CS726 Programming Assignment – 4 Report

Saksham Rathi (22B1003)
Sharvanee Sonawane (22B0943)
Deeksha Dhiwakar (22B0988)
Department of Computer Science,
Indian Institute of Technology Bombay

Task 0: Environment Setup and Result Reproduction

Here is how the model was loaded:

```
model = EnergyRegressor(FEAT_DIM).to(DEVICE)
```

And here is how the trained weights were loaded:

```
model.load_state_dict(torch.load('../trained_model_weights.pth', map_location=DEVICE))
```

Here is the output generated when we run the script:

```
Using device: cuda

--- Model Architecture ---
EnergyRegressor(
  (net): Sequential(
    (0): Linear(in_features=784, out_features=4096, bias=True)
    (1): ReLU(inplace=True)
    (2): Linear(in_features=4096, out_features=2048, bias=True)
    (3): ReLU(inplace=True)
    (4): Linear(in_features=2048, out_features=1024, bias=True)
    (5): ReLU(inplace=True)
    (6): Linear(in_features=1024, out_features=512, bias=True)
    (7): ReLU(inplace=True)
    (8): Linear(in_features=512, out_features=256, bias=True)
    (9): ReLU(inplace=True)
    (10): Linear(in_features=256, out_features=128, bias=True)
    (11): ReLU(inplace=True)
    (12): Linear(in_features=128, out_features=64, bias=True)
    (13): ReLU(inplace=True)
    (14): Linear(in_features=64, out_features=32, bias=True)
    (15): ReLU(inplace=True)
    (16): Linear(in_features=32, out_features=16, bias=True)
    (17): ReLU(inplace=True)
    (18): Linear(in_features=16, out_features=8, bias=True)
    (19): ReLU(inplace=True)
    (20): Linear(in_features=8, out_features=4, bias=True)
    (21): ReLU(inplace=True)
```

```
    (22): Linear(in_features=4, out_features=2, bias=True)
    (23): ReLU(inplace=True)
    (24): Linear(in_features=2, out_features=1, bias=True)
  )
)
------------------------

Loading dataset from ../A4_test_data.pt...
Dataset loaded in 0.17s. Shape: x=torch.Size([100000, 784]), energy=torch.Size([100000,
1])

--- Test Results ---
Loss: 288.1554
--- Script Finished ---
```

As shown in the output above, the model was and dataset were loaded successfully. The model architecture is a feedforward neural network with 24 layers, and the dataset contains 100,000 samples. The loss value of 288.1554 indicates the performance of the model on the test dataset.

Task 1: MCMC Sampling Implementation

Apart from calculating the acceptance probability, and the burn in time, we calculate the mean probability of the samples which are generated. This is basically the average of $e^{-E(x)}$, where x is the sample (the expression is un-normalized). This lets us know, whether after the burn-in, we were able to reach the high probability regions or not. The heat map below summarizes all the experiments we had run to check the performance of both the algorithms:

