M.Sc Project

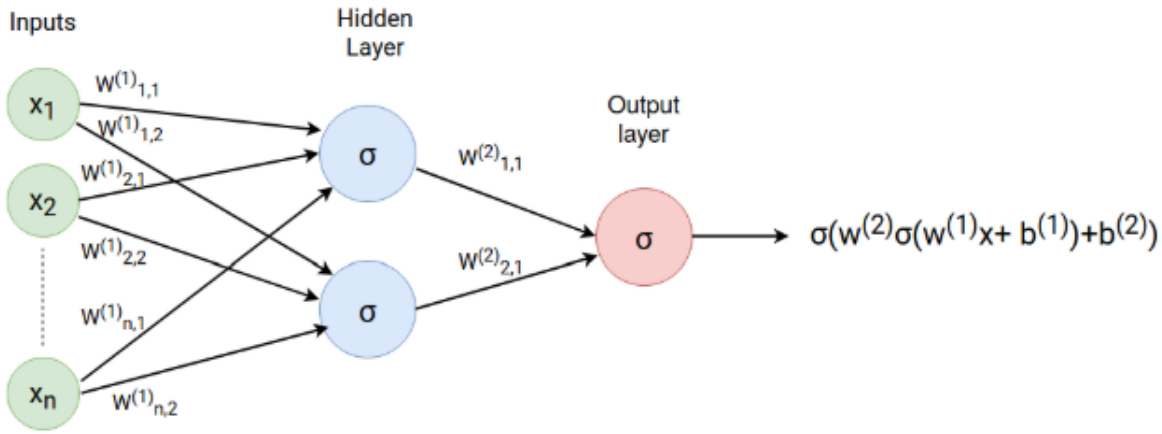
Submitted by - Shubham Singh
MA22C043
MA6650

Submitted to - Dr. Sivaram Ambikasaran

Content

1. Universal Approximation Theorem

The universal approximation theorem states that any continuous function $f : [0, 1]^n \rightarrow [0, 1]$ can be approximated arbitrarily well by a neural network with at least 1 hidden layer with a finite number of weights.



2. ANNs for Solving Ordinary and Partial Differential Equations (Research Paper)

Abstract—**We present a method to solve initial and boundary value problems using artificial neural networks. A trial solution of the differential equation is written as a sum of two parts. The first part satisfies the initial/boundary conditions and contains no adjustable parameters. The second part is constructed so as not to affect the initial/boundary conditions. This part involves a feedforward neural network containing adjustable parameters (the weights). Hence by construction the initial/boundary conditions are satisfied and the network is trained to satisfy the differential equation. The applicability of this approach ranges from single ordinary differential equations (ODE's), to systems of coupled ODE's and also to partial differential equations (PDE's). In this article, we illustrate the method by solving a variety of model problems and present comparisons with solutions obtained using the Galekrkin finite element method for several cases of partial differential equations. With the advent of neuroprocessors and digital signal processors the method becomes particularly interesting due to the expected essential gains in the execution speed.**

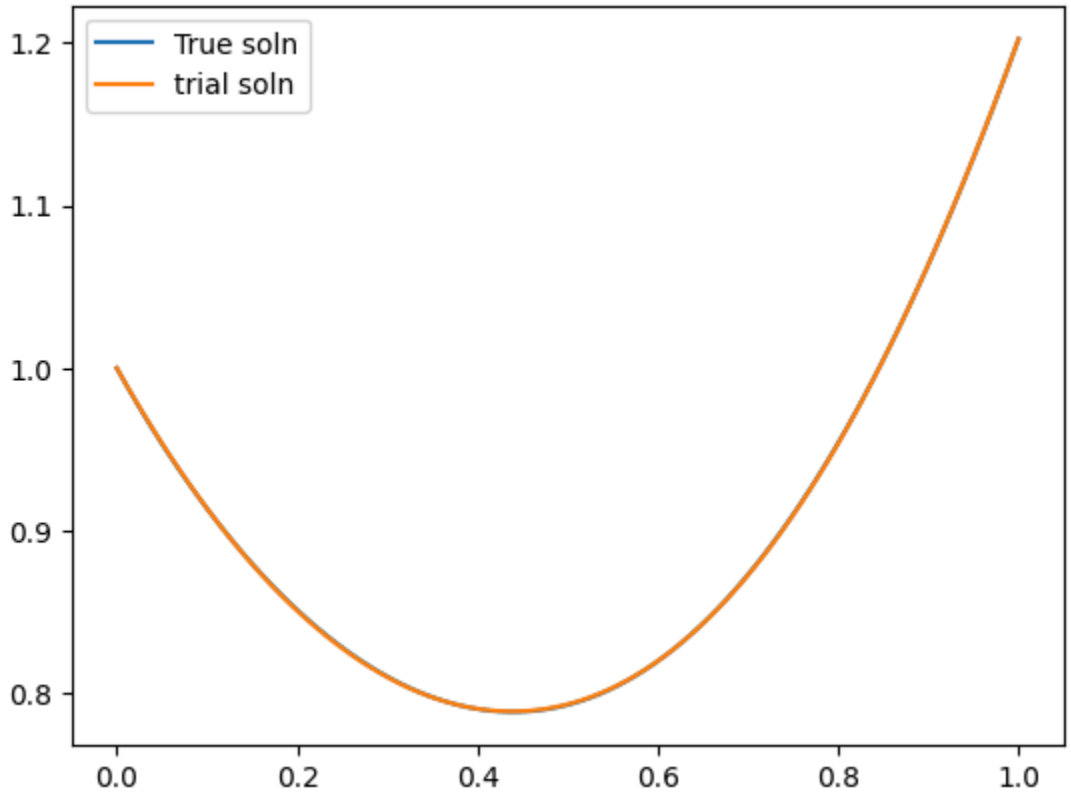
Reference:
https://arxiv.org/abs/physics/9705023

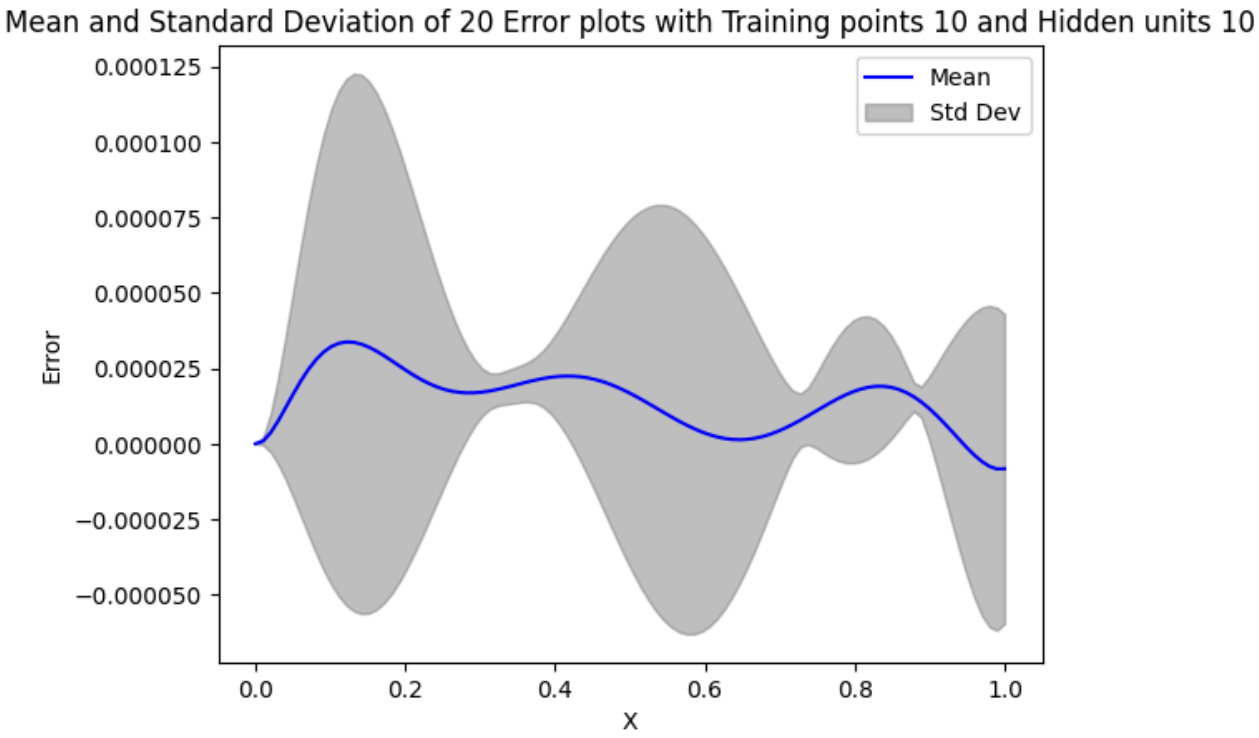
3. ANNs solutions and comparison with FEMs Methods

Ques1 1:

$$\frac{d\Psi}{dx} + \left(x + \frac{1+3x^2}{1+x+x^3}\right)\Psi = x^3 + 2x + x^2\,\frac{1+3x^2}{1+x+x^3} \tag{27}$$

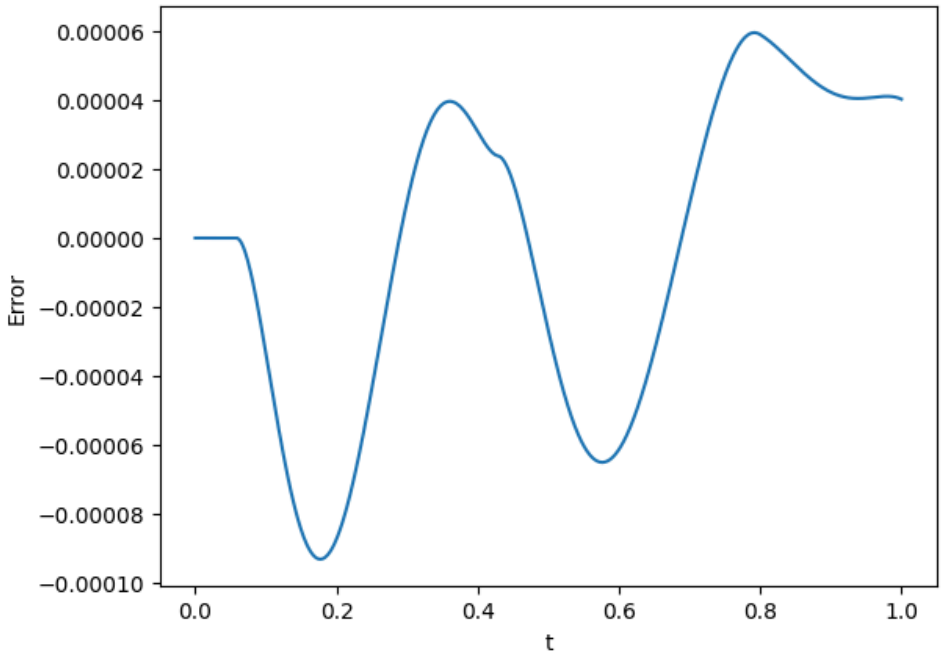
True Solution:





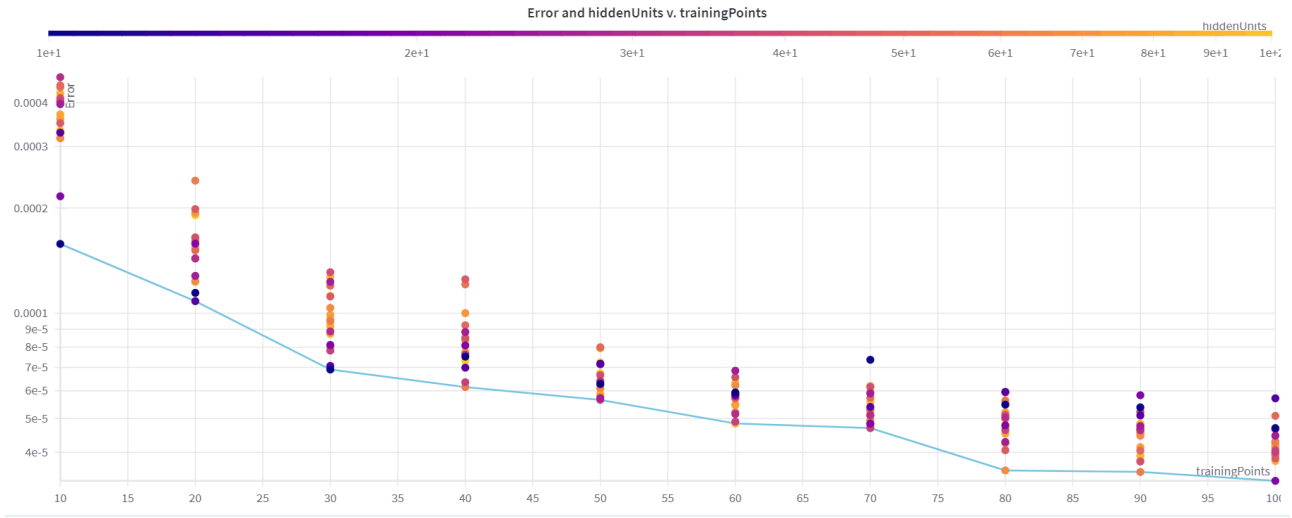
Mean time taken: 0.140820 seconds
Standard deviation: 0.100088 seconds

Error plot using Finite Element Method.
(obs: error can't be reduced further.)

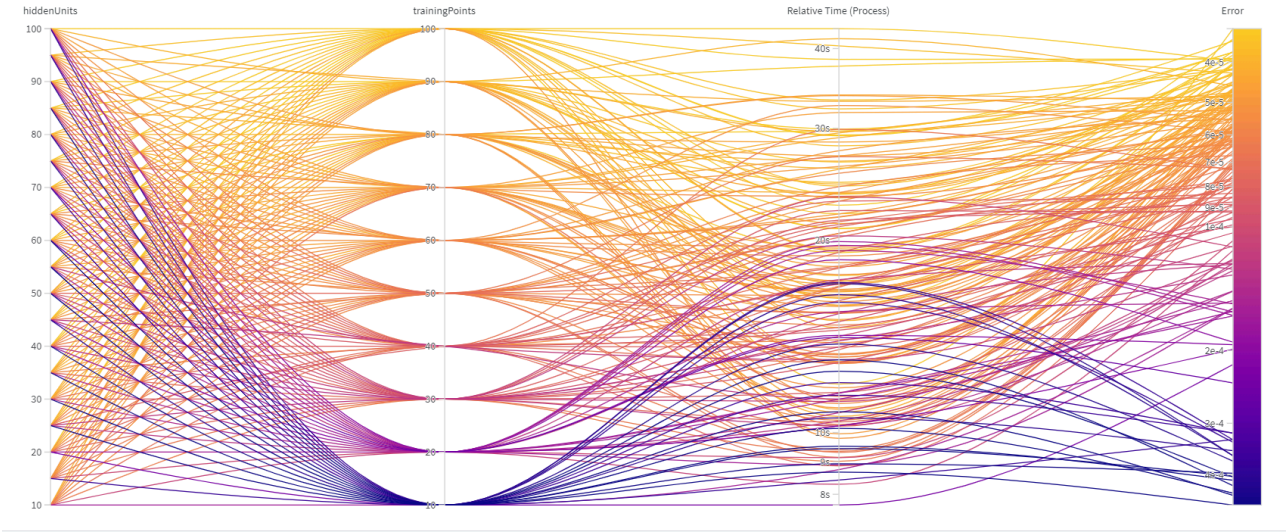


Time Taken:

Mean time taken: 0.001815 seconds
Standard deviation: 0.002000 seconds



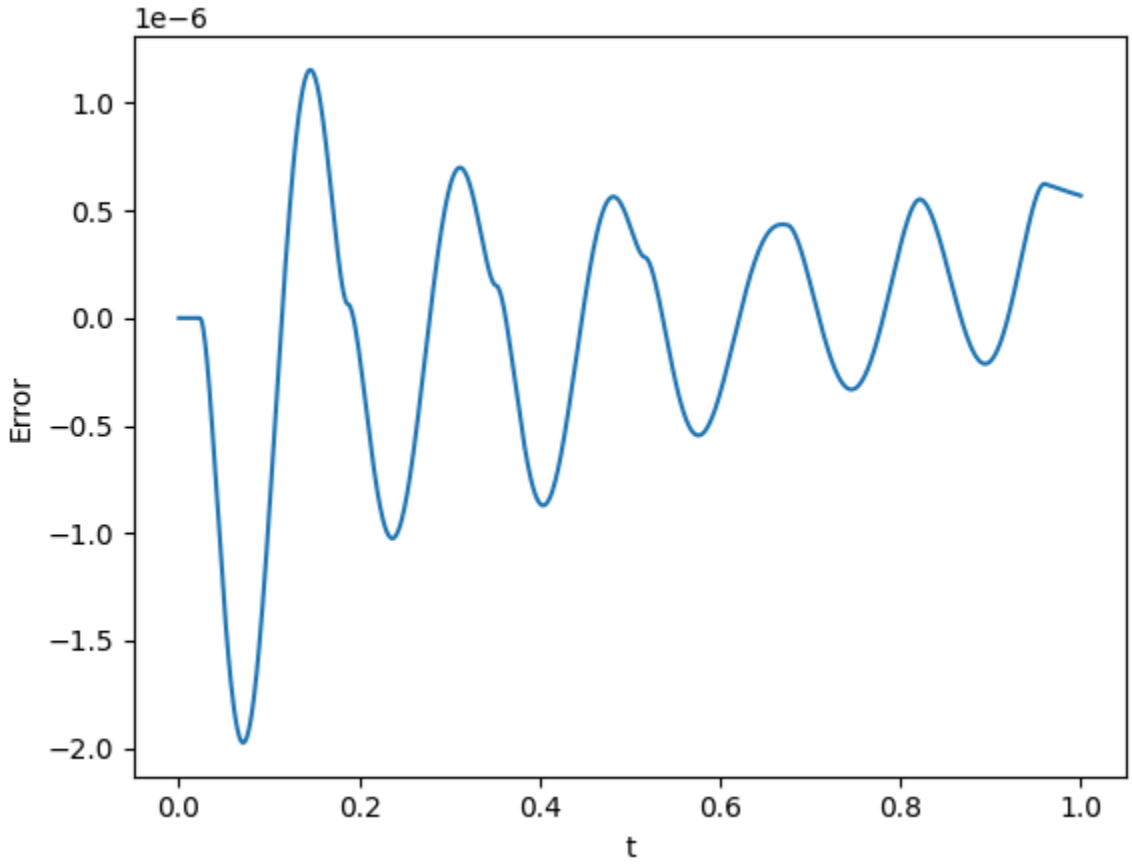
Y-axis represents Error, X-axis represents training points and color bar(z-axis) represents the no. of hidden units



Error here is the L2 Norm of the difference between analytical solution and trail solution

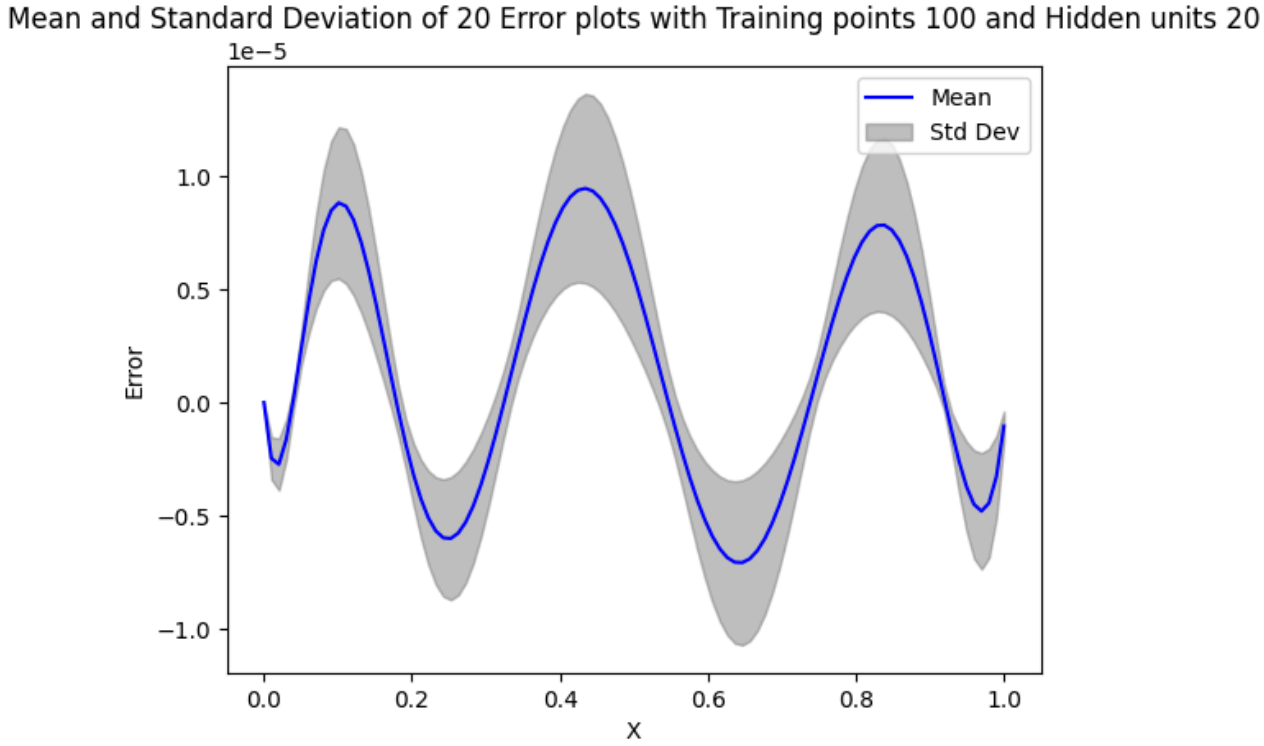
Using grid-search to find the best parameters: we can clearly see that no. of training points matters the most in terms of improving accuracy (but increasing training point can increase the relative time) and the explanation for that is we don't need high no. of hidden units to explain the function, since the solution is not a complex function relatively low hidden unit could approximate the given function, and increasing training points gives more points to interpolate the function however increasing the no. of hidden unit may improve the accuracy for the given training point but its not conclusive. Note that all the runs are averaged over 20 iterations hence, relative time taken for actual calculation will be approx (relative time)/20, which is almost similar to FEM methods.

