IMPLEMENTING OPTIMIZATION TECHNIQUES ON NEURAL NETWORK BACKPROPAGATION

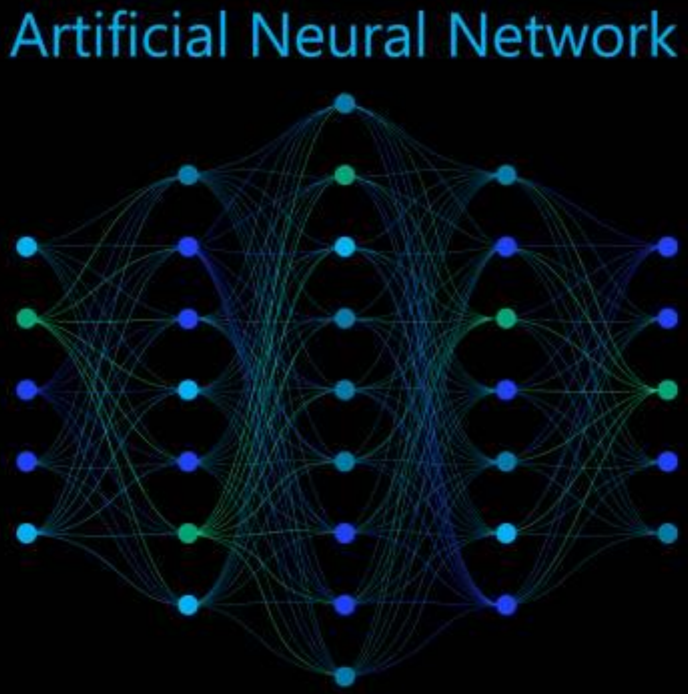
BY TEAM NINE

ADITYA KUMAR,ADITYA NAYAK,MEDHASWETA SEN

Introduction

Project Scope:

- Initial ambition
- Final achievement



DATASETS

- Three types of datasets considered to be implemented on our neural network:

1. **Small: Housing dataset**

- Source : Kaggle
- Dataset based on data from the 1990 California census.
- Includes features such as population, median income, no. of bedrooms etc.
- Features can be used to predict the price of each house.
- Using this dataset in order to check if small features give optimal results by the neural network.

2. Medium: Spotify - Youtube dataset

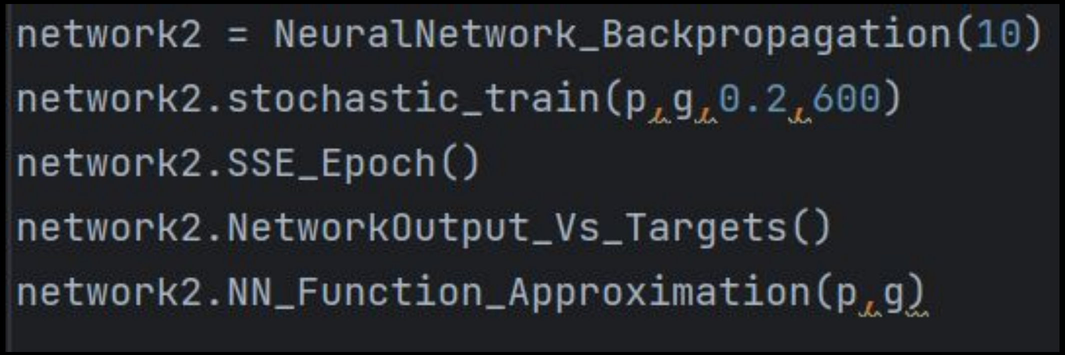
- Source : Kaggle
- Dataset of songs of various artist in the world and for each song is present.
- Number of views of the official music video of the song on YouTube.
- The dataset consists of features like Valence, Tempo, Duration etc. which can be used to predict the no. of Likes/Views the music receives.

3. Large: Heart Cancer dataset

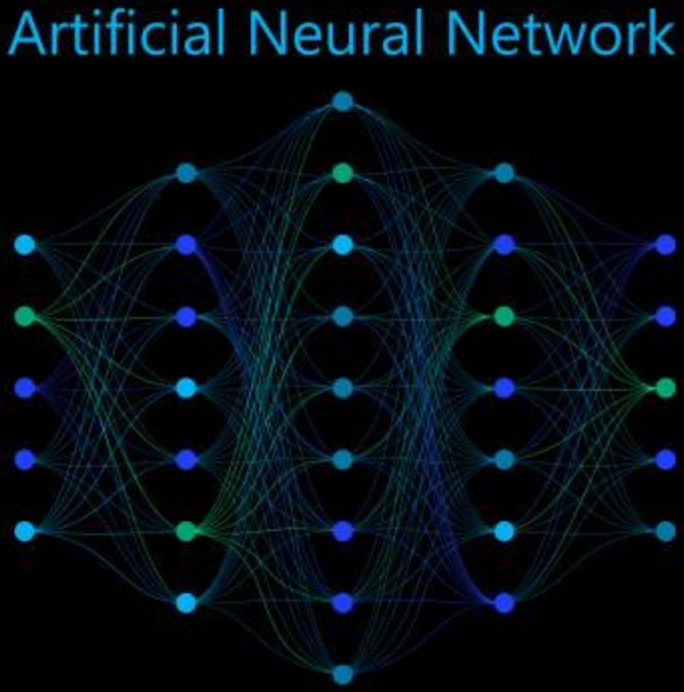
- Source: Kaggle
- The following dataset consists of 33 predictor variables such as incident_rate, medincome, povertypercent etc.
- These features can be used to predict the deathrate caused by cancer throughout the US.
- No further description of the data variables was provided on Kaggle.

