PROJECT1:INVERSE PROBLEMS WITH ML SURROGATE MODELS

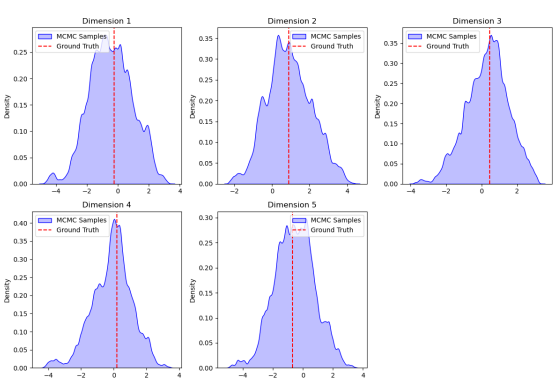
CHU ZHUYIHENG 12455799

1. Solve an inverse problem in the numerical Lorenz96 model

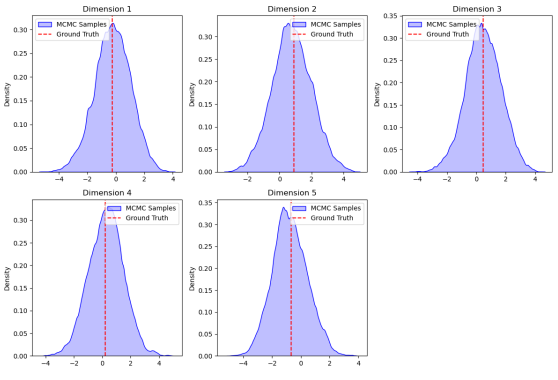
The script `mcmc_num.py` has completed the task, where the adjustable parameter is `num_samples`, which specifies the number of samples drawn using the MCMC method. I used the latter half of the samples as output values for subsequent analysis, while discarding the first half as the burn-in period.
x0 represents the true initial condition, and `initial_state` is the randomly chosen starting state for the MCMC method.
By analyzing the relationship between the sum of squared errors (SSE) between the true value and the mean of the samples obtained through MCMC and the parameter `num_samples`, I evaluate whether the results of the MCMC method have converged.

num-samples	20000	200000	400000	4000000	30000000
SSE	27.825963	22.311534	13.838954	1.86841	0.360134

The SSE of the random initial value is 25.031237. Therefore, it is not difficult to see that when num-samples reaches 400000, the accuracy of mcmc sampling is guaranteed to a certain extent (at least more accurate than the initial value)
The following is the distribution diagram of the first five dimensions and the comparison diagram of all dimensions when num-samples=4000000 and 30000000.

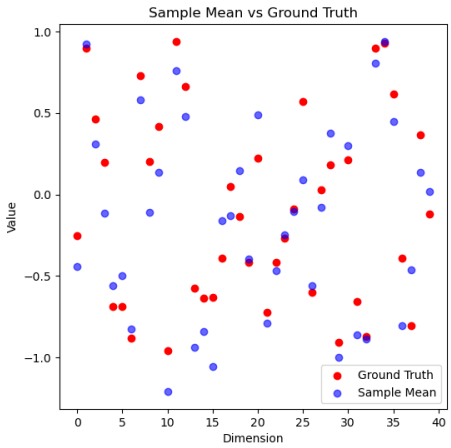


(A) First 5 Dimension Distribution (4M samples)

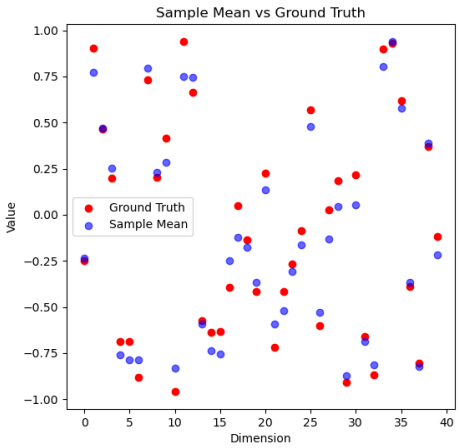


(B) First 5 Dimension Distribution (30M samples)

FIGURE 1. Full observations result(part1)



(A) Comparison of All Dimensions (4M samples)



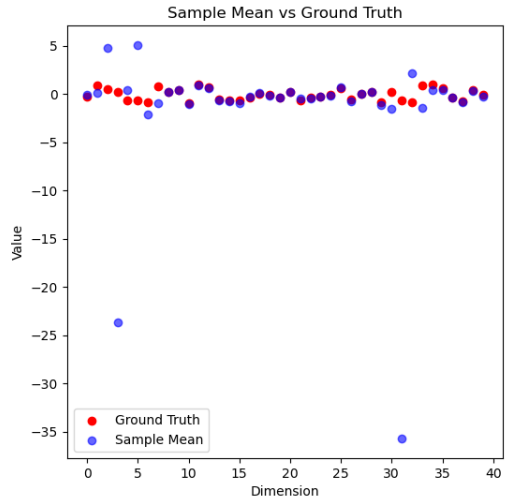
(B) Comparison of All Dimensions (30M samples)