Error and Time Taken by FEM:

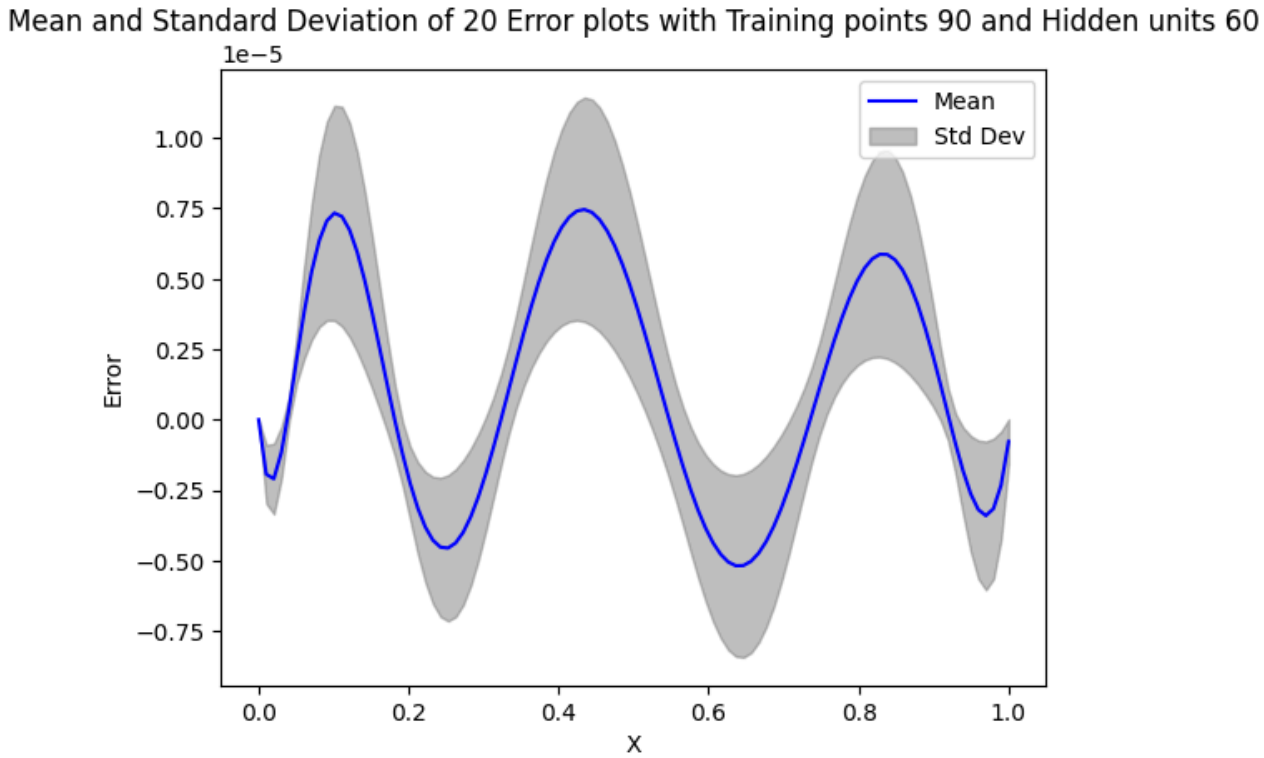


Mean time taken: 0.002818 seconds
Standard deviation: 0.001795 seconds

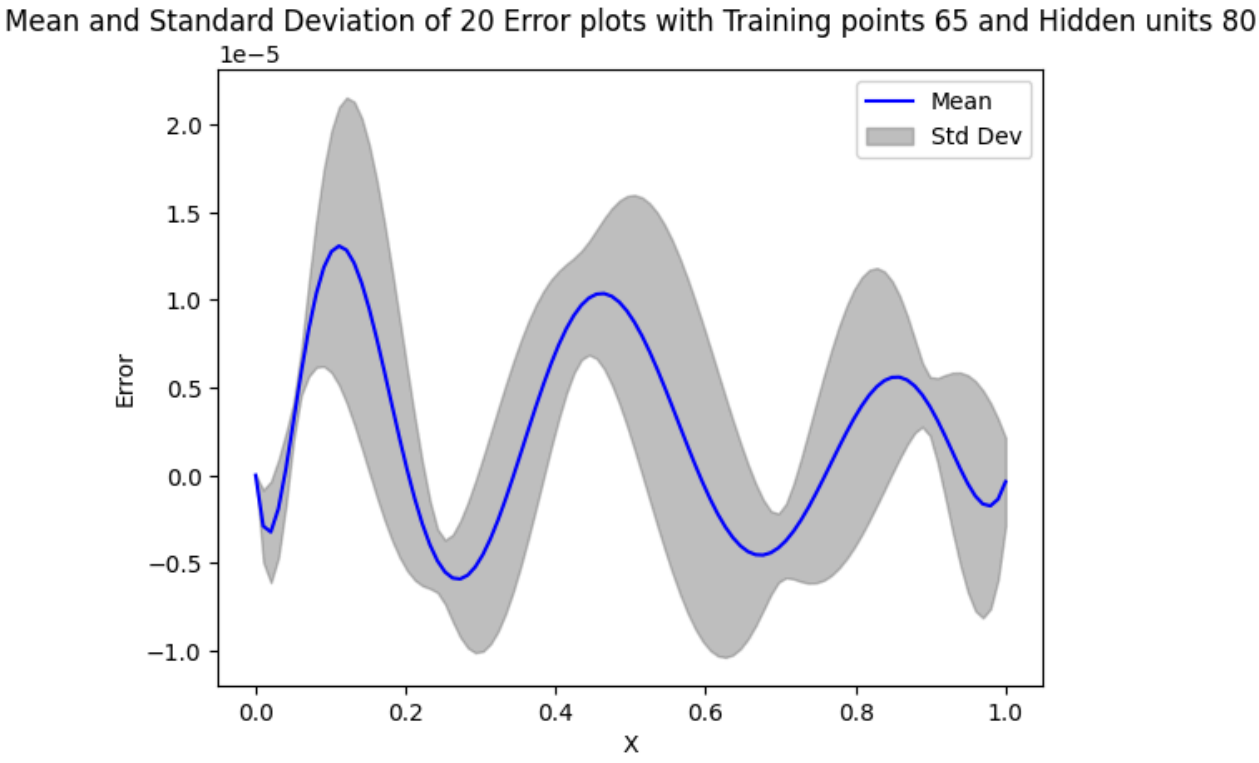
Top 3 Parameters



Mean time taken: 0.273391 seconds
Standard deviation: 0.159960 seconds



Mean time taken: 0.891434 seconds
Standard deviation: 0.632548 seconds



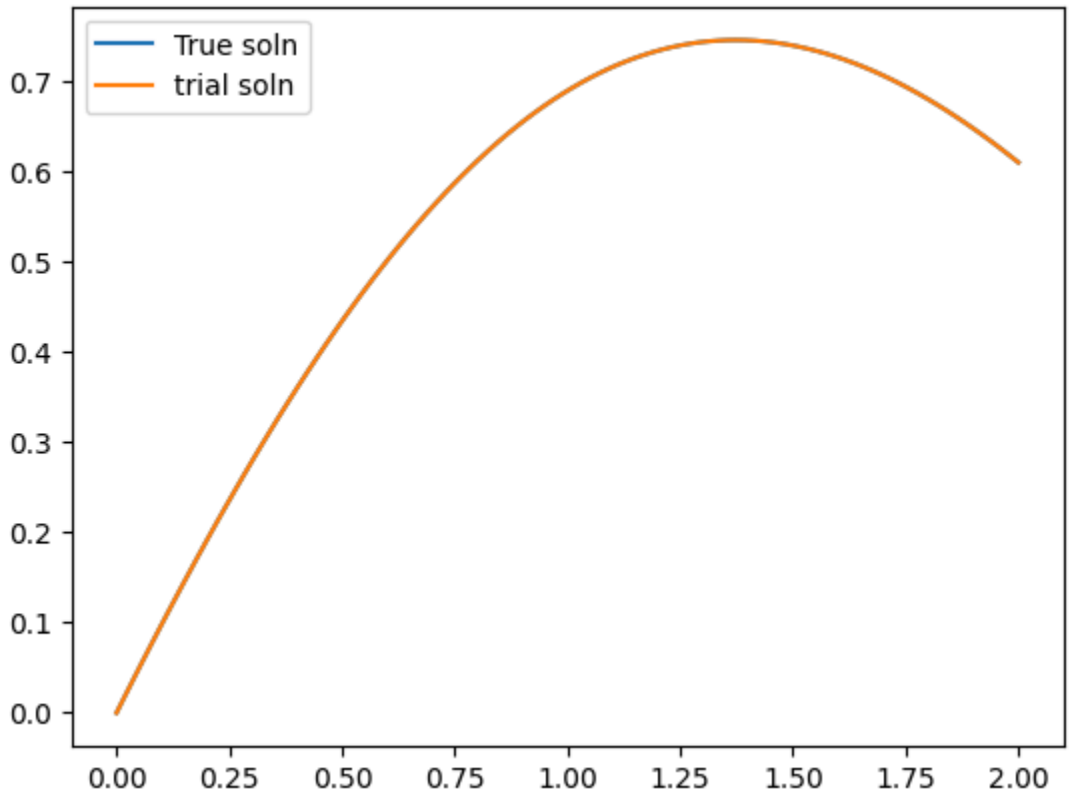
Mean time taken: 1.033208 seconds
Standard deviation: 0.674101 seconds

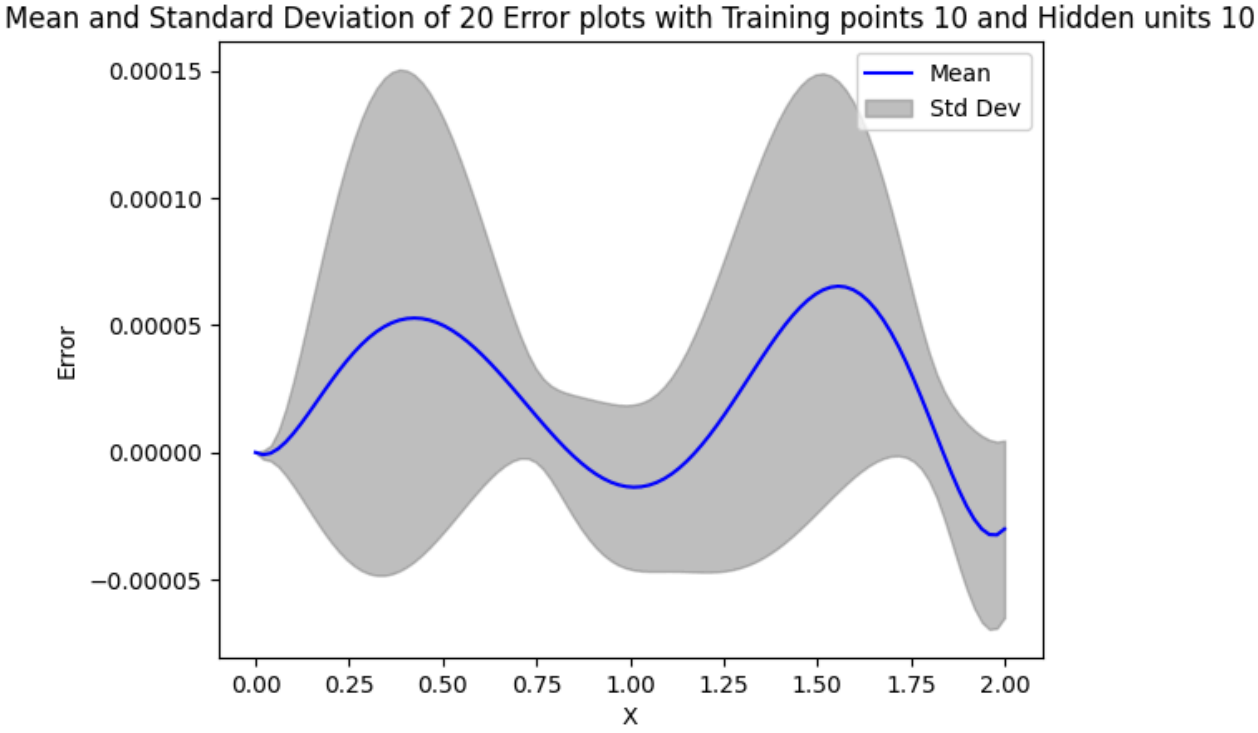
Ques 2:

$$\frac{d\Psi}{dx} + \frac{1}{5}\Psi = e^{-(x/5)}\cos(x)$$

True soln:

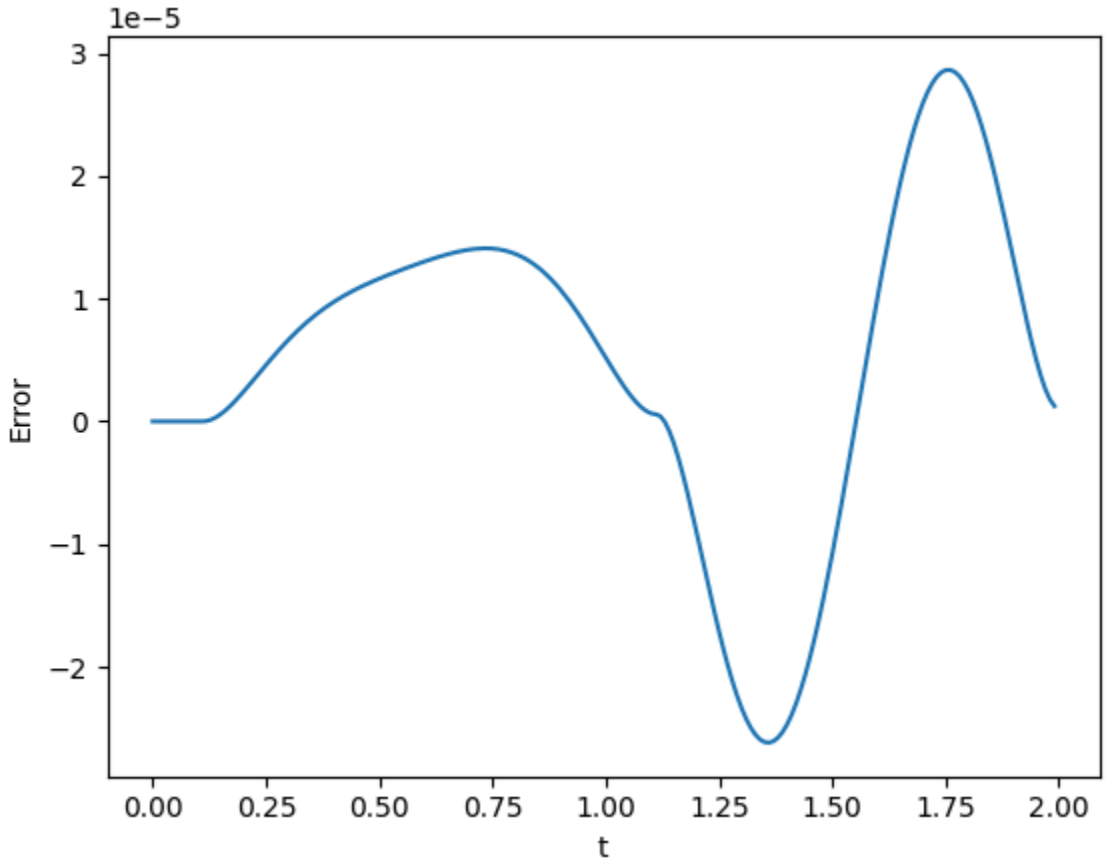
$$\Psi_a(x) = e^{-(x/5)}\sin(x)$$





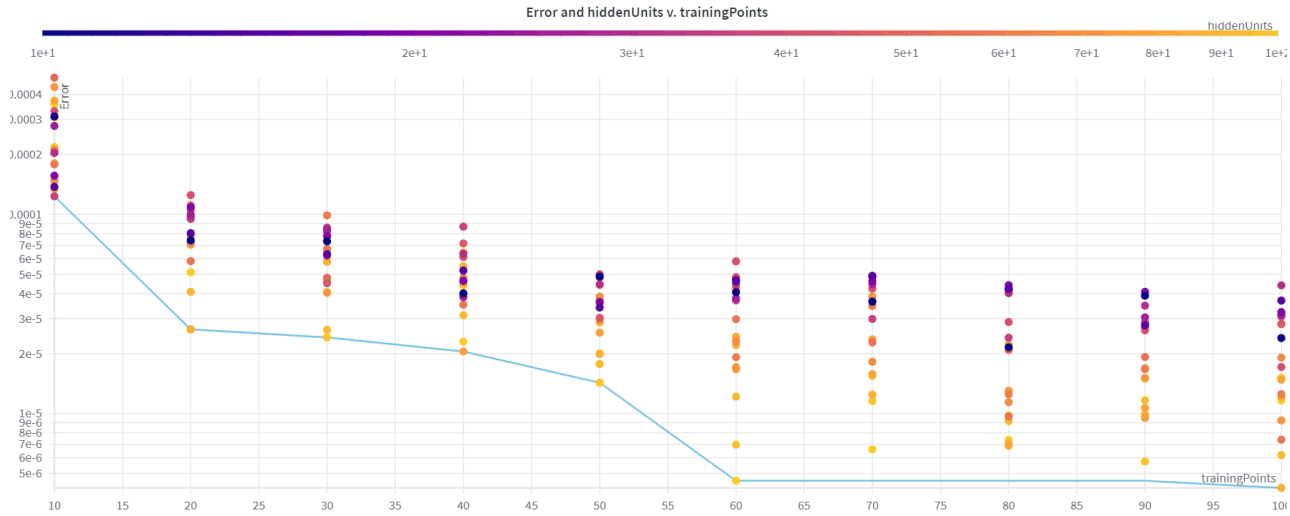
Mean time taken: 0.072410 seconds
Standard deviation: 0.063957 seconds

Error plot using Finite Element Method.

