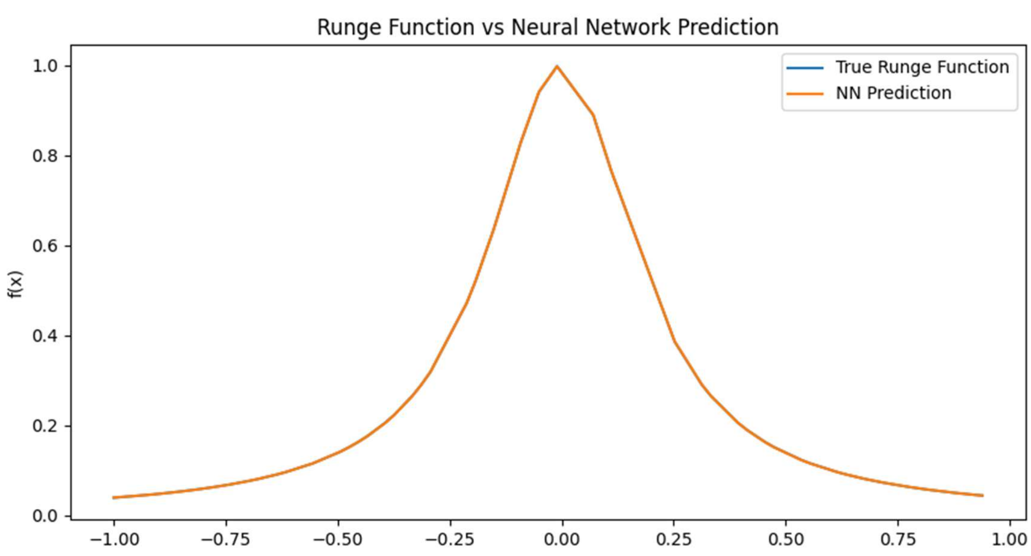
Programming assignment report

一、如何求出導數的：

在作業二的程式中，因為只有要算出最後的訓練函數，所以為了節省記憶體的佔用增加計算效率，所以是沒有記錄每一點在每一層的 gradient 數值，為了求出訓練函數的導函數，在計算完訓練函數後，要再透過 torch 函數啟動紀錄 gradient 的功能，並透過 backward 的運算計算出最後的導數，選用的點一樣是在作業二中的 validation set 中的點，而最後求誤差的部分同樣有算出 MSE 與 Maximum error。

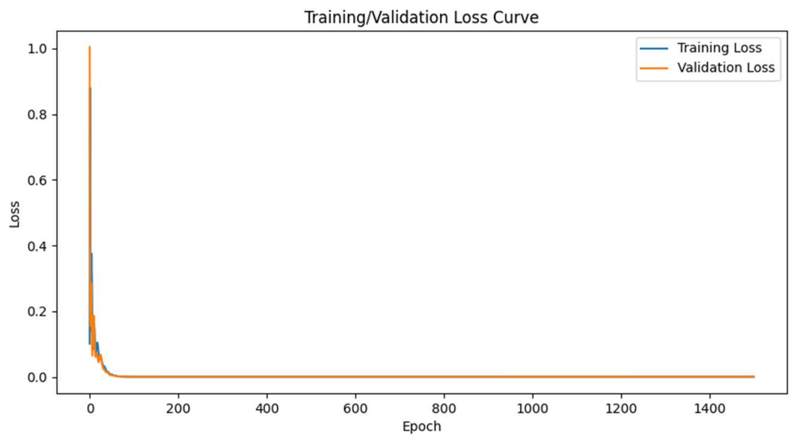
```
# NN函數導函數計算 (autograd)
x_val_torch = torch.FloatTensor(x_val_sorted).unsqueeze(1)
x_val_torch.requires_grad = True
with torch.no_grad():
    pass # 保證參數不更新
# 重新前向需能反向
y_pred = model(x_val_torch)
y_pred.backward(torch.ones_like(y_pred))
nn_deriv = x_val_torch.grad.detach().numpy().squeeze()
```

二、實際訓練後的結果：

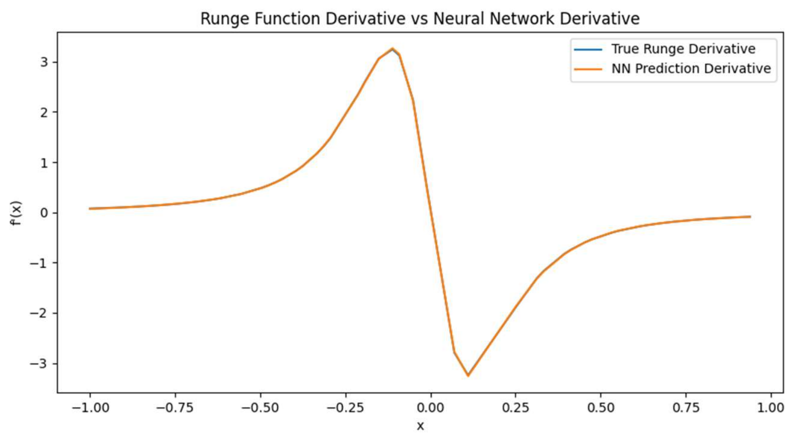


實際 Runge function 與預測函數比較

1500 次訓練中每 100 次的誤差與訓練後的函數/導函數誤差



Training/Validation Loss Curve



實際 Runge function 與預測函數的導和數比較

```
Epoch 100/1500, Training Loss: 0.000039, Validation Loss: 0.000036
Epoch 200/1500, Training Loss: 0.000009, Validation Loss: 0.000009
Epoch 300/1500, Training Loss: 0.000006, Validation Loss: 0.000006
Epoch 400/1500, Training Loss: 0.000004, Validation Loss: 0.000004
Epoch 500/1500, Training Loss: 0.000003, Validation Loss: 0.000003
Epoch 600/1500, Training Loss: 0.000002, Validation Loss: 0.000002
Epoch 700/1500, Training Loss: 0.000001, Validation Loss: 0.000001
Epoch 800/1500, Training Loss: 0.000001, Validation Loss: 0.000001
Epoch 900/1500, Training Loss: 0.000000, Validation Loss: 0.000000
Epoch 1000/1500, Training Loss: 0.000000, Validation Loss: 0.000000
Epoch 1100/1500, Training Loss: 0.000000, Validation Loss: 0.000000
Epoch 1200/1500, Training Loss: 0.000000, Validation Loss: 0.000000
Epoch 1300/1500, Training Loss: 0.000000, Validation Loss: 0.000000
Epoch 1400/1500, Training Loss: 0.000000, Validation Loss: 0.000000
Epoch 1500/1500, Training Loss: 0.000000, Validation Loss: 0.000000
Mean Squared Error (MSE): 0.000000
Maximum Error: 0.000651
Derivative MSE: 0.000057
Derivative Maximum Error: 0.022460
```

1500 次訓練中每 100 次的誤差與訓練後的函數/導函數誤差