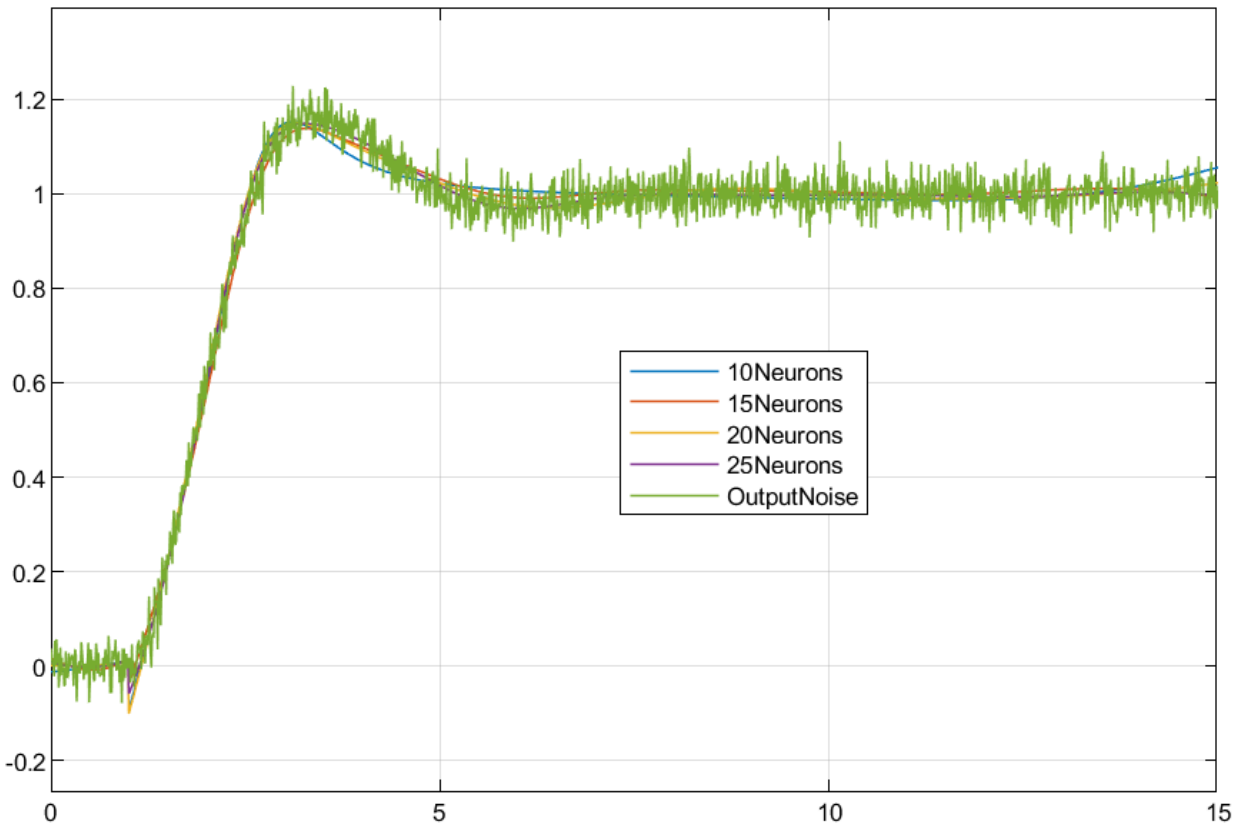


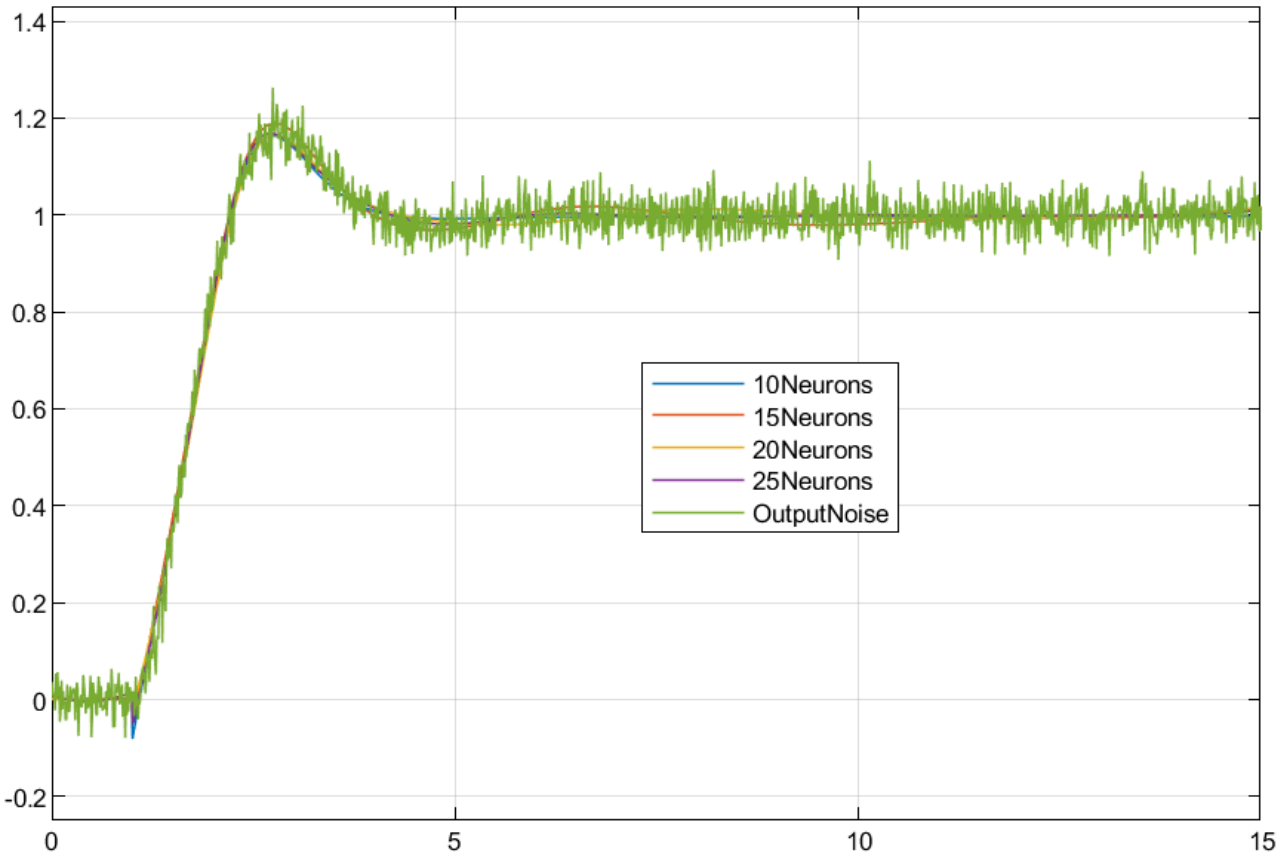
Omega = 0.5



Plot above, omega = 1

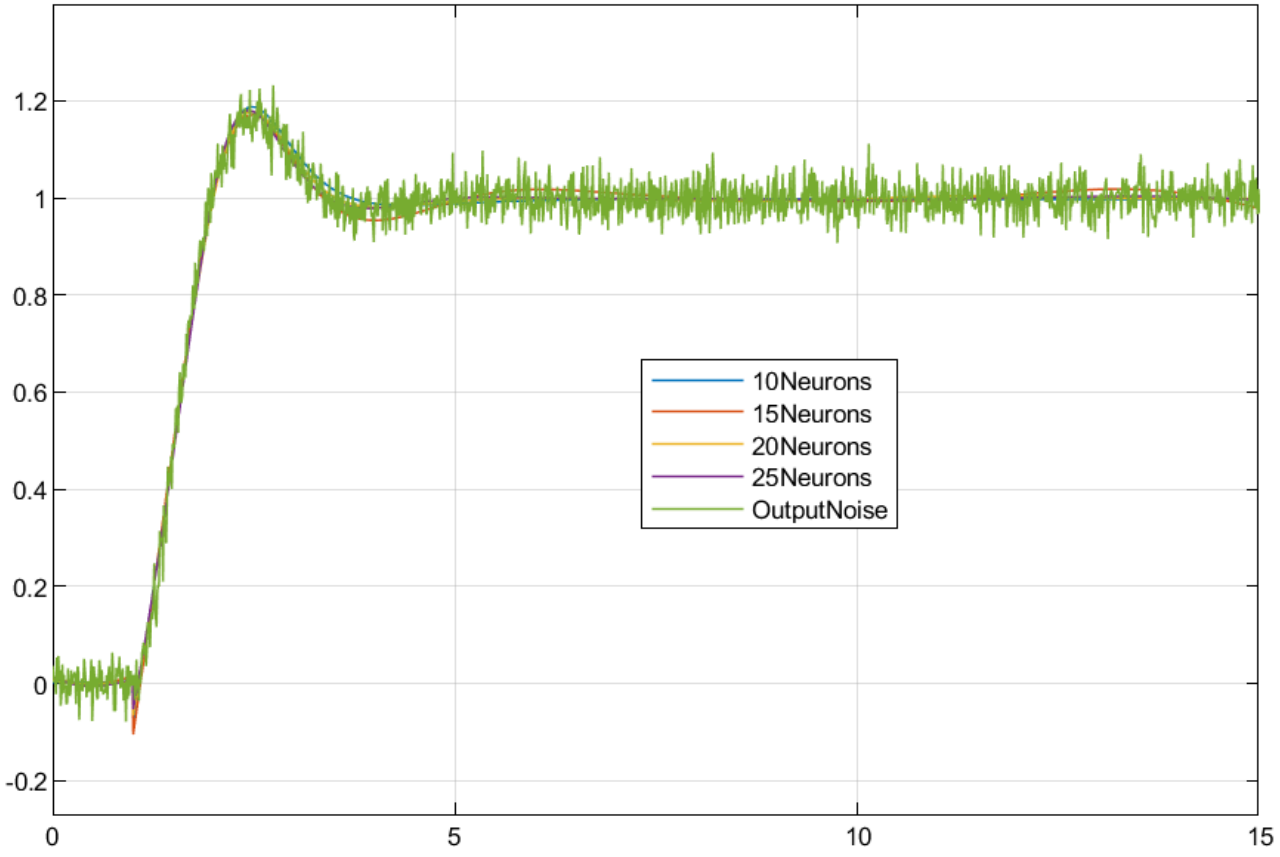
