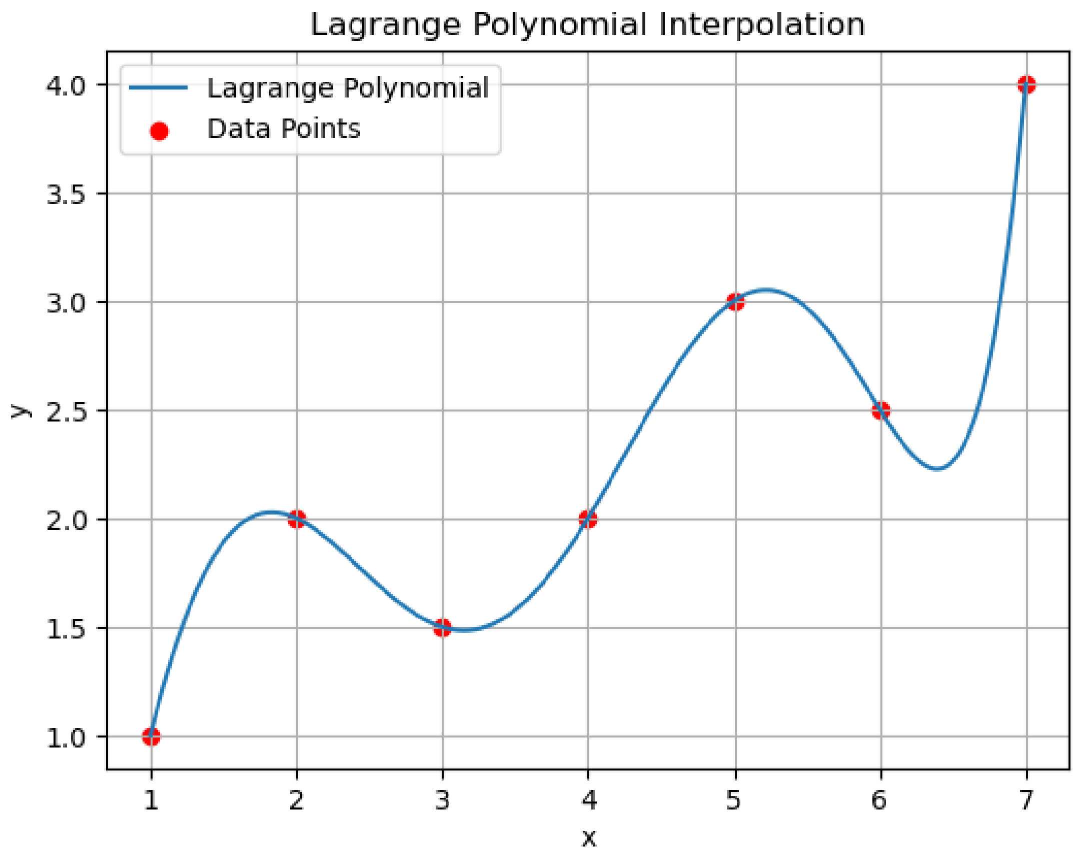
```python
# Import dependencies
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import lagrange
```

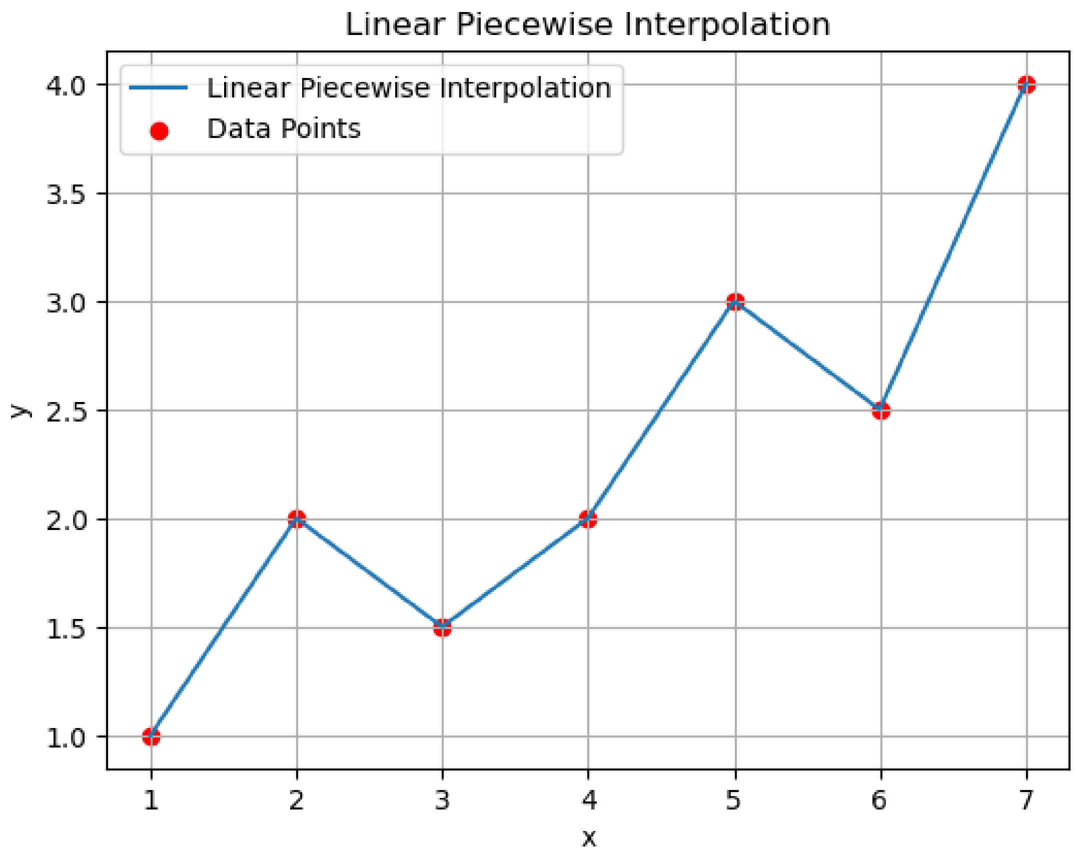
Problem 1.1

```python
# Define x and y
x = np.array([1, 2, 3, 4, 5, 6, 7])
y = np.array([1, 2, 1.5, 2, 3, 2.5, 4])

# Perform Lagrange interpolation
polynomial = lagrange(x, y)

# Generate x-axis points for plotting the interpolated polynomial
x_plot = np.linspace(1, 7, 300)
y_plot = polynomial(x_plot)

# Plot the interpolated polynomial and data points
plt.plot(x_plot, y_plot, label="Lagrange Polynomial")
plt.scatter(x, y, color='red', label="Data Points")
plt.xlabel('x')
plt.ylabel('y')
plt.title('Lagrange Polynomial Interpolation')
plt.legend()
plt.grid(True)
plt.show()
```



Problem 1.2

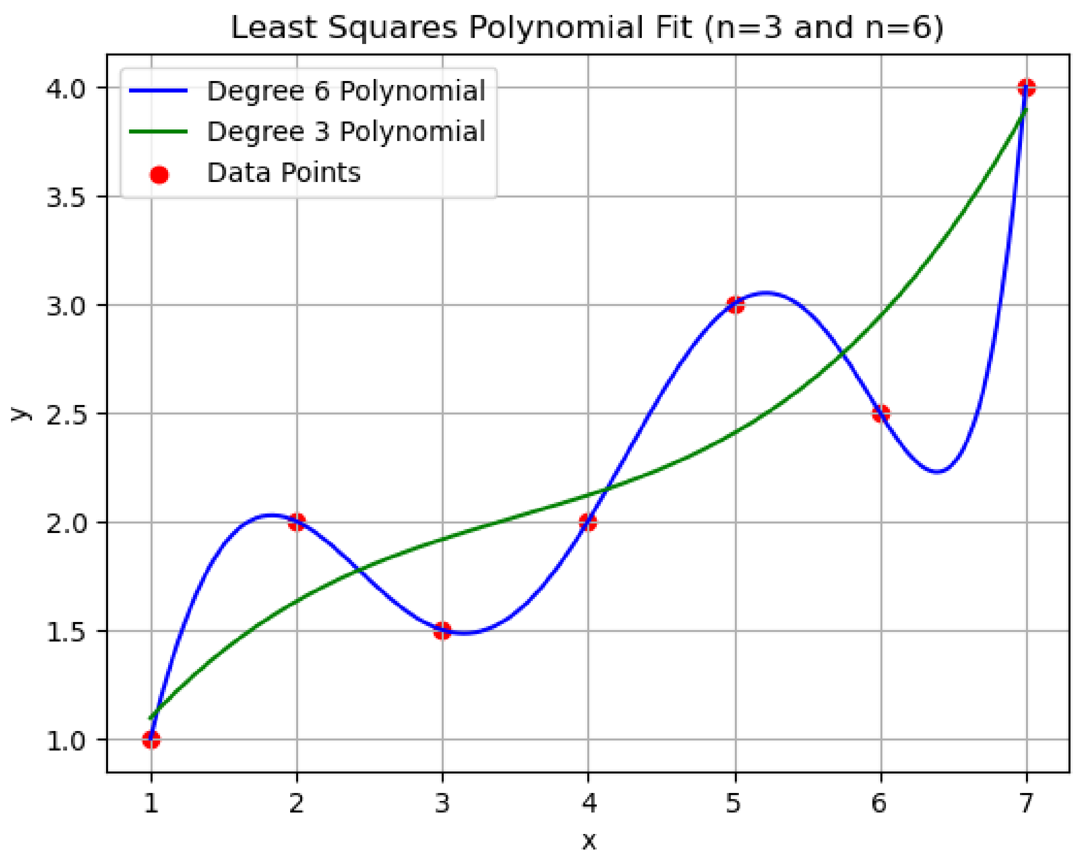
```python
# Define x and y values
x = np.array([1, 2, 3, 4, 5, 6, 7])
y = np.array([1, 2, 1.5, 2, 3, 2.5, 4])

# Perform linear piecewise interpolation
x_plot = np.linspace(1, 7, 300)
y_plot = np.interp(x_plot, x, y)

# Plot the linear interpolation and data points
plt.plot(x_plot, y_plot, label="Linear Piecewise Interpolation")
plt.scatter(x, y, color='red', label="Data Points")
plt.xlabel('x')
plt.ylabel('y')
plt.title('Linear Piecewise Interpolation')
plt.legend()
plt.grid(True)
plt.show()
```



Problem 1.3

```
# Define x and y values
x = np.array([1, 2, 3, 4, 5, 6, 7])
y = np.array([1, 2, 1.5, 2, 3, 2.5, 4])

# Fit a polynomial of degree 6 (n=6)
coeffs_n6 = np.polyfit(x, y, 6)
poly_n6 = np.poly1d(coeffs_n6)

# Fit a polynomial of degree 3 (n=3)
coeffs_n3 = np.polyfit(x, y, 3)
poly_n3 = np.poly1d(coeffs_n3)

# Generate x-axis points for plotting
x_plot = np.linspace(1, 7, 300)
y_n6 = poly_n6(x_plot)
y_n3 = poly_n3(x_plot)

# Plot both degree 6 and degree 3 polynomials along with data points
plt.plot(x_plot, y_n6, label="Degree 6 Polynomial", color='blue')
plt.plot(x_plot, y_n3, label="Degree 3 Polynomial", color='green')
plt.scatter(x, y, color='red', label="Data Points")
plt.xlabel('x')
```

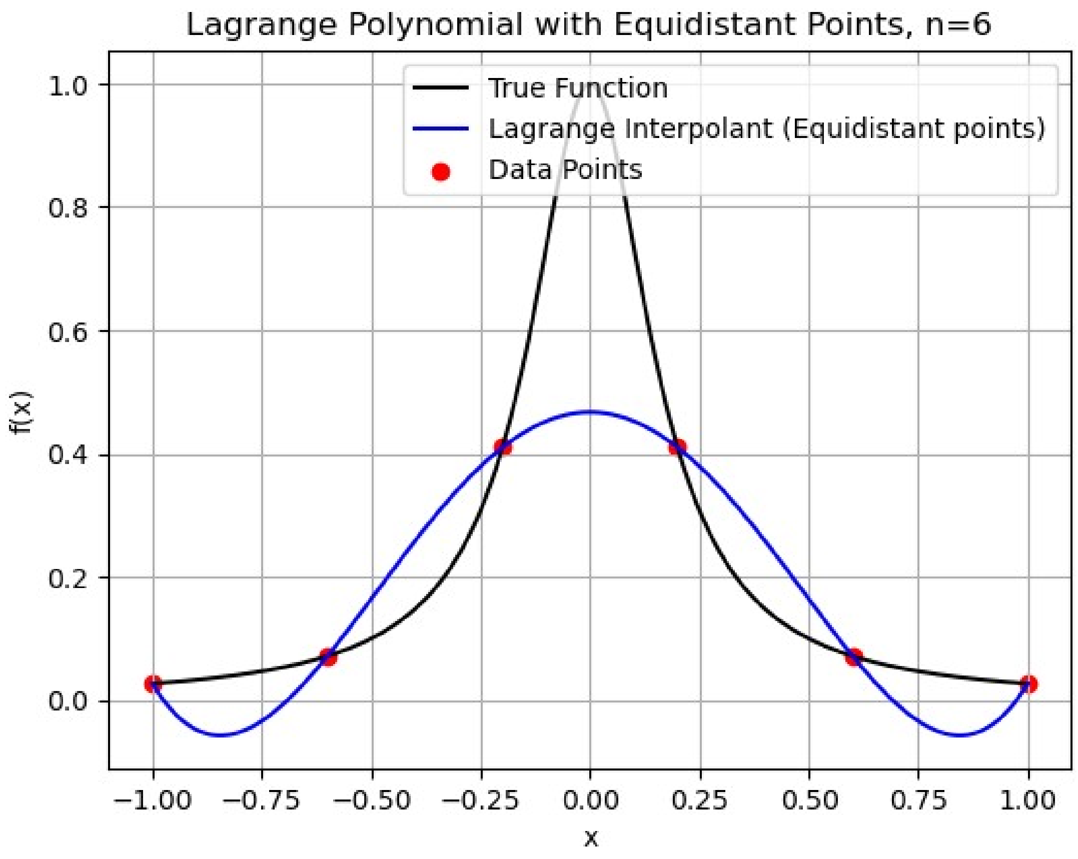
```python
plt.ylabel('y')
plt.title('Least Squares Polynomial Fit (n=3 and n=6)')
plt.legend()
plt.grid(True)
plt.show()
```



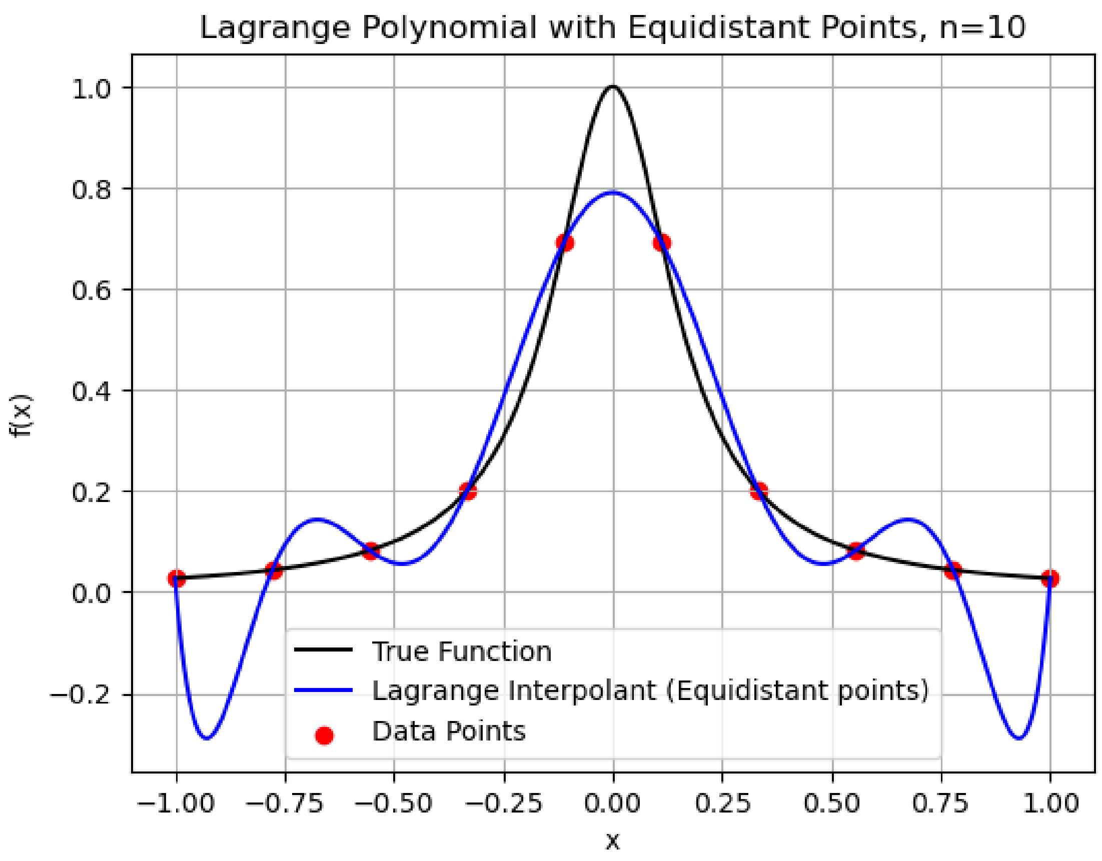
```python
# Import dependencies
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import lagrange
```

Problem 2.1.a

```python
# Define the function f(x)
def f(x):
    return 1 / (1 + 36 * x**2)

# Generate equidistant points
n = 6  # n can be adjusted
x_equidistant = np.linspace(-1, 1, n)
y_equidistant = f(x_equidistant)

# Perform Lagrange interpolation
polynomial_equidistant = lagrange(x_equidistant, y_equidistant)

# Generate points for plotting
x_plot = np.linspace(-1, 1, 300)
y_plot = polynomial_equidistant(x_plot)

# Plot the true function and the Lagrange interpolant
plt.plot(x_plot, f(x_plot), label="True Function", color='black')
plt.plot(x_plot, y_plot, label="Lagrange Interpolant (Equidistant
points)", color='blue')
plt.scatter(x_equidistant, y_equidistant, color='red', label="Data
Points")
plt.xlabel('x')
plt.ylabel('f(x)')
plt.legend()
plt.title('Lagrange Polynomial with Equidistant Points, n=6')
plt.grid(True)
plt.show()
```

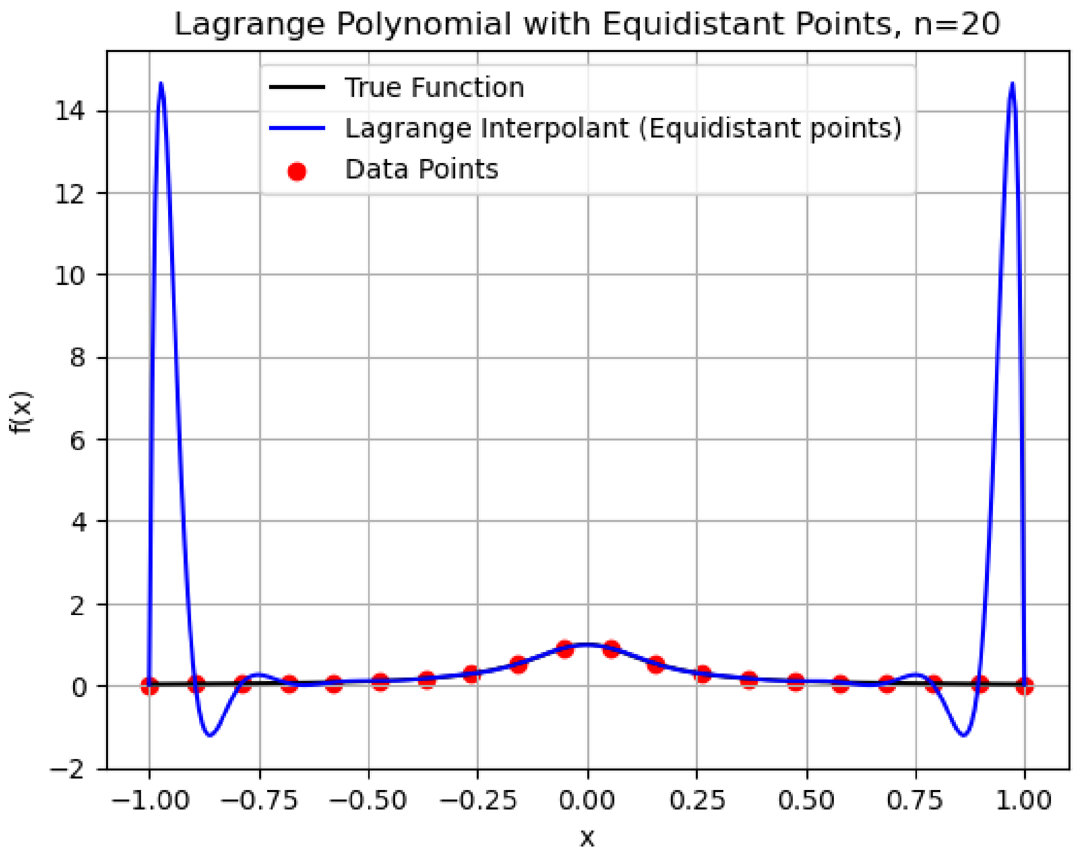


```python
# Define the function f(x)
def f(x):
    return 1 / (1 + 36 * x**2)

# Generate equidistant points
n = 10  # n can be adjusted
x_equidistant = np.linspace(-1, 1, n)
y_equidistant = f(x_equidistant)

# Perform Lagrange interpolation
polynomial_equidistant = lagrange(x_equidistant, y_equidistant)

# Generate points for plotting
x_plot = np.linspace(-1, 1, 300)
y_plot = polynomial_equidistant(x_plot)

# Plot the true function and the Lagrange interpolant
plt.plot(x_plot, f(x_plot), label="True Function", color='black')
plt.plot(x_plot, y_plot, label="Lagrange Interpolant (Equidistant
points)", color='blue')
plt.scatter(x_equidistant, y_equidistant, color='red', label="Data
Points")
plt.xlabel('x')
plt.ylabel('f(x)')
plt.legend()
plt.title('Lagrange Polynomial with Equidistant Points, n=10')
plt.grid(True)
plt.show()
```



```
# Define the function f(x)
def f(x):
    return 1 / (1 + 36 * x**2)

# Generate equidistant points
n = 20  # n can be adjusted
x_equidistant = np.linspace(-1, 1, n)
y_equidistant = f(x_equidistant)

# Perform Lagrange interpolation
polynomial_equidistant = lagrange(x_equidistant, y_equidistant)

# Generate points for plotting
x_plot = np.linspace(-1, 1, 300)
y_plot = polynomial_equidistant(x_plot)

# Plot the true function and the Lagrange interpolant
```

```
plt.plot(x_plot, f(x_plot), label="True Function", color='black')
plt.plot(x_plot, y_plot, label="Lagrange Interpolant (Equidistant
points)", color='blue')
plt.scatter(x_equidistant, y_equidistant, color='red', label="Data
Points")
plt.xlabel('x')
plt.ylabel('f(x)')
plt.legend()
plt.title('Lagrange Polynomial with Equidistant Points, n=20')
plt.grid(True)
plt.show()
```



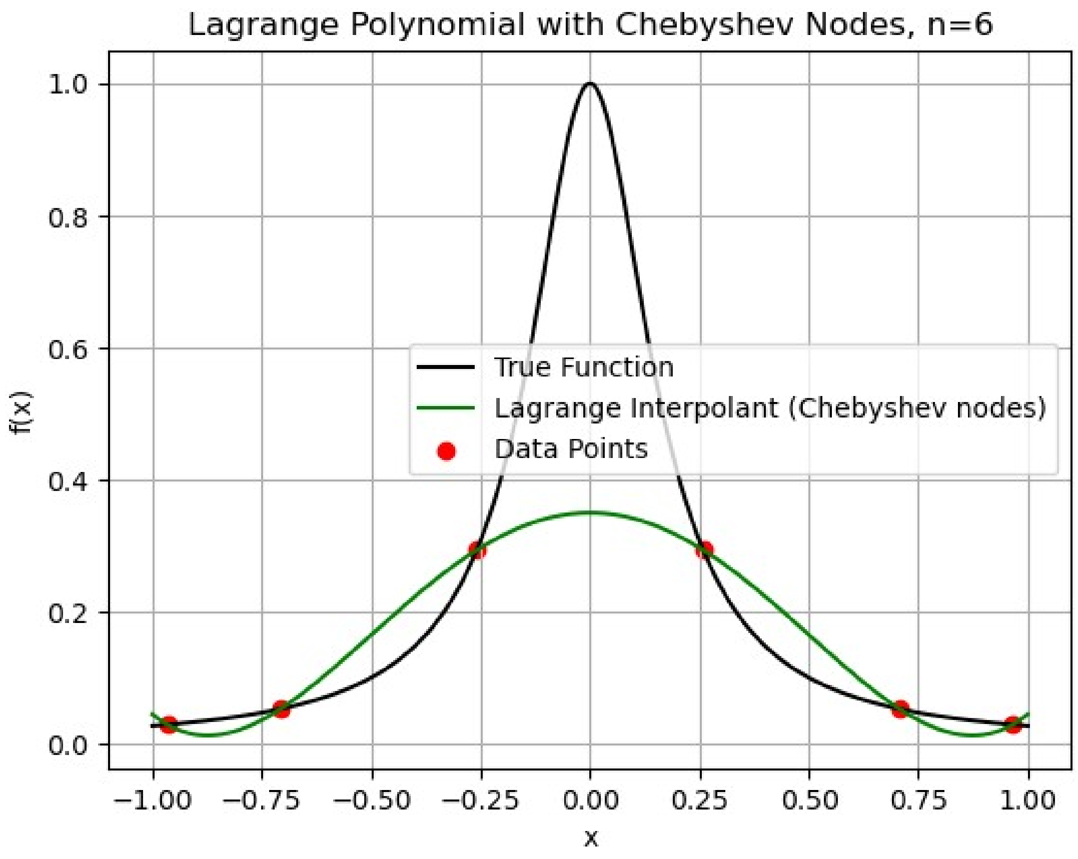
When using equidistant nodes, the polynomial interpolant can oscillate significantly near the boundaries (this is known as the Runge phenomenon), especially as n increases.

- For small n (e.g. n=6): The interpolation is less accurate compared to the true function.
- For moderate n (e.g. n=10): As n increases, the interpolant starts showing more pronounced oscillations, especially at the edges of the interval. The approximation is still relatively accurate in the middle of the interval, but near the boundaries ($x = -1$ and $x = 1$), the polynomial starts to deviate from the true function, introducing larger errors.

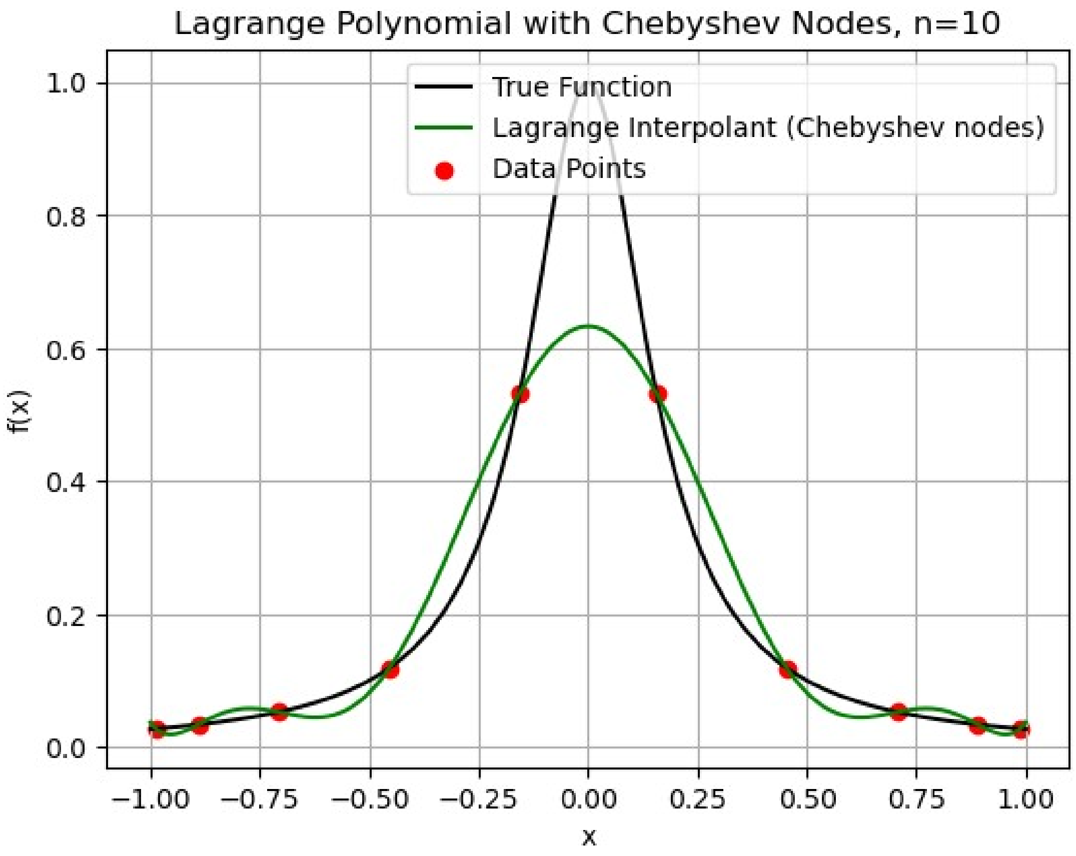
- For large n (e.g. n=20): The interpolant exhibits significant oscillations, particularly near the boundaries, which is a manifestation of Runge's phenomenon. While the interpolant passes through all the data points exactly, it no longer provides a good approximation of the true function at the edges. The peaks and valleys near $x = -1$ and $x = 1$ are highly exaggerated, indicating poor approximation in those regions, even though the fit remains accurate in the middle part of the interval.
- Runge's phenomenon is the observation that, for certain functions (especially those with rapid changes), using high-degree polynomials for interpolation with equidistant nodes can cause large oscillations, particularly at the interval's edges. This phenomenon occurs because the polynomial tries to fit all the data points exactly, and as the number of points increases, it becomes more difficult for the polynomial to balance the fit in the middle of the interval with the fit at the boundaries. As a result, the polynomial tends to oscillate more at the edges.

Problem 2.1.b

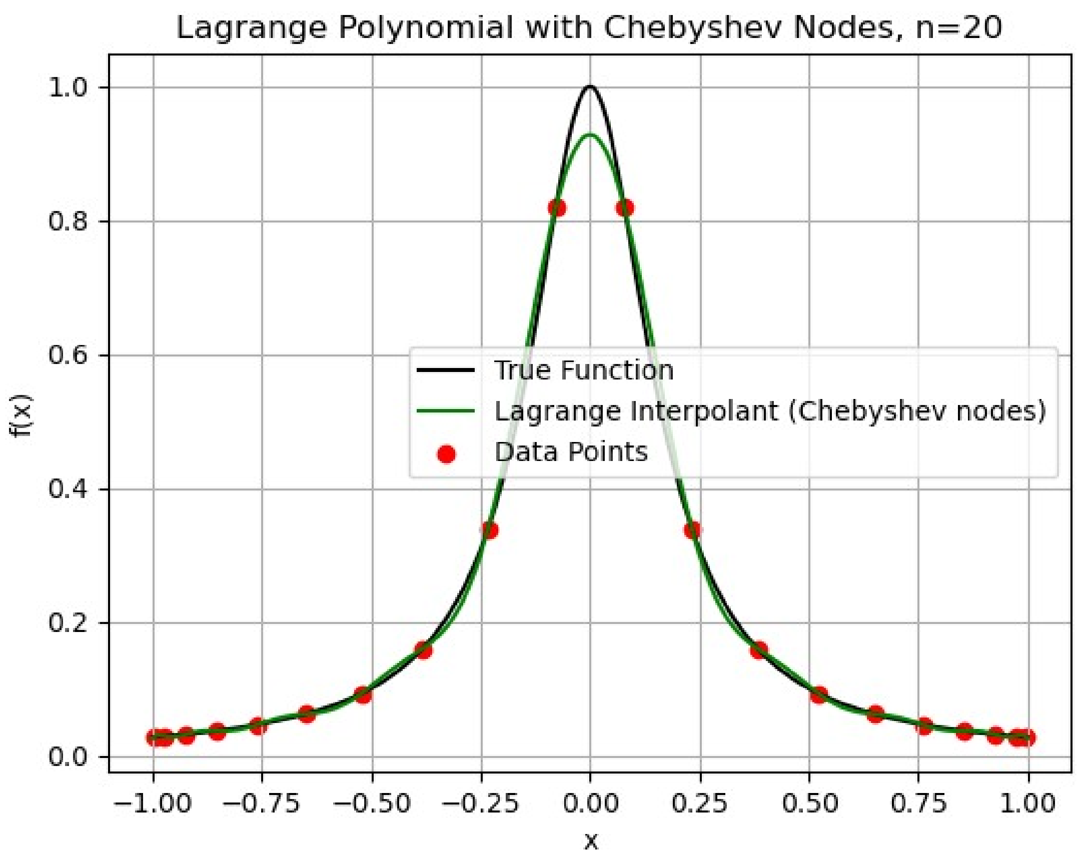
```
# Generate Chebyshev nodes
n = 6  # n can be adjusted
i = np.arange(n)
x_chebyshev = np.cos((2*i+1) * np.pi / (2*(n)))
y_chebyshev = f(x_chebyshev)

# Perform Lagrange interpolation with Chebyshev nodes
polynomial_chebyshev = lagrange(x_chebyshev, y_chebyshev)

# Generate points for plotting
y_plot_chebyshev = polynomial_chebyshev(x_plot)

# Plot the true function and the Lagrange interpolant (Chebyshev
nodes)
plt.plot(x_plot, f(x_plot), label="True Function", color='black')
plt.plot(x_plot, y_plot_chebyshev, label="Lagrange Interpolant
(Chebyshev nodes)", color='green')
plt.scatter(x_chebyshev, y_chebyshev, color='red', label="Data
Points")
plt.xlabel('x')
plt.ylabel('f(x)')
plt.legend()
plt.title('Lagrange Polynomial with Chebyshev Nodes, n=6')
plt.grid(True)
plt.show()
```



```
# Generate Chebyshev nodes
n = 10  # n can be adjusted
i = np.arange(n)
x_chebyshev = np.cos((2*i+1) * np.pi / (2*(n)))
y_chebyshev = f(x_chebyshev)

# Perform Lagrange interpolation with Chebyshev nodes
polynomial_chebyshev = lagrange(x_chebyshev, y_chebyshev)

# Generate points for plotting
y_plot_chebyshev = polynomial_chebyshev(x_plot)

# Plot the true function and the Lagrange interpolant (Chebyshev
nodes)
plt.plot(x_plot, f(x_plot), label="True Function", color='black')
plt.plot(x_plot, y_plot_chebyshev, label="Lagrange Interpolant
(Chebyshev nodes)", color='green')
plt.scatter(x_chebyshev, y_chebyshev, color='red', label="Data
Points")
plt.xlabel('x')
plt.ylabel('f(x)')
plt.legend()
plt.title('Lagrange Polynomial with Chebyshev Nodes, n=10')
plt.grid(True)
plt.show()
```



```
# Generate Chebyshev nodes
n = 20  # n can be adjusted
i = np.arange(n)
x_chebyshev = np.cos((2*i + 1) * np.pi / (2*(n)))
y_chebyshev = f(x_chebyshev)

# Perform Lagrange interpolation with Chebyshev nodes
polynomial_chebyshev = lagrange(x_chebyshev, y_chebyshev)

# Generate points for plotting
y_plot_chebyshev = polynomial_chebyshev(x_plot)

# Plot the true function and the Lagrange interpolant (Chebyshev
nodes)
plt.plot(x_plot, f(x_plot), label="True Function", color='black')
plt.plot(x_plot, y_plot_chebyshev, label="Lagrange Interpolant
(Chebyshev nodes)", color='green')
plt.scatter(x_chebyshev, y_chebyshev, color='red', label="Data
Points")
plt.xlabel('x')
plt.ylabel('f(x)')
plt.legend()
plt.title('Lagrange Polynomial with Chebyshev Nodes, n=20')
plt.grid(True)
plt.show()
```



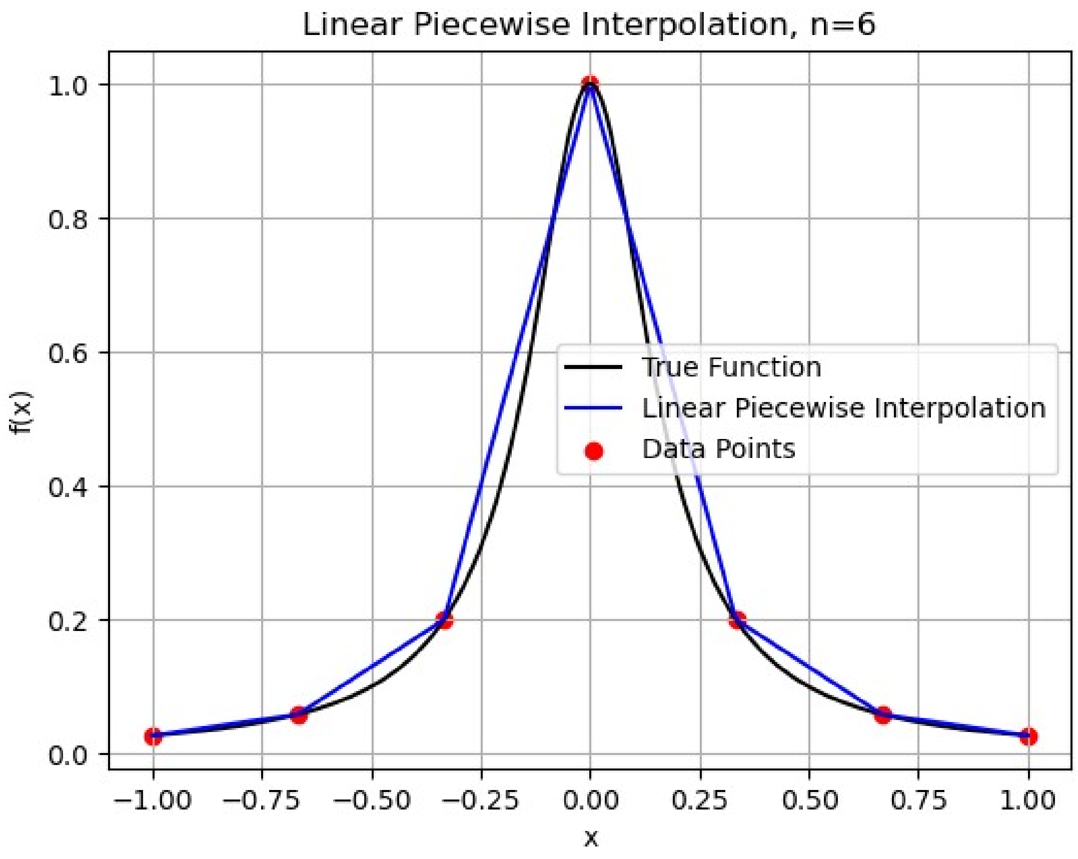
Using Chebyshev nodes significantly improves the interpolation quality, particularly as the number of nodes n increases, which helps avoid the oscillations seen in the case of equidistant nodes (Runge's phenomenon).

- For small n (e.g. n=6): The interpolant is still somewhat inaccurate, especially near the edges of the interval. The interpolant does not perfectly match the true function but performs reasonably well.
- For moderate n (e.g. n=10): The interpolation improves, with the polynomial becoming more accurate across the interval. There are still minor deviations from the true function near the boundaries, but this is much less severe than the oscillations seen with equidistant nodes.
- For large n (e.g. n=20): The interpolant closely follows the true function throughout the interval. The interpolation near the boundaries is significantly more accurate compared to equidistant nodes, with almost no severe oscillations.

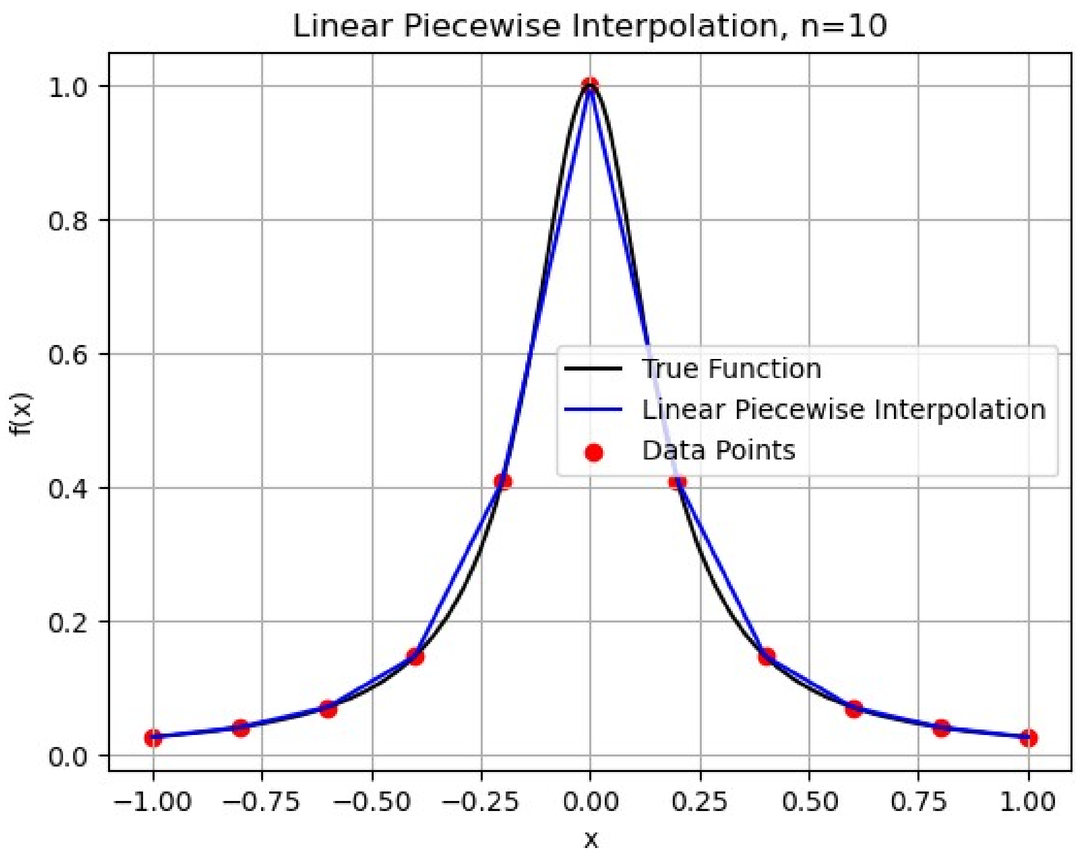
- Chebyshev nodes minimize the maximum error in polynomial interpolation according to Chebyshev's minimax theorem, which states that the interpolation error is minimized when the nodes are distributed in a Chebyshev fashion.
- Chebyshev nodes are not uniformly distributed like equidistant nodes. Instead, they are more densely spaced near the boundaries of the interval. This clustering of points near the edges reduces the error that typically arises at the boundaries in polynomial interpolation with equidistant nodes.

