MCMC-PINNs: A modified Markov chain Monte-Carlo method for sampling collocation points of PINNs adaptively

Tengchao Yu [1], Heng Yong [2], Li Liu [2], Han wang [2], and Hua chen [2]

[1]Institute of Applied Physics and Computational Mathematics
[2]Affiliation not available

October 30, 2023

MCMC-PINNs: A modified Markov chain Monte-Carlo method for sampling collocation points of PINNs adaptively*

Tengchao Yu† Heng Yong† Li Liu† Han Wang‡
Hua Chen†

January 28, 2023

Abstract

In recent years, Physics-informed neural networks(PINNs) have became one of the most popular methods for solving partial differential equations(PDEs). PINNs have many advantages, such as not facing the curse of dimensionality, low data requirements, and easy inverse problem solving. Introducing collocation points and PDE loss is the key to the PINNs while the distribution of collocation points has a significant impact on the training efficiency of the PINNs. Uniformly distributed collocation points may not capture the different scale features of the solution. To address this problem, we propose an adaptive method of sampling collocation points with modified Markov chain Monte Carlo method(MCMC). The MCMC-based adaptive collocation point sampling method(MCMC-PINNs) is divided into two steps, constructing distribution of collocation points and sampling in it effctively. Using residual as an indicator for collocation points distribution has been validated and residual of PDE converge to 0 as training progrosses, so we choose a monotone increasing function of residual as the unnomalized probability distribution of the collocation points and use MCMC to sample the collocation points. MCMC-PINNs can improve accuracy of PINNs and we prove a error bound of it. Finally, several numerical examples are used to illustrate the performance of MCMC-PINNs.

Keywords: PINNs, MCMC, adaptive sampling.

*The work was supported by the National Key R&D Program of China (Grant No.2022YFA1004500), NSFC (Grant No.12201058), NSAF (Grant No.U2230208) and the Key Laboratory of Nuclear Data foundation(Grant No.JCKY2022201C155)

†Institute of Applied Physics and Computational Mathematics, Beijing 100094, China.

‡Corresponding Author, Institute of Applied Physics and Computational Mathematics, Beijing 100094, China

1 Introduction

Partial differential equations(PDEs) are an important model for describing laws and systems in the real world, and the study of numerical algorithms for PDEs has a very well-developed framework[16, 19]. Limited by the grid discretization, inevitably, the traditional numerical algorithms will face the curse of dimensionality, while it is difficult to apply to the inverse problems. With the improvement of computer performance and the increase of data, data-driven machine learning methods have been greatly developed[5, 17, 23, 26]. However, there are significant limitations in solving PDEs using data-driven methods due to the difficulty in obtaining data from complex PDEs. To solve the problems above, PINNs was proposed[18]. Combined with PDEs, PINNs could give 0-label data at any location in the computation domain by introducing colocation point. The 0-label data greatly expands the data sources for the model, which removes the limitation of solving PDEs by data-driven methods[8, 9]. In recent years, PINNs methods have recieved more and more attention and be widely used to solve forward and inverse problems for PDEs[3, 15, 12, 11].

Since the collocation points can be placed anywhere in the computation domain, we can choose as many collocation points as we want, theoretically. But the number of collocation points affects the efficiency of solving PINNs method, so how to choose collocation points affects the efficiency of the PINNs. The traditional PINNs method apply uniform distribution to sample the collocation points, which is obviously not an optimal choice and has been confirmed by many works. First of all when the computation domain is unbounded, it is not possible to distribute collocation points using uniform distribution. When there is a lack of knowledge about the structure of the solution, it is difficult to choose a bounded region and solve PDEs accurately only by the collocation points in it. On the other hand, for problems with bounded computation domain, the structure of the PDEs solutions may be very complex, with multi-scale features. It is difficult to capture the structural features of the solution at all scales with high efficiency by sampling collocation points uniformly. Too few collocation points will cause the model to ignore small-scale features of the solution[22, 13], and too many collocation points will be computationally wasteful for other regions. Referring to the idea of adaptive encryption in traditional numerical methods, the development of adaptive collocation points sampling methods is an effective exploration for improving the efficiency of the method[24, 14, 10, 7, 4, 20]. The earliest proposed adaptive collocation points sampling method is residual-based refinement method (RAR)[14]. RAR improves PINNs by collecting a large number of collocation point candidates and selecting those with the largest residuals to join the training set. But when the dimension is high or the small-scale feature regions of solution are small, it is difficult for the candidate points to fall into the regions with large residuals, which affects the efficiency of the algorithm. In [6], the idea of using p-th power of the absolute residuals as the unnormalized probability distribution has been proposd. Using Metropolis-Hastings method or self-normalized sampling method to sample collocation points is another highlight of this paper. But how to choose the number