Plot below, omega = 1.5

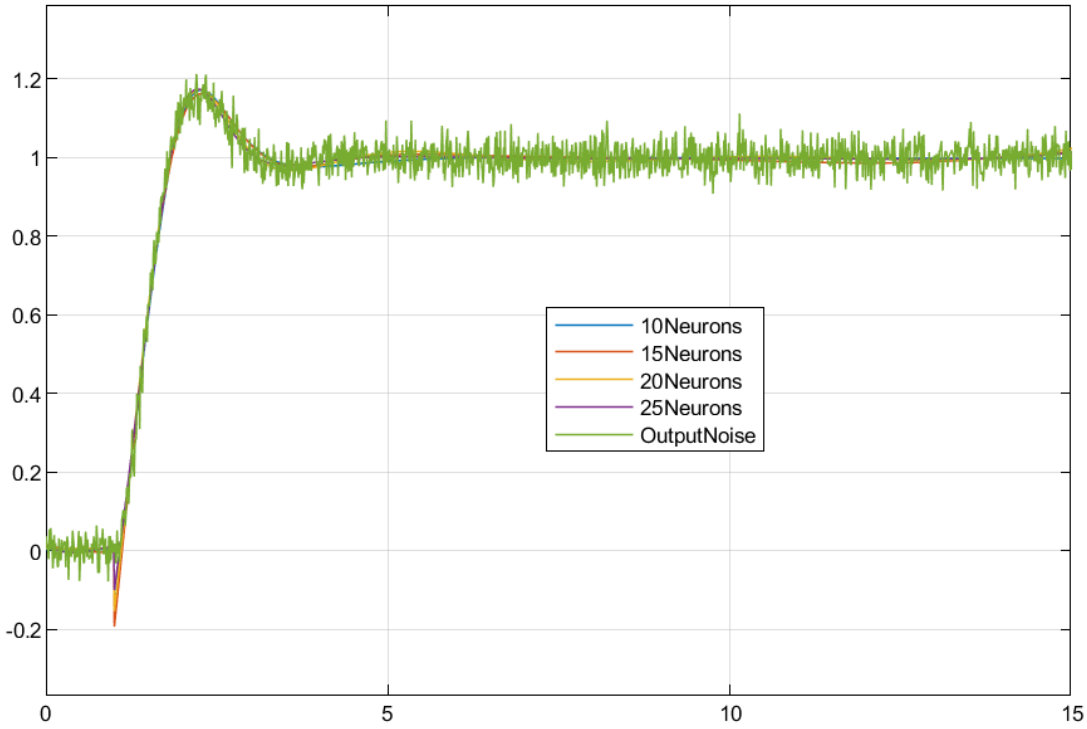




Plot above, omega = 2

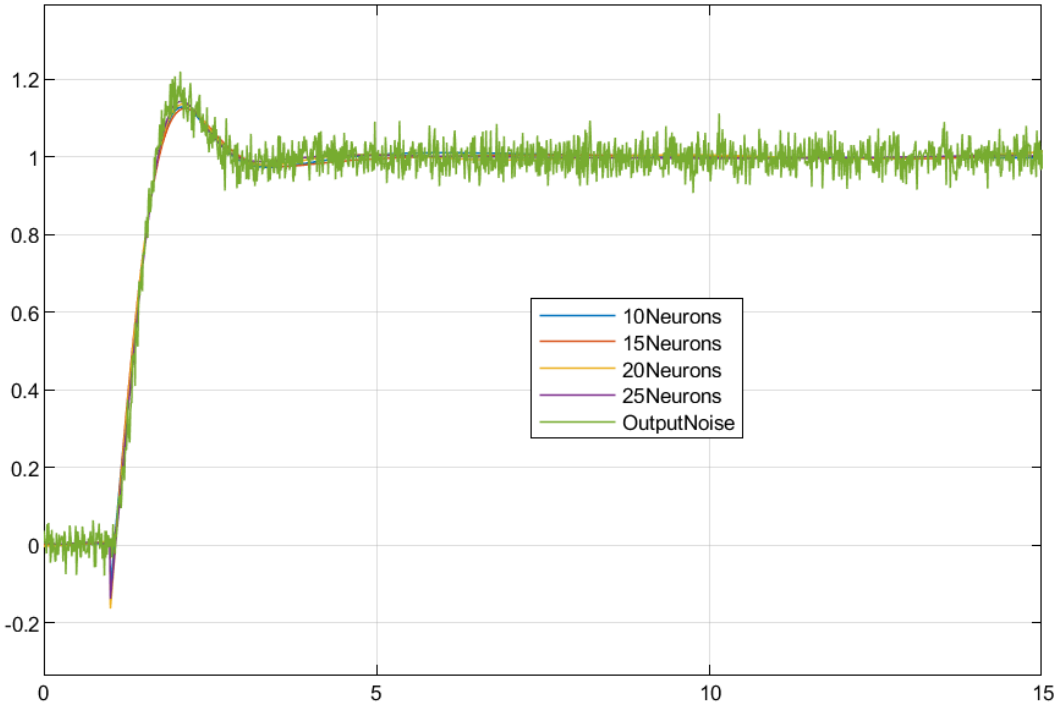
Plot below, omega = 2.5





Plot above, omega = 3

