Week 2 Programming assignment

1. Use a neural network to approximate the Runge function

$$f(x) = \frac{1}{1 + 25x^2}, \quad x \in [-1, 1].$$

Write a short report (1–2 pages) explaining method, results, and discussion including

- Plot the true function and the neural network prediction together.
- Show the training/validation loss curves.
- Compute and report errors (MSE or max error).

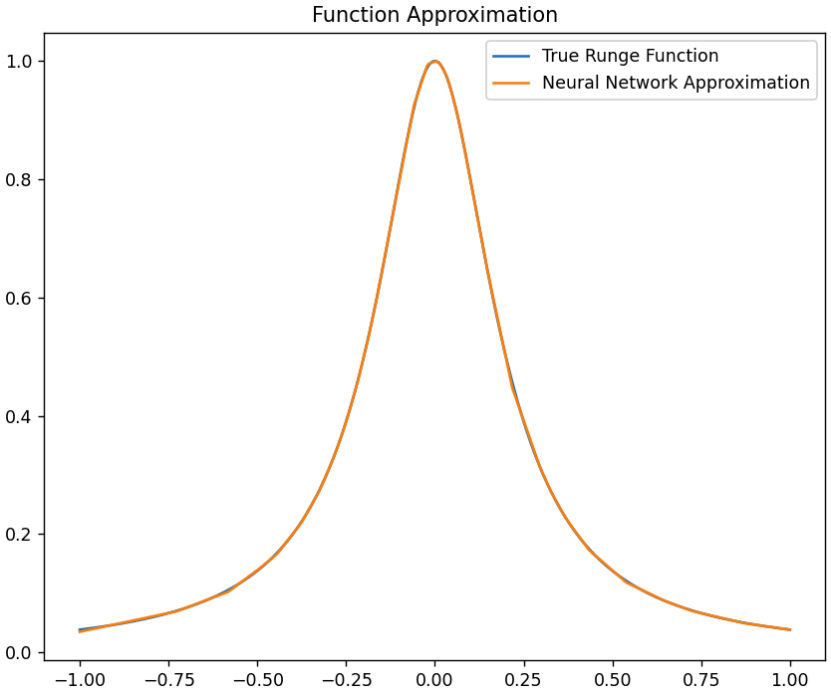
1. Data Generation
   I sampled 500 points evenly between [-1, 1].

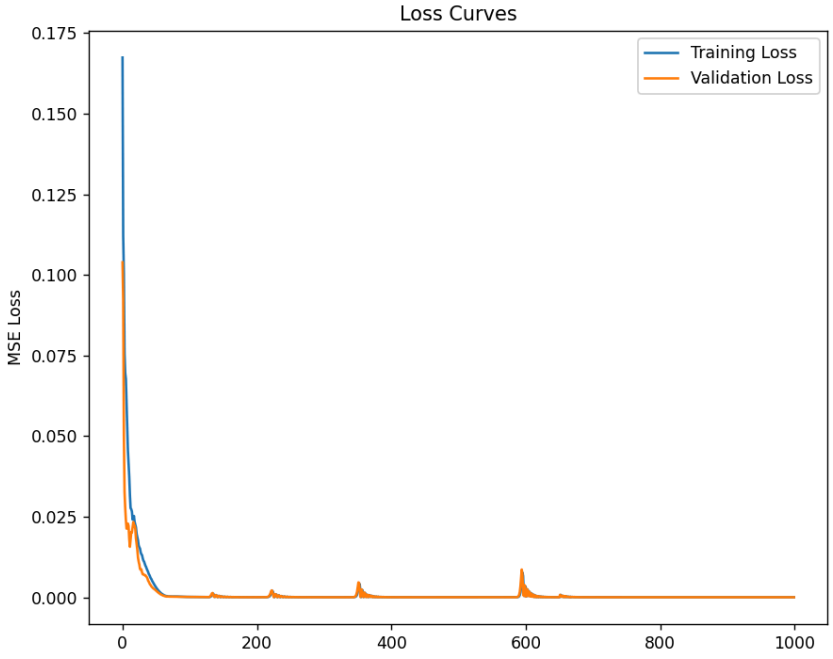
2. Neural Network
   I use a neural network with an input layer, 64 neurons on 2 hidden layers, and an output layer.

Raw Code:

```
import numpy as np
import matplotlib.pyplot as plot
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, max_error
import torch
import torch.nn as n
import torch.optim as opt

def rungeFunction(x):
    return 1 / (1 + 25 * x**2)

x = np.linspace(-1, 1, 500)
y = rungeFunction(x)

xTrain, xVal, yTrain, yVal = train_test_split(x, y, test_size=0.2, random_state=42)
xTrainTensor = torch.tensor(xTrain, dtype=torch.float32).view(-1, 1)
yTrainTensor = torch.tensor(yTrain, dtype=torch.float32).view(-1, 1)
xValueTensor = torch.tensor(xVal, dtype=torch.float32).view(-1, 1)
yValueTensor = torch.tensor(yVal, dtype=torch.float32).view(-1, 1)

class Net(n.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = n.Linear(1, 64)
```

```
        self.fc2 = n.Linear(64, 64)
        self.fc3 = n.Linear(64, 1)
        self.relu = n.ReLU()

    def forward(self, x):
        x = self.relu(self.fc1(x))
        x = self.relu(self.fc2(x))
        return self.fc3(x)

model = Net()
criterion = n.MSELoss()
optimizer = opt.Adam(model.parameters(), lr=0.01)

numEpochs = 1000
trainLosses = []
valLosses = []

for epoch in range(numEpochs):
    model.train()
    optimizer.zero_grad()
    outputs = model(xTrainTensor)
    loss = criterion(outputs, yTrainTensor)
    loss.backward()
    optimizer.step()
    trainLosses.append(loss.item())

    model.eval()
    with torch.no_grad():
        valOutputs = model(xValueTensor)
        valLoss = criterion(valOutputs, yValueTensor)
        valLosses.append(valLoss.item())


xPlot = torch.tensor(x, dtype=torch.float32).view(-1, 1)
with torch.no_grad():
    yPred = model(xPlot).numpy()

mse = mean_squared_error(y, yPred.flatten())
maxError = max_error(y, yPred.flatten())

plot.figure(figsize=(14, 6))

plot.subplot(1, 2, 1)
plot.plot(x, y, label='True Runge Function')
plot.plot(x, yPred, label='Neural Network Approximation')
plot.title('Function Approximation')
plot.legend()

plot.subplot(1, 2, 2)
plot.plot(trainLosses, label='Training Loss')
plot.plot(valLosses, label='Validation Loss')
plot.title('Loss Curves')
plot.xlabel('Epoch')
plot.ylabel('MSE Loss')
plot.legend()

plot.tight_layout()
plot.show()

print("MSE:", mse)
print("Max Error:", maxError)
```





MSE: 1.7442729295259521e-06
Max Error: 0.009409478846341635