Problem 2.2

```
# Define the function f(x)
def f(x):
    return 1 / (1 + 36 * x**2)

# Number of points (n can be adjusted)
n = 6   # n can be adjusted

# Generate equidistant points
x_equidistant = np.linspace(-1, 1, n+1)
y_equidistant = f(x_equidistant)

# Generate x values for the plot
x_plot = np.linspace(-1, 1, 300)
y_plot = f(x_plot)

# Perform linear piecewise interpolation
y_linear = np.interp(x_plot, x_equidistant, y_equidistant)

# Plot the true function and the linear interpolation
plt.plot(x_plot, y_plot, label="True Function", color='black')
plt.plot(x_plot, y_linear, label="Linear Piecewise Interpolation", 
color='blue')
plt.scatter(x_equidistant, y_equidistant, color='red', label="Data 
Points")
plt.xlabel('x')
plt.ylabel('f(x)')
plt.legend()
plt.title('Linear Piecewise Interpolation, n=6')
plt.grid(True)
plt.show()
```

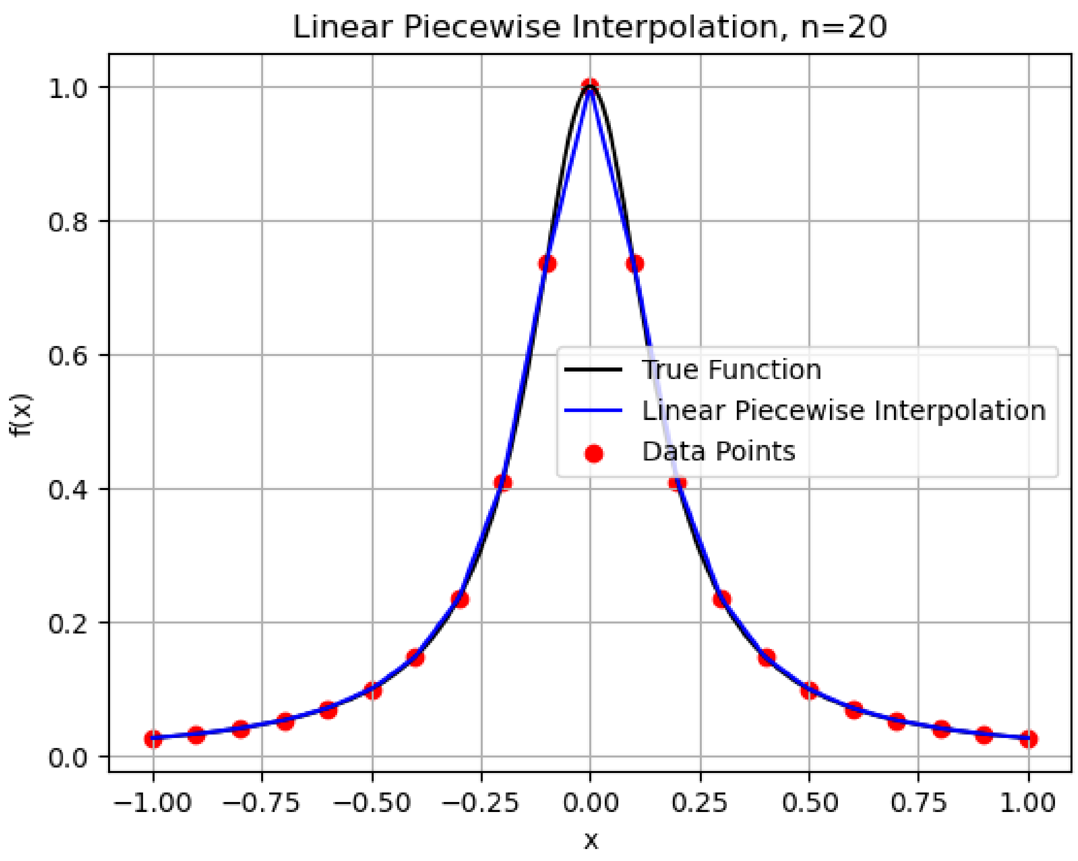


```
# Define the function f(x)
def f(x):
    return 1 / (1 + 36 * x**2)

# Number of points (n can be adjusted)
n = 10   # n can be adjusted

# Generate equidistant points
x_equidistant = np.linspace(-1, 1, n+1)
y_equidistant = f(x_equidistant)

# Generate x values for the plot
x_plot = np.linspace(-1, 1, 300)
y_plot = f(x_plot)

# Perform linear piecewise interpolation
y_linear = np.interp(x_plot, x_equidistant, y_equidistant)

# Plot the true function and the linear interpolation
plt.plot(x_plot, y_plot, label="True Function", color='black')
plt.plot(x_plot, y_linear, label="Linear Piecewise Interpolation",
color='blue')
plt.scatter(x_equidistant, y_equidistant, color='red', label="Data
Points")
plt.xlabel('x')
plt.ylabel('f(x)')
plt.legend()
plt.title('Linear Piecewise Interpolation, n=10')
plt.grid(True)
plt.show()
```



```
# Define the function f(x)
def f(x):
    return 1 / (1 + 36 * x**2)

# Number of points (n can be adjusted)
n = 20   # n can be adjusted

# Generate equidistant points
x_equidistant = np.linspace(-1, 1, n+1)
y_equidistant = f(x_equidistant)

# Generate x values for the plot
x_plot = np.linspace(-1, 1, 300)
y_plot = f(x_plot)
```

```
# Perform linear piecewise interpolation
y_linear = np.interp(x_plot, x_equidistant, y_equidistant)

# Plot the true function and the linear interpolation
plt.plot(x_plot, y_plot, label="True Function", color='black')
plt.plot(x_plot, y_linear, label="Linear Piecewise Interpolation",
color='blue')
plt.scatter(x_equidistant, y_equidistant, color='red', label="Data
Points")
plt.xlabel('x')
plt.ylabel('f(x)')
plt.legend()
plt.title('Linear Piecewise Interpolation, n=20')
plt.grid(True)
plt.show()
```



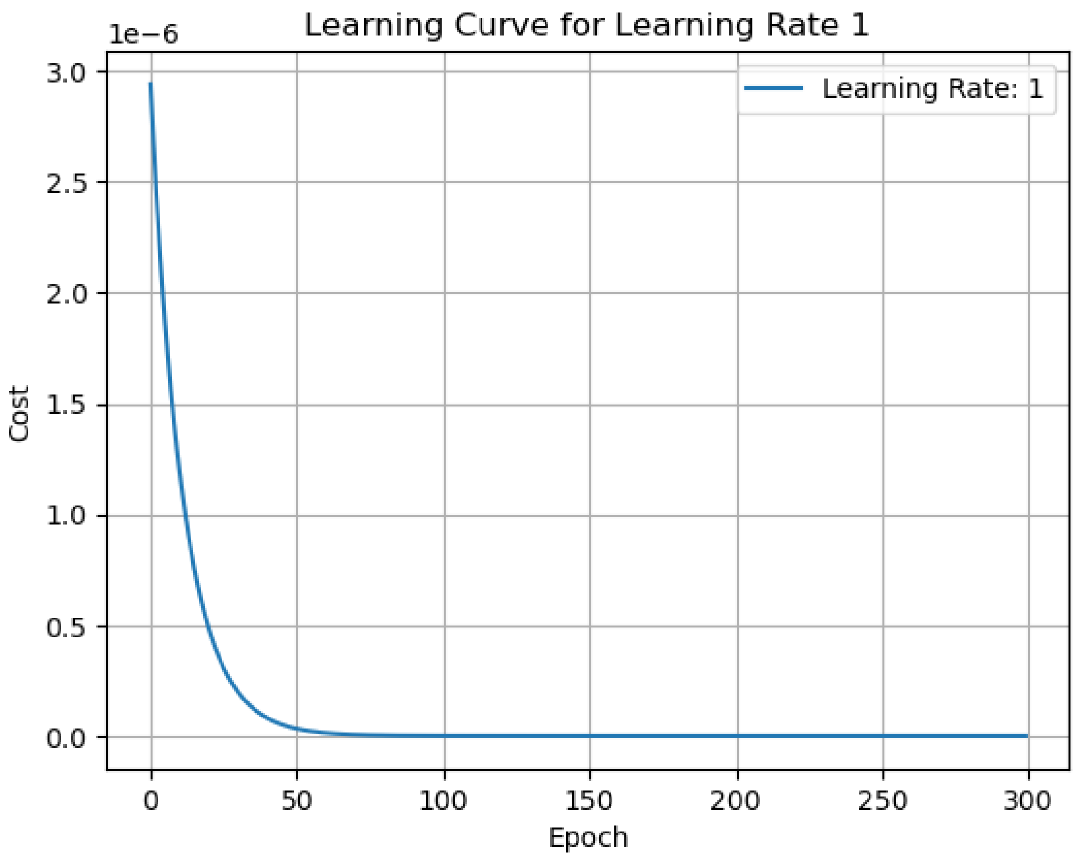
- As n increases, the linear piecewise interpolant approximates the true function better.
- Increasing n leads to shorter segments, which better approximate the local shape of the function.

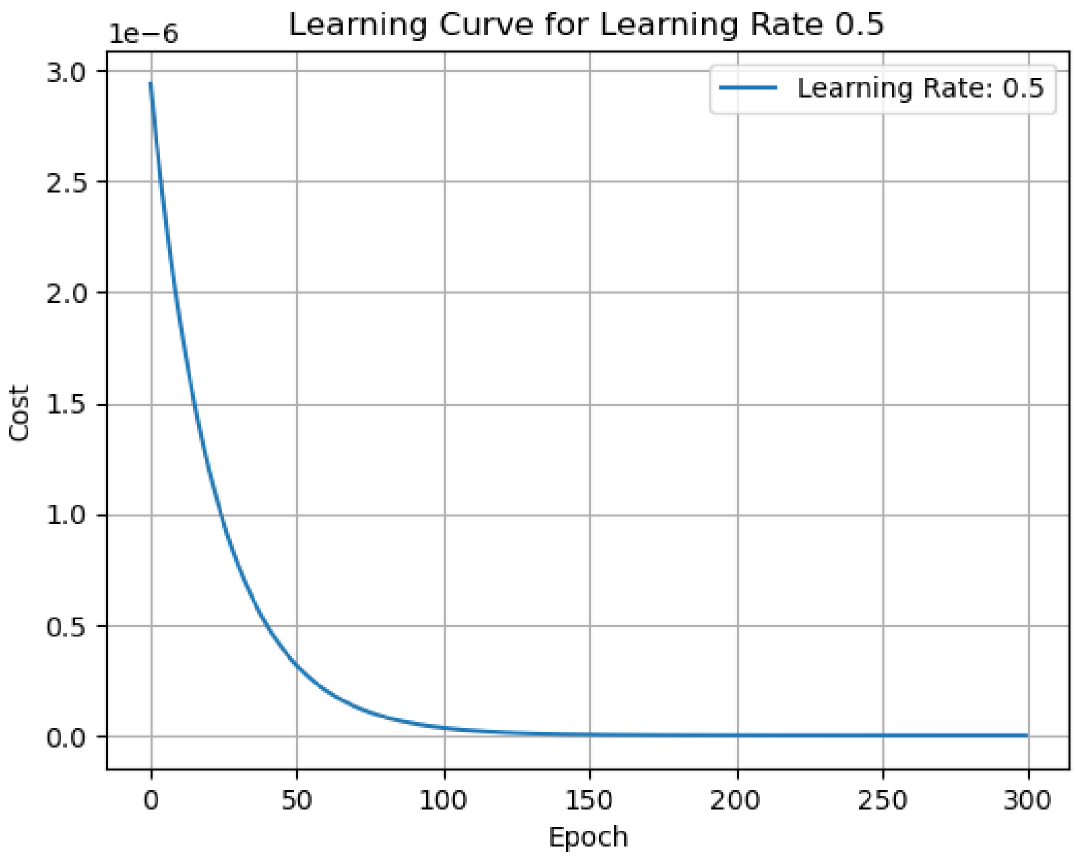
- More data points provide more information, leading to a better approximation of the true function.
- Stability is maintained, as linear interpolation avoids the oscillatory problems seen with high-degree polynomials.

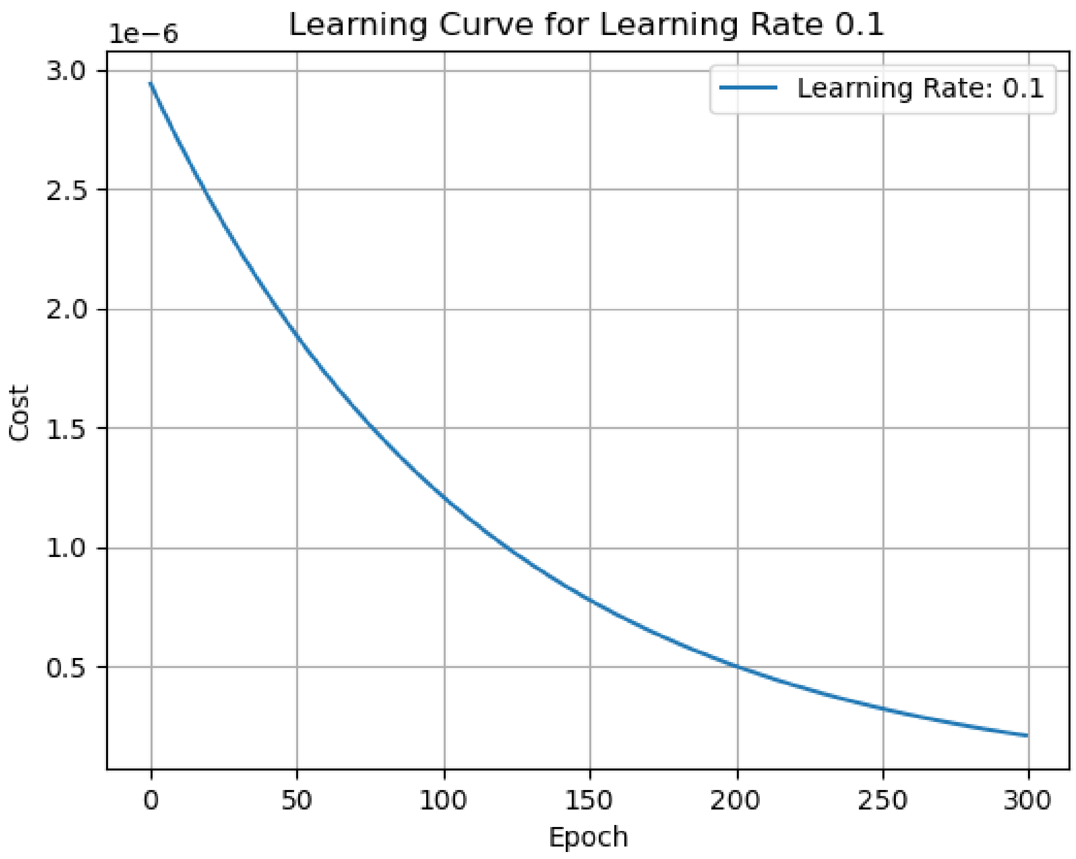
```python
# Import dependencies
import numpy as np
import matplotlib.pyplot as plt
```

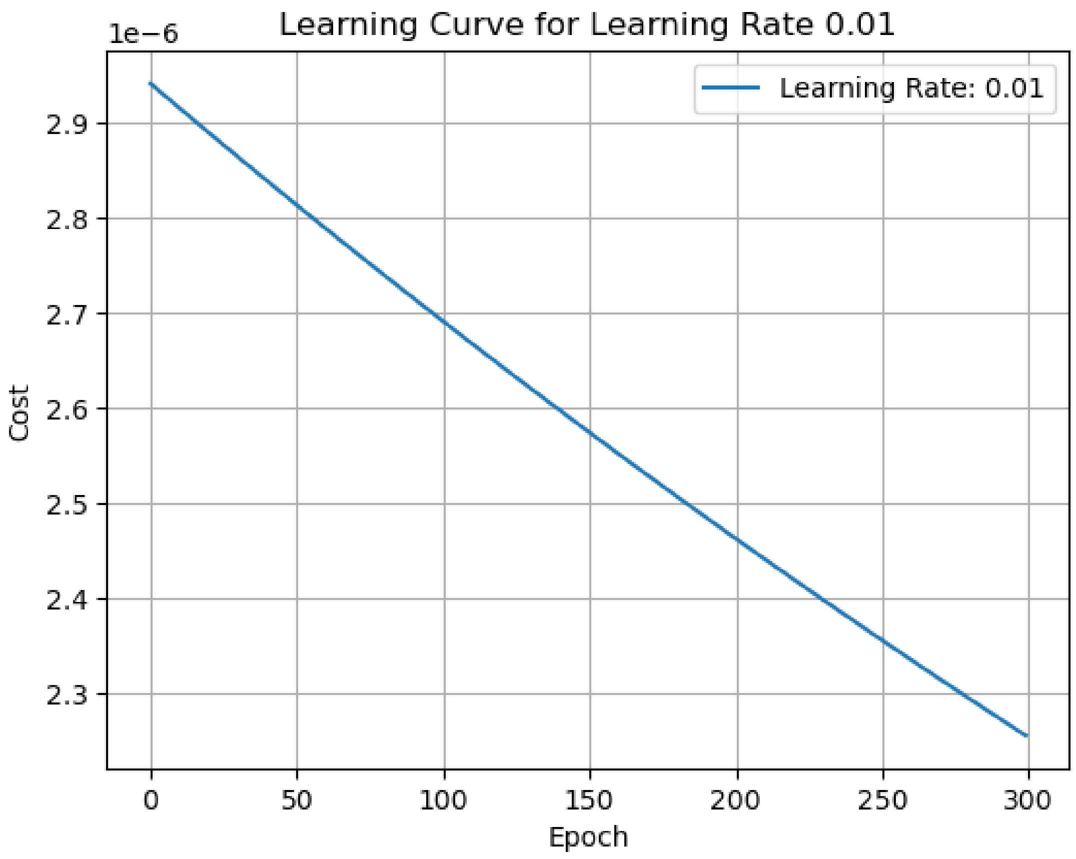
Problem 3

```python
# Define Sigmoid function
def sigmoid(z):
    return 1 / (1 + np.exp(-z))

# Derivative of sigmoid function
def sigmoid_derivative(z):
    return sigmoid(z) * (1 - sigmoid(z))

# Perform gradient descent algorithm
def gradient_descent(w, b, target, learning_rate, epochs):
    costs = []
    for epoch in range(epochs):
        # Forward pass
        z = w * 1 + b  # Input is 1
        output = sigmoid(z)

        # Calculate the cost (loss)
        cost = 0.5 * (output - target) ** 2
        costs.append(cost)

        # Backpropagation - compute gradients
        d_cost_d_output = output - target
        d_output_d_z = sigmoid_derivative(z)

        # Gradients for weight and bias
        d_cost_d_w = d_cost_d_output * d_output_d_z * 1  # Input x = 1
        d_cost_d_b = d_cost_d_output * d_output_d_z

        # Update weight and bias
        w -= learning_rate * d_cost_d_w
        b -= learning_rate * d_cost_d_b

    return w, b, costs

# Parameters for Case 1
w1, b1 = 0.60, 0.90  # Initial weight and bias for case 1
target1 = 0.82       # Target output for case 1
epochs = 300         # Number of epochs

# Define a list of learning rates to try
learning_rates = [1, 0.5, 0.1, 0.01, 0.001]

# Loop through each learning rate
for lr in learning_rates:
    # Re-initialize weight and bias for each learning rate
    w1, b1 = 0.60, 0.90  # Reset initial weight and bias for each run

    # Run gradient descent for each learning rate
    w1, b1, costs1 = gradient_descent(w1, b1, target1, lr, epochs)

    # Plot the learning curve for each learning rate
    plt.plot(costs1, label=f'Learning Rate: {lr}')
    plt.xlabel('Epoch')
    plt.ylabel('Cost')
    plt.title(f'Learning Curve for Learning Rate {lr}')
    plt.legend()
    plt.grid(True)
    plt.show()
```

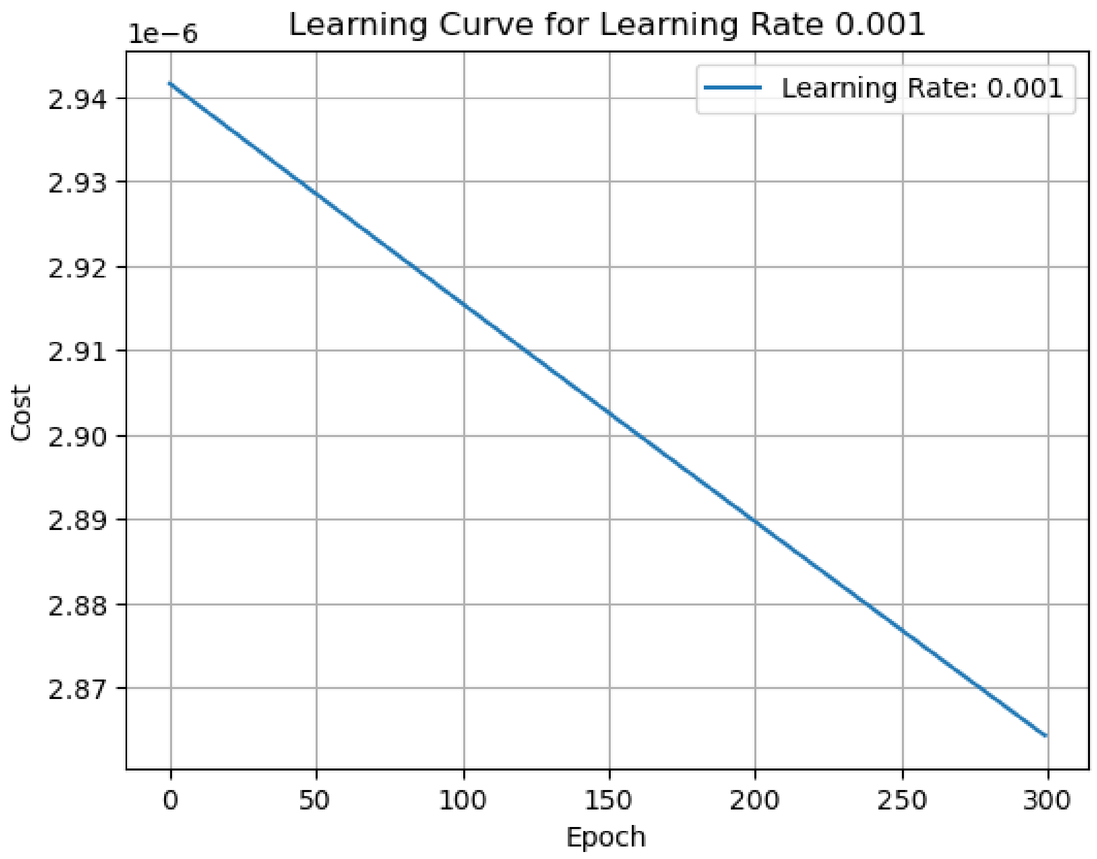



Learning Curve for Learning Rate 0.5
1e−6
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Cost
0
50
100
150
200
250
300
Epoch
Learning Rate: 0.5

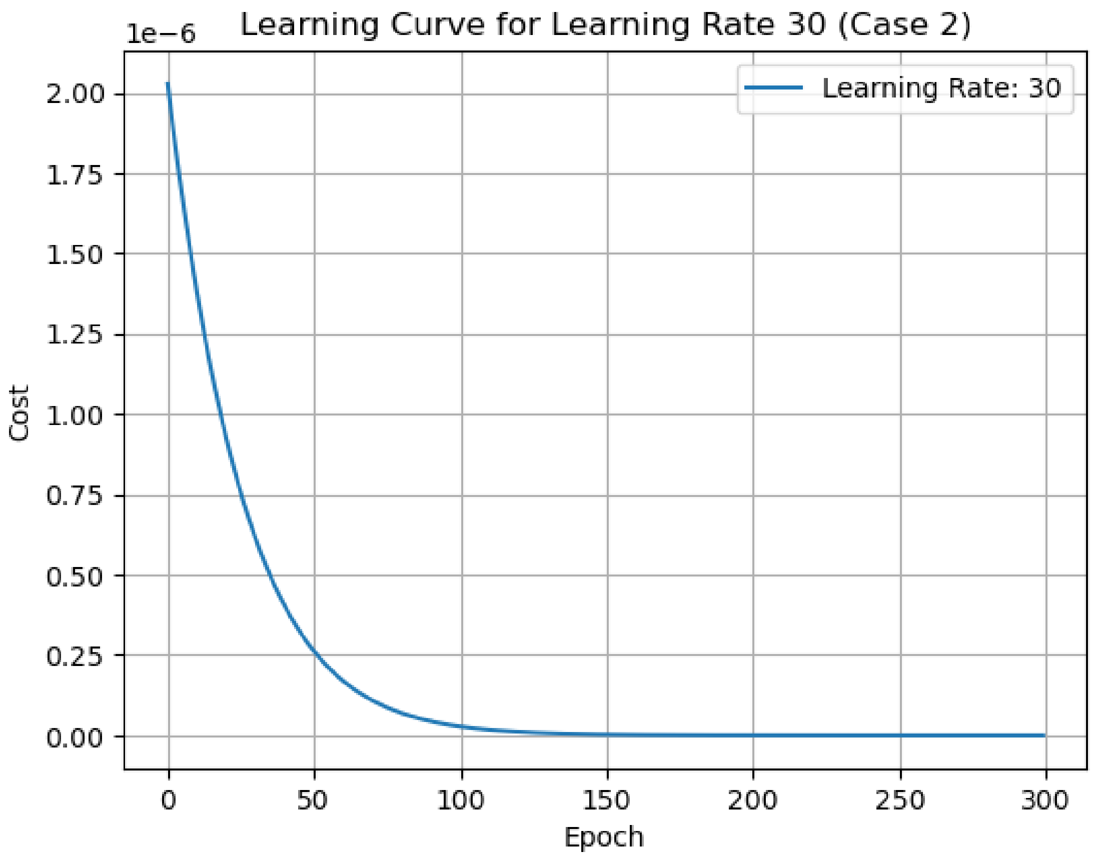

Learning Curve for Learning Rate 0.1
1e−6
3.0
2.5
2.0
1.5
1.0
0.5
Cost
0
50
100
150
200
250
300
Epoch
Learning Rate: 0.1

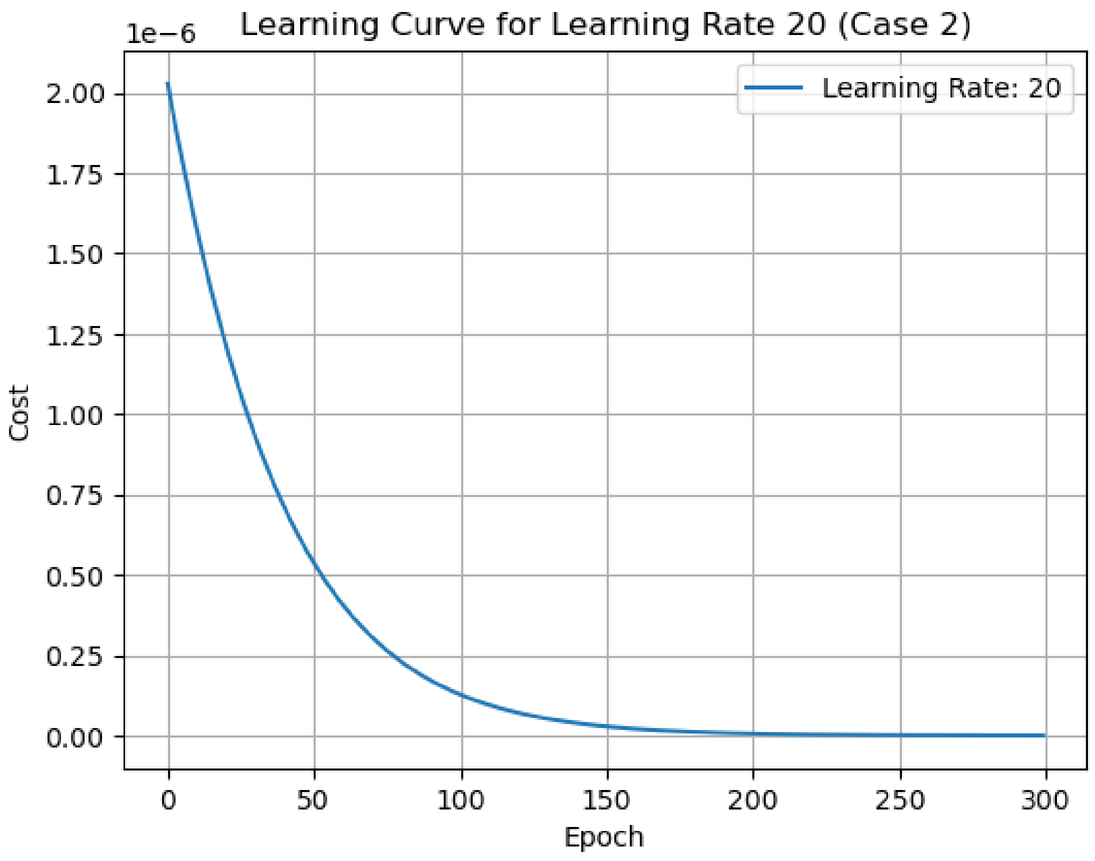


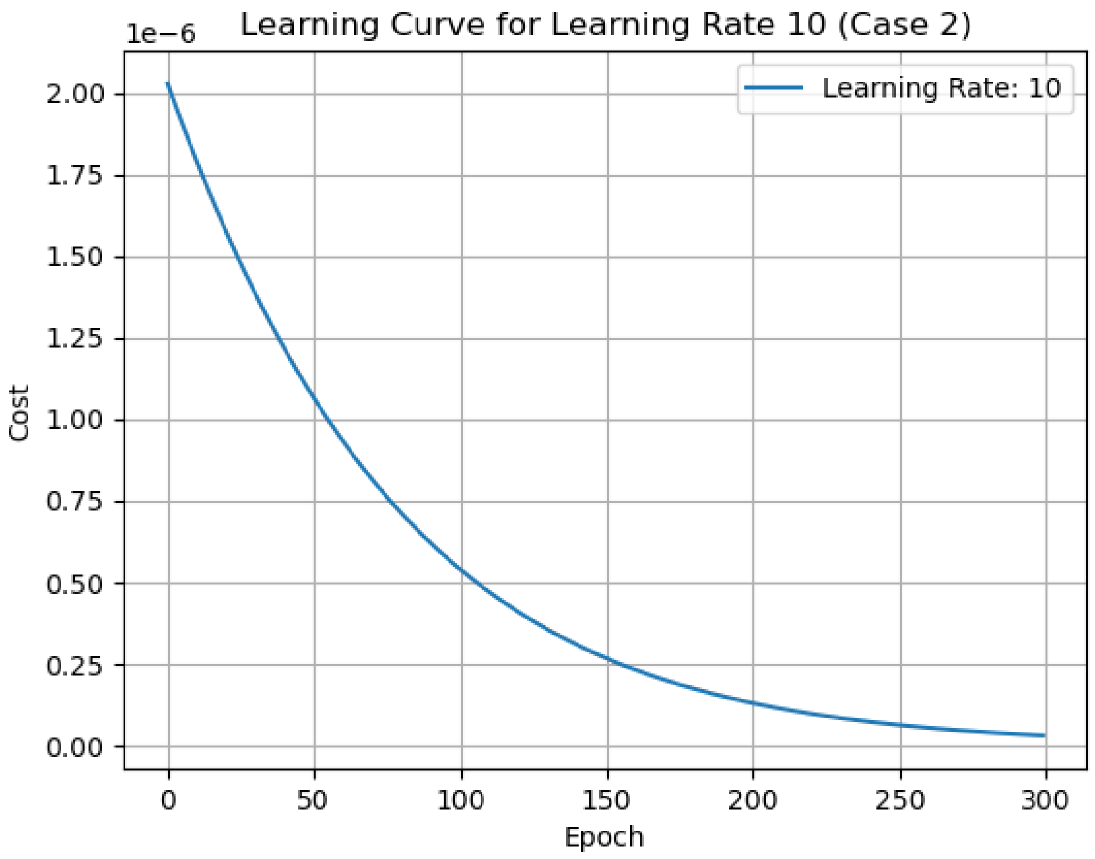
Learning Curve for Learning Rate 0.01
1e−6
Learning Rate: 0.01
Cost
Epoch
2.9
2.8
2.7
2.6
2.5
2.4
2.3
0
50
100
150
200
250
300

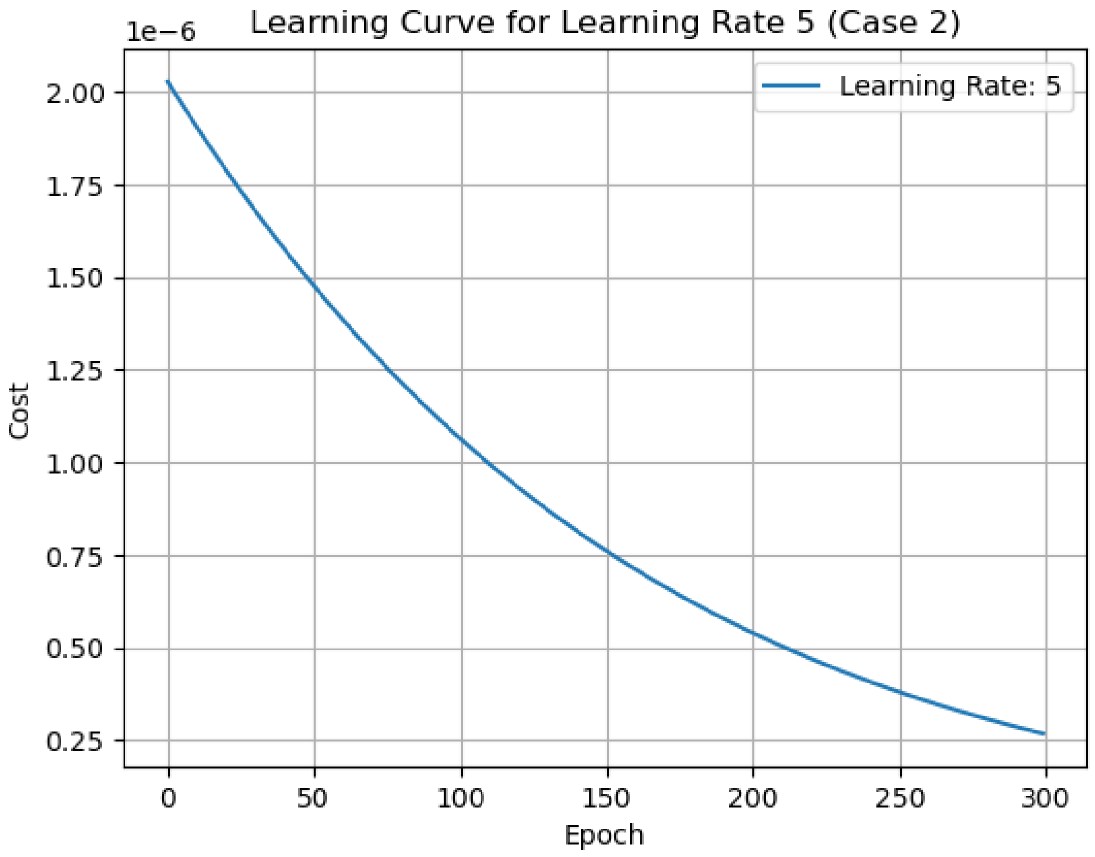


```python
# Define Sigmoid function
def sigmoid(z):
    return 1 / (1 + np.exp(-z))

# Derivative of sigmoid function
def sigmoid_derivative(z):
    return sigmoid(z) * (1 - sigmoid(z))

# Perform gradient descent algorithm
def gradient_descent(w, b, target, learning_rate, epochs):
    costs = []
    for epoch in range(epochs):
        # Forward pass
        z = w * 1 + b  # Input is 1
        output = sigmoid(z)

        # Calculate the cost (loss)
        cost = 0.5 * (output - target) ** 2
        costs.append(cost)

        # Backpropagation - compute gradients
        d_cost_d_output = output - target
        d_output_d_z = sigmoid_derivative(z)
        # Gradients for weight and bias
        d_cost_d_w = d_cost_d_output * d_output_d_z * 1  # Input x = 1
        d_cost_d_b = d_cost_d_output * d_output_d_z

        # Update weight and bias
        w -= learning_rate * d_cost_d_w
        b -= learning_rate * d_cost_d_b

    return w, b, costs

# Parameters for Case 2
w2, b2 = 2.00, 2.00  # Initial weight and bias for case 2
target2 = 0.98       # Target output for case 2
epochs = 300         # Number of epochs

# Define a list of learning rates to try
learning_rates = [30, 20, 10, 5, 1]

# Loop through each learning rate
for lr in learning_rates:
    # Re-initialize weight and bias for each learning rate
    w2, b2 = 2.00, 2.00  # Reset initial weight and bias for each run

    # Run gradient descent for each learning rate
    w2, b2, costs2 = gradient_descent(w2, b2, target2, lr, epochs)

    # Plot the learning curve for each learning rate
    plt.plot(costs2, label=f'Learning Rate: {lr}')
    plt.xlabel('Epoch')
    plt.ylabel('Cost')
    plt.title(f'Learning Curve for Learning Rate {lr} (Case 2)')
    plt.legend()
    plt.grid(True)
    plt.show()
```

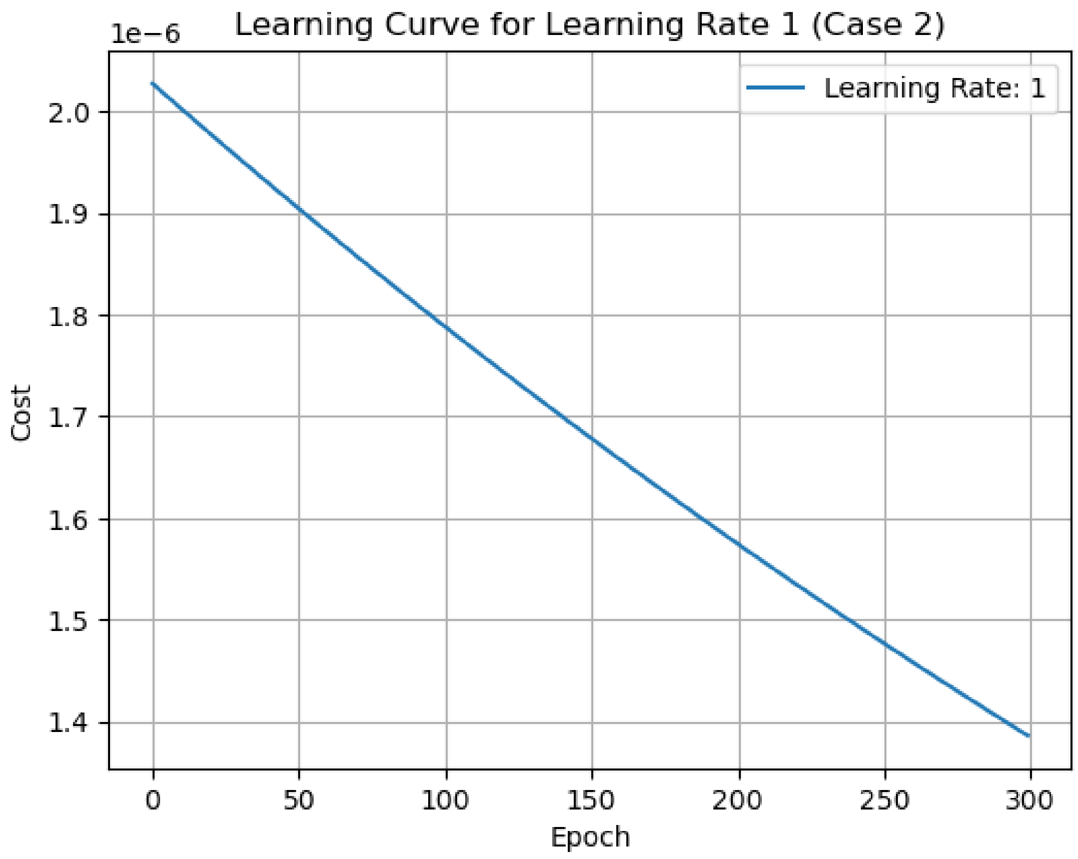

Learning Curve for Learning Rate 30 (Case 2)
1e−6
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Cost
0
50
100
150
200
250
300
Epoch
Learning Rate: 30


Learning Curve for Learning Rate 20 (Case 2)
1e−6
Learning Rate: 20
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Cost
0
50
100
150
200
250
300
Epoch

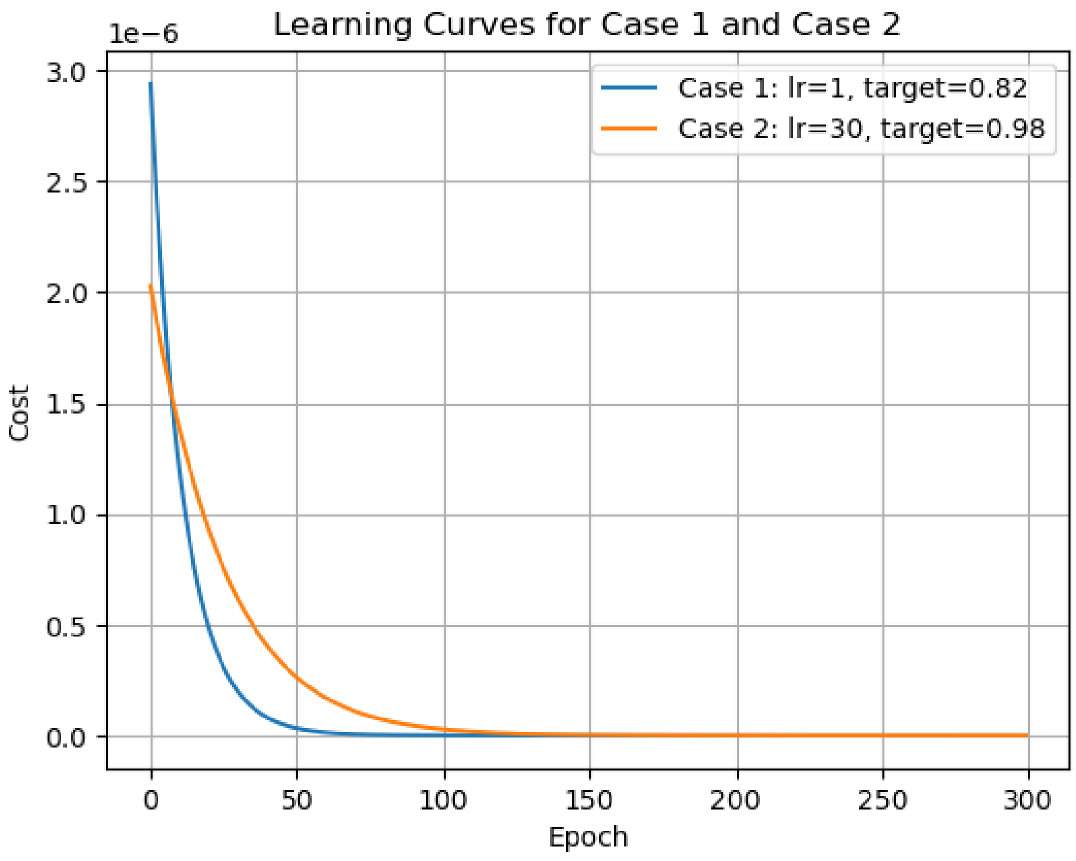

Learning Curve for Learning Rate 10 (Case 2)
1e−6
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Cost
0
50
100
150
200
250
300
Epoch
Learning Rate: 10



Learning Curve for Learning Rate 5 (Case 2)
1e−6
Learning Rate: 5
Cost
Epoch
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0
50
100
150
200
250
300



```python
# Define Sigmoid function
def sigmoid(z):
    return 1 / (1 + np.exp(-z))

# Derivative of sigmoid function
def sigmoid_derivative(z):
    return sigmoid(z) * (1 - sigmoid(z))

# Perform gradient descent algorithm
def gradient_descent(w, b, target, learning_rate, epochs):
    costs = []
    for epoch in range(epochs):
        # Forward pass
        z = w * 1 + b  # Input is 1
        output = sigmoid(z)

        # Calculate the cost (loss)
        cost = 0.5 * (output - target) ** 2
        costs.append(cost)

        # Backpropagation - compute gradients
        d_cost_d_output = output - target
        d_output_d_z = sigmoid_derivative(z)
        # Gradients for weight and bias
        d_cost_d_w = d_cost_d_output * d_output_d_z * 1  # Input x = 1
        d_cost_d_b = d_cost_d_output * d_output_d_z

        # Update weight and bias
        w -= learning_rate * d_cost_d_w
        b -= learning_rate * d_cost_d_b

    return w, b, costs

# Parameters for Case 1
w1, b1 = 0.60, 0.90  # Initial weight and bias for Case 1
target1 = 0.82       # Target output for Case 1
learning_rate1 = 1  # Learning rate for Case 1
epochs = 300         # Number of epochs

# Parameters for Case 2
w2, b2 = 2.00, 2.00  # Initial weight and bias for Case 2
target2 = 0.98       # Target output for Case 2
learning_rate2 = 30  # Learning rate for Case 2

# Run gradient descent for Case 1
w1, b1, costs1 = gradient_descent(w1, b1, target1, learning_rate1,
epochs)

# Run gradient descent for Case 2
w2, b2, costs2 = gradient_descent(w2, b2, target2, learning_rate2,
epochs)

# Plot the learning curves for both cases on the same plot
plt.plot(costs1, label=f'Case 1: lr={learning_rate1}, target=0.82')
plt.plot(costs2, label=f'Case 2: lr={learning_rate2}, target=0.98')
plt.xlabel('Epoch')
plt.ylabel('Cost')
plt.title('Learning Curves for Case 1 and Case 2')
plt.legend()
plt.grid(True)
plt.show()
```


Learning Curves for Case 1 and Case 2
1e−6
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Cost
0
50
100
150
200
250
300
Epoch
Case 1: lr=1, target=0.82
Case 2: lr=30, target=0.98

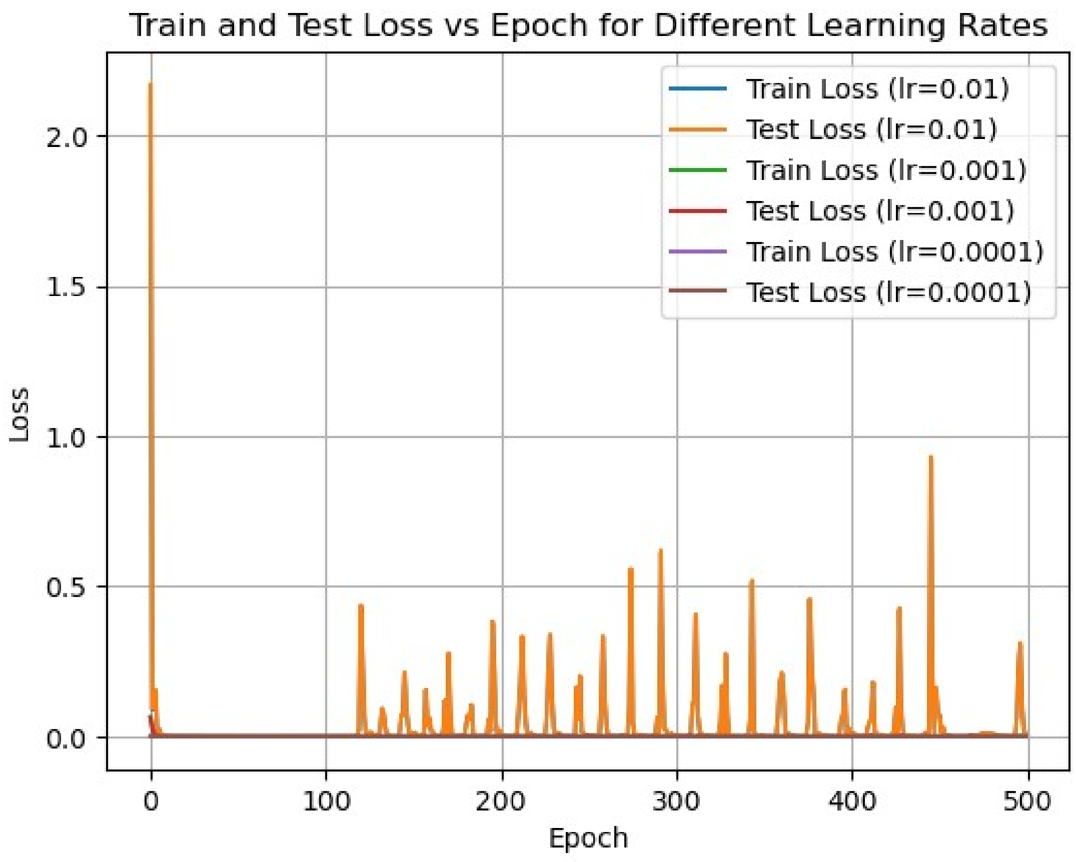
```
# Import dependencies
import numpy as np
import torch
import torchvision
from torch.utils.data.dataset import Dataset
from torchvision import datasets, transforms
from torch import nn, optim
import matplotlib.pyplot as plt
```

Problem 4.1 Determine an appropriate learning rate η

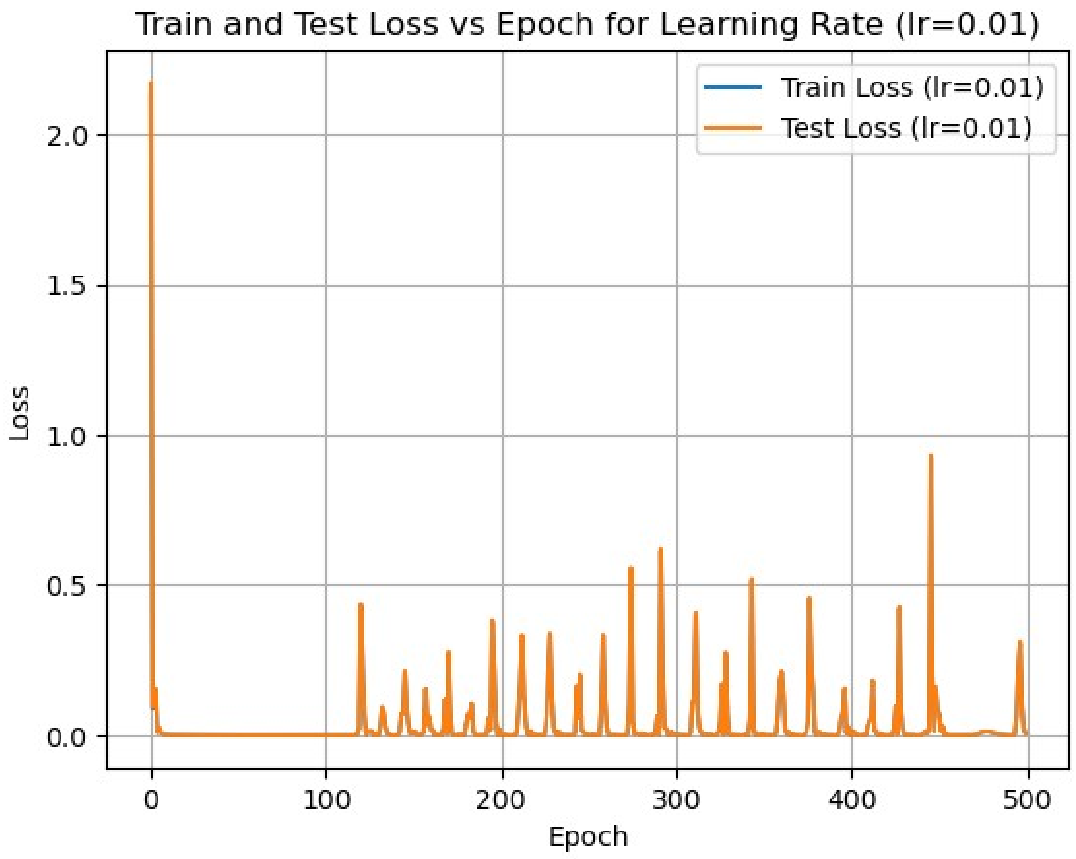
```
# ################## Part 1: load data and create batch
##################
N_total = 600
N_train = 500
x = torch.unsqueeze(torch.linspace(0, 1, N_total), dim=1)
r = torch.randperm(N_total)
x = x[r, :]
y = 0.2 + 0.4 * torch.pow(x, 2) + 0.3 * x * torch.sin(15 * x) + 0.05 *
torch.cos(50 * x)

class CustomDataset(Dataset):
    def __init__(self, x, y):
        self.y = y
        self.x = x

    def __len__(self):
        return len(self.y)

    def __getitem__(self, idx):
        y1 = self.y[idx]
        x1 = self.x[idx]
        return (x1, y1)

trainset = CustomDataset(x[0:N_train, :], y[0:N_train, :])
testset = CustomDataset(x[N_train:N_total, :], y[N_train:N_total, :])
train_loader = torch.utils.data.DataLoader(trainset, batch_size=50)
test_loader = torch.utils.data.DataLoader(testset, batch_size=50)

# ################## Part 2: Define Model and initialize
##################
# Neural network model
model = nn.Sequential(
    nn.Linear(1, 1024, bias=True),
    nn.ReLU(),
    nn.Linear(1024, 1, bias=True)
)

def init_weights(m):
    if isinstance(m, nn.Linear):
        m.weight.data.uniform_(-1, 1)
        m.bias.data.uniform_(-1, 1)

model.apply(init_weights)

Sequential(
  (0): Linear(in_features=1, out_features=1024, bias=True)
  (1): ReLU()
  (2): Linear(in_features=1024, out_features=1, bias=True)
)

# ################## Part 3: Define Loss and optimizer
##################
# Using L2 loss (mean squared error)
criterion = torch.nn.MSELoss()

# ################## Part 4: Optimization ##################
def train_NN(optimizer):
    model.train()
    for images, labels in train_loader:
        images = images.view(images.shape[0], -1)
        out = model(images)
        loss = criterion(out, labels)
        loss.backward()
        optimizer.step()
        optimizer.zero_grad()
    return loss.item()

def test_NN(loader):
    model.eval()
    loss = 0
    with torch.no_grad():
        for images, labels in loader:
            images = images.view(images.shape[0], -1)
            out = model(images)
            loss += criterion(out, labels).item()
    return loss / len(loader)

# ################## Experiment with Different Learning Rates
##################
N_epoch = 500
learning_rates = [0.01, 0.001, 0.0001]
train_losses = {}
test_losses = {}

for lr in learning_rates:
    # Initialize a new optimizer with the current learning rate
    optimizer = torch.optim.Adam(model.parameters(), lr=lr)

    train_loss = np.zeros((N_epoch, 1))
    test_loss = np.zeros((N_epoch, 1))

    # Training loop
    for epoch in range(N_epoch):
        train_NN(optimizer)  # Train the model
        train_loss[epoch, 0] = test_NN(train_loader)  # Evaluate train
loss
        test_loss[epoch, 0] = test_NN(test_loader)  # Evaluate test
loss
        if epoch % 50 == 0:
            print(f'Epoch: {epoch:03d}, LR: {lr}, Train loss:
{train_loss[epoch, 0]:.7f}, Test loss: {test_loss[epoch, 0]:.7f}')

    train_losses[lr] = train_loss
    test_losses[lr] = test_loss
```

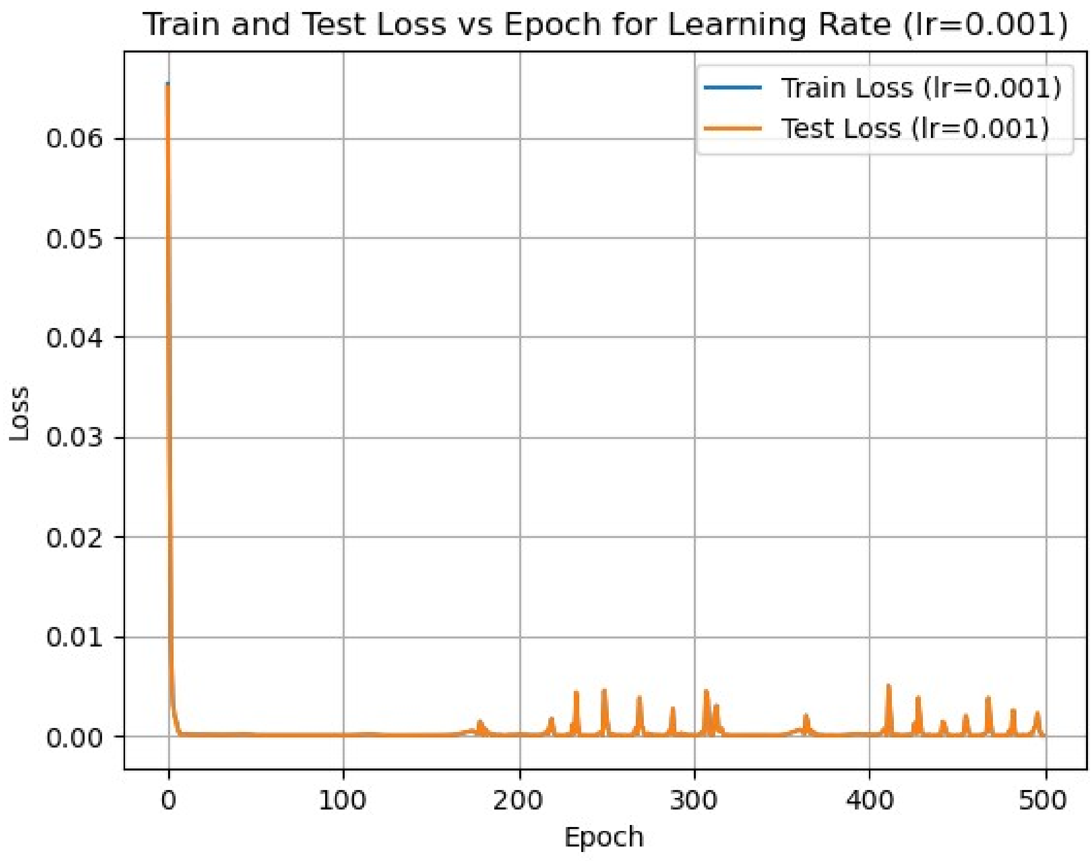
```
Epoch: 000, LR: 0.01, Train loss: 2.1317052, Test loss: 2.1690880
Epoch: 050, LR: 0.01, Train loss: 0.0001825, Test loss: 0.0001889
Epoch: 100, LR: 0.01, Train loss: 0.0000593, Test loss: 0.0000747
Epoch: 150, LR: 0.01, Train loss: 0.0078053, Test loss: 0.0078132
Epoch: 200, LR: 0.01, Train loss: 0.0020632, Test loss: 0.0019835
Epoch: 250, LR: 0.01, Train loss: 0.0019578, Test loss: 0.0020027
Epoch: 300, LR: 0.01, Train loss: 0.0029916, Test loss: 0.0030689
Epoch: 350, LR: 0.01, Train loss: 0.0030511, Test loss: 0.0030991
Epoch: 400, LR: 0.01, Train loss: 0.0010209, Test loss: 0.0009583
Epoch: 450, LR: 0.01, Train loss: 0.0716941, Test loss: 0.0711721
Epoch: 000, LR: 0.001, Train loss: 0.0653664, Test loss: 0.0650113
Epoch: 050, LR: 0.001, Train loss: 0.0000592, Test loss: 0.0000514
Epoch: 100, LR: 0.001, Train loss: 0.0000378, Test loss: 0.0000346
Epoch: 150, LR: 0.001, Train loss: 0.0000286, Test loss: 0.0000330
Epoch: 200, LR: 0.001, Train loss: 0.0001094, Test loss: 0.0001163
Epoch: 250, LR: 0.001, Train loss: 0.0019679, Test loss: 0.0020278
Epoch: 300, LR: 0.001, Train loss: 0.0000367, Test loss: 0.0000370
Epoch: 350, LR: 0.001, Train loss: 0.0000467, Test loss: 0.0000538
Epoch: 400, LR: 0.001, Train loss: 0.0000985, Test loss: 0.0001059
Epoch: 450, LR: 0.001, Train loss: 0.0000418, Test loss: 0.0000469
Epoch: 000, LR: 0.0001, Train loss: 0.0010729, Test loss: 0.0010951
Epoch: 050, LR: 0.0001, Train loss: 0.0000196, Test loss: 0.0000221
Epoch: 100, LR: 0.0001, Train loss: 0.0000143, Test loss: 0.0000154
Epoch: 150, LR: 0.0001, Train loss: 0.0000155, Test loss: 0.0000162
Epoch: 200, LR: 0.0001, Train loss: 0.0000192, Test loss: 0.0000193
Epoch: 250, LR: 0.0001, Train loss: 0.0000186, Test loss: 0.0000185
Epoch: 300, LR: 0.0001, Train loss: 0.0000185, Test loss: 0.0000182
Epoch: 350, LR: 0.0001, Train loss: 0.0000186, Test loss: 0.0000182
Epoch: 400, LR: 0.0001, Train loss: 0.0000179, Test loss: 0.0000175
Epoch: 450, LR: 0.0001, Train loss: 0.0000171, Test loss: 0.0000166
```

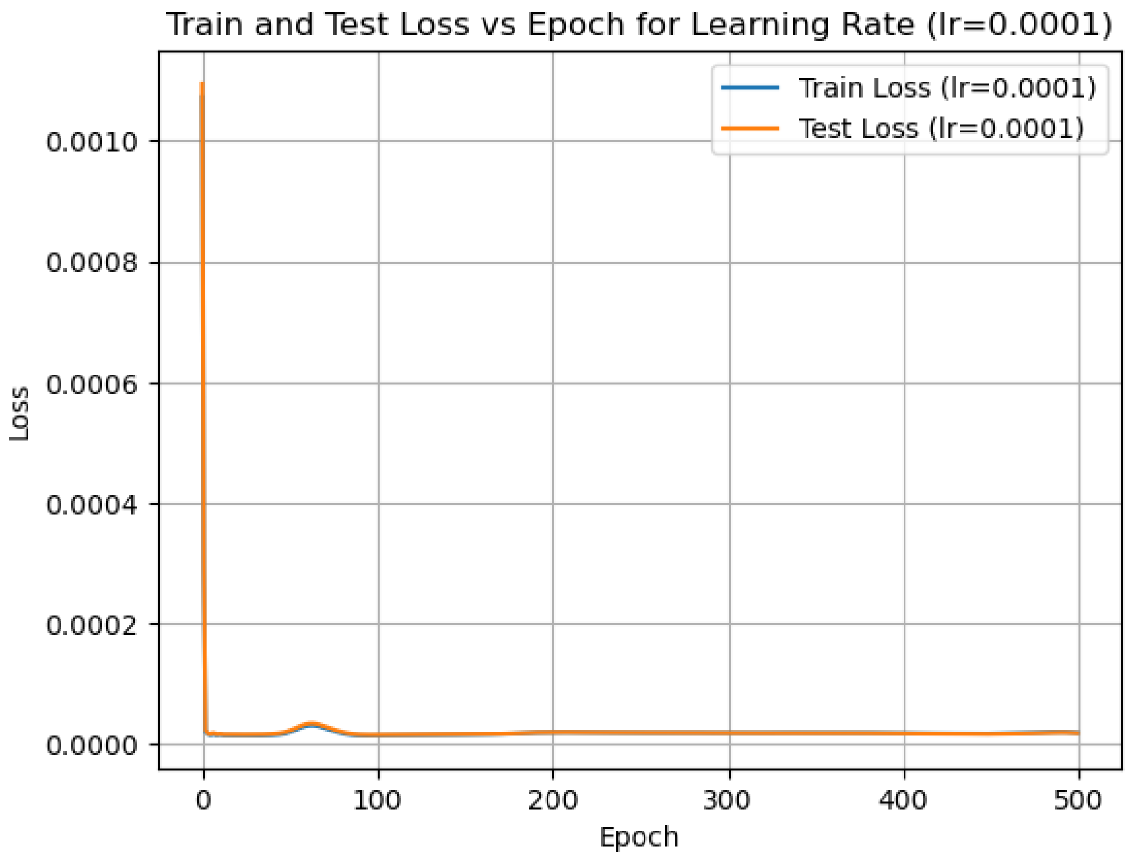
```
# ################## Plot Results ##################
for lr in learning_rates:
    plt.plot(train_losses[lr], label=f'Train Loss (lr={lr})')
    plt.plot(test_losses[lr], label=f'Test Loss (lr={lr})')

plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Train and Test Loss vs Epoch for Different Learning Rates')
plt.legend()
plt.grid(True)
plt.show()
```



```
# ################## Plot Results for each learning rate
##################
for lr in learning_rates:
    plt.figure()  # Create a new figure for each learning rate
    plt.plot(train_losses[lr], label=f'Train Loss (lr={lr})')
    plt.plot(test_losses[lr], label=f'Test Loss (lr={lr})')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.title(f'Train and Test Loss vs Epoch for Learning Rate
(lr={lr})')
    plt.legend()
    plt.grid(True)
    plt.show()
```


Train and Test Loss vs Epoch for Learning Rate (lr=0.01)
Train Loss (lr=0.01)
Test Loss (lr=0.01)
Loss
Epoch
0.0
0.5
1.0
1.5
2.0
0
100
200
300
400
500


Train and Test Loss vs Epoch for Learning Rate (lr=0.001)
Train Loss (lr=0.001)
Test Loss (lr=0.001)
Loss
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
100
200
300
400
500
Epoch



- Learning Rate η = 0.01: In the first graph, although the training and test losses decrease rapidly at the start, there are significant fluctuations throughout the training process. These fluctuations indicate that the learning rate is too large, causing instability. The model updates too aggressively, resulting in the loss bouncing around and not converging smoothly.

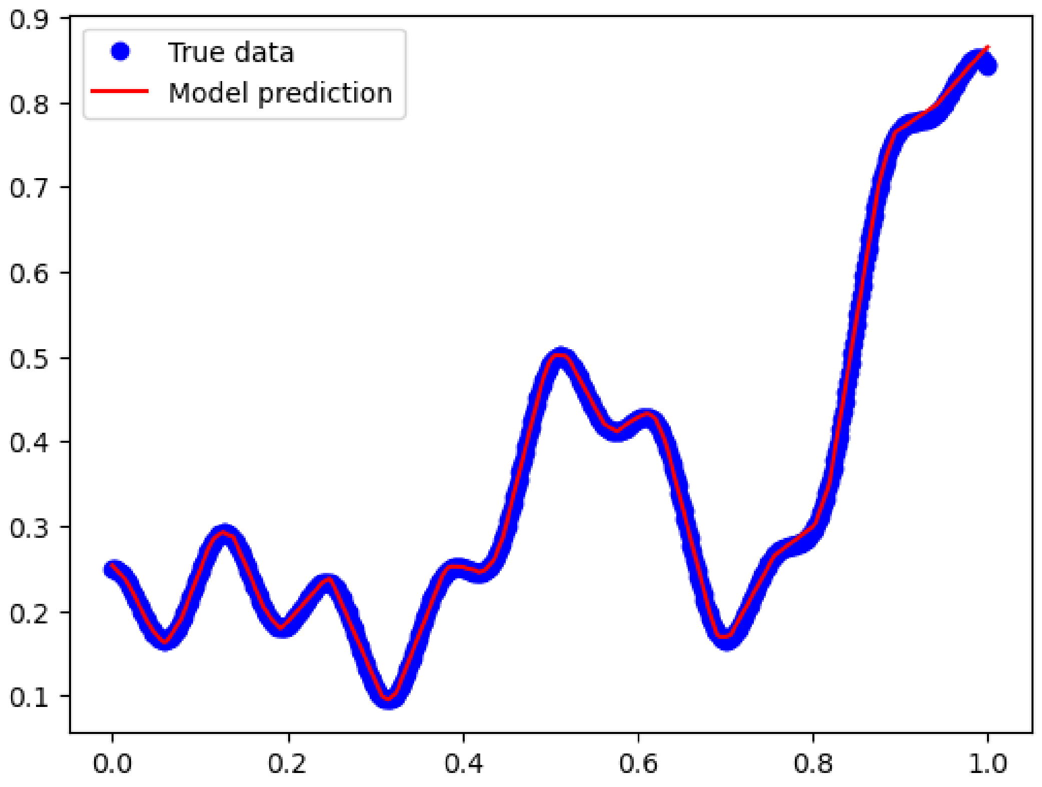
- Learning Rate η = 0.001: The second graph shows a rapid initial decrease in the loss, with relatively minor fluctuations. The loss continues to decrease and eventually stabilizes, though some minor oscillations remain. Compared to η = 0.01, this learning rate results in more stable training, and the model converges relatively fast without large fluctuations.

- Learning Rate η = 0.0001: In the third graph, the loss decreases very smoothly but converges more slowly. The loss is very small throughout the training process, but the model takes longer to reach this point. Although this learning rate is stable, it is slower in convergence, which may require more epochs to achieve optimal results.

- Conclusion: The recommended learning rate is η = 0.001.

```
# ################## Final Prediction ##################
x_test = torch.unsqueeze(torch.linspace(0, 1, 1999), dim=1)
y_test = model(x_test)
```

```python
# Plot the results
plt.plot(x[0:N_total], y[0:N_total], 'bo', label='True data')
plt.plot(x_test, y_test.detach().numpy(), 'r', label='Model
prediction')
plt.legend()
plt.show()
```



Determine an appropriate learning rate η = 0.001

```python
# Import dependencies
import numpy as np
import torch
import torchvision
from torch.utils.data.dataset import Dataset
from torchvision import datasets, transforms
from torch import nn, optim
import matplotlib.pyplot as plt
```

Problem 4.2 Determine appropriate mini-batch size = 32, epoch = 100

```python
# ################## Part 1: Load data and create batch
##################
N_total = 600
N_train = 500
x = torch.unsqueeze(torch.linspace(0, 1, N_total), dim=1)
r = torch.randperm(N_total)
x = x[r, :]
y = 0.2 + 0.4 * torch.pow(x, 2) + 0.3 * x * torch.sin(15 * x) + 0.05 *
torch.cos(50 * x)

class CustomDataset(Dataset):
    def __init__(self, x, y):
        self.y = y
        self.x = x

    def __len__(self):
        return len(self.y)

    def __getitem__(self, idx):
        y1 = self.y[idx]
        x1 = self.x[idx]
        return (x1, y1)

# Change batch_size here to test different values
batch_size = 32  # Experiment with different batch sizes: 32, 64, 128
trainset = CustomDataset(x[0:N_train, :], y[0:N_train, :])
testset = CustomDataset(x[N_train:N_total, :], y[N_train:N_total, :])
train_loader = torch.utils.data.DataLoader(trainset,
batch_size=batch_size)
test_loader = torch.utils.data.DataLoader(testset,
batch_size=batch_size)

# ################## Part 2: Define Model ##################
model = nn.Sequential(
    nn.Linear(1, 1024, bias=True),
    nn.ReLU(),
    nn.Linear(1024, 1, bias=True)
)
```

```
def init_weights(m):
    if isinstance(m, nn.Linear):
        m.weight.data.uniform_(-1, 1)
        m.bias.data.uniform_(-1, 1)

model.apply(init_weights)
```

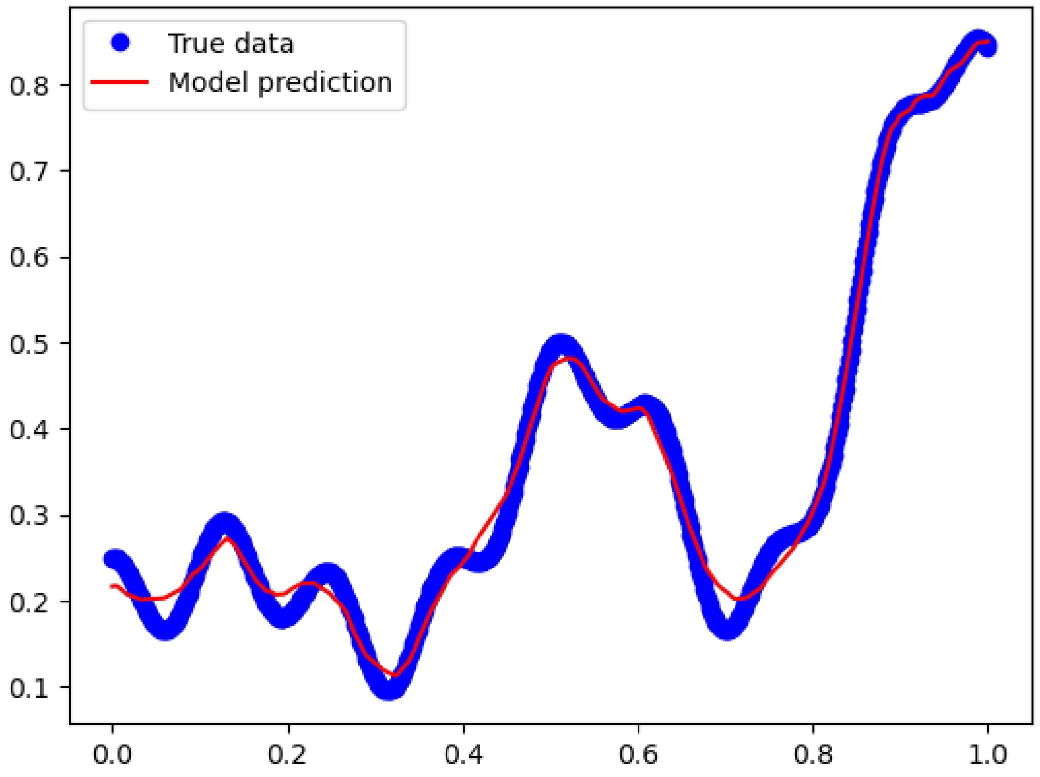
```
Sequential(
  (0): Linear(in_features=1, out_features=1024, bias=True)
  (1): ReLU()
  (2): Linear(in_features=1024, out_features=1, bias=True)
)
```

```
# ################## Part 3: Define Loss and Optimizer
##################
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

# ################## Part 4: Training and Testing ##################
def train_NN():
    model.train()
    for images, labels in train_loader:
        out = model(images)
        loss = criterion(out, labels)
        loss.backward()
        optimizer.step()
        optimizer.zero_grad()
    return loss

def test_NN(loader):
    model.eval()
    loss = 0
    with torch.no_grad():
        for images, labels in loader:
            out = model(images)
            loss += criterion(out, labels).item()
    loss = loss / len(loader)
    return loss

# Experiment with different numbers of epochs
N_epoch = 100
train_loss = np.zeros((N_epoch, 1))
test_loss = np.zeros((N_epoch, 1))

for epoch in range(N_epoch):
    train_NN()
    train_loss[epoch, 0] = test_NN(train_loader)
    test_loss[epoch, 0] = test_NN(test_loader)
    print(f'Epoch: {epoch+1:03d}, Train Loss: {train_loss[epoch,
0]:.7f}, Test Loss: {test_loss[epoch, 0]:.7f}')
```

```
Epoch: 001, Train Loss: 3.0408284, Test Loss: 2.6833682
Epoch: 002, Train Loss: 0.1522782, Test Loss: 0.1523975
Epoch: 003, Train Loss: 0.0454572, Test Loss: 0.0447318
Epoch: 004, Train Loss: 0.0322065, Test Loss: 0.0324125
Epoch: 005, Train Loss: 0.0231701, Test Loss: 0.0234418
Epoch: 006, Train Loss: 0.0173245, Test Loss: 0.0179732
Epoch: 007, Train Loss: 0.0132794, Test Loss: 0.0145717
Epoch: 008, Train Loss: 0.0106569, Test Loss: 0.0114137
Epoch: 009, Train Loss: 0.0086298, Test Loss: 0.0096973
Epoch: 010, Train Loss: 0.0073026, Test Loss: 0.0081288
Epoch: 011, Train Loss: 0.0063120, Test Loss: 0.0070580
Epoch: 012, Train Loss: 0.0055604, Test Loss: 0.0062362
Epoch: 013, Train Loss: 0.0049826, Test Loss: 0.0055524
Epoch: 014, Train Loss: 0.0045235, Test Loss: 0.0050087
Epoch: 015, Train Loss: 0.0041535, Test Loss: 0.0045653
Epoch: 016, Train Loss: 0.0038502, Test Loss: 0.0041954
Epoch: 017, Train Loss: 0.0035964, Test Loss: 0.0038838
Epoch: 018, Train Loss: 0.0033807, Test Loss: 0.0036204
Epoch: 019, Train Loss: 0.0031939, Test Loss: 0.0033937
Epoch: 020, Train Loss: 0.0030302, Test Loss: 0.0031965
Epoch: 021, Train Loss: 0.0028844, Test Loss: 0.0030234
Epoch: 022, Train Loss: 0.0027535, Test Loss: 0.0028683
Epoch: 023, Train Loss: 0.0026343, Test Loss: 0.0027294
Epoch: 024, Train Loss: 0.0025246, Test Loss: 0.0026046
Epoch: 025, Train Loss: 0.0024233, Test Loss: 0.0024920
Epoch: 026, Train Loss: 0.0023288, Test Loss: 0.0023890
Epoch: 027, Train Loss: 0.0022407, Test Loss: 0.0022948
Epoch: 028, Train Loss: 0.0021583, Test Loss: 0.0022080
Epoch: 029, Train Loss: 0.0020810, Test Loss: 0.0021280
Epoch: 030, Train Loss: 0.0020086, Test Loss: 0.0020535
Epoch: 031, Train Loss: 0.0019404, Test Loss: 0.0019846
Epoch: 032, Train Loss: 0.0018763, Test Loss: 0.0019204
Epoch: 033, Train Loss: 0.0018161, Test Loss: 0.0018605
Epoch: 034, Train Loss: 0.0017592, Test Loss: 0.0018048
Epoch: 035, Train Loss: 0.0017059, Test Loss: 0.0017534
Epoch: 036, Train Loss: 0.0016557, Test Loss: 0.0017055
Epoch: 037, Train Loss: 0.0016081, Test Loss: 0.0016602
Epoch: 038, Train Loss: 0.0015631, Test Loss: 0.0016174
Epoch: 039, Train Loss: 0.0015205, Test Loss: 0.0015769
Epoch: 040, Train Loss: 0.0014798, Test Loss: 0.0015377
Epoch: 041, Train Loss: 0.0014411, Test Loss: 0.0014996
Epoch: 042, Train Loss: 0.0014041, Test Loss: 0.0014621
Epoch: 043, Train Loss: 0.0013684, Test Loss: 0.0014252
Epoch: 044, Train Loss: 0.0013336, Test Loss: 0.0013883
Epoch: 045, Train Loss: 0.0012997, Test Loss: 0.0013510
Epoch: 046, Train Loss: 0.0012664, Test Loss: 0.0013139
Epoch: 047, Train Loss: 0.0012337, Test Loss: 0.0012764
Epoch: 048, Train Loss: 0.0012013, Test Loss: 0.0012382
Epoch: 049, Train Loss: 0.0011696, Test Loss: 0.0012003
Epoch: 050, Train Loss: 0.0011387, Test Loss: 0.0011627
```

```
Epoch: 051, Train Loss: 0.0011085, Test Loss: 0.0011258
Epoch: 052, Train Loss: 0.0010792, Test Loss: 0.0010898
Epoch: 053, Train Loss: 0.0010507, Test Loss: 0.0010550
Epoch: 054, Train Loss: 0.0010232, Test Loss: 0.0010215
Epoch: 055, Train Loss: 0.0009967, Test Loss: 0.0009893
Epoch: 056, Train Loss: 0.0009712, Test Loss: 0.0009583
Epoch: 057, Train Loss: 0.0009468, Test Loss: 0.0009286
Epoch: 058, Train Loss: 0.0009233, Test Loss: 0.0009000
Epoch: 059, Train Loss: 0.0009008, Test Loss: 0.0008730
Epoch: 060, Train Loss: 0.0008792, Test Loss: 0.0008472
Epoch: 061, Train Loss: 0.0008583, Test Loss: 0.0008225
Epoch: 062, Train Loss: 0.0008382, Test Loss: 0.0007989
Epoch: 063, Train Loss: 0.0008187, Test Loss: 0.0007765
Epoch: 064, Train Loss: 0.0008000, Test Loss: 0.0007552
Epoch: 065, Train Loss: 0.0007819, Test Loss: 0.0007349
Epoch: 066, Train Loss: 0.0007644, Test Loss: 0.0007156
Epoch: 067, Train Loss: 0.0007475, Test Loss: 0.0006971
Epoch: 068, Train Loss: 0.0007312, Test Loss: 0.0006794
Epoch: 069, Train Loss: 0.0007153, Test Loss: 0.0006624
Epoch: 070, Train Loss: 0.0006999, Test Loss: 0.0006460
Epoch: 071, Train Loss: 0.0006849, Test Loss: 0.0006304
Epoch: 072, Train Loss: 0.0006705, Test Loss: 0.0006155
Epoch: 073, Train Loss: 0.0006564, Test Loss: 0.0006010
Epoch: 074, Train Loss: 0.0006427, Test Loss: 0.0005871
Epoch: 075, Train Loss: 0.0006294, Test Loss: 0.0005737
Epoch: 076, Train Loss: 0.0006166, Test Loss: 0.0005608
Epoch: 077, Train Loss: 0.0006040, Test Loss: 0.0005485
Epoch: 078, Train Loss: 0.0005917, Test Loss: 0.0005365
Epoch: 079, Train Loss: 0.0005798, Test Loss: 0.0005250
Epoch: 080, Train Loss: 0.0005683, Test Loss: 0.0005138
Epoch: 081, Train Loss: 0.0005570, Test Loss: 0.0005031
Epoch: 082, Train Loss: 0.0005459, Test Loss: 0.0004926
Epoch: 083, Train Loss: 0.0005352, Test Loss: 0.0004824
Epoch: 084, Train Loss: 0.0005248, Test Loss: 0.0004723
Epoch: 085, Train Loss: 0.0005147, Test Loss: 0.0004624
Epoch: 086, Train Loss: 0.0005050, Test Loss: 0.0004528
Epoch: 087, Train Loss: 0.0004954, Test Loss: 0.0004435
Epoch: 088, Train Loss: 0.0004861, Test Loss: 0.0004343
Epoch: 089, Train Loss: 0.0004769, Test Loss: 0.0004254
Epoch: 090, Train Loss: 0.0004681, Test Loss: 0.0004168
Epoch: 091, Train Loss: 0.0004594, Test Loss: 0.0004084
Epoch: 092, Train Loss: 0.0004510, Test Loss: 0.0004001
Epoch: 093, Train Loss: 0.0004428, Test Loss: 0.0003921
Epoch: 094, Train Loss: 0.0004346, Test Loss: 0.0003844
Epoch: 095, Train Loss: 0.0004266, Test Loss: 0.0003767
Epoch: 096, Train Loss: 0.0004188, Test Loss: 0.0003693
Epoch: 097, Train Loss: 0.0004112, Test Loss: 0.0003621
Epoch: 098, Train Loss: 0.0004038, Test Loss: 0.0003550
```

```
Epoch: 099, Train Loss: 0.0003965, Test Loss: 0.0003481
Epoch: 100, Train Loss: 0.0003893, Test Loss: 0.0003414

# ################## Final Prediction ##################
x_test = torch.unsqueeze(torch.linspace(0, 1, 1999), dim=1)
y_test = model(x_test)

# Plot the results
plt.plot(x[0:N_total], y[0:N_total], 'bo', label='True data')
plt.plot(x_test, y_test.detach().numpy(), 'r', label='Model
prediction')
plt.legend()
plt.show()
```



Problem 4.2 Determine appropriate mini-batch size = 16, epoch = 100

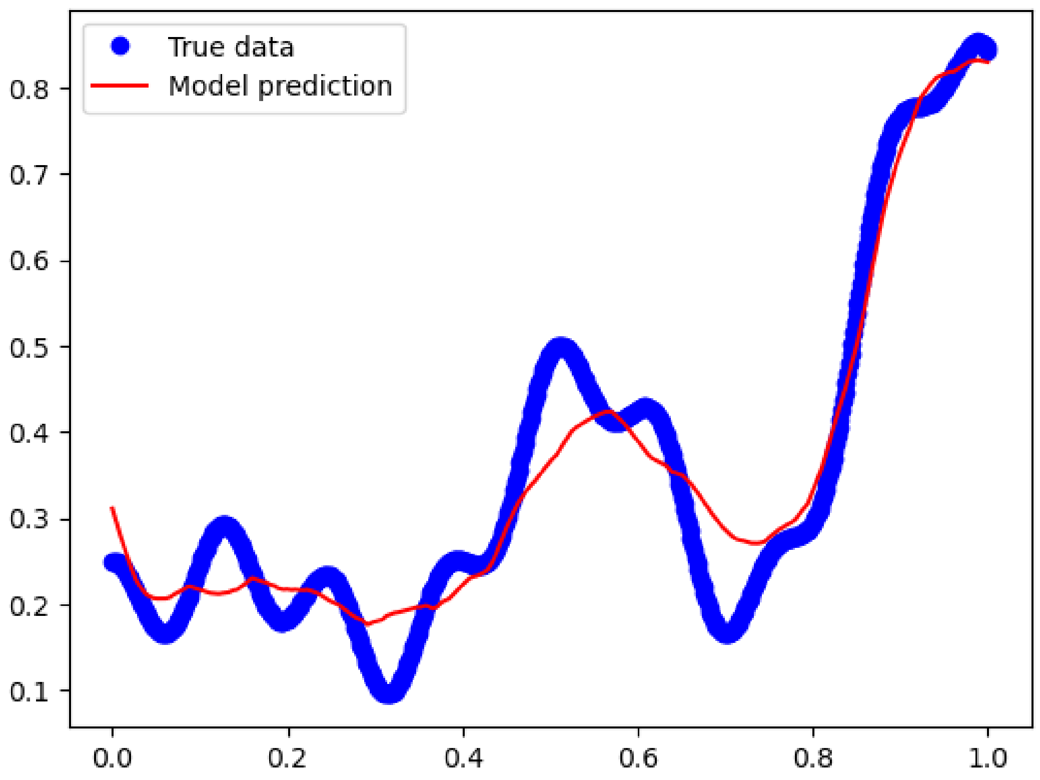
```
# ################## Part 1: Load data and create batch
##################
N_total = 600
N_train = 500
x = torch.unsqueeze(torch.linspace(0, 1, N_total), dim=1)
r = torch.randperm(N_total)
x = x[r, :]
y = 0.2 + 0.4 * torch.pow(x, 2) + 0.3 * x * torch.sin(15 * x) + 0.05 *
torch.cos(50 * x)
```

```
class CustomDataset(Dataset):
    def __init__(self, x, y):
        self.y = y
        self.x = x

    def __len__(self):
        return len(self.y)

    def __getitem__(self, idx):
        y1 = self.y[idx]
        x1 = self.x[idx]
        return (x1, y1)

# Change batch_size here to test different values
batch_size = 128  # Experiment with different batch sizes: 32, 64, 128
trainset = CustomDataset(x[0:N_train, :], y[0:N_train, :])
testset = CustomDataset(x[N_train:N_total, :], y[N_train:N_total, :])
train_loader = torch.utils.data.DataLoader(trainset,
batch_size=batch_size)
test_loader = torch.utils.data.DataLoader(testset,
batch_size=batch_size)

# ################## Part 2: Define Model ##################
model = nn.Sequential(
    nn.Linear(1, 1024, bias=True),
    nn.ReLU(),
    nn.Linear(1024, 1, bias=True)
)

def init_weights(m):
    if isinstance(m, nn.Linear):
        m.weight.data.uniform_(-1, 1)
        m.bias.data.uniform_(-1, 1)

model.apply(init_weights)

# ################## Part 3: Define Loss and Optimizer
##################
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

# ################## Part 4: Training and Testing ##################
def train_NN():
    model.train()
    for images, labels in train_loader:
        out = model(images)
        loss = criterion(out, labels)
        loss.backward()
        optimizer.step()
        optimizer.zero_grad()
    return loss

def test_NN(loader):
    model.eval()
    loss = 0
    with torch.no_grad():
        for images, labels in loader:
            out = model(images)
            loss += criterion(out, labels).item()
    loss = loss / len(loader)
    return loss

# Experiment with different numbers of epochs
N_epoch = 100
train_loss = np.zeros((N_epoch, 1))
test_loss = np.zeros((N_epoch, 1))

for epoch in range(N_epoch):
    train_NN()
    train_loss[epoch, 0] = test_NN(train_loader)
    test_loss[epoch, 0] = test_NN(test_loader)
    print(f'Epoch: {epoch+1:03d}, Train Loss: {train_loss[epoch,
0]:.7f}, Test Loss: {test_loss[epoch, 0]:.7f}')

# ################## Final Prediction ##################
x_test = torch.unsqueeze(torch.linspace(0, 1, 1999), dim=1)
y_test = model(x_test)

# Plot the results
plt.plot(x[0:N_total], y[0:N_total], 'bo', label='True data')
plt.plot(x_test, y_test.detach().numpy(), 'r', label='Model
prediction')
plt.legend()
plt.show()
```

```
Epoch: 001, Train Loss: 20.4632916, Test Loss: 21.1496086
Epoch: 002, Train Loss: 11.2453420, Test Loss: 12.3400145
Epoch: 003, Train Loss: 9.9087901, Test Loss: 11.2133293
Epoch: 004, Train Loss: 9.3603978, Test Loss: 10.5421286
Epoch: 005, Train Loss: 7.1279855, Test Loss: 8.0017729
Epoch: 006, Train Loss: 4.5086163, Test Loss: 5.0775385
Epoch: 007, Train Loss: 2.8985459, Test Loss: 3.2626579
Epoch: 008, Train Loss: 2.2327487, Test Loss: 2.4915483
Epoch: 009, Train Loss: 1.7291094, Test Loss: 1.9289263
Epoch: 010, Train Loss: 1.1206206, Test Loss: 1.2632594
Epoch: 011, Train Loss: 0.6755160, Test Loss: 0.7543423
Epoch: 012, Train Loss: 0.5096188, Test Loss: 0.5315038
Epoch: 013, Train Loss: 0.4424562, Test Loss: 0.4302731
Epoch: 014, Train Loss: 0.3534248, Test Loss: 0.3353279
```

```
Epoch: 015, Train Loss: 0.2882117, Test Loss: 0.2828371
Epoch: 016, Train Loss: 0.2728086, Test Loss: 0.2817203
Epoch: 017, Train Loss: 0.2624435, Test Loss: 0.2744622
Epoch: 018, Train Loss: 0.2374812, Test Loss: 0.2411609
Epoch: 019, Train Loss: 0.2158103, Test Loss: 0.2082577
Epoch: 020, Train Loss: 0.2009166, Test Loss: 0.1872455
Epoch: 021, Train Loss: 0.1834664, Test Loss: 0.1709518
Epoch: 022, Train Loss: 0.1649070, Test Loss: 0.1576907
Epoch: 023, Train Loss: 0.1499047, Test Loss: 0.1474956
Epoch: 024, Train Loss: 0.1368052, Test Loss: 0.1360662
Epoch: 025, Train Loss: 0.1241553, Test Loss: 0.1224505
Epoch: 026, Train Loss: 0.1130936, Test Loss: 0.1098194
Epoch: 027, Train Loss: 0.1035068, Test Loss: 0.0996372
Epoch: 028, Train Loss: 0.0945654, Test Loss: 0.0913040
Epoch: 029, Train Loss: 0.0863846, Test Loss: 0.0841472
Epoch: 030, Train Loss: 0.0790432, Test Loss: 0.0774125
Epoch: 031, Train Loss: 0.0722813, Test Loss: 0.0705894
Epoch: 032, Train Loss: 0.0660765, Test Loss: 0.0639731
Epoch: 033, Train Loss: 0.0604625, Test Loss: 0.0580483
Epoch: 034, Train Loss: 0.0553455, Test Loss: 0.0529255
Epoch: 035, Train Loss: 0.0506900, Test Loss: 0.0484510
Epoch: 036, Train Loss: 0.0464819, Test Loss: 0.0444016
Epoch: 037, Train Loss: 0.0426712, Test Loss: 0.0406250
Epoch: 038, Train Loss: 0.0392197, Test Loss: 0.0371288
Epoch: 039, Train Loss: 0.0361013, Test Loss: 0.0339912
Epoch: 040, Train Loss: 0.0332790, Test Loss: 0.0312273
Epoch: 041, Train Loss: 0.0307252, Test Loss: 0.0287764
Epoch: 042, Train Loss: 0.0284149, Test Loss: 0.0265639
Epoch: 043, Train Loss: 0.0263250, Test Loss: 0.0245394
Epoch: 044, Train Loss: 0.0244344, Test Loss: 0.0226980
Epoch: 045, Train Loss: 0.0227231, Test Loss: 0.0210416
Epoch: 046, Train Loss: 0.0211722, Test Loss: 0.0195588
Epoch: 047, Train Loss: 0.0197658, Test Loss: 0.0182223
Epoch: 048, Train Loss: 0.0184888, Test Loss: 0.0170080
Epoch: 049, Train Loss: 0.0173279, Test Loss: 0.0159006
Epoch: 050, Train Loss: 0.0162708, Test Loss: 0.0148928
Epoch: 051, Train Loss: 0.0153068, Test Loss: 0.0139772
Epoch: 052, Train Loss: 0.0144259, Test Loss: 0.0131443
Epoch: 053, Train Loss: 0.0136195, Test Loss: 0.0123831
Epoch: 054, Train Loss: 0.0128795, Test Loss: 0.0116848
Epoch: 055, Train Loss: 0.0121990, Test Loss: 0.0110430
Epoch: 056, Train Loss: 0.0115725, Test Loss: 0.0104528
Epoch: 057, Train Loss: 0.0109943, Test Loss: 0.0099096
Epoch: 058, Train Loss: 0.0104592, Test Loss: 0.0094082
Epoch: 059, Train Loss: 0.0099631, Test Loss: 0.0089443
Epoch: 060, Train Loss: 0.0095024, Test Loss: 0.0085143
Epoch: 061, Train Loss: 0.0090736, Test Loss: 0.0081151
Epoch: 062, Train Loss: 0.0086732, Test Loss: 0.0077439
Epoch: 063, Train Loss: 0.0082994, Test Loss: 0.0073980
```

```
Epoch: 064, Train Loss: 0.0079496, Test Loss: 0.0070753
Epoch: 065, Train Loss: 0.0076216, Test Loss: 0.0067739
Epoch: 066, Train Loss: 0.0073139, Test Loss: 0.0064919
Epoch: 067, Train Loss: 0.0070245, Test Loss: 0.0062282
Epoch: 068, Train Loss: 0.0067519, Test Loss: 0.0059808
Epoch: 069, Train Loss: 0.0064954, Test Loss: 0.0057490
Epoch: 070, Train Loss: 0.0062535, Test Loss: 0.0055315
Epoch: 071, Train Loss: 0.0060251, Test Loss: 0.0053267
Epoch: 072, Train Loss: 0.0058093, Test Loss: 0.0051344
Epoch: 073, Train Loss: 0.0056051, Test Loss: 0.0049535
Epoch: 074, Train Loss: 0.0054119, Test Loss: 0.0047829
Epoch: 075, Train Loss: 0.0052291, Test Loss: 0.0046220
Epoch: 076, Train Loss: 0.0050560, Test Loss: 0.0044699
Epoch: 077, Train Loss: 0.0048918, Test Loss: 0.0043266
Epoch: 078, Train Loss: 0.0047361, Test Loss: 0.0041909
Epoch: 079, Train Loss: 0.0045884, Test Loss: 0.0040627
Epoch: 080, Train Loss: 0.0044482, Test Loss: 0.0039412
Epoch: 081, Train Loss: 0.0043151, Test Loss: 0.0038265
Epoch: 082, Train Loss: 0.0041884, Test Loss: 0.0037177
Epoch: 083, Train Loss: 0.0040681, Test Loss: 0.0036145
Epoch: 084, Train Loss: 0.0039535, Test Loss: 0.0035166
Epoch: 085, Train Loss: 0.0038444, Test Loss: 0.0034235
Epoch: 086, Train Loss: 0.0037405, Test Loss: 0.0033353
Epoch: 087, Train Loss: 0.0036414, Test Loss: 0.0032513
Epoch: 088, Train Loss: 0.0035469, Test Loss: 0.0031714
Epoch: 089, Train Loss: 0.0034568, Test Loss: 0.0030951
Epoch: 090, Train Loss: 0.0033708, Test Loss: 0.0030225
Epoch: 091, Train Loss: 0.0032886, Test Loss: 0.0029533
Epoch: 092, Train Loss: 0.0032101, Test Loss: 0.0028868
Epoch: 093, Train Loss: 0.0031351, Test Loss: 0.0028231
Epoch: 094, Train Loss: 0.0030633, Test Loss: 0.0027623
Epoch: 095, Train Loss: 0.0029947, Test Loss: 0.0027041
Epoch: 096, Train Loss: 0.0029288, Test Loss: 0.0026480
Epoch: 097, Train Loss: 0.0028657, Test Loss: 0.0025943
Epoch: 098, Train Loss: 0.0028051, Test Loss: 0.0025427
Epoch: 099, Train Loss: 0.0027470, Test Loss: 0.0024932
Epoch: 100, Train Loss: 0.0026913, Test Loss: 0.0024457
```



Problem 4.2 Determine appropriate mini-batch size = 64, epoch = 100

```
# ################## Part 1: Load data and create batch
##################
N_total = 600
N_train = 500
x = torch.unsqueeze(torch.linspace(0, 1, N_total), dim=1)
r = torch.randperm(N_total)
x = x[r, :]
y = 0.2 + 0.4 * torch.pow(x, 2) + 0.3 * x * torch.sin(15 * x) + 0.05 *
torch.cos(50 * x)

class CustomDataset(Dataset):
    def __init__(self, x, y):
        self.y = y
        self.x = x

    def __len__(self):
        return len(self.y)

    def __getitem__(self, idx):
        y1 = self.y[idx]
        x1 = self.x[idx]
        return (x1, y1)

# Change batch_size here to test different values
```

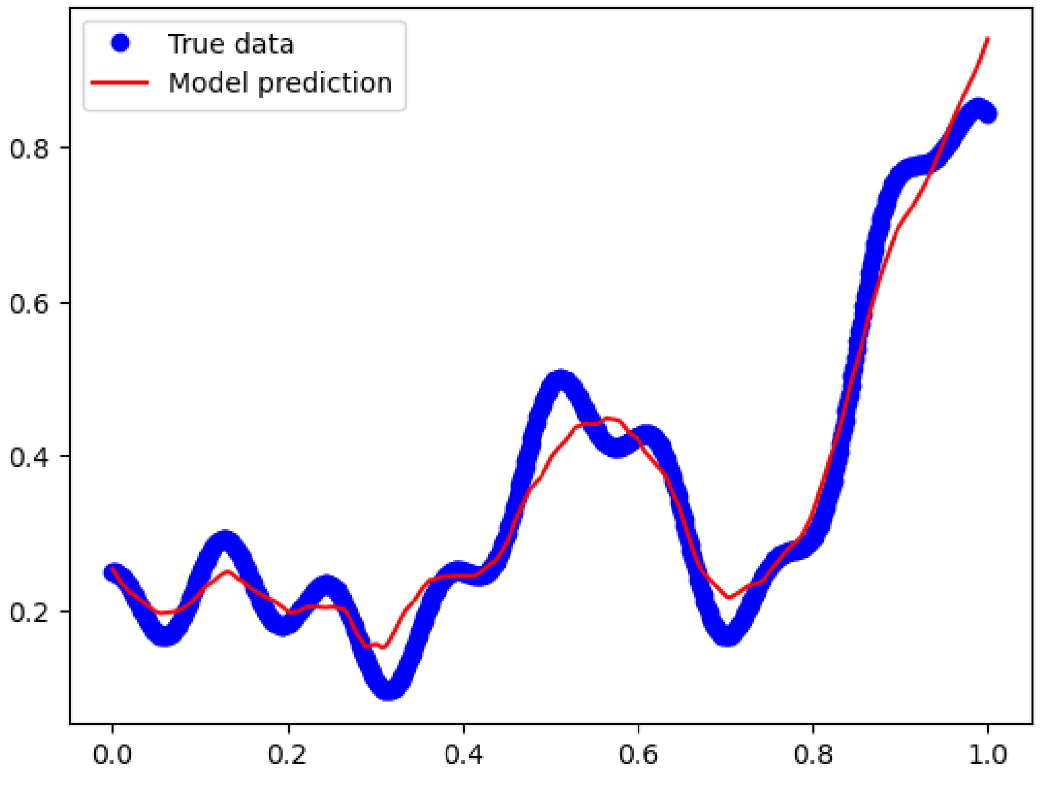
```python
batch_size = 64  # Experiment with different batch sizes: 32, 64, 128
trainset = CustomDataset(x[0:N_train, :], y[0:N_train, :])
testset = CustomDataset(x[N_train:N_total, :], y[N_train:N_total, :])
train_loader = torch.utils.data.DataLoader(trainset,
batch_size=batch_size)
test_loader = torch.utils.data.DataLoader(testset,
batch_size=batch_size)

# ################## Part 2: Define Model ##################
model = nn.Sequential(
    nn.Linear(1, 1024, bias=True),
    nn.ReLU(),
    nn.Linear(1024, 1, bias=True)
)

def init_weights(m):
    if isinstance(m, nn.Linear):
        m.weight.data.uniform_(-1, 1)
        m.bias.data.uniform_(-1, 1)

model.apply(init_weights)

# ################## Part 3: Define Loss and Optimizer
##################
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

# ################## Part 4: Training and Testing ##################
def train_NN():
    model.train()
    for images, labels in train_loader:
        out = model(images)
        loss = criterion(out, labels)
        loss.backward()
        optimizer.step()
        optimizer.zero_grad()
    return loss

def test_NN(loader):
    model.eval()
    loss = 0
    with torch.no_grad():
        for images, labels in loader:
            out = model(images)
            loss += criterion(out, labels).item()
    loss = loss / len(loader)
    return loss

# Experiment with different numbers of epochs
N_epoch = 100
```

```
train_loss = np.zeros((N_epoch, 1))
test_loss = np.zeros((N_epoch, 1))

for epoch in range(N_epoch):
    train_NN()
    train_loss[epoch, 0] = test_NN(train_loader)
    test_loss[epoch, 0] = test_NN(test_loader)
    print(f'Epoch: {epoch+1:03d}, Train Loss: {train_loss[epoch,
0]:.7f}, Test Loss: {test_loss[epoch, 0]:.7f}')

# ################## Final Prediction ##################
x_test = torch.unsqueeze(torch.linspace(0, 1, 1999), dim=1)
y_test = model(x_test)

# Plot the results
plt.plot(x[0:N_total], y[0:N_total], 'bo', label='True data')
plt.plot(x_test, y_test.detach().numpy(), 'r', label='Model
prediction')
plt.legend()
plt.show()
```

```
Epoch: 001, Train Loss: 15.6746806, Test Loss: 16.2465343
Epoch: 002, Train Loss: 2.8316783, Test Loss: 2.5567751
Epoch: 003, Train Loss: 5.4225901, Test Loss: 4.9618430
Epoch: 004, Train Loss: 2.1876323, Test Loss: 1.9957126
Epoch: 005, Train Loss: 0.6743857, Test Loss: 0.7711622
Epoch: 006, Train Loss: 0.7423441, Test Loss: 0.9157936
Epoch: 007, Train Loss: 0.2941593, Test Loss: 0.3851405
Epoch: 008, Train Loss: 0.2239520, Test Loss: 0.2369578
Epoch: 009, Train Loss: 0.1893579, Test Loss: 0.1959976
Epoch: 010, Train Loss: 0.1461559, Test Loss: 0.1745133
Epoch: 011, Train Loss: 0.1372006, Test Loss: 0.1686388
Epoch: 012, Train Loss: 0.1171399, Test Loss: 0.1371895
Epoch: 013, Train Loss: 0.1056668, Test Loss: 0.1206233
Epoch: 014, Train Loss: 0.0933011, Test Loss: 0.1111685
Epoch: 015, Train Loss: 0.0835575, Test Loss: 0.1031929
Epoch: 016, Train Loss: 0.0746684, Test Loss: 0.0925782
Epoch: 017, Train Loss: 0.0670494, Test Loss: 0.0831618
Epoch: 018, Train Loss: 0.0603105, Test Loss: 0.0757660
Epoch: 019, Train Loss: 0.0544268, Test Loss: 0.0691215
Epoch: 020, Train Loss: 0.0492410, Test Loss: 0.0627212
Epoch: 021, Train Loss: 0.0446951, Test Loss: 0.0570725
Epoch: 022, Train Loss: 0.0406946, Test Loss: 0.0522399
Epoch: 023, Train Loss: 0.0371691, Test Loss: 0.0479210
Epoch: 024, Train Loss: 0.0340490, Test Loss: 0.0440055
Epoch: 025, Train Loss: 0.0312833, Test Loss: 0.0405369
Epoch: 026, Train Loss: 0.0288218, Test Loss: 0.0374416
Epoch: 027, Train Loss: 0.0266230, Test Loss: 0.0346435
Epoch: 028, Train Loss: 0.0246500, Test Loss: 0.0321161
Epoch: 029, Train Loss: 0.0228736, Test Loss: 0.0298368
```

```
Epoch: 030, Train Loss: 0.0212682, Test Loss: 0.0277690
Epoch: 031, Train Loss: 0.0198125, Test Loss: 0.0258874
Epoch: 032, Train Loss: 0.0184883, Test Loss: 0.0241738
Epoch: 033, Train Loss: 0.0172810, Test Loss: 0.0226092
Epoch: 034, Train Loss: 0.0161777, Test Loss: 0.0211761
Epoch: 035, Train Loss: 0.0151688, Test Loss: 0.0198630
Epoch: 036, Train Loss: 0.0142440, Test Loss: 0.0186585
Epoch: 037, Train Loss: 0.0133935, Test Loss: 0.0175488
Epoch: 038, Train Loss: 0.0126108, Test Loss: 0.0165297
Epoch: 039, Train Loss: 0.0118898, Test Loss: 0.0155915
Epoch: 040, Train Loss: 0.0112241, Test Loss: 0.0147275
Epoch: 041, Train Loss: 0.0106093, Test Loss: 0.0139282
Epoch: 042, Train Loss: 0.0100410, Test Loss: 0.0131867
Epoch: 043, Train Loss: 0.0095152, Test Loss: 0.0124994
Epoch: 044, Train Loss: 0.0090282, Test Loss: 0.0118601
Epoch: 045, Train Loss: 0.0085765, Test Loss: 0.0112658
Epoch: 046, Train Loss: 0.0081566, Test Loss: 0.0107125
Epoch: 047, Train Loss: 0.0077662, Test Loss: 0.0101956
Epoch: 048, Train Loss: 0.0074025, Test Loss: 0.0097150
Epoch: 049, Train Loss: 0.0070629, Test Loss: 0.0092660
Epoch: 050, Train Loss: 0.0067456, Test Loss: 0.0088456
Epoch: 051, Train Loss: 0.0064488, Test Loss: 0.0084519
Epoch: 052, Train Loss: 0.0061710, Test Loss: 0.0080831
Epoch: 053, Train Loss: 0.0059105, Test Loss: 0.0077361
Epoch: 054, Train Loss: 0.0056661, Test Loss: 0.0074098
Epoch: 055, Train Loss: 0.0054362, Test Loss: 0.0071021
Epoch: 056, Train Loss: 0.0052199, Test Loss: 0.0068117
Epoch: 057, Train Loss: 0.0050157, Test Loss: 0.0065380
Epoch: 058, Train Loss: 0.0048233, Test Loss: 0.0062797
Epoch: 059, Train Loss: 0.0046413, Test Loss: 0.0060352
Epoch: 060, Train Loss: 0.0044693, Test Loss: 0.0058043
Epoch: 061, Train Loss: 0.0043063, Test Loss: 0.0055854
Epoch: 062, Train Loss: 0.0041516, Test Loss: 0.0053775
Epoch: 063, Train Loss: 0.0040047, Test Loss: 0.0051804
Epoch: 064, Train Loss: 0.0038651, Test Loss: 0.0049924
Epoch: 065, Train Loss: 0.0037326, Test Loss: 0.0048135
Epoch: 066, Train Loss: 0.0036064, Test Loss: 0.0046423
Epoch: 067, Train Loss: 0.0034859, Test Loss: 0.0044787
Epoch: 068, Train Loss: 0.0033712, Test Loss: 0.0043229
Epoch: 069, Train Loss: 0.0032616, Test Loss: 0.0041740
Epoch: 070, Train Loss: 0.0031568, Test Loss: 0.0040319
Epoch: 071, Train Loss: 0.0030566, Test Loss: 0.0038962
Epoch: 072, Train Loss: 0.0029608, Test Loss: 0.0037660
Epoch: 073, Train Loss: 0.0028691, Test Loss: 0.0036417
Epoch: 074, Train Loss: 0.0027810, Test Loss: 0.0035225
Epoch: 075, Train Loss: 0.0026965, Test Loss: 0.0034083
Epoch: 076, Train Loss: 0.0026155, Test Loss: 0.0032989
Epoch: 077, Train Loss: 0.0025379, Test Loss: 0.0031936
Epoch: 078, Train Loss: 0.0024634, Test Loss: 0.0030926
```

```
Epoch: 079, Train Loss: 0.0023919, Test Loss: 0.0029952
Epoch: 080, Train Loss: 0.0023234, Test Loss: 0.0029022
Epoch: 081, Train Loss: 0.0022573, Test Loss: 0.0028124
Epoch: 082, Train Loss: 0.0021938, Test Loss: 0.0027265
Epoch: 083, Train Loss: 0.0021328, Test Loss: 0.0026439
Epoch: 084, Train Loss: 0.0020741, Test Loss: 0.0025645
Epoch: 085, Train Loss: 0.0020177, Test Loss: 0.0024882
Epoch: 086, Train Loss: 0.0019632, Test Loss: 0.0024149
Epoch: 087, Train Loss: 0.0019108, Test Loss: 0.0023444
Epoch: 088, Train Loss: 0.0018605, Test Loss: 0.0022767
Epoch: 089, Train Loss: 0.0018121, Test Loss: 0.0022114
Epoch: 090, Train Loss: 0.0017654, Test Loss: 0.0021486
Epoch: 091, Train Loss: 0.0017204, Test Loss: 0.0020882
Epoch: 092, Train Loss: 0.0016771, Test Loss: 0.0020299
Epoch: 093, Train Loss: 0.0016355, Test Loss: 0.0019737
Epoch: 094, Train Loss: 0.0015954, Test Loss: 0.0019196
Epoch: 095, Train Loss: 0.0015568, Test Loss: 0.0018675
Epoch: 096, Train Loss: 0.0015194, Test Loss: 0.0018172
Epoch: 097, Train Loss: 0.0014834, Test Loss: 0.0017688
Epoch: 098, Train Loss: 0.0014487, Test Loss: 0.0017224
Epoch: 099, Train Loss: 0.0014151, Test Loss: 0.0016777
Epoch: 100, Train Loss: 0.0013827, Test Loss: 0.0016345
```



Problem 4.2 Determine appropriate mini-batch size = 128, epoch = 100

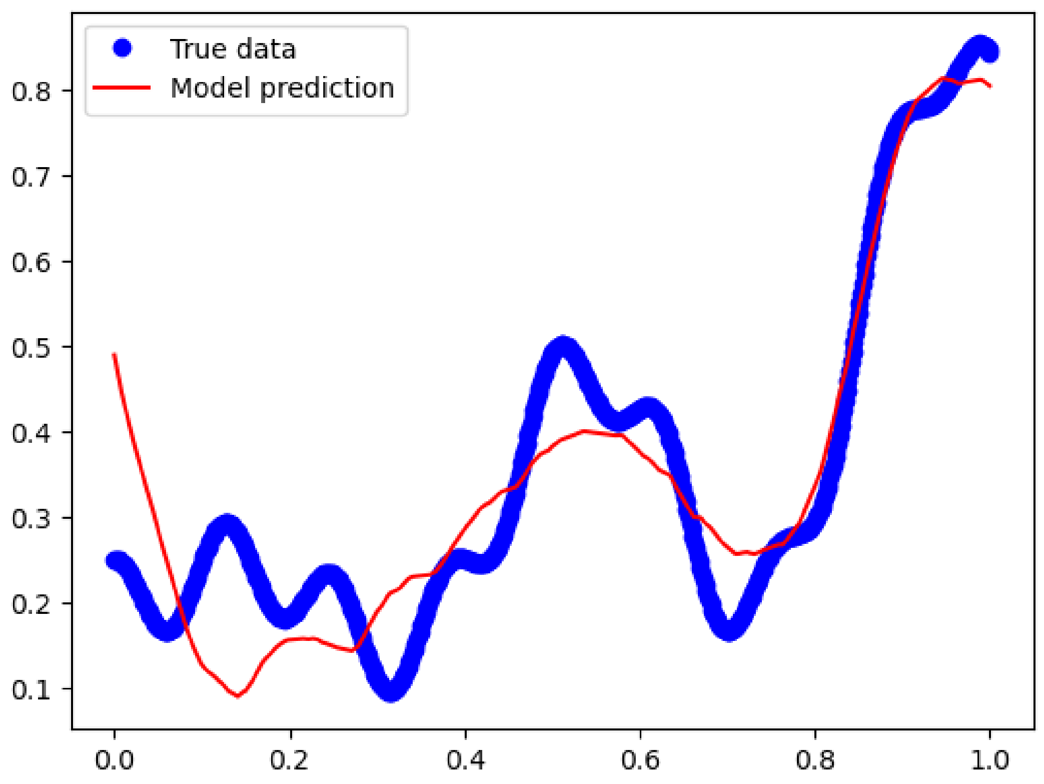
```
# ################## Part 1: Load data and create batch
##################
N_total = 600
N_train = 500
x = torch.unsqueeze(torch.linspace(0, 1, N_total), dim=1)
r = torch.randperm(N_total)
x = x[r, :]
y = 0.2 + 0.4 * torch.pow(x, 2) + 0.3 * x * torch.sin(15 * x) + 0.05 *
torch.cos(50 * x)

class CustomDataset(Dataset):
    def __init__(self, x, y):
        self.y = y
        self.x = x

    def __len__(self):
        return len(self.y)

    def __getitem__(self, idx):
        y1 = self.y[idx]
        x1 = self.x[idx]
        return (x1, y1)

# Change batch_size here to test different values
batch_size = 128  # Experiment with different batch sizes: 32, 64, 128
trainset = CustomDataset(x[0:N_train, :], y[0:N_train, :])
testset = CustomDataset(x[N_train:N_total, :], y[N_train:N_total, :])
train_loader = torch.utils.data.DataLoader(trainset,
batch_size=batch_size)
test_loader = torch.utils.data.DataLoader(testset,
batch_size=batch_size)

# ################## Part 2: Define Model ##################
model = nn.Sequential(
    nn.Linear(1, 1024, bias=True),
    nn.ReLU(),
    nn.Linear(1024, 1, bias=True)
)

def init_weights(m):
    if isinstance(m, nn.Linear):
        m.weight.data.uniform_(-1, 1)
        m.bias.data.uniform_(-1, 1)

model.apply(init_weights)

# ################## Part 3: Define Loss and Optimizer
##################
criterion = nn.MSELoss()
```

```
optimizer = optim.Adam(model.parameters(), lr=0.001)

# ################## Part 4: Training and Testing ##################
def train_NN():
    model.train()
    for images, labels in train_loader:
        out = model(images)
        loss = criterion(out, labels)
        loss.backward()
        optimizer.step()
        optimizer.zero_grad()
    return loss

def test_NN(loader):
    model.eval()
    loss = 0
    with torch.no_grad():
        for images, labels in loader:
            out = model(images)
            loss += criterion(out, labels).item()
    loss = loss / len(loader)
    return loss

# Experiment with different numbers of epochs
N_epoch = 100
train_loss = np.zeros((N_epoch, 1))
test_loss = np.zeros((N_epoch, 1))

for epoch in range(N_epoch):
    train_NN()
    train_loss[epoch, 0] = test_NN(train_loader)
    test_loss[epoch, 0] = test_NN(test_loader)
    print(f'Epoch: {epoch+1:03d}, Train Loss: {train_loss[epoch,
0]:.7f}, Test Loss: {test_loss[epoch, 0]:.7f}')

# ################## Final Prediction ##################
x_test = torch.unsqueeze(torch.linspace(0, 1, 1999), dim=1)
y_test = model(x_test)

# Plot the results
plt.plot(x[0:N_total], y[0:N_total], 'bo', label='True data')
plt.plot(x_test, y_test.detach().numpy(), 'r', label='Model
prediction')
plt.legend()
plt.show()
```

```
Epoch: 001, Train Loss: 22.4205217, Test Loss: 19.2805557
Epoch: 002, Train Loss: 5.3662879, Test Loss: 4.1010160
Epoch: 003, Train Loss: 0.8423916, Test Loss: 0.5674591
Epoch: 004, Train Loss: 2.5692085, Test Loss: 2.6380594
```

```
Epoch: 005, Train Loss: 4.0418289, Test Loss: 4.1497045
Epoch: 006, Train Loss: 3.1026638, Test Loss: 3.1714888
Epoch: 007, Train Loss: 1.2745357, Test Loss: 1.2814338
Epoch: 008, Train Loss: 0.2924522, Test Loss: 0.2166031
Epoch: 009, Train Loss: 0.3348801, Test Loss: 0.1885524
Epoch: 010, Train Loss: 0.5696861, Test Loss: 0.4075289
Epoch: 011, Train Loss: 0.4670437, Test Loss: 0.3442137
Epoch: 012, Train Loss: 0.1974116, Test Loss: 0.1294619
Epoch: 013, Train Loss: 0.0737167, Test Loss: 0.0414731
Epoch: 014, Train Loss: 0.1056979, Test Loss: 0.0857564
Epoch: 015, Train Loss: 0.1311933, Test Loss: 0.1138015
Epoch: 016, Train Loss: 0.0954354, Test Loss: 0.0802353
Epoch: 017, Train Loss: 0.0599697, Test Loss: 0.0470680
Epoch: 018, Train Loss: 0.0591872, Test Loss: 0.0475546
Epoch: 019, Train Loss: 0.0664397, Test Loss: 0.0557374
Epoch: 020, Train Loss: 0.0606549, Test Loss: 0.0509934
Epoch: 021, Train Loss: 0.0517262, Test Loss: 0.0426373
Epoch: 022, Train Loss: 0.0496269, Test Loss: 0.0404446
Epoch: 023, Train Loss: 0.0497127, Test Loss: 0.0403316
Epoch: 024, Train Loss: 0.0471728, Test Loss: 0.0378564
Epoch: 025, Train Loss: 0.0441844, Test Loss: 0.0350393
Epoch: 026, Train Loss: 0.0427473, Test Loss: 0.0336998
Epoch: 027, Train Loss: 0.0416015, Test Loss: 0.0326296
Epoch: 028, Train Loss: 0.0398994, Test Loss: 0.0310387
Epoch: 029, Train Loss: 0.0382882, Test Loss: 0.0295686
Epoch: 030, Train Loss: 0.0370320, Test Loss: 0.0284997
Epoch: 031, Train Loss: 0.0357766, Test Loss: 0.0275182
Epoch: 032, Train Loss: 0.0344608, Test Loss: 0.0265612
Epoch: 033, Train Loss: 0.0332472, Test Loss: 0.0257453
Epoch: 034, Train Loss: 0.0321139, Test Loss: 0.0249952
Epoch: 035, Train Loss: 0.0309853, Test Loss: 0.0242101
Epoch: 036, Train Loss: 0.0298918, Test Loss: 0.0234216
Epoch: 037, Train Loss: 0.0288579, Test Loss: 0.0226794
Epoch: 038, Train Loss: 0.0278647, Test Loss: 0.0219856
Epoch: 039, Train Loss: 0.0269045, Test Loss: 0.0213372
Epoch: 040, Train Loss: 0.0259856, Test Loss: 0.0207238
Epoch: 041, Train Loss: 0.0251032, Test Loss: 0.0201202
Epoch: 042, Train Loss: 0.0242530, Test Loss: 0.0195181
Epoch: 043, Train Loss: 0.0234370, Test Loss: 0.0189339
Epoch: 044, Train Loss: 0.0226565, Test Loss: 0.0183753
Epoch: 045, Train Loss: 0.0219085, Test Loss: 0.0178479
Epoch: 046, Train Loss: 0.0211923, Test Loss: 0.0173505
Epoch: 047, Train Loss: 0.0205068, Test Loss: 0.0168726
Epoch: 048, Train Loss: 0.0198505, Test Loss: 0.0164079
Epoch: 049, Train Loss: 0.0192220, Test Loss: 0.0159580
Epoch: 050, Train Loss: 0.0186207, Test Loss: 0.0155293
Epoch: 051, Train Loss: 0.0180448, Test Loss: 0.0151263
Epoch: 052, Train Loss: 0.0174940, Test Loss: 0.0147447
Epoch: 053, Train Loss: 0.0169666, Test Loss: 0.0143809
```

```
Epoch: 054, Train Loss: 0.0164621, Test Loss: 0.0140302
Epoch: 055, Train Loss: 0.0159792, Test Loss: 0.0136921
Epoch: 056, Train Loss: 0.0155167, Test Loss: 0.0133680
Epoch: 057, Train Loss: 0.0150744, Test Loss: 0.0130588
Epoch: 058, Train Loss: 0.0146510, Test Loss: 0.0127638
Epoch: 059, Train Loss: 0.0142451, Test Loss: 0.0124811
Epoch: 060, Train Loss: 0.0138567, Test Loss: 0.0122086
Epoch: 061, Train Loss: 0.0134850, Test Loss: 0.0119459
Epoch: 062, Train Loss: 0.0131284, Test Loss: 0.0116936
Epoch: 063, Train Loss: 0.0127868, Test Loss: 0.0114525
Epoch: 064, Train Loss: 0.0124591, Test Loss: 0.0112216
Epoch: 065, Train Loss: 0.0121444, Test Loss: 0.0110009
Epoch: 066, Train Loss: 0.0118426, Test Loss: 0.0107887
Epoch: 067, Train Loss: 0.0115530, Test Loss: 0.0105838
Epoch: 068, Train Loss: 0.0112750, Test Loss: 0.0103863
Epoch: 069, Train Loss: 0.0110078, Test Loss: 0.0101962
Epoch: 070, Train Loss: 0.0107508, Test Loss: 0.0100143
Epoch: 071, Train Loss: 0.0105033, Test Loss: 0.0098393
Epoch: 072, Train Loss: 0.0102650, Test Loss: 0.0096701
Epoch: 073, Train Loss: 0.0100356, Test Loss: 0.0095059
Epoch: 074, Train Loss: 0.0098149, Test Loss: 0.0093472
Epoch: 075, Train Loss: 0.0096021, Test Loss: 0.0091933
Epoch: 076, Train Loss: 0.0093970, Test Loss: 0.0090446
Epoch: 077, Train Loss: 0.0091991, Test Loss: 0.0089003
Epoch: 078, Train Loss: 0.0090083, Test Loss: 0.0087601
Epoch: 079, Train Loss: 0.0088242, Test Loss: 0.0086234
Epoch: 080, Train Loss: 0.0086462, Test Loss: 0.0084903
Epoch: 081, Train Loss: 0.0084740, Test Loss: 0.0083608
Epoch: 082, Train Loss: 0.0083074, Test Loss: 0.0082342
Epoch: 083, Train Loss: 0.0081463, Test Loss: 0.0081094
Epoch: 084, Train Loss: 0.0079903, Test Loss: 0.0079873
Epoch: 085, Train Loss: 0.0078389, Test Loss: 0.0078680
Epoch: 086, Train Loss: 0.0076922, Test Loss: 0.0077511
Epoch: 087, Train Loss: 0.0075497, Test Loss: 0.0076365
Epoch: 088, Train Loss: 0.0074113, Test Loss: 0.0075242
Epoch: 089, Train Loss: 0.0072772, Test Loss: 0.0074144
Epoch: 090, Train Loss: 0.0071467, Test Loss: 0.0073074
Epoch: 091, Train Loss: 0.0070196, Test Loss: 0.0072017
Epoch: 092, Train Loss: 0.0068958, Test Loss: 0.0070973
Epoch: 093, Train Loss: 0.0067752, Test Loss: 0.0069955
Epoch: 094, Train Loss: 0.0066577, Test Loss: 0.0068957
Epoch: 095, Train Loss: 0.0065431, Test Loss: 0.0067977
Epoch: 096, Train Loss: 0.0064315, Test Loss: 0.0067013
Epoch: 097, Train Loss: 0.0063226, Test Loss: 0.0066064
Epoch: 098, Train Loss: 0.0062163, Test Loss: 0.0065133
Epoch: 099, Train Loss: 0.0061123, Test Loss: 0.0064226
Epoch: 100, Train Loss: 0.0060108, Test Loss: 0.0063334
```



Determine appropriate mini-batch size = 32

```python
# Import dependencies
import numpy as np
import torch
import torchvision
from torch.utils.data.dataset import Dataset
from torchvision import datasets, transforms
from torch import nn, optim
import matplotlib.pyplot as plt
```

Problem 4.2 Determine appropriate mini-batch size = 32, epoch = 100

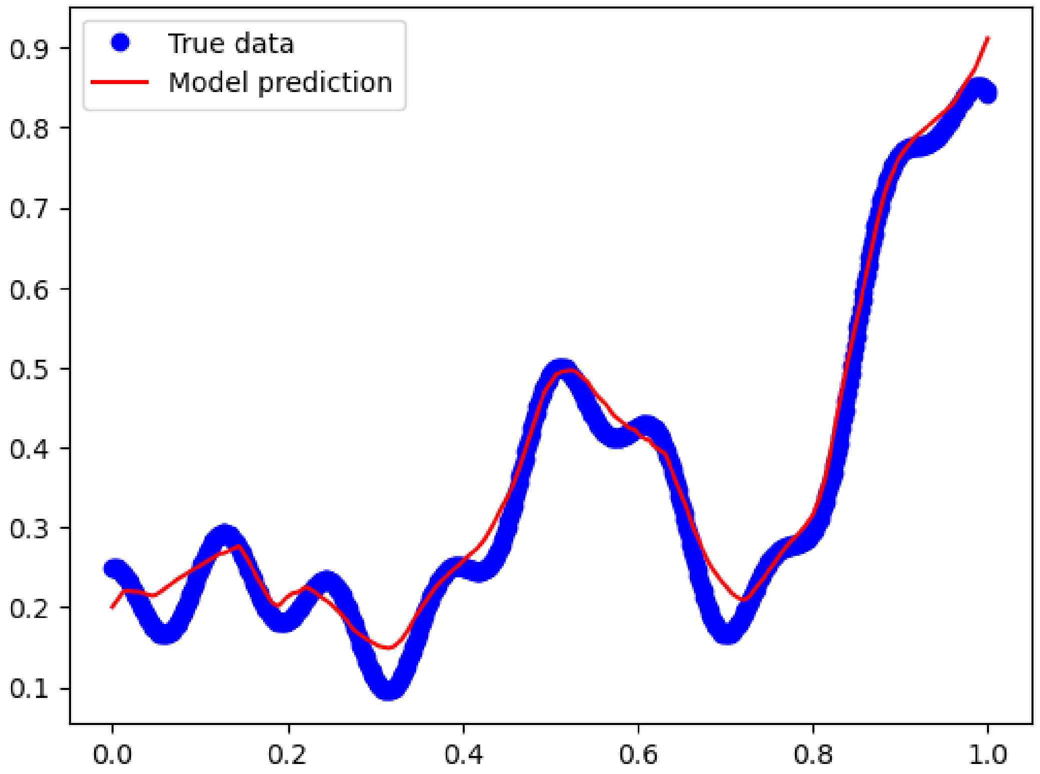
```python
# ################## Part 1: Load data and create batch
##################
N_total = 600
N_train = 500
x = torch.unsqueeze(torch.linspace(0, 1, N_total), dim=1)
r = torch.randperm(N_total)
x = x[r, :]
y = 0.2 + 0.4 * torch.pow(x, 2) + 0.3 * x * torch.sin(15 * x) + 0.05 *
torch.cos(50 * x)

class CustomDataset(Dataset):
    def __init__(self, x, y):
        self.y = y
        self.x = x

    def __len__(self):
        return len(self.y)

    def __getitem__(self, idx):
        y1 = self.y[idx]
        x1 = self.x[idx]
        return (x1, y1)

# Change batch_size here to test different values
batch_size = 32  # Experiment with different batch sizes: 32, 64, 128
trainset = CustomDataset(x[0:N_train, :], y[0:N_train, :])
testset = CustomDataset(x[N_train:N_total, :], y[N_train:N_total, :])
train_loader = torch.utils.data.DataLoader(trainset,
batch_size=batch_size)
test_loader = torch.utils.data.DataLoader(testset,
batch_size=batch_size)

# ################## Part 2: Define Model ##################
model = nn.Sequential(
    nn.Linear(1, 1024, bias=True),
    nn.ReLU(),
    nn.Linear(1024, 1, bias=True)
)
```

```
def init_weights(m):
    if isinstance(m, nn.Linear):
        m.weight.data.uniform_(-1, 1)
        m.bias.data.uniform_(-1, 1)

model.apply(init_weights)

Sequential(
  (0): Linear(in_features=1, out_features=1024, bias=True)
  (1): ReLU()
  (2): Linear(in_features=1024, out_features=1, bias=True)
)

# ################## Part 3: Define Loss and Optimizer
##################
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

# ################## Part 4: Training and Testing ##################
def train_NN():
    model.train()
    for images, labels in train_loader:
        out = model(images)
        loss = criterion(out, labels)
        loss.backward()
        optimizer.step()
        optimizer.zero_grad()
    return loss

def test_NN(loader):
    model.eval()
    loss = 0
    with torch.no_grad():
        for images, labels in loader:
            out = model(images)
            loss += criterion(out, labels).item()
    loss = loss / len(loader)
    return loss

# Experiment with different numbers of epochs
N_epoch = 100  # Try values like 100, 200, 500
train_loss = np.zeros((N_epoch, 1))
test_loss = np.zeros((N_epoch, 1))

for epoch in range(N_epoch):
    train_NN()
    train_loss[epoch, 0] = test_NN(train_loader)
    test_loss[epoch, 0] = test_NN(test_loader)
    print(f'Epoch: {epoch+1:03d}, Train Loss: {train_loss[epoch,
0]:.7f}, Test Loss: {test_loss[epoch, 0]:.7f}')
```

```
Epoch: 001, Train Loss: 2.4354299, Test Loss: 2.4755852
Epoch: 002, Train Loss: 3.9236410, Test Loss: 5.1143095
Epoch: 003, Train Loss: 0.5623158, Test Loss: 0.6476966
Epoch: 004, Train Loss: 0.2407783, Test Loss: 0.2889059
Epoch: 005, Train Loss: 0.1428621, Test Loss: 0.1905185
Epoch: 006, Train Loss: 0.0941955, Test Loss: 0.0943288
Epoch: 007, Train Loss: 0.0746617, Test Loss: 0.0724883
Epoch: 008, Train Loss: 0.0617504, Test Loss: 0.0571178
Epoch: 009, Train Loss: 0.0513076, Test Loss: 0.0466694
Epoch: 010, Train Loss: 0.0430132, Test Loss: 0.0386989
Epoch: 011, Train Loss: 0.0365885, Test Loss: 0.0324178
Epoch: 012, Train Loss: 0.0316494, Test Loss: 0.0277373
Epoch: 013, Train Loss: 0.0278367, Test Loss: 0.0241990
Epoch: 014, Train Loss: 0.0248465, Test Loss: 0.0215016
Epoch: 015, Train Loss: 0.0224383, Test Loss: 0.0193680
Epoch: 016, Train Loss: 0.0204374, Test Loss: 0.0176288
Epoch: 017, Train Loss: 0.0187305, Test Loss: 0.0161740
Epoch: 018, Train Loss: 0.0172499, Test Loss: 0.0149224
Epoch: 019, Train Loss: 0.0159318, Test Loss: 0.0138053
Epoch: 020, Train Loss: 0.0147235, Test Loss: 0.0127849
Epoch: 021, Train Loss: 0.0136354, Test Loss: 0.0118672
Epoch: 022, Train Loss: 0.0126465, Test Loss: 0.0110175
Epoch: 023, Train Loss: 0.0117301, Test Loss: 0.0102177
Epoch: 024, Train Loss: 0.0108768, Test Loss: 0.0094759
Epoch: 025, Train Loss: 0.0100872, Test Loss: 0.0087866
Epoch: 026, Train Loss: 0.0093559, Test Loss: 0.0081463
Epoch: 027, Train Loss: 0.0086779, Test Loss: 0.0075576
Epoch: 028, Train Loss: 0.0080505, Test Loss: 0.0070149
Epoch: 029, Train Loss: 0.0074707, Test Loss: 0.0065112
Epoch: 030, Train Loss: 0.0069354, Test Loss: 0.0060476
Epoch: 031, Train Loss: 0.0064402, Test Loss: 0.0056201
Epoch: 032, Train Loss: 0.0059825, Test Loss: 0.0052276
Epoch: 033, Train Loss: 0.0055611, Test Loss: 0.0048635
Epoch: 034, Train Loss: 0.0051722, Test Loss: 0.0045270
Epoch: 035, Train Loss: 0.0048144, Test Loss: 0.0042171
Epoch: 036, Train Loss: 0.0044843, Test Loss: 0.0039327
Epoch: 037, Train Loss: 0.0041800, Test Loss: 0.0036707
Epoch: 038, Train Loss: 0.0039007, Test Loss: 0.0034307
Epoch: 039, Train Loss: 0.0036431, Test Loss: 0.0032092
Epoch: 040, Train Loss: 0.0034060, Test Loss: 0.0030050
Epoch: 041, Train Loss: 0.0031879, Test Loss: 0.0028171
Epoch: 042, Train Loss: 0.0029874, Test Loss: 0.0026439
Epoch: 043, Train Loss: 0.0028028, Test Loss: 0.0024836
Epoch: 044, Train Loss: 0.0026331, Test Loss: 0.0023360
Epoch: 045, Train Loss: 0.0024772, Test Loss: 0.0021994
Epoch: 046, Train Loss: 0.0023339, Test Loss: 0.0020734
Epoch: 047, Train Loss: 0.0022023, Test Loss: 0.0019576
Epoch: 048, Train Loss: 0.0020812, Test Loss: 0.0018507
Epoch: 049, Train Loss: 0.0019697, Test Loss: 0.0017521
Epoch: 050, Train Loss: 0.0018672, Test Loss: 0.0016613
```

```
Epoch: 051, Train Loss: 0.0017732, Test Loss: 0.0015775
Epoch: 052, Train Loss: 0.0016867, Test Loss: 0.0015003
Epoch: 053, Train Loss: 0.0016074, Test Loss: 0.0014286
Epoch: 054, Train Loss: 0.0015344, Test Loss: 0.0013624
Epoch: 055, Train Loss: 0.0014672, Test Loss: 0.0013019
Epoch: 056, Train Loss: 0.0014049, Test Loss: 0.0012461
Epoch: 057, Train Loss: 0.0013476, Test Loss: 0.0011952
Epoch: 058, Train Loss: 0.0012948, Test Loss: 0.0011483
Epoch: 059, Train Loss: 0.0012462, Test Loss: 0.0011058
Epoch: 060, Train Loss: 0.0012013, Test Loss: 0.0010673
Epoch: 061, Train Loss: 0.0011601, Test Loss: 0.0010318
Epoch: 062, Train Loss: 0.0011223, Test Loss: 0.0009991
Epoch: 063, Train Loss: 0.0010875, Test Loss: 0.0009688
Epoch: 064, Train Loss: 0.0010555, Test Loss: 0.0009415
Epoch: 065, Train Loss: 0.0010264, Test Loss: 0.0009166
Epoch: 066, Train Loss: 0.0009998, Test Loss: 0.0008934
Epoch: 067, Train Loss: 0.0009755, Test Loss: 0.0008722
Epoch: 068, Train Loss: 0.0009533, Test Loss: 0.0008526
Epoch: 069, Train Loss: 0.0009333, Test Loss: 0.0008347
Epoch: 070, Train Loss: 0.0009149, Test Loss: 0.0008183
Epoch: 071, Train Loss: 0.0008983, Test Loss: 0.0008036
Epoch: 072, Train Loss: 0.0008834, Test Loss: 0.0007903
Epoch: 073, Train Loss: 0.0008702, Test Loss: 0.0007783
Epoch: 074, Train Loss: 0.0008586, Test Loss: 0.0007677
Epoch: 075, Train Loss: 0.0008485, Test Loss: 0.0007581
Epoch: 076, Train Loss: 0.0008397, Test Loss: 0.0007494
Epoch: 077, Train Loss: 0.0008321, Test Loss: 0.0007417
Epoch: 078, Train Loss: 0.0008260, Test Loss: 0.0007350
Epoch: 079, Train Loss: 0.0008210, Test Loss: 0.0007294
Epoch: 080, Train Loss: 0.0008174, Test Loss: 0.0007250
Epoch: 081, Train Loss: 0.0008148, Test Loss: 0.0007215
Epoch: 082, Train Loss: 0.0008130, Test Loss: 0.0007187
Epoch: 083, Train Loss: 0.0008119, Test Loss: 0.0007167
Epoch: 084, Train Loss: 0.0008114, Test Loss: 0.0007149
Epoch: 085, Train Loss: 0.0008116, Test Loss: 0.0007137
Epoch: 086, Train Loss: 0.0008123, Test Loss: 0.0007130
Epoch: 087, Train Loss: 0.0008134, Test Loss: 0.0007127
Epoch: 088, Train Loss: 0.0008145, Test Loss: 0.0007126
Epoch: 089, Train Loss: 0.0008154, Test Loss: 0.0007123
Epoch: 090, Train Loss: 0.0008156, Test Loss: 0.0007114
Epoch: 091, Train Loss: 0.0008154, Test Loss: 0.0007102
Epoch: 092, Train Loss: 0.0008145, Test Loss: 0.0007086
Epoch: 093, Train Loss: 0.0008130, Test Loss: 0.0007065
Epoch: 094, Train Loss: 0.0008103, Test Loss: 0.0007035
Epoch: 095, Train Loss: 0.0008066, Test Loss: 0.0006999
Epoch: 096, Train Loss: 0.0008016, Test Loss: 0.0006956
Epoch: 097, Train Loss: 0.0007956, Test Loss: 0.0006908
Epoch: 098, Train Loss: 0.0007884, Test Loss: 0.0006851
```

```
Epoch: 099, Train Loss: 0.0007800, Test Loss: 0.0006785
Epoch: 100, Train Loss: 0.0007706, Test Loss: 0.0006711

# ################## Final Prediction ##################
x_test = torch.unsqueeze(torch.linspace(0, 1, 1999), dim=1)
y_test = model(x_test)

# Plot the results
plt.plot(x[0:N_total], y[0:N_total], 'bo', label='True data')
plt.plot(x_test, y_test.detach().numpy(), 'r', label='Model
prediction')
plt.legend()
plt.show()
```



Problem 4.2 Determine appropriate mini-batch size = 32, epoch = 200

```
# ################## Part 1: Load data and create batch
##################
N_total = 600
N_train = 500
x = torch.unsqueeze(torch.linspace(0, 1, N_total), dim=1)
r = torch.randperm(N_total)
x = x[r, :]
y = 0.2 + 0.4 * torch.pow(x, 2) + 0.3 * x * torch.sin(15 * x) + 0.05 *
torch.cos(50 * x)
```

```python
class CustomDataset(Dataset):
    def __init__(self, x, y):
        self.y = y
        self.x = x

    def __len__(self):
        return len(self.y)

    def __getitem__(self, idx):
        y1 = self.y[idx]
        x1 = self.x[idx]
        return (x1, y1)

# Change batch_size here to test different values
batch_size = 32  # Experiment with different batch sizes: 32, 64, 128
trainset = CustomDataset(x[0:N_train, :], y[0:N_train, :])
testset = CustomDataset(x[N_train:N_total, :], y[N_train:N_total, :])
train_loader = torch.utils.data.DataLoader(trainset,
batch_size=batch_size)
test_loader = torch.utils.data.DataLoader(testset,
batch_size=batch_size)

# ################## Part 2: Define Model ##################
model = nn.Sequential(
    nn.Linear(1, 1024, bias=True),
    nn.ReLU(),
    nn.Linear(1024, 1, bias=True)
)

def init_weights(m):
    if isinstance(m, nn.Linear):
        m.weight.data.uniform_(-1, 1)
        m.bias.data.uniform_(-1, 1)

model.apply(init_weights)

# ################## Part 3: Define Loss and Optimizer
##################
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

# ################## Part 4: Training and Testing ##################
def train_NN():
    model.train()
    for images, labels in train_loader:
        out = model(images)
        loss = criterion(out, labels)
        loss.backward()
        optimizer.step()
        optimizer.zero_grad()
    return loss

def test_NN(loader):
    model.eval()
    loss = 0
    with torch.no_grad():
        for images, labels in loader:
            out = model(images)
            loss += criterion(out, labels).item()
    loss = loss / len(loader)
    return loss

# Experiment with different numbers of epochs
N_epoch = 200  # Try values like 100, 200, 500
train_loss = np.zeros((N_epoch, 1))
test_loss = np.zeros((N_epoch, 1))

for epoch in range(N_epoch):
    train_NN()
    train_loss[epoch, 0] = test_NN(train_loader)
    test_loss[epoch, 0] = test_NN(test_loader)
    print(f'Epoch: {epoch+1:03d}, Train Loss: {train_loss[epoch,
0]:.7f}, Test Loss: {test_loss[epoch, 0]:.7f}')

# ################## Final Prediction ##################
x_test = torch.unsqueeze(torch.linspace(0, 1, 1999), dim=1)
y_test = model(x_test)

# Plot the results
plt.plot(x[0:N_total], y[0:N_total], 'bo', label='True data')
plt.plot(x_test, y_test.detach().numpy(), 'r', label='Model
prediction')
plt.legend()
plt.show()
```