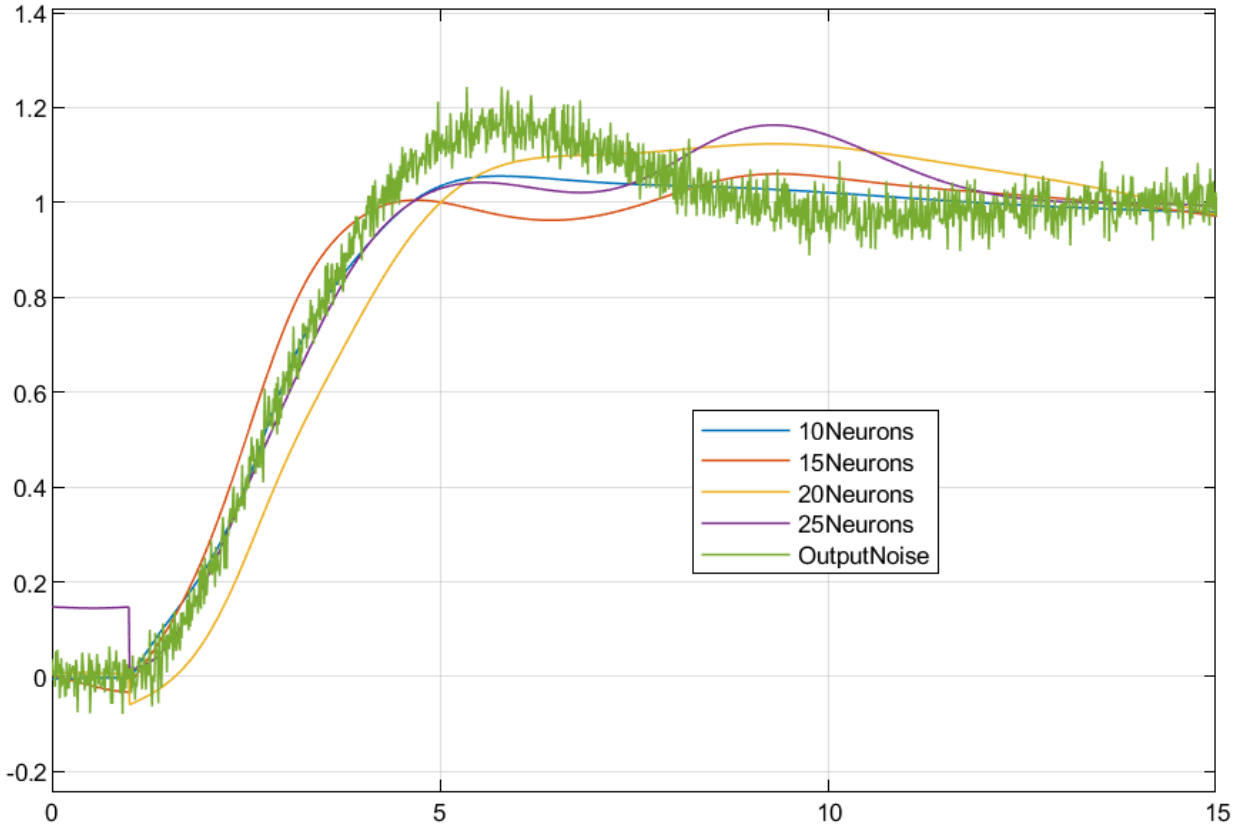
Plot below, omega = 3.5



Conclusion – Based on the validation data performance and plots above, increasing the number of neurons improves the fit. This is supported by the best performance for validation data and the general performance (in the Neural Network Training window) metrics.