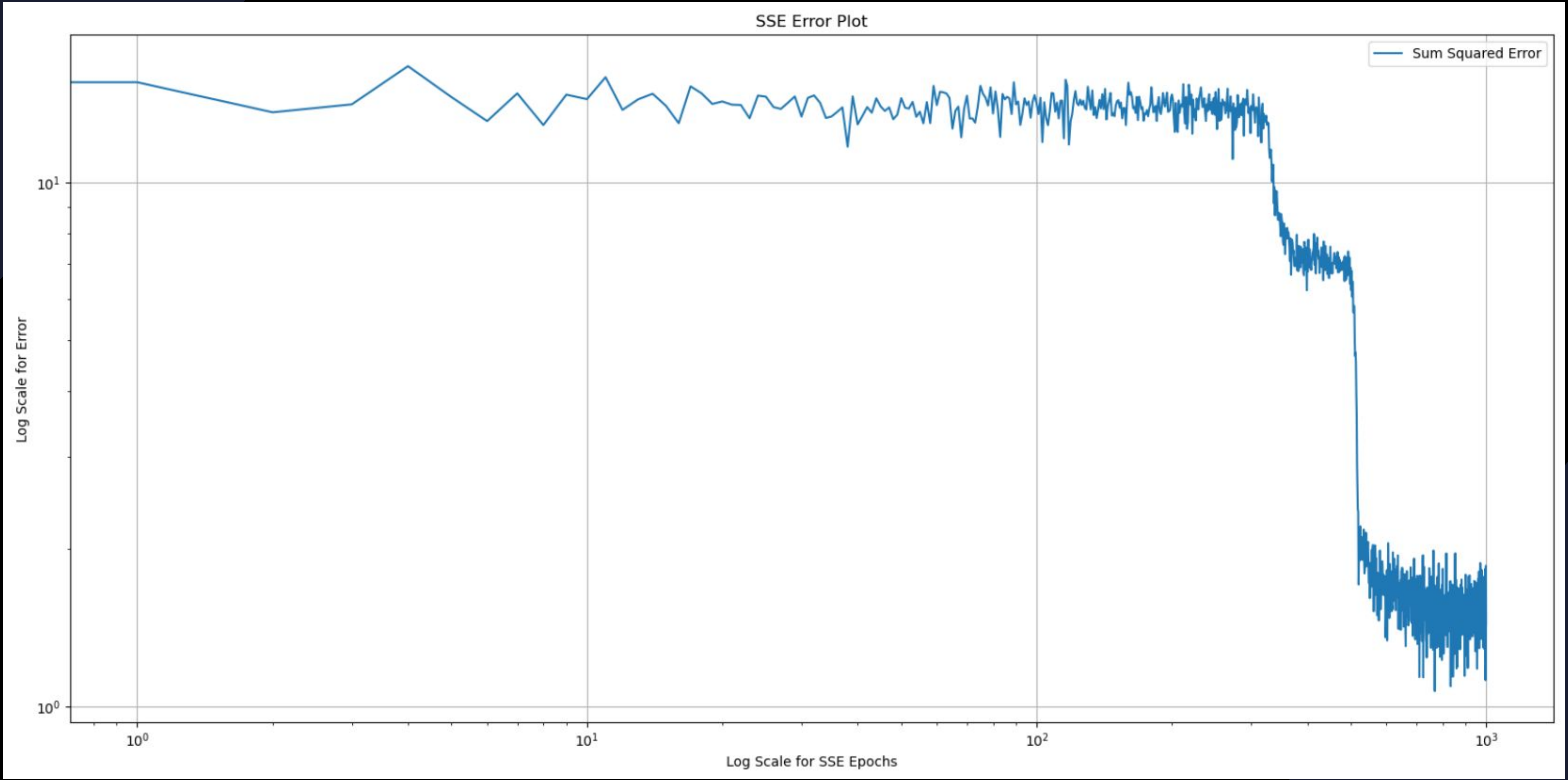
Coding Journey

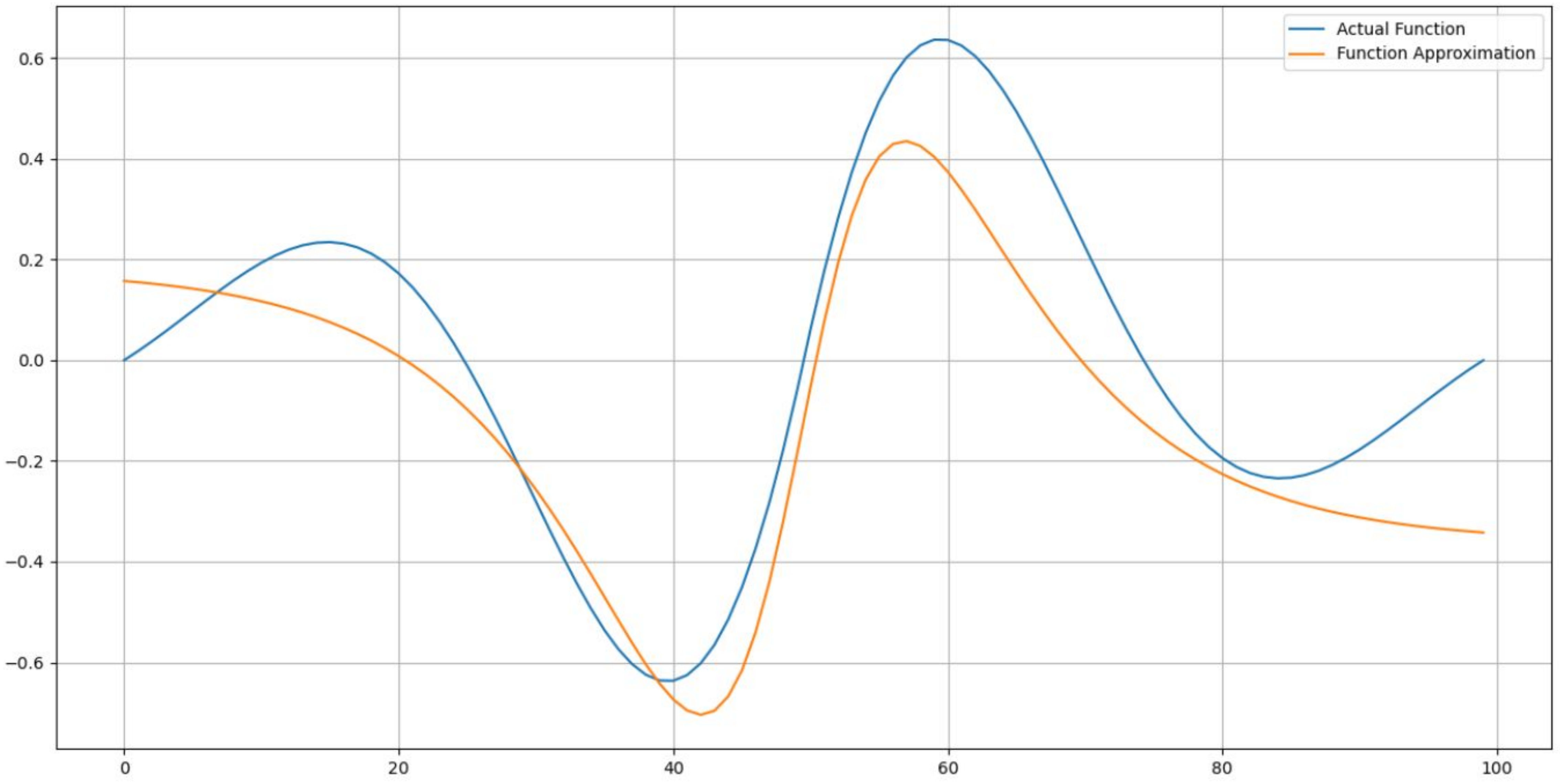
Homework 6: Where it all began

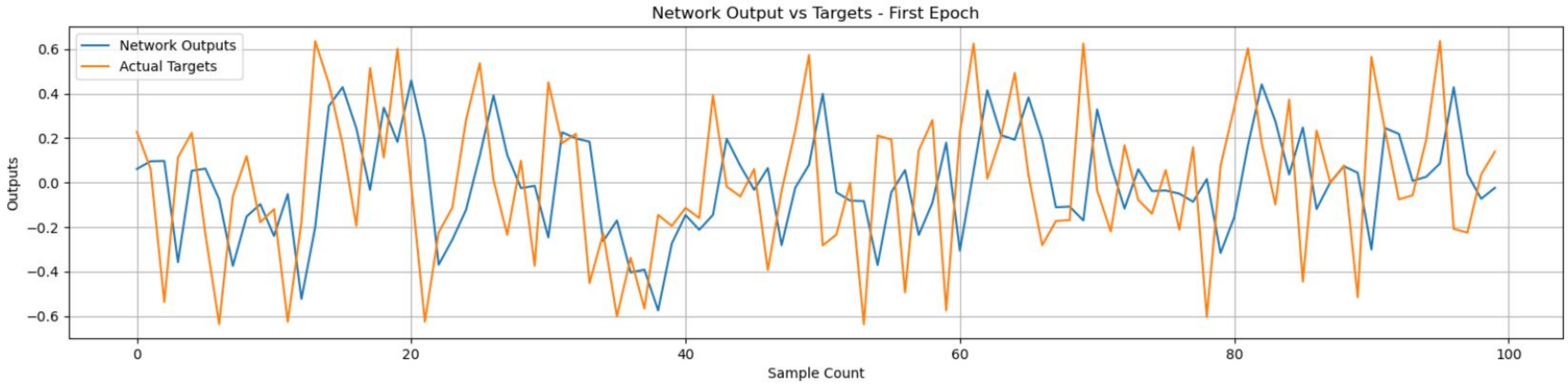


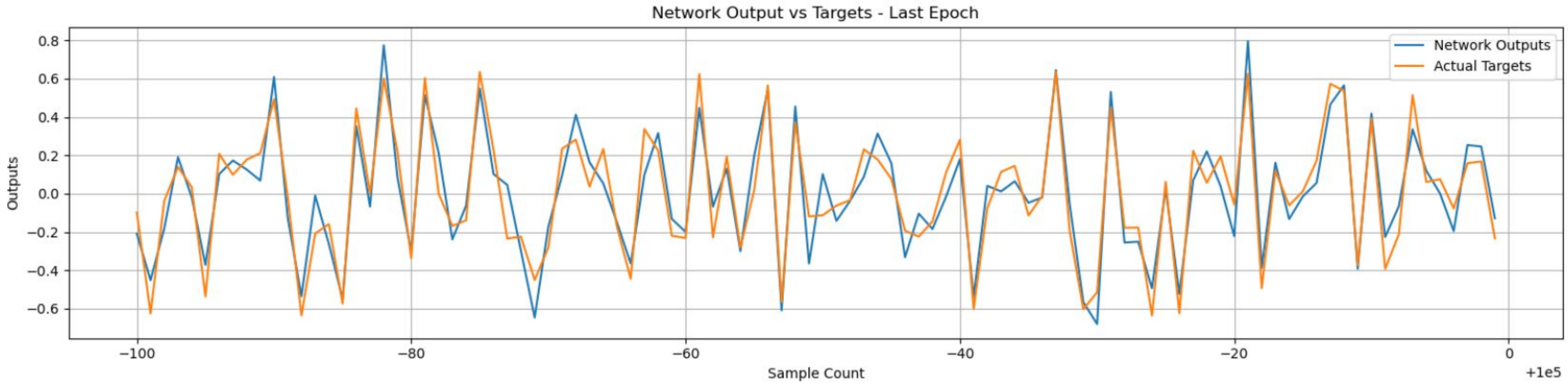
```
network2 = NeuralNetwork_Backpropagation(10)
network2.stochastic_train(p,g,0.2,600)
network2.SSE_Epoch()
network2.NetworkOutput_Vs_Targets()
network2.NN_Function_Approximation(p,g)
```

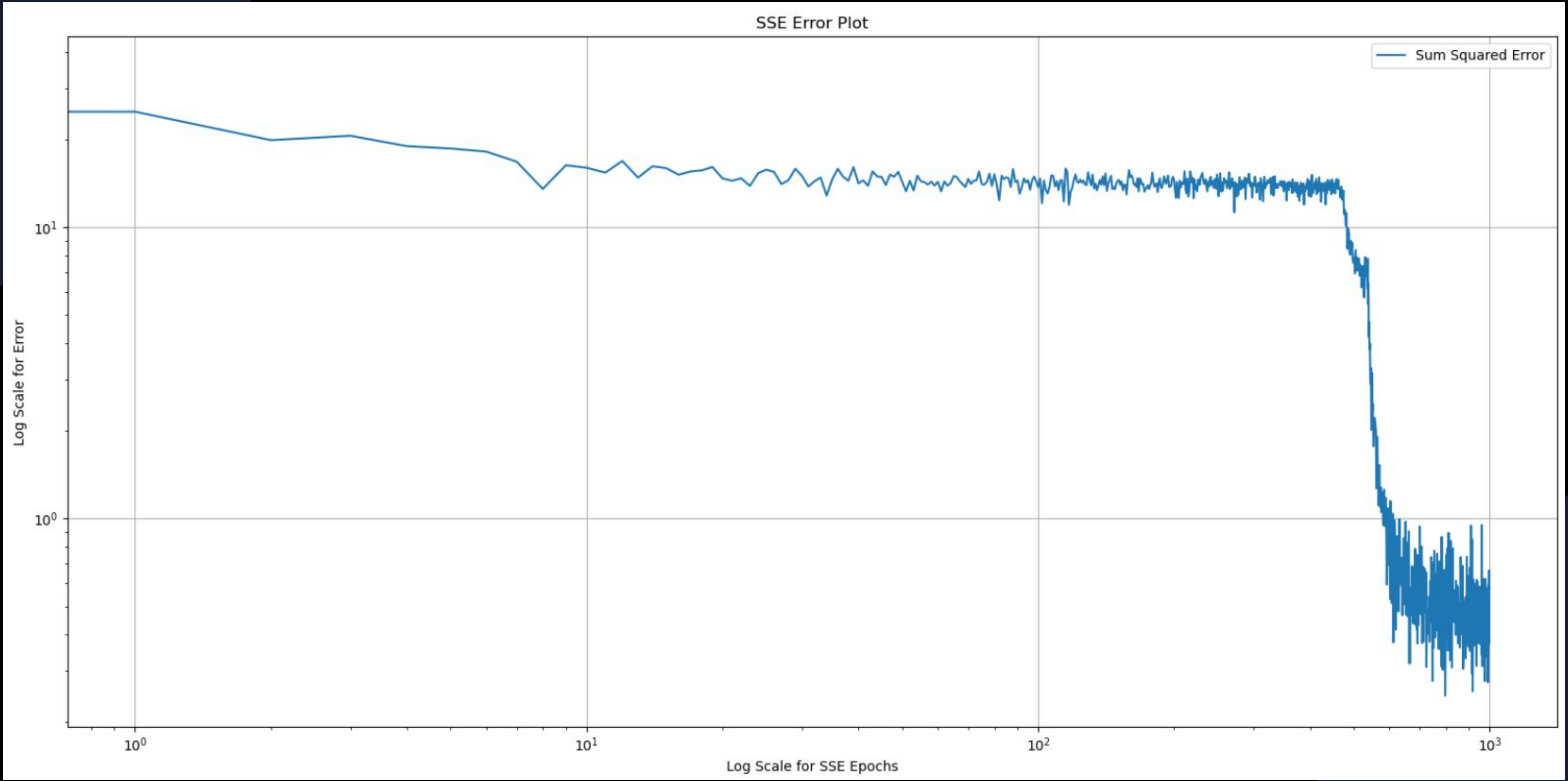


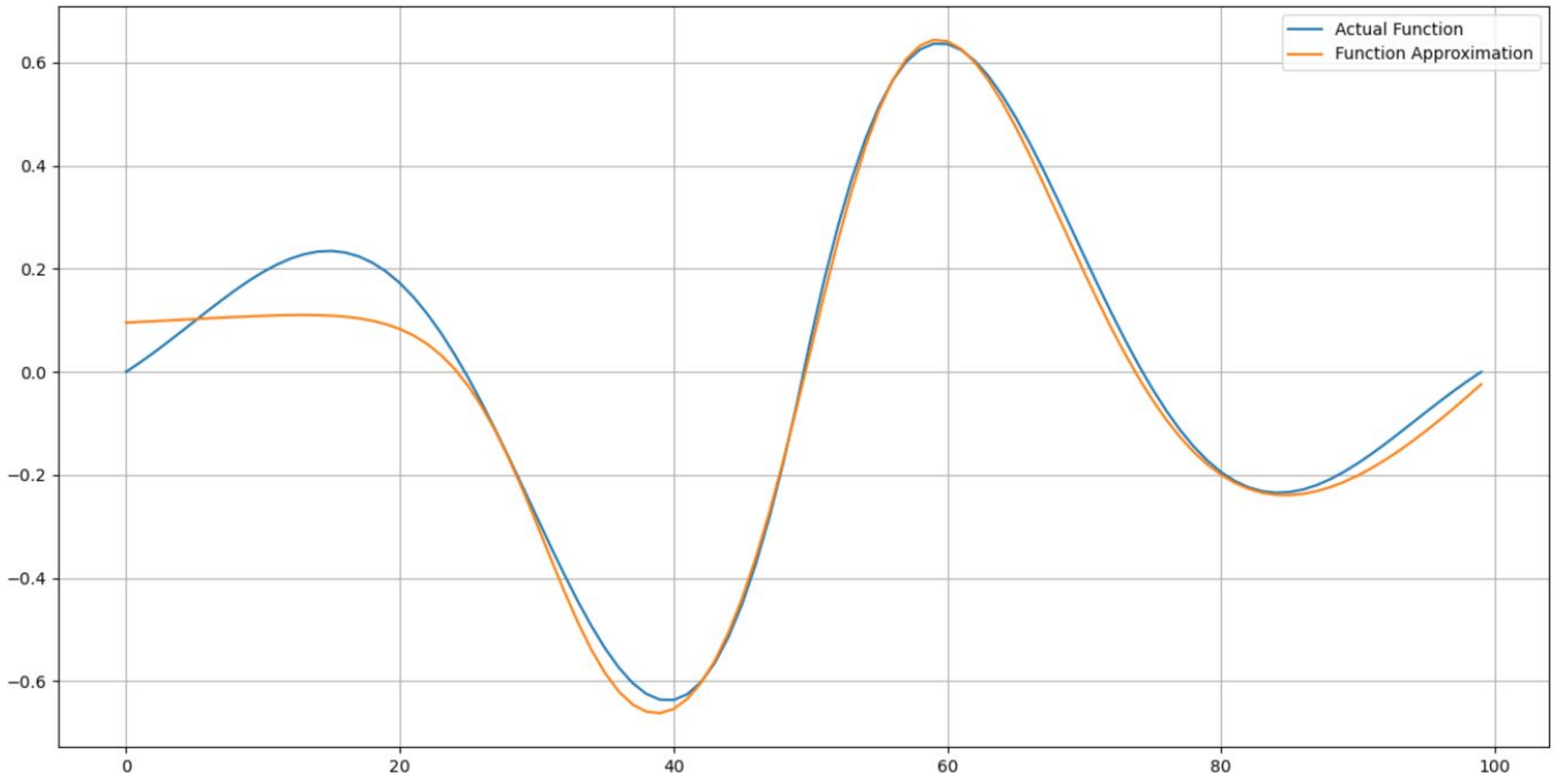

SSE Error Plot
Sum Squared Error
Log Scale for Error
Log Scale for SSE Epochs
10^1
10^0
10^0
10^1
10^2
10^3