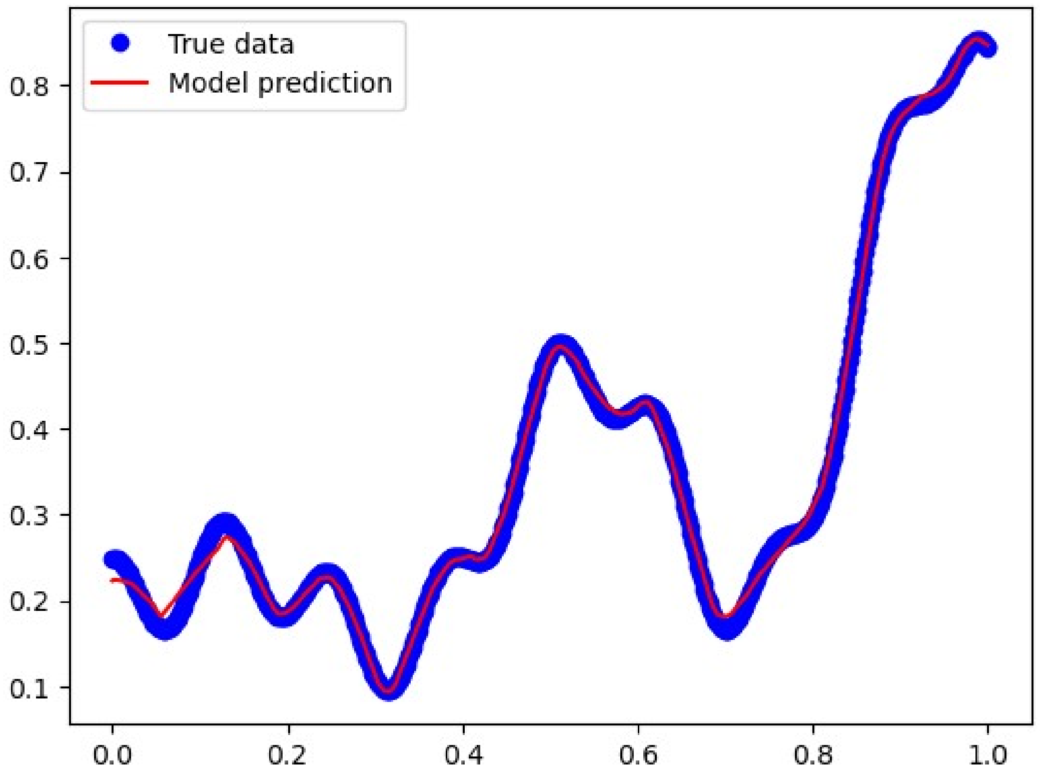
```
Epoch: 001, Train Loss: 2.0267818, Test Loss: 2.1714951
Epoch: 002, Train Loss: 3.1269895, Test Loss: 3.0556849
Epoch: 003, Train Loss: 0.4870037, Test Loss: 0.5017008
Epoch: 004, Train Loss: 0.1158632, Test Loss: 0.1175885
Epoch: 005, Train Loss: 0.0763223, Test Loss: 0.0932067
Epoch: 006, Train Loss: 0.0526999, Test Loss: 0.0570447
Epoch: 007, Train Loss: 0.0422577, Test Loss: 0.0513590
Epoch: 008, Train Loss: 0.0347963, Test Loss: 0.0403812
Epoch: 009, Train Loss: 0.0297586, Test Loss: 0.0346408
Epoch: 010, Train Loss: 0.0259011, Test Loss: 0.0297962
Epoch: 011, Train Loss: 0.0229931, Test Loss: 0.0262312
Epoch: 012, Train Loss: 0.0207413, Test Loss: 0.0234615
Epoch: 013, Train Loss: 0.0189482, Test Loss: 0.0212690
Epoch: 014, Train Loss: 0.0174663, Test Loss: 0.0194782
```

```
Epoch: 015, Train Loss: 0.0161937, Test Loss: 0.0179651
Epoch: 016, Train Loss: 0.0150643, Test Loss: 0.0166443
Epoch: 017, Train Loss: 0.0140387, Test Loss: 0.0154675
Epoch: 018, Train Loss: 0.0130912, Test Loss: 0.0143999
Epoch: 019, Train Loss: 0.0122091, Test Loss: 0.0134190
Epoch: 020, Train Loss: 0.0113824, Test Loss: 0.0125114
Epoch: 021, Train Loss: 0.0106074, Test Loss: 0.0116700
Epoch: 022, Train Loss: 0.0098817, Test Loss: 0.0108860
Epoch: 023, Train Loss: 0.0092031, Test Loss: 0.0101552
Epoch: 024, Train Loss: 0.0085675, Test Loss: 0.0094750
Epoch: 025, Train Loss: 0.0079730, Test Loss: 0.0088395
Epoch: 026, Train Loss: 0.0074178, Test Loss: 0.0082504
Epoch: 027, Train Loss: 0.0069005, Test Loss: 0.0077029
Epoch: 028, Train Loss: 0.0064193, Test Loss: 0.0071953
Epoch: 029, Train Loss: 0.0059723, Test Loss: 0.0067247
Epoch: 030, Train Loss: 0.0055574, Test Loss: 0.0062900
Epoch: 031, Train Loss: 0.0051734, Test Loss: 0.0058875
Epoch: 032, Train Loss: 0.0048172, Test Loss: 0.0055127
Epoch: 033, Train Loss: 0.0044882, Test Loss: 0.0051667
Epoch: 034, Train Loss: 0.0041846, Test Loss: 0.0048479
Epoch: 035, Train Loss: 0.0039042, Test Loss: 0.0045526
Epoch: 036, Train Loss: 0.0036460, Test Loss: 0.0042804
Epoch: 037, Train Loss: 0.0034088, Test Loss: 0.0040287
Epoch: 038, Train Loss: 0.0031904, Test Loss: 0.0037963
Epoch: 039, Train Loss: 0.0029893, Test Loss: 0.0035821
Epoch: 040, Train Loss: 0.0028049, Test Loss: 0.0033842
Epoch: 041, Train Loss: 0.0026359, Test Loss: 0.0032019
Epoch: 042, Train Loss: 0.0024810, Test Loss: 0.0030329
Epoch: 043, Train Loss: 0.0023388, Test Loss: 0.0028752
Epoch: 044, Train Loss: 0.0022087, Test Loss: 0.0027286
Epoch: 045, Train Loss: 0.0020896, Test Loss: 0.0025934
Epoch: 046, Train Loss: 0.0019807, Test Loss: 0.0024681
Epoch: 047, Train Loss: 0.0018808, Test Loss: 0.0023527
Epoch: 048, Train Loss: 0.0017889, Test Loss: 0.0022451
Epoch: 049, Train Loss: 0.0017044, Test Loss: 0.0021448
Epoch: 050, Train Loss: 0.0016269, Test Loss: 0.0020527
Epoch: 051, Train Loss: 0.0015557, Test Loss: 0.0019655
Epoch: 052, Train Loss: 0.0014901, Test Loss: 0.0018847
Epoch: 053, Train Loss: 0.0014295, Test Loss: 0.0018091
Epoch: 054, Train Loss: 0.0013736, Test Loss: 0.0017383
Epoch: 055, Train Loss: 0.0013218, Test Loss: 0.0016730
Epoch: 056, Train Loss: 0.0012734, Test Loss: 0.0016114
Epoch: 057, Train Loss: 0.0012284, Test Loss: 0.0015544
Epoch: 058, Train Loss: 0.0011865, Test Loss: 0.0015011
Epoch: 059, Train Loss: 0.0011472, Test Loss: 0.0014512
Epoch: 060, Train Loss: 0.0011105, Test Loss: 0.0014043
Epoch: 061, Train Loss: 0.0010760, Test Loss: 0.0013608
Epoch: 062, Train Loss: 0.0010437, Test Loss: 0.0013200
Epoch: 063, Train Loss: 0.0010134, Test Loss: 0.0012809
```

```
Epoch: 064, Train Loss: 0.0009853, Test Loss: 0.0012443
Epoch: 065, Train Loss: 0.0009589, Test Loss: 0.0012098
Epoch: 066, Train Loss: 0.0009340, Test Loss: 0.0011767
Epoch: 067, Train Loss: 0.0009104, Test Loss: 0.0011458
Epoch: 068, Train Loss: 0.0008881, Test Loss: 0.0011165
Epoch: 069, Train Loss: 0.0008671, Test Loss: 0.0010888
Epoch: 070, Train Loss: 0.0008472, Test Loss: 0.0010627
Epoch: 071, Train Loss: 0.0008283, Test Loss: 0.0010378
Epoch: 072, Train Loss: 0.0008103, Test Loss: 0.0010142
Epoch: 073, Train Loss: 0.0007933, Test Loss: 0.0009918
Epoch: 074, Train Loss: 0.0007771, Test Loss: 0.0009706
Epoch: 075, Train Loss: 0.0007616, Test Loss: 0.0009502
Epoch: 076, Train Loss: 0.0007468, Test Loss: 0.0009307
Epoch: 077, Train Loss: 0.0007327, Test Loss: 0.0009122
Epoch: 078, Train Loss: 0.0007192, Test Loss: 0.0008946
Epoch: 079, Train Loss: 0.0007063, Test Loss: 0.0008777
Epoch: 080, Train Loss: 0.0006939, Test Loss: 0.0008614
Epoch: 081, Train Loss: 0.0006819, Test Loss: 0.0008458
Epoch: 082, Train Loss: 0.0006702, Test Loss: 0.0008306
Epoch: 083, Train Loss: 0.0006590, Test Loss: 0.0008158
Epoch: 084, Train Loss: 0.0006481, Test Loss: 0.0008016
Epoch: 085, Train Loss: 0.0006375, Test Loss: 0.0007876
Epoch: 086, Train Loss: 0.0006272, Test Loss: 0.0007740
Epoch: 087, Train Loss: 0.0006170, Test Loss: 0.0007606
Epoch: 088, Train Loss: 0.0006071, Test Loss: 0.0007476
Epoch: 089, Train Loss: 0.0005973, Test Loss: 0.0007348
Epoch: 090, Train Loss: 0.0005876, Test Loss: 0.0007222
Epoch: 091, Train Loss: 0.0005781, Test Loss: 0.0007099
Epoch: 092, Train Loss: 0.0005687, Test Loss: 0.0006978
Epoch: 093, Train Loss: 0.0005594, Test Loss: 0.0006857
Epoch: 094, Train Loss: 0.0005501, Test Loss: 0.0006736
Epoch: 095, Train Loss: 0.0005409, Test Loss: 0.0006616
Epoch: 096, Train Loss: 0.0005317, Test Loss: 0.0006497
Epoch: 097, Train Loss: 0.0005227, Test Loss: 0.0006379
Epoch: 098, Train Loss: 0.0005137, Test Loss: 0.0006262
Epoch: 099, Train Loss: 0.0005048, Test Loss: 0.0006146
Epoch: 100, Train Loss: 0.0004961, Test Loss: 0.0006032
Epoch: 101, Train Loss: 0.0004873, Test Loss: 0.0005919
Epoch: 102, Train Loss: 0.0004787, Test Loss: 0.0005806
Epoch: 103, Train Loss: 0.0004702, Test Loss: 0.0005694
Epoch: 104, Train Loss: 0.0004616, Test Loss: 0.0005582
Epoch: 105, Train Loss: 0.0004533, Test Loss: 0.0005473
Epoch: 106, Train Loss: 0.0004450, Test Loss: 0.0005364
Epoch: 107, Train Loss: 0.0004370, Test Loss: 0.0005257
Epoch: 108, Train Loss: 0.0004291, Test Loss: 0.0005153
Epoch: 109, Train Loss: 0.0004213, Test Loss: 0.0005050
Epoch: 110, Train Loss: 0.0004138, Test Loss: 0.0004950
Epoch: 111, Train Loss: 0.0004064, Test Loss: 0.0004852
Epoch: 112, Train Loss: 0.0003991, Test Loss: 0.0004757
```

```
Epoch: 113, Train Loss: 0.0003921, Test Loss: 0.0004664
Epoch: 114, Train Loss: 0.0003851, Test Loss: 0.0004572
Epoch: 115, Train Loss: 0.0003784, Test Loss: 0.0004483
Epoch: 116, Train Loss: 0.0003718, Test Loss: 0.0004396
Epoch: 117, Train Loss: 0.0003653, Test Loss: 0.0004310
Epoch: 118, Train Loss: 0.0003589, Test Loss: 0.0004226
Epoch: 119, Train Loss: 0.0003527, Test Loss: 0.0004144
Epoch: 120, Train Loss: 0.0003466, Test Loss: 0.0004064
Epoch: 121, Train Loss: 0.0003406, Test Loss: 0.0003989
Epoch: 122, Train Loss: 0.0003347, Test Loss: 0.0003914
Epoch: 123, Train Loss: 0.0003289, Test Loss: 0.0003842
Epoch: 124, Train Loss: 0.0003233, Test Loss: 0.0003773
Epoch: 125, Train Loss: 0.0003177, Test Loss: 0.0003704
Epoch: 126, Train Loss: 0.0003123, Test Loss: 0.0003638
Epoch: 127, Train Loss: 0.0003070, Test Loss: 0.0003573
Epoch: 128, Train Loss: 0.0003018, Test Loss: 0.0003510
Epoch: 129, Train Loss: 0.0002967, Test Loss: 0.0003448
Epoch: 130, Train Loss: 0.0002918, Test Loss: 0.0003387
Epoch: 131, Train Loss: 0.0002869, Test Loss: 0.0003328
Epoch: 132, Train Loss: 0.0002821, Test Loss: 0.0003271
Epoch: 133, Train Loss: 0.0002774, Test Loss: 0.0003214
Epoch: 134, Train Loss: 0.0002728, Test Loss: 0.0003159
Epoch: 135, Train Loss: 0.0002683, Test Loss: 0.0003103
Epoch: 136, Train Loss: 0.0002639, Test Loss: 0.0003048
Epoch: 137, Train Loss: 0.0002596, Test Loss: 0.0002995
Epoch: 138, Train Loss: 0.0002554, Test Loss: 0.0002941
Epoch: 139, Train Loss: 0.0002512, Test Loss: 0.0002889
Epoch: 140, Train Loss: 0.0002472, Test Loss: 0.0002837
Epoch: 141, Train Loss: 0.0002432, Test Loss: 0.0002786
Epoch: 142, Train Loss: 0.0002393, Test Loss: 0.0002736
Epoch: 143, Train Loss: 0.0002354, Test Loss: 0.0002687
Epoch: 144, Train Loss: 0.0002317, Test Loss: 0.0002640
Epoch: 145, Train Loss: 0.0002280, Test Loss: 0.0002594
Epoch: 146, Train Loss: 0.0002244, Test Loss: 0.0002548
Epoch: 147, Train Loss: 0.0002209, Test Loss: 0.0002503
Epoch: 148, Train Loss: 0.0002174, Test Loss: 0.0002459
Epoch: 149, Train Loss: 0.0002139, Test Loss: 0.0002416
Epoch: 150, Train Loss: 0.0002105, Test Loss: 0.0002373
Epoch: 151, Train Loss: 0.0002072, Test Loss: 0.0002331
Epoch: 152, Train Loss: 0.0002040, Test Loss: 0.0002289
Epoch: 153, Train Loss: 0.0002008, Test Loss: 0.0002250
Epoch: 154, Train Loss: 0.0001977, Test Loss: 0.0002211
Epoch: 155, Train Loss: 0.0001946, Test Loss: 0.0002173
Epoch: 156, Train Loss: 0.0001916, Test Loss: 0.0002136
Epoch: 157, Train Loss: 0.0001887, Test Loss: 0.0002099
Epoch: 158, Train Loss: 0.0001858, Test Loss: 0.0002063
Epoch: 159, Train Loss: 0.0001829, Test Loss: 0.0002028
Epoch: 160, Train Loss: 0.0001801, Test Loss: 0.0001994
Epoch: 161, Train Loss: 0.0001774, Test Loss: 0.0001960
```

```
Epoch: 162, Train Loss: 0.0001747, Test Loss: 0.0001927
Epoch: 163, Train Loss: 0.0001720, Test Loss: 0.0001895
Epoch: 164, Train Loss: 0.0001694, Test Loss: 0.0001864
Epoch: 165, Train Loss: 0.0001668, Test Loss: 0.0001834
Epoch: 166, Train Loss: 0.0001643, Test Loss: 0.0001804
Epoch: 167, Train Loss: 0.0001618, Test Loss: 0.0001775
Epoch: 168, Train Loss: 0.0001594, Test Loss: 0.0001745
Epoch: 169, Train Loss: 0.0001570, Test Loss: 0.0001717
Epoch: 170, Train Loss: 0.0001547, Test Loss: 0.0001689
Epoch: 171, Train Loss: 0.0001524, Test Loss: 0.0001661
Epoch: 172, Train Loss: 0.0001501, Test Loss: 0.0001635
Epoch: 173, Train Loss: 0.0001479, Test Loss: 0.0001608
Epoch: 174, Train Loss: 0.0001457, Test Loss: 0.0001582
Epoch: 175, Train Loss: 0.0001436, Test Loss: 0.0001557
Epoch: 176, Train Loss: 0.0001414, Test Loss: 0.0001533
Epoch: 177, Train Loss: 0.0001394, Test Loss: 0.0001509
Epoch: 178, Train Loss: 0.0001373, Test Loss: 0.0001485
Epoch: 179, Train Loss: 0.0001353, Test Loss: 0.0001462
Epoch: 180, Train Loss: 0.0001333, Test Loss: 0.0001439
Epoch: 181, Train Loss: 0.0001314, Test Loss: 0.0001417
Epoch: 182, Train Loss: 0.0001295, Test Loss: 0.0001395
Epoch: 183, Train Loss: 0.0001276, Test Loss: 0.0001374
Epoch: 184, Train Loss: 0.0001257, Test Loss: 0.0001352
Epoch: 185, Train Loss: 0.0001239, Test Loss: 0.0001331
Epoch: 186, Train Loss: 0.0001221, Test Loss: 0.0001311
Epoch: 187, Train Loss: 0.0001204, Test Loss: 0.0001292
Epoch: 188, Train Loss: 0.0001186, Test Loss: 0.0001272
Epoch: 189, Train Loss: 0.0001169, Test Loss: 0.0001253
Epoch: 190, Train Loss: 0.0001153, Test Loss: 0.0001234
Epoch: 191, Train Loss: 0.0001136, Test Loss: 0.0001216
Epoch: 192, Train Loss: 0.0001120, Test Loss: 0.0001198
Epoch: 193, Train Loss: 0.0001104, Test Loss: 0.0001181
Epoch: 194, Train Loss: 0.0001089, Test Loss: 0.0001163
Epoch: 195, Train Loss: 0.0001073, Test Loss: 0.0001147
Epoch: 196, Train Loss: 0.0001058, Test Loss: 0.0001130
Epoch: 197, Train Loss: 0.0001043, Test Loss: 0.0001113
Epoch: 198, Train Loss: 0.0001029, Test Loss: 0.0001097
Epoch: 199, Train Loss: 0.0001014, Test Loss: 0.0001082
Epoch: 200, Train Loss: 0.0001000, Test Loss: 0.0001066
```



Problem 4.2 Determine appropriate mini-batch size = 32, epoch = 300

```
# ################## Part 1: Load data and create batch
##################
N_total = 600
N_train = 500
x = torch.unsqueeze(torch.linspace(0, 1, N_total), dim=1)
r = torch.randperm(N_total)
x = x[r, :]
y = 0.2 + 0.4 * torch.pow(x, 2) + 0.3 * x * torch.sin(15 * x) + 0.05 *
torch.cos(50 * x)

class CustomDataset(Dataset):
    def __init__(self, x, y):
        self.y = y
        self.x = x

    def __len__(self):
        return len(self.y)

    def __getitem__(self, idx):
        y1 = self.y[idx]
        x1 = self.x[idx]
        return (x1, y1)

# Change batch_size here to test different values
```

```
batch_size = 32  # Experiment with different batch sizes: 32, 64, 128
trainset = CustomDataset(x[0:N_train, :], y[0:N_train, :])
testset = CustomDataset(x[N_train:N_total, :], y[N_train:N_total, :])
train_loader = torch.utils.data.DataLoader(trainset,
batch_size=batch_size)
test_loader = torch.utils.data.DataLoader(testset,
batch_size=batch_size)

# ################## Part 2: Define Model ##################
model = nn.Sequential(
    nn.Linear(1, 1024, bias=True),
    nn.ReLU(),
    nn.Linear(1024, 1, bias=True)
)

def init_weights(m):
    if isinstance(m, nn.Linear):
        m.weight.data.uniform_(-1, 1)
        m.bias.data.uniform_(-1, 1)

model.apply(init_weights)

# ################## Part 3: Define Loss and Optimizer
##################
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

# ################## Part 4: Training and Testing ##################
def train_NN():
    model.train()
    for images, labels in train_loader:
        out = model(images)
        loss = criterion(out, labels)
        loss.backward()
        optimizer.step()
        optimizer.zero_grad()
    return loss

def test_NN(loader):
    model.eval()
    loss = 0
    with torch.no_grad():
        for images, labels in loader:
            out = model(images)
            loss += criterion(out, labels).item()
    loss = loss / len(loader)
    return loss

# Experiment with different numbers of epochs
N_epoch = 300  # Try values like 100, 200, 500
```

```
train_loss = np.zeros((N_epoch, 1))
test_loss = np.zeros((N_epoch, 1))

for epoch in range(N_epoch):
    train_NN()
    train_loss[epoch, 0] = test_NN(train_loader)
    test_loss[epoch, 0] = test_NN(test_loader)
    print(f'Epoch: {epoch+1:03d}, Train Loss: {train_loss[epoch,
0]:.7f}, Test Loss: {test_loss[epoch, 0]:.7f}')

# ################## Final Prediction ##################
x_test = torch.unsqueeze(torch.linspace(0, 1, 1999), dim=1)
y_test = model(x_test)

# Plot the results
plt.plot(x[0:N_total], y[0:N_total], 'bo', label='True data')
plt.plot(x_test, y_test.detach().numpy(), 'r', label='Model
prediction')
plt.legend()
plt.show()
```

```
Epoch: 001, Train Loss: 1.6943940, Test Loss: 1.3576902
Epoch: 002, Train Loss: 0.2147134, Test Loss: 0.1538809
Epoch: 003, Train Loss: 0.2042176, Test Loss: 0.3091932
Epoch: 004, Train Loss: 0.0874204, Test Loss: 0.0568524
Epoch: 005, Train Loss: 0.0379086, Test Loss: 0.0642589
Epoch: 006, Train Loss: 0.0275121, Test Loss: 0.0280169
Epoch: 007, Train Loss: 0.0204654, Test Loss: 0.0293006
Epoch: 008, Train Loss: 0.0172203, Test Loss: 0.0208747
Epoch: 009, Train Loss: 0.0140973, Test Loss: 0.0179452
Epoch: 010, Train Loss: 0.0118216, Test Loss: 0.0144192
Epoch: 011, Train Loss: 0.0099079, Test Loss: 0.0119318
Epoch: 012, Train Loss: 0.0083451, Test Loss: 0.0099552
Epoch: 013, Train Loss: 0.0070816, Test Loss: 0.0083374
Epoch: 014, Train Loss: 0.0060404, Test Loss: 0.0070783
Epoch: 015, Train Loss: 0.0051920, Test Loss: 0.0060603
Epoch: 016, Train Loss: 0.0044988, Test Loss: 0.0052423
Epoch: 017, Train Loss: 0.0039305, Test Loss: 0.0045852
Epoch: 018, Train Loss: 0.0034660, Test Loss: 0.0040514
Epoch: 019, Train Loss: 0.0030858, Test Loss: 0.0036177
Epoch: 020, Train Loss: 0.0027742, Test Loss: 0.0032636
Epoch: 021, Train Loss: 0.0025181, Test Loss: 0.0029728
Epoch: 022, Train Loss: 0.0023065, Test Loss: 0.0027344
Epoch: 023, Train Loss: 0.0021306, Test Loss: 0.0025380
Epoch: 024, Train Loss: 0.0019837, Test Loss: 0.0023759
Epoch: 025, Train Loss: 0.0018604, Test Loss: 0.0022404
Epoch: 026, Train Loss: 0.0017561, Test Loss: 0.0021269
Epoch: 027, Train Loss: 0.0016667, Test Loss: 0.0020320
Epoch: 028, Train Loss: 0.0015893, Test Loss: 0.0019518
Epoch: 029, Train Loss: 0.0015223, Test Loss: 0.0018834
```

```
Epoch: 030, Train Loss: 0.0014633, Test Loss: 0.0018240
Epoch: 031, Train Loss: 0.0014109, Test Loss: 0.0017724
Epoch: 032, Train Loss: 0.0013638, Test Loss: 0.0017273
Epoch: 033, Train Loss: 0.0013214, Test Loss: 0.0016872
Epoch: 034, Train Loss: 0.0012829, Test Loss: 0.0016511
Epoch: 035, Train Loss: 0.0012472, Test Loss: 0.0016184
Epoch: 036, Train Loss: 0.0012142, Test Loss: 0.0015887
Epoch: 037, Train Loss: 0.0011835, Test Loss: 0.0015614
Epoch: 038, Train Loss: 0.0011547, Test Loss: 0.0015359
Epoch: 039, Train Loss: 0.0011278, Test Loss: 0.0015123
Epoch: 040, Train Loss: 0.0011025, Test Loss: 0.0014909
Epoch: 041, Train Loss: 0.0010786, Test Loss: 0.0014706
Epoch: 042, Train Loss: 0.0010560, Test Loss: 0.0014513
Epoch: 043, Train Loss: 0.0010346, Test Loss: 0.0014326
Epoch: 044, Train Loss: 0.0010143, Test Loss: 0.0014152
Epoch: 045, Train Loss: 0.0009953, Test Loss: 0.0013994
Epoch: 046, Train Loss: 0.0009774, Test Loss: 0.0013845
Epoch: 047, Train Loss: 0.0009608, Test Loss: 0.0013707
Epoch: 048, Train Loss: 0.0009458, Test Loss: 0.0013587
Epoch: 049, Train Loss: 0.0009324, Test Loss: 0.0013483
Epoch: 050, Train Loss: 0.0009206, Test Loss: 0.0013392
Epoch: 051, Train Loss: 0.0009103, Test Loss: 0.0013318
Epoch: 052, Train Loss: 0.0009018, Test Loss: 0.0013260
Epoch: 053, Train Loss: 0.0008949, Test Loss: 0.0013218
Epoch: 054, Train Loss: 0.0008893, Test Loss: 0.0013190
Epoch: 055, Train Loss: 0.0008846, Test Loss: 0.0013167
Epoch: 056, Train Loss: 0.0008807, Test Loss: 0.0013146
Epoch: 057, Train Loss: 0.0008774, Test Loss: 0.0013129
Epoch: 058, Train Loss: 0.0008744, Test Loss: 0.0013115
Epoch: 059, Train Loss: 0.0008713, Test Loss: 0.0013094
Epoch: 060, Train Loss: 0.0008677, Test Loss: 0.0013065
Epoch: 061, Train Loss: 0.0008635, Test Loss: 0.0013021
Epoch: 062, Train Loss: 0.0008585, Test Loss: 0.0012968
Epoch: 063, Train Loss: 0.0008522, Test Loss: 0.0012894
Epoch: 064, Train Loss: 0.0008445, Test Loss: 0.0012801
Epoch: 065, Train Loss: 0.0008352, Test Loss: 0.0012685
Epoch: 066, Train Loss: 0.0008244, Test Loss: 0.0012546
Epoch: 067, Train Loss: 0.0008122, Test Loss: 0.0012392
Epoch: 068, Train Loss: 0.0007990, Test Loss: 0.0012226
Epoch: 069, Train Loss: 0.0007850, Test Loss: 0.0012049
Epoch: 070, Train Loss: 0.0007704, Test Loss: 0.0011862
Epoch: 071, Train Loss: 0.0007554, Test Loss: 0.0011672
Epoch: 072, Train Loss: 0.0007402, Test Loss: 0.0011476
Epoch: 073, Train Loss: 0.0007247, Test Loss: 0.0011273
Epoch: 074, Train Loss: 0.0007091, Test Loss: 0.0011066
Epoch: 075, Train Loss: 0.0006935, Test Loss: 0.0010863
Epoch: 076, Train Loss: 0.0006782, Test Loss: 0.0010661
Epoch: 077, Train Loss: 0.0006631, Test Loss: 0.0010459
Epoch: 078, Train Loss: 0.0006481, Test Loss: 0.0010261
```

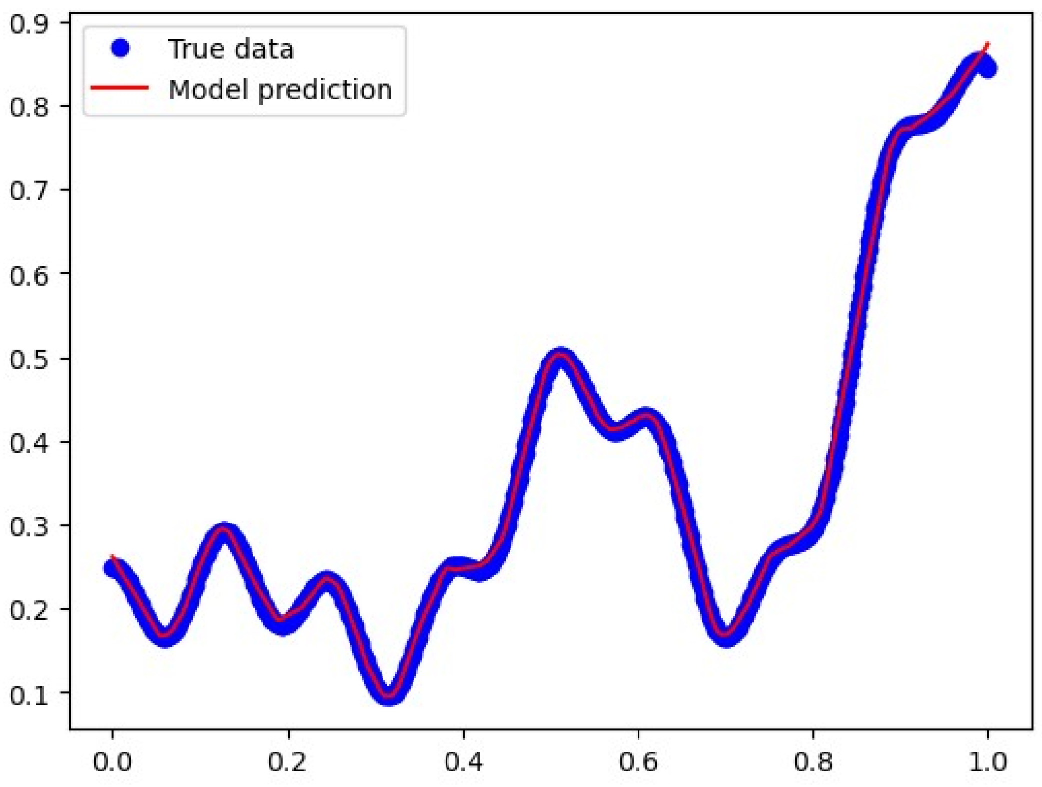
```
Epoch: 079, Train Loss: 0.0006338, Test Loss: 0.0010068
Epoch: 080, Train Loss: 0.0006197, Test Loss: 0.0009879
Epoch: 081, Train Loss: 0.0006058, Test Loss: 0.0009691
Epoch: 082, Train Loss: 0.0005923, Test Loss: 0.0009509
Epoch: 083, Train Loss: 0.0005792, Test Loss: 0.0009331
Epoch: 084, Train Loss: 0.0005665, Test Loss: 0.0009158
Epoch: 085, Train Loss: 0.0005542, Test Loss: 0.0008988
Epoch: 086, Train Loss: 0.0005423, Test Loss: 0.0008825
Epoch: 087, Train Loss: 0.0005309, Test Loss: 0.0008666
Epoch: 088, Train Loss: 0.0005196, Test Loss: 0.0008507
Epoch: 089, Train Loss: 0.0005087, Test Loss: 0.0008352
Epoch: 090, Train Loss: 0.0004978, Test Loss: 0.0008199
Epoch: 091, Train Loss: 0.0004873, Test Loss: 0.0008051
Epoch: 092, Train Loss: 0.0004773, Test Loss: 0.0007911
Epoch: 093, Train Loss: 0.0004676, Test Loss: 0.0007772
Epoch: 094, Train Loss: 0.0004582, Test Loss: 0.0007639
Epoch: 095, Train Loss: 0.0004490, Test Loss: 0.0007508
Epoch: 096, Train Loss: 0.0004400, Test Loss: 0.0007378
Epoch: 097, Train Loss: 0.0004314, Test Loss: 0.0007254
Epoch: 098, Train Loss: 0.0004229, Test Loss: 0.0007132
Epoch: 099, Train Loss: 0.0004147, Test Loss: 0.0007013
Epoch: 100, Train Loss: 0.0004065, Test Loss: 0.0006894
Epoch: 101, Train Loss: 0.0003986, Test Loss: 0.0006778
Epoch: 102, Train Loss: 0.0003910, Test Loss: 0.0006663
Epoch: 103, Train Loss: 0.0003836, Test Loss: 0.0006553
Epoch: 104, Train Loss: 0.0003764, Test Loss: 0.0006444
Epoch: 105, Train Loss: 0.0003693, Test Loss: 0.0006336
Epoch: 106, Train Loss: 0.0003625, Test Loss: 0.0006231
Epoch: 107, Train Loss: 0.0003558, Test Loss: 0.0006129
Epoch: 108, Train Loss: 0.0003493, Test Loss: 0.0006030
Epoch: 109, Train Loss: 0.0003429, Test Loss: 0.0005935
Epoch: 110, Train Loss: 0.0003367, Test Loss: 0.0005842
Epoch: 111, Train Loss: 0.0003306, Test Loss: 0.0005750
Epoch: 112, Train Loss: 0.0003247, Test Loss: 0.0005660
Epoch: 113, Train Loss: 0.0003188, Test Loss: 0.0005571
Epoch: 114, Train Loss: 0.0003130, Test Loss: 0.0005484
Epoch: 115, Train Loss: 0.0003073, Test Loss: 0.0005398
Epoch: 116, Train Loss: 0.0003018, Test Loss: 0.0005312
Epoch: 117, Train Loss: 0.0002964, Test Loss: 0.0005229
Epoch: 118, Train Loss: 0.0002911, Test Loss: 0.0005147
Epoch: 119, Train Loss: 0.0002859, Test Loss: 0.0005066
Epoch: 120, Train Loss: 0.0002809, Test Loss: 0.0004986
Epoch: 121, Train Loss: 0.0002760, Test Loss: 0.0004908
Epoch: 122, Train Loss: 0.0002711, Test Loss: 0.0004830
Epoch: 123, Train Loss: 0.0002664, Test Loss: 0.0004754
Epoch: 124, Train Loss: 0.0002617, Test Loss: 0.0004678
Epoch: 125, Train Loss: 0.0002572, Test Loss: 0.0004604
Epoch: 126, Train Loss: 0.0002527, Test Loss: 0.0004534
Epoch: 127, Train Loss: 0.0002484, Test Loss: 0.0004465
```

```
Epoch: 128, Train Loss: 0.0002441, Test Loss: 0.0004396
Epoch: 129, Train Loss: 0.0002399, Test Loss: 0.0004330
Epoch: 130, Train Loss: 0.0002358, Test Loss: 0.0004263
Epoch: 131, Train Loss: 0.0002317, Test Loss: 0.0004199
Epoch: 132, Train Loss: 0.0002278, Test Loss: 0.0004134
Epoch: 133, Train Loss: 0.0002239, Test Loss: 0.0004071
Epoch: 134, Train Loss: 0.0002201, Test Loss: 0.0004008
Epoch: 135, Train Loss: 0.0002164, Test Loss: 0.0003946
Epoch: 136, Train Loss: 0.0002127, Test Loss: 0.0003885
Epoch: 137, Train Loss: 0.0002092, Test Loss: 0.0003825
Epoch: 138, Train Loss: 0.0002057, Test Loss: 0.0003766
Epoch: 139, Train Loss: 0.0002022, Test Loss: 0.0003709
Epoch: 140, Train Loss: 0.0001989, Test Loss: 0.0003652
Epoch: 141, Train Loss: 0.0001956, Test Loss: 0.0003594
Epoch: 142, Train Loss: 0.0001924, Test Loss: 0.0003538
Epoch: 143, Train Loss: 0.0001892, Test Loss: 0.0003482
Epoch: 144, Train Loss: 0.0001862, Test Loss: 0.0003427
Epoch: 145, Train Loss: 0.0001832, Test Loss: 0.0003373
Epoch: 146, Train Loss: 0.0001802, Test Loss: 0.0003319
Epoch: 147, Train Loss: 0.0001773, Test Loss: 0.0003267
Epoch: 148, Train Loss: 0.0001745, Test Loss: 0.0003217
Epoch: 149, Train Loss: 0.0001717, Test Loss: 0.0003167
Epoch: 150, Train Loss: 0.0001690, Test Loss: 0.0003118
Epoch: 151, Train Loss: 0.0001664, Test Loss: 0.0003070
Epoch: 152, Train Loss: 0.0001638, Test Loss: 0.0003023
Epoch: 153, Train Loss: 0.0001613, Test Loss: 0.0002977
Epoch: 154, Train Loss: 0.0001589, Test Loss: 0.0002932
Epoch: 155, Train Loss: 0.0001565, Test Loss: 0.0002888
Epoch: 156, Train Loss: 0.0001542, Test Loss: 0.0002843
Epoch: 157, Train Loss: 0.0001520, Test Loss: 0.0002800
Epoch: 158, Train Loss: 0.0001499, Test Loss: 0.0002758
Epoch: 159, Train Loss: 0.0001479, Test Loss: 0.0002718
Epoch: 160, Train Loss: 0.0001459, Test Loss: 0.0002678
Epoch: 161, Train Loss: 0.0001440, Test Loss: 0.0002637
Epoch: 162, Train Loss: 0.0001422, Test Loss: 0.0002597
Epoch: 163, Train Loss: 0.0001405, Test Loss: 0.0002557
Epoch: 164, Train Loss: 0.0001389, Test Loss: 0.0002519
Epoch: 165, Train Loss: 0.0001374, Test Loss: 0.0002482
Epoch: 166, Train Loss: 0.0001360, Test Loss: 0.0002446
Epoch: 167, Train Loss: 0.0001347, Test Loss: 0.0002412
Epoch: 168, Train Loss: 0.0001335, Test Loss: 0.0002379
Epoch: 169, Train Loss: 0.0001325, Test Loss: 0.0002348
Epoch: 170, Train Loss: 0.0001316, Test Loss: 0.0002318
Epoch: 171, Train Loss: 0.0001309, Test Loss: 0.0002290
Epoch: 172, Train Loss: 0.0001303, Test Loss: 0.0002263
Epoch: 173, Train Loss: 0.0001301, Test Loss: 0.0002240
Epoch: 174, Train Loss: 0.0001300, Test Loss: 0.0002218
Epoch: 175, Train Loss: 0.0001301, Test Loss: 0.0002199
Epoch: 176, Train Loss: 0.0001305, Test Loss: 0.0002183
```

```
Epoch: 177, Train Loss: 0.0001312, Test Loss: 0.0002170
Epoch: 178, Train Loss: 0.0001322, Test Loss: 0.0002161
Epoch: 179, Train Loss: 0.0001336, Test Loss: 0.0002155
Epoch: 180, Train Loss: 0.0001353, Test Loss: 0.0002153
Epoch: 181, Train Loss: 0.0001374, Test Loss: 0.0002155
Epoch: 182, Train Loss: 0.0001399, Test Loss: 0.0002159
Epoch: 183, Train Loss: 0.0001427, Test Loss: 0.0002168
Epoch: 184, Train Loss: 0.0001459, Test Loss: 0.0002180
Epoch: 185, Train Loss: 0.0001494, Test Loss: 0.0002198
Epoch: 186, Train Loss: 0.0001531, Test Loss: 0.0002216
Epoch: 187, Train Loss: 0.0001568, Test Loss: 0.0002234
Epoch: 188, Train Loss: 0.0001605, Test Loss: 0.0002250
Epoch: 189, Train Loss: 0.0001639, Test Loss: 0.0002264
Epoch: 190, Train Loss: 0.0001669, Test Loss: 0.0002274
Epoch: 191, Train Loss: 0.0001691, Test Loss: 0.0002277
Epoch: 192, Train Loss: 0.0001706, Test Loss: 0.0002274
Epoch: 193, Train Loss: 0.0001710, Test Loss: 0.0002261
Epoch: 194, Train Loss: 0.0001702, Test Loss: 0.0002237
Epoch: 195, Train Loss: 0.0001683, Test Loss: 0.0002205
Epoch: 196, Train Loss: 0.0001651, Test Loss: 0.0002160
Epoch: 197, Train Loss: 0.0001609, Test Loss: 0.0002108
Epoch: 198, Train Loss: 0.0001558, Test Loss: 0.0002048
Epoch: 199, Train Loss: 0.0001500, Test Loss: 0.0001982
Epoch: 200, Train Loss: 0.0001438, Test Loss: 0.0001914
Epoch: 201, Train Loss: 0.0001375, Test Loss: 0.0001844
Epoch: 202, Train Loss: 0.0001311, Test Loss: 0.0001775
Epoch: 203, Train Loss: 0.0001252, Test Loss: 0.0001708
Epoch: 204, Train Loss: 0.0001196, Test Loss: 0.0001646
Epoch: 205, Train Loss: 0.0001144, Test Loss: 0.0001588
Epoch: 206, Train Loss: 0.0001096, Test Loss: 0.0001533
Epoch: 207, Train Loss: 0.0001052, Test Loss: 0.0001482
Epoch: 208, Train Loss: 0.0001011, Test Loss: 0.0001436
Epoch: 209, Train Loss: 0.0000975, Test Loss: 0.0001392
Epoch: 210, Train Loss: 0.0000941, Test Loss: 0.0001351
Epoch: 211, Train Loss: 0.0000910, Test Loss: 0.0001312
Epoch: 212, Train Loss: 0.0000882, Test Loss: 0.0001276
Epoch: 213, Train Loss: 0.0000855, Test Loss: 0.0001242
Epoch: 214, Train Loss: 0.0000831, Test Loss: 0.0001211
Epoch: 215, Train Loss: 0.0000809, Test Loss: 0.0001182
Epoch: 216, Train Loss: 0.0000790, Test Loss: 0.0001156
Epoch: 217, Train Loss: 0.0000771, Test Loss: 0.0001128
Epoch: 218, Train Loss: 0.0000753, Test Loss: 0.0001105
Epoch: 219, Train Loss: 0.0000738, Test Loss: 0.0001083
Epoch: 220, Train Loss: 0.0000722, Test Loss: 0.0001059
Epoch: 221, Train Loss: 0.0000709, Test Loss: 0.0001040
Epoch: 222, Train Loss: 0.0000696, Test Loss: 0.0001021
Epoch: 223, Train Loss: 0.0000683, Test Loss: 0.0001001
Epoch: 224, Train Loss: 0.0000671, Test Loss: 0.0000982
Epoch: 225, Train Loss: 0.0000661, Test Loss: 0.0000966
```

```
Epoch: 226, Train Loss: 0.0000651, Test Loss: 0.0000951
Epoch: 227, Train Loss: 0.0000641, Test Loss: 0.0000934
Epoch: 228, Train Loss: 0.0000631, Test Loss: 0.0000918
Epoch: 229, Train Loss: 0.0000622, Test Loss: 0.0000903
Epoch: 230, Train Loss: 0.0000614, Test Loss: 0.0000889
Epoch: 231, Train Loss: 0.0000605, Test Loss: 0.0000875
Epoch: 232, Train Loss: 0.0000597, Test Loss: 0.0000862
Epoch: 233, Train Loss: 0.0000590, Test Loss: 0.0000849
Epoch: 234, Train Loss: 0.0000582, Test Loss: 0.0000837
Epoch: 235, Train Loss: 0.0000575, Test Loss: 0.0000825
Epoch: 236, Train Loss: 0.0000568, Test Loss: 0.0000813
Epoch: 237, Train Loss: 0.0000561, Test Loss: 0.0000802
Epoch: 238, Train Loss: 0.0000555, Test Loss: 0.0000791
Epoch: 239, Train Loss: 0.0000548, Test Loss: 0.0000780
Epoch: 240, Train Loss: 0.0000542, Test Loss: 0.0000770
Epoch: 241, Train Loss: 0.0000536, Test Loss: 0.0000760
Epoch: 242, Train Loss: 0.0000530, Test Loss: 0.0000750
Epoch: 243, Train Loss: 0.0000525, Test Loss: 0.0000739
Epoch: 244, Train Loss: 0.0000519, Test Loss: 0.0000730
Epoch: 245, Train Loss: 0.0000514, Test Loss: 0.0000720
Epoch: 246, Train Loss: 0.0000509, Test Loss: 0.0000710
Epoch: 247, Train Loss: 0.0000503, Test Loss: 0.0000701
Epoch: 248, Train Loss: 0.0000498, Test Loss: 0.0000692
Epoch: 249, Train Loss: 0.0000493, Test Loss: 0.0000683
Epoch: 250, Train Loss: 0.0000488, Test Loss: 0.0000674
Epoch: 251, Train Loss: 0.0000483, Test Loss: 0.0000666
Epoch: 252, Train Loss: 0.0000479, Test Loss: 0.0000657
Epoch: 253, Train Loss: 0.0000474, Test Loss: 0.0000649
Epoch: 254, Train Loss: 0.0000470, Test Loss: 0.0000641
Epoch: 255, Train Loss: 0.0000465, Test Loss: 0.0000633
Epoch: 256, Train Loss: 0.0000461, Test Loss: 0.0000625
Epoch: 257, Train Loss: 0.0000457, Test Loss: 0.0000617
Epoch: 258, Train Loss: 0.0000453, Test Loss: 0.0000610
Epoch: 259, Train Loss: 0.0000449, Test Loss: 0.0000603
Epoch: 260, Train Loss: 0.0000445, Test Loss: 0.0000596
Epoch: 261, Train Loss: 0.0000441, Test Loss: 0.0000589
Epoch: 262, Train Loss: 0.0000437, Test Loss: 0.0000582
Epoch: 263, Train Loss: 0.0000433, Test Loss: 0.0000575
Epoch: 264, Train Loss: 0.0000430, Test Loss: 0.0000568
Epoch: 265, Train Loss: 0.0000426, Test Loss: 0.0000562
Epoch: 266, Train Loss: 0.0000423, Test Loss: 0.0000556
Epoch: 267, Train Loss: 0.0000420, Test Loss: 0.0000550
Epoch: 268, Train Loss: 0.0000417, Test Loss: 0.0000544
Epoch: 269, Train Loss: 0.0000414, Test Loss: 0.0000539
Epoch: 270, Train Loss: 0.0000411, Test Loss: 0.0000533
Epoch: 271, Train Loss: 0.0000408, Test Loss: 0.0000527
Epoch: 272, Train Loss: 0.0000405, Test Loss: 0.0000522
Epoch: 273, Train Loss: 0.0000403, Test Loss: 0.0000517
Epoch: 274, Train Loss: 0.0000400, Test Loss: 0.0000511
```

```
Epoch: 275, Train Loss: 0.0000398, Test Loss: 0.0000506
Epoch: 276, Train Loss: 0.0000395, Test Loss: 0.0000501
Epoch: 277, Train Loss: 0.0000392, Test Loss: 0.0000496
Epoch: 278, Train Loss: 0.0000389, Test Loss: 0.0000491
Epoch: 279, Train Loss: 0.0000386, Test Loss: 0.0000486
Epoch: 280, Train Loss: 0.0000384, Test Loss: 0.0000481
Epoch: 281, Train Loss: 0.0000381, Test Loss: 0.0000476
Epoch: 282, Train Loss: 0.0000378, Test Loss: 0.0000471
Epoch: 283, Train Loss: 0.0000374, Test Loss: 0.0000466
Epoch: 284, Train Loss: 0.0000370, Test Loss: 0.0000461
Epoch: 285, Train Loss: 0.0000366, Test Loss: 0.0000455
Epoch: 286, Train Loss: 0.0000362, Test Loss: 0.0000450
Epoch: 287, Train Loss: 0.0000358, Test Loss: 0.0000445
Epoch: 288, Train Loss: 0.0000353, Test Loss: 0.0000439
Epoch: 289, Train Loss: 0.0000349, Test Loss: 0.0000434
Epoch: 290, Train Loss: 0.0000344, Test Loss: 0.0000429
Epoch: 291, Train Loss: 0.0000341, Test Loss: 0.0000424
Epoch: 292, Train Loss: 0.0000337, Test Loss: 0.0000419
Epoch: 293, Train Loss: 0.0000333, Test Loss: 0.0000415
Epoch: 294, Train Loss: 0.0000330, Test Loss: 0.0000411
Epoch: 295, Train Loss: 0.0000327, Test Loss: 0.0000407
Epoch: 296, Train Loss: 0.0000325, Test Loss: 0.0000403
Epoch: 297, Train Loss: 0.0000323, Test Loss: 0.0000400
Epoch: 298, Train Loss: 0.0000321, Test Loss: 0.0000396
Epoch: 299, Train Loss: 0.0000319, Test Loss: 0.0000394
Epoch: 300, Train Loss: 0.0000317, Test Loss: 0.0000390
```