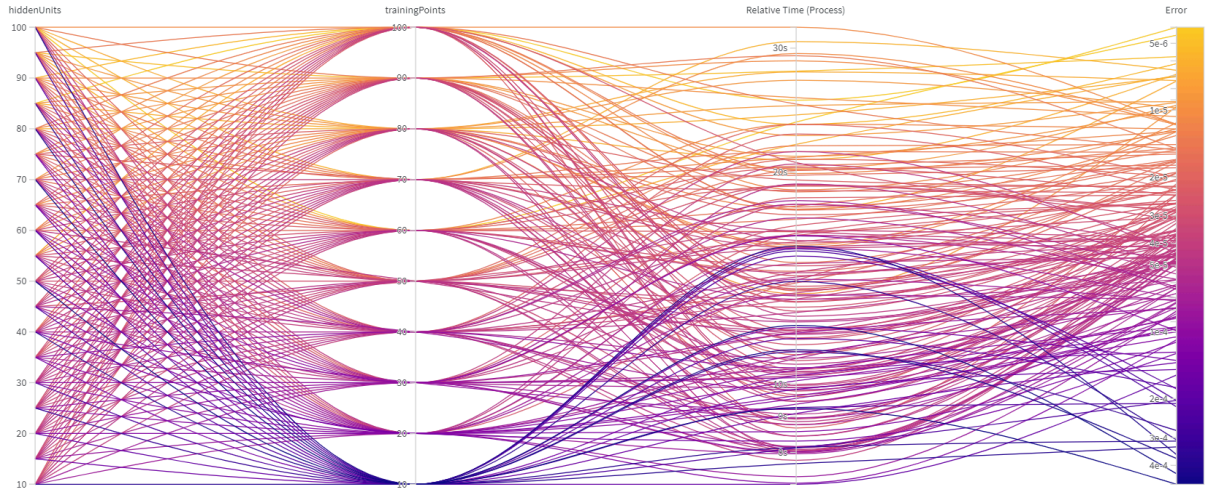


Time Taken:

Mean time taken: 0.000904 seconds
Standard deviation: 0.002583 seconds



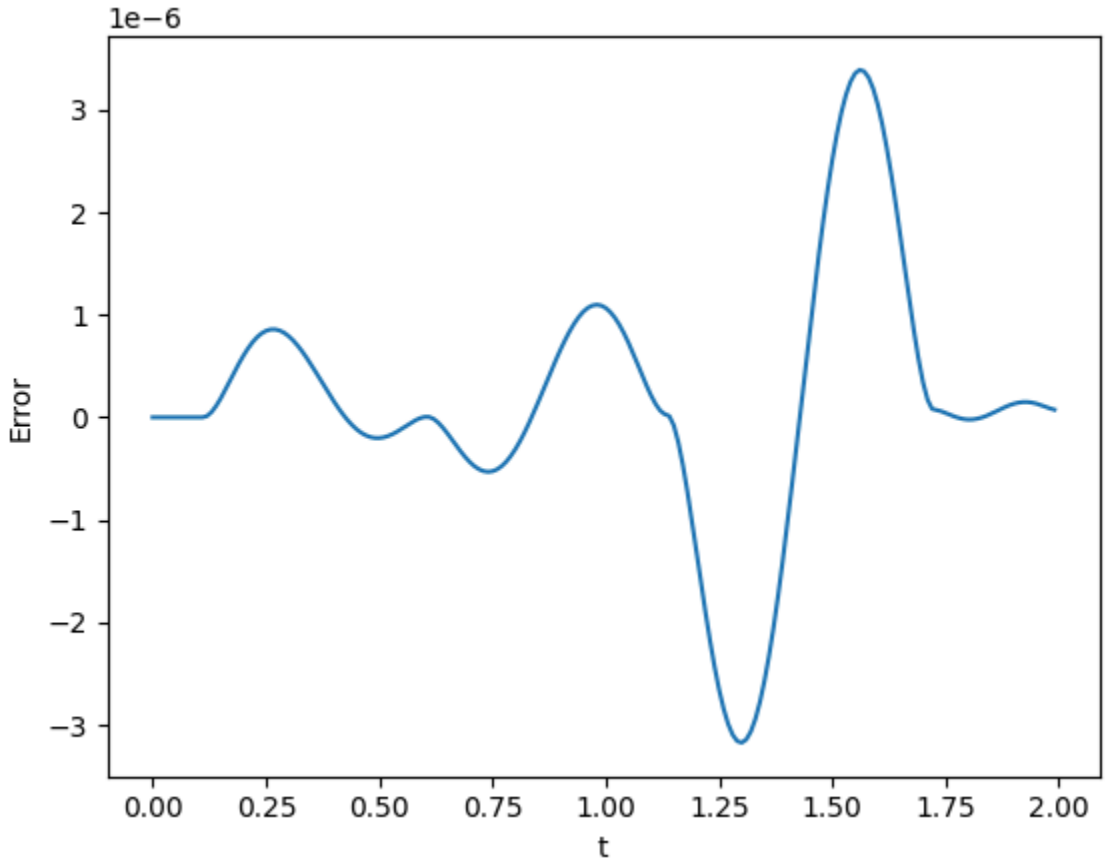
Y-axis represents Error, X-axis represents training points and color bar(z-axis) represents the no. of hidden units



Error here is the L2 Norm of the difference between analytical solution and trail solution

Using grid-search to find the best parameters: we can clearly see that no. of training points matters the most in terms of improving accuracy (but increasing training point can increase the relative time) and the explanation for that is we don't need high no. of hidden units to explain the function, since the solution is not a complex function relatively low hidden unit could approximate the given function, and increasing training points gives more points to interpolate the function however in this case high no. of hidden unit has better accuracy for the given training point. Note that all the runs are averaged over 20 iterations hence, relative time taken for actual calculation will be approx (relative time)/20, which is almost similar to FEM methods.

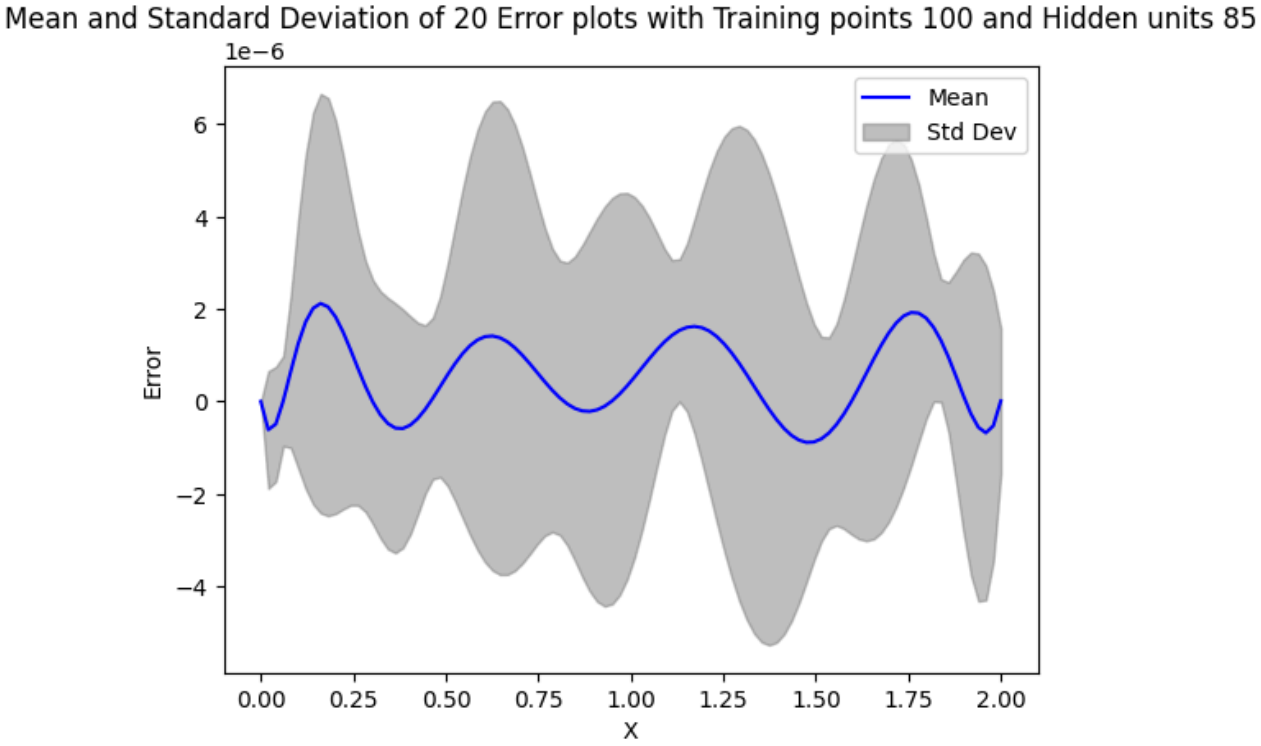
Error plot using Finite Element Method.



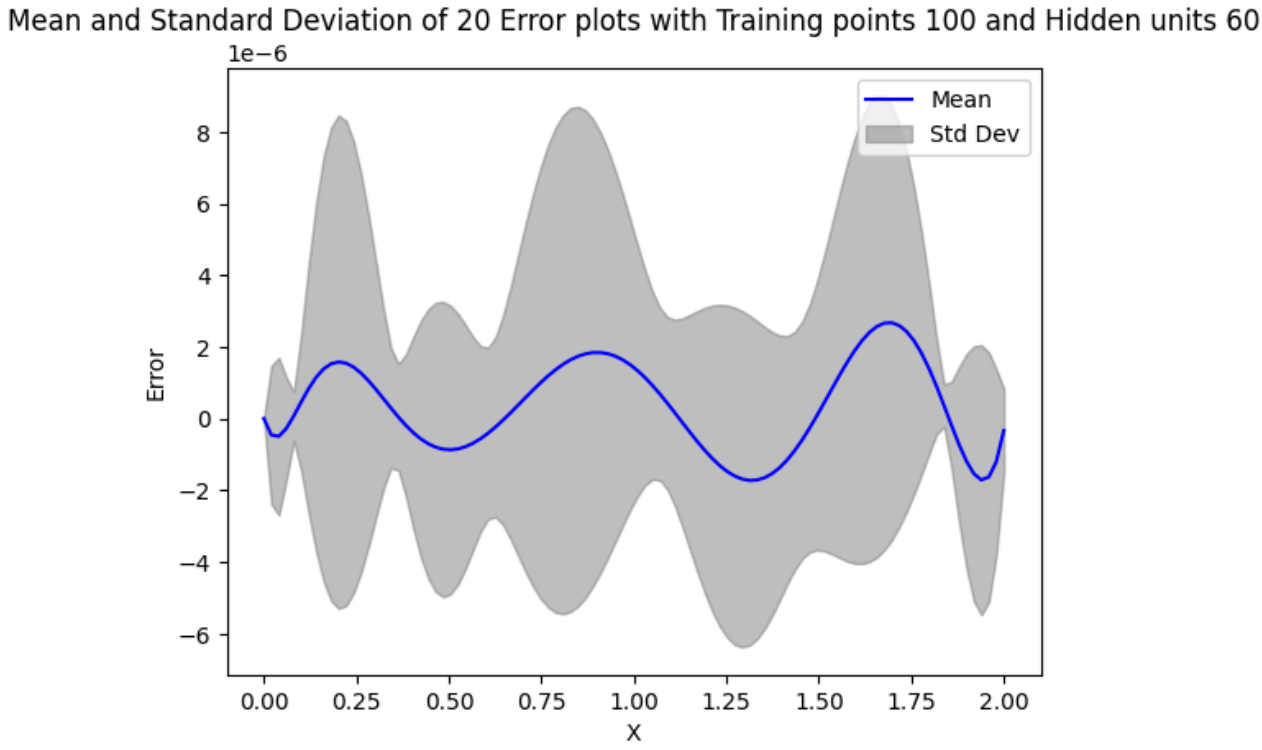
Time Taken:

Mean time taken: 0.001658 seconds
Standard deviation: 0.001624 seconds

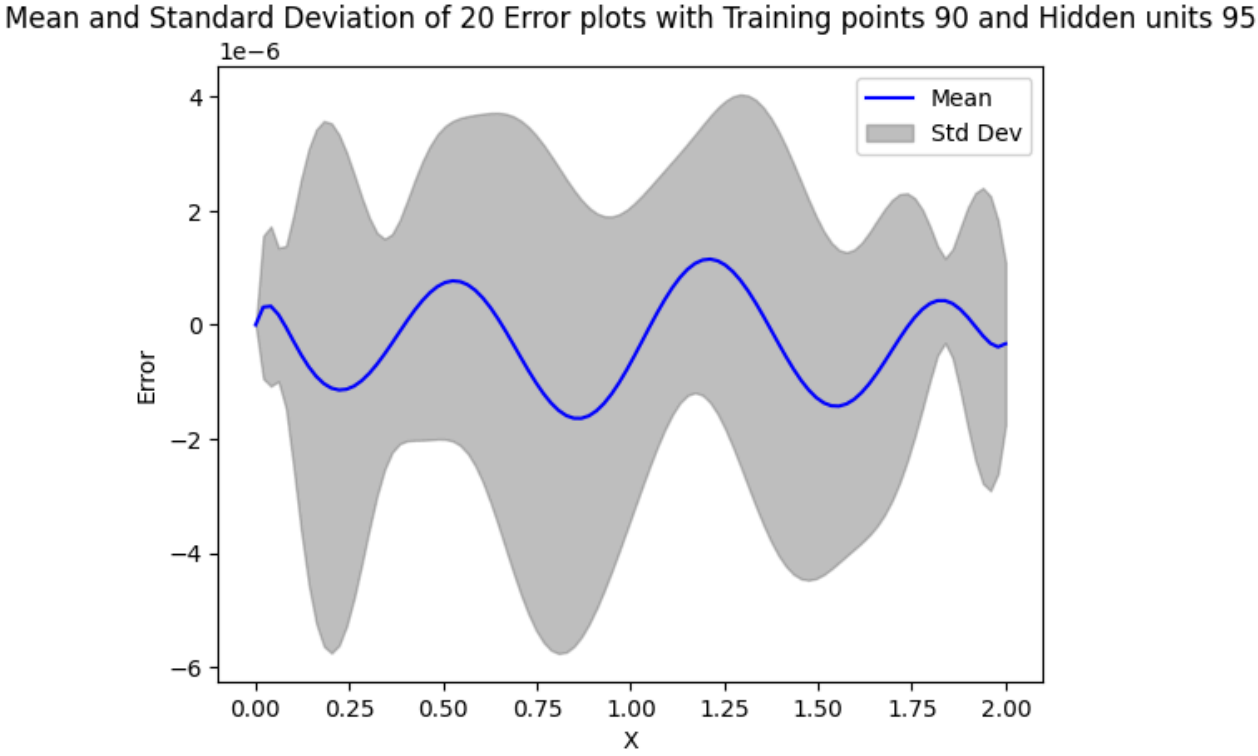
Top 3 Parameters



Mean time taken: 0.874376 seconds
Standard deviation: 0.638585 seconds



Mean time taken: 0.548392 seconds
Standard deviation: 0.364957 seconds



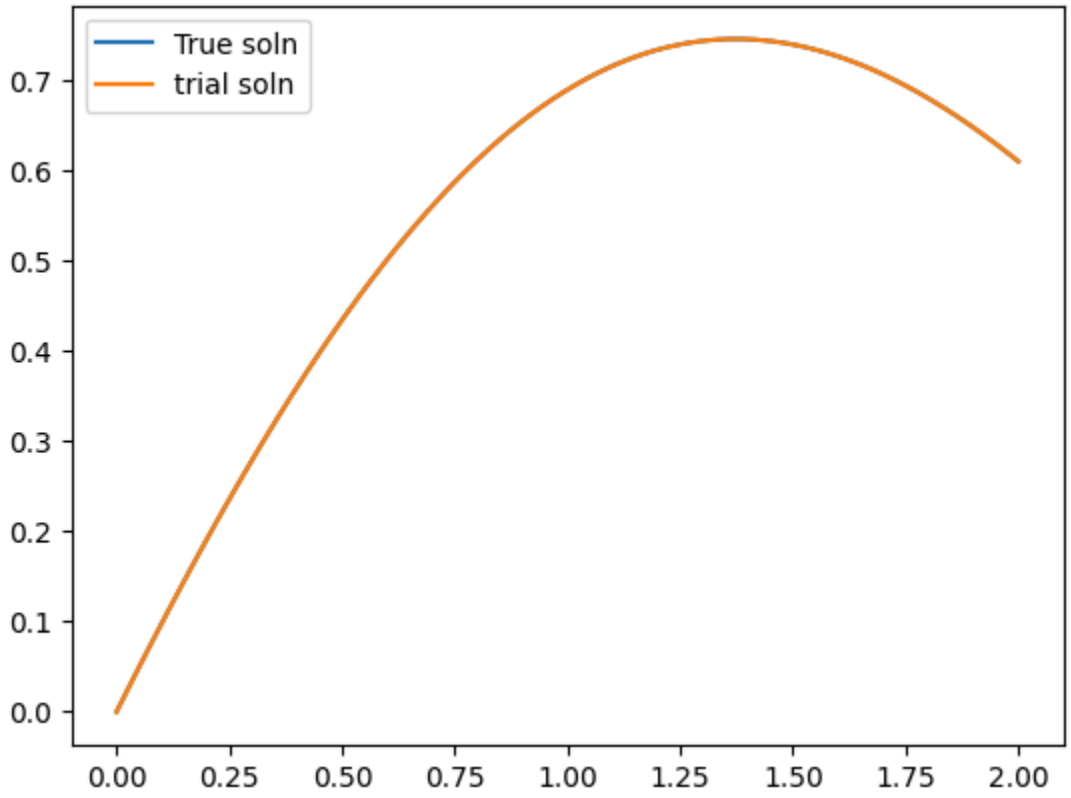
Mean time taken: 0.884956 seconds
Standard deviation: 0.467875 seconds

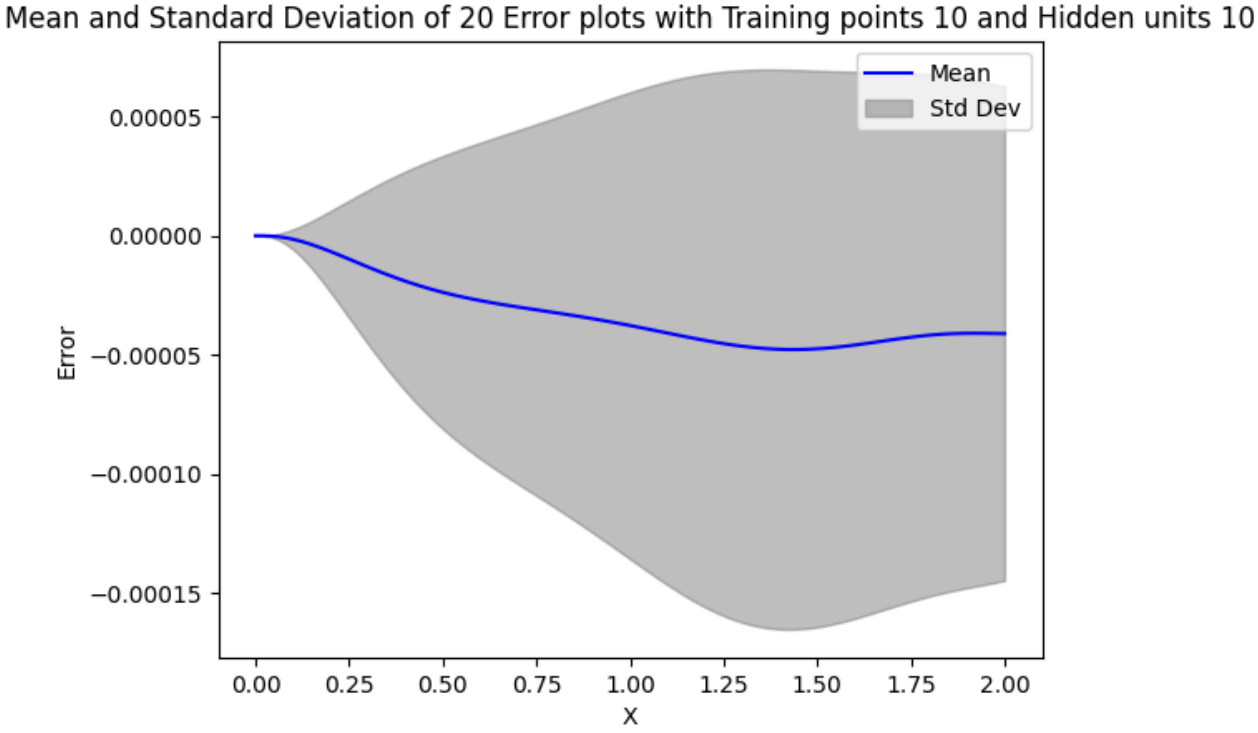
Ques 3(a): (IVP)

$$\frac{d^2\Psi}{dx^2} + \frac{1}{5}\frac{d\Psi}{dx} + \Psi = -\frac{1}{5}e^{-(x/5)}\cos x$$

True Soln:

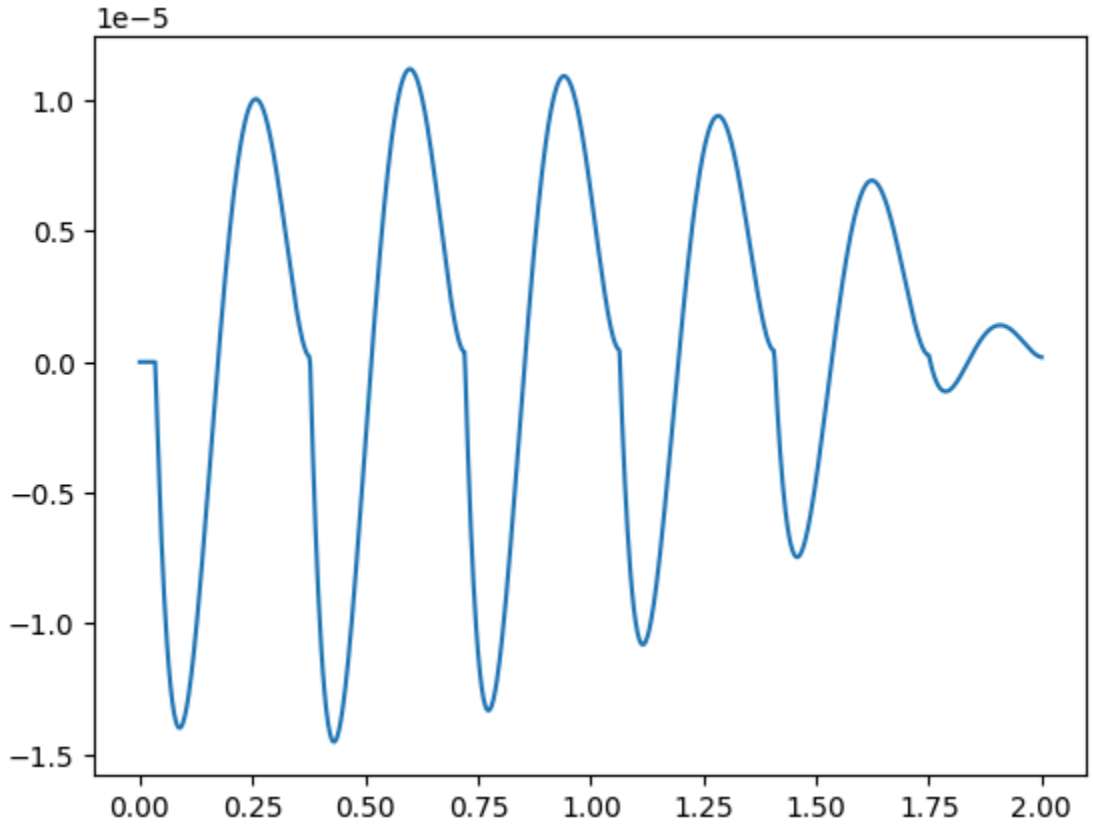
$$e^{-(x/5)}\sin(x)$$





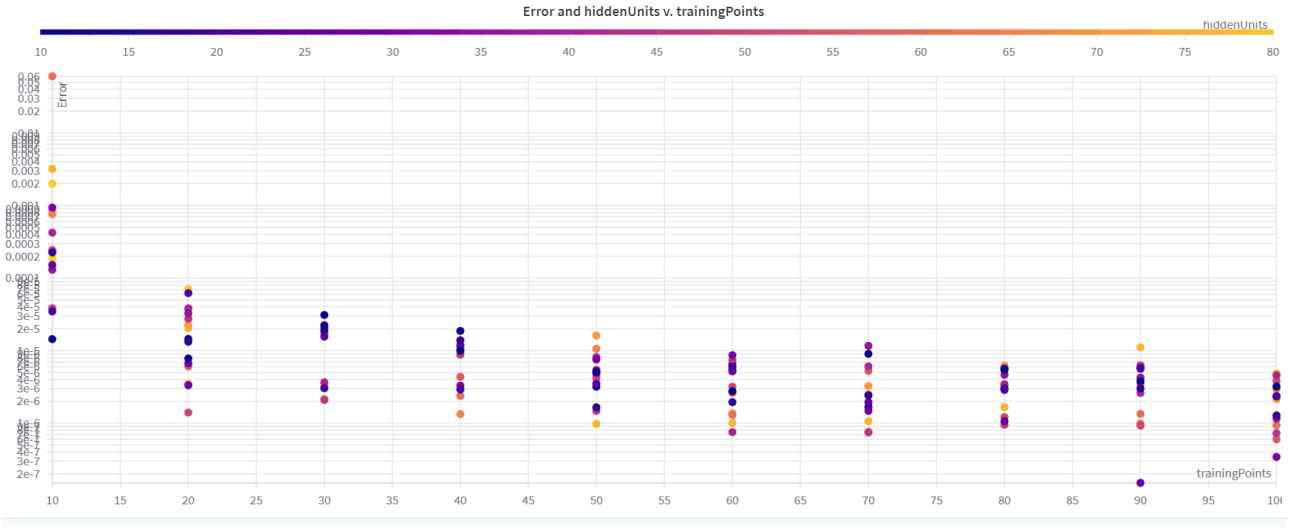
Mean time taken: 0.085349 seconds
Standard deviation: 0.056551 seconds

Error plot using Finite Element Method.

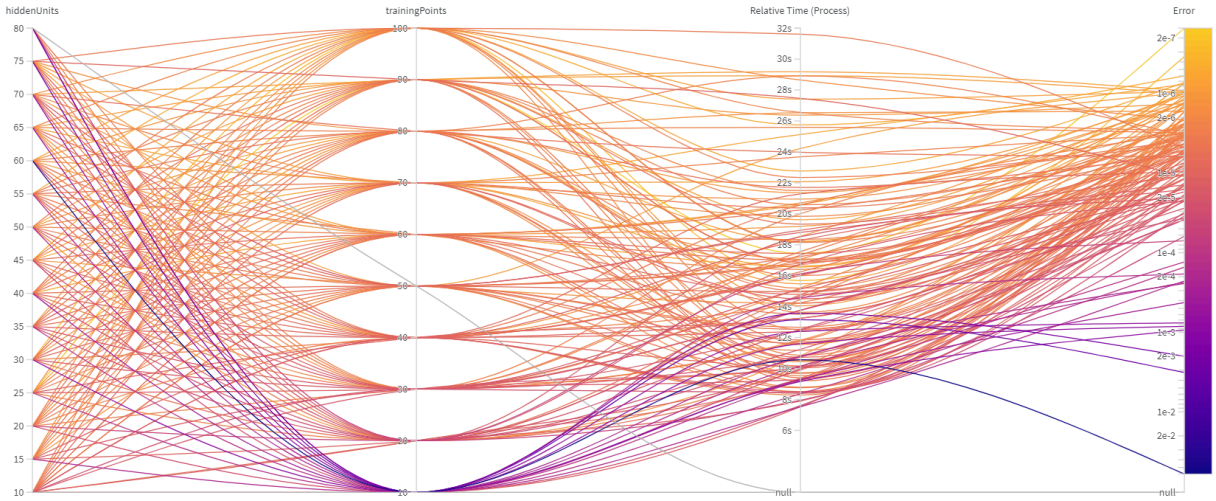


Time Taken:

Mean time taken: 0.003915 seconds
Standard deviation: 0.002751 seconds



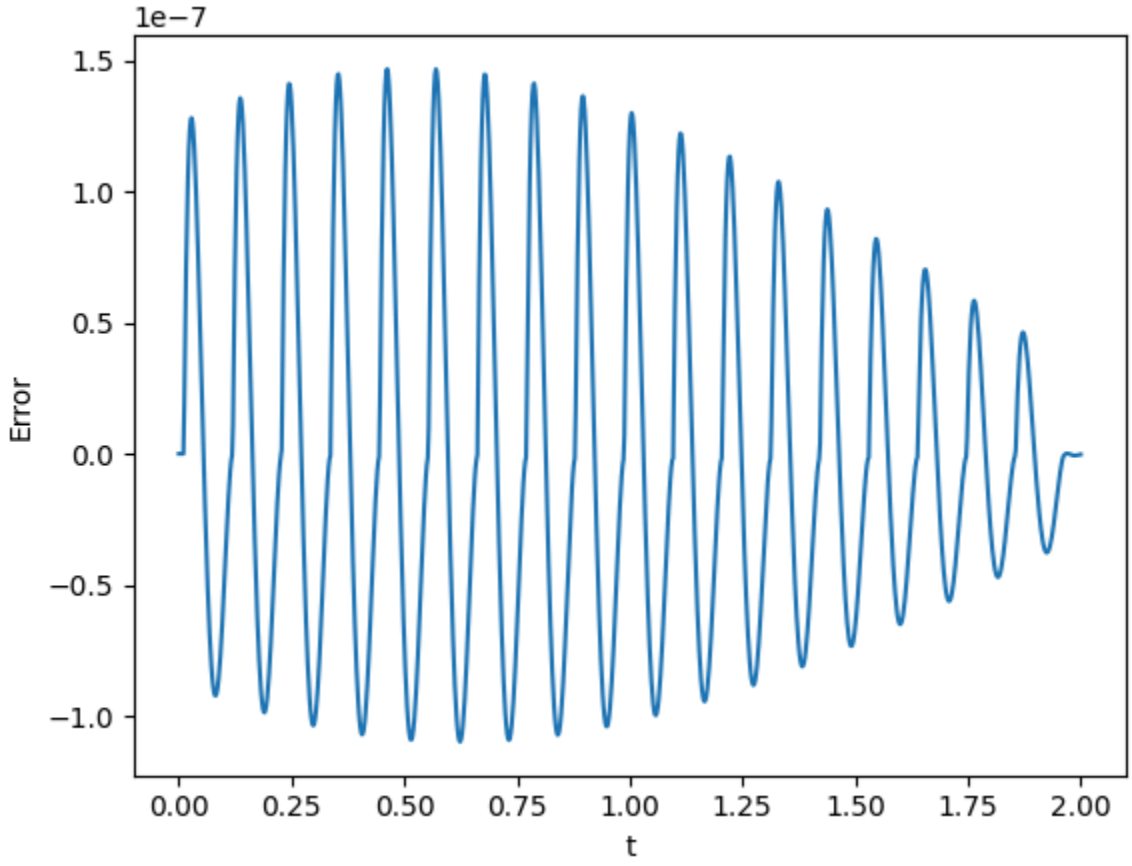
Y-axis represents Error, X-axis represents training points and color bar(z-axis) represents the no. of hidden units



Error here is the L2 Norm of the difference between analytical solution and trail solution

Using grid-search to find the best parameters: we can clearly see that no. of training points matters the most in terms of improving accuracy (but increasing training point can increase the relative time) and the explanation for that is we don't need high no. of hidden units to explain the function, since the solution is not a complex function relatively low hidden unit could approximate the given function, and increasing training points gives more points to interpolate the function however increasing the no. of hidden unit may improve the accuracy for the given training point but its not conclusive. Note that all the runs are averaged over 20 iterations hence, relative time taken for actual calculation will be approx (relative time)/20, which is almost similar to FEM methods.

Error plot using Finite Element Method.

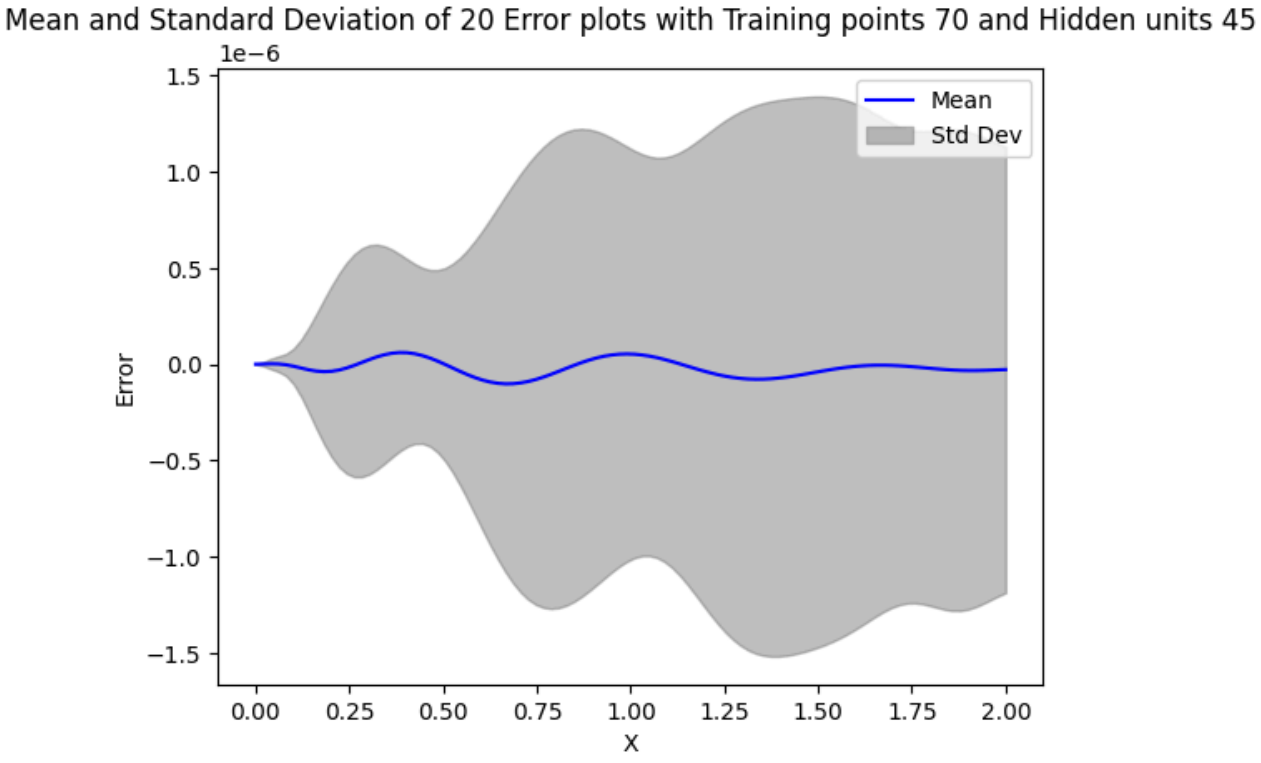


Time Taken:

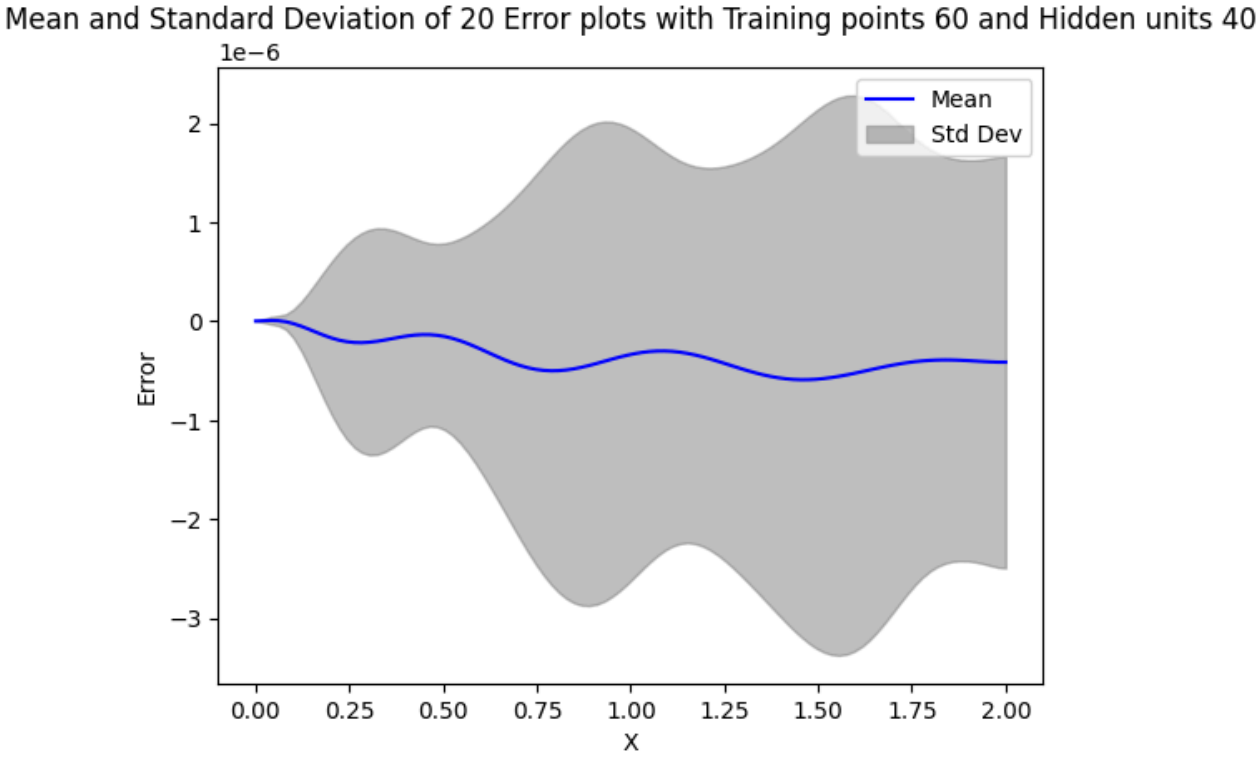
Mean time taken: 0.008768 seconds
Standard deviation: 0.005273 seconds

Top 3 Parameters

Mean time taken: 0.563798 seconds
Standard deviation: 0.344687 seconds



Mean time taken: 0.852681 seconds
Standard deviation: 0.675234 seconds



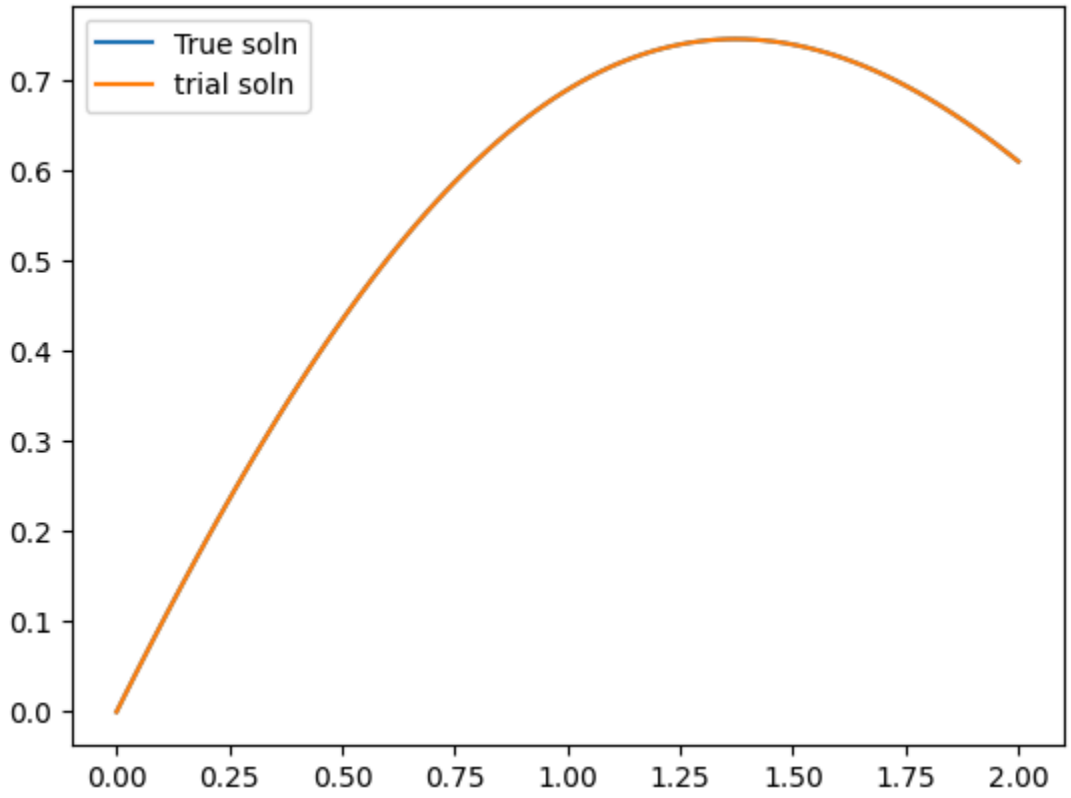
Mean time taken: 0.575902 seconds
Standard deviation: 0.378526 seconds