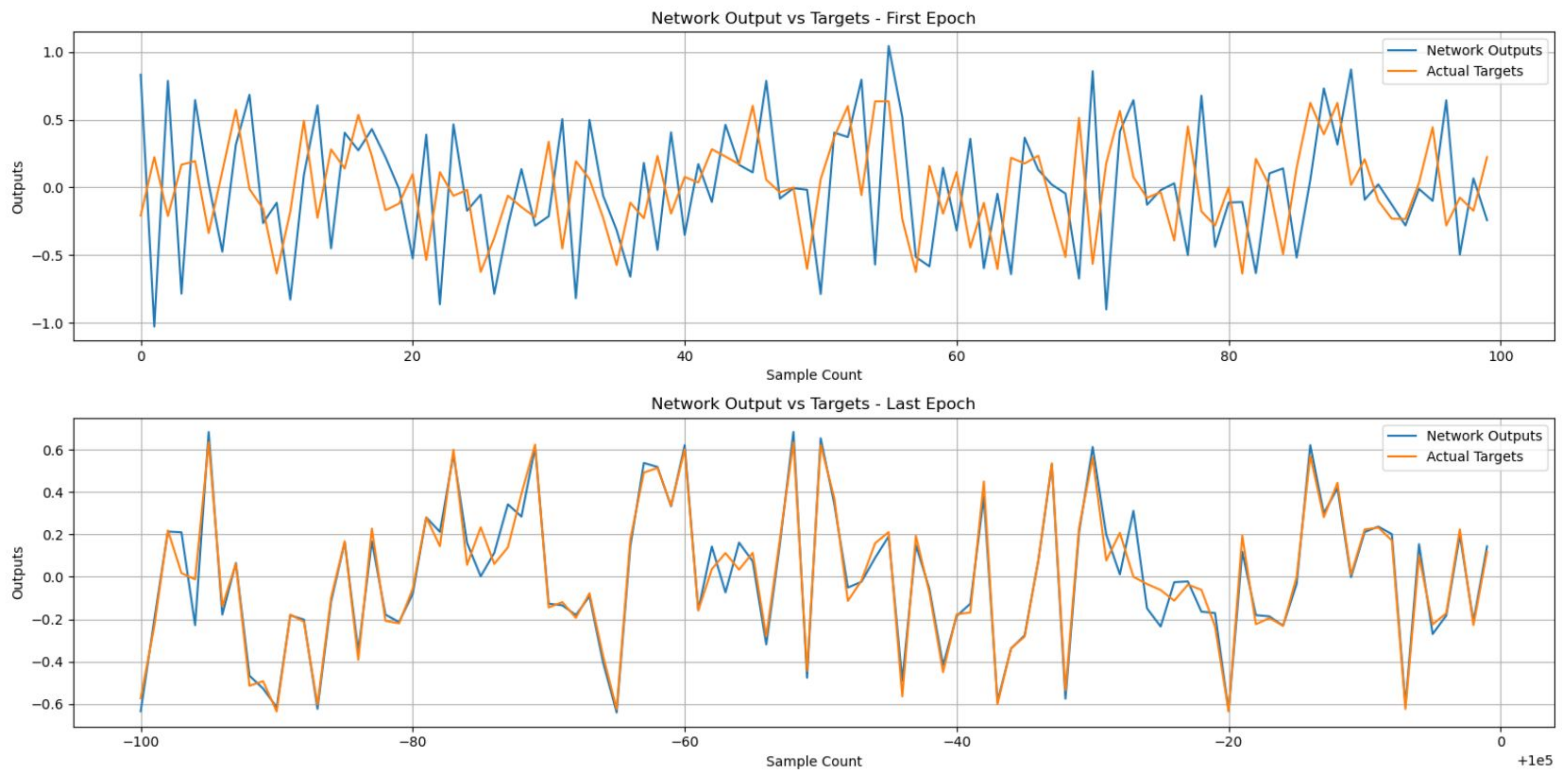

Actual Function
Function Approximation
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
20
40
60
80
100


Network Output vs Targets - First Epoch
Network Outputs
Actual Targets
Outputs
Sample Count
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
20
40
60
80
100


Network Output vs Targets - Last Epoch
Network Outputs
Actual Targets
Outputs
Sample Count
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−100
−80
−60
−40
−20
0
+1e5

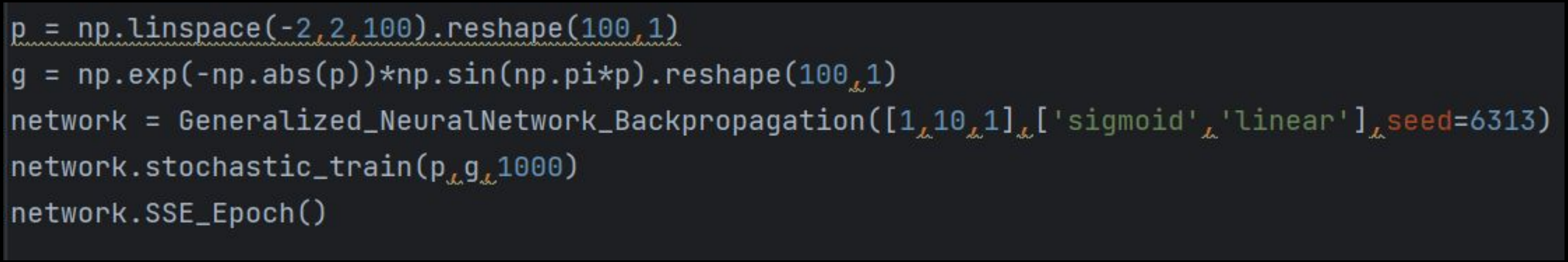

SSE Error Plot
Sum Squared Error
Log Scale for Error
Log Scale for SSE Epochs
10^1
10^0
10^0
10^1
10^2
10^3


Actual Function
Function Approximation
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
20
40
60
80
100


Network Output vs Targets - First Epoch
Outputs
Sample Count
Network Outputs
Actual Targets
Network Output vs Targets - Last Epoch
Outputs
Sample Count
+1e5

Motivation

- Why not general neural network?



```
p = np.linspace(-2,2,100).reshape(100,1)
g = np.exp(-np.abs(p))*np.sin(np.pi*p).reshape(100,1)
network = Generalized_NeuralNetwork_Backpropagation([1,10,1],['sigmoid','linear'],seed=6313)
network.stochastic_train(p,g,1000)
network.SSE_Epoch()
```

- Challenges?

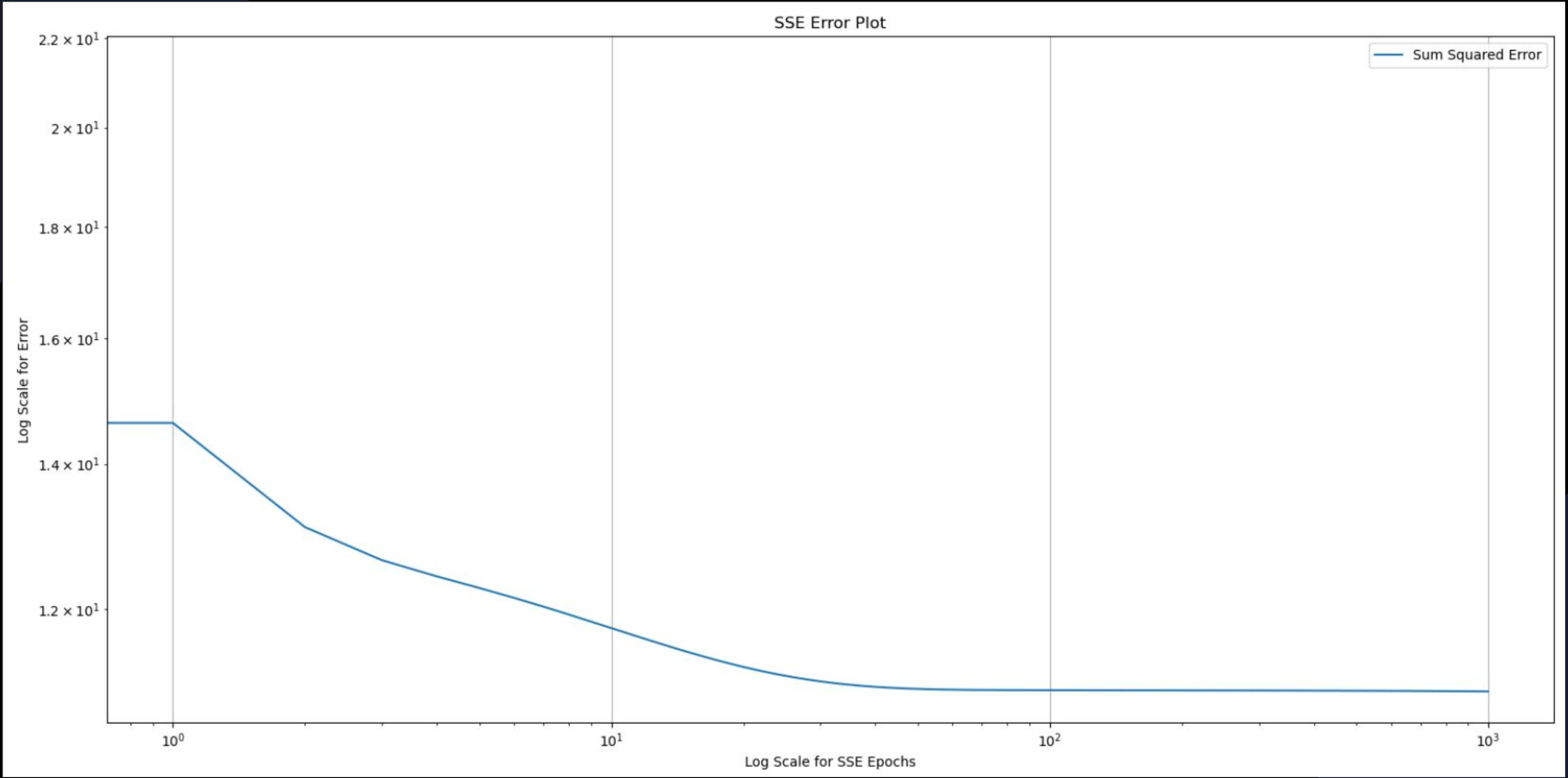
Stochastic Training ✓

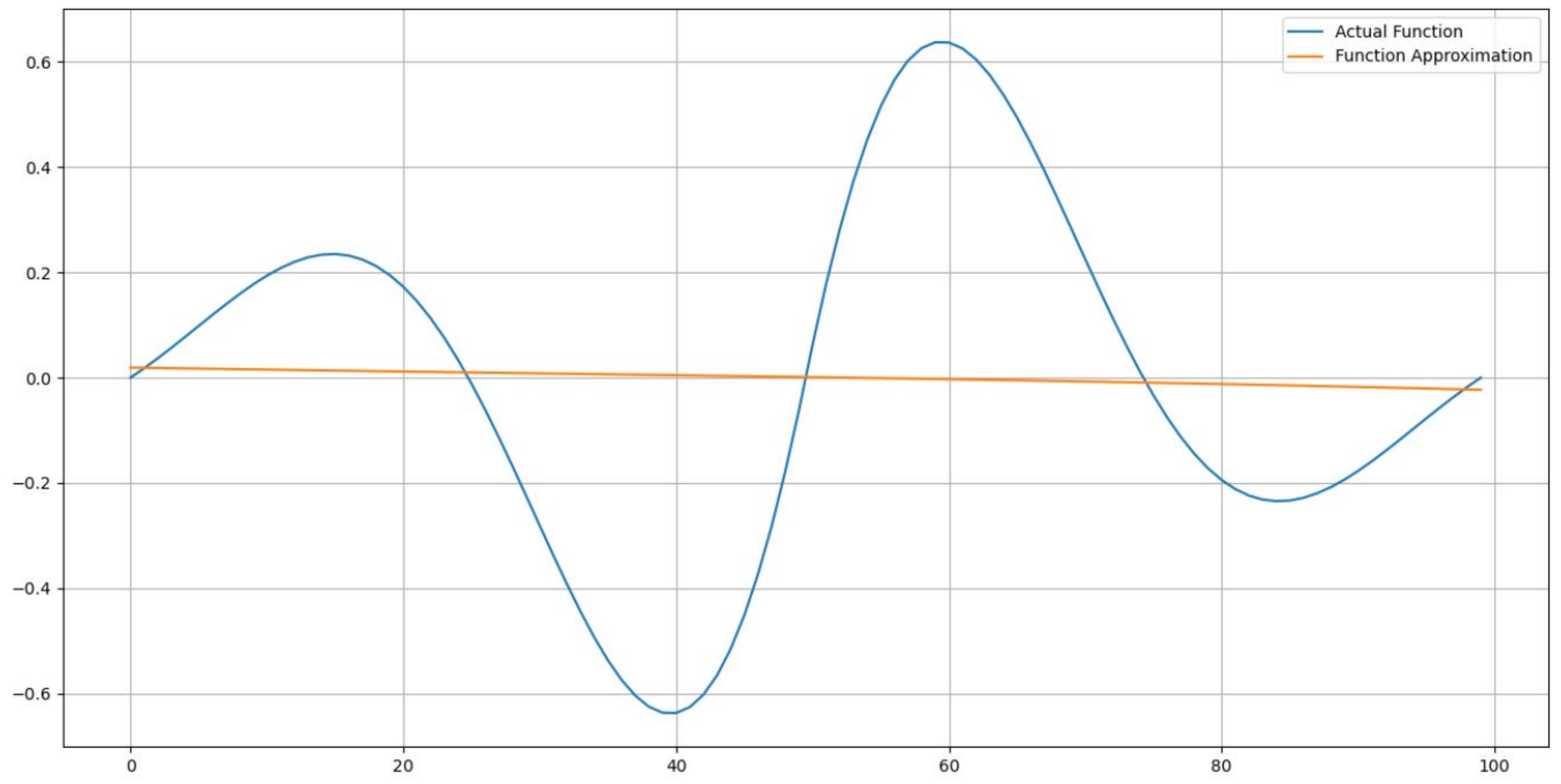
Time for Batch Training

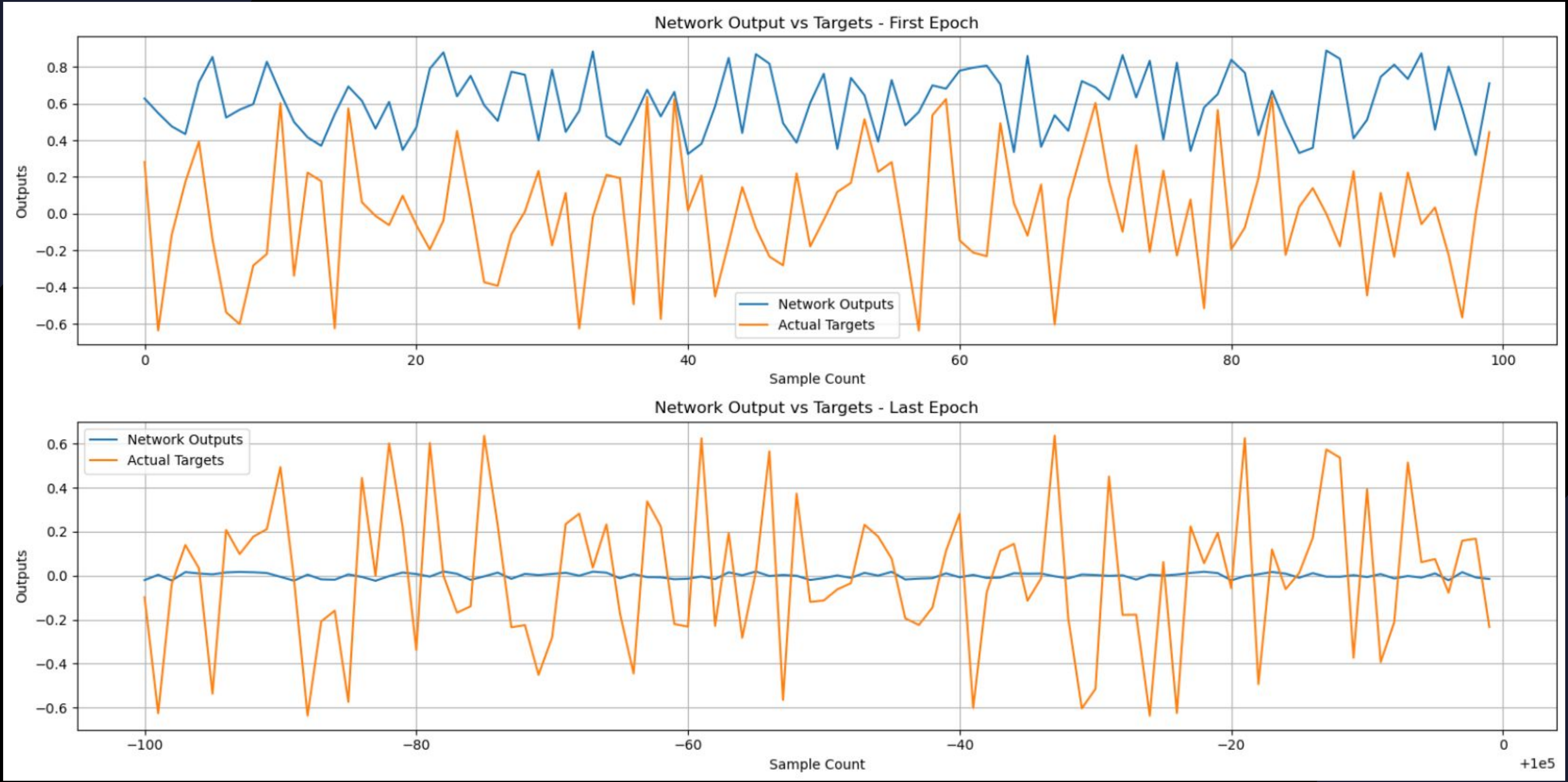
Key Observations:

Batch Training vs Online Training

- Maybe Batch training is not all that better than online.
- Prior Assumption - faster convergence
- Assumption challenged - Different result

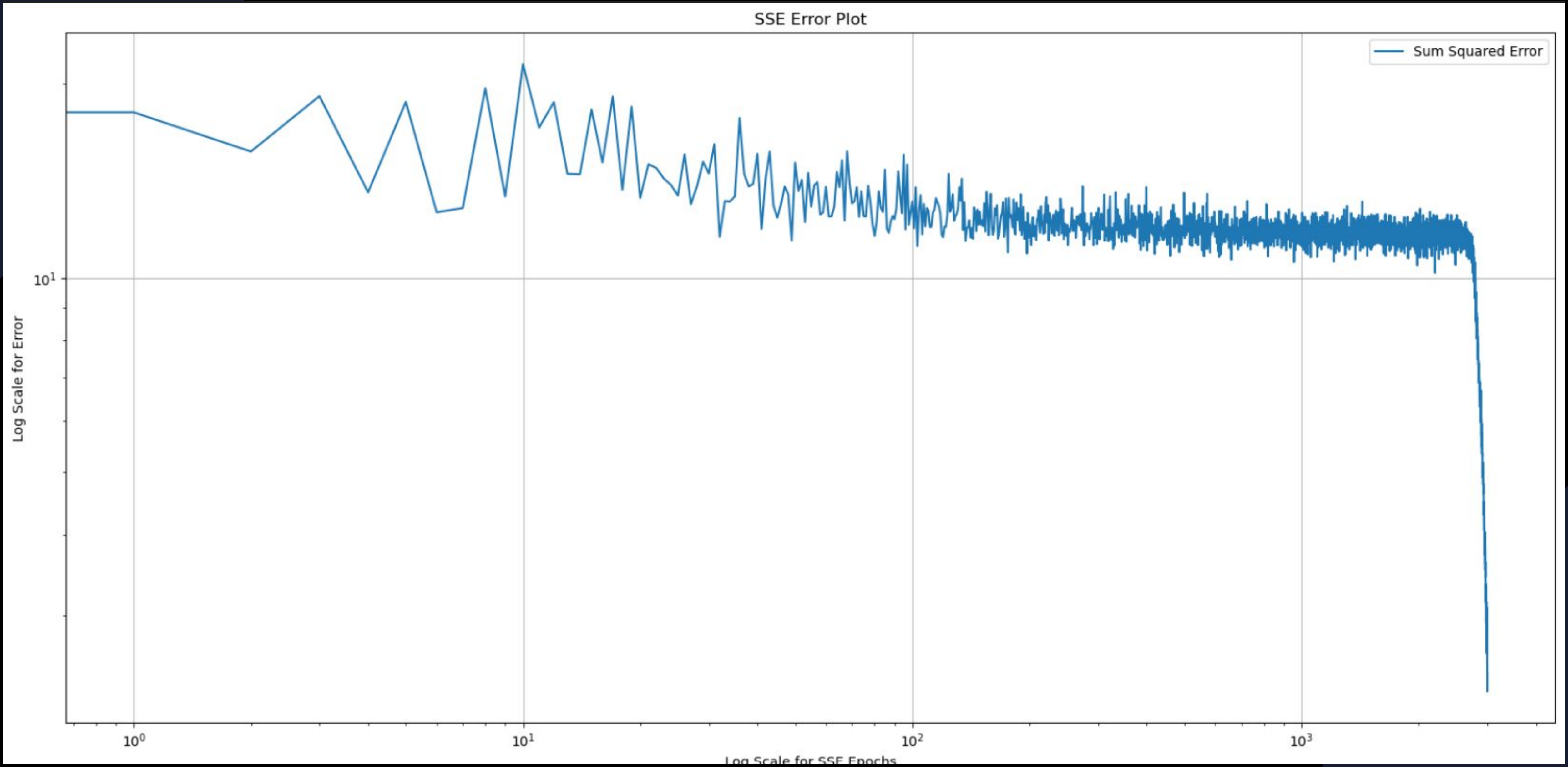

SSE Error Plot
Sum Squared Error
2.2×10^1
2×10^1
1.8×10^1
1.6×10^1
1.4×10^1
1.2×10^1
Log Scale for Error
10^0
10^1
10^2
10^3
Log Scale for SSE Epochs

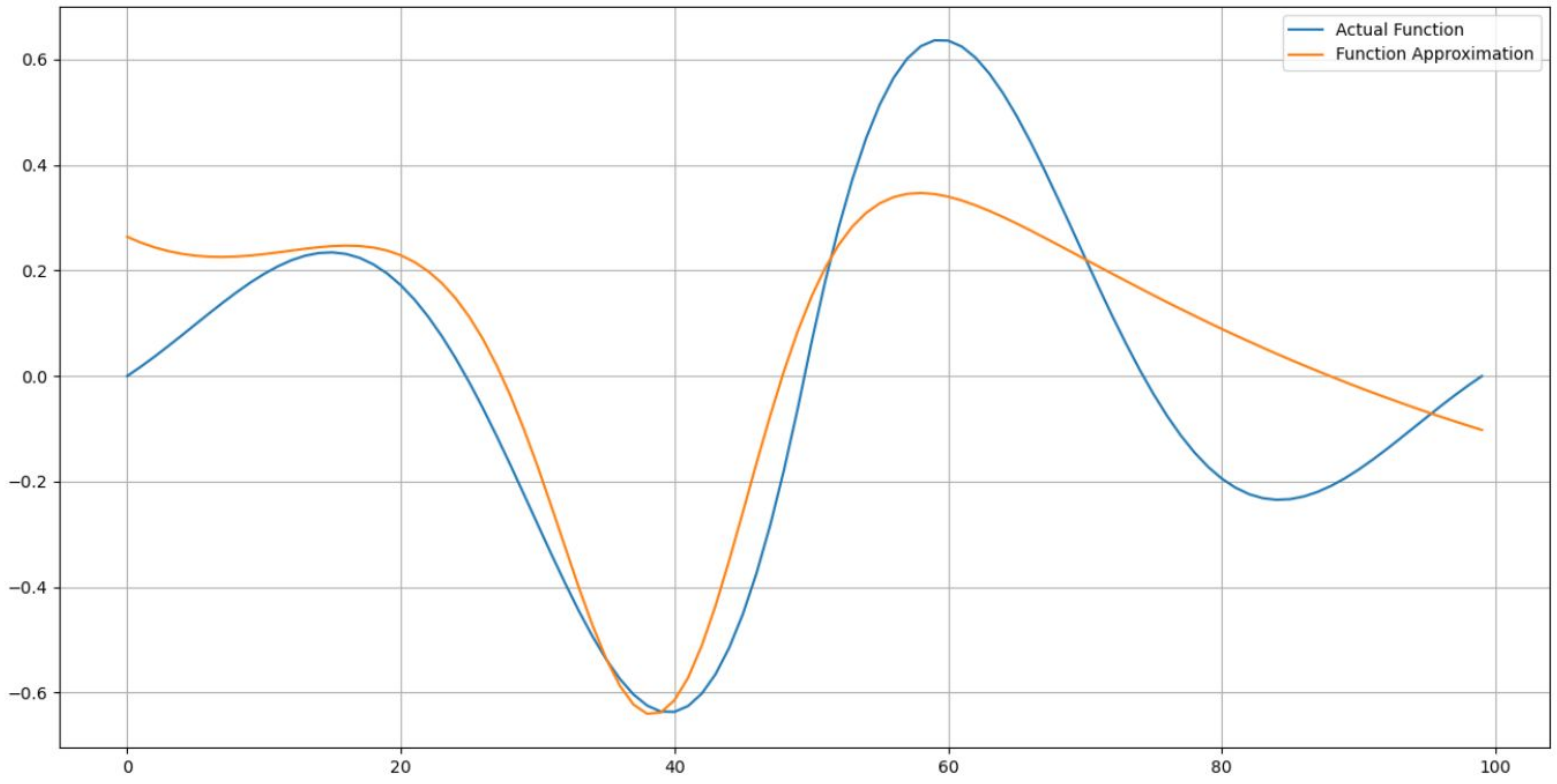

Actual Function
Function Approximation
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
20
40
60
80
100

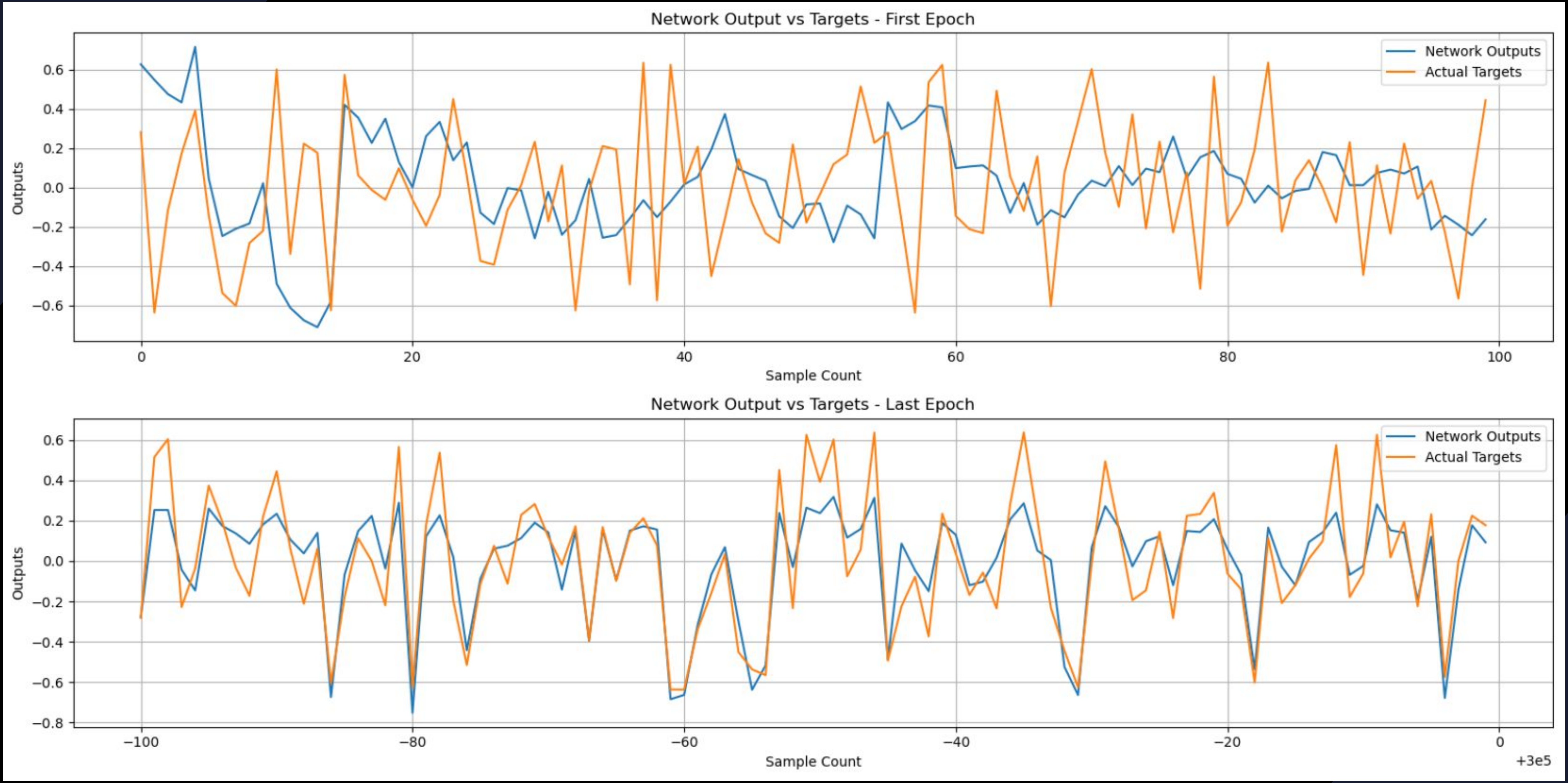

Network Output vs Targets - First Epoch
Outputs
Sample Count
Network Outputs
Actual Targets
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
20
40
60
80
100
Network Output vs Targets - Last Epoch
Outputs
Sample Count
Network Outputs
Actual Targets
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−100
−80
−60
−40
−20
0
+1e5

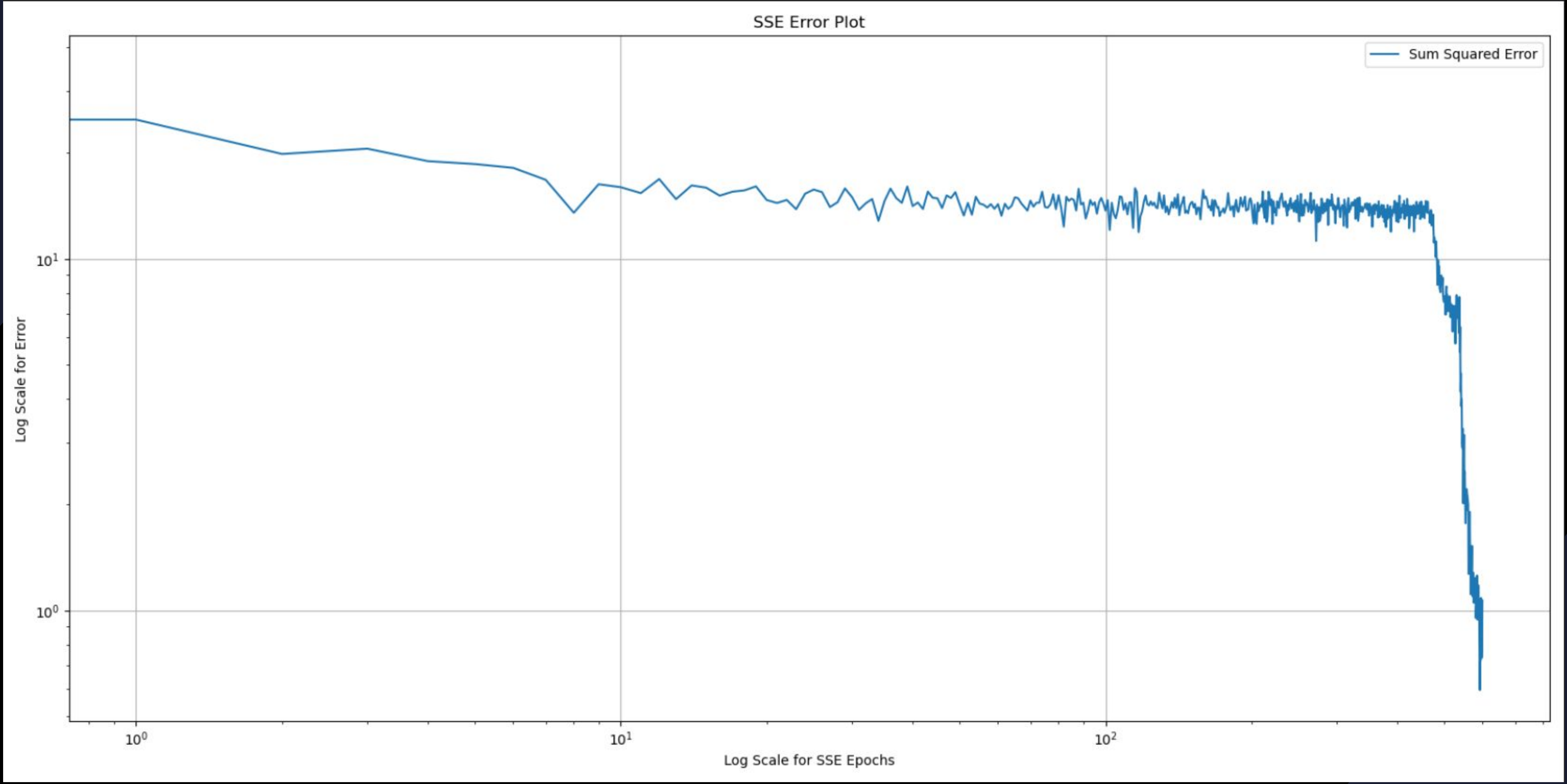
Hypothesis testing

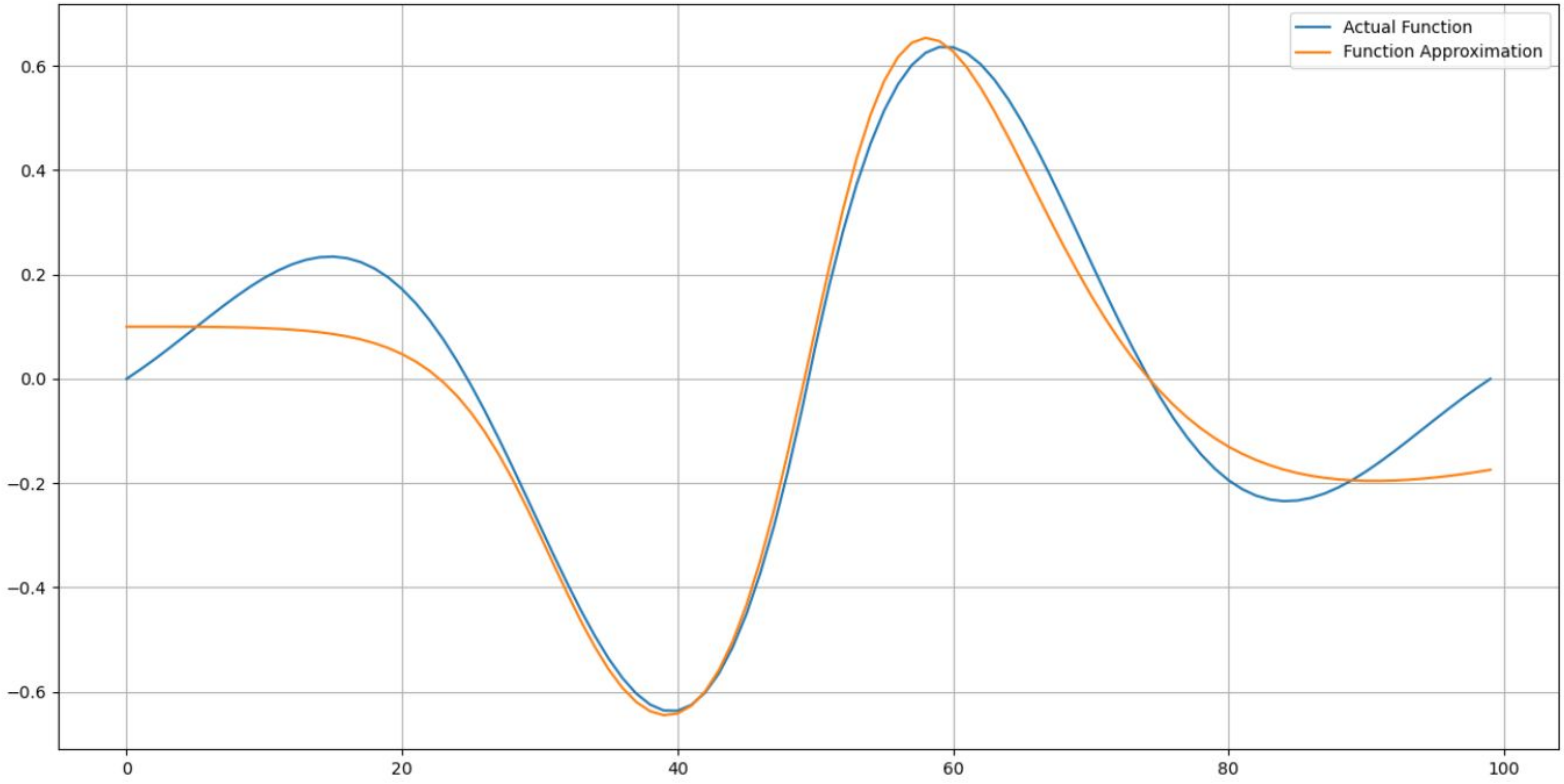
- Comparison between same number of weight updates for batch vs online.

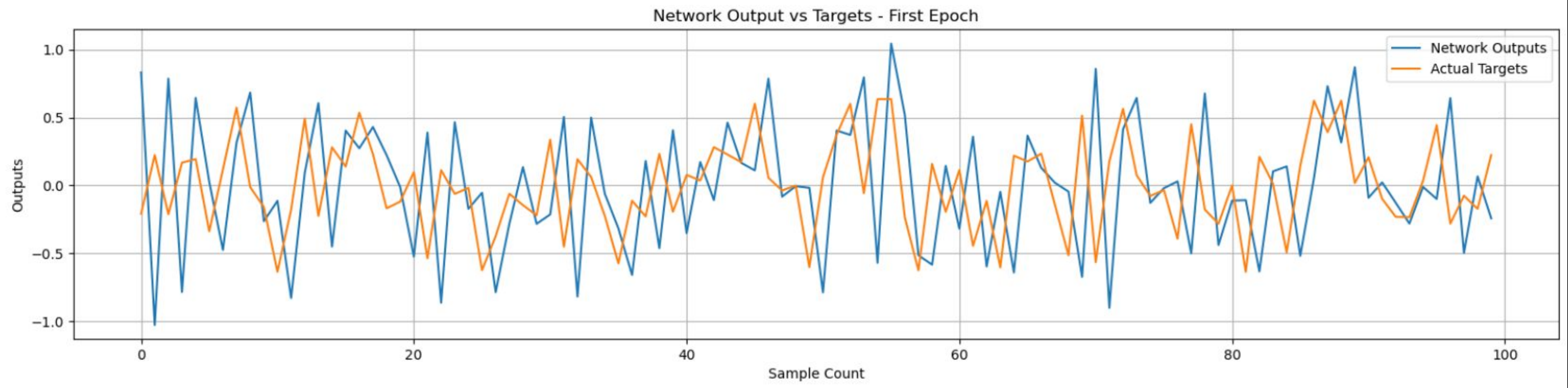

SSE Error Plot
Sum Squared Error
10^1
Log Scale for Error
10^0
10^1
10^2
10^3
Log Scale for SSE Epochs

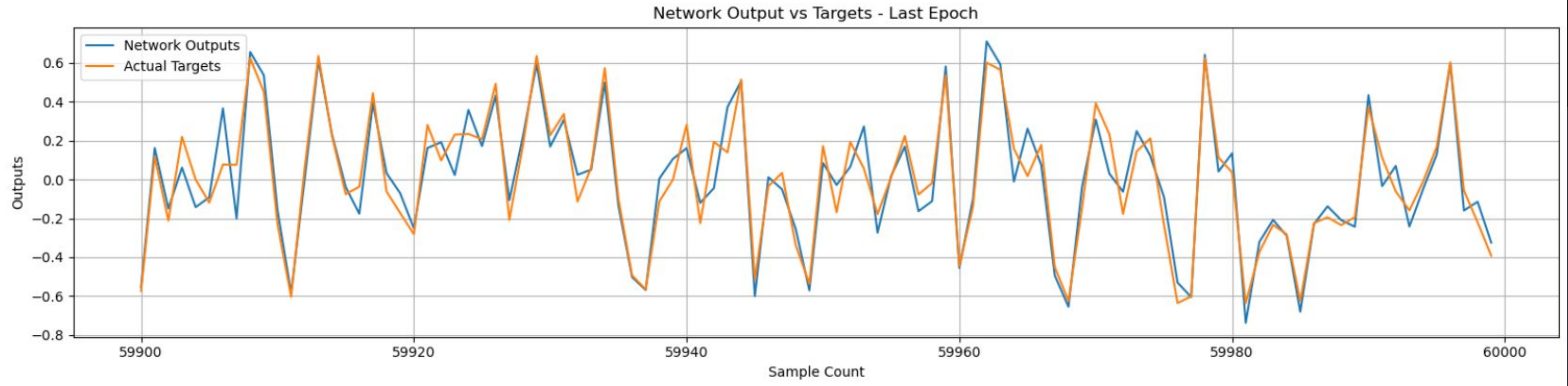

Actual Function
Function Approximation
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
20
40
60
80
100


Network Output vs Targets - First Epoch
Network Outputs
Actual Targets
Outputs
Sample Count
Network Output vs Targets - Last Epoch
Network Outputs
Actual Targets
Outputs
Sample Count
+3e5

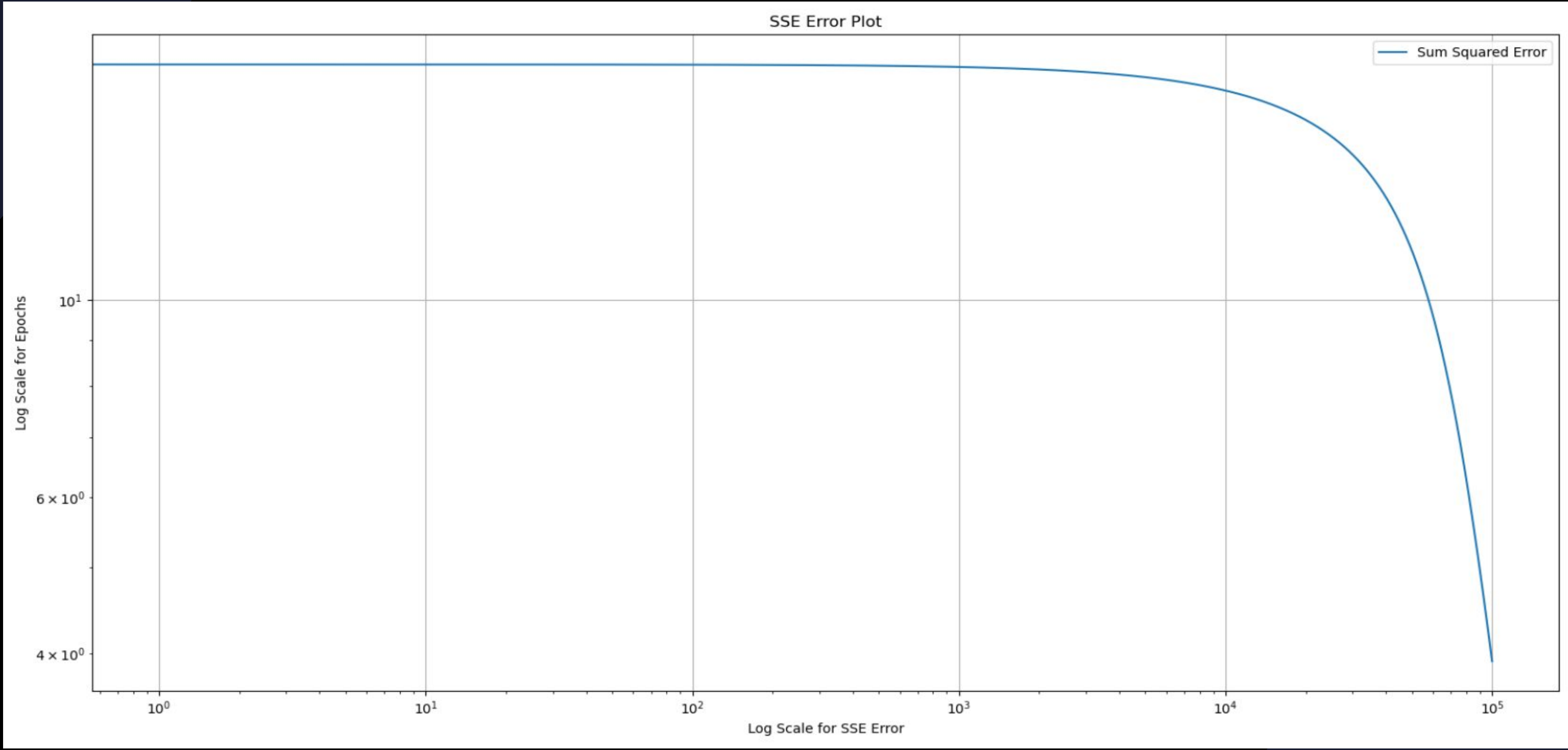

SSE Error Plot
Sum Squared Error
Log Scale for Error
Log Scale for SSE Epochs
10^1
10^0
10^0
10^1
10^2


Actual Function
Function Approximation
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
20
40
60
80
100


Network Output vs Targets - First Epoch
Network Outputs
Actual Targets
Outputs
Sample Count
1.0
0.5
0.0
−0.5
−1.0
0
20
40
60
80
100


Network Output vs Targets - Last Epoch
Network Outputs
Actual Targets
Outputs
Sample Count
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
59900
59920
59940
59960
59980
60000

However!


SSE Error Plot
Sum Squared Error
Log Scale for Epochs
Log Scale for SSE Error
10^1
6×10^0
4×10^0
10^0
10^1
10^2
10^3
10^4
10^5

Next up - LM Algo

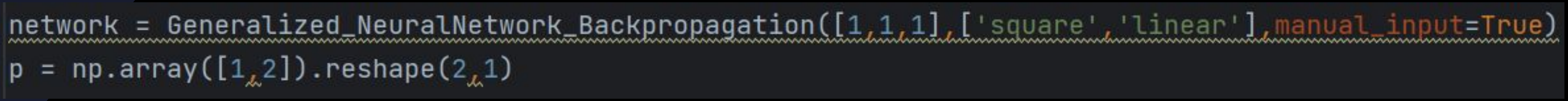
Challenge 1 - The Jakobian.