of power of the the absolute residuals and the adaptivity iterations are still problems. In [21], instead of sampling the collocation points, the generated model is used to approximate the distribution of the residuals through an additional designed network. But the additional network will increase the complexity of the model and the training time. In [25], the authors provide a broader overview of methods for adaptive collocation point sampling, extending the coverage of the methods by adding a constant c to the unnormalized probability distribution.

Based on the discussions above, we propose an adaptive collocation point sampling method based on a modified MCMC[2] method for PINNs(MCMC-PINNs). In this paper, to achieve the purpose of sample more collocation points with larger absolute residuals, the absolute residuals are regarded as the kinetic energy at the locations, and the probability distribution of the collocation points can be constructed by canonical distribution based on the energy. In order to improve the efficiency of sampling method, we modify the standard MCMC method to sample the collocation points. With the MCMC-PINNs strategy, PINNs can achieve more accurate result. This work can give three contributions as below:

- Consider the absolute residuals from the energy perspective and construct the probability distribution of the collocation points using the canonical distribution based on the energy.
- Use the above probability distribution sample collocation points adaptively, give an adaptive stopping criterion corresponding to it and analysis the convergence of the method.
- Improve the MH method by a given anisotropy proposal distribution based on the computational region and a map from outside the computation domain to inside without changing the stationary distribution.

The remainder of the paper is organized as follows. In Section 2, we briefly introduce the knowledge of PINNs and MCMC. After that, we give the ooverall MCMC-PINNs algorithm. In Section 4, we present the convergence analysis of MCMC-PINNs. In Section 5, we illustrate the proformance of MCMC-PINNs by several numerical experimentes and we conclude this work in Section 6.

2 Preliminaries of PINNs and MCMC

For the proposed MCMC-PINNs method, there are two main parts of preliminary knowledges, which are PINNs and MCMC, and in this section we will introduce them briefly

2.1 Physics-Informed neural networks

Let $\Omega \in \mathbb{R}^d$ be the spatial domain and $[0, T]$ be the calculation time interval. A general PDE can be represented as:

$$\mathcal{N}(\boldsymbol{x}, t; u(\boldsymbol{x}, t)) = 0, \quad \boldsymbol{x} \in \Omega \quad t \in [0, T] \tag{2.1}$$

where $\mathcal{N}[\cdot]$ is a differential operator and $u(x,t)$ is the solution. If the PDE is time-independent, we can also use 2.1 to represent it with omitting t from it. Then the initial condition and boundary condition can be expressed as

$$\begin{cases} u(\boldsymbol{x},0) = u_0(\boldsymbol{x}), & \boldsymbol{x} \in \Omega \\ \mathcal{B}(\boldsymbol{x},t;u(\boldsymbol{x},t)) = 0, & \boldsymbol{x} \in \partial\Omega \quad t \in [0,T] \end{cases} \tag{2.2}$$

where $u_0 : \Omega \to \mathcal{R}$ is the initial operator and $\mathcal{B} : \mathcal{R}^{d+1} \to \mathcal{R}$ represents the boundary operator. To approximate the solution $u(\boldsymbol{x},t)$ with $\hat{u}(\boldsymbol{x},t)$, three errors can be introduced to measure the difference between $\hat{u}$ and u.

$$\varepsilon_{ini} = \int_{\Omega} |\hat{u}(\boldsymbol{x},0) - u_0(\boldsymbol{x})|^2 d\boldsymbol{x}, \qquad \varepsilon_{bou} = \int_{\partial\Omega\times[0,T]} |\mathcal{B}(\boldsymbol{x},t;\hat{u}(\boldsymbol{x},t))|^2 d\boldsymbol{x}dt,$$
$$\varepsilon_{PDE} = \int_{\Omega\times[0,T]} |\mathcal{N}(\boldsymbol{x},t,\hat{u}(\boldsymbol{x},t))|^2 d\boldsymbol{x}dt \tag{2.3}$$

represent the difference between $\hat{u}$ and u in terms of initial condition, boundary condition and PDE respectively. Obviously, if we can solve the PDE problem 2.1, 2.2 exactly, i.e. $\hat{u}(\boldsymbol{x},t) = u(\boldsymbol{x},t)$, then $\varepsilon_{ini},\ \varepsilon_{bou},\ and\ \varepsilon_{PDE}$ are exact 0, so a good approximation $\hat{u}(\boldsymbol{x},t)$ should minimize the generalization error:

$$\varepsilon(\hat{u}) = \varepsilon_{ini} + \varepsilon_{bou} + \varepsilon_{PDE} \tag{2.4}$$