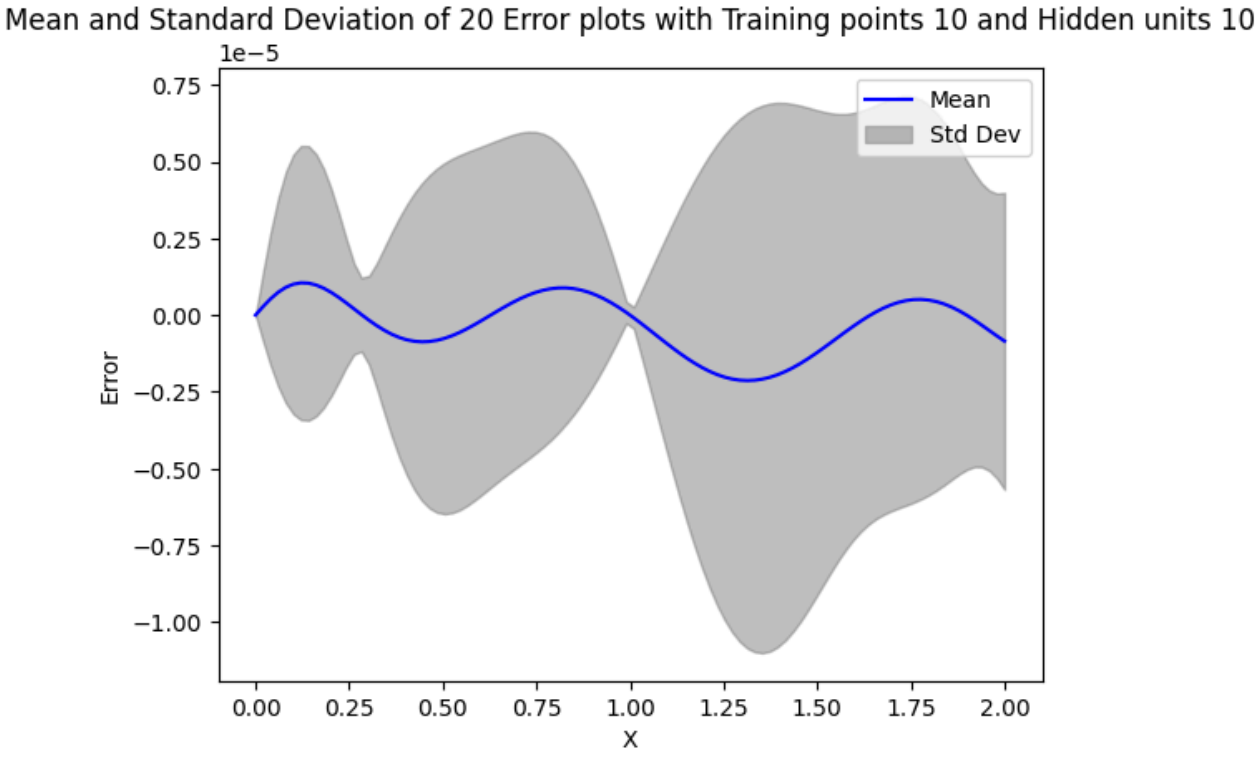
Ques 3(b): (BVP)

$$\frac{d^2\Psi}{dx^2} + \frac{1}{5}\frac{d\Psi}{dx} + \Psi = -\frac{1}{5}e^{-(x/5)}\cos x$$

True Soln:

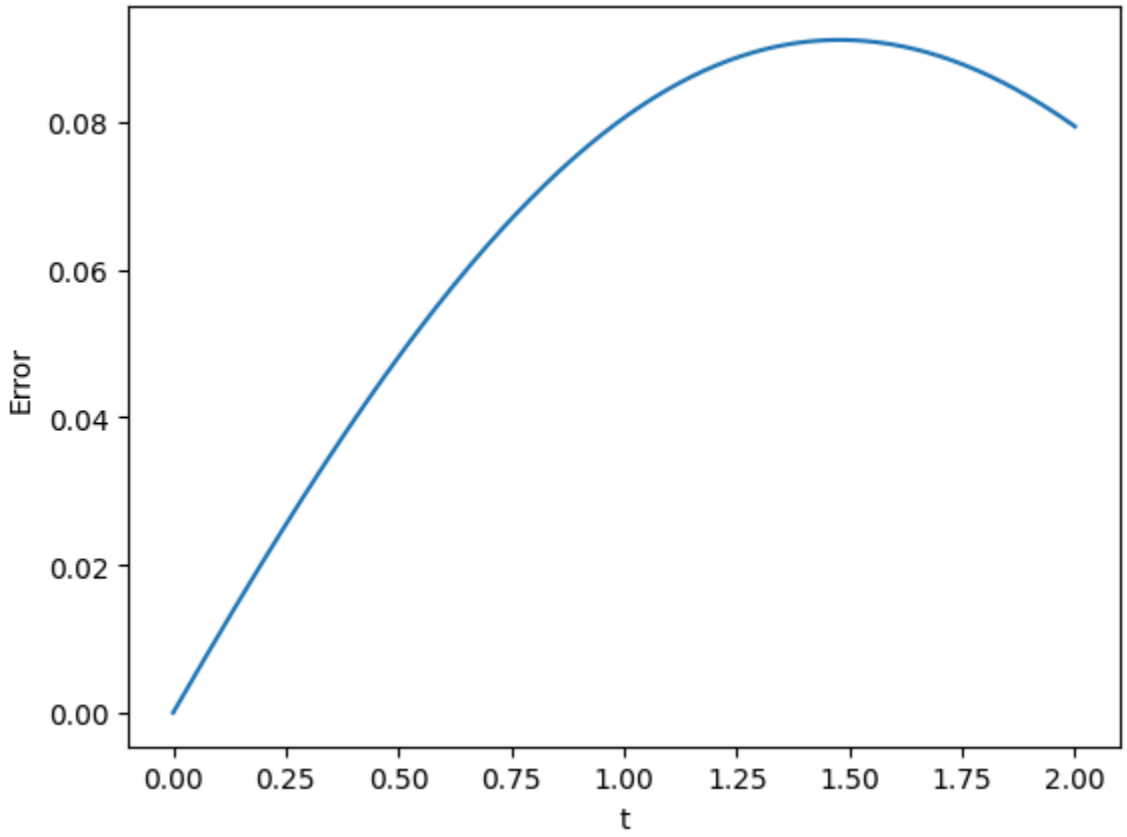
$$e^{-(x/5)}\sin(x)$$





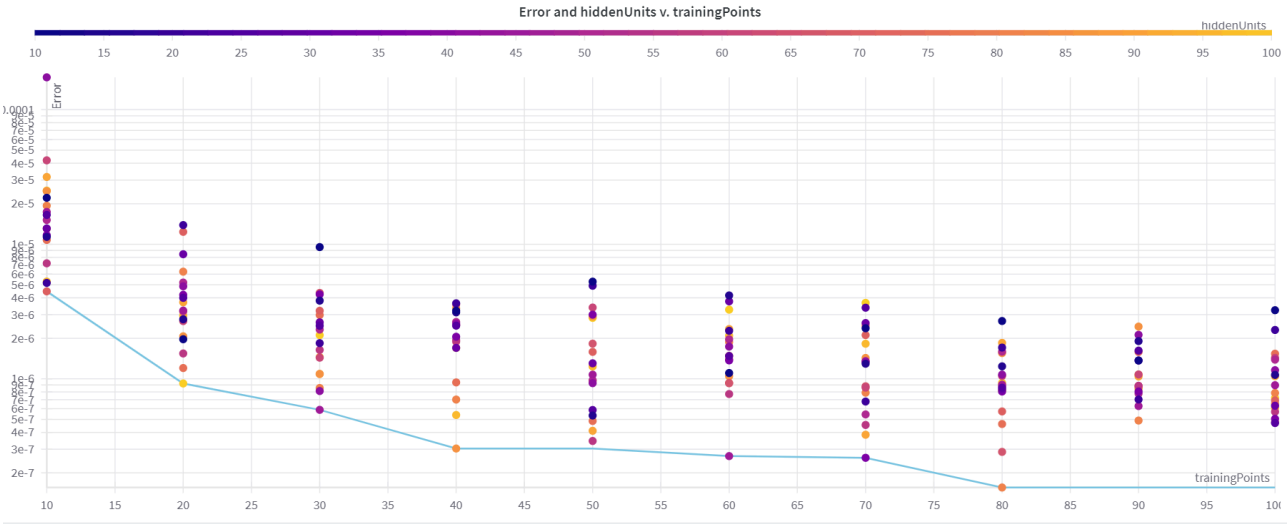
Mean time taken: 0.092788 seconds
Standard deviation: 0.078678 seconds

Error plot using Finite Element Method.
(obs: error can't be reduced further.)

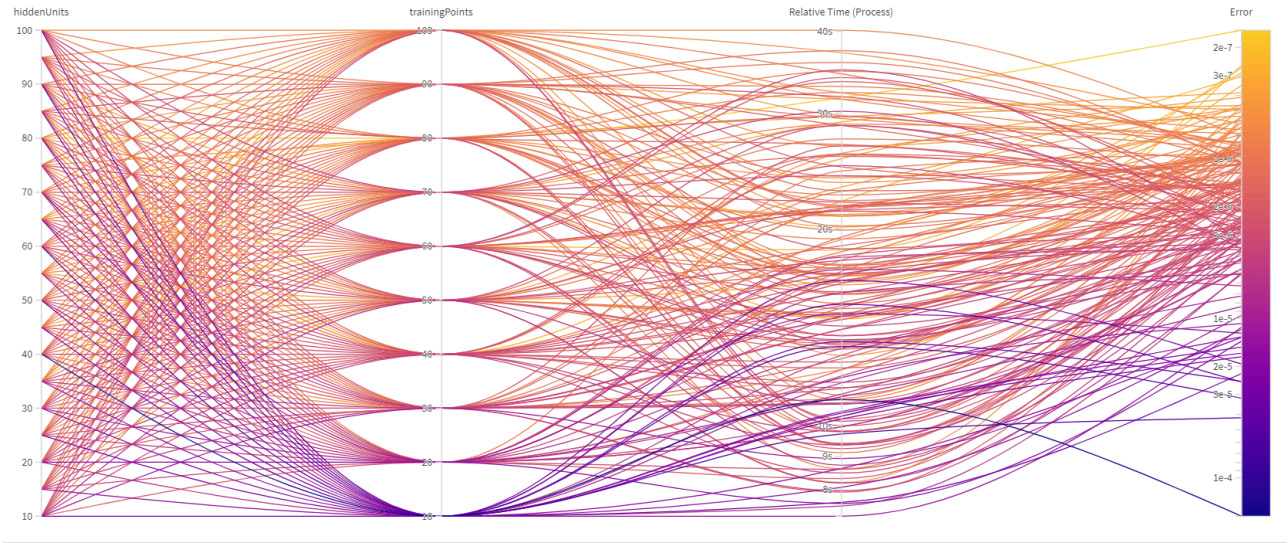


Time Taken:

Mean time taken: 0.014992 seconds
Standard deviation: 0.013378 seconds



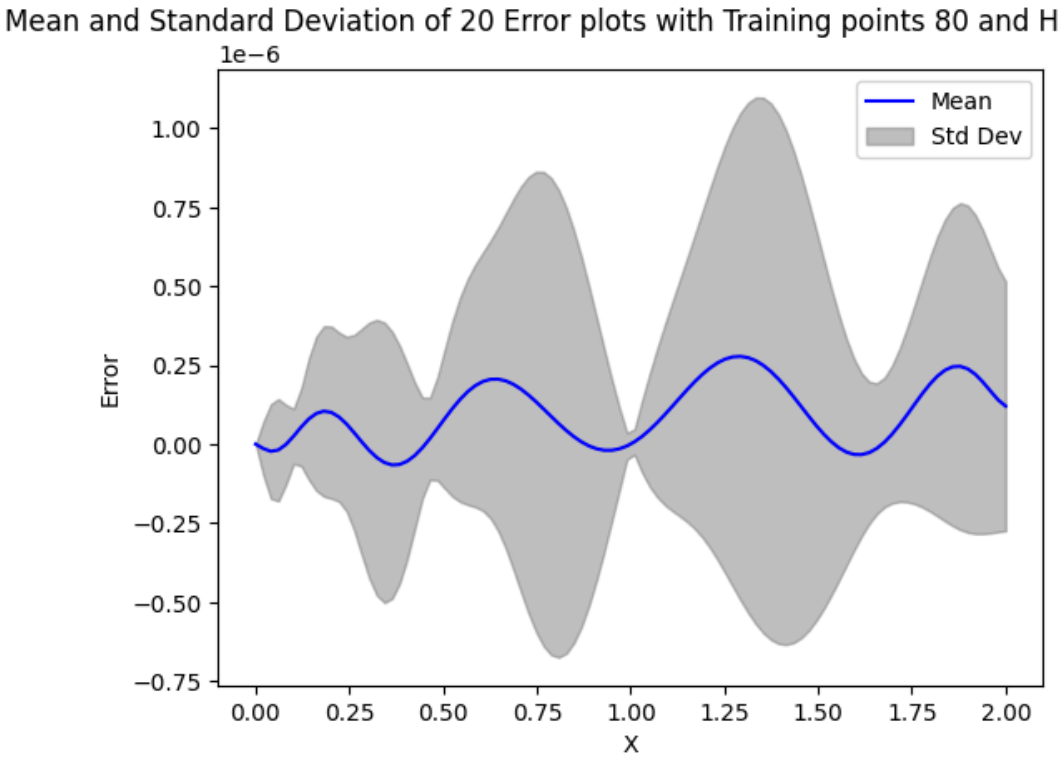
Y-axis represents Error, X-axis represents training points and color bar(z-axis) represents the no. of hidden units



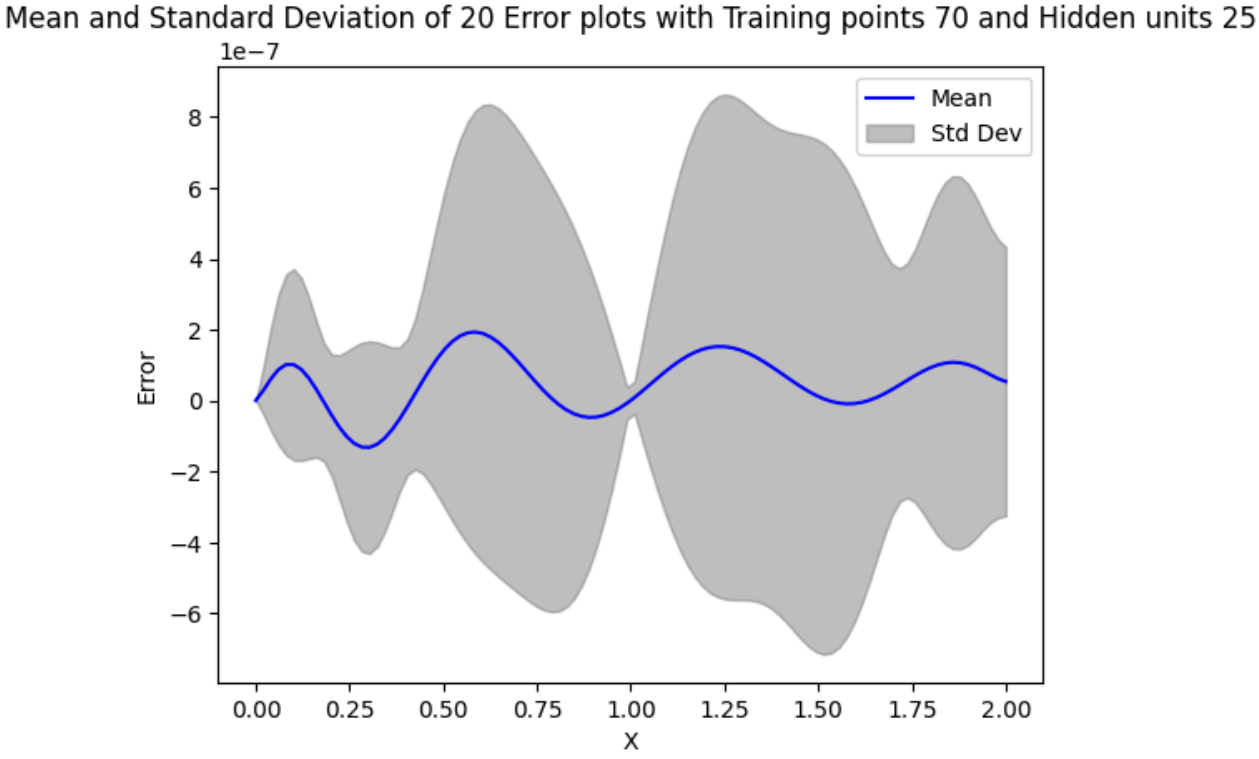
Error here is the L2 Norm of the difference between analytical solution and trail solution

Using grid-search to find the best parameters: we can clearly see that no. of training points matters the most in terms of improving accuracy (but increasing training point can increase the relative time) and the explanation for that is we don't need high no. of hidden units to explain the function, since the solution is not a complex function relatively low hidden unit could approximate the given function, and increasing training points gives more points to interpolate the function however increasing the no. of hidden unit may improve the accuracy for the given training point but its not conclusive. Note that all the runs are averaged over 20 iterations hence, relative time taken for actual calculation will be approx (relative time)/20, which is almost similar to FEM methods.