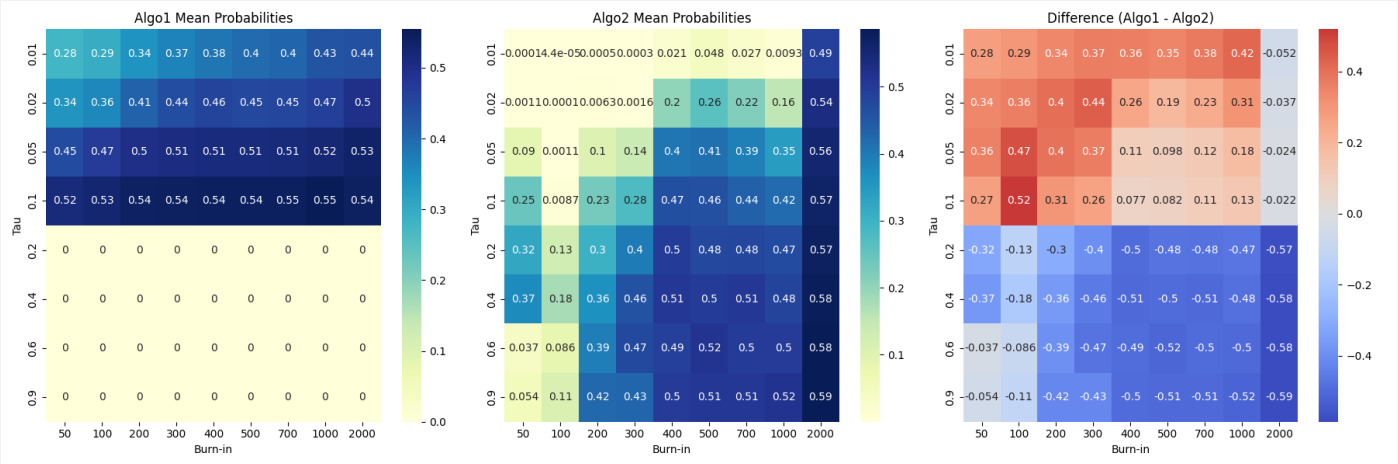
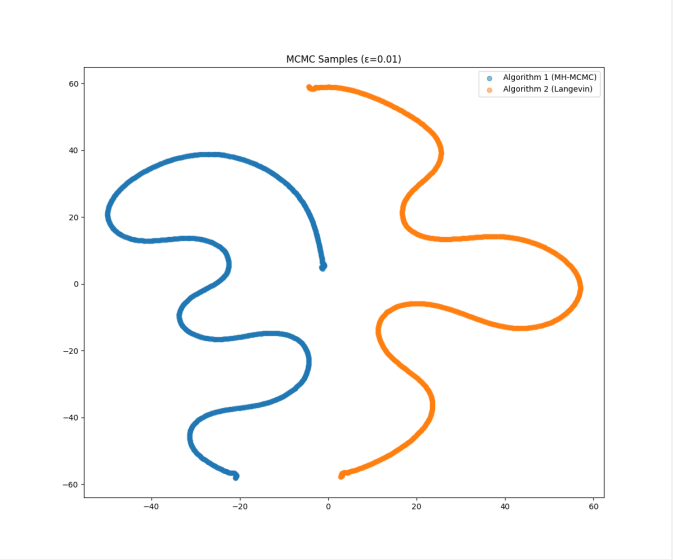


Figure 1: Heatmap showing the mean probability across various burn-in values and the number of samples for both the algorithms

Firstly, we observe that maxima of this probability is detected around $\tau = 0.1$, for both the algorithms, across various burn-in values.

Also, the probabilities typically increase, as we have more burn-in samples (which is actually obvious, because the actual samples are generated after this burn-in period, so less likely samples are already rejected).

Next, we used t-Sne, to plot these high dimensional samples into 1-D or 2-D. Here we show some of the results. Each image has samples plotted from both the algorithms:



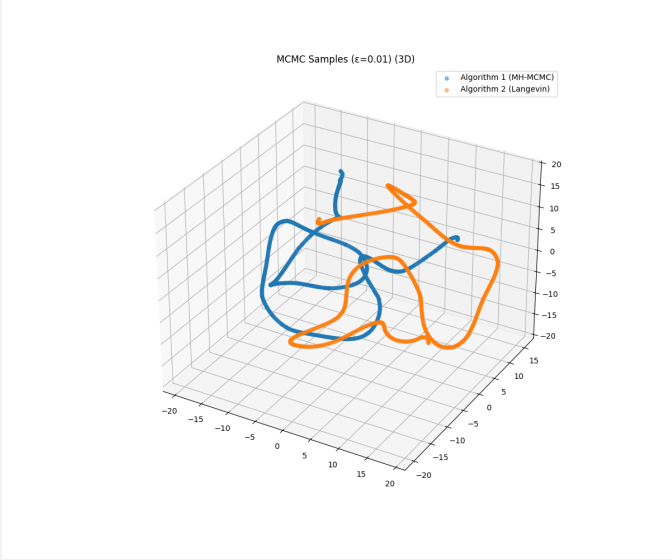
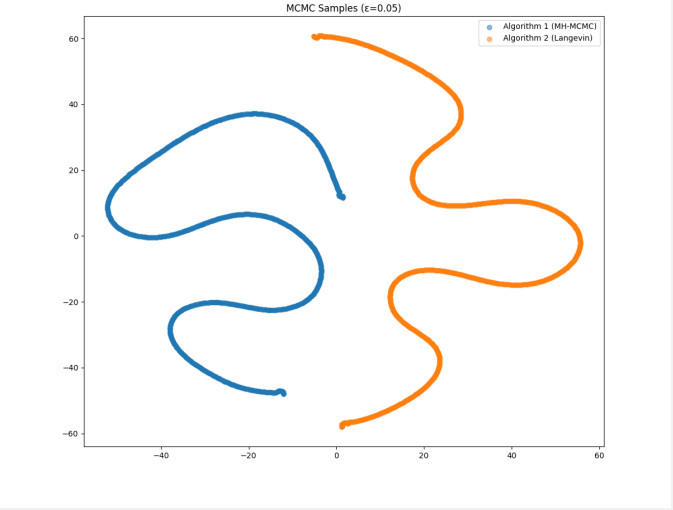


Figure 2: $\tau = 0.01$, Number of samples = 1000, burn-in samples = 1000



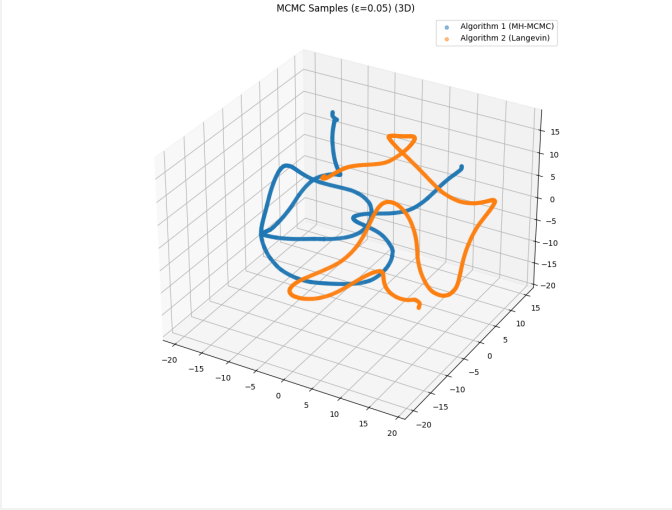
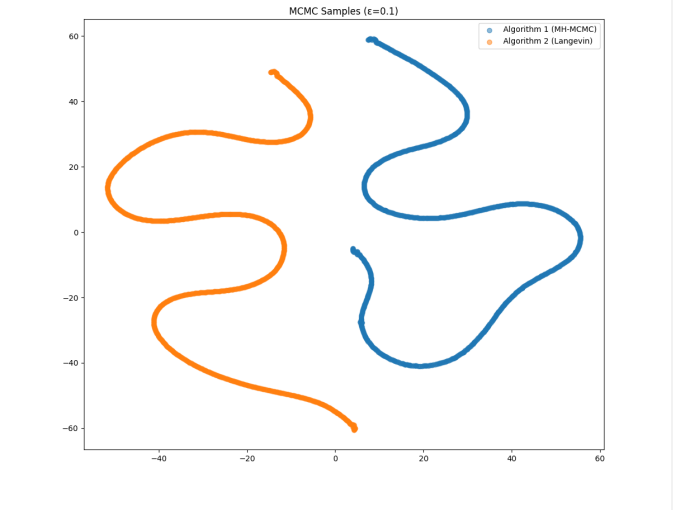


Figure 3: $\tau = 0.05$, Number of samples = 1000, burn-in samples = 1000



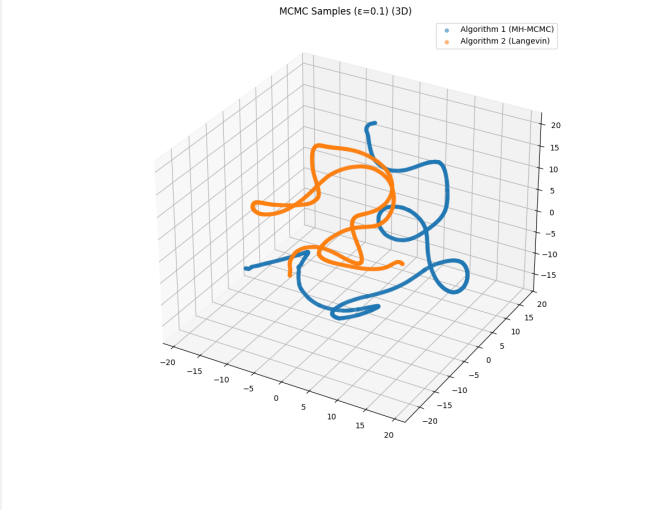
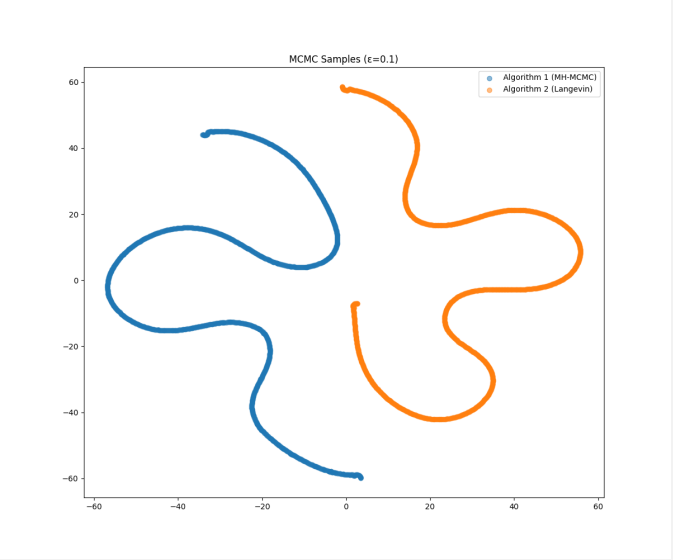


Figure 4: $\tau = 0.1$, Number of samples = 1000, burn-in samples = 50



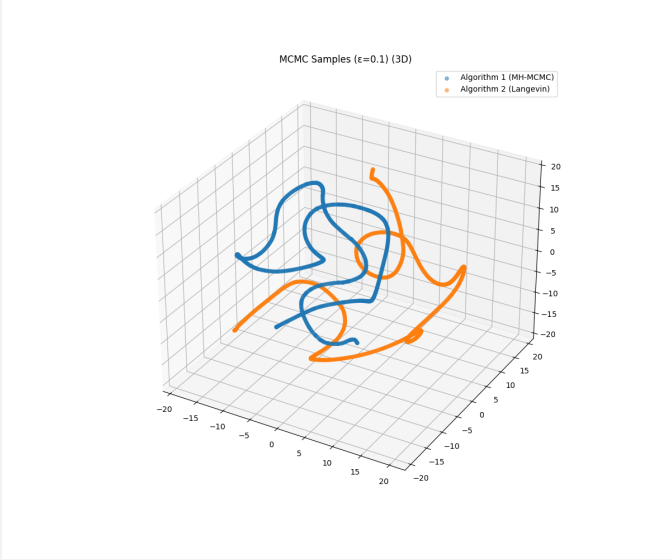
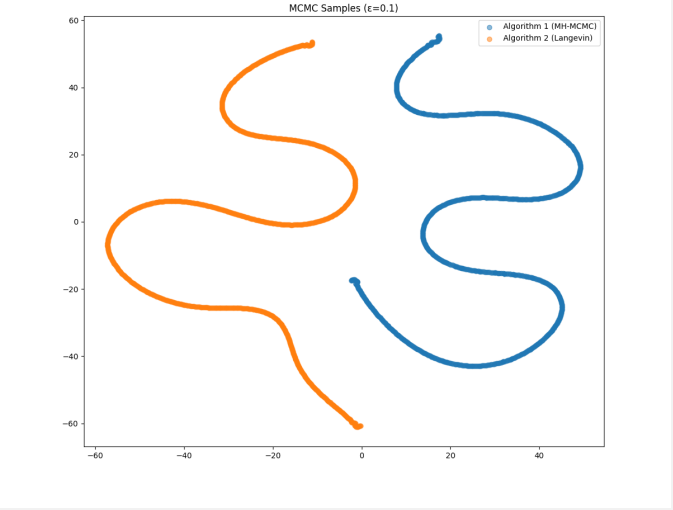