Problem 4.2 Determine appropriate mini-batch size = 32, epoch = 400

```
# ################## Part 1: Load data and create batch
##################
N_total = 600
N_train = 500
x = torch.unsqueeze(torch.linspace(0, 1, N_total), dim=1)
r = torch.randperm(N_total)
x = x[r, :]
y = 0.2 + 0.4 * torch.pow(x, 2) + 0.3 * x * torch.sin(15 * x) + 0.05 *
torch.cos(50 * x)

class CustomDataset(Dataset):
    def __init__(self, x, y):
        self.y = y
        self.x = x

    def __len__(self):
        return len(self.y)

    def __getitem__(self, idx):
        y1 = self.y[idx]
        x1 = self.x[idx]
        return (x1, y1)

# Change batch_size here to test different values
```

```
batch_size = 32  # Experiment with different batch sizes: 32, 64, 128
trainset = CustomDataset(x[0:N_train, :], y[0:N_train, :])
testset = CustomDataset(x[N_train:N_total, :], y[N_train:N_total, :])
train_loader = torch.utils.data.DataLoader(trainset,
batch_size=batch_size)
test_loader = torch.utils.data.DataLoader(testset,
batch_size=batch_size)

# ################## Part 2: Define Model ##################
model = nn.Sequential(
    nn.Linear(1, 1024, bias=True),
    nn.ReLU(),
    nn.Linear(1024, 1, bias=True)
)

def init_weights(m):
    if isinstance(m, nn.Linear):
        m.weight.data.uniform_(-1, 1)
        m.bias.data.uniform_(-1, 1)

model.apply(init_weights)

# ################## Part 3: Define Loss and Optimizer
##################
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

# ################## Part 4: Training and Testing ##################
def train_NN():
    model.train()
    for images, labels in train_loader:
        out = model(images)
        loss = criterion(out, labels)
        loss.backward()
        optimizer.step()
        optimizer.zero_grad()
    return loss

def test_NN(loader):
    model.eval()
    loss = 0
    with torch.no_grad():
        for images, labels in loader:
            out = model(images)
            loss += criterion(out, labels).item()
    loss = loss / len(loader)
    return loss

# Experiment with different numbers of epochs
N_epoch = 400  # Try values like 100, 200, 500
```

```
train_loss = np.zeros((N_epoch, 1))
test_loss = np.zeros((N_epoch, 1))

for epoch in range(N_epoch):
    train_NN()
    train_loss[epoch, 0] = test_NN(train_loader)
    test_loss[epoch, 0] = test_NN(test_loader)
    print(f'Epoch: {epoch+1:03d}, Train Loss: {train_loss[epoch,
0]:.7f}, Test Loss: {test_loss[epoch, 0]:.7f}')

# ################## Final Prediction ##################
x_test = torch.unsqueeze(torch.linspace(0, 1, 1999), dim=1)
y_test = model(x_test)

# Plot the results
plt.plot(x[0:N_total], y[0:N_total], 'bo', label='True data')
plt.plot(x_test, y_test.detach().numpy(), 'r', label='Model
prediction')
plt.legend()
plt.show()
```

```
Epoch: 001, Train Loss: 56.2691040, Test Loss: 54.2276945
Epoch: 002, Train Loss: 0.4105022, Test Loss: 0.3816116
Epoch: 003, Train Loss: 2.4412567, Test Loss: 1.9652464
Epoch: 004, Train Loss: 0.3370843, Test Loss: 0.2396506
Epoch: 005, Train Loss: 0.2370436, Test Loss: 0.2311195
Epoch: 006, Train Loss: 0.1464007, Test Loss: 0.1341241
Epoch: 007, Train Loss: 0.1239662, Test Loss: 0.1118252
Epoch: 008, Train Loss: 0.1073626, Test Loss: 0.1034263
Epoch: 009, Train Loss: 0.0966345, Test Loss: 0.0970294
Epoch: 010, Train Loss: 0.0872553, Test Loss: 0.0897591
Epoch: 011, Train Loss: 0.0792265, Test Loss: 0.0830397
Epoch: 012, Train Loss: 0.0721541, Test Loss: 0.0765846
Epoch: 013, Train Loss: 0.0658721, Test Loss: 0.0703609
Epoch: 014, Train Loss: 0.0602676, Test Loss: 0.0646378
Epoch: 015, Train Loss: 0.0552482, Test Loss: 0.0592907
Epoch: 016, Train Loss: 0.0507279, Test Loss: 0.0542920
Epoch: 017, Train Loss: 0.0466342, Test Loss: 0.0497256
Epoch: 018, Train Loss: 0.0429211, Test Loss: 0.0455847
Epoch: 019, Train Loss: 0.0395484, Test Loss: 0.0418451
Epoch: 020, Train Loss: 0.0364827, Test Loss: 0.0384666
Epoch: 021, Train Loss: 0.0336970, Test Loss: 0.0354084
Epoch: 022, Train Loss: 0.0311635, Test Loss: 0.0326435
Epoch: 023, Train Loss: 0.0288581, Test Loss: 0.0301402
Epoch: 024, Train Loss: 0.0267556, Test Loss: 0.0278737
Epoch: 025, Train Loss: 0.0248367, Test Loss: 0.0258231
Epoch: 026, Train Loss: 0.0230815, Test Loss: 0.0239607
Epoch: 027, Train Loss: 0.0214760, Test Loss: 0.0222691
Epoch: 028, Train Loss: 0.0200044, Test Loss: 0.0207338
Epoch: 029, Train Loss: 0.0186555, Test Loss: 0.0193401
```

```
Epoch: 030, Train Loss: 0.0174163, Test Loss: 0.0180688
Epoch: 031, Train Loss: 0.0162783, Test Loss: 0.0169079
Epoch: 032, Train Loss: 0.0152326, Test Loss: 0.0158442
Epoch: 033, Train Loss: 0.0142707, Test Loss: 0.0148664
Epoch: 034, Train Loss: 0.0133845, Test Loss: 0.0139693
Epoch: 035, Train Loss: 0.0125673, Test Loss: 0.0131427
Epoch: 036, Train Loss: 0.0118132, Test Loss: 0.0123794
Epoch: 037, Train Loss: 0.0111169, Test Loss: 0.0116740
Epoch: 038, Train Loss: 0.0104719, Test Loss: 0.0110181
Epoch: 039, Train Loss: 0.0098745, Test Loss: 0.0104100
Epoch: 040, Train Loss: 0.0093202, Test Loss: 0.0098442
Epoch: 041, Train Loss: 0.0088056, Test Loss: 0.0093153
Epoch: 042, Train Loss: 0.0083272, Test Loss: 0.0088192
Epoch: 043, Train Loss: 0.0078820, Test Loss: 0.0083534
Epoch: 044, Train Loss: 0.0074675, Test Loss: 0.0079162
Epoch: 045, Train Loss: 0.0070800, Test Loss: 0.0075051
Epoch: 046, Train Loss: 0.0067175, Test Loss: 0.0071191
Epoch: 047, Train Loss: 0.0063783, Test Loss: 0.0067554
Epoch: 048, Train Loss: 0.0060614, Test Loss: 0.0064129
Epoch: 049, Train Loss: 0.0057648, Test Loss: 0.0060897
Epoch: 050, Train Loss: 0.0054865, Test Loss: 0.0057849
Epoch: 051, Train Loss: 0.0052253, Test Loss: 0.0054977
Epoch: 052, Train Loss: 0.0049806, Test Loss: 0.0052282
Epoch: 053, Train Loss: 0.0047508, Test Loss: 0.0049751
Epoch: 054, Train Loss: 0.0045350, Test Loss: 0.0047357
Epoch: 055, Train Loss: 0.0043327, Test Loss: 0.0045102
Epoch: 056, Train Loss: 0.0041436, Test Loss: 0.0042988
Epoch: 057, Train Loss: 0.0039664, Test Loss: 0.0040999
Epoch: 058, Train Loss: 0.0037994, Test Loss: 0.0039132
Epoch: 059, Train Loss: 0.0036429, Test Loss: 0.0037387
Epoch: 060, Train Loss: 0.0034962, Test Loss: 0.0035750
Epoch: 061, Train Loss: 0.0033584, Test Loss: 0.0034210
Epoch: 062, Train Loss: 0.0032289, Test Loss: 0.0032763
Epoch: 063, Train Loss: 0.0031068, Test Loss: 0.0031411
Epoch: 064, Train Loss: 0.0029911, Test Loss: 0.0030154
Epoch: 065, Train Loss: 0.0028821, Test Loss: 0.0028971
Epoch: 066, Train Loss: 0.0027796, Test Loss: 0.0027863
Epoch: 067, Train Loss: 0.0026830, Test Loss: 0.0026817
Epoch: 068, Train Loss: 0.0025916, Test Loss: 0.0025827
Epoch: 069, Train Loss: 0.0025055, Test Loss: 0.0024902
Epoch: 070, Train Loss: 0.0024243, Test Loss: 0.0023950
Epoch: 071, Train Loss: 0.0023482, Test Loss: 0.0023057
Epoch: 072, Train Loss: 0.0022759, Test Loss: 0.0022201
Epoch: 073, Train Loss: 0.0022070, Test Loss: 0.0021394
Epoch: 074, Train Loss: 0.0021416, Test Loss: 0.0020635
Epoch: 075, Train Loss: 0.0020795, Test Loss: 0.0019912
Epoch: 076, Train Loss: 0.0020203, Test Loss: 0.0019224
Epoch: 077, Train Loss: 0.0019641, Test Loss: 0.0018569
Epoch: 078, Train Loss: 0.0019101, Test Loss: 0.0017936
```

```
Epoch: 079, Train Loss: 0.0018579, Test Loss: 0.0017323
Epoch: 080, Train Loss: 0.0018076, Test Loss: 0.0016742
Epoch: 081, Train Loss: 0.0017590, Test Loss: 0.0016183
Epoch: 082, Train Loss: 0.0017117, Test Loss: 0.0015642
Epoch: 083, Train Loss: 0.0016660, Test Loss: 0.0015128
Epoch: 084, Train Loss: 0.0016223, Test Loss: 0.0014626
Epoch: 085, Train Loss: 0.0015804, Test Loss: 0.0014136
Epoch: 086, Train Loss: 0.0015402, Test Loss: 0.0013675
Epoch: 087, Train Loss: 0.0015007, Test Loss: 0.0013222
Epoch: 088, Train Loss: 0.0014620, Test Loss: 0.0012785
Epoch: 089, Train Loss: 0.0014246, Test Loss: 0.0012374
Epoch: 090, Train Loss: 0.0013883, Test Loss: 0.0011981
Epoch: 091, Train Loss: 0.0013533, Test Loss: 0.0011611
Epoch: 092, Train Loss: 0.0013196, Test Loss: 0.0011257
Epoch: 093, Train Loss: 0.0012875, Test Loss: 0.0010922
Epoch: 094, Train Loss: 0.0012568, Test Loss: 0.0010610
Epoch: 095, Train Loss: 0.0012277, Test Loss: 0.0010316
Epoch: 096, Train Loss: 0.0011999, Test Loss: 0.0010039
Epoch: 097, Train Loss: 0.0011737, Test Loss: 0.0009786
Epoch: 098, Train Loss: 0.0011487, Test Loss: 0.0009550
Epoch: 099, Train Loss: 0.0011251, Test Loss: 0.0009329
Epoch: 100, Train Loss: 0.0011027, Test Loss: 0.0009119
Epoch: 101, Train Loss: 0.0010814, Test Loss: 0.0008923
Epoch: 102, Train Loss: 0.0010612, Test Loss: 0.0008737
Epoch: 103, Train Loss: 0.0010418, Test Loss: 0.0008565
Epoch: 104, Train Loss: 0.0010234, Test Loss: 0.0008400
Epoch: 105, Train Loss: 0.0010060, Test Loss: 0.0008246
Epoch: 106, Train Loss: 0.0009893, Test Loss: 0.0008098
Epoch: 107, Train Loss: 0.0009735, Test Loss: 0.0007962
Epoch: 108, Train Loss: 0.0009584, Test Loss: 0.0007833
Epoch: 109, Train Loss: 0.0009439, Test Loss: 0.0007713
Epoch: 110, Train Loss: 0.0009298, Test Loss: 0.0007596
Epoch: 111, Train Loss: 0.0009162, Test Loss: 0.0007490
Epoch: 112, Train Loss: 0.0009030, Test Loss: 0.0007388
Epoch: 113, Train Loss: 0.0008902, Test Loss: 0.0007289
Epoch: 114, Train Loss: 0.0008779, Test Loss: 0.0007199
Epoch: 115, Train Loss: 0.0008660, Test Loss: 0.0007112
Epoch: 116, Train Loss: 0.0008544, Test Loss: 0.0007030
Epoch: 117, Train Loss: 0.0008431, Test Loss: 0.0006953
Epoch: 118, Train Loss: 0.0008321, Test Loss: 0.0006876
Epoch: 119, Train Loss: 0.0008214, Test Loss: 0.0006806
Epoch: 120, Train Loss: 0.0008110, Test Loss: 0.0006737
Epoch: 121, Train Loss: 0.0008009, Test Loss: 0.0006670
Epoch: 122, Train Loss: 0.0007910, Test Loss: 0.0006606
Epoch: 123, Train Loss: 0.0007815, Test Loss: 0.0006546
Epoch: 124, Train Loss: 0.0007721, Test Loss: 0.0006486
Epoch: 125, Train Loss: 0.0007629, Test Loss: 0.0006428
Epoch: 126, Train Loss: 0.0007540, Test Loss: 0.0006371
Epoch: 127, Train Loss: 0.0007452, Test Loss: 0.0006314
```

```
Epoch: 128, Train Loss: 0.0007366, Test Loss: 0.0006260
Epoch: 129, Train Loss: 0.0007282, Test Loss: 0.0006208
Epoch: 130, Train Loss: 0.0007201, Test Loss: 0.0006156
Epoch: 131, Train Loss: 0.0007121, Test Loss: 0.0006107
Epoch: 132, Train Loss: 0.0007041, Test Loss: 0.0006058
Epoch: 133, Train Loss: 0.0006964, Test Loss: 0.0006010
Epoch: 134, Train Loss: 0.0006888, Test Loss: 0.0005964
Epoch: 135, Train Loss: 0.0006813, Test Loss: 0.0005919
Epoch: 136, Train Loss: 0.0006739, Test Loss: 0.0005873
Epoch: 137, Train Loss: 0.0006666, Test Loss: 0.0005827
Epoch: 138, Train Loss: 0.0006594, Test Loss: 0.0005782
Epoch: 139, Train Loss: 0.0006524, Test Loss: 0.0005737
Epoch: 140, Train Loss: 0.0006454, Test Loss: 0.0005692
Epoch: 141, Train Loss: 0.0006385, Test Loss: 0.0005647
Epoch: 142, Train Loss: 0.0006317, Test Loss: 0.0005604
Epoch: 143, Train Loss: 0.0006250, Test Loss: 0.0005561
Epoch: 144, Train Loss: 0.0006183, Test Loss: 0.0005519
Epoch: 145, Train Loss: 0.0006118, Test Loss: 0.0005476
Epoch: 146, Train Loss: 0.0006053, Test Loss: 0.0005433
Epoch: 147, Train Loss: 0.0005989, Test Loss: 0.0005390
Epoch: 148, Train Loss: 0.0005925, Test Loss: 0.0005348
Epoch: 149, Train Loss: 0.0005863, Test Loss: 0.0005305
Epoch: 150, Train Loss: 0.0005801, Test Loss: 0.0005262
Epoch: 151, Train Loss: 0.0005739, Test Loss: 0.0005221
Epoch: 152, Train Loss: 0.0005678, Test Loss: 0.0005179
Epoch: 153, Train Loss: 0.0005618, Test Loss: 0.0005137
Epoch: 154, Train Loss: 0.0005557, Test Loss: 0.0005095
Epoch: 155, Train Loss: 0.0005498, Test Loss: 0.0005053
Epoch: 156, Train Loss: 0.0005438, Test Loss: 0.0005010
Epoch: 157, Train Loss: 0.0005380, Test Loss: 0.0004966
Epoch: 158, Train Loss: 0.0005322, Test Loss: 0.0004923
Epoch: 159, Train Loss: 0.0005265, Test Loss: 0.0004879
Epoch: 160, Train Loss: 0.0005208, Test Loss: 0.0004835
Epoch: 161, Train Loss: 0.0005152, Test Loss: 0.0004791
Epoch: 162, Train Loss: 0.0005095, Test Loss: 0.0004748
Epoch: 163, Train Loss: 0.0005040, Test Loss: 0.0004704
Epoch: 164, Train Loss: 0.0004985, Test Loss: 0.0004660
Epoch: 165, Train Loss: 0.0004930, Test Loss: 0.0004617
Epoch: 166, Train Loss: 0.0004875, Test Loss: 0.0004573
Epoch: 167, Train Loss: 0.0004821, Test Loss: 0.0004529
Epoch: 168, Train Loss: 0.0004767, Test Loss: 0.0004485
Epoch: 169, Train Loss: 0.0004713, Test Loss: 0.0004440
Epoch: 170, Train Loss: 0.0004661, Test Loss: 0.0004394
Epoch: 171, Train Loss: 0.0004608, Test Loss: 0.0004349
Epoch: 172, Train Loss: 0.0004556, Test Loss: 0.0004303
Epoch: 173, Train Loss: 0.0004505, Test Loss: 0.0004259
Epoch: 174, Train Loss: 0.0004453, Test Loss: 0.0004214
Epoch: 175, Train Loss: 0.0004402, Test Loss: 0.0004170
Epoch: 176, Train Loss: 0.0004351, Test Loss: 0.0004125
```

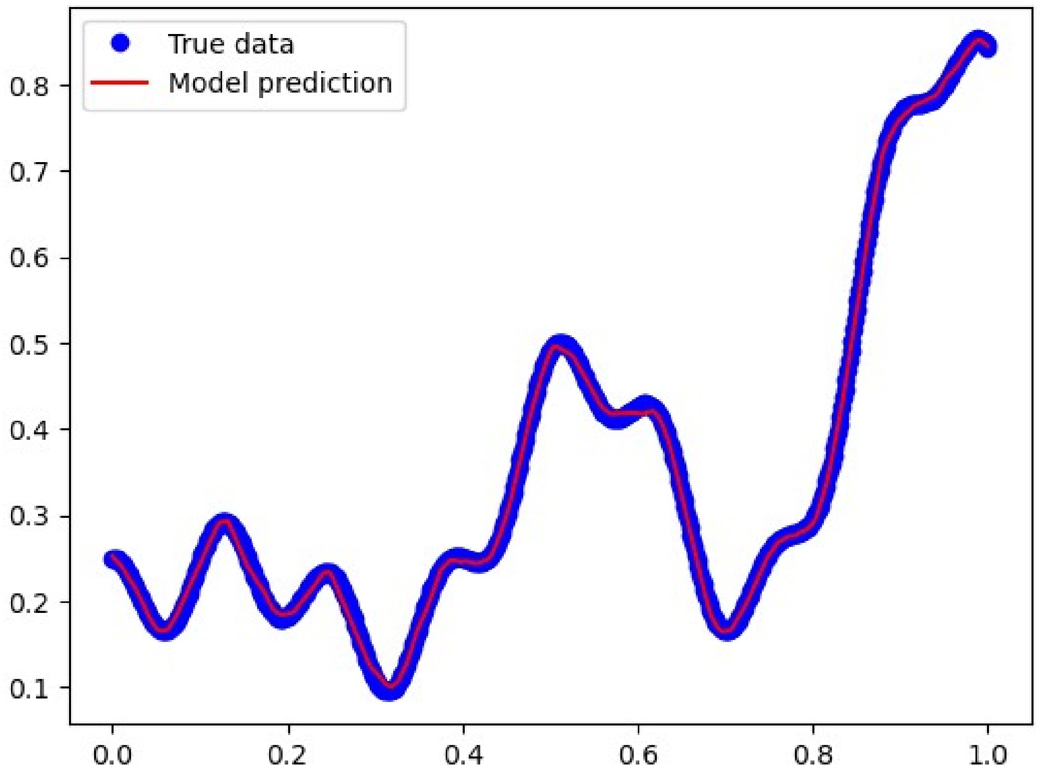
```
Epoch: 177, Train Loss: 0.0004301, Test Loss: 0.0004082
Epoch: 178, Train Loss: 0.0004251, Test Loss: 0.0004038
Epoch: 179, Train Loss: 0.0004202, Test Loss: 0.0003995
Epoch: 180, Train Loss: 0.0004153, Test Loss: 0.0003952
Epoch: 181, Train Loss: 0.0004105, Test Loss: 0.0003911
Epoch: 182, Train Loss: 0.0004056, Test Loss: 0.0003870
Epoch: 183, Train Loss: 0.0004009, Test Loss: 0.0003829
Epoch: 184, Train Loss: 0.0003962, Test Loss: 0.0003789
Epoch: 185, Train Loss: 0.0003915, Test Loss: 0.0003749
Epoch: 186, Train Loss: 0.0003869, Test Loss: 0.0003709
Epoch: 187, Train Loss: 0.0003823, Test Loss: 0.0003669
Epoch: 188, Train Loss: 0.0003777, Test Loss: 0.0003630
Epoch: 189, Train Loss: 0.0003732, Test Loss: 0.0003591
Epoch: 190, Train Loss: 0.0003688, Test Loss: 0.0003551
Epoch: 191, Train Loss: 0.0003644, Test Loss: 0.0003513
Epoch: 192, Train Loss: 0.0003600, Test Loss: 0.0003474
Epoch: 193, Train Loss: 0.0003557, Test Loss: 0.0003436
Epoch: 194, Train Loss: 0.0003514, Test Loss: 0.0003398
Epoch: 195, Train Loss: 0.0003471, Test Loss: 0.0003360
Epoch: 196, Train Loss: 0.0003429, Test Loss: 0.0003322
Epoch: 197, Train Loss: 0.0003387, Test Loss: 0.0003285
Epoch: 198, Train Loss: 0.0003345, Test Loss: 0.0003248
Epoch: 199, Train Loss: 0.0003304, Test Loss: 0.0003211
Epoch: 200, Train Loss: 0.0003263, Test Loss: 0.0003176
Epoch: 201, Train Loss: 0.0003223, Test Loss: 0.0003140
Epoch: 202, Train Loss: 0.0003183, Test Loss: 0.0003106
Epoch: 203, Train Loss: 0.0003144, Test Loss: 0.0003070
Epoch: 204, Train Loss: 0.0003105, Test Loss: 0.0003036
Epoch: 205, Train Loss: 0.0003066, Test Loss: 0.0003002
Epoch: 206, Train Loss: 0.0003028, Test Loss: 0.0002968
Epoch: 207, Train Loss: 0.0002990, Test Loss: 0.0002934
Epoch: 208, Train Loss: 0.0002952, Test Loss: 0.0002900
Epoch: 209, Train Loss: 0.0002915, Test Loss: 0.0002867
Epoch: 210, Train Loss: 0.0002878, Test Loss: 0.0002833
Epoch: 211, Train Loss: 0.0002842, Test Loss: 0.0002799
Epoch: 212, Train Loss: 0.0002806, Test Loss: 0.0002766
Epoch: 213, Train Loss: 0.0002770, Test Loss: 0.0002733
Epoch: 214, Train Loss: 0.0002735, Test Loss: 0.0002700
Epoch: 215, Train Loss: 0.0002699, Test Loss: 0.0002668
Epoch: 216, Train Loss: 0.0002665, Test Loss: 0.0002636
Epoch: 217, Train Loss: 0.0002630, Test Loss: 0.0002604
Epoch: 218, Train Loss: 0.0002596, Test Loss: 0.0002573
Epoch: 219, Train Loss: 0.0002562, Test Loss: 0.0002542
Epoch: 220, Train Loss: 0.0002529, Test Loss: 0.0002511
Epoch: 221, Train Loss: 0.0002496, Test Loss: 0.0002481
Epoch: 222, Train Loss: 0.0002463, Test Loss: 0.0002452
Epoch: 223, Train Loss: 0.0002431, Test Loss: 0.0002423
Epoch: 224, Train Loss: 0.0002399, Test Loss: 0.0002394
Epoch: 225, Train Loss: 0.0002367, Test Loss: 0.0002366
```

```
Epoch: 226, Train Loss: 0.0002335, Test Loss: 0.0002336
Epoch: 227, Train Loss: 0.0002304, Test Loss: 0.0002309
Epoch: 228, Train Loss: 0.0002273, Test Loss: 0.0002281
Epoch: 229, Train Loss: 0.0002242, Test Loss: 0.0002252
Epoch: 230, Train Loss: 0.0002212, Test Loss: 0.0002225
Epoch: 231, Train Loss: 0.0002182, Test Loss: 0.0002197
Epoch: 232, Train Loss: 0.0002152, Test Loss: 0.0002169
Epoch: 233, Train Loss: 0.0002123, Test Loss: 0.0002143
Epoch: 234, Train Loss: 0.0002094, Test Loss: 0.0002116
Epoch: 235, Train Loss: 0.0002065, Test Loss: 0.0002090
Epoch: 236, Train Loss: 0.0002037, Test Loss: 0.0002063
Epoch: 237, Train Loss: 0.0002008, Test Loss: 0.0002037
Epoch: 238, Train Loss: 0.0001980, Test Loss: 0.0002011
Epoch: 239, Train Loss: 0.0001953, Test Loss: 0.0001986
Epoch: 240, Train Loss: 0.0001925, Test Loss: 0.0001962
Epoch: 241, Train Loss: 0.0001898, Test Loss: 0.0001937
Epoch: 242, Train Loss: 0.0001871, Test Loss: 0.0001912
Epoch: 243, Train Loss: 0.0001845, Test Loss: 0.0001890
Epoch: 244, Train Loss: 0.0001818, Test Loss: 0.0001864
Epoch: 245, Train Loss: 0.0001793, Test Loss: 0.0001841
Epoch: 246, Train Loss: 0.0001767, Test Loss: 0.0001818
Epoch: 247, Train Loss: 0.0001742, Test Loss: 0.0001795
Epoch: 248, Train Loss: 0.0001717, Test Loss: 0.0001772
Epoch: 249, Train Loss: 0.0001692, Test Loss: 0.0001750
Epoch: 250, Train Loss: 0.0001668, Test Loss: 0.0001727
Epoch: 251, Train Loss: 0.0001643, Test Loss: 0.0001704
Epoch: 252, Train Loss: 0.0001619, Test Loss: 0.0001683
Epoch: 253, Train Loss: 0.0001595, Test Loss: 0.0001661
Epoch: 254, Train Loss: 0.0001572, Test Loss: 0.0001639
Epoch: 255, Train Loss: 0.0001548, Test Loss: 0.0001618
Epoch: 256, Train Loss: 0.0001526, Test Loss: 0.0001597
Epoch: 257, Train Loss: 0.0001503, Test Loss: 0.0001576
Epoch: 258, Train Loss: 0.0001481, Test Loss: 0.0001556
Epoch: 259, Train Loss: 0.0001459, Test Loss: 0.0001536
Epoch: 260, Train Loss: 0.0001437, Test Loss: 0.0001517
Epoch: 261, Train Loss: 0.0001416, Test Loss: 0.0001498
Epoch: 262, Train Loss: 0.0001395, Test Loss: 0.0001478
Epoch: 263, Train Loss: 0.0001374, Test Loss: 0.0001459
Epoch: 264, Train Loss: 0.0001353, Test Loss: 0.0001440
Epoch: 265, Train Loss: 0.0001333, Test Loss: 0.0001422
Epoch: 266, Train Loss: 0.0001313, Test Loss: 0.0001404
Epoch: 267, Train Loss: 0.0001294, Test Loss: 0.0001386
Epoch: 268, Train Loss: 0.0001274, Test Loss: 0.0001368
Epoch: 269, Train Loss: 0.0001255, Test Loss: 0.0001351
Epoch: 270, Train Loss: 0.0001236, Test Loss: 0.0001333
Epoch: 271, Train Loss: 0.0001217, Test Loss: 0.0001315
Epoch: 272, Train Loss: 0.0001199, Test Loss: 0.0001299
Epoch: 273, Train Loss: 0.0001181, Test Loss: 0.0001282
Epoch: 274, Train Loss: 0.0001164, Test Loss: 0.0001265
```

```
Epoch: 275, Train Loss: 0.0001146, Test Loss: 0.0001249
Epoch: 276, Train Loss: 0.0001129, Test Loss: 0.0001233
Epoch: 277, Train Loss: 0.0001112, Test Loss: 0.0001217
Epoch: 278, Train Loss: 0.0001095, Test Loss: 0.0001202
Epoch: 279, Train Loss: 0.0001079, Test Loss: 0.0001186
Epoch: 280, Train Loss: 0.0001063, Test Loss: 0.0001172
Epoch: 281, Train Loss: 0.0001047, Test Loss: 0.0001156
Epoch: 282, Train Loss: 0.0001032, Test Loss: 0.0001142
Epoch: 283, Train Loss: 0.0001016, Test Loss: 0.0001127
Epoch: 284, Train Loss: 0.0001001, Test Loss: 0.0001113
Epoch: 285, Train Loss: 0.0000987, Test Loss: 0.0001098
Epoch: 286, Train Loss: 0.0000972, Test Loss: 0.0001084
Epoch: 287, Train Loss: 0.0000958, Test Loss: 0.0001070
Epoch: 288, Train Loss: 0.0000944, Test Loss: 0.0001056
Epoch: 289, Train Loss: 0.0000930, Test Loss: 0.0001043
Epoch: 290, Train Loss: 0.0000916, Test Loss: 0.0001029
Epoch: 291, Train Loss: 0.0000903, Test Loss: 0.0001015
Epoch: 292, Train Loss: 0.0000890, Test Loss: 0.0001002
Epoch: 293, Train Loss: 0.0000877, Test Loss: 0.0000989
Epoch: 294, Train Loss: 0.0000865, Test Loss: 0.0000977
Epoch: 295, Train Loss: 0.0000853, Test Loss: 0.0000964
Epoch: 296, Train Loss: 0.0000841, Test Loss: 0.0000952
Epoch: 297, Train Loss: 0.0000829, Test Loss: 0.0000940
Epoch: 298, Train Loss: 0.0000818, Test Loss: 0.0000929
Epoch: 299, Train Loss: 0.0000807, Test Loss: 0.0000918
Epoch: 300, Train Loss: 0.0000796, Test Loss: 0.0000907
Epoch: 301, Train Loss: 0.0000786, Test Loss: 0.0000896
Epoch: 302, Train Loss: 0.0000776, Test Loss: 0.0000886
Epoch: 303, Train Loss: 0.0000766, Test Loss: 0.0000875
Epoch: 304, Train Loss: 0.0000757, Test Loss: 0.0000866
Epoch: 305, Train Loss: 0.0000748, Test Loss: 0.0000857
Epoch: 306, Train Loss: 0.0000739, Test Loss: 0.0000847
Epoch: 307, Train Loss: 0.0000731, Test Loss: 0.0000839
Epoch: 308, Train Loss: 0.0000723, Test Loss: 0.0000831
Epoch: 309, Train Loss: 0.0000716, Test Loss: 0.0000823
Epoch: 310, Train Loss: 0.0000709, Test Loss: 0.0000816
Epoch: 311, Train Loss: 0.0000703, Test Loss: 0.0000809
Epoch: 312, Train Loss: 0.0000698, Test Loss: 0.0000803
Epoch: 313, Train Loss: 0.0000692, Test Loss: 0.0000798
Epoch: 314, Train Loss: 0.0000688, Test Loss: 0.0000793
Epoch: 315, Train Loss: 0.0000685, Test Loss: 0.0000789
Epoch: 316, Train Loss: 0.0000683, Test Loss: 0.0000787
Epoch: 317, Train Loss: 0.0000682, Test Loss: 0.0000786
Epoch: 318, Train Loss: 0.0000682, Test Loss: 0.0000786
Epoch: 319, Train Loss: 0.0000683, Test Loss: 0.0000787
Epoch: 320, Train Loss: 0.0000686, Test Loss: 0.0000790
Epoch: 321, Train Loss: 0.0000691, Test Loss: 0.0000794
Epoch: 322, Train Loss: 0.0000697, Test Loss: 0.0000801
Epoch: 323, Train Loss: 0.0000705, Test Loss: 0.0000809
```

```
Epoch: 324, Train Loss: 0.0000714, Test Loss: 0.0000818
Epoch: 325, Train Loss: 0.0000725, Test Loss: 0.0000830
Epoch: 326, Train Loss: 0.0000738, Test Loss: 0.0000843
Epoch: 327, Train Loss: 0.0000752, Test Loss: 0.0000857
Epoch: 328, Train Loss: 0.0000768, Test Loss: 0.0000873
Epoch: 329, Train Loss: 0.0000784, Test Loss: 0.0000889
Epoch: 330, Train Loss: 0.0000800, Test Loss: 0.0000906
Epoch: 331, Train Loss: 0.0000816, Test Loss: 0.0000923
Epoch: 332, Train Loss: 0.0000831, Test Loss: 0.0000939
Epoch: 333, Train Loss: 0.0000845, Test Loss: 0.0000953
Epoch: 334, Train Loss: 0.0000856, Test Loss: 0.0000965
Epoch: 335, Train Loss: 0.0000863, Test Loss: 0.0000973
Epoch: 336, Train Loss: 0.0000867, Test Loss: 0.0000977
Epoch: 337, Train Loss: 0.0000866, Test Loss: 0.0000976
Epoch: 338, Train Loss: 0.0000860, Test Loss: 0.0000971
Epoch: 339, Train Loss: 0.0000850, Test Loss: 0.0000961
Epoch: 340, Train Loss: 0.0000837, Test Loss: 0.0000947
Epoch: 341, Train Loss: 0.0000820, Test Loss: 0.0000929
Epoch: 342, Train Loss: 0.0000800, Test Loss: 0.0000909
Epoch: 343, Train Loss: 0.0000778, Test Loss: 0.0000887
Epoch: 344, Train Loss: 0.0000755, Test Loss: 0.0000863
Epoch: 345, Train Loss: 0.0000731, Test Loss: 0.0000838
Epoch: 346, Train Loss: 0.0000707, Test Loss: 0.0000812
Epoch: 347, Train Loss: 0.0000683, Test Loss: 0.0000787
Epoch: 348, Train Loss: 0.0000660, Test Loss: 0.0000762
Epoch: 349, Train Loss: 0.0000639, Test Loss: 0.0000739
Epoch: 350, Train Loss: 0.0000617, Test Loss: 0.0000716
Epoch: 351, Train Loss: 0.0000597, Test Loss: 0.0000694
Epoch: 352, Train Loss: 0.0000579, Test Loss: 0.0000674
Epoch: 353, Train Loss: 0.0000562, Test Loss: 0.0000655
Epoch: 354, Train Loss: 0.0000545, Test Loss: 0.0000637
Epoch: 355, Train Loss: 0.0000530, Test Loss: 0.0000619
Epoch: 356, Train Loss: 0.0000516, Test Loss: 0.0000604
Epoch: 357, Train Loss: 0.0000503, Test Loss: 0.0000588
Epoch: 358, Train Loss: 0.0000490, Test Loss: 0.0000573
Epoch: 359, Train Loss: 0.0000478, Test Loss: 0.0000559
Epoch: 360, Train Loss: 0.0000467, Test Loss: 0.0000546
Epoch: 361, Train Loss: 0.0000455, Test Loss: 0.0000533
Epoch: 362, Train Loss: 0.0000445, Test Loss: 0.0000520
Epoch: 363, Train Loss: 0.0000434, Test Loss: 0.0000508
Epoch: 364, Train Loss: 0.0000425, Test Loss: 0.0000496
Epoch: 365, Train Loss: 0.0000415, Test Loss: 0.0000485
Epoch: 366, Train Loss: 0.0000407, Test Loss: 0.0000474
Epoch: 367, Train Loss: 0.0000398, Test Loss: 0.0000464
Epoch: 368, Train Loss: 0.0000390, Test Loss: 0.0000454
Epoch: 369, Train Loss: 0.0000382, Test Loss: 0.0000444
Epoch: 370, Train Loss: 0.0000374, Test Loss: 0.0000434
Epoch: 371, Train Loss: 0.0000366, Test Loss: 0.0000424
Epoch: 372, Train Loss: 0.0000359, Test Loss: 0.0000415
```

```
Epoch: 373, Train Loss: 0.0000351, Test Loss: 0.0000406
Epoch: 374, Train Loss: 0.0000344, Test Loss: 0.0000396
Epoch: 375, Train Loss: 0.0000337, Test Loss: 0.0000387
Epoch: 376, Train Loss: 0.0000330, Test Loss: 0.0000378
Epoch: 377, Train Loss: 0.0000323, Test Loss: 0.0000369
Epoch: 378, Train Loss: 0.0000316, Test Loss: 0.0000361
Epoch: 379, Train Loss: 0.0000310, Test Loss: 0.0000352
Epoch: 380, Train Loss: 0.0000303, Test Loss: 0.0000344
Epoch: 381, Train Loss: 0.0000297, Test Loss: 0.0000336
Epoch: 382, Train Loss: 0.0000291, Test Loss: 0.0000328
Epoch: 383, Train Loss: 0.0000285, Test Loss: 0.0000320
Epoch: 384, Train Loss: 0.0000279, Test Loss: 0.0000312
Epoch: 385, Train Loss: 0.0000274, Test Loss: 0.0000304
Epoch: 386, Train Loss: 0.0000268, Test Loss: 0.0000297
Epoch: 387, Train Loss: 0.0000263, Test Loss: 0.0000290
Epoch: 388, Train Loss: 0.0000258, Test Loss: 0.0000283
Epoch: 389, Train Loss: 0.0000253, Test Loss: 0.0000276
Epoch: 390, Train Loss: 0.0000249, Test Loss: 0.0000270
Epoch: 391, Train Loss: 0.0000244, Test Loss: 0.0000263
Epoch: 392, Train Loss: 0.0000240, Test Loss: 0.0000257
Epoch: 393, Train Loss: 0.0000236, Test Loss: 0.0000252
Epoch: 394, Train Loss: 0.0000232, Test Loss: 0.0000246
Epoch: 395, Train Loss: 0.0000229, Test Loss: 0.0000241
Epoch: 396, Train Loss: 0.0000226, Test Loss: 0.0000237
Epoch: 397, Train Loss: 0.0000223, Test Loss: 0.0000232
Epoch: 398, Train Loss: 0.0000219, Test Loss: 0.0000228
Epoch: 399, Train Loss: 0.0000217, Test Loss: 0.0000224
Epoch: 400, Train Loss: 0.0000214, Test Loss: 0.0000220
```