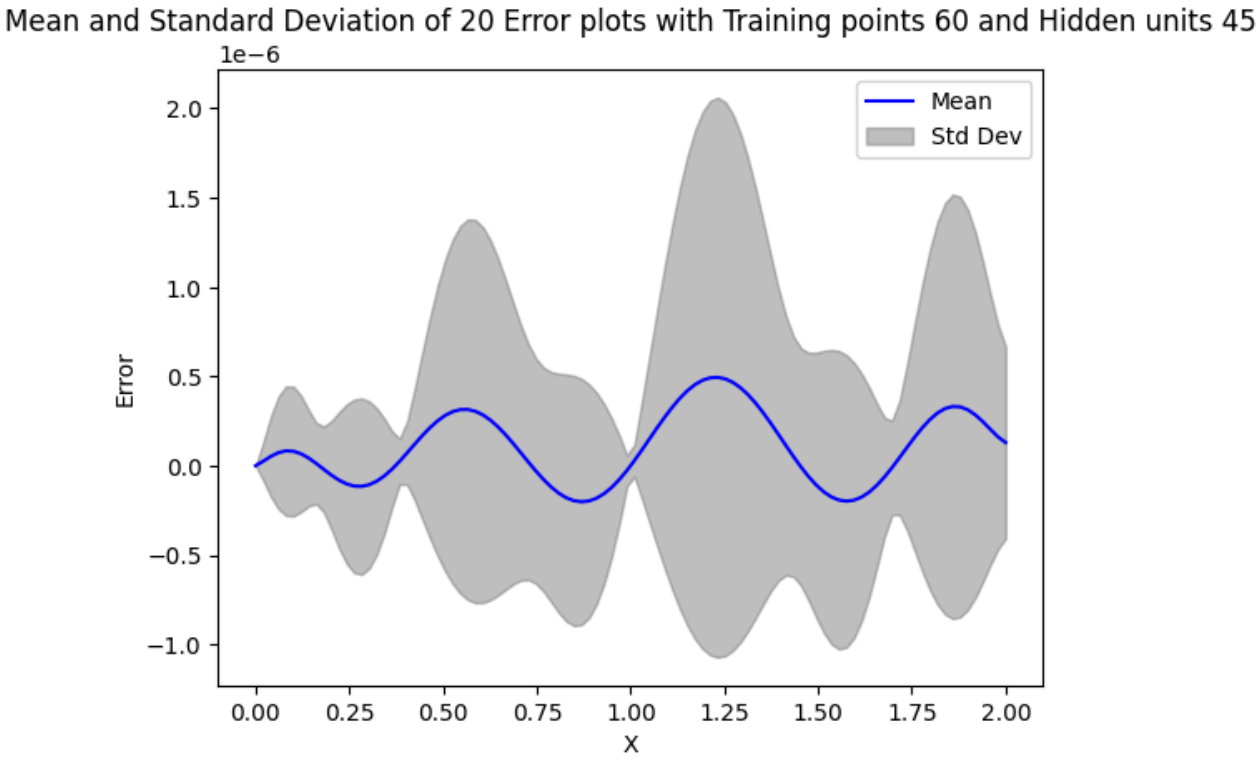
Top 3 Parameters



Mean time taken: 0.881904 seconds
Standard deviation: 0.469948 seconds



Mean time taken: 0.412660 seconds
Standard deviation: 0.227955 seconds



Mean time taken: 0.514710 seconds
Standard deviation: 0.273341 seconds

4. Unstability of System of Coupled Nonlinear ODE

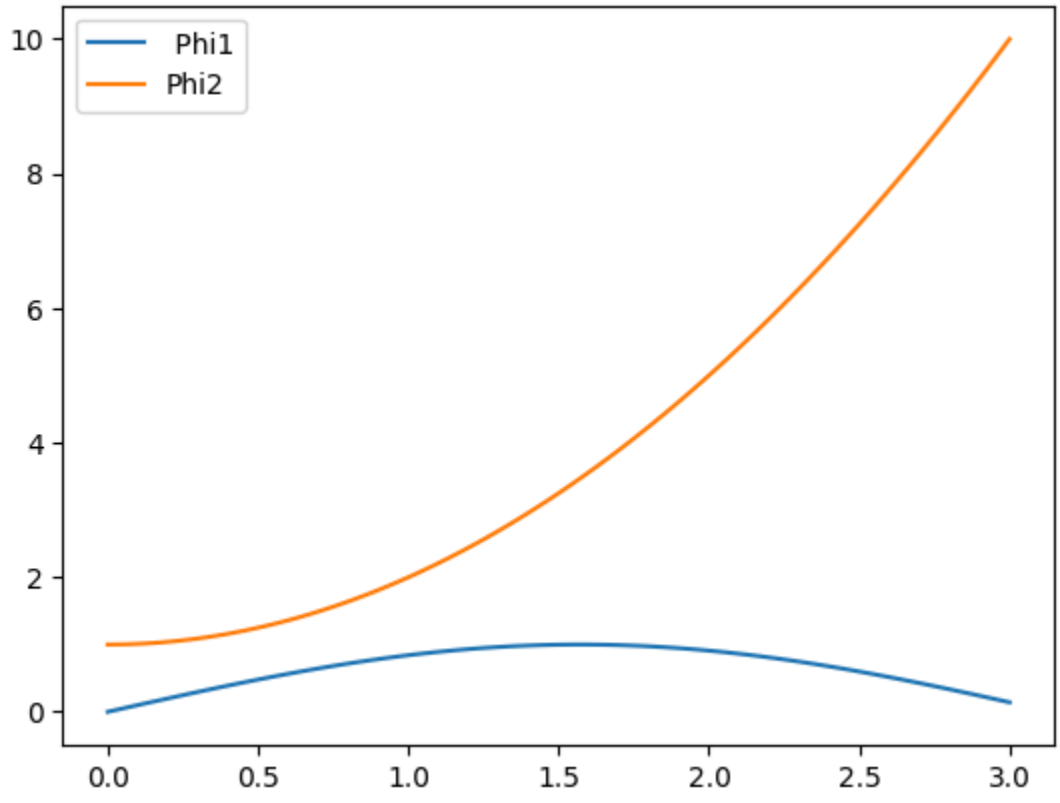
Ques 4:

$$\frac{d\Psi_1}{dx} = \cos(x) + \Psi_1^2 + \Psi_2 - (1 + x^2 + \sin^2(x)) \quad (30)$$

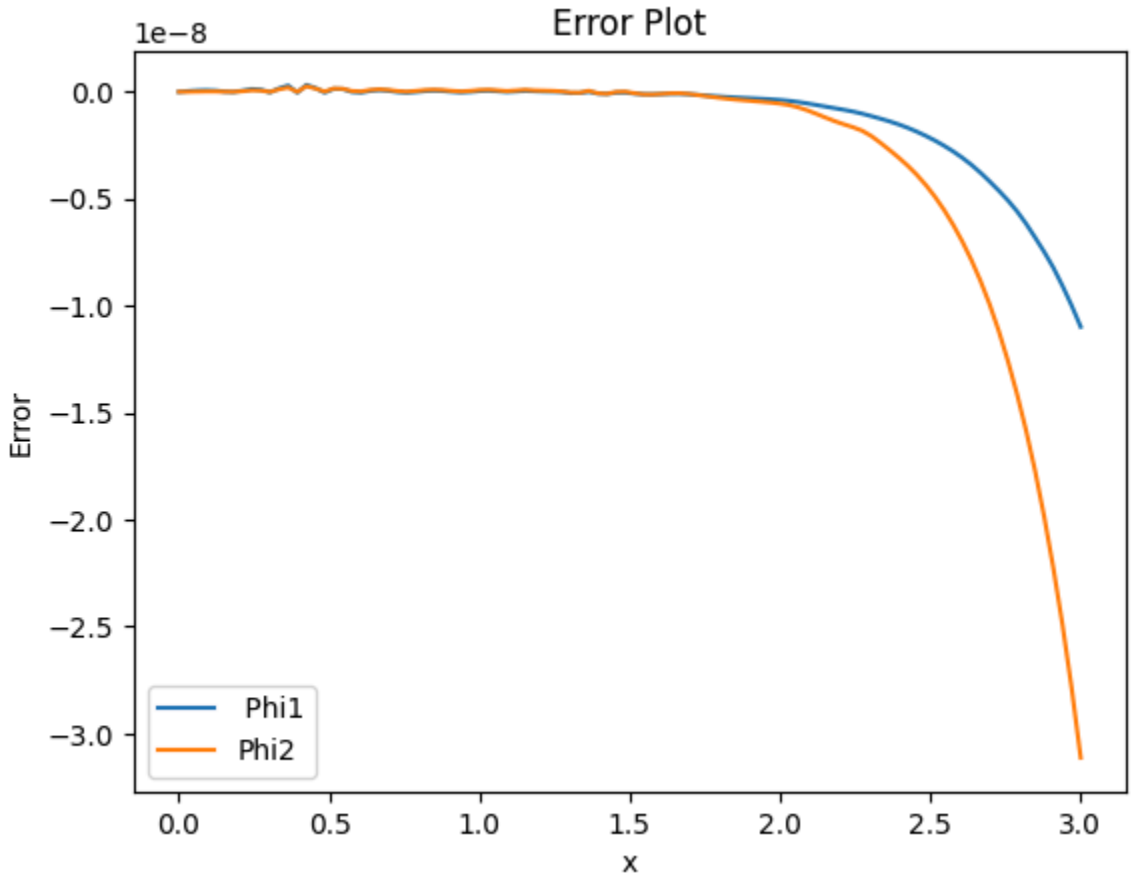
$$\frac{d\Psi_2}{dx} = 2x - (1 + x^2)\sin(x) + \Psi_1\Psi_2 \quad (31)$$

ı $x \in [0, 3]$ and $\Psi_1(0) = 0$ and $\Psi_2(0) = 1$. The analytic

True Soln:

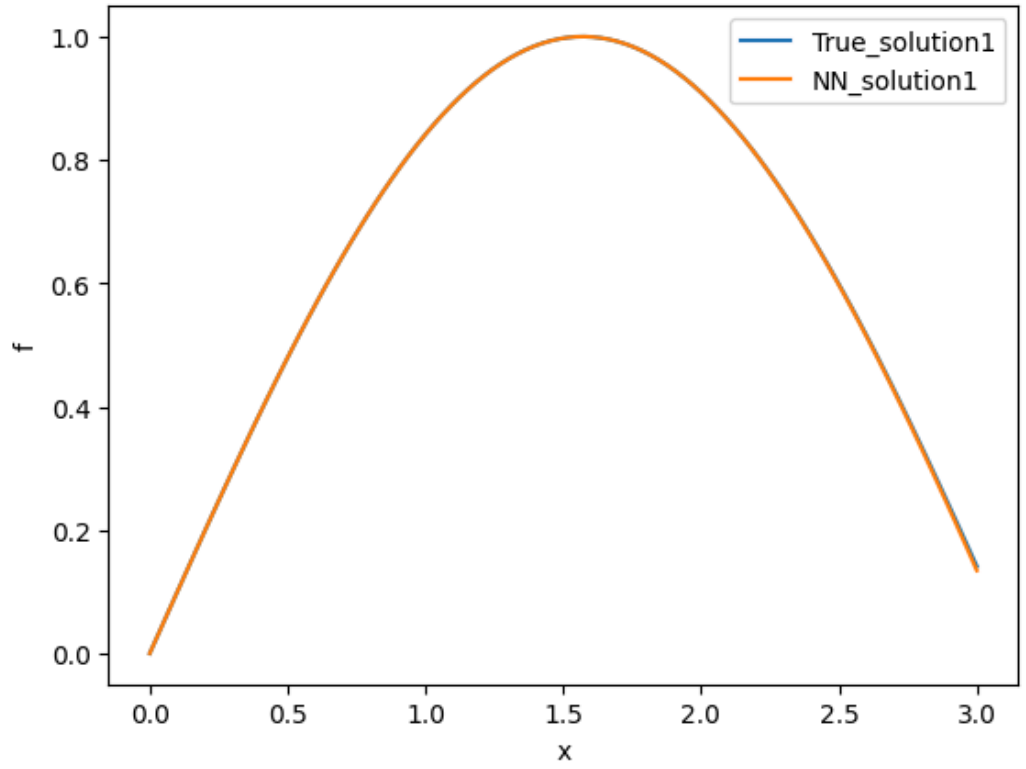


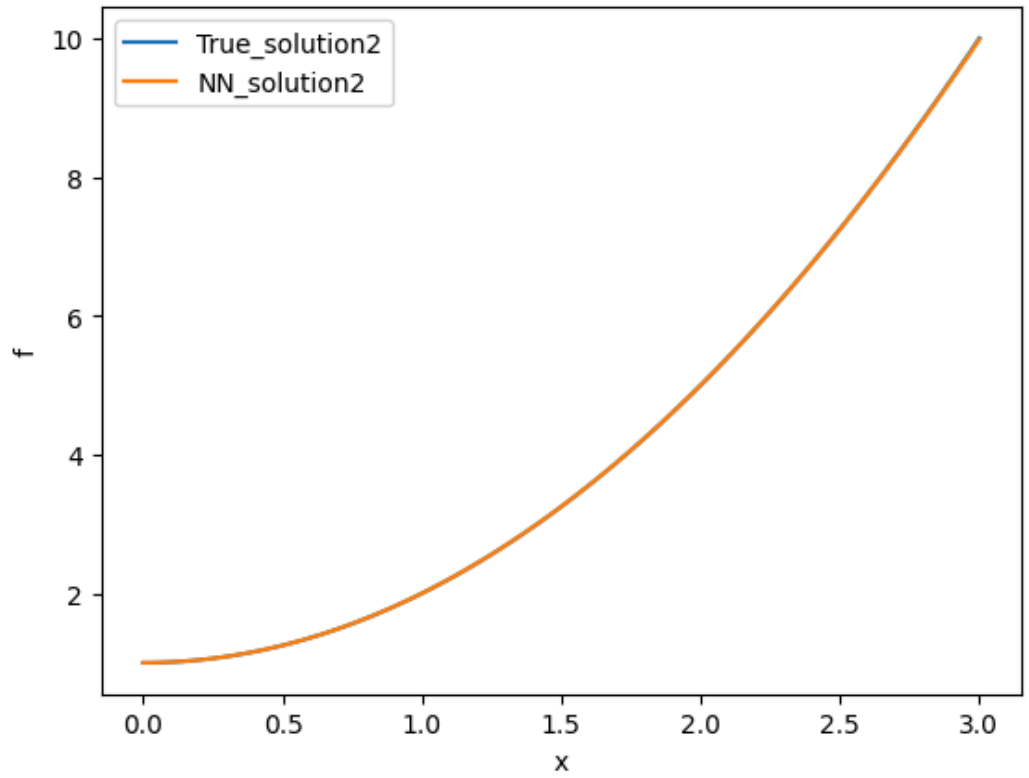
Error for Finite Element Method



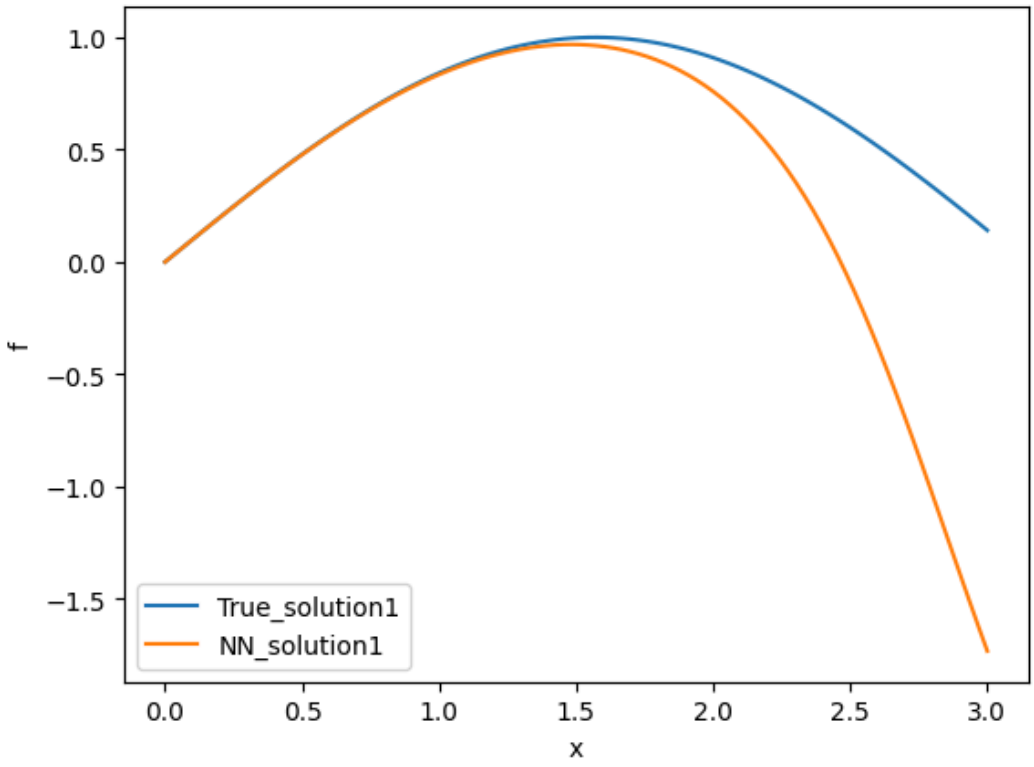
Time Taken:
0.032514095306396484

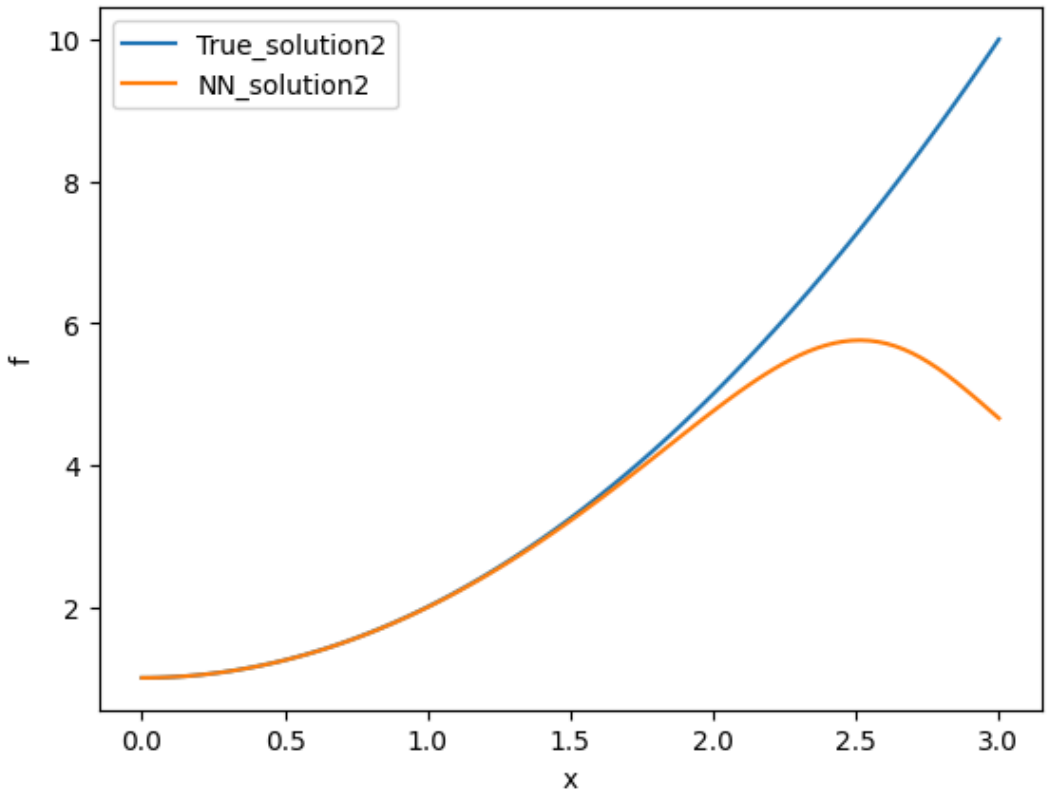
Solution 1:


True_solution1
NN_solution1
1.0
0.8
0.6
0.4
0.2
0.0
f
0.0
0.5
1.0
1.5
2.0
2.5
3.0
x