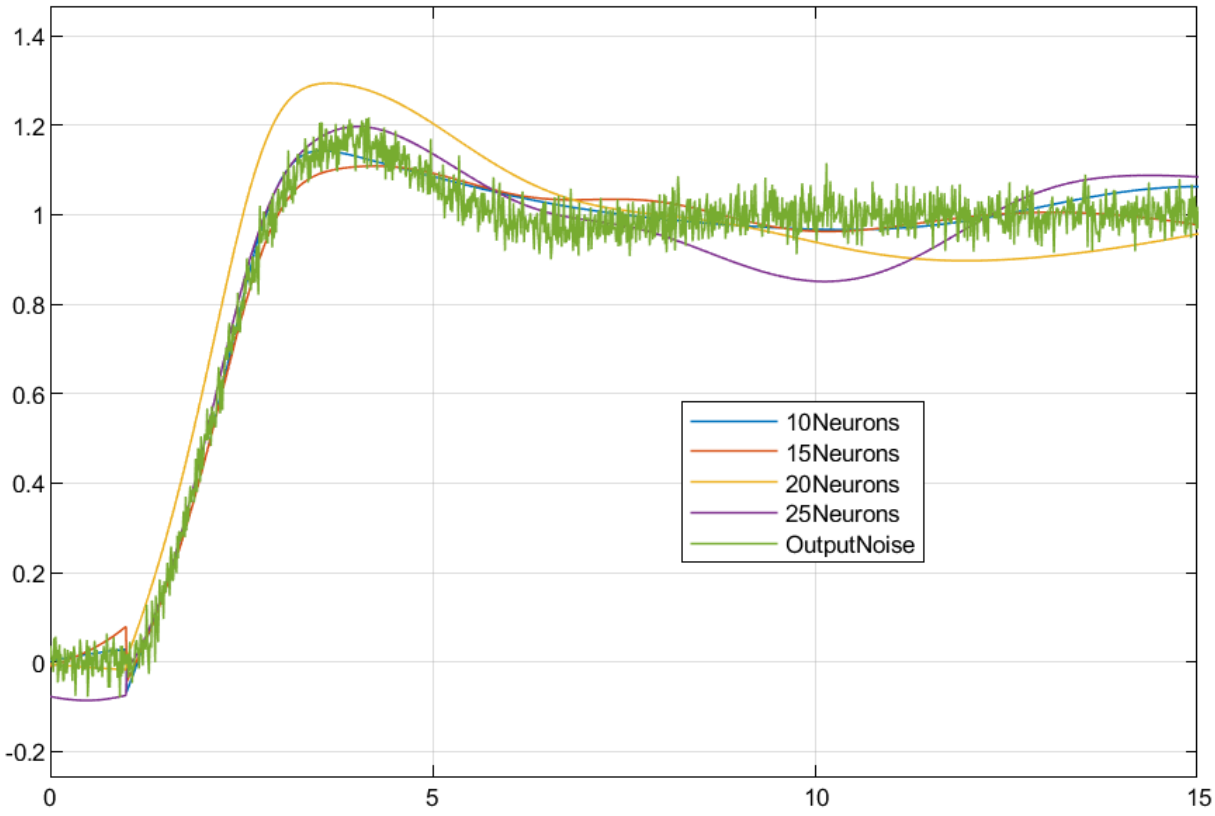
Other than that, the output is slightly erroneous in the transition state but is very good once steady state is reached for almost all neural networks. However, it is notable that other than the neural network with 10 Neurons, the output dips downwards just before the step input. The dipping amplitude decreases with the increase of neurons.

