Homework #1，繳交期限 2024/10/04
Couette Flow Regression

系所：工海所

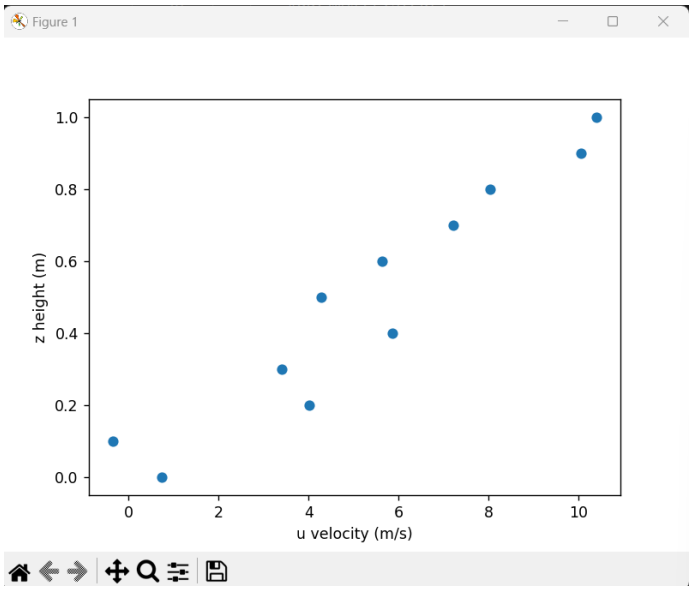
姓名：劉樺

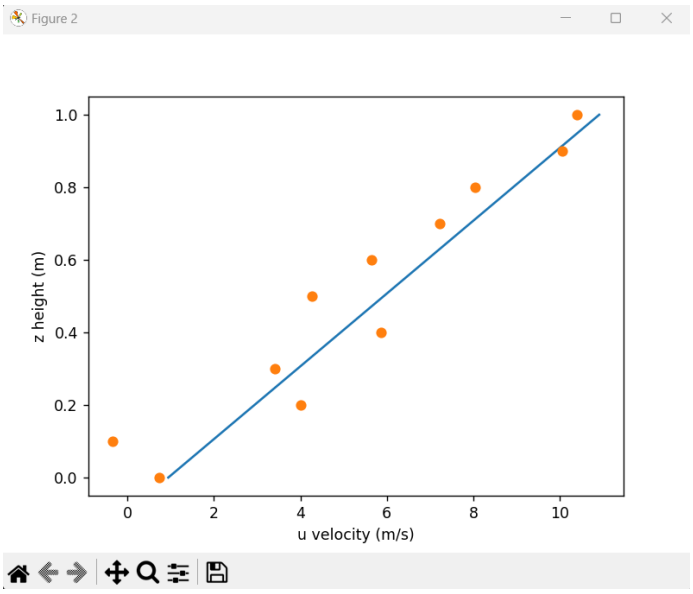
學號：R13525030

Basic

Code：

```
''' Homework!!! You have to fill the number in this block to achieve basic grade '''
params=torch.tensor([10, 1]) # weight and bias, arbitrary number
config={
    "number epoch": 10000, # number of epoch, suggest from 100 to 50000
    "learning rate": 1e-5, # learning rate, suggest from 1e-2 to 1e-5
   }
```





Homework #1，繳交期限 2024/10/04
Couette Flow Regression

系所：工海所

姓名：劉樺

學號：R13525030

Intermediate

(1)使用torch背向傳播自動計算梯度(backward)與SGD優化器進行更新參數。
(2)課堂上有提到使用L2 regularization可使weight之值降低，先將上述第一項任務完成，加入在其中加入L2 regularization。