Figure 5: $\tau = 0.1$, Number of samples = 1000, burn-in samples = 200



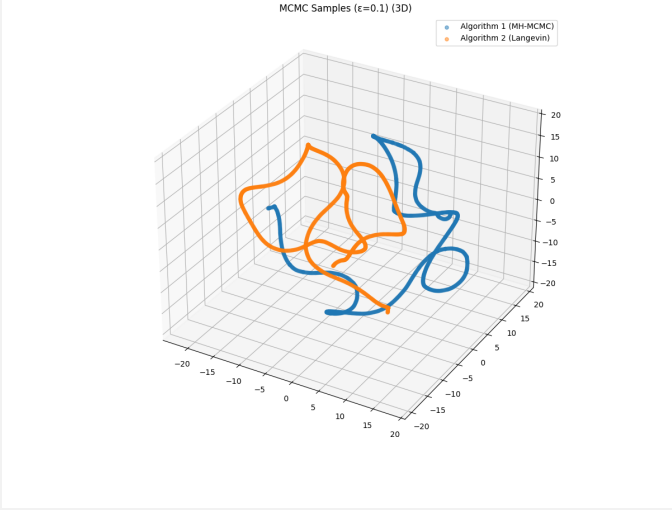
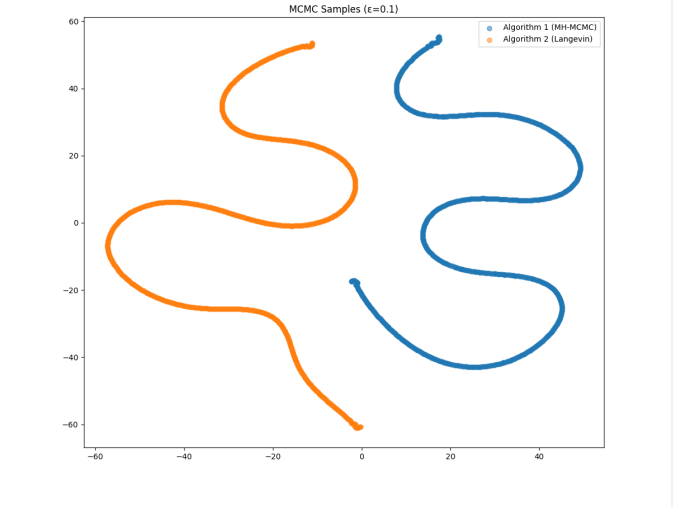


Figure 6: $\tau = 0.1$, Number of samples = 1000, burn-in samples = 500



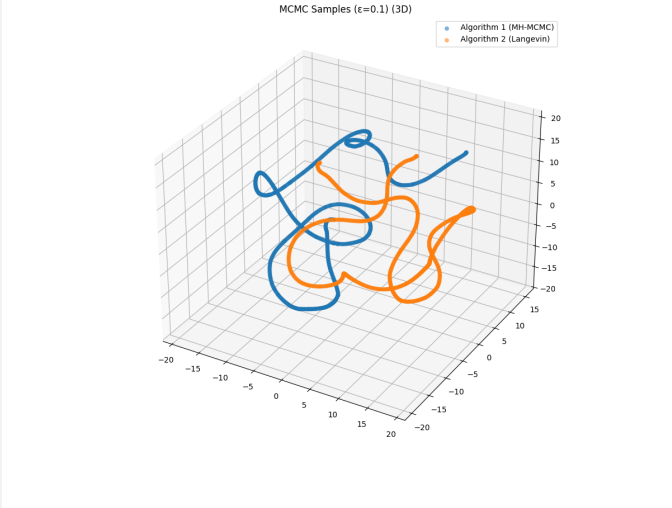
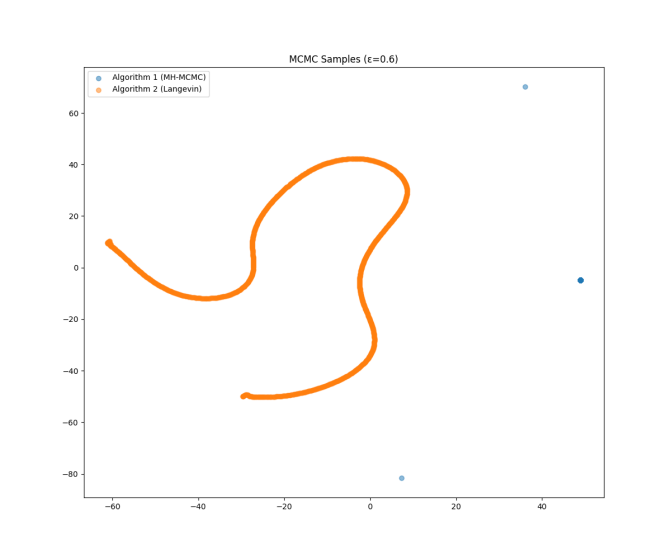


Figure 7: $\tau = 0.1$, Number of samples = 1000, burn-in samples = 1000



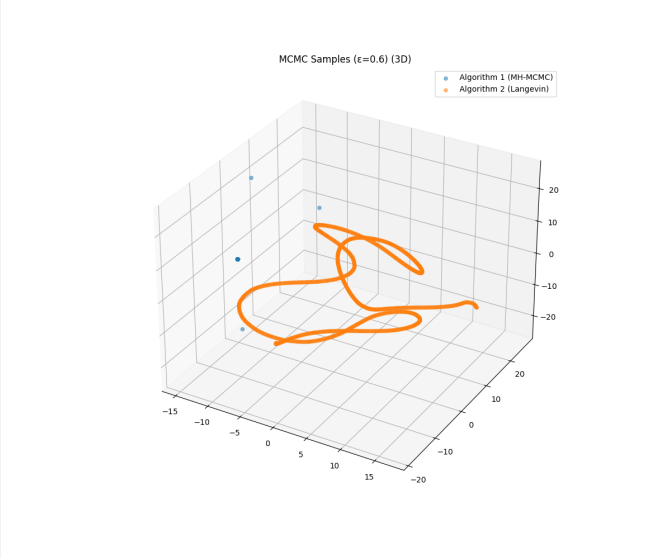


Figure 8: $\tau = 0.6$, Number of samples = 1000, burn-in samples = 1000

As, we can see, algorithm-1 performs quite badly, in case the value of τ is more than 0.4 (the acceptance probability is roughly 0, so we aren't able to move much from our initial guess). The detailed result are present in the file `TASK-0-1/out`, which shows the burn-in times, along with acceptance probabilities, for all the experiments done.