Problem 4.2 Determine appropriate mini-batch size = 32, epoch = 500

```
# ################## Part 1: Load data and create batch
##################
N_total = 600
N_train = 500
x = torch.unsqueeze(torch.linspace(0, 1, N_total), dim=1)
r = torch.randperm(N_total)
x = x[r, :]
y = 0.2 + 0.4 * torch.pow(x, 2) + 0.3 * x * torch.sin(15 * x) + 0.05 *
torch.cos(50 * x)

class CustomDataset(Dataset):
    def __init__(self, x, y):
        self.y = y
        self.x = x

    def __len__(self):
        return len(self.y)

    def __getitem__(self, idx):
        y1 = self.y[idx]
        x1 = self.x[idx]
        return (x1, y1)

# Change batch_size here to test different values
```

```
batch_size = 32  # Experiment with different batch sizes: 32, 64, 128
trainset = CustomDataset(x[0:N_train, :], y[0:N_train, :])
testset = CustomDataset(x[N_train:N_total, :], y[N_train:N_total, :])
train_loader = torch.utils.data.DataLoader(trainset,
batch_size=batch_size)
test_loader = torch.utils.data.DataLoader(testset,
batch_size=batch_size)

# ################## Part 2: Define Model ##################
model = nn.Sequential(
    nn.Linear(1, 1024, bias=True),
    nn.ReLU(),
    nn.Linear(1024, 1, bias=True)
)

def init_weights(m):
    if isinstance(m, nn.Linear):
        m.weight.data.uniform_(-1, 1)
        m.bias.data.uniform_(-1, 1)

model.apply(init_weights)

# ################## Part 3: Define Loss and Optimizer
##################
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

# ################## Part 4: Training and Testing ##################
def train_NN():
    model.train()
    for images, labels in train_loader:
        out = model(images)
        loss = criterion(out, labels)
        loss.backward()
        optimizer.step()
        optimizer.zero_grad()
    return loss

def test_NN(loader):
    model.eval()
    loss = 0
    with torch.no_grad():
        for images, labels in loader:
            out = model(images)
            loss += criterion(out, labels).item()
    loss = loss / len(loader)
    return loss

# Experiment with different numbers of epochs
N_epoch = 500  # Try values like 100, 200, 500
```

```python
train_loss = np.zeros((N_epoch, 1))
test_loss = np.zeros((N_epoch, 1))

for epoch in range(N_epoch):
    train_NN()
    train_loss[epoch, 0] = test_NN(train_loader)
    test_loss[epoch, 0] = test_NN(test_loader)
    print(f'Epoch: {epoch+1:03d}, Train Loss: {train_loss[epoch,
0]:.7f}, Test Loss: {test_loss[epoch, 0]:.7f}')

# ################## Final Prediction ##################
x_test = torch.unsqueeze(torch.linspace(0, 1, 1999), dim=1)
y_test = model(x_test)

# Plot the results
plt.plot(x[0:N_total], y[0:N_total], 'bo', label='True data')
plt.plot(x_test, y_test.detach().numpy(), 'r', label='Model
prediction')
plt.legend()
plt.show()
```

```
Epoch: 001, Train Loss: 1.8591078, Test Loss: 1.8779527
Epoch: 002, Train Loss: 3.3536461, Test Loss: 3.5683156
Epoch: 003, Train Loss: 0.6762136, Test Loss: 0.6476456
Epoch: 004, Train Loss: 0.3699355, Test Loss: 0.3676183
Epoch: 005, Train Loss: 0.2734823, Test Loss: 0.3078416
Epoch: 006, Train Loss: 0.2020788, Test Loss: 0.2080572
Epoch: 007, Train Loss: 0.1507670, Test Loss: 0.1631345
Epoch: 008, Train Loss: 0.1131928, Test Loss: 0.1192604
Epoch: 009, Train Loss: 0.0844115, Test Loss: 0.0875499
Epoch: 010, Train Loss: 0.0629654, Test Loss: 0.0637244
Epoch: 011, Train Loss: 0.0472580, Test Loss: 0.0465967
Epoch: 012, Train Loss: 0.0359955, Test Loss: 0.0346528
Epoch: 013, Train Loss: 0.0280716, Test Loss: 0.0263436
Epoch: 014, Train Loss: 0.0225176, Test Loss: 0.0206735
Epoch: 015, Train Loss: 0.0186077, Test Loss: 0.0167733
Epoch: 016, Train Loss: 0.0158094, Test Loss: 0.0140455
Epoch: 017, Train Loss: 0.0137526, Test Loss: 0.0120892
Epoch: 018, Train Loss: 0.0121942, Test Loss: 0.0106317
Epoch: 019, Train Loss: 0.0109726, Test Loss: 0.0095144
Epoch: 020, Train Loss: 0.0099813, Test Loss: 0.0086213
Epoch: 021, Train Loss: 0.0091465, Test Loss: 0.0078800
Epoch: 022, Train Loss: 0.0084258, Test Loss: 0.0072504
Epoch: 023, Train Loss: 0.0077910, Test Loss: 0.0067061
Epoch: 024, Train Loss: 0.0072258, Test Loss: 0.0062259
Epoch: 025, Train Loss: 0.0067190, Test Loss: 0.0058017
Epoch: 026, Train Loss: 0.0062623, Test Loss: 0.0054227
Epoch: 027, Train Loss: 0.0058486, Test Loss: 0.0050836
Epoch: 028, Train Loss: 0.0054735, Test Loss: 0.0047807
Epoch: 029, Train Loss: 0.0051327, Test Loss: 0.0045074
```

```
Epoch: 030, Train Loss: 0.0048227, Test Loss: 0.0042612
Epoch: 031, Train Loss: 0.0045403, Test Loss: 0.0040395
Epoch: 032, Train Loss: 0.0042820, Test Loss: 0.0038380
Epoch: 033, Train Loss: 0.0040467, Test Loss: 0.0036555
Epoch: 034, Train Loss: 0.0038313, Test Loss: 0.0034894
Epoch: 035, Train Loss: 0.0036342, Test Loss: 0.0033396
Epoch: 036, Train Loss: 0.0034531, Test Loss: 0.0032029
Epoch: 037, Train Loss: 0.0032870, Test Loss: 0.0030781
Epoch: 038, Train Loss: 0.0031342, Test Loss: 0.0029640
Epoch: 039, Train Loss: 0.0029932, Test Loss: 0.0028602
Epoch: 040, Train Loss: 0.0028638, Test Loss: 0.0027645
Epoch: 041, Train Loss: 0.0027448, Test Loss: 0.0026763
Epoch: 042, Train Loss: 0.0026346, Test Loss: 0.0025947
Epoch: 043, Train Loss: 0.0025336, Test Loss: 0.0025183
Epoch: 044, Train Loss: 0.0024404, Test Loss: 0.0024477
Epoch: 045, Train Loss: 0.0023545, Test Loss: 0.0023832
Epoch: 046, Train Loss: 0.0022755, Test Loss: 0.0023234
Epoch: 047, Train Loss: 0.0022027, Test Loss: 0.0022678
Epoch: 048, Train Loss: 0.0021361, Test Loss: 0.0022171
Epoch: 049, Train Loss: 0.0020744, Test Loss: 0.0021698
Epoch: 050, Train Loss: 0.0020178, Test Loss: 0.0021262
Epoch: 051, Train Loss: 0.0019658, Test Loss: 0.0020859
Epoch: 052, Train Loss: 0.0019177, Test Loss: 0.0020485
Epoch: 053, Train Loss: 0.0018729, Test Loss: 0.0020134
Epoch: 054, Train Loss: 0.0018310, Test Loss: 0.0019808
Epoch: 055, Train Loss: 0.0017918, Test Loss: 0.0019499
Epoch: 056, Train Loss: 0.0017547, Test Loss: 0.0019213
Epoch: 057, Train Loss: 0.0017197, Test Loss: 0.0018938
Epoch: 058, Train Loss: 0.0016864, Test Loss: 0.0018670
Epoch: 059, Train Loss: 0.0016540, Test Loss: 0.0018414
Epoch: 060, Train Loss: 0.0016223, Test Loss: 0.0018156
Epoch: 061, Train Loss: 0.0015910, Test Loss: 0.0017893
Epoch: 062, Train Loss: 0.0015597, Test Loss: 0.0017621
Epoch: 063, Train Loss: 0.0015289, Test Loss: 0.0017344
Epoch: 064, Train Loss: 0.0014994, Test Loss: 0.0017073
Epoch: 065, Train Loss: 0.0014720, Test Loss: 0.0016823
Epoch: 066, Train Loss: 0.0014473, Test Loss: 0.0016595
Epoch: 067, Train Loss: 0.0014250, Test Loss: 0.0016396
Epoch: 068, Train Loss: 0.0014053, Test Loss: 0.0016218
Epoch: 069, Train Loss: 0.0013873, Test Loss: 0.0016060
Epoch: 070, Train Loss: 0.0013705, Test Loss: 0.0015925
Epoch: 071, Train Loss: 0.0013545, Test Loss: 0.0015798
Epoch: 072, Train Loss: 0.0013391, Test Loss: 0.0015669
Epoch: 073, Train Loss: 0.0013242, Test Loss: 0.0015551
Epoch: 074, Train Loss: 0.0013099, Test Loss: 0.0015430
Epoch: 075, Train Loss: 0.0012960, Test Loss: 0.0015313
Epoch: 076, Train Loss: 0.0012826, Test Loss: 0.0015198
Epoch: 077, Train Loss: 0.0012697, Test Loss: 0.0015088
Epoch: 078, Train Loss: 0.0012571, Test Loss: 0.0014979
```

```
Epoch: 079, Train Loss: 0.0012448, Test Loss: 0.0014874
Epoch: 080, Train Loss: 0.0012331, Test Loss: 0.0014774
Epoch: 081, Train Loss: 0.0012218, Test Loss: 0.0014680
Epoch: 082, Train Loss: 0.0012110, Test Loss: 0.0014592
Epoch: 083, Train Loss: 0.0012007, Test Loss: 0.0014508
Epoch: 084, Train Loss: 0.0011907, Test Loss: 0.0014430
Epoch: 085, Train Loss: 0.0011814, Test Loss: 0.0014356
Epoch: 086, Train Loss: 0.0011726, Test Loss: 0.0014287
Epoch: 087, Train Loss: 0.0011644, Test Loss: 0.0014223
Epoch: 088, Train Loss: 0.0011566, Test Loss: 0.0014162
Epoch: 089, Train Loss: 0.0011491, Test Loss: 0.0014103
Epoch: 090, Train Loss: 0.0011420, Test Loss: 0.0014047
Epoch: 091, Train Loss: 0.0011351, Test Loss: 0.0013993
Epoch: 092, Train Loss: 0.0011285, Test Loss: 0.0013941
Epoch: 093, Train Loss: 0.0011221, Test Loss: 0.0013891
Epoch: 094, Train Loss: 0.0011157, Test Loss: 0.0013841
Epoch: 095, Train Loss: 0.0011091, Test Loss: 0.0013784
Epoch: 096, Train Loss: 0.0011025, Test Loss: 0.0013720
Epoch: 097, Train Loss: 0.0010956, Test Loss: 0.0013654
Epoch: 098, Train Loss: 0.0010886, Test Loss: 0.0013584
Epoch: 099, Train Loss: 0.0010812, Test Loss: 0.0013509
Epoch: 100, Train Loss: 0.0010736, Test Loss: 0.0013433
Epoch: 101, Train Loss: 0.0010657, Test Loss: 0.0013352
Epoch: 102, Train Loss: 0.0010574, Test Loss: 0.0013267
Epoch: 103, Train Loss: 0.0010488, Test Loss: 0.0013174
Epoch: 104, Train Loss: 0.0010399, Test Loss: 0.0013078
Epoch: 105, Train Loss: 0.0010307, Test Loss: 0.0012976
Epoch: 106, Train Loss: 0.0010211, Test Loss: 0.0012871
Epoch: 107, Train Loss: 0.0010113, Test Loss: 0.0012763
Epoch: 108, Train Loss: 0.0010014, Test Loss: 0.0012653
Epoch: 109, Train Loss: 0.0009913, Test Loss: 0.0012542
Epoch: 110, Train Loss: 0.0009810, Test Loss: 0.0012428
Epoch: 111, Train Loss: 0.0009705, Test Loss: 0.0012311
Epoch: 112, Train Loss: 0.0009600, Test Loss: 0.0012192
Epoch: 113, Train Loss: 0.0009494, Test Loss: 0.0012072
Epoch: 114, Train Loss: 0.0009388, Test Loss: 0.0011951
Epoch: 115, Train Loss: 0.0009283, Test Loss: 0.0011831
Epoch: 116, Train Loss: 0.0009178, Test Loss: 0.0011711
Epoch: 117, Train Loss: 0.0009074, Test Loss: 0.0011594
Epoch: 118, Train Loss: 0.0008969, Test Loss: 0.0011475
Epoch: 119, Train Loss: 0.0008865, Test Loss: 0.0011357
Epoch: 120, Train Loss: 0.0008762, Test Loss: 0.0011238
Epoch: 121, Train Loss: 0.0008661, Test Loss: 0.0011123
Epoch: 122, Train Loss: 0.0008562, Test Loss: 0.0011008
Epoch: 123, Train Loss: 0.0008463, Test Loss: 0.0010894
Epoch: 124, Train Loss: 0.0008366, Test Loss: 0.0010784
Epoch: 125, Train Loss: 0.0008269, Test Loss: 0.0010675
Epoch: 126, Train Loss: 0.0008174, Test Loss: 0.0010567
Epoch: 127, Train Loss: 0.0008082, Test Loss: 0.0010463
```

```
Epoch: 128, Train Loss: 0.0007992, Test Loss: 0.0010359
Epoch: 129, Train Loss: 0.0007904, Test Loss: 0.0010257
Epoch: 130, Train Loss: 0.0007817, Test Loss: 0.0010159
Epoch: 131, Train Loss: 0.0007732, Test Loss: 0.0010060
Epoch: 132, Train Loss: 0.0007649, Test Loss: 0.0009965
Epoch: 133, Train Loss: 0.0007566, Test Loss: 0.0009870
Epoch: 134, Train Loss: 0.0007485, Test Loss: 0.0009777
Epoch: 135, Train Loss: 0.0007406, Test Loss: 0.0009686
Epoch: 136, Train Loss: 0.0007327, Test Loss: 0.0009596
Epoch: 137, Train Loss: 0.0007251, Test Loss: 0.0009508
Epoch: 138, Train Loss: 0.0007175, Test Loss: 0.0009424
Epoch: 139, Train Loss: 0.0007102, Test Loss: 0.0009340
Epoch: 140, Train Loss: 0.0007029, Test Loss: 0.0009258
Epoch: 141, Train Loss: 0.0006958, Test Loss: 0.0009176
Epoch: 142, Train Loss: 0.0006888, Test Loss: 0.0009096
Epoch: 143, Train Loss: 0.0006819, Test Loss: 0.0009017
Epoch: 144, Train Loss: 0.0006750, Test Loss: 0.0008937
Epoch: 145, Train Loss: 0.0006683, Test Loss: 0.0008861
Epoch: 146, Train Loss: 0.0006618, Test Loss: 0.0008785
Epoch: 147, Train Loss: 0.0006554, Test Loss: 0.0008711
Epoch: 148, Train Loss: 0.0006490, Test Loss: 0.0008636
Epoch: 149, Train Loss: 0.0006427, Test Loss: 0.0008561
Epoch: 150, Train Loss: 0.0006365, Test Loss: 0.0008489
Epoch: 151, Train Loss: 0.0006304, Test Loss: 0.0008417
Epoch: 152, Train Loss: 0.0006244, Test Loss: 0.0008346
Epoch: 153, Train Loss: 0.0006184, Test Loss: 0.0008275
Epoch: 154, Train Loss: 0.0006125, Test Loss: 0.0008205
Epoch: 155, Train Loss: 0.0006066, Test Loss: 0.0008135
Epoch: 156, Train Loss: 0.0006008, Test Loss: 0.0008066
Epoch: 157, Train Loss: 0.0005950, Test Loss: 0.0007999
Epoch: 158, Train Loss: 0.0005895, Test Loss: 0.0007931
Epoch: 159, Train Loss: 0.0005840, Test Loss: 0.0007864
Epoch: 160, Train Loss: 0.0005787, Test Loss: 0.0007798
Epoch: 161, Train Loss: 0.0005734, Test Loss: 0.0007731
Epoch: 162, Train Loss: 0.0005683, Test Loss: 0.0007667
Epoch: 163, Train Loss: 0.0005632, Test Loss: 0.0007601
Epoch: 164, Train Loss: 0.0005581, Test Loss: 0.0007539
Epoch: 165, Train Loss: 0.0005532, Test Loss: 0.0007475
Epoch: 166, Train Loss: 0.0005483, Test Loss: 0.0007413
Epoch: 167, Train Loss: 0.0005435, Test Loss: 0.0007351
Epoch: 168, Train Loss: 0.0005387, Test Loss: 0.0007291
Epoch: 169, Train Loss: 0.0005341, Test Loss: 0.0007231
Epoch: 170, Train Loss: 0.0005295, Test Loss: 0.0007173
Epoch: 171, Train Loss: 0.0005250, Test Loss: 0.0007116
Epoch: 172, Train Loss: 0.0005206, Test Loss: 0.0007060
Epoch: 173, Train Loss: 0.0005162, Test Loss: 0.0007004
Epoch: 174, Train Loss: 0.0005118, Test Loss: 0.0006947
Epoch: 175, Train Loss: 0.0005076, Test Loss: 0.0006891
Epoch: 176, Train Loss: 0.0005032, Test Loss: 0.0006834
```

```
Epoch: 177, Train Loss: 0.0004991, Test Loss: 0.0006778
Epoch: 178, Train Loss: 0.0004949, Test Loss: 0.0006722
Epoch: 179, Train Loss: 0.0004908, Test Loss: 0.0006669
Epoch: 180, Train Loss: 0.0004868, Test Loss: 0.0006613
Epoch: 181, Train Loss: 0.0004828, Test Loss: 0.0006560
Epoch: 182, Train Loss: 0.0004787, Test Loss: 0.0006505
Epoch: 183, Train Loss: 0.0004749, Test Loss: 0.0006451
Epoch: 184, Train Loss: 0.0004709, Test Loss: 0.0006397
Epoch: 185, Train Loss: 0.0004671, Test Loss: 0.0006344
Epoch: 186, Train Loss: 0.0004633, Test Loss: 0.0006292
Epoch: 187, Train Loss: 0.0004596, Test Loss: 0.0006240
Epoch: 188, Train Loss: 0.0004559, Test Loss: 0.0006188
Epoch: 189, Train Loss: 0.0004523, Test Loss: 0.0006137
Epoch: 190, Train Loss: 0.0004487, Test Loss: 0.0006086
Epoch: 191, Train Loss: 0.0004452, Test Loss: 0.0006036
Epoch: 192, Train Loss: 0.0004417, Test Loss: 0.0005986
Epoch: 193, Train Loss: 0.0004382, Test Loss: 0.0005937
Epoch: 194, Train Loss: 0.0004350, Test Loss: 0.0005890
Epoch: 195, Train Loss: 0.0004317, Test Loss: 0.0005842
Epoch: 196, Train Loss: 0.0004285, Test Loss: 0.0005794
Epoch: 197, Train Loss: 0.0004255, Test Loss: 0.0005749
Epoch: 198, Train Loss: 0.0004224, Test Loss: 0.0005703
Epoch: 199, Train Loss: 0.0004194, Test Loss: 0.0005657
Epoch: 200, Train Loss: 0.0004166, Test Loss: 0.0005612
Epoch: 201, Train Loss: 0.0004139, Test Loss: 0.0005568
Epoch: 202, Train Loss: 0.0004112, Test Loss: 0.0005526
Epoch: 203, Train Loss: 0.0004086, Test Loss: 0.0005483
Epoch: 204, Train Loss: 0.0004060, Test Loss: 0.0005443
Epoch: 205, Train Loss: 0.0004036, Test Loss: 0.0005402
Epoch: 206, Train Loss: 0.0004012, Test Loss: 0.0005362
Epoch: 207, Train Loss: 0.0003990, Test Loss: 0.0005325
Epoch: 208, Train Loss: 0.0003969, Test Loss: 0.0005288
Epoch: 209, Train Loss: 0.0003949, Test Loss: 0.0005252
Epoch: 210, Train Loss: 0.0003930, Test Loss: 0.0005217
Epoch: 211, Train Loss: 0.0003913, Test Loss: 0.0005183
Epoch: 212, Train Loss: 0.0003897, Test Loss: 0.0005150
Epoch: 213, Train Loss: 0.0003882, Test Loss: 0.0005118
Epoch: 214, Train Loss: 0.0003869, Test Loss: 0.0005087
Epoch: 215, Train Loss: 0.0003856, Test Loss: 0.0005057
Epoch: 216, Train Loss: 0.0003844, Test Loss: 0.0005027
Epoch: 217, Train Loss: 0.0003834, Test Loss: 0.0004998
Epoch: 218, Train Loss: 0.0003823, Test Loss: 0.0004968
Epoch: 219, Train Loss: 0.0003812, Test Loss: 0.0004941
Epoch: 220, Train Loss: 0.0003803, Test Loss: 0.0004914
Epoch: 221, Train Loss: 0.0003792, Test Loss: 0.0004882
Epoch: 222, Train Loss: 0.0003780, Test Loss: 0.0004853
Epoch: 223, Train Loss: 0.0003767, Test Loss: 0.0004818
Epoch: 224, Train Loss: 0.0003751, Test Loss: 0.0004784
Epoch: 225, Train Loss: 0.0003733, Test Loss: 0.0004747
```

```
Epoch: 226, Train Loss: 0.0003711, Test Loss: 0.0004705
Epoch: 227, Train Loss: 0.0003686, Test Loss: 0.0004661
Epoch: 228, Train Loss: 0.0003656, Test Loss: 0.0004610
Epoch: 229, Train Loss: 0.0003620, Test Loss: 0.0004556
Epoch: 230, Train Loss: 0.0003577, Test Loss: 0.0004493
Epoch: 231, Train Loss: 0.0003527, Test Loss: 0.0004426
Epoch: 232, Train Loss: 0.0003470, Test Loss: 0.0004349
Epoch: 233, Train Loss: 0.0003404, Test Loss: 0.0004262
Epoch: 234, Train Loss: 0.0003331, Test Loss: 0.0004172
Epoch: 235, Train Loss: 0.0003252, Test Loss: 0.0004075
Epoch: 236, Train Loss: 0.0003167, Test Loss: 0.0003975
Epoch: 237, Train Loss: 0.0003078, Test Loss: 0.0003871
Epoch: 238, Train Loss: 0.0002989, Test Loss: 0.0003767
Epoch: 239, Train Loss: 0.0002899, Test Loss: 0.0003668
Epoch: 240, Train Loss: 0.0002816, Test Loss: 0.0003575
Epoch: 241, Train Loss: 0.0002740, Test Loss: 0.0003492
Epoch: 242, Train Loss: 0.0002677, Test Loss: 0.0003422
Epoch: 243, Train Loss: 0.0002627, Test Loss: 0.0003370
Epoch: 244, Train Loss: 0.0002590, Test Loss: 0.0003330
Epoch: 245, Train Loss: 0.0002567, Test Loss: 0.0003303
Epoch: 246, Train Loss: 0.0002557, Test Loss: 0.0003289
Epoch: 247, Train Loss: 0.0002557, Test Loss: 0.0003285
Epoch: 248, Train Loss: 0.0002562, Test Loss: 0.0003285
Epoch: 249, Train Loss: 0.0002573, Test Loss: 0.0003291
Epoch: 250, Train Loss: 0.0002586, Test Loss: 0.0003296
Epoch: 251, Train Loss: 0.0002598, Test Loss: 0.0003300
Epoch: 252, Train Loss: 0.0002607, Test Loss: 0.0003300
Epoch: 253, Train Loss: 0.0002615, Test Loss: 0.0003297
Epoch: 254, Train Loss: 0.0002619, Test Loss: 0.0003290
Epoch: 255, Train Loss: 0.0002620, Test Loss: 0.0003277
Epoch: 256, Train Loss: 0.0002618, Test Loss: 0.0003261
Epoch: 257, Train Loss: 0.0002613, Test Loss: 0.0003242
Epoch: 258, Train Loss: 0.0002606, Test Loss: 0.0003220
Epoch: 259, Train Loss: 0.0002596, Test Loss: 0.0003195
Epoch: 260, Train Loss: 0.0002586, Test Loss: 0.0003168
Epoch: 261, Train Loss: 0.0002573, Test Loss: 0.0003140
Epoch: 262, Train Loss: 0.0002560, Test Loss: 0.0003112
Epoch: 263, Train Loss: 0.0002549, Test Loss: 0.0003084
Epoch: 264, Train Loss: 0.0002536, Test Loss: 0.0003055
Epoch: 265, Train Loss: 0.0002522, Test Loss: 0.0003024
Epoch: 266, Train Loss: 0.0002508, Test Loss: 0.0002994
Epoch: 267, Train Loss: 0.0002493, Test Loss: 0.0002963
Epoch: 268, Train Loss: 0.0002478, Test Loss: 0.0002933
Epoch: 269, Train Loss: 0.0002463, Test Loss: 0.0002903
Epoch: 270, Train Loss: 0.0002449, Test Loss: 0.0002875
Epoch: 271, Train Loss: 0.0002436, Test Loss: 0.0002847
Epoch: 272, Train Loss: 0.0002422, Test Loss: 0.0002818
Epoch: 273, Train Loss: 0.0002410, Test Loss: 0.0002792
Epoch: 274, Train Loss: 0.0002396, Test Loss: 0.0002764
```

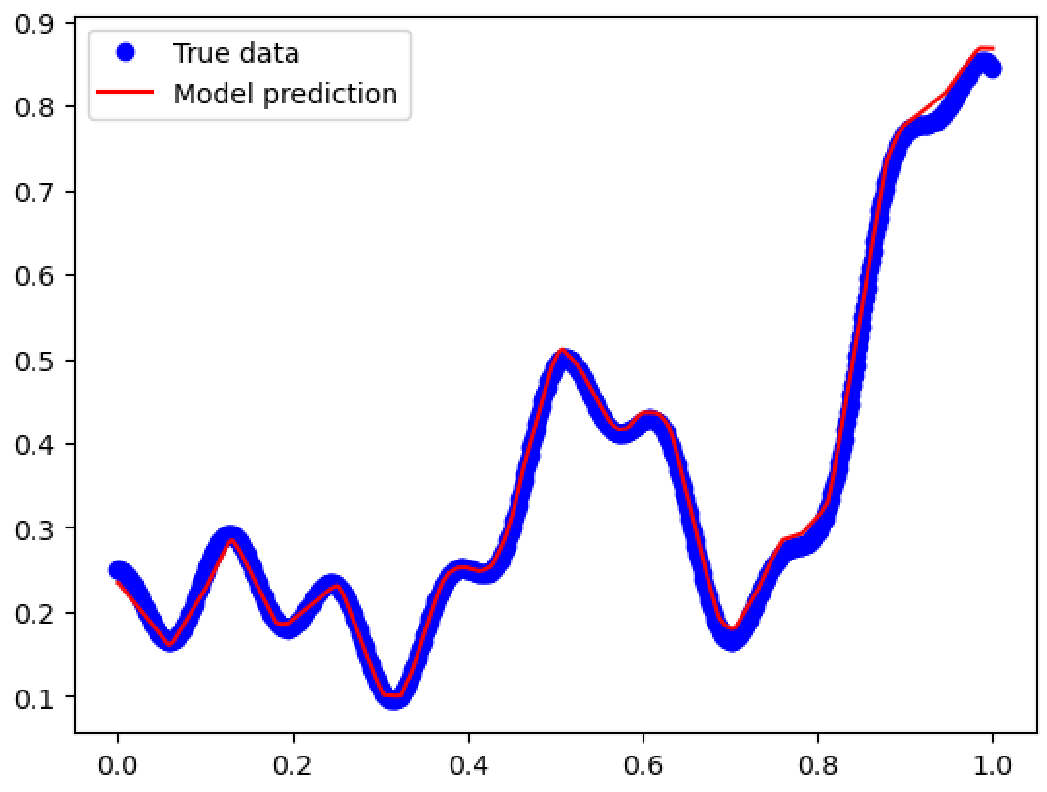
```
Epoch: 275, Train Loss: 0.0002384, Test Loss: 0.0002738
Epoch: 276, Train Loss: 0.0002370, Test Loss: 0.0002710
Epoch: 277, Train Loss: 0.0002357, Test Loss: 0.0002685
Epoch: 278, Train Loss: 0.0002344, Test Loss: 0.0002660
Epoch: 279, Train Loss: 0.0002331, Test Loss: 0.0002635
Epoch: 280, Train Loss: 0.0002318, Test Loss: 0.0002611
Epoch: 281, Train Loss: 0.0002303, Test Loss: 0.0002586
Epoch: 282, Train Loss: 0.0002288, Test Loss: 0.0002560
Epoch: 283, Train Loss: 0.0002272, Test Loss: 0.0002535
Epoch: 284, Train Loss: 0.0002255, Test Loss: 0.0002509
Epoch: 285, Train Loss: 0.0002235, Test Loss: 0.0002481
Epoch: 286, Train Loss: 0.0002217, Test Loss: 0.0002454
Epoch: 287, Train Loss: 0.0002195, Test Loss: 0.0002425
Epoch: 288, Train Loss: 0.0002173, Test Loss: 0.0002397
Epoch: 289, Train Loss: 0.0002148, Test Loss: 0.0002367
Epoch: 290, Train Loss: 0.0002123, Test Loss: 0.0002336
Epoch: 291, Train Loss: 0.0002097, Test Loss: 0.0002306
Epoch: 292, Train Loss: 0.0002069, Test Loss: 0.0002276
Epoch: 293, Train Loss: 0.0002040, Test Loss: 0.0002244
Epoch: 294, Train Loss: 0.0002011, Test Loss: 0.0002212
Epoch: 295, Train Loss: 0.0001980, Test Loss: 0.0002180
Epoch: 296, Train Loss: 0.0001949, Test Loss: 0.0002148
Epoch: 297, Train Loss: 0.0001918, Test Loss: 0.0002116
Epoch: 298, Train Loss: 0.0001886, Test Loss: 0.0002084
Epoch: 299, Train Loss: 0.0001854, Test Loss: 0.0002052
Epoch: 300, Train Loss: 0.0001822, Test Loss: 0.0002020
Epoch: 301, Train Loss: 0.0001789, Test Loss: 0.0001988
Epoch: 302, Train Loss: 0.0001758, Test Loss: 0.0001957
Epoch: 303, Train Loss: 0.0001726, Test Loss: 0.0001927
Epoch: 304, Train Loss: 0.0001695, Test Loss: 0.0001896
Epoch: 305, Train Loss: 0.0001663, Test Loss: 0.0001865
Epoch: 306, Train Loss: 0.0001633, Test Loss: 0.0001836
Epoch: 307, Train Loss: 0.0001604, Test Loss: 0.0001808
Epoch: 308, Train Loss: 0.0001574, Test Loss: 0.0001779
Epoch: 309, Train Loss: 0.0001545, Test Loss: 0.0001751
Epoch: 310, Train Loss: 0.0001517, Test Loss: 0.0001725
Epoch: 311, Train Loss: 0.0001490, Test Loss: 0.0001698
Epoch: 312, Train Loss: 0.0001463, Test Loss: 0.0001672
Epoch: 313, Train Loss: 0.0001437, Test Loss: 0.0001648
Epoch: 314, Train Loss: 0.0001412, Test Loss: 0.0001623
Epoch: 315, Train Loss: 0.0001388, Test Loss: 0.0001600
Epoch: 316, Train Loss: 0.0001364, Test Loss: 0.0001577
Epoch: 317, Train Loss: 0.0001341, Test Loss: 0.0001555
Epoch: 318, Train Loss: 0.0001319, Test Loss: 0.0001535
Epoch: 319, Train Loss: 0.0001297, Test Loss: 0.0001514
Epoch: 320, Train Loss: 0.0001277, Test Loss: 0.0001495
Epoch: 321, Train Loss: 0.0001257, Test Loss: 0.0001476
Epoch: 322, Train Loss: 0.0001239, Test Loss: 0.0001459
Epoch: 323, Train Loss: 0.0001220, Test Loss: 0.0001442
```

```
Epoch: 324, Train Loss: 0.0001203, Test Loss: 0.0001426
Epoch: 325, Train Loss: 0.0001188, Test Loss: 0.0001411
Epoch: 326, Train Loss: 0.0001172, Test Loss: 0.0001396
Epoch: 327, Train Loss: 0.0001158, Test Loss: 0.0001382
Epoch: 328, Train Loss: 0.0001144, Test Loss: 0.0001369
Epoch: 329, Train Loss: 0.0001131, Test Loss: 0.0001356
Epoch: 330, Train Loss: 0.0001119, Test Loss: 0.0001343
Epoch: 331, Train Loss: 0.0001106, Test Loss: 0.0001330
Epoch: 332, Train Loss: 0.0001094, Test Loss: 0.0001316
Epoch: 333, Train Loss: 0.0001081, Test Loss: 0.0001303
Epoch: 334, Train Loss: 0.0001069, Test Loss: 0.0001289
Epoch: 335, Train Loss: 0.0001056, Test Loss: 0.0001274
Epoch: 336, Train Loss: 0.0001044, Test Loss: 0.0001259
Epoch: 337, Train Loss: 0.0001031, Test Loss: 0.0001245
Epoch: 338, Train Loss: 0.0001020, Test Loss: 0.0001230
Epoch: 339, Train Loss: 0.0001009, Test Loss: 0.0001217
Epoch: 340, Train Loss: 0.0000999, Test Loss: 0.0001204
Epoch: 341, Train Loss: 0.0000990, Test Loss: 0.0001193
Epoch: 342, Train Loss: 0.0000981, Test Loss: 0.0001182
Epoch: 343, Train Loss: 0.0000973, Test Loss: 0.0001172
Epoch: 344, Train Loss: 0.0000966, Test Loss: 0.0001162
Epoch: 345, Train Loss: 0.0000958, Test Loss: 0.0001153
Epoch: 346, Train Loss: 0.0000951, Test Loss: 0.0001143
Epoch: 347, Train Loss: 0.0000943, Test Loss: 0.0001134
Epoch: 348, Train Loss: 0.0000935, Test Loss: 0.0001124
Epoch: 349, Train Loss: 0.0000927, Test Loss: 0.0001115
Epoch: 350, Train Loss: 0.0000919, Test Loss: 0.0001105
Epoch: 351, Train Loss: 0.0000911, Test Loss: 0.0001096
Epoch: 352, Train Loss: 0.0000903, Test Loss: 0.0001086
Epoch: 353, Train Loss: 0.0000896, Test Loss: 0.0001077
Epoch: 354, Train Loss: 0.0000887, Test Loss: 0.0001067
Epoch: 355, Train Loss: 0.0000880, Test Loss: 0.0001058
Epoch: 356, Train Loss: 0.0000872, Test Loss: 0.0001049
Epoch: 357, Train Loss: 0.0000864, Test Loss: 0.0001040
Epoch: 358, Train Loss: 0.0000857, Test Loss: 0.0001031
Epoch: 359, Train Loss: 0.0000850, Test Loss: 0.0001022
Epoch: 360, Train Loss: 0.0000843, Test Loss: 0.0001014
Epoch: 361, Train Loss: 0.0000837, Test Loss: 0.0001007
Epoch: 362, Train Loss: 0.0000831, Test Loss: 0.0000999
Epoch: 363, Train Loss: 0.0000826, Test Loss: 0.0000992
Epoch: 364, Train Loss: 0.0000819, Test Loss: 0.0000984
Epoch: 365, Train Loss: 0.0000815, Test Loss: 0.0000977
Epoch: 366, Train Loss: 0.0000809, Test Loss: 0.0000970
Epoch: 367, Train Loss: 0.0000805, Test Loss: 0.0000964
Epoch: 368, Train Loss: 0.0000800, Test Loss: 0.0000958
Epoch: 369, Train Loss: 0.0000796, Test Loss: 0.0000952
Epoch: 370, Train Loss: 0.0000792, Test Loss: 0.0000946
Epoch: 371, Train Loss: 0.0000789, Test Loss: 0.0000942
Epoch: 372, Train Loss: 0.0000785, Test Loss: 0.0000938
```

```
Epoch: 373, Train Loss: 0.0000782, Test Loss: 0.0000934
Epoch: 374, Train Loss: 0.0000780, Test Loss: 0.0000931
Epoch: 375, Train Loss: 0.0000777, Test Loss: 0.0000928
Epoch: 376, Train Loss: 0.0000777, Test Loss: 0.0000927
Epoch: 377, Train Loss: 0.0000775, Test Loss: 0.0000925
Epoch: 378, Train Loss: 0.0000773, Test Loss: 0.0000923
Epoch: 379, Train Loss: 0.0000770, Test Loss: 0.0000921
Epoch: 380, Train Loss: 0.0000768, Test Loss: 0.0000919
Epoch: 381, Train Loss: 0.0000764, Test Loss: 0.0000916
Epoch: 382, Train Loss: 0.0000760, Test Loss: 0.0000912
Epoch: 383, Train Loss: 0.0000753, Test Loss: 0.0000905
Epoch: 384, Train Loss: 0.0000744, Test Loss: 0.0000898
Epoch: 385, Train Loss: 0.0000731, Test Loss: 0.0000886
Epoch: 386, Train Loss: 0.0000715, Test Loss: 0.0000872
Epoch: 387, Train Loss: 0.0000696, Test Loss: 0.0000855
Epoch: 388, Train Loss: 0.0000674, Test Loss: 0.0000836
Epoch: 389, Train Loss: 0.0000655, Test Loss: 0.0000820
Epoch: 390, Train Loss: 0.0000642, Test Loss: 0.0000810
Epoch: 391, Train Loss: 0.0000643, Test Loss: 0.0000816
Epoch: 392, Train Loss: 0.0000660, Test Loss: 0.0000837
Epoch: 393, Train Loss: 0.0000703, Test Loss: 0.0000886
Epoch: 394, Train Loss: 0.0000762, Test Loss: 0.0000951
Epoch: 395, Train Loss: 0.0000829, Test Loss: 0.0001026
Epoch: 396, Train Loss: 0.0000901, Test Loss: 0.0001105
Epoch: 397, Train Loss: 0.0000943, Test Loss: 0.0001153
Epoch: 398, Train Loss: 0.0000976, Test Loss: 0.0001193
Epoch: 399, Train Loss: 0.0000974, Test Loss: 0.0001194
Epoch: 400, Train Loss: 0.0000956, Test Loss: 0.0001178
Epoch: 401, Train Loss: 0.0000944, Test Loss: 0.0001168
Epoch: 402, Train Loss: 0.0000909, Test Loss: 0.0001131
Epoch: 403, Train Loss: 0.0000882, Test Loss: 0.0001104
Epoch: 404, Train Loss: 0.0000860, Test Loss: 0.0001082
Epoch: 405, Train Loss: 0.0000825, Test Loss: 0.0001044
Epoch: 406, Train Loss: 0.0000800, Test Loss: 0.0001016
Epoch: 407, Train Loss: 0.0000777, Test Loss: 0.0000994
Epoch: 408, Train Loss: 0.0000749, Test Loss: 0.0000962
Epoch: 409, Train Loss: 0.0000728, Test Loss: 0.0000938
Epoch: 410, Train Loss: 0.0000712, Test Loss: 0.0000921
Epoch: 411, Train Loss: 0.0000686, Test Loss: 0.0000892
Epoch: 412, Train Loss: 0.0000674, Test Loss: 0.0000877
Epoch: 413, Train Loss: 0.0000654, Test Loss: 0.0000856
Epoch: 414, Train Loss: 0.0000638, Test Loss: 0.0000836
Epoch: 415, Train Loss: 0.0000626, Test Loss: 0.0000823
Epoch: 416, Train Loss: 0.0000609, Test Loss: 0.0000803
Epoch: 417, Train Loss: 0.0000600, Test Loss: 0.0000791
Epoch: 418, Train Loss: 0.0000587, Test Loss: 0.0000776
Epoch: 419, Train Loss: 0.0000578, Test Loss: 0.0000764
Epoch: 420, Train Loss: 0.0000567, Test Loss: 0.0000751
Epoch: 421, Train Loss: 0.0000561, Test Loss: 0.0000743
```

```
Epoch: 422, Train Loss: 0.0000552, Test Loss: 0.0000731
Epoch: 423, Train Loss: 0.0000545, Test Loss: 0.0000722
Epoch: 424, Train Loss: 0.0000537, Test Loss: 0.0000711
Epoch: 425, Train Loss: 0.0000533, Test Loss: 0.0000704
Epoch: 426, Train Loss: 0.0000526, Test Loss: 0.0000694
Epoch: 427, Train Loss: 0.0000521, Test Loss: 0.0000687
Epoch: 428, Train Loss: 0.0000517, Test Loss: 0.0000681
Epoch: 429, Train Loss: 0.0000514, Test Loss: 0.0000675
Epoch: 430, Train Loss: 0.0000510, Test Loss: 0.0000668
Epoch: 431, Train Loss: 0.0000505, Test Loss: 0.0000661
Epoch: 432, Train Loss: 0.0000501, Test Loss: 0.0000653
Epoch: 433, Train Loss: 0.0000495, Test Loss: 0.0000644
Epoch: 434, Train Loss: 0.0000490, Test Loss: 0.0000635
Epoch: 435, Train Loss: 0.0000485, Test Loss: 0.0000627
Epoch: 436, Train Loss: 0.0000485, Test Loss: 0.0000621
Epoch: 437, Train Loss: 0.0000491, Test Loss: 0.0000623
Epoch: 438, Train Loss: 0.0000506, Test Loss: 0.0000633
Epoch: 439, Train Loss: 0.0000528, Test Loss: 0.0000650
Epoch: 440, Train Loss: 0.0000550, Test Loss: 0.0000669
Epoch: 441, Train Loss: 0.0000565, Test Loss: 0.0000681
Epoch: 442, Train Loss: 0.0000570, Test Loss: 0.0000684
Epoch: 443, Train Loss: 0.0000569, Test Loss: 0.0000681
Epoch: 444, Train Loss: 0.0000568, Test Loss: 0.0000679
Epoch: 445, Train Loss: 0.0000572, Test Loss: 0.0000681
Epoch: 446, Train Loss: 0.0000583, Test Loss: 0.0000690
Epoch: 447, Train Loss: 0.0000598, Test Loss: 0.0000703
Epoch: 448, Train Loss: 0.0000614, Test Loss: 0.0000716
Epoch: 449, Train Loss: 0.0000629, Test Loss: 0.0000729
Epoch: 450, Train Loss: 0.0000643, Test Loss: 0.0000742
Epoch: 451, Train Loss: 0.0000659, Test Loss: 0.0000756
Epoch: 452, Train Loss: 0.0000675, Test Loss: 0.0000770
Epoch: 453, Train Loss: 0.0000689, Test Loss: 0.0000784
Epoch: 454, Train Loss: 0.0000701, Test Loss: 0.0000796
Epoch: 455, Train Loss: 0.0000712, Test Loss: 0.0000806
Epoch: 456, Train Loss: 0.0000721, Test Loss: 0.0000814
Epoch: 457, Train Loss: 0.0000727, Test Loss: 0.0000819
Epoch: 458, Train Loss: 0.0000730, Test Loss: 0.0000822
Epoch: 459, Train Loss: 0.0000736, Test Loss: 0.0000826
Epoch: 460, Train Loss: 0.0000754, Test Loss: 0.0000842
Epoch: 461, Train Loss: 0.0000812, Test Loss: 0.0000898
Epoch: 462, Train Loss: 0.0000961, Test Loss: 0.0001042
Epoch: 463, Train Loss: 0.0001282, Test Loss: 0.0001354
Epoch: 464, Train Loss: 0.0001830, Test Loss: 0.0001888
Epoch: 465, Train Loss: 0.0002584, Test Loss: 0.0002621
Epoch: 466, Train Loss: 0.0003454, Test Loss: 0.0003459
Epoch: 467, Train Loss: 0.0004270, Test Loss: 0.0004241
Epoch: 468, Train Loss: 0.0004963, Test Loss: 0.0004898
Epoch: 469, Train Loss: 0.0005589, Test Loss: 0.0005494
Epoch: 470, Train Loss: 0.0006017, Test Loss: 0.0005906
```

```
Epoch: 471, Train Loss: 0.0006152, Test Loss: 0.0006024
Epoch: 472, Train Loss: 0.0006238, Test Loss: 0.0006108
Epoch: 473, Train Loss: 0.0006230, Test Loss: 0.0006100
Epoch: 474, Train Loss: 0.0005955, Test Loss: 0.0005818
Epoch: 475, Train Loss: 0.0005701, Test Loss: 0.0005573
Epoch: 476, Train Loss: 0.0005472, Test Loss: 0.0005348
Epoch: 477, Train Loss: 0.0005196, Test Loss: 0.0005075
Epoch: 478, Train Loss: 0.0004899, Test Loss: 0.0004782
Epoch: 479, Train Loss: 0.0004710, Test Loss: 0.0004595
Epoch: 480, Train Loss: 0.0004528, Test Loss: 0.0004407
Epoch: 481, Train Loss: 0.0004419, Test Loss: 0.0004295
Epoch: 482, Train Loss: 0.0004317, Test Loss: 0.0004185
Epoch: 483, Train Loss: 0.0004278, Test Loss: 0.0004136
Epoch: 484, Train Loss: 0.0004216, Test Loss: 0.0004060
Epoch: 485, Train Loss: 0.0004094, Test Loss: 0.0003926
Epoch: 486, Train Loss: 0.0003774, Test Loss: 0.0003604
Epoch: 487, Train Loss: 0.0003084, Test Loss: 0.0002938
Epoch: 488, Train Loss: 0.0002278, Test Loss: 0.0002182
Epoch: 489, Train Loss: 0.0001745, Test Loss: 0.0001699
Epoch: 490, Train Loss: 0.0001789, Test Loss: 0.0001777
Epoch: 491, Train Loss: 0.0002409, Test Loss: 0.0002447
Epoch: 492, Train Loss: 0.0004290, Test Loss: 0.0004521
Epoch: 493, Train Loss: 0.0020765, Test Loss: 0.0021151
Epoch: 494, Train Loss: 0.0006884, Test Loss: 0.0006391
Epoch: 495, Train Loss: 0.0001314, Test Loss: 0.0001394
Epoch: 496, Train Loss: 0.0001207, Test Loss: 0.0001310
Epoch: 497, Train Loss: 0.0001498, Test Loss: 0.0001607
Epoch: 498, Train Loss: 0.0002108, Test Loss: 0.0002221
Epoch: 499, Train Loss: 0.0002303, Test Loss: 0.0002353
Epoch: 500, Train Loss: 0.0001442, Test Loss: 0.0001489
```



Determine appropriate number of epoch = 400

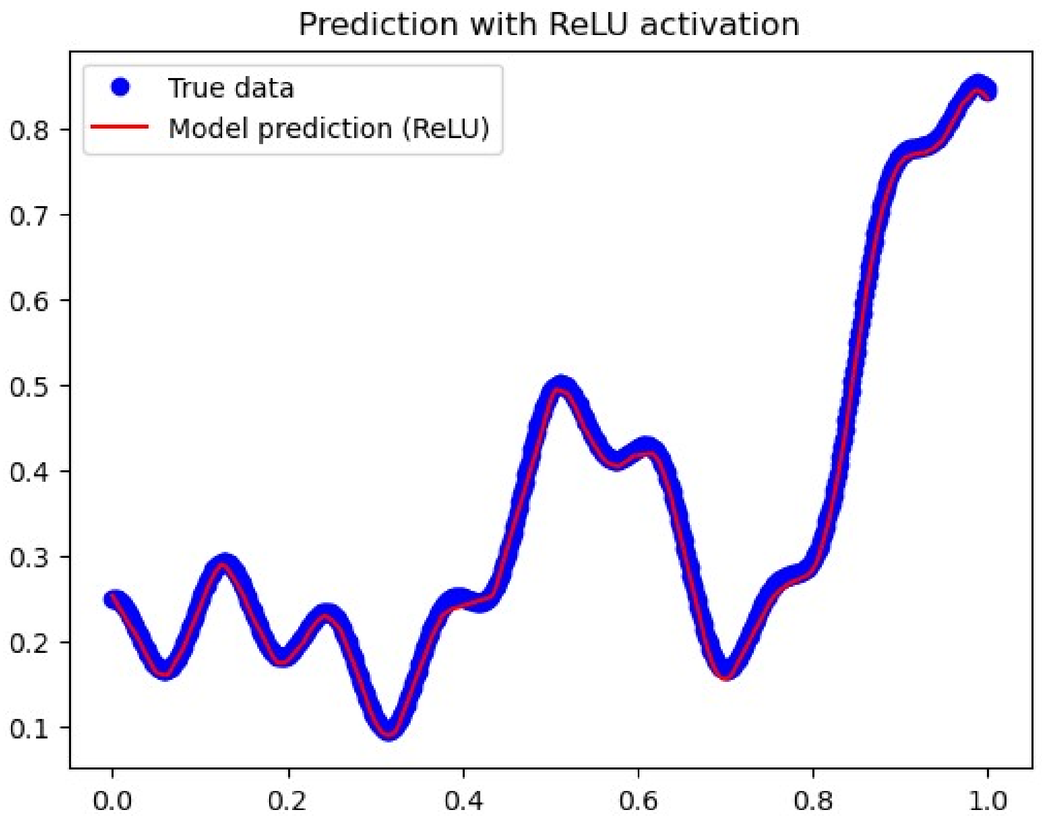
```
# Import dependencies
import numpy as np
import torch
import torchvision
from torch.utils.data.dataset import Dataset
from torchvision import datasets, transforms
from torch import nn, optim
import matplotlib.pyplot as plt
```

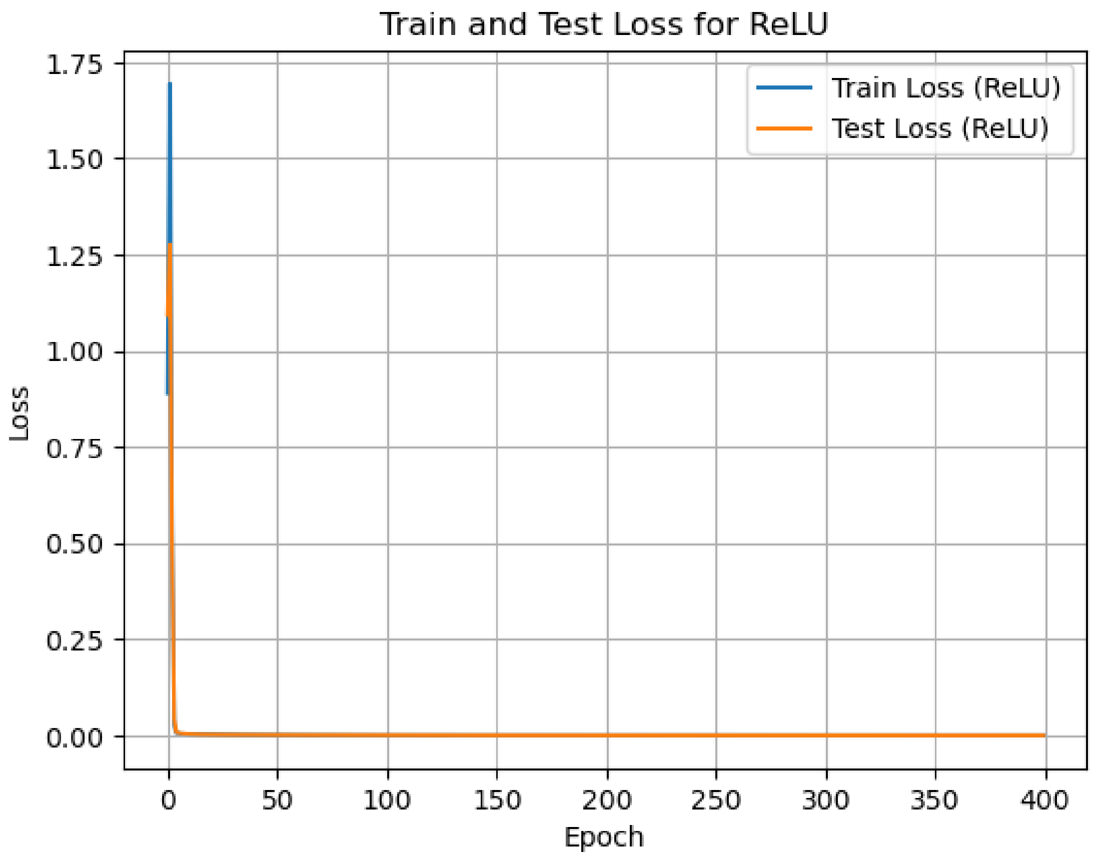
Problem 4.3 Compare different activation functions (sigmoid, Tanh, and ReLU)

```
# ################## Part 1: Load data and create batch
##################
N_total = 600
N_train = 500
x = torch.unsqueeze(torch.linspace(0, 1, N_total), dim=1)
r = torch.randperm(N_total)
x = x[r, :]
y = 0.2 + 0.4 * torch.pow(x, 2) + 0.3 * x * torch.sin(15 * x) + 0.05 *
torch.cos(50 * x)

class CustomDataset(Dataset):
    def __init__(self, x, y):
        self.y = y
        self.x = x

    def __len__(self):
        return len(self.y)

    def __getitem__(self, idx):
        y1 = self.y[idx]
        x1 = self.x[idx]
        return (x1, y1)

# Change batch_size here to test different values
batch_size = 32  # Experiment with different batch sizes: 32, 64, 128
trainset = CustomDataset(x[0:N_train, :], y[0:N_train, :])
testset = CustomDataset(x[N_train:N_total, :], y[N_train:N_total, :])
train_loader = torch.utils.data.DataLoader(trainset,
batch_size=batch_size)
test_loader = torch.utils.data.DataLoader(testset,
batch_size=batch_size)

# ################## Part 2: Define Model and initialize
##################
def init_weights(m):
    if isinstance(m, nn.Linear):
```

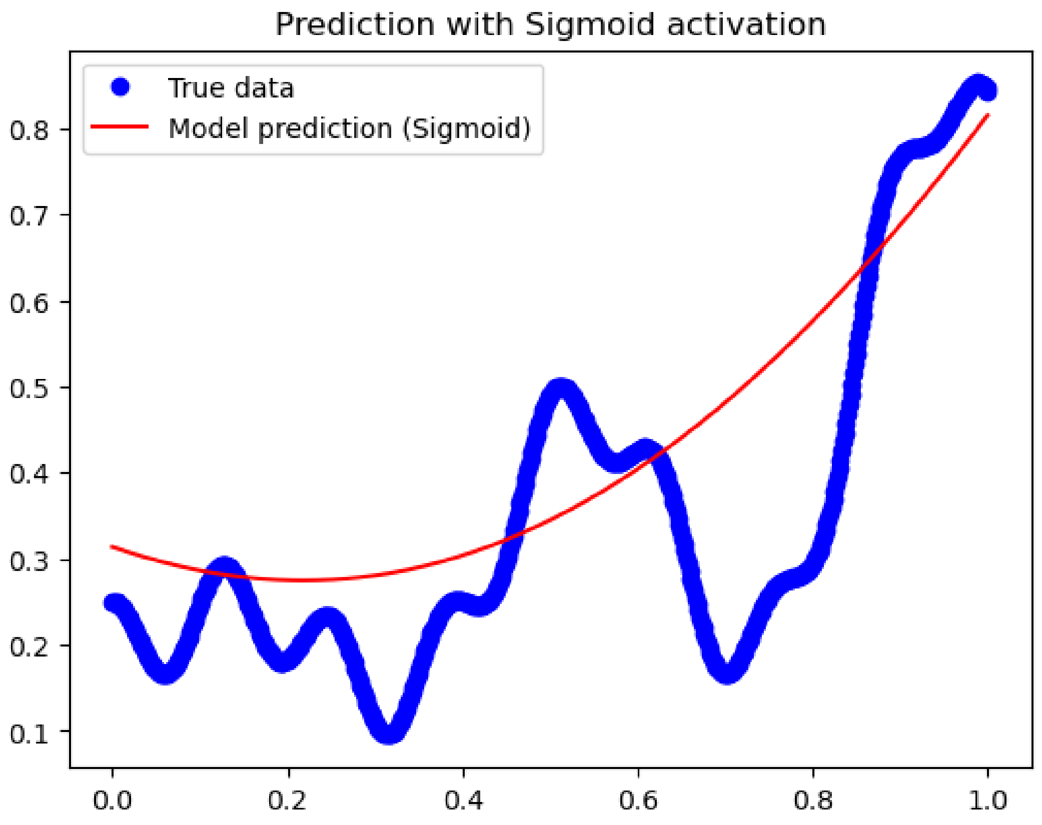
```python
        m.weight.data.uniform_(-1, 1)
        m.bias.data.uniform_(-1, 1)

# ################## Part 3: Define Loss and optimizer
##################
criterion = torch.nn.MSELoss()

# ################## Part 4: Train and Test Functions
##################
def train_NN(optimizer, model):
    model.train()
    for images, labels in train_loader:
        out = model(images)
        loss = criterion(out, labels)
        loss.backward()
        optimizer.step()
        optimizer.zero_grad()
    return loss.item()

def test_NN(model, loader):
    model.eval()
    loss = 0
    with torch.no_grad():
        for images, labels in loader:
            out = model(images)
            loss += criterion(out, labels).item()
    loss /= len(loader)
    return loss

# ################## Compare Activation Functions ##################
activation_functions = {
    "ReLU": nn.ReLU(),
    "Sigmoid": nn.Sigmoid(),
    "Tanh": nn.Tanh()
}

N_epoch = 400  # You can adjust the number of epochs

for activation_name, activation in activation_functions.items():
    # Define the model with the current activation function
    model = nn.Sequential(
        nn.Linear(1, 1024, bias=True),
        activation,  # Use the current activation function
        nn.Linear(1024, 1, bias=True)
    )

    # Initialize weights
    model.apply(init_weights)

    # Initialize optimizer
    optimizer = torch.optim.Adam(model.parameters(), lr=0.001)

    # Initialize loss trackers
    train_loss = np.zeros((N_epoch, 1))
    test_loss = np.zeros((N_epoch, 1))

    print(f"Testing {activation_name} activation function")

    # Train the model
    for epoch in range(N_epoch):
        train_loss[epoch, 0] = train_NN(optimizer, model)
        test_loss[epoch, 0] = test_NN(model, test_loader)
        if epoch % 100 == 0:
            print(f'Epoch: {epoch:03d}, Train Loss: {train_loss[epoch,
0]:.7f}, Test Loss: {test_loss[epoch, 0]:.7f}')

    # Final prediction and plot
    x_test = torch.unsqueeze(torch.linspace(0, 1, 1999), dim=1)
    y_test = model(x_test)

    # Plot the results
    plt.figure()
    plt.plot(x[0:N_total], y[0:N_total], 'bo', label='True data')
    plt.plot(x_test, y_test.detach().numpy(), 'r', label=f'Model
prediction ({activation_name})')
    plt.legend()
    plt.title(f"Prediction with {activation_name} activation")
    plt.show()

    # Plot train and test loss
    plt.figure()
    plt.plot(train_loss, label=f'Train Loss ({activation_name})')
    plt.plot(test_loss, label=f'Test Loss ({activation_name})')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()
    plt.title(f'Train and Test Loss for {activation_name}')
    plt.grid(True)
    plt.show()
```

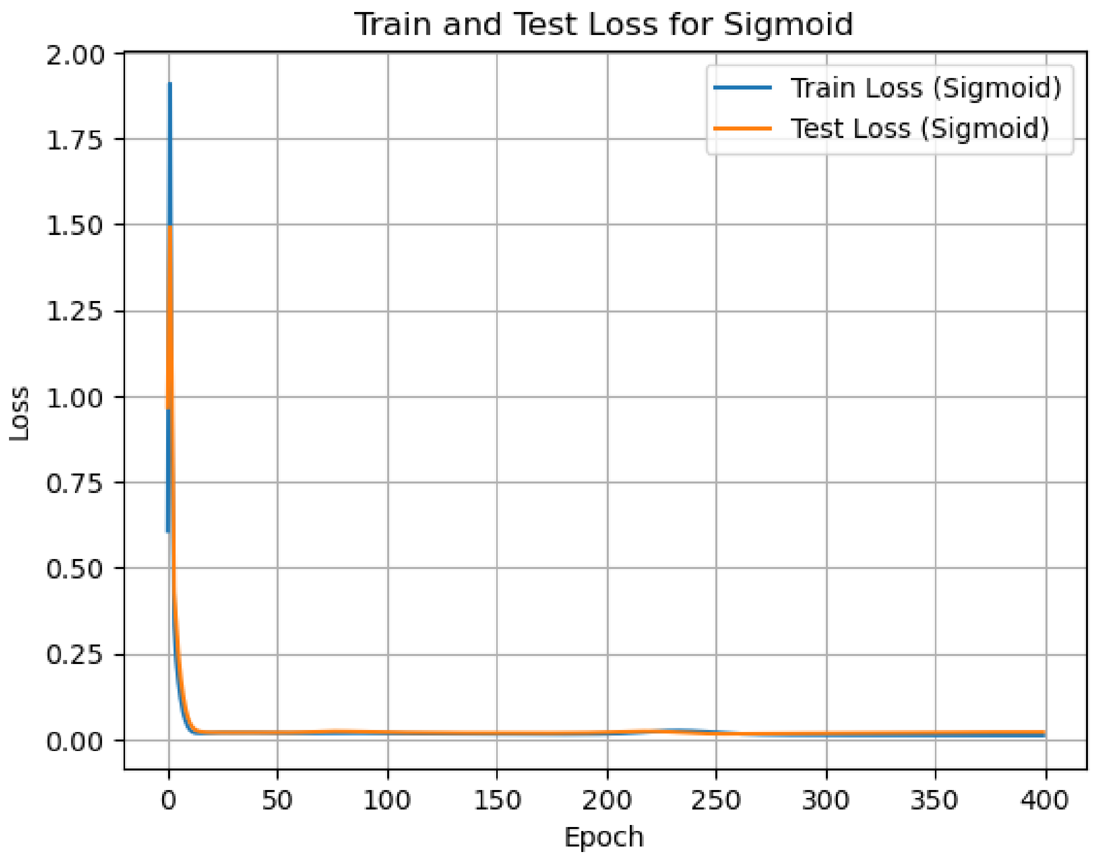
```
Testing ReLU activation function
Epoch: 000, Train Loss: 0.8893986, Test Loss: 1.0934367
Epoch: 100, Train Loss: 0.0005511, Test Loss: 0.0007331
Epoch: 200, Train Loss: 0.0003145, Test Loss: 0.0001574
Epoch: 300, Train Loss: 0.0000522, Test Loss: 0.0000628
```


Prediction with ReLU activation
True data
Model prediction (ReLU)
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.2
0.4
0.6
0.8
1.0

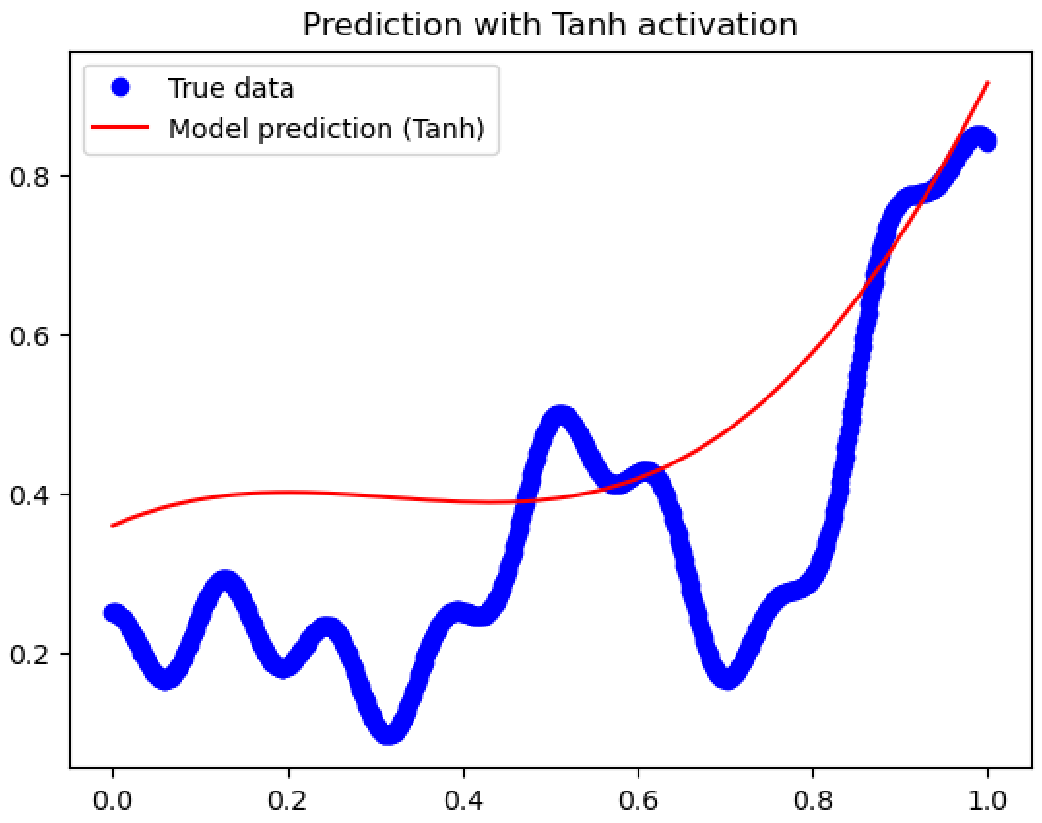


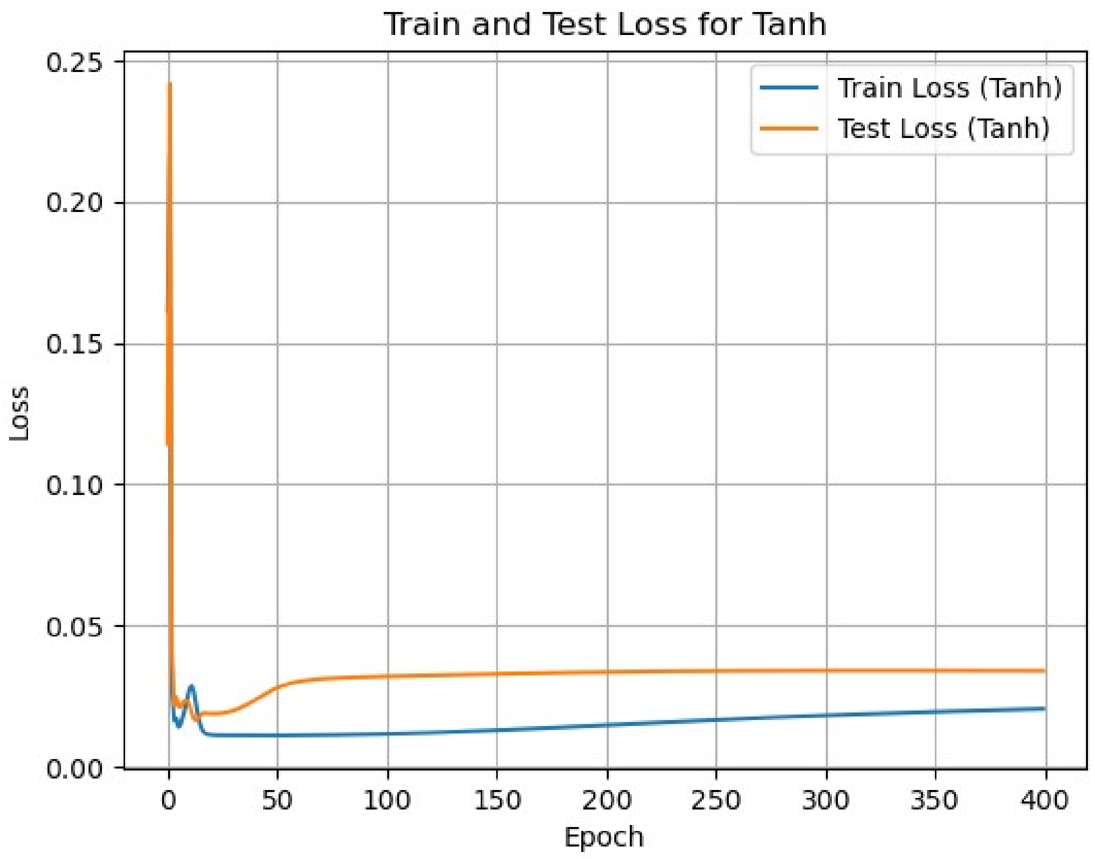
```
Testing Sigmoid activation function
Epoch: 000, Train Loss: 0.6068853, Test Loss: 0.9664480
Epoch: 100, Train Loss: 0.0169263, Test Loss: 0.0201778
Epoch: 200, Train Loss: 0.0152558, Test Loss: 0.0202157
Epoch: 300, Train Loss: 0.0119938, Test Loss: 0.0173541
```


Prediction with Sigmoid activation
True data
Model prediction (Sigmoid)
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.2
0.4
0.6
0.8
1.0



```
Testing Tanh activation function
Epoch: 000, Train Loss: 0.1610926, Test Loss: 0.1142495
Epoch: 100, Train Loss: 0.0116777, Test Loss: 0.0320398
Epoch: 200, Train Loss: 0.0148498, Test Loss: 0.0335571
Epoch: 300, Train Loss: 0.0182647, Test Loss: 0.0340703
```


Prediction with Tanh activation
True data
Model prediction (Tanh)
0.2
0.4
0.6
0.8
0.0
0.2
0.4
0.6
0.8
1.0



Compare different activation functions (sigmoid, Tanh, and ReLU): ReLU

```
# Import dependencies
import numpy as np
import torch
import torchvision
from torch.utils.data.dataset import Dataset
from torchvision import datasets, transforms
from torch import nn, optim
import matplotlib.pyplot as plt
```

Problem 4.4 Compare different network architectures

```
# ################## Part 1: Load data and create batch
##################
N_total = 600
N_train = 500
x = torch.unsqueeze(torch.linspace(0, 1, N_total), dim=1)
r = torch.randperm(N_total)
x = x[r, :]
y = 0.2 + 0.4 * torch.pow(x, 2) + 0.3 * x * torch.sin(15 * x) + 0.05 *
torch.cos(50 * x)

class CustomDataset(Dataset):
    def __init__(self, x, y):
        self.y = y
        self.x = x

    def __len__(self):
        return len(self.y)

    def __getitem__(self, idx):
        y1 = self.y[idx]
        x1 = self.x[idx]
        return (x1, y1)

# Change batch_size here to test different values
batch_size = 32  # Experiment with different batch sizes: 32, 64, 128
trainset = CustomDataset(x[0:N_train, :], y[0:N_train, :])
testset = CustomDataset(x[N_train:N_total, :], y[N_train:N_total, :])
train_loader = torch.utils.data.DataLoader(trainset,
batch_size=batch_size)
test_loader = torch.utils.data.DataLoader(testset,
batch_size=batch_size)

# ################## Part 2: Define Different Network Architectures
##################

# Architecture 1: One hidden layer with 16 neurons
model1 = nn.Sequential(nn.Linear(1, 16),
                       nn.ReLU(),
                       nn.Linear(16, 1))
```

```python
# Architecture 2: One hidden layer with 32 neurons
model2 = nn.Sequential(nn.Linear(1, 32),
                       nn.ReLU(),
                       nn.Linear(32, 1))

# Architecture 3: One hidden layer with 64 neurons
model3 = nn.Sequential(nn.Linear(1, 64),
                       nn.ReLU(),
                       nn.Linear(64, 1))

# Function to initialize weights
def init_weights(m):
    if isinstance(m, nn.Linear):
        m.weight.data.uniform_(-1, 1)
        m.bias.data.uniform_(-1, 1)

# Initialize weights for all models
model1.apply(init_weights)
model2.apply(init_weights)
model3.apply(init_weights)
```

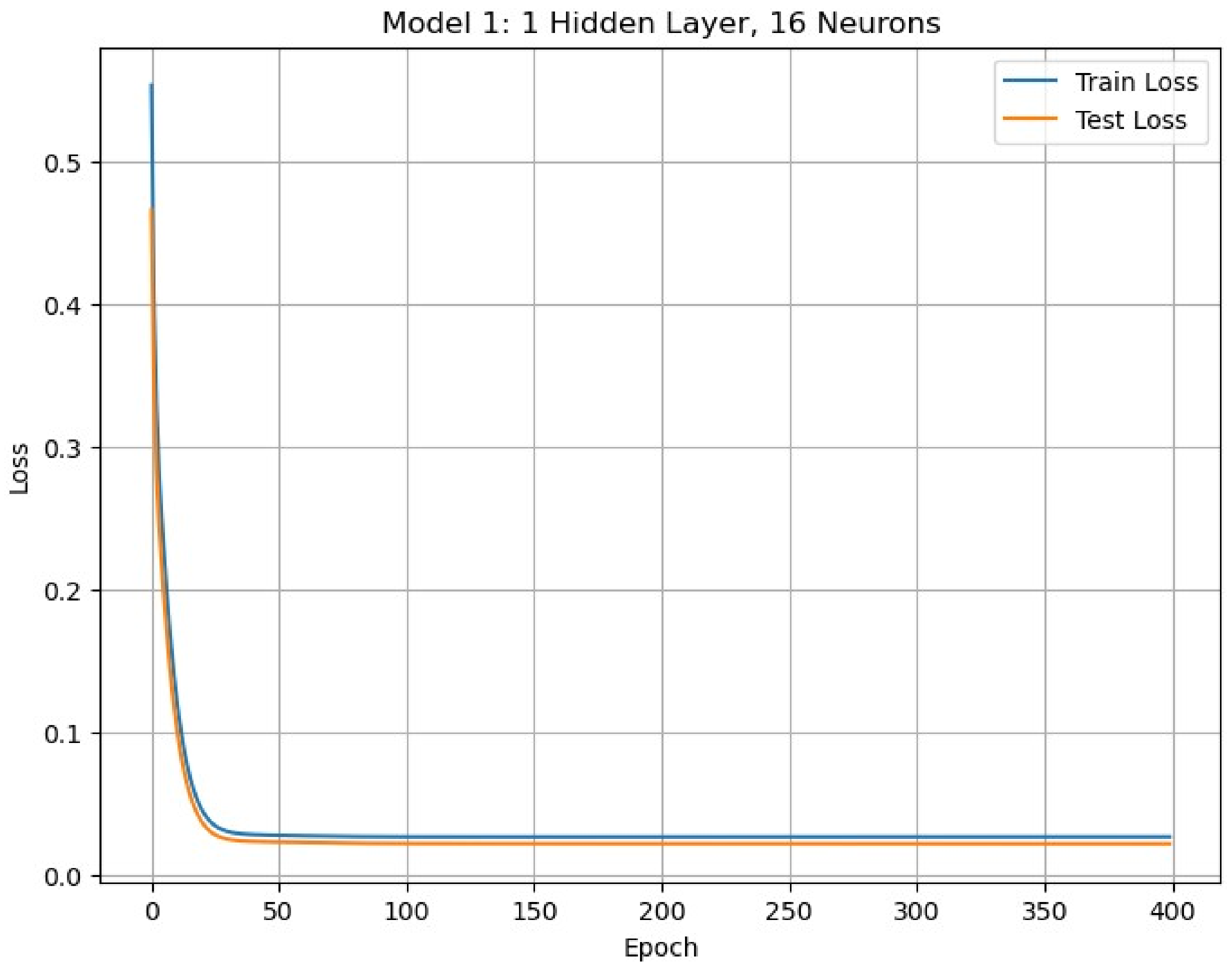
```
Sequential(
  (0): Linear(in_features=1, out_features=64, bias=True)
  (1): ReLU()
  (2): Linear(in_features=64, out_features=1, bias=True)
)
```

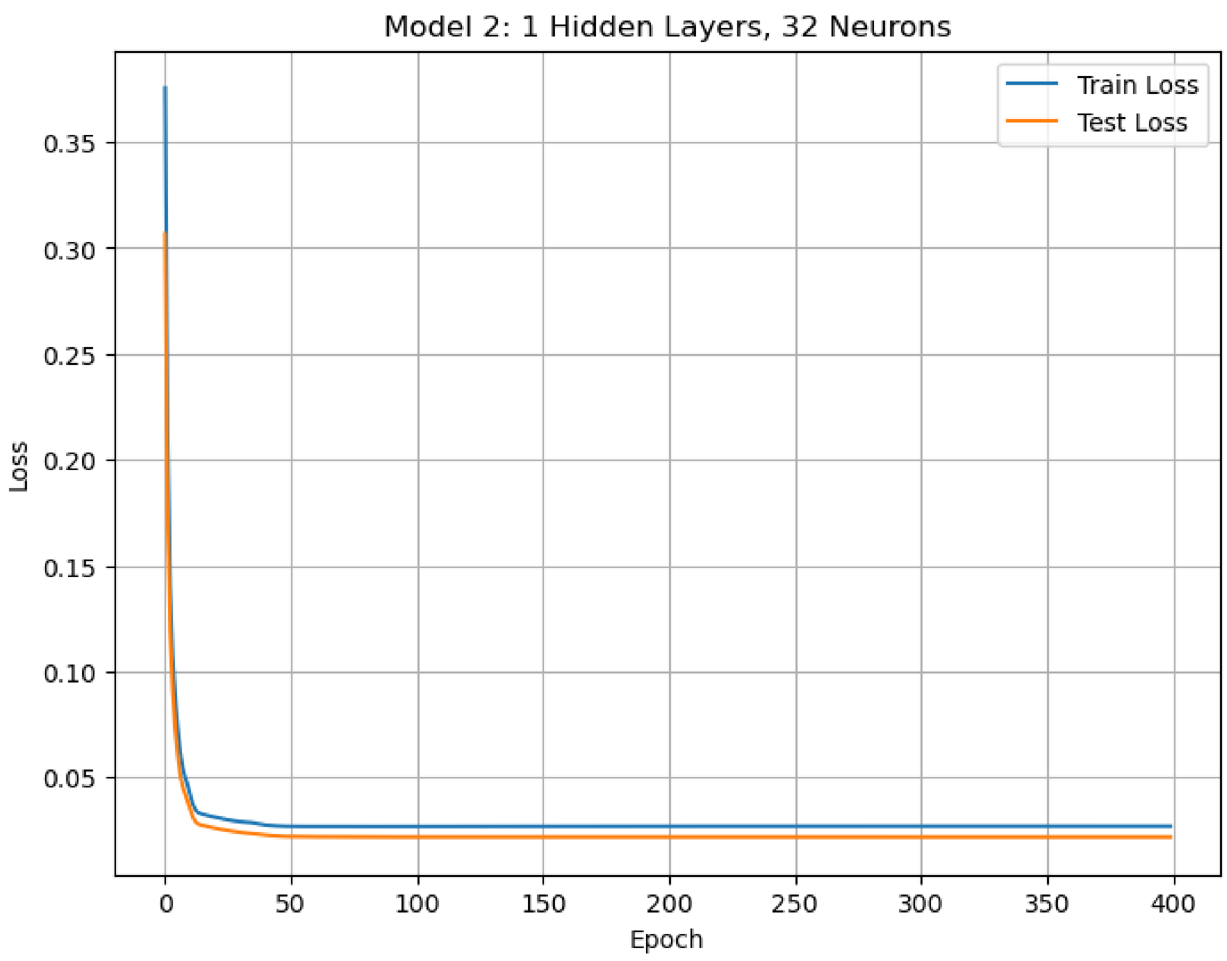
```python
# ################## Part 3: Define Loss and Optimizer
##################
criterion = torch.nn.MSELoss()

# You can adjust learning rates here
optimizer1 = torch.optim.Adam(model1.parameters(), lr=0.001)
optimizer2 = torch.optim.Adam(model2.parameters(), lr=0.001)
optimizer3 = torch.optim.Adam(model3.parameters(), lr=0.001)

# ################## Part 4: Train and Test Function
##################
def train_NN(model, optimizer):
    model.train()
    for images, labels in train_loader:
        out = model(images)
        loss = criterion(out, labels)
        loss.backward()
        optimizer.step()
        optimizer.zero_grad()
    return loss.item()

def test_NN(model, loader):
    model.eval()
    loss = 0
    with torch.no_grad():
        for images, labels in loader:
            out = model(images)
            loss += criterion(out, labels).item()
    return loss / len(loader)

# ################## Part 5: Train Models and Track Loss
##################
N_epoch = 400

def run_training(model, optimizer):
    train_loss = []
    test_loss = []
    for epoch in range(N_epoch):
        train_l = train_NN(model, optimizer)
        test_l = test_NN(model, test_loader)
        train_loss.append(train_l)
        test_loss.append(test_l)
        if epoch % 50 == 0:
            print(f'Epoch {epoch}, Train Loss: {train_l}, Test Loss:
{test_l}')
    return train_loss, test_loss

# Train all models
train_loss1, test_loss1 = run_training(model1, optimizer1)
train_loss2, test_loss2 = run_training(model2, optimizer2)
train_loss3, test_loss3 = run_training(model3, optimizer3)
```

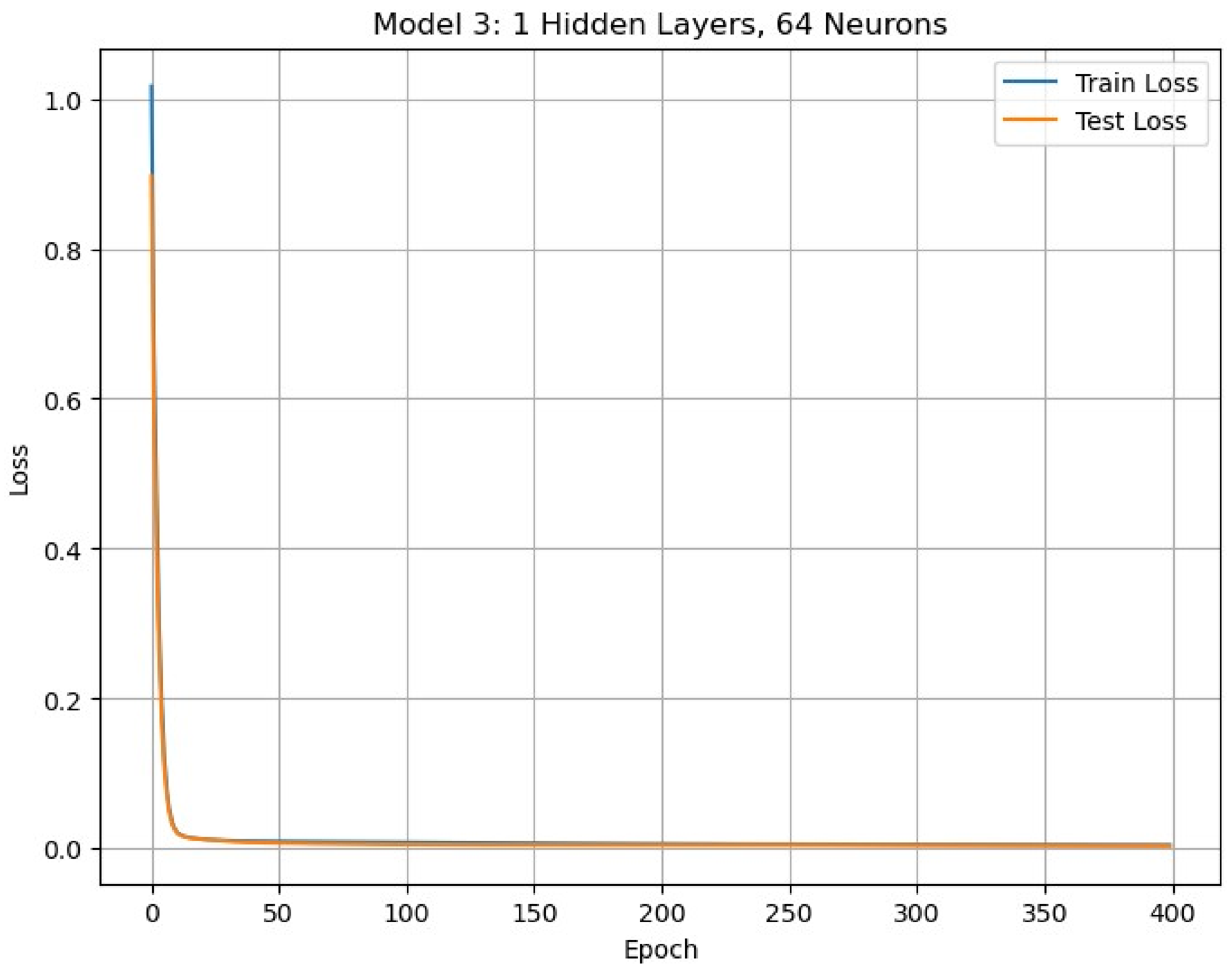
```
Epoch 0, Train Loss: 0.5534979701042175, Test Loss:
0.46619994193315506
Epoch 50, Train Loss: 0.027902286499738693, Test Loss:
0.023273158818483353
Epoch 100, Train Loss: 0.026946479454636574, Test Loss:
0.022135890321806073
Epoch 150, Train Loss: 0.026902684941887856, Test Loss:
0.02204208425246179
Epoch 200, Train Loss: 0.026918213814496994, Test Loss:
0.022009468637406826
Epoch 250, Train Loss: 0.02693040296435356, Test Loss:
0.021983390441164374
Epoch 300, Train Loss: 0.02694222331047058, Test Loss:
0.021972597111016512
Epoch 350, Train Loss: 0.026946749538183212, Test Loss:
0.021967295557260513
Epoch 0, Train Loss: 0.3755437433719635, Test Loss:
0.30665967613458633
Epoch 50, Train Loss: 0.026898186653852463, Test Loss:
0.02210253570228815
Epoch 100, Train Loss: 0.026842549443244934, Test Loss:
```

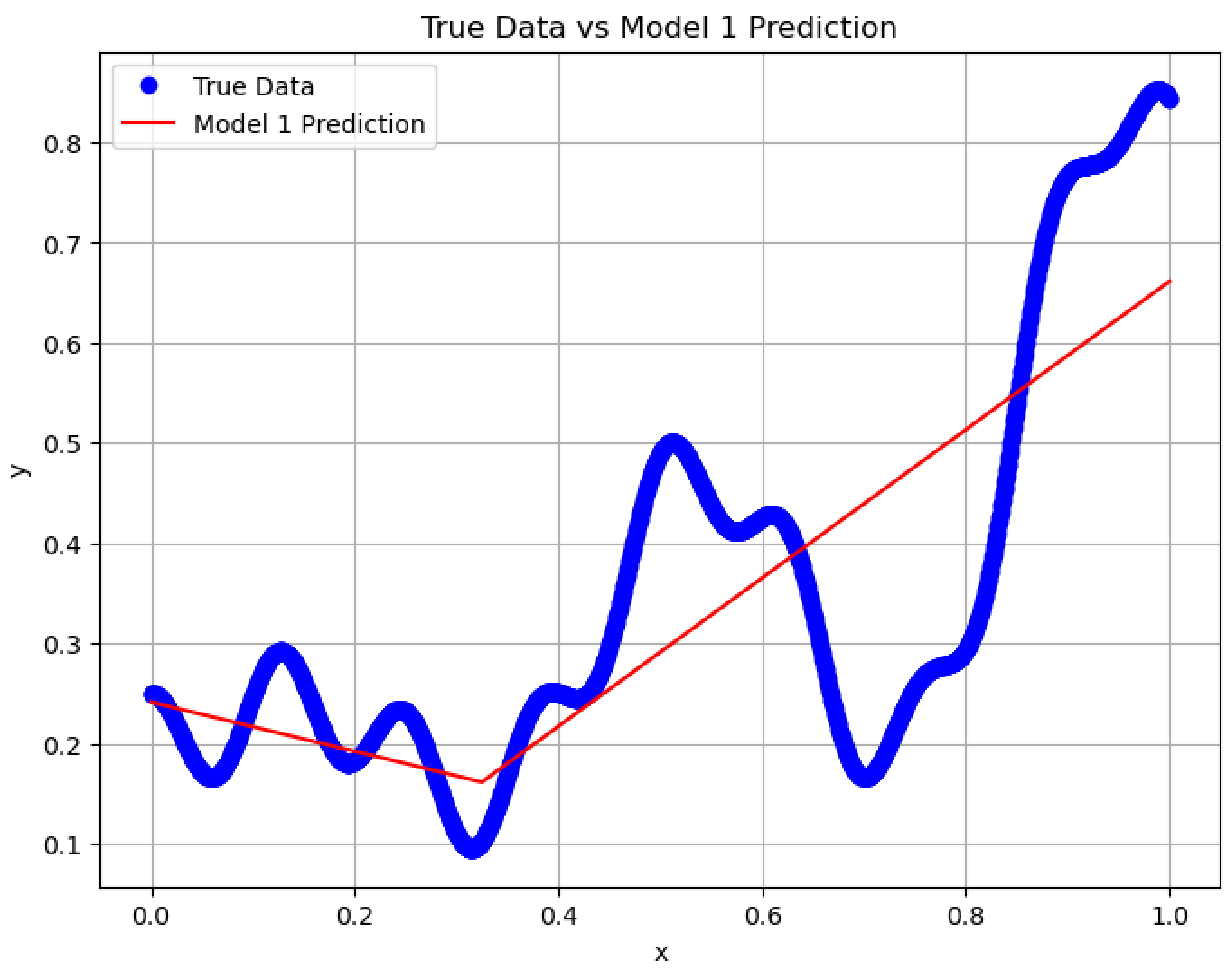
```
0.021872150478884578
Epoch 150, Train Loss: 0.026902323588728905, Test Loss:
0.02186821843497455
Epoch 200, Train Loss: 0.026947304606437683, Test Loss:
0.021860392997041345
Epoch 250, Train Loss: 0.02696821093559265, Test Loss:
0.021859925240278244
Epoch 300, Train Loss: 0.026974279433488846, Test Loss:
0.021858484717085958
Epoch 350, Train Loss: 0.026981288567185402, Test Loss:
0.021858786698430777
Epoch 0, Train Loss: 1.0164848566055298, Test Loss: 0.8969292938709259
Epoch 50, Train Loss: 0.009616399183869362, Test Loss:
0.007627377170138061
Epoch 100, Train Loss: 0.008318718522787094, Test Loss:
0.005719899199903011
Epoch 150, Train Loss: 0.007492950651794672, Test Loss:
0.005141673900652677
Epoch 200, Train Loss: 0.006459412164986134, Test Loss:
0.0051222327165305614
Epoch 250, Train Loss: 0.005756204016506672, Test Loss:
0.00505948078352958
Epoch 300, Train Loss: 0.005308591760694981, Test Loss:
0.004682371974922717
Epoch 350, Train Loss: 0.004956355784088373, Test Loss:
0.004287457326427102

# ################## Part 6: Plot Results ##################

# Plot for Model 1
plt.figure(figsize=(8, 6))
plt.plot(train_loss1, label='Train Loss')
plt.plot(test_loss1, label='Test Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.title('Model 1: 1 Hidden Layer, 16 Neurons')
plt.grid(True)
plt.show()

# Plot for Model 2
plt.figure(figsize=(8, 6))
plt.plot(train_loss2, label='Train Loss')
plt.plot(test_loss2, label='Test Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.title('Model 2: 1 Hidden Layers, 32 Neurons')
plt.grid(True)
plt.show()
```

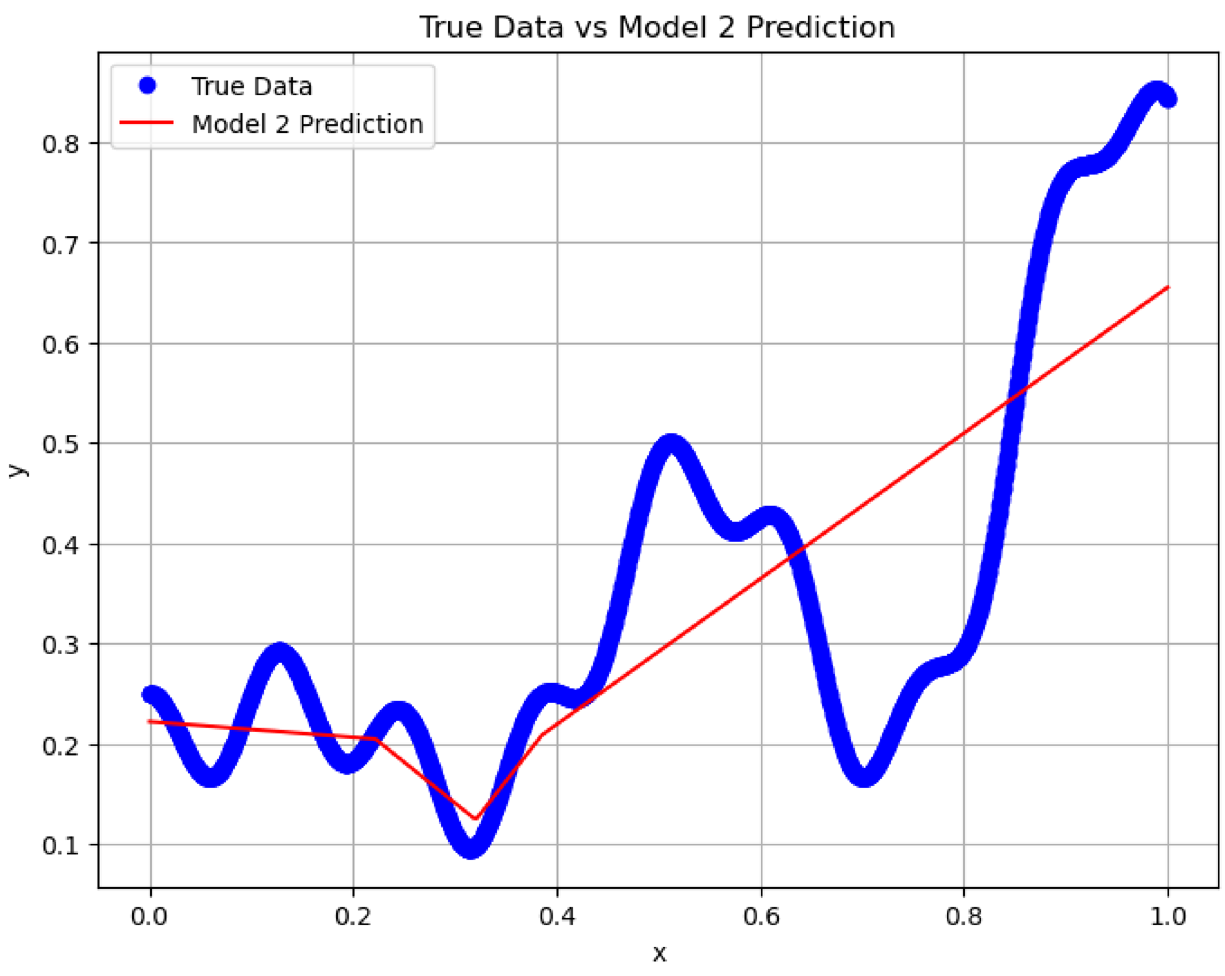
```
# Plot for Model 3
plt.figure(figsize=(8, 6))
plt.plot(train_loss3, label='Train Loss')
plt.plot(test_loss3, label='Test Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.title('Model 3: 1 Hidden Layers, 64 Neurons')
plt.grid(True)
plt.show()

# ################# Part 7: Plot Predictions #################

# Generate test data for prediction (true data and predictions)
x_test = torch.unsqueeze(torch.linspace(0, 1, 1999), dim=1)

# Predictions for Model 1
y_pred1 = model1(x_test).detach().numpy()

# Predictions for Model 2
y_pred2 = model2(x_test).detach().numpy()

# Predictions for Model 3
y_pred3 = model3(x_test).detach().numpy()

# True data (based on original function f(x))
y_true = 0.2 + 0.4 * torch.pow(x_test, 2) + 0.3 * x_test *
torch.sin(15 * x_test) + 0.05 * torch.cos(50 * x_test)

# Plot for Model 1 predictions vs true data
plt.figure(figsize=(8, 6))
plt.plot(x_test, y_true, 'bo', label='True Data')
plt.plot(x_test, y_pred1, 'r', label='Model 1 Prediction')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.title('True Data vs Model 1 Prediction')
plt.grid(True)
plt.show()

# Plot for Model 2 predictions vs true data
plt.figure(figsize=(8, 6))
plt.plot(x_test, y_true, 'bo', label='True Data')
plt.plot(x_test, y_pred2, 'r', label='Model 2 Prediction')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.title('True Data vs Model 2 Prediction')
plt.grid(True)
plt.show()

# Plot for Model 3 predictions vs true data
plt.figure(figsize=(8, 6))
plt.plot(x_test, y_true, 'bo', label='True Data')
plt.plot(x_test, y_pred3, 'r', label='Model 3 Prediction')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.title('True Data vs Model 3 Prediction')
plt.grid(True)
plt.show()
```

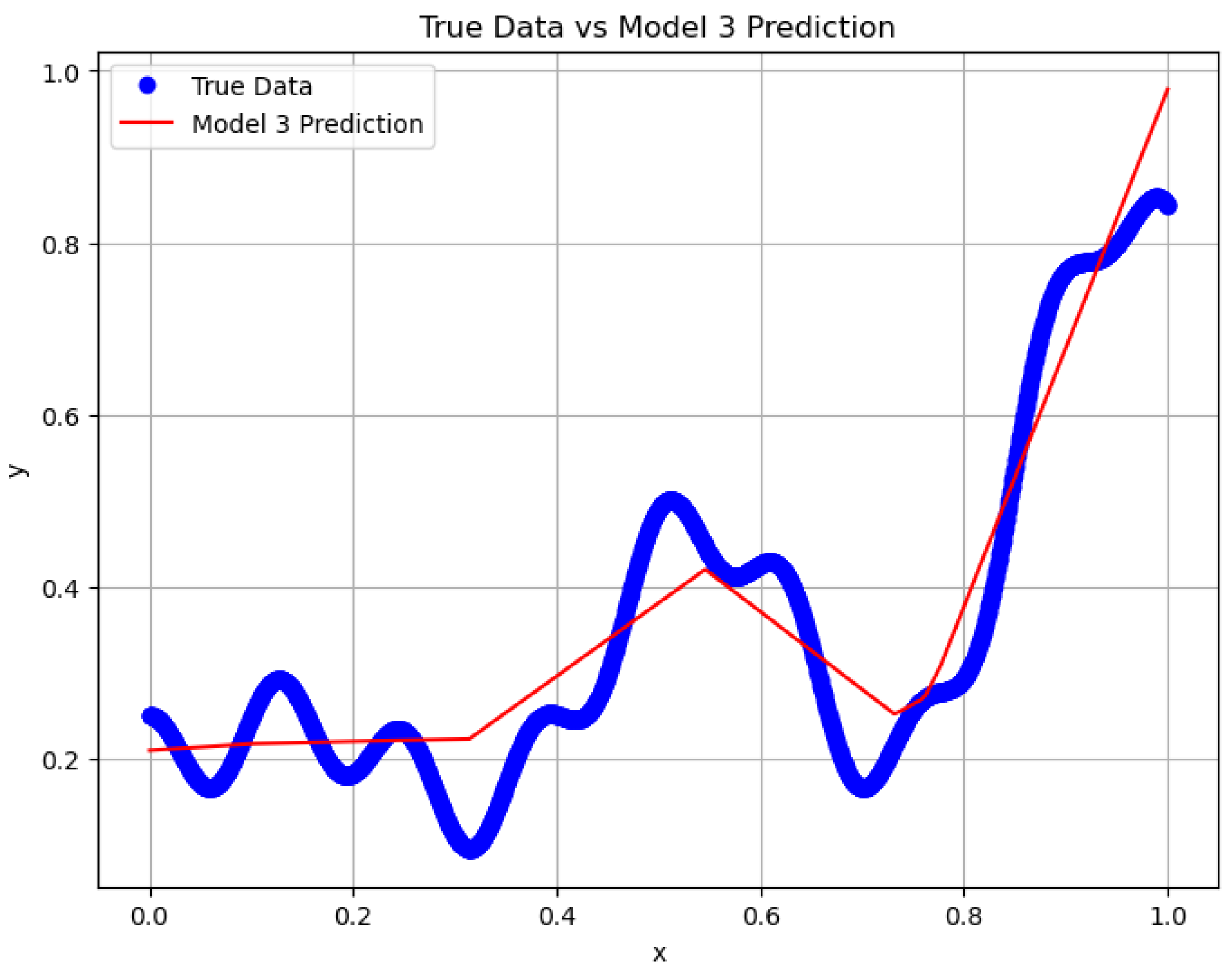



Model 2: 1 Hidden Layers, 32 Neurons
Train Loss
Test Loss
Loss
Epoch
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0
50
100
150
200
250
300
350
400


Model 3: 1 Hidden Layers, 64 Neurons
Train Loss
Test Loss
Loss
Epoch
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
250
300
350
400


True Data vs Model 1 Prediction
True Data
Model 1 Prediction
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
y
0.0
0.2
0.4
0.6
0.8
1.0
x


True Data vs Model 2 Prediction
True Data
Model 2 Prediction
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
y
0.0
0.2
0.4
0.6
0.8
1.0
x


True Data vs Model 3 Prediction
True Data
Model 3 Prediction
y
x
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0