For PINNs, deep neural network(DNN) is usd to approximate the solution $u(\boldsymbol{x},t)$. Assume the parameters of the DNN is θ, the approximation solution is $\hat{u}(\boldsymbol{x},t;\theta)$. Then PINNs is essentially designed to solve the following optimization problem:

$$\min_{\theta\in\Theta} \varepsilon(\hat{u}(\boldsymbol{x},t;\theta)) = \min_{\theta\in\Theta} \varepsilon_{ini} + \varepsilon_{bou} + \varepsilon_{PDE} \tag{2.5}$$

Because the generalization error cannot be optimized directly during training, three pointwise residuals are applied to approximate the generalization error of DNN.

$$\mathcal{L}_{ini}(\theta) = \frac{1}{N_{ini}} \sum_{i=1}^{N_{ini}} |\hat{u}(\boldsymbol{x}_i^{ini},0;\theta) - u_0(\boldsymbol{x}_i^{ini})|^2 \quad \mathcal{L}_{bou}(\theta) = \frac{1}{N_{bou}} \sum_{i=1}^{N_{bou}} |\mathcal{B}(\boldsymbol{x}_i^{bou},t_i^{bou};\hat{u}(\boldsymbol{x}_i^{bou},t_i^{bou};\theta))|^2$$
$$\mathcal{L}_{PDE}(\theta) = \frac{1}{N_{PDE}} \sum_{i=1}^{N_{PDE}} |\mathcal{N}(\boldsymbol{x}_i,t_i;\hat{u}(\boldsymbol{x}_i,t_i;\theta))|^2 \tag{2.6}$$

where $\{\boldsymbol{x}_i^{ini}\}_{i=1}^{N_{ini}} \in \Omega$, $\{(\boldsymbol{x}_i^{bou},t_i^{bou})\}_{i=1}^{N_{bou}} \in \partial\Omega\times[0,T]$, $(x_i,t_i) \in \Omega\times[0,T]$ are all uniformly distributed and $N_{ini},\ N_{bou},\ N_{PDE}$ are the numbers of points used to calculate the residuals of initial condition, boundary condition and PDE. Then the optimization problem 2.5 can be transformed into:

$$\min_{\theta\in\Theta} \mathcal{L}(\hat{u}(\boldsymbol{x},t;\theta)) = \min_{\theta\in\Theta} \mathcal{L}_{ini}(\theta) + \mathcal{L}_{bou}(\theta) + \mathcal{L}_{PDE}(\theta) \tag{2.7}$$

For the initial condition and Direchlet boundary conditions, the corresponding term of the loss function $\mathcal{L}(\hat{u})$ is the standard supervised learing loss, so the

initial and boundary training data should be label-data, i.e $\{(\boldsymbol{x}_i^{ini}, t_i^{ini}), u_i^{ini}\}_{i=1}^{N_{ini}}$ and $\{(\boldsymbol{x}_i^{bou}, t_i^{bou}), u_i^{bou}\}_{i=1}^{N_{bou}}$. For PDE and the other types of boundary condition, the location of the collocation points and boundary point is the only thing we should know. Moreover the PDE loss can be regarded as a kind of regularization.

2.2 Markov chain Monte Carlo method

Given an unnormalized probability distribution, there are difficulties in sampling from it, because the normalization constant is unknown, so Markov chain Monte Carlo method can be used. MCMC is an advanced sampling method that generates samples in the target distribution with the help of Markov chains. Consider an unnormalized probability distribution $\pi(\boldsymbol{x})$, whose state space is $\mathcal{X}$, the MCMC method aims to construct a Markov chain on $\mathcal{X}$ with $\pi(\boldsymbol{x})$ as its stationary distribution. A sufficient condition of it called reversibility condition, is to find a transition kernel $K(\boldsymbol{x}, \boldsymbol{y})$ satisfies that

$$\int_{\boldsymbol{x}\in\mathcal{X}} \pi(d\boldsymbol{x})K(\boldsymbol{x}, d\boldsymbol{y}) = \pi(d\boldsymbol{y}). \tag{2.8}$$