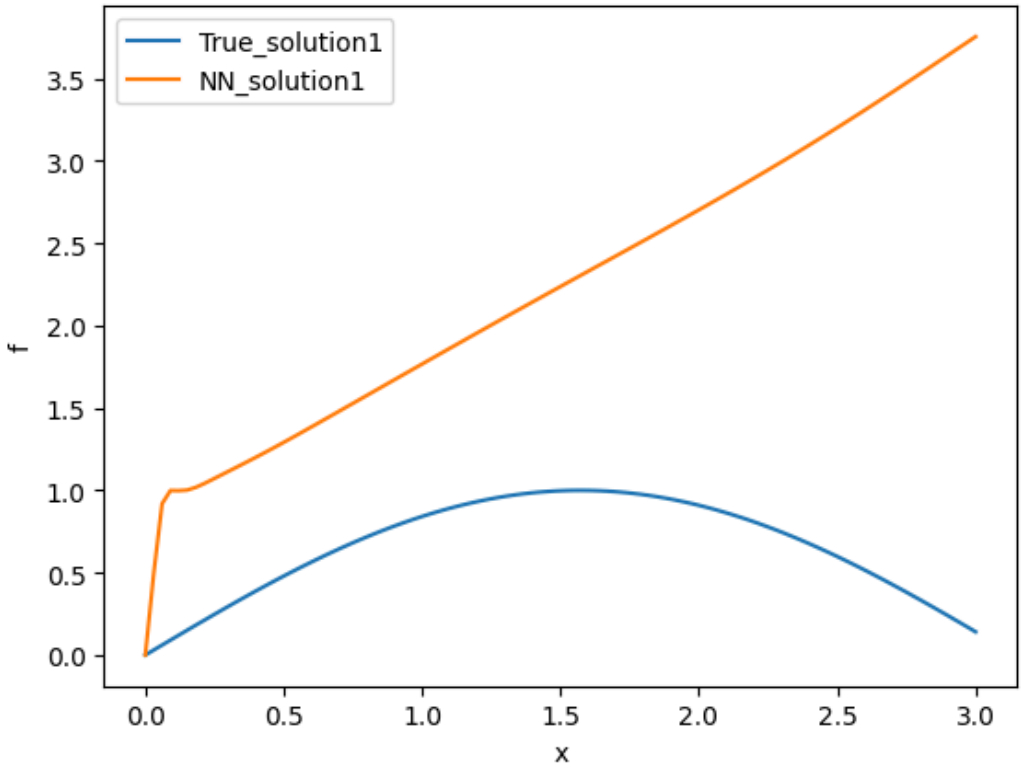

True_solution2
NN_solution2
10
8
6
4
2
f
0.0
0.5
1.0
1.5
2.0
2.5
3.0
x

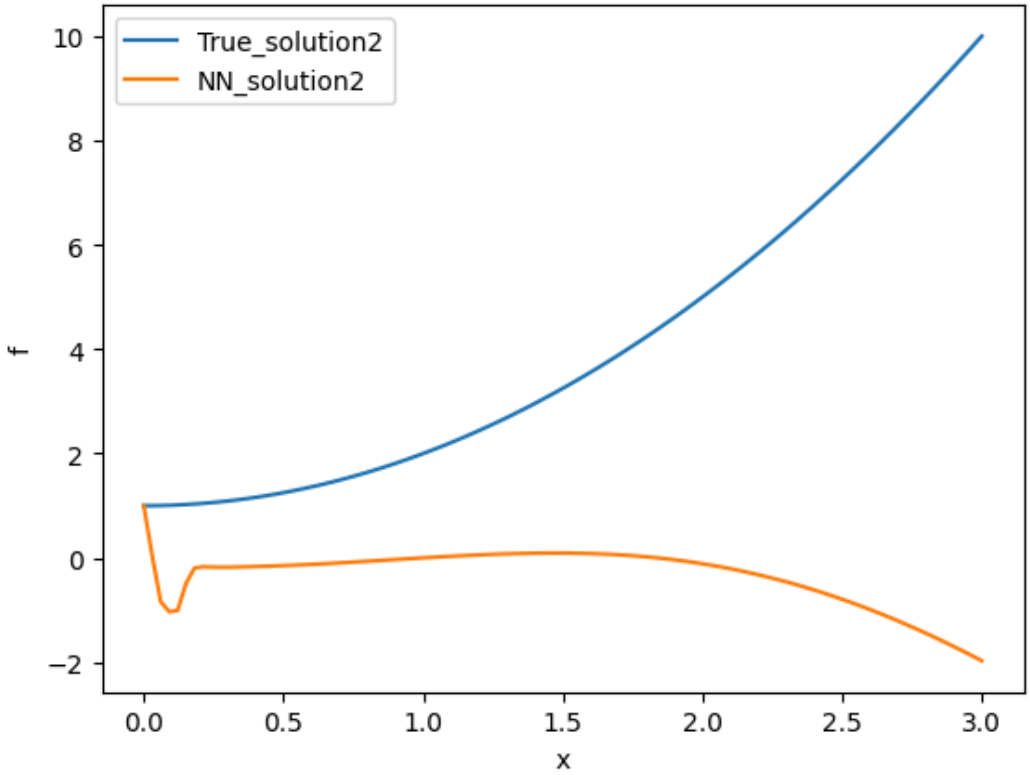
Solution 2:


1.0
0.5
0.0
−0.5
−1.0
−1.5
f
True_solution1
NN_solution1
0.0
0.5
1.0
1.5
2.0
2.5
3.0
x


10
8
6
4
2
f
True_solution2
NN_solution2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
x

Solution 3:


True_solution1
NN_solution1
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
f
0.0
0.5
1.0
1.5
2.0
2.5
3.0
x


True_solution2
NN_solution2
10
8
6
4
2
0
−2
f
0.0
0.5
1.0
1.5
2.0
2.5
3.0
x

Observation:

1. Training Error is very low, where as the accuracy wrt true solution is very low(in last 2 solutions) which shows that this NN Solution minimises the error function which is supposed to be,

$$E[\vec{p}] = \sum_{k=1}^{K} \sum_{i} \left\{ \frac{d\Psi_{t_k}(x_i)}{dx} - f_k(x_i, \Psi_{t_1}, \Psi_{t_2}, \cdots, \Psi_{t_K}) \right\}^2. \tag{20}$$

But not the error wrt true solution. It might indicates that the system is not numerically stable.

C. Coupled differential equation example

When discussing the calculation of cosmological phase transitions, we will study the solution of coupled nonlinear differential equations, for which no closed analytic form is known. Here, we will first show that such solutions can be obtained with our approach, for a case where an analytic solution is known. We consider

$$\frac{d\phi_1}{dx} - \cos x - \phi_1^2 - \phi_2 + 1 + x^2 + \sin^2 x = 0,$$
$$\frac{d\phi_2}{dx} - 2x + (1 + x^2)\sin x - \phi_1\phi_2 = 0, \qquad (6)$$

with boundary conditions

$$\phi_1(0) = 0, \qquad \phi_2(0) = 1. \qquad (7)$$

If the boundary conditions are set on one end of the domain, e.g., here at $x = 0$, it requires an increasingly elaborate network to maintain numerical stability for the solution over a large domain, e.g., where $x \gg 1$. This is due to small numerical instabilities during backpropagation because of the complexity of the loss hypersurface. If such numerical instability leads the network to choose a path that is in close proximity to the true solution, the NN can settle on a local minimum with a small value for the loss function. To solve this problem, we propose to incrementally extend the domain on which a solution should be found, by partitioning the training examples and increasing the number of partitions the NN is trained on in each step. If the weights the NN has learned in the previous step are then retained before training for the next step—i.e., the network only has to learn the function on the part of the

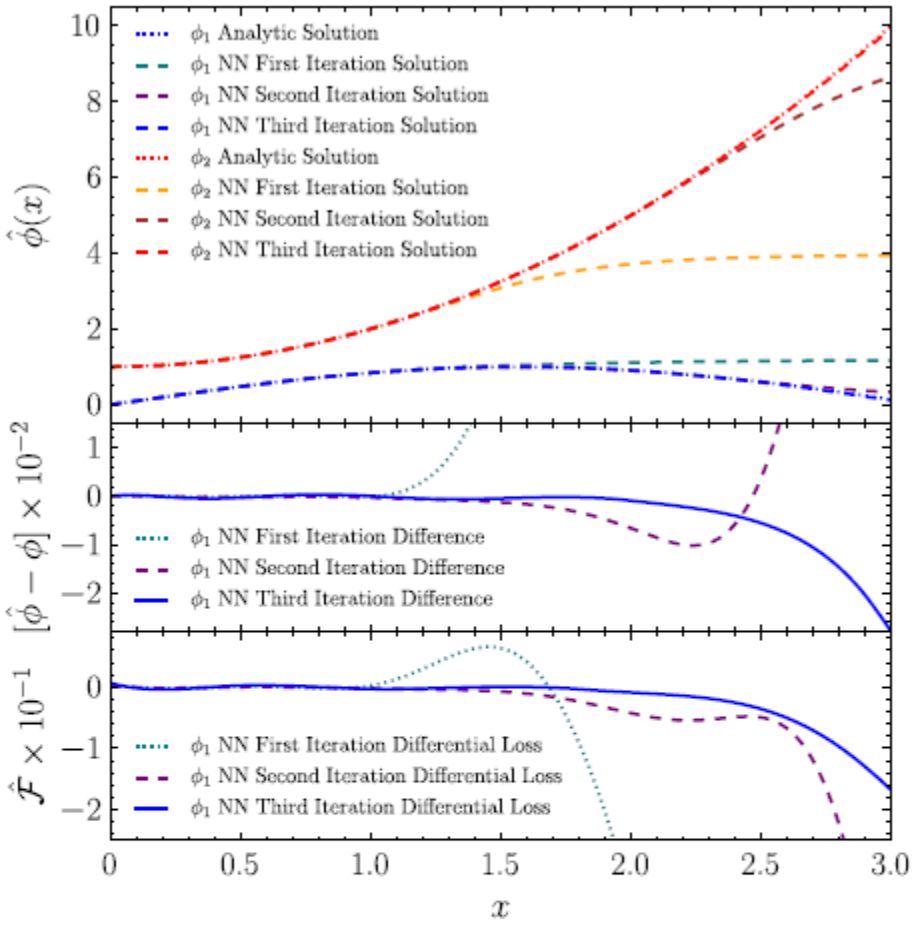


FIG. 2. The upper panel shows the solutions for the functions ϕ_1 and ϕ_2 to the coupled differential equation of Eq. (6). The middle panel displays the numerical difference between the analytic solution and the NN predicted solution for ϕ_1. The lower panel shows the differential contribution $\hat{\mathcal{F}}$ to the loss across the entire domain, from the equation for ϕ_1. The three NN curves in each panel correspond to the first, second and third iteration steps in the training of the network, with iterative increase of the training domain, as described in the text.

Here is the reference for the paper that discussed the same problem very briefly.
https://arxiv.org/abs/1902.05563

5. Analysis of Increasing Hidden units and Training points

Hypothesis: ($\lim_{nh \to \infty}$ ($\lim_{nt \to \infty}$ Error)) goes to zero

Continuing previous work, as all the experiments were done using small numbers of train and test data, And it shows increasing training data can decrease the error but nothing concretely can be said about hidden units. Those results were generated on google colab.

We are now increasing the no. of training dataset and hidden units from (100- 10k) and testing the same hypothesis. These results were generated on Aqua cluster(IITM).

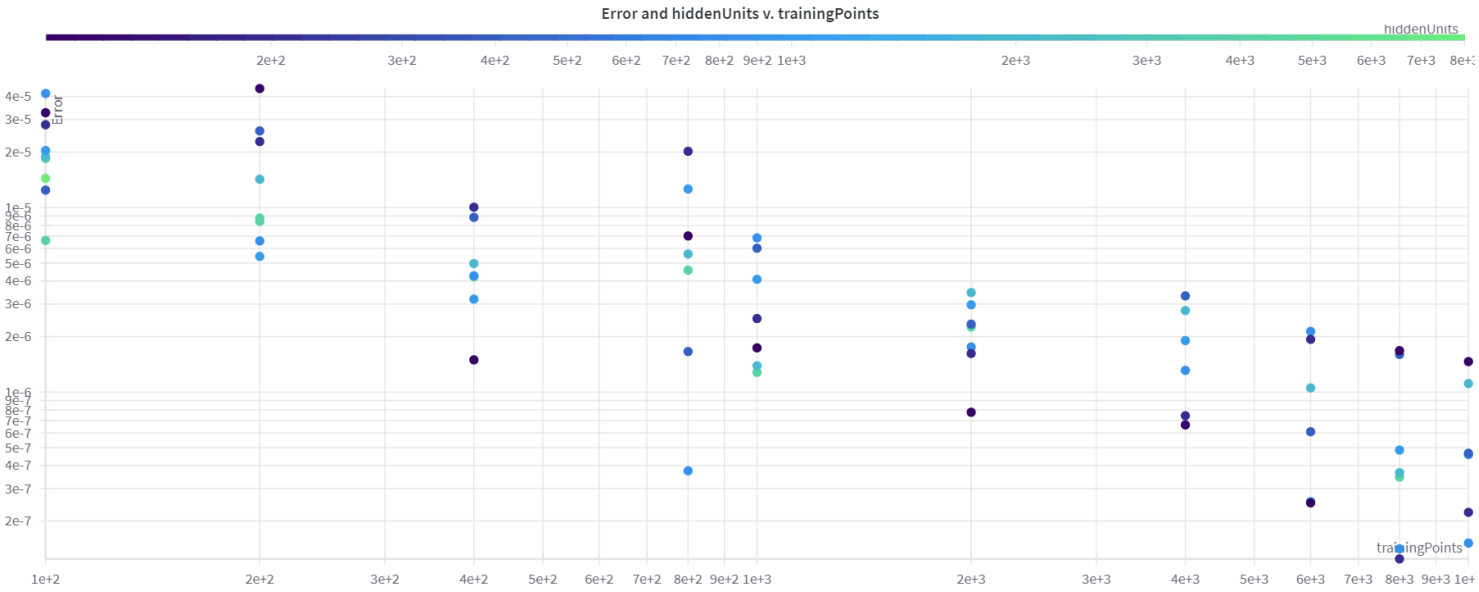


Fig1: This scatter plot represent training points in x-axis and Error in y-axis and no. of hidden units as z-axis(color bar)

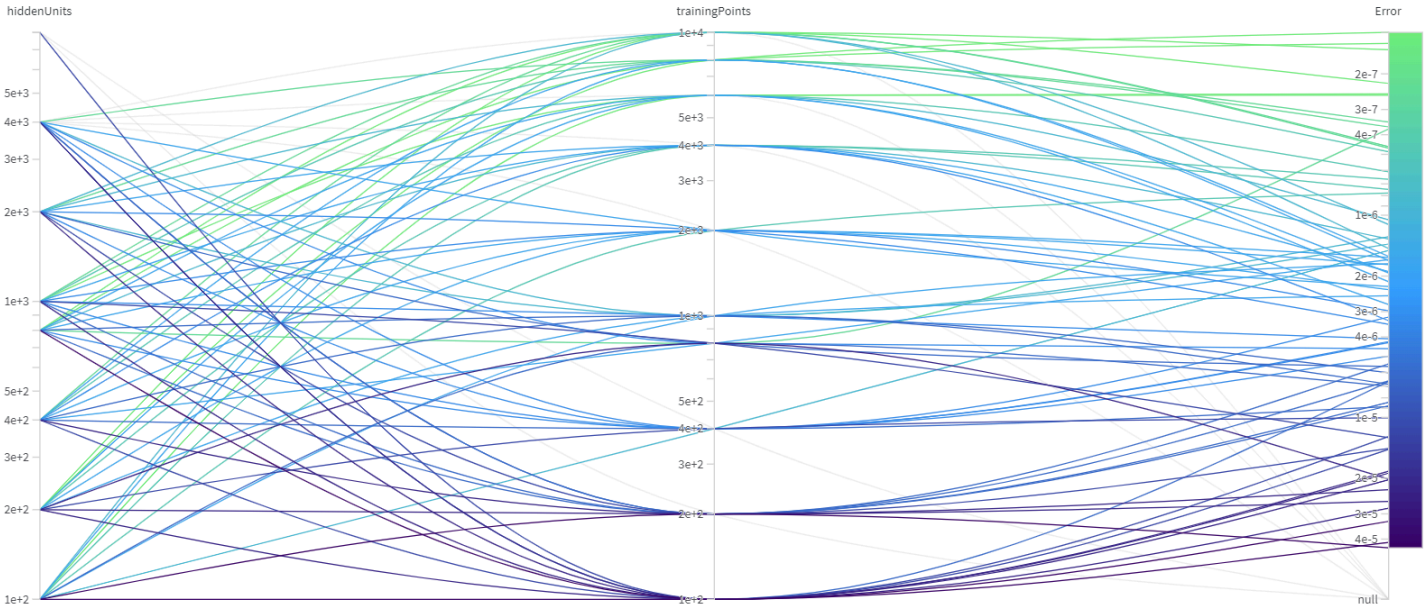


Fig 2: This Parallel coordinate plots represents the error wrt hidden units and training points.

This clearly shows that increasing training points decreases the error. However, the variance is very high and unpredictable and since the computational cost is very high, we can't afford to run each 70 experiments for 50-100 runs to make it less randomized, and we still can’t say anything about hidden units. Here's why.

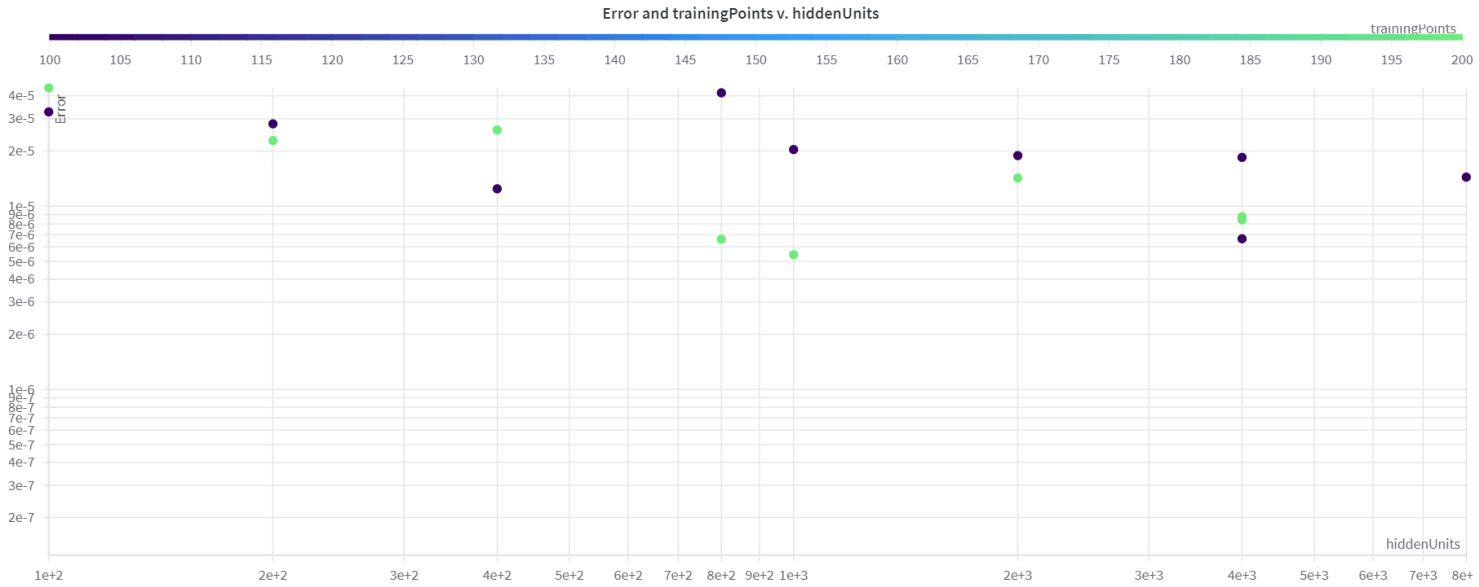


Fig 3: x-axis represents the hidden units and y-axis represent Error, Blue dots represents 100 training points and Green dots represents 200 training points.

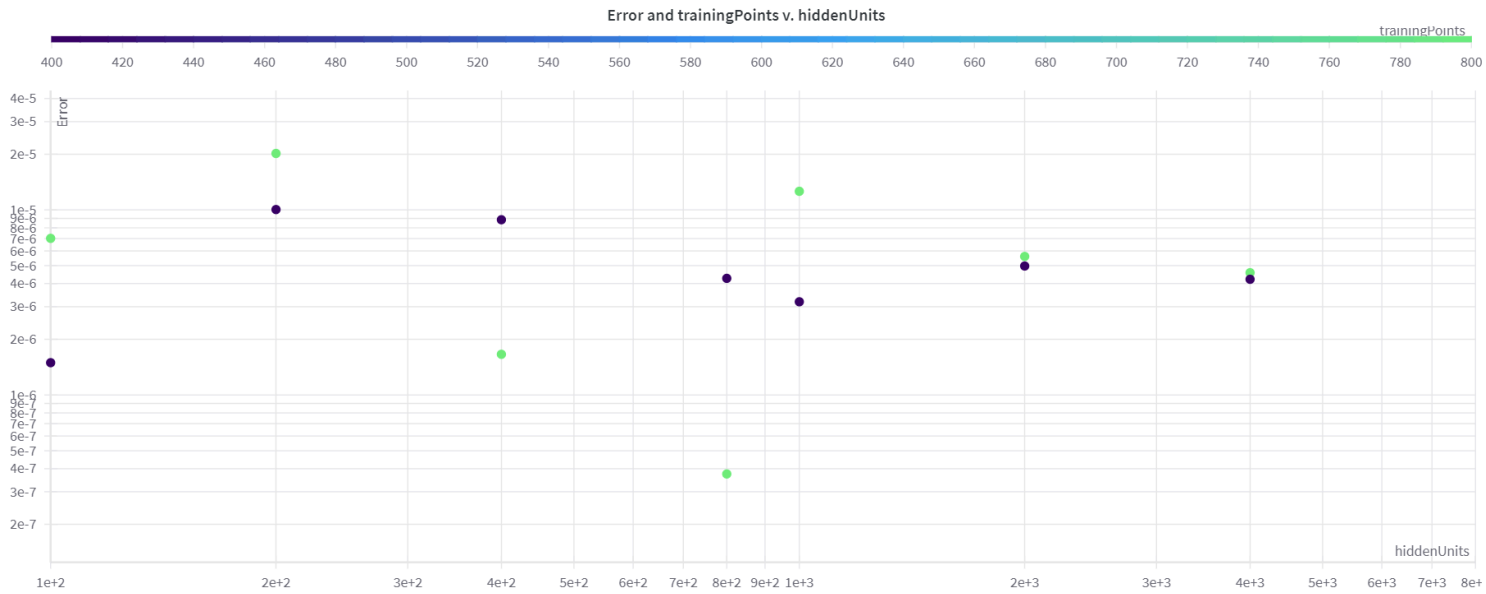


Fig 4: x-axis represents the hidden units and y-axis represent Error, Blue dots represents 400 training points and Green dots represents 800 training points.