NN_design to the rescue - Problem 12.5

- Added Capability of manual input

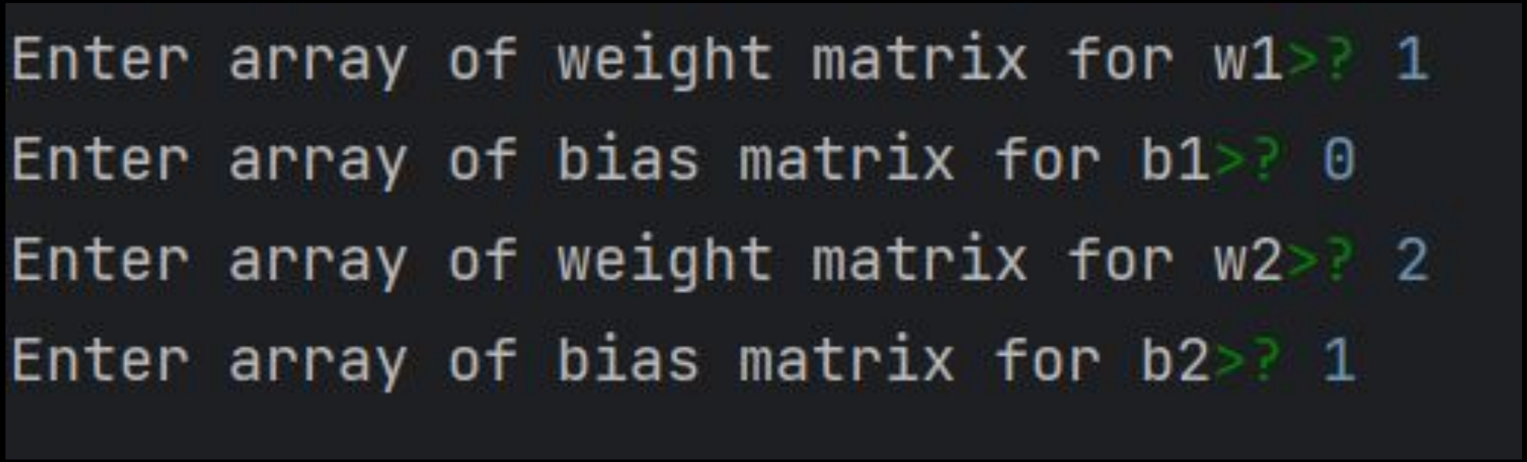
Challenge 2 - Weight update

The Code:

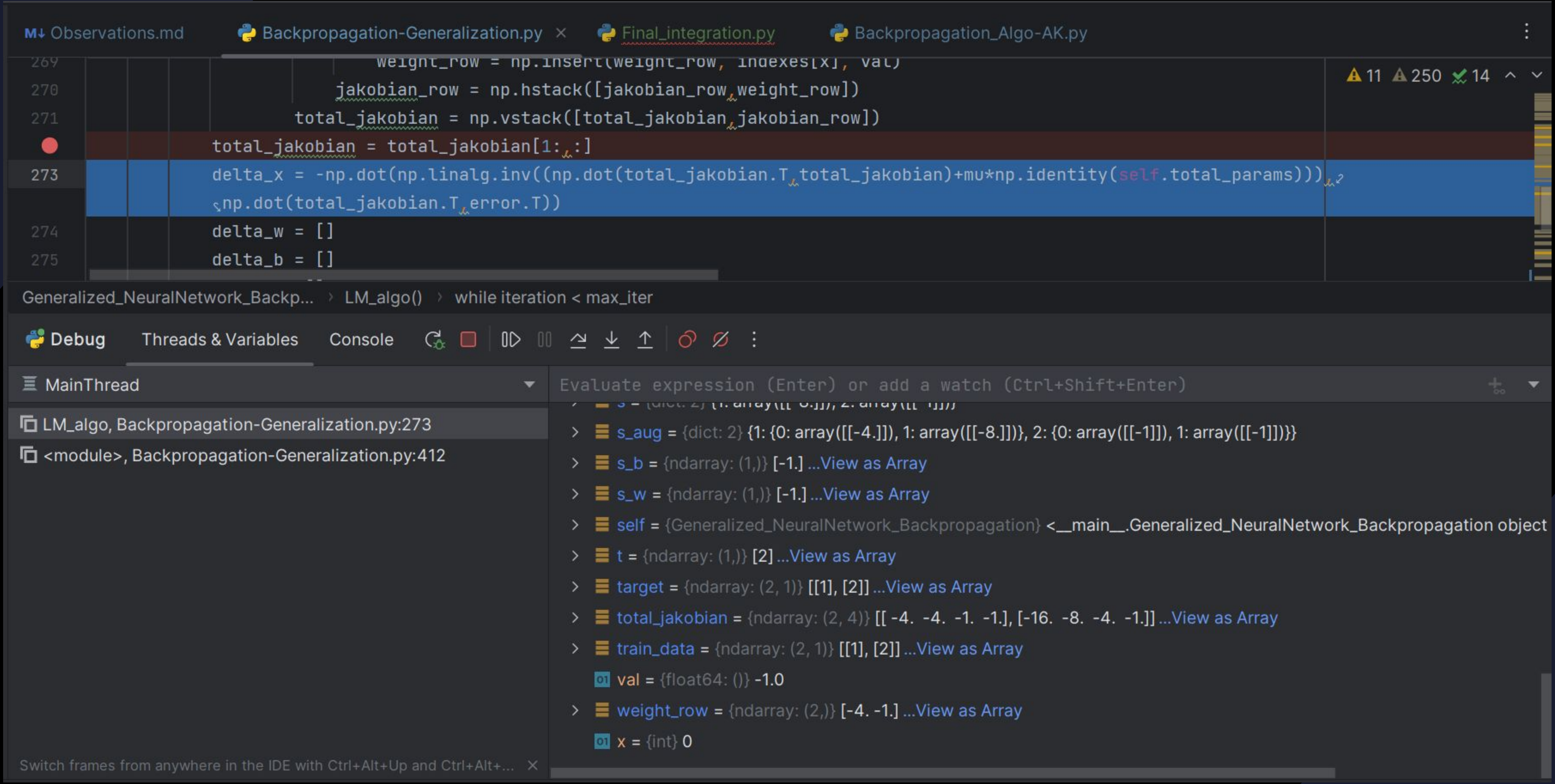


```
network = Generalized_NeuralNetwork_Backpropagation([1,1,1],['square','linear'],manual_input=True)
p = np.array([1,2]).reshape(2,1)
```

Weight matrix from the book



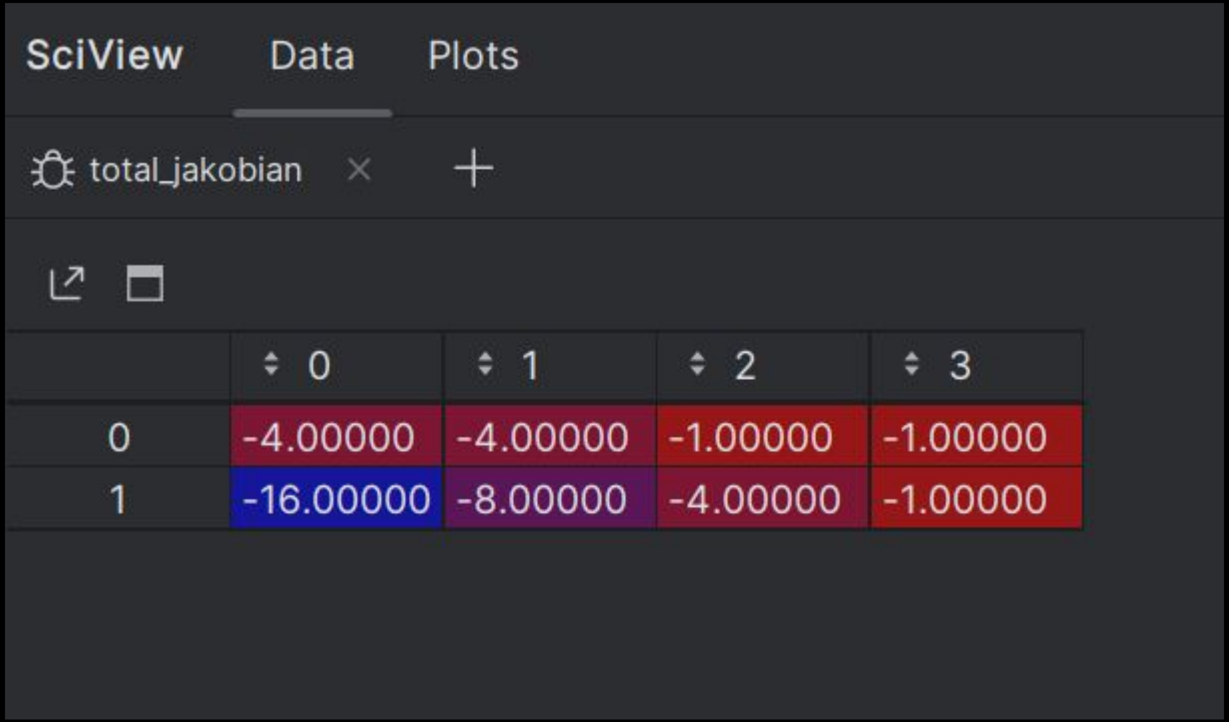
```
Enter array of weight matrix for w1>? 1
Enter array of bias matrix for b1>? 0
Enter array of weight matrix for w2>? 2
Enter array of bias matrix for b2>? 1
```


Observations.md
Backpropagation-Generalization.py
Final_integration.py
Backpropagation_Algo-AK.py
11 250 14
Generalized_NeuralNetwork_Backp... › LM_algo() › while iteration < max_iter
Debug
Threads & Variables
Console
MainThread
Evaluate expression (Enter) or add a watch (Ctrl+Shift+Enter)
Switch frames from anywhere in the IDE with Ctrl+Alt+Up and Ctrl+Alt+...

```
                    jakobian_row = np.hstack([jakobian_row,weight_row])
                total_jakobian = np.vstack([total_jakobian,jakobian_row])
            total_jakobian = total_jakobian[1:,:]
            delta_x = -np.dot(np.linalg.inv((np.dot(total_jakobian.T,total_jakobian)+mu*np.identity(self.total_params)),
            np.dot(total_jakobian.T,error.T))
            delta_w = []
            delta_b = []
```

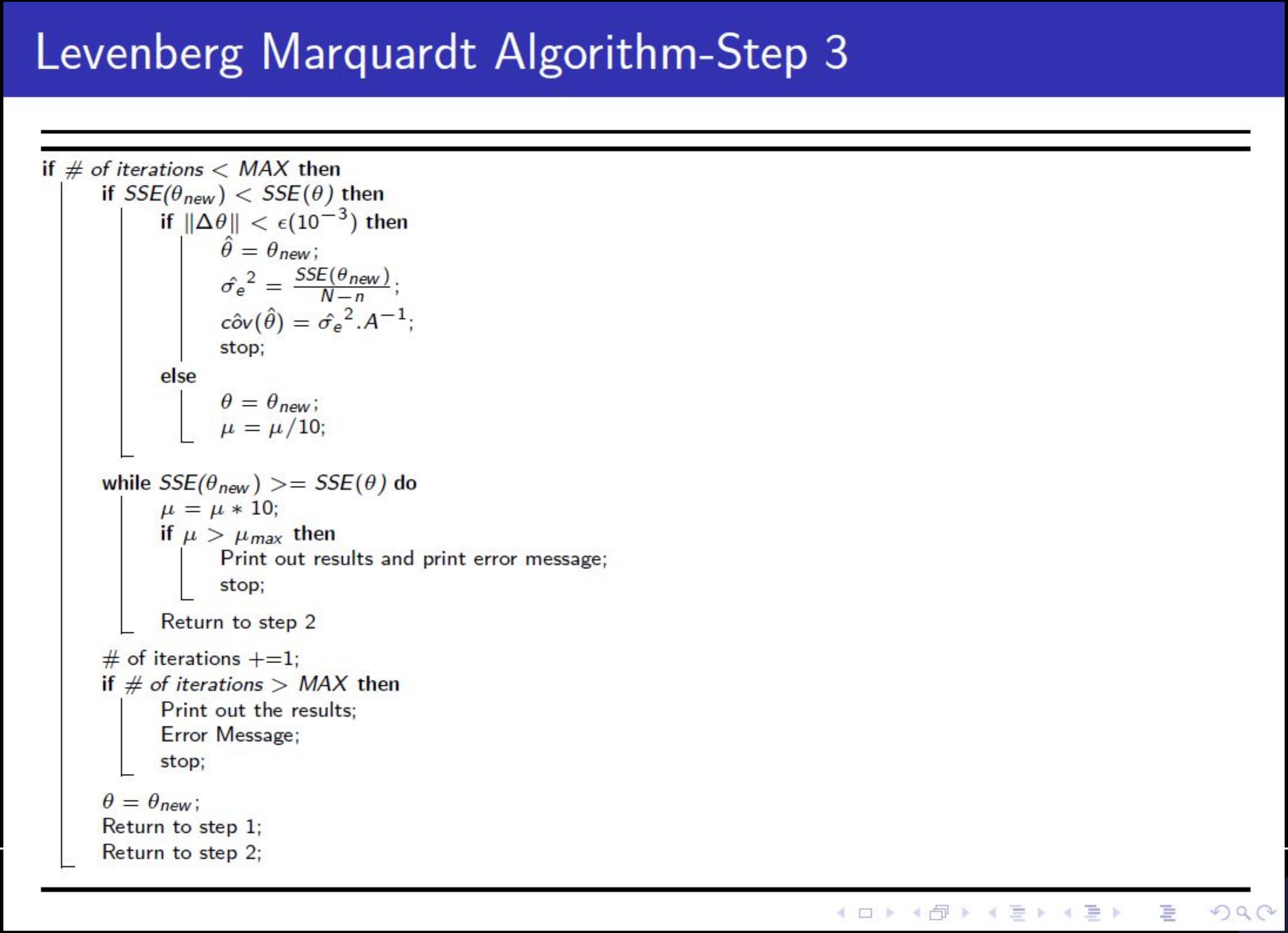
```
LM_algo, Backpropagation-Generalization.py:273
<module>, Backpropagation-Generalization.py:412
```

```
s_aug = {dict: 2} {1: {0: array([[-4.]]), 1: array([[-8.]])}, 2: {0: array([[-1]]), 1: array([[-1]])}}
s_b = {ndarray: (1,)} [-1.] ...View as Array
s_w = {ndarray: (1,)} [-1.] ...View as Array
self = {Generalized_NeuralNetwork_Backpropagation} <__main__.Generalized_NeuralNetwork_Backpropagation object
t = {ndarray: (1,)} [2] ...View as Array
target = {ndarray: (2, 1)} [[1], [2]] ...View as Array
total_jakobian = {ndarray: (2, 4)} [[ -4.  -4.  -1.  -1.], [-16.  -8.  -4.  -1.]] ...View as Array
train_data = {ndarray: (2, 1)} [[1], [2]] ...View as Array
val = {float64: ()} -1.0
weight_row = {ndarray: (2,)} [-4. -1.] ...View as Array
x = {int} 0
```