Detailed balance condition is an equivalence condition of the reversibility condition. It can be described as blow:

Definition 2.1. *A markov chain with transition kernel K satisfies the detailed balance condition if there exists a function $\pi(\boldsymbol{x})$ satisfying*

$$K(\boldsymbol{y}, \boldsymbol{x})\pi(\boldsymbol{y}) = K(\boldsymbol{x}, \boldsymbol{y})\pi(\boldsymbol{x}) \tag{2.9}$$

for every $(\boldsymbol{x}, \boldsymbol{y})$

Then if the Markov chain can be simulated for a long enough time, we can say that the state of the Markov chain is the samples from the stationary distribution.

Metropolis-Hasting method is one of the most widely used MCMC algorithm and it provides a simple method for constructing transition distributions that satisfy the reversibility condition. It splits the transition probability into two terms. One of them is proposal distribution which is a probability distribution for generating new state. The other one is acceptance probability, used to determine whether to accept the new state. Assuming that $q(\boldsymbol{y}|\boldsymbol{x})$ is the proposal distribution from $\boldsymbol{x}$ to $\boldsymbol{y}$, then it can be proved that

$$\alpha(\boldsymbol{x}, \boldsymbol{y}) = \min\{1, \frac{\pi(\boldsymbol{x})q(\boldsymbol{y}|\boldsymbol{x})}{\pi(\boldsymbol{y})q(\boldsymbol{x}|\boldsymbol{y})}\} \tag{2.10}$$

is the optimal acceptance probability satisfying the reversibility condition. The MH method is shown in 2.2.

Algorithm 1 Metropolis-Hasting algorithm

Require: Initial state $\boldsymbol{x}^0$, proposal distribution $q(\cdot|\boldsymbol{x})$, number of samples N.

1: **for** $i = 1 : N$ **do**
2: Sample u from a uniform distribution, $u \sim \mathcal{U}[0, 1]$.
3: Sample new state $\boldsymbol{y}$ from proposal distribution, $\boldsymbol{y} \sim q(\boldsymbol{y}|\boldsymbol{x}^{(i-1)})$.
4: Calculate the acceptance probability of the new state,

$$\alpha(\boldsymbol{x}^{(i-1)}, \boldsymbol{y}) = \min\{1, \frac{\pi(\boldsymbol{x}^{(i-1)})q(\boldsymbol{y}|\boldsymbol{x}^{(i-1)})}{\pi(\boldsymbol{y})q(\boldsymbol{x}^{(i-1)}|\boldsymbol{y})}\}$$

5: **if** $u < \alpha(\boldsymbol{x}^{(i-1)}, y)$ **then**
6: Accept the new state, $\boldsymbol{x}^{(i)} = \boldsymbol{y}$
7: Reject the new state, $\boldsymbol{x}^{(i)} = \boldsymbol{x}^{(i-1)}$
8: **end if**
9: **end for**

Ensure: Samples $\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, ..., \boldsymbol{x}^{(N)}\}$

3 PINNs with modified MCMC collocation point sampling method(MCMC-PINNs)

In this section, we will introduce the MCMC-PINNs method in detail. MCMC-PINNs updates the distribution of collocation points based on residuals at every adaptivity iterations and use modified MCMC algorithm to sample them, which can improve the result numerical accuracy and reduce the need of the number of collocation points. The MCMC-PINNs can be devided into two mainly steps.

- **Construct the probability distribution of collocation points.** The collocation points are related to $\mathcal{L}_{PDE}$ in loss function and ε_{PDE} in generalization error and updating the collocation points aims to minimize the PDE reisdual directly. So the distribution can be represented as:

$$\pi(\boldsymbol{x}, t) \propto p(\boldsymbol{x}, t) = F_p(r(\boldsymbol{x}, t)) \tag{3.1}$$

  where $F_p : \mathcal{R} \to \mathcal{R}$ and $r(\boldsymbol{x}_t) = |\mathcal{N}(\boldsymbol{x}, t; u(\boldsymbol{x}, t))|$ is the absolute residuals of PDE.

- **Sample collocation points from the probability distribution.** According to 3.1, only an unnormalized probability density function based on residuