Topics
My posts
More
CATEGORIES
General
Site Feedback
All categories

# 2025 USA-NA-AIO Round 2, Problem 1, Part 10

USAAIO May 2025

## Part 10 (10 points, coding task)

**This part asks you to do a mini-batch training of the model.**

1. Set the parameter in the PDE `alpha = 0.1`. (This is not for learning. In PINN, we know the exact form of a PDE. We just need neural networks to help us solve it.)

2. Set the number of epochs as 1000.

3. Define

   ```
   device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
   ```

4. While iterating over epochs, use `tqdm` to track the progress:

   ```
   for epoch in tqdm(range(num_epochs)):
   ```

5. In each epoch,

   (1) Configure the model to the training mode.

   (2) Iterate over all mini-batches of `dataset_train_PDE`.

   (3) For each of the above mini-batch of the PDE dataset, while computing the total loss function, you also need to use **all** data in `dataset_train_IC` and `dataset_train_BC`.

   (4) Do all these tasks on **GPU**.

6. In each epoch, after training over all mini-batches, if the epoch index is divisible by 100, do the following tasks:

Skip to main content

(1) Configure the model to the evaluation mode.

(2) Compute the total loss over the entire three datasets: `dataset_train_PDE`, `dataset_train_IC`, `dataset_train_BC`.

(3) Print the epoch index, the residual loss from PDE, the IC loss, the BC loss, and the total loss.

(4) Do all these tasks on **CPU**.

---

**USAAIO** **May 2025**

```
### WRITE YOUR SOLUTION HERE ###

alpha = 0.1
num_epochs = 1000
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

for epoch in tqdm(range(num_epochs)):
    model.train()
    model.to(device)
    for batch_PDE in dataloader_PDE:
        batch_PDE = batch_PDE.to(device)
        optimizer.zero_grad()
        U_PDE = model(batch_PDE)
        U_t_and_x_PDE = autograd.grad(torch.sum(U_PDE), batch_PDE, create_graph=T
        U_t_PDE = U_t_and_x_PDE[:,0]
        U_x_PDE = U_t_and_x_PDE[:,1]
        U_xx_PDE = autograd.grad(torch.sum(U_x_PDE), batch_PDE, create_graph=True
        loss_PDE = torch.mean((U_t_PDE - alpha * U_xx_PDE)**2)

        U_IC = model(dataset_train_IC.to(device))
        loss_IC = torch.mean((U_IC - u_IC.to(device))**2)

        U_BC = model(dataset_train_BC.to(device))
        loss_BC = torch.mean(U_BC**2)
```

Skip to main content

```
            loss = loss_PDE + loss_IC + loss_BC
            loss.backward()
            optimizer.step()

        if epoch % 100 == 0:
            model.eval()
            model.to('cpu')
            batch_PDE = next(iter(dataloader_PDE))
            batch_PDE.requires_grad_(True)
            U_PDE = model(batch_PDE)
            U_t_and_x_PDE = autograd.grad(torch.sum(U_PDE), batch_PDE, create_graph=T
            U_xx_PDE = autograd.grad(torch.sum(U_t_and_x_PDE[:,1]), batch_PDE, create
            loss_PDE = torch.mean((U_t_and_x_PDE[:,0] - alpha * U_xx_PDE)**2)

            with torch.no_grad():
                U_IC = model(dataset_train_IC)
                loss_IC = torch.mean((U_IC - u_IC)**2)
                U_BC = model(dataset_train_BC)
                loss_BC = torch.mean(model(dataset_train_BC)**2)
                loss = loss_PDE + loss_IC + loss_BC

            print(f"{loss_PDE.item():.4e}, {loss_IC.item():.4e}, {loss_BC.item():.4e}
            print(f"Epoch {epoch}, Loss: {loss.item():.4e}")


    """ END OF THIS PART """
```

Skip to main content

## ✨ Related topics

| Topic | Replies | Activity |
|---|---|---|

| Topic | Replies | Activity |
|---|---|---|
| 2025 USA-NA-AIO Round 2, Problem 1, Part 2 | 2 | Oct 2025 |
| 2025 USA-NA-AIO Round 2, Problem 1, Part 12 | 1 | May 2025 |
| 2025 USA-NA-AIO Round 2, Problem 1, Part 1 | 1 | May 2025 |
| 2025 USA-NA-AIO Round 2, Problem 1, Part 11 | 1 | May 2025 |
| 2025 USA-NA-AIO Round 1, Problem 2, Part 13 | 2 | Oct 2025 |