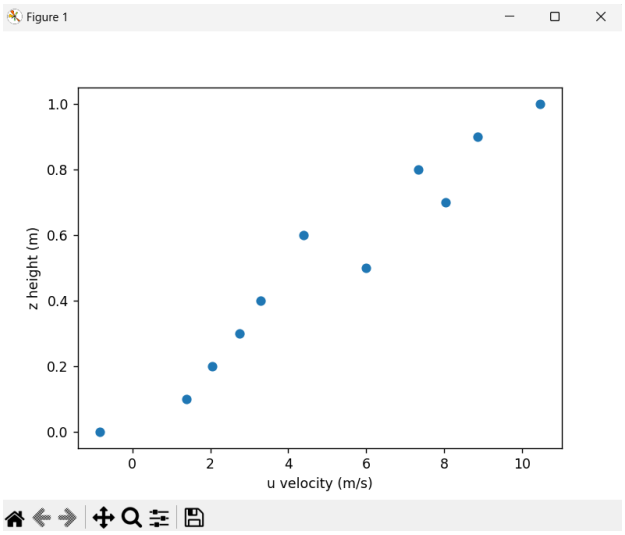
Code：

自動計算梯度
在 training loop 中，使用 loss.backward()來自動計算梯度
使用 SGD 優化器：
在 training loop 中，使用 optimizer.step()來更新參數
加入 L2 正則化：
設置了 weight_decay 參數，實現 L2 正則化在損失計算中，添加了 L2 正則化項，以便能夠顯示包含正則化的總損失
結果比較：
可以從結果看出每次 Loss 相較於 Basic 版本誤差較小

```python
# Mean Squared Error Loss with L2 Regularization
def loss_fn(x_p, y, params, weight_decay):
    mse_loss = ((x_p - y) ** 2).mean()
    l2_reg = weight_decay * (params[0]**2 + params[1]**2)  # L2 regularization term
    return mse_loss + l2_reg  # Total Loss

# Parameters initialization
params = torch.tensor([10.0, 1.0], requires_grad=True)  # weight and bias
learning_rate = 1e-5
weight_decay = 0.01  # L2 regularization factor

# Define optimizer
optimizer = optim.SGD([params], lr=learning_rate)

# Training function
def training(x, y, params, n_epoch, optimizer, weight_decay):
    for epoch in range(1, n_epoch + 1):
        w, b = params

        # Forward pass
        x_p = model(x, w, b)

        # Calculate loss
        loss = loss_fn(x_p, y, params, weight_decay)

        # Zero gradients, perform backpropagation, and update parameters
        optimizer.zero_grad()  # Clear previous gradients
        loss.backward()  # Backpropagation to compute gradients
        optimizer.step()  # Update parameters

        # Print loss every 100 epochs
        if epoch % 100 == 0:
            print('Epoch: %d, Loss: %f' % (epoch, loss.item()))

    return params

# Training configuration
config = {
    "number epoch": 10000,  # number of epochs
}

# Train the model
params = training(z, u, params, config["number epoch"], optimizer, weight_decay)
```

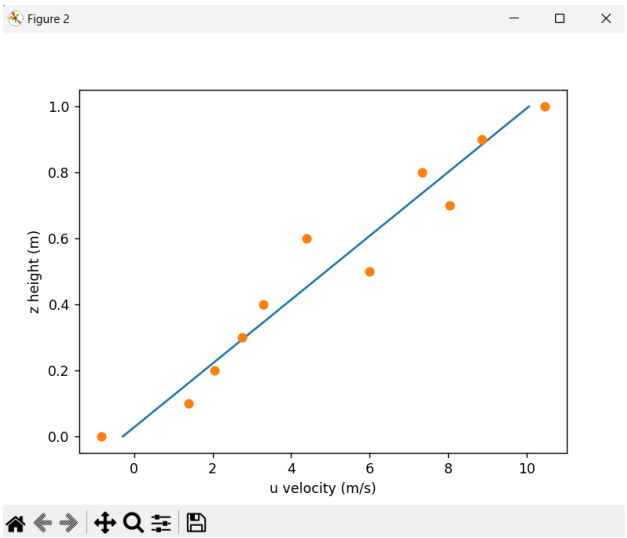
Basic

```
Epoch: 3997, Loss: 2.595068
Epoch: 3998, Loss: 2.594971
Epoch: 3999, Loss: 2.594874
Epoch: 4000, Loss: 2.594776
Epoch: 4001, Loss: 2.594679
Epoch: 4002, Loss: 2.594582
Epoch: 4003, Loss: 2.594485
Epoch: 4004, Loss: 2.594388
Epoch: 4005, Loss: 2.594291
Epoch: 4006, Loss: 2.594194
Epoch: 4007, Loss: 2.594097
Epoch: 4008, Loss: 2.594000
Epoch: 4009, Loss: 2.593903
Epoch: 4010, Loss: 2.593806
Epoch: 4011, Loss: 2.593709
Epoch: 4012, Loss: 2.593612
Epoch: 4013, Loss: 2.593515
Epoch: 4014, Loss: 2.593418
Epoch: 4015, Loss: 2.593321
Epoch: 4016, Loss: 2.593224
Epoch: 4017, Loss: 2.593127
Epoch: 4018, Loss: 2.593030
Epoch: 4019, Loss: 2.592933
Epoch: 4020, Loss: 2.592836
Epoch: 4021, Loss: 2.592739
Epoch: 4022, Loss: 2.592642
Epoch: 4023, Loss: 2.592545
Epoch: 4024, Loss: 2.592448
Epoch: 4025, Loss: 2.592351
Epoch: 4026, Loss: 2.592254
Epoch: 4027, Loss: 2.592157
Epoch: 4028, Loss: 2.592060
Epoch: 4029, Loss: 2.591963
Epoch: 4030, Loss: 2.591866
```