	0	1	2	3
0	-4.00000	-4.00000	-1.00000	-1.00000
1	-16.00000	-8.00000	-4.00000	-1.00000

Reference for pseudo code



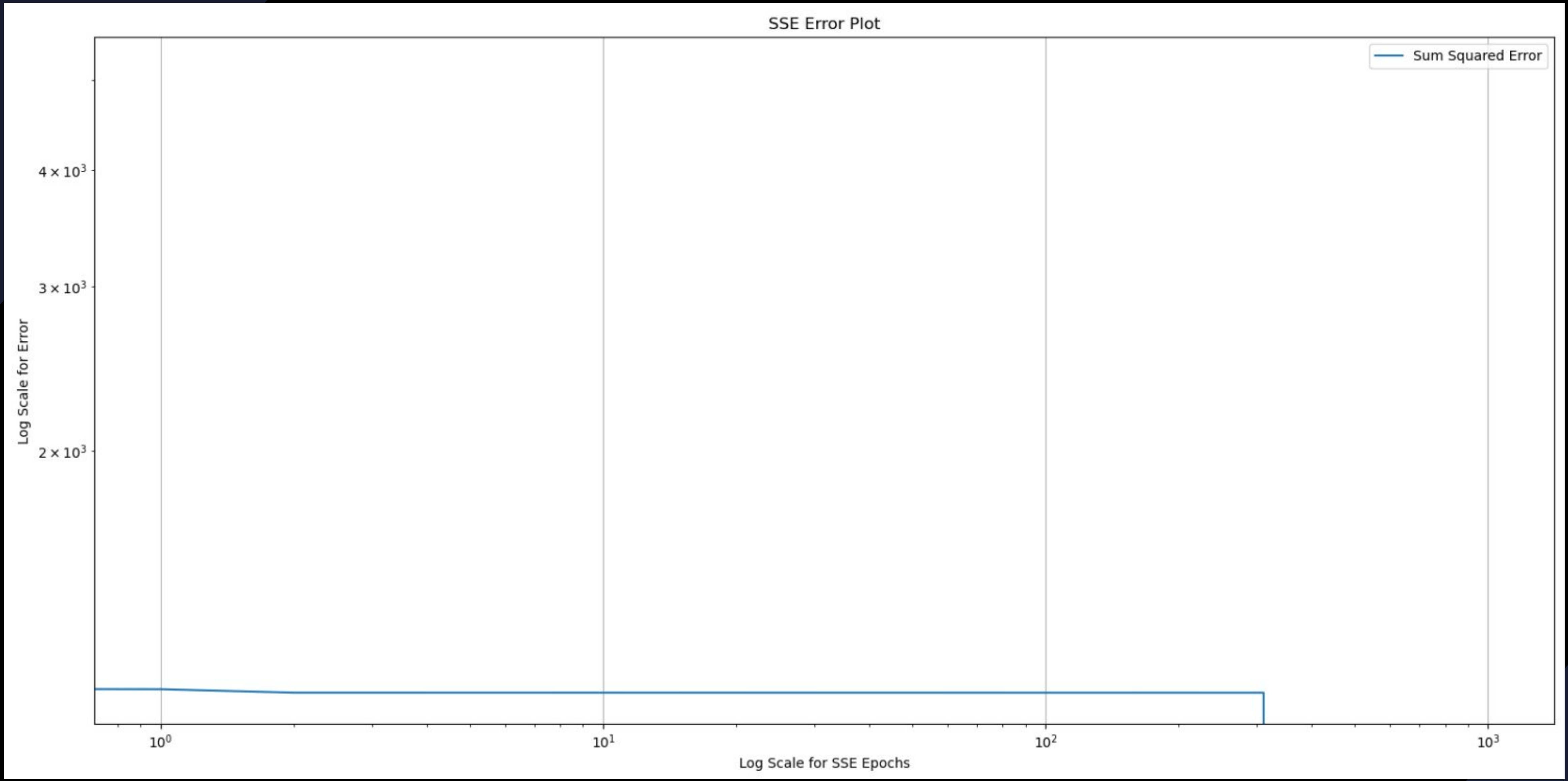
Levenberg Marquardt Algorithm-Step 3

```
if $\#$ of iterations $< MAX$ then
    if $SSE(\theta_{new}) < SSE(\theta)$ then
        if $\|\Delta\theta\| < \epsilon(10^{-3})$ then
            $\hat{\theta} = \theta_{new}$;
            $\hat{\sigma_e}^2 = \frac{SSE(\theta_{new})}{N-n}$;
            $\hat{cov}(\hat{\theta}) = \hat{\sigma_e}^2.A^{-1}$;
            stop;
        else
            $\theta = \theta_{new}$;
            $\mu = \mu/10$;
    while $SSE(\theta_{new}) >= SSE(\theta)$ do
        $\mu = \mu * 10$;
        if $\mu > \mu_{max}$ then
            Print out results and print error message;
            stop;
        Return to step 2
    $\#$ of iterations +=1;
    if $\#$ of iterations $> MAX$ then
        Print out the results;
        Error Message;
        stop;
    $\theta = \theta_{new}$;
    Return to step 1;
    Return to step 2;
```

Another Challenge: Theoretical concept confirmed

LM - Successful!

Testing fail on synthetic function.



```
Algorithm has reached instability with mu value becoming too large. Saving current weights and biases
187
```

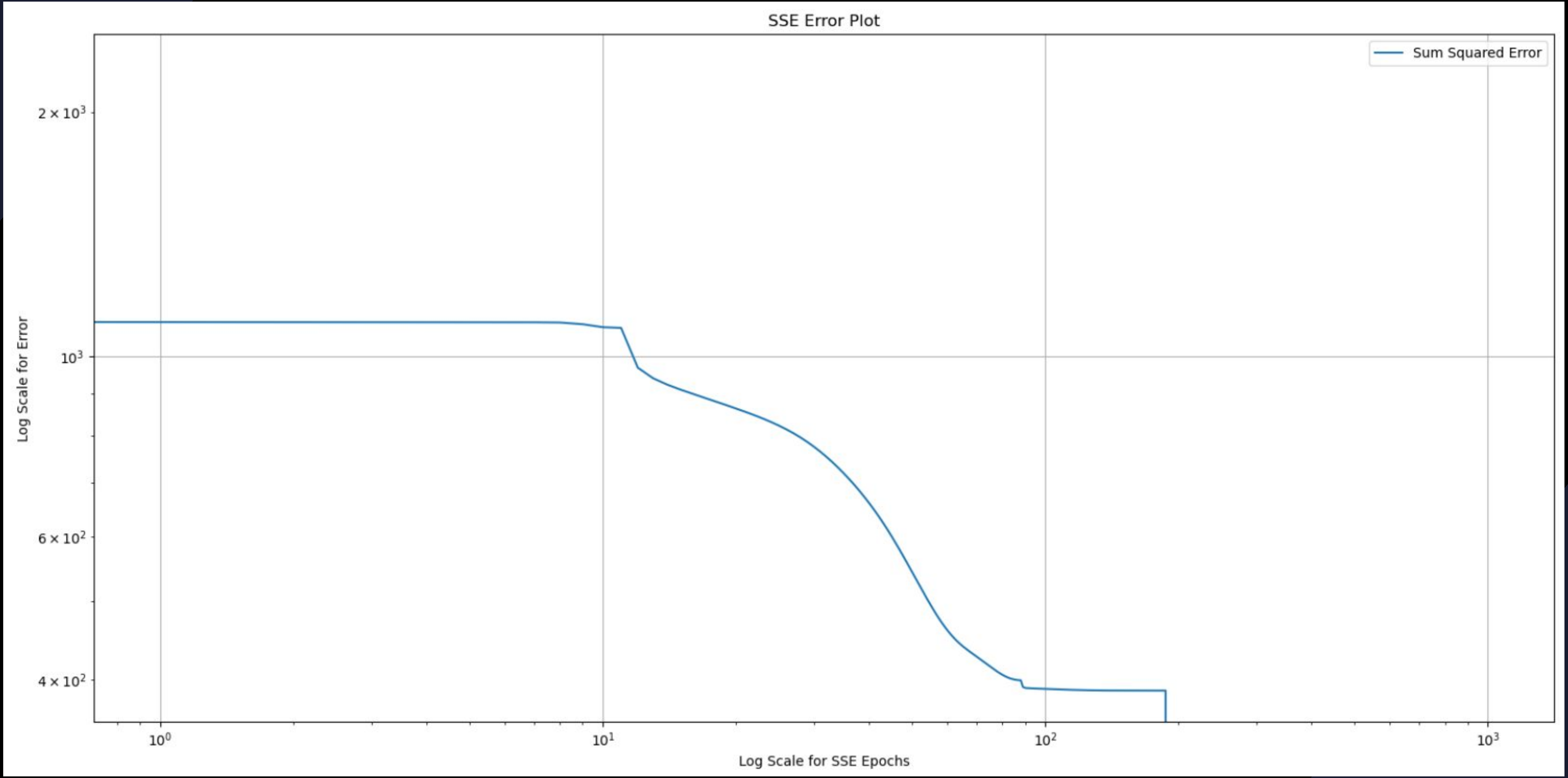
Issue resolution

Parameter tweakings: mu_max, convergence criteria, max number of iterations…

No luck

Forgot about local minima/saddle point!

Issue resolution



Learning

After observing the LM algo issue, it has become clear how it is important to initialize the network with different weights and biases to get optimum results and avoid saddle point/local minima issues.

Future Work

- Add Verbose type feature in custom training codes so that the status of training can be viewed easily.
- Create an interactive dash enable dashboard that will display the findings of this project effectively.
- Due to observed issue with LM Algo, another added feature in code could be, multiple seed initializations and picking the lowest SSE score/scores model.
- Time complexity comparison in backend execution.
- Another added argument that takes in the metric argument - Typically SSE for classification and MSE for regression.
- Perform in-depth comparison using previously defined criterias.
- Comparison with conventional models.

REFERENCES

1. “Neural Network Design” (2nd Ed), by Martin T Hagan, ISBN 0971732116
2. Henry P. Gavin (2022), “The Levenberg-Marquardt algorithm for nonlinear least squares curve-fitting problems”, Duke University, Department of Civil & Environmental Engineering.
3. Rauf Bhat (2020), “Gradient Descent with Momentum”, Towards Data Science.
4. I. Khan et al (2020), "Design of Neural Network with Levenberg-Marquardt and Bayesian Regularization Backpropagation for Solving Pantograph Delay Differential Equations" in IEEE Access, vol. 8, pp. 137918-137933, doi: 10.1109/ACCESS.2020.3011820.