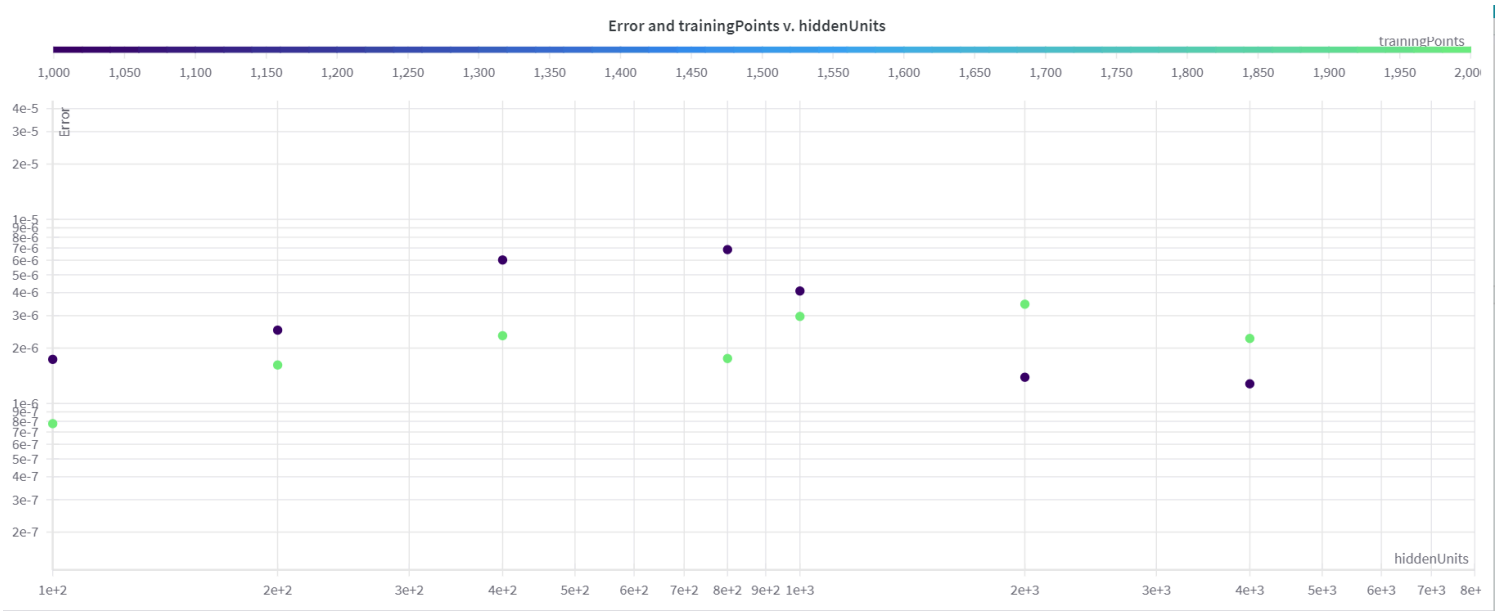


Fig 5: x-axis represents the hidden units and y-axis represent Error, Blue dots represents 1000 training points and Green dots represents 2000 training points.

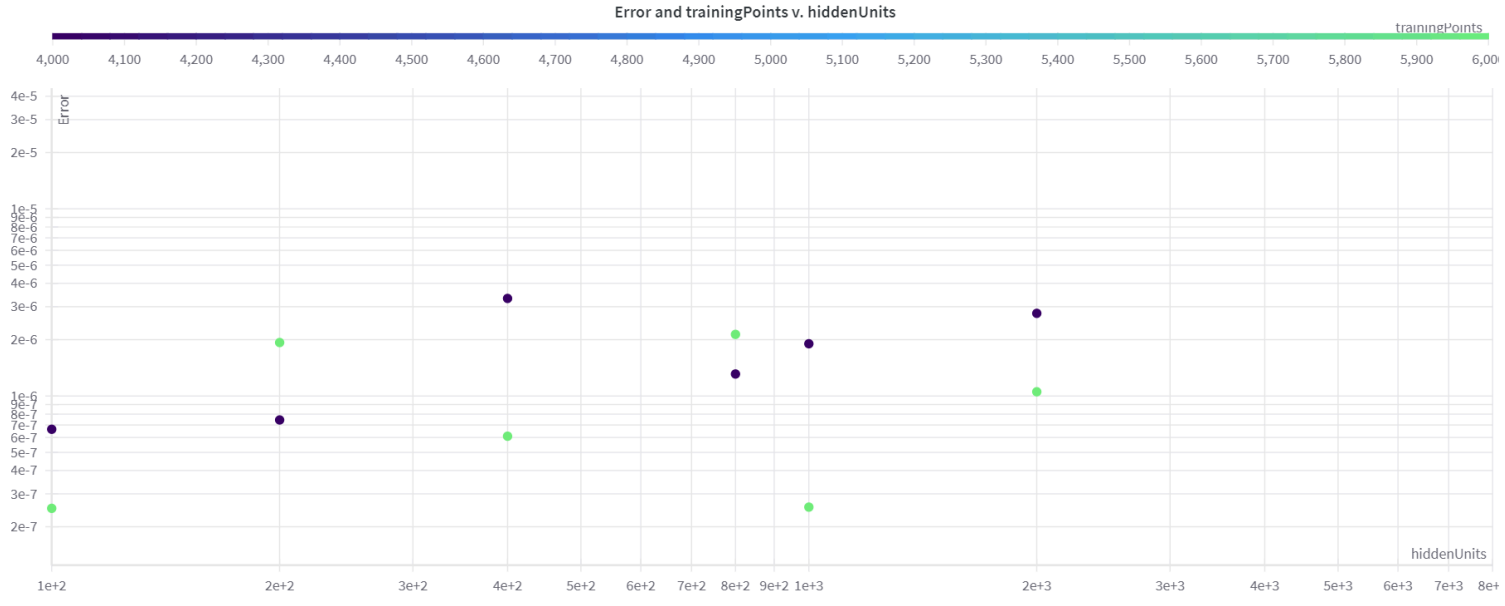


Fig 6: x-axis represents the hidden units and y-axis represent Error, Blue dots represents 4000 training points and Green dots represents 6000 training points.

Fig 7: x-axis represents the hidden units and y-axis represent Error, Blue dots represents 8000 training points and Green dots represents 10,000 training points.

Summary: Here, It's not very clear whether for a particular training point, increasing the hidden units increases or decreases the error. The possible reason could be the ***high variance***(due to random initialization), we saw in Fig1 and Fig2.

About the variance

Now, we fixed the training point to 100, and increased the hidden units from (100-2000) and each experiment ran for 100 runs except 2k training point, it ran for 50 runs, after that we calculated the mean error, and its standard deviation.

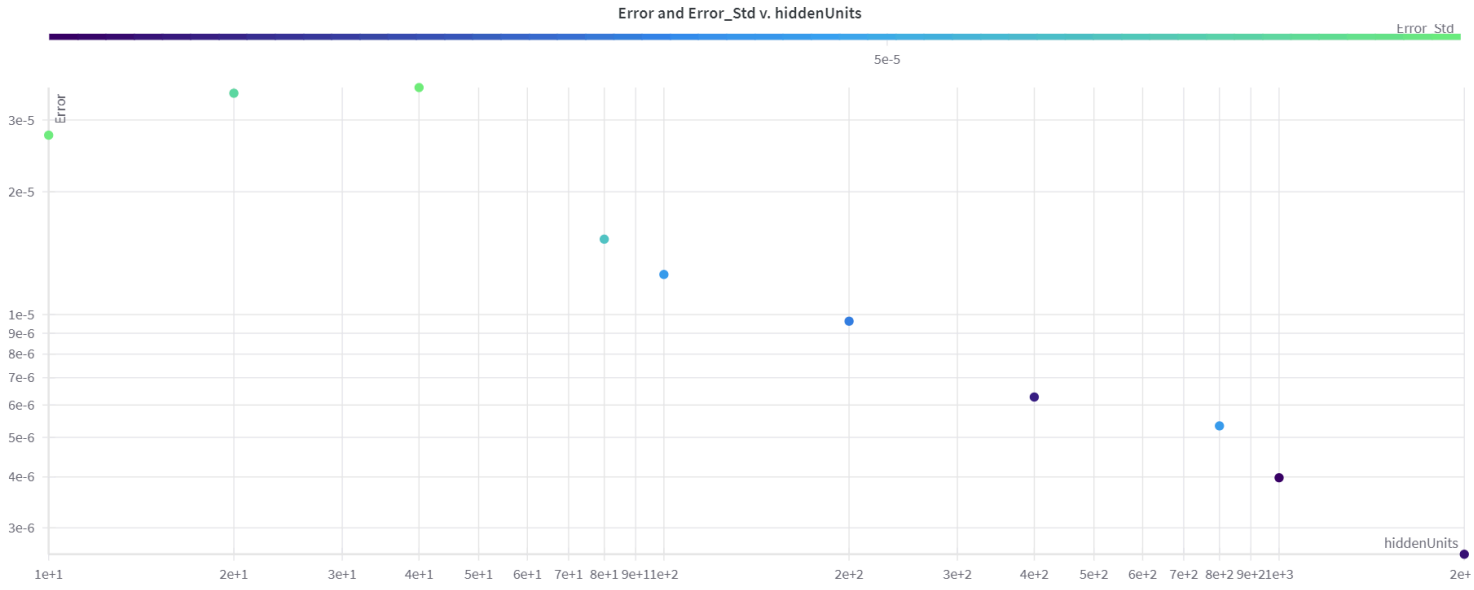
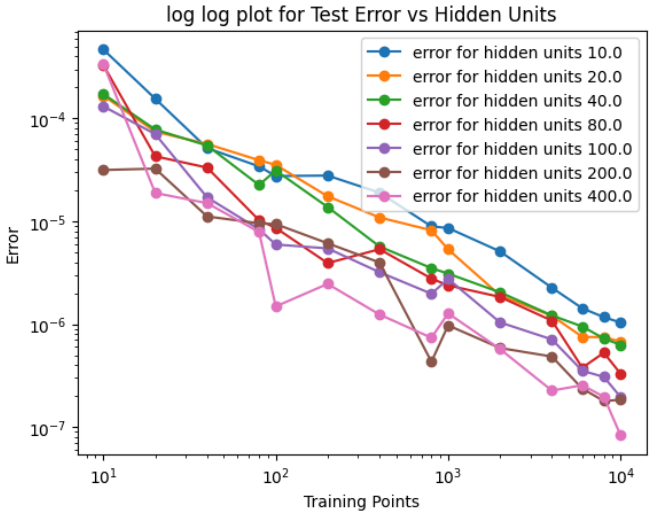
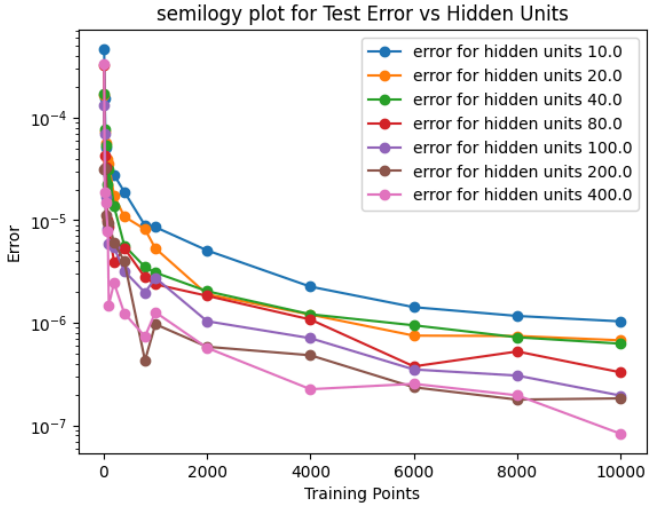


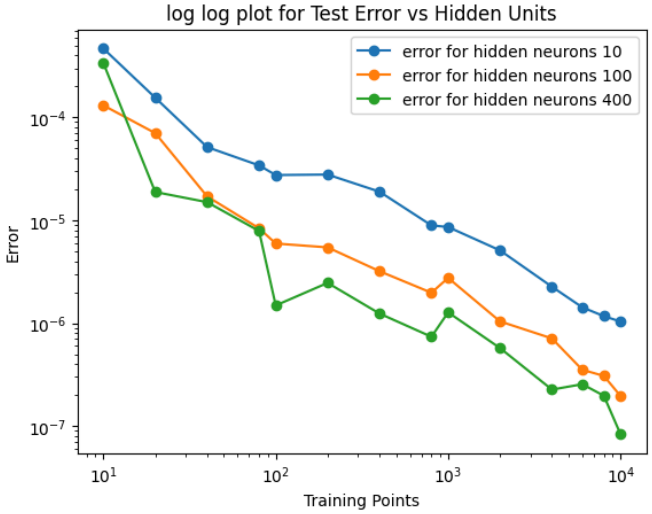
Fig8: x-axis represents the hidden units and y-axis represents Training Error. And the z-axis represents standard deviation.

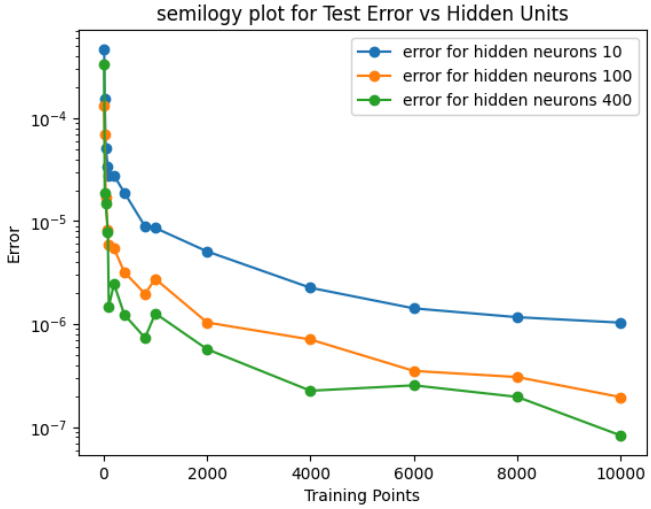
Fig9: x-axis represents the hidden units and y-axis represents Test Error. And the z-axis represents standard deviation.

Summary: We can see that both the training error and test error decreases with increasing hidden units and so the standard deviation.


log log plot for Test Error vs Hidden Units
error for hidden units 10.0
error for hidden units 20.0
error for hidden units 40.0
error for hidden units 80.0
error for hidden units 100.0
error for hidden units 200.0
error for hidden units 400.0
Error
Training Points


semilogy plot for Test Error vs Hidden Units
error for hidden units 10.0
error for hidden units 20.0
error for hidden units 40.0
error for hidden units 80.0
error for hidden units 100.0
error for hidden units 200.0
error for hidden units 400.0
Error
Training Points


log log plot for Test Error vs Hidden Units
error for hidden neurons 10
error for hidden neurons 100
error for hidden neurons 400
Error
Training Points


semilogy plot for Test Error vs Hidden Units
error for hidden neurons 10
error for hidden neurons 100
error for hidden neurons 400
Error
Training Points

6. Comparison b/w BFGS and nBFGS

BFGS or nBFGS

The above question is solved using ANNs and to train the ANNs we need to minimize the Error function and backpropagate to update the weights. Usually we use gradient descent algorithm or its variant to minimize error in Neural Networks with different activation functions but for sigmoid activation function BFGS works better. Now in this process we need to calculate the derivative of the error function wrt to weights and training points i.e Gradients. There could be two different ways to do it:

1. Analytic solution
2. Approximate solution

Analytic solution

BFGS needs a function in argument that calculates the derivative of the error function. And for different questions we have to define different gradient functions, which can be a tedious task. But it promises high accuracy with less runtime and all the above plots were generated using the BFGS method.

Approximate solution

nBFGS does not require any gradients in order to minimize the error, it approximates the gradients. It shows almost the same result for small no. of training points and hidden units but when we increase the no. of training points and hidden units the accuracy decreases with exponentially increased runtime, and does not follow the hypothesis we discussed above. The possible reasons could be the error propagated during the approximation of the gradients.

Here the is the small comparisons for reference:

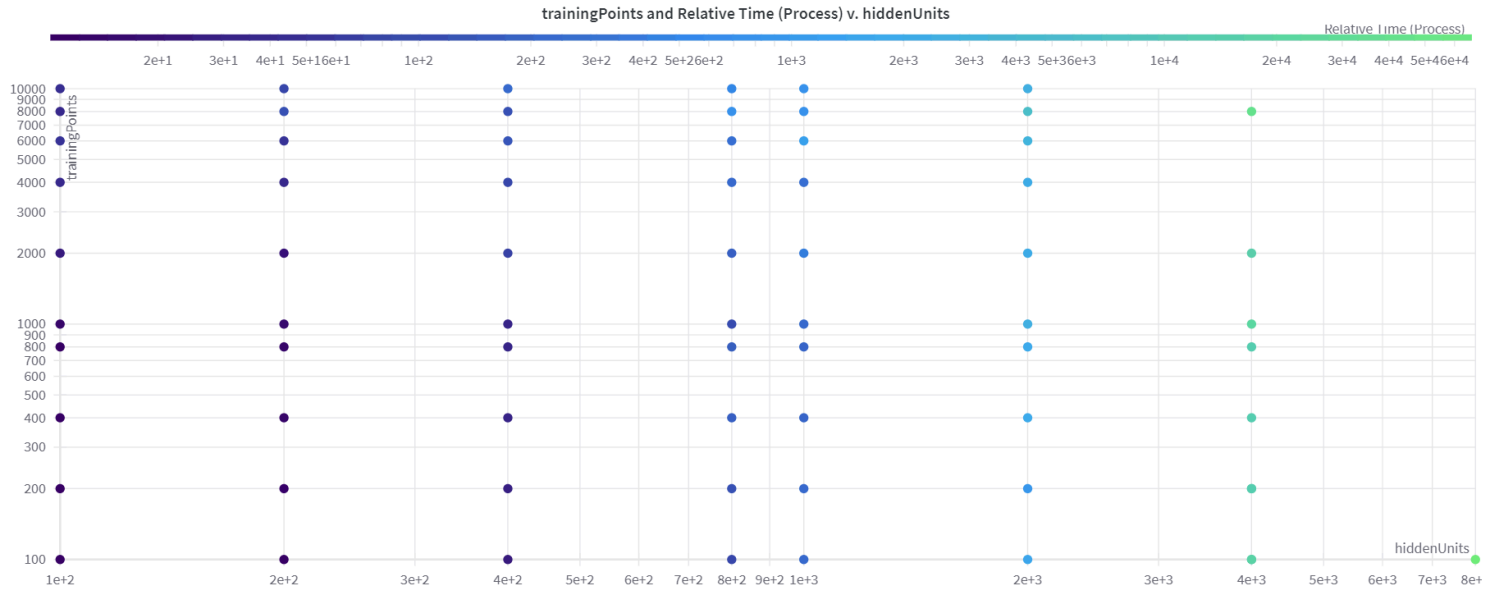


Fig10: scatter plot to represent runtime for BFGS method, Here x-axis represents Hidden units, y-axis represents training points and z-axis represents runtime(color bar).

Fig11: scatter plot to represent runtime for nBFGS method, Here x-axis represents Hidden units, y-axis represents training points and z-axis represents runtime(color bar).

Here the nBFGS method took 13h 12m 23sec for 800 training points and 4000 hidden units whereas the BFGS method took only 3m 27s with better accuracy.