Intermediate

```
Epoch: 300, Loss: 2.6790664196014404
Epoch: 400, Loss: 2.1142420768737793
Epoch: 500, Loss: 1.7020081281661987
Epoch: 600, Loss: 1.4011383056640625
Epoch: 700, Loss: 1.181546688079834
Epoch: 800, Loss: 1.0212759971618652
Epoch: 900, Loss: 0.9042989611625671
Epoch: 1000, Loss: 0.8189213275909424
Epoch: 1100, Loss: 0.7566062808036804
Epoch: 1200, Loss: 0.7111225128173828
Epoch: 1300, Loss: 0.677925705909729
Epoch: 1400, Loss: 0.6536951065063477
Epoch: 1500, Loss: 0.6360092759132385
Epoch: 1600, Loss: 0.6230998039245605
Epoch: 1700, Loss: 0.6136763691902161
Epoch: 1800, Loss: 0.606797993183136
Epoch: 1900, Loss: 0.6017769575119019
Epoch: 2000, Loss: 0.5981116890907288
Epoch: 2100, Loss: 0.5954359173774719
Epoch: 2200, Loss: 0.593482255935669
Epoch: 2300, Loss: 0.5920560359954834
Epoch: 2400, Loss: 0.5910146236419678
Epoch: 2500, Loss: 0.5902542471885681
Epoch: 2600, Loss: 0.5896990299224854
Epoch: 2700, Loss: 0.5892934799194336
Epoch: 2800, Loss: 0.5889972448348999
Epoch: 2900, Loss: 0.58878093957901
Epoch: 3000, Loss: 0.5886229276657104
```



Homework #1，繳交期限 2024/10/04
Couette Flow Regression

系所：工海所

姓名：劉樺

學號：R13525030

Advanced

(1)為了解模型是否有(overfitting)之情形出現，可將訓練資料分成training set與validation set以檢查損失函數之下降情況判斷訓練是否過擬和。將資料透過任何方法分割成training set與validation set，並印出validation loss。
(2)起透過增加一個變數early stop，讓validate loss若沒持續下降一定次數則提早停止訓練。

Code：

```
PS D:\資料備份-勿刪\Desktop\Pytorch> python dpl1_adv.py
Epoch: 100, Train Loss: 6.9003, Val Loss: 3.4178
Epoch: 200, Train Loss: 4.9596, Val Loss: 1.7956
Epoch: 300, Train Loss: 3.6845, Val Loss: 1.3463
Epoch: 400, Train Loss: 2.7819, Val Loss: 1.1201
Epoch: 500, Train Loss: 2.1425, Val Loss: 1.0128
Epoch: 600, Train Loss: 1.6895, Val Loss: 0.9800
Early stopping at epoch 619
```