FIGURE 2. Full observations result(part2)

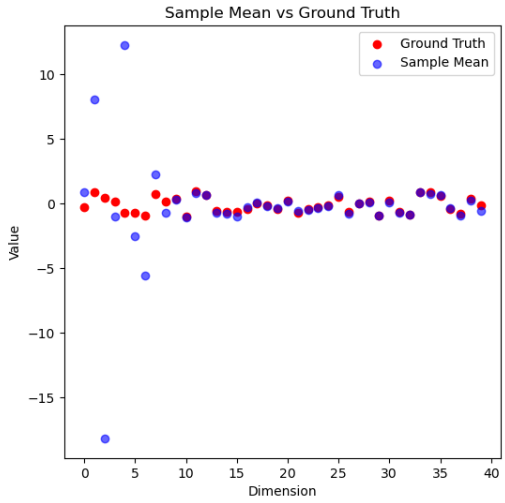
When using partial observations, the error does not seem to converge as the number of samples increases.

num-samples	150000	300000	3000000
SSE	172.800723	309.825094	4340.741020

I tried some observations in 38 or 39 dimensions and found that it seems that only the unobserved dimensions and nearby dimensions will produce huge errors, while most other dimensions are relatively accurate. For details, see the following two comparison charts of the sampling mean and true value of each dimension.



(A) Sampling Mean vs. True Value (38 Dimensions)



(B) Sampling Mean vs. True Value (39 Dimensions)

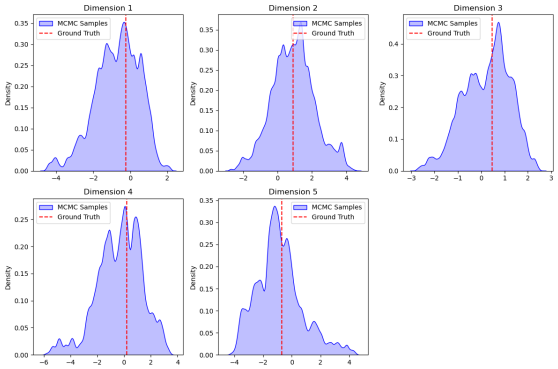
FIGURE 3. Parital observations result

2. Repeat the same with the ML Lorenz96 model

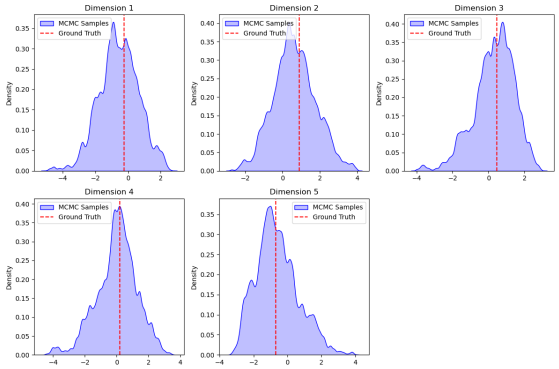
The script `mcmc_ml.py` has completed the task.

num-samples	20000	40000	80000	150000	3000000
SSE	26.808195	39.710715	44.904860	26.932605	5.450223

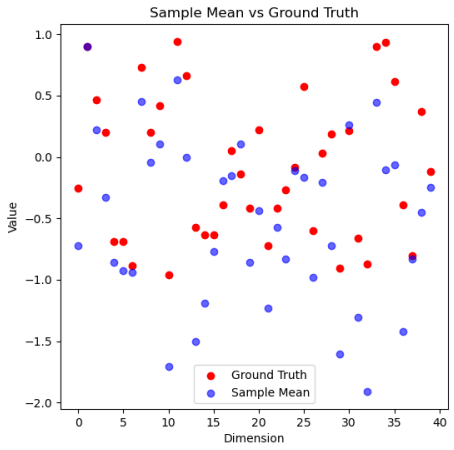
As can be seen from the table, although the average error of the MCMC sampling performed using the ML model is also converging, the accuracy is not high. More sampling times are required to achieve the same error accuracy. And the running speed is also slower.
The number of iterations of the drawn image is less than that in the previous section because the running time is too long.



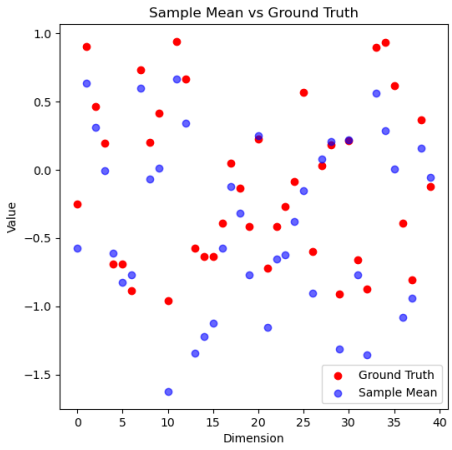
(A) First 5 Dimension Distribution (2M samples)



(B) First 5 Dimension Distribution (3M samples)



(C) Comparison of All Dimensions (2M samples)



(D) Comparison of All Dimensions (3M samples)

Figure 4. Full observations result of ML model MCMC

I also tried using mcmc of the ml model for partial observations and the results also seemed to not converge.

num-samples	150000	300000	3000000
SSE	172.800723	309.825094	4340.741020

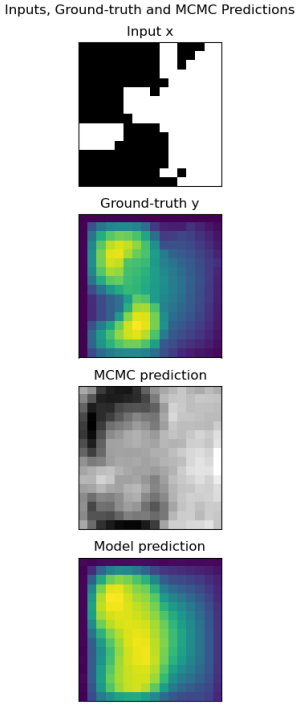
3. Solve the inverse problem of Darcy equation

The script `mcmc_dancy.py` has completed the task.
I found that the required true value at each point is often only 0 or 1, so I tried to perform mcmc sampling only in the value range of 0 and 1, and the effect did become better.

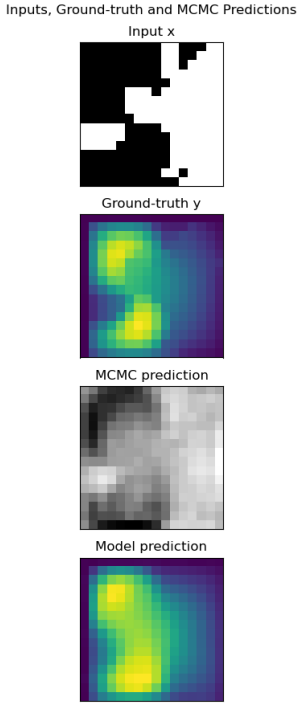
num-samples	100	1000	10000	100000	1000000
SSE	321.9372	280.0072	229.1814	186.9065	181.1333
SSE(01 range sample)			189.1669	146.8316	143.3735

Table 1. SSE Trends(in 32*32 grid)

The following figures illustrate the comparison results for different sampling strategies and sampling counts:
1. **Comparison of Two Sampling Strategies**: Figure 5 shows the results of two different sampling strategies for a 16*16 resolution with `num_samples` = 6,000,000. The results indicate that sampling within the 01 range improves performance.
2. **Effect of Sampling Count at 32*32 Resolution**: Figures 6 demonstrate the results for a 32*32 resolution with the 01 range sampling strategy. Different numbers of samples (`num_samples` = 10,000, 100,000, and 1,000,000) were used. The results show that increasing the number of samples leads to improved outcomes.

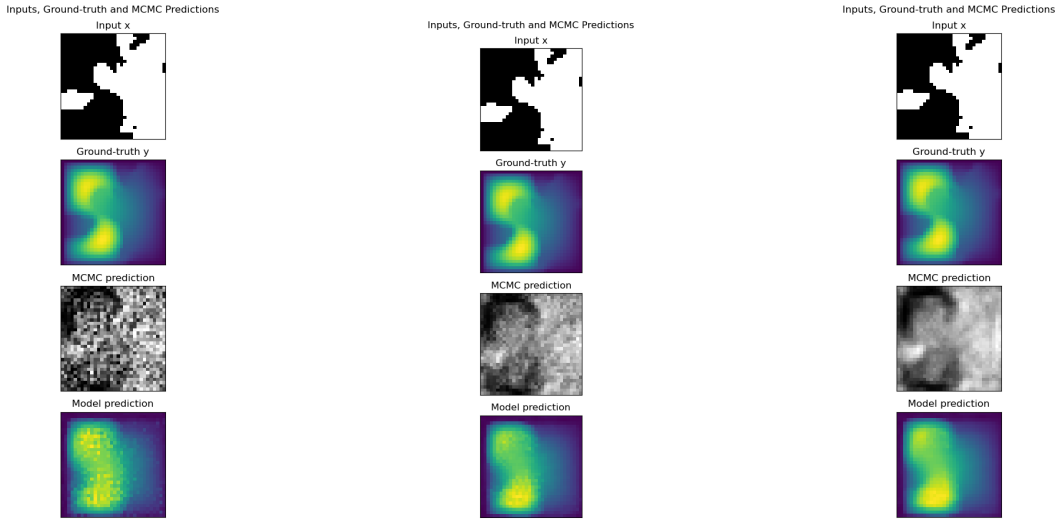


(a) 16*16 Resolution with Sampling Strategy A (`num_samples` = 6,000,000).



(b) 16×16 Resolution with Sampling Strategy B (`num_samples` = 6,000,000).

Figure 5. Comparison of Two Sampling Strategies at 16*16 Resolution.



(A) `num_samples` = 10,000. (B) `num_samples` = 100,000. (C) `num_samples` = 1,000,000.

FIGURE 6. Effect of Sampling Count at 32*32 Resolution with the 01 Range Sampling Strategy.