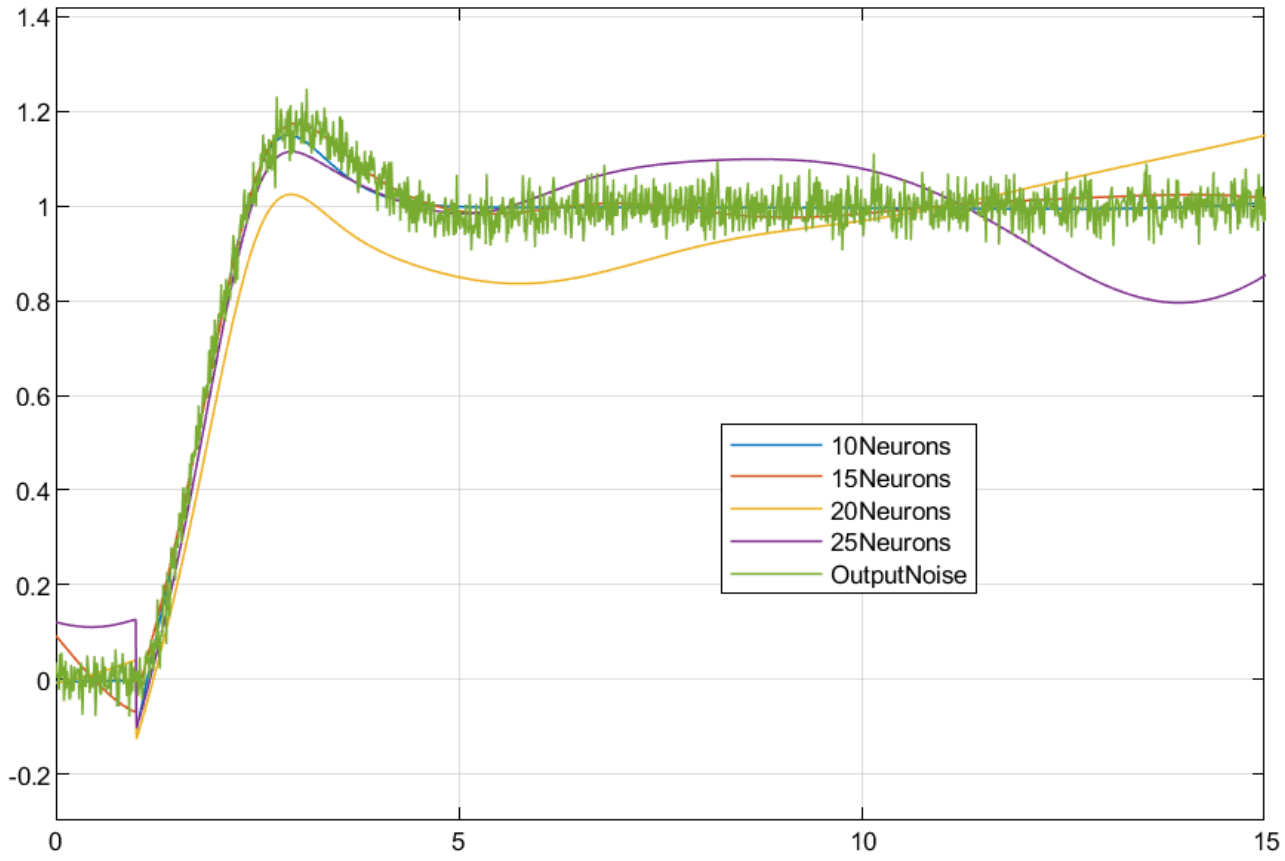
b. Interpolation



Plot above, omega = 0.75

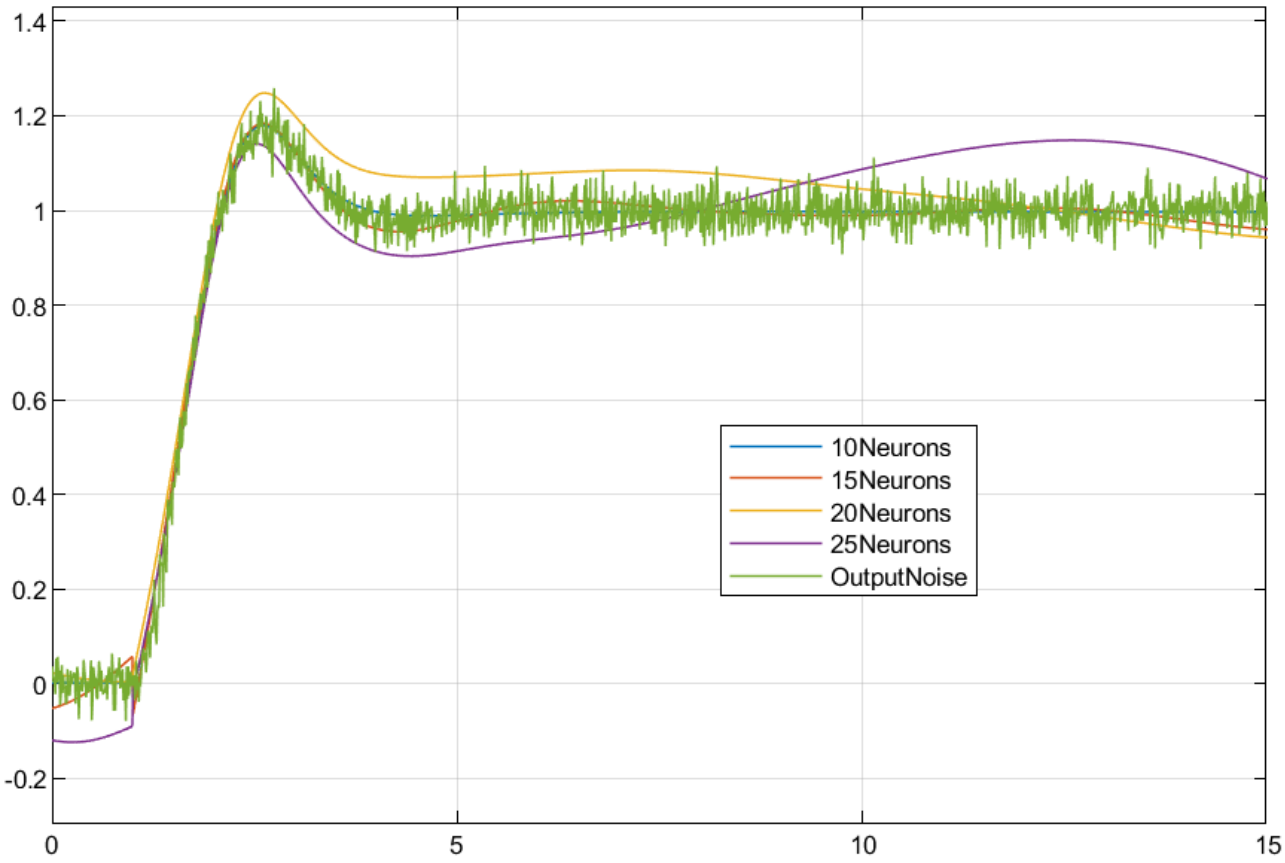
Plot below, omega = 1.25

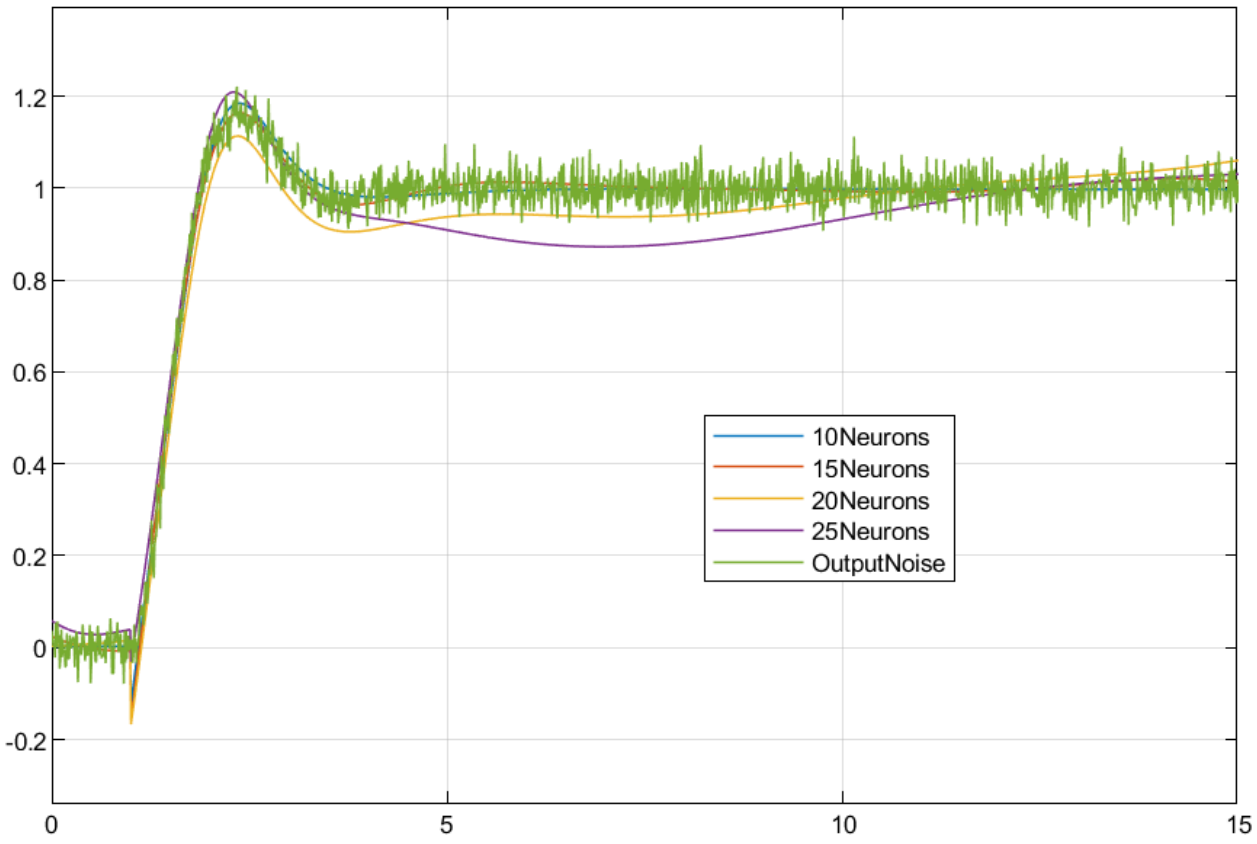




Plot above, omega = 1.75

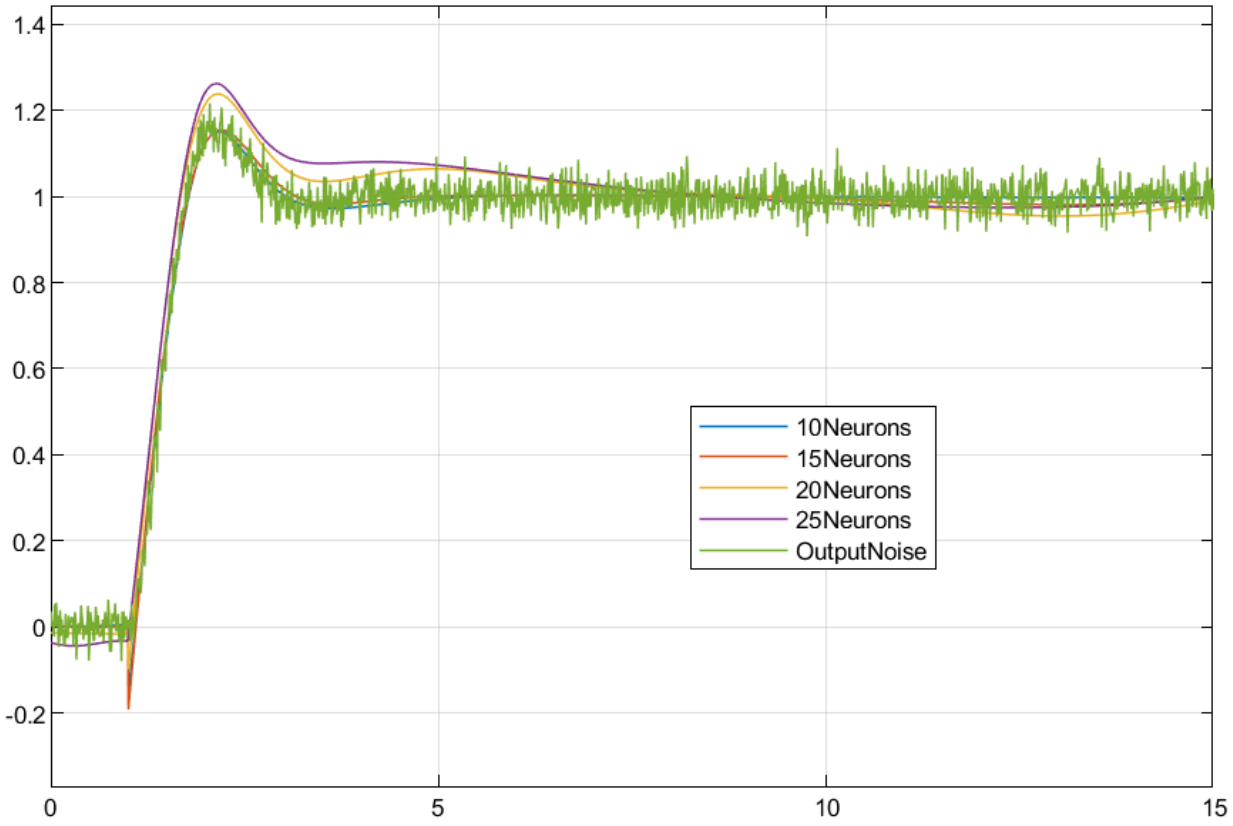
Plot below, omega = 2.25





Plot above, omega = 2.75

Plot below, omega = 3.25



Conclusion – Similar to the previous scenario without noise, it is observed that smaller frequencies are tougher to interpolate as compared to larger frequencies. The neural network with 10 neurons does the best job at interpolation as it provides the nearest output at omega = 0.75. More number of neurons may cause overfitting which is why more neurons lead to less efficient interpolation.

C. The neural network with 10 neurons does a good job of interpolating frequency at lower as well as higher frequencies. Thus, **neural networks with around 10 neurons** will work effectively for this system if they are trained using inputs and outputs over all the operating frequencies. If the operating frequency doesn't change or stays constant at regions used for training data then more neurons may be good. Note that more neurons are good at fitting but they are not so good at interpolation for this system. Also, none of the neural networks is really close to the system so they might not be usable if the acceptable errors in output are less than 10^-3. Although noise seems to have no effect on the ability of network to model the data, its effect can be clearly seen when comparing the best validation performance and the mean squared error values. Thus, **noise reduces the fitting ability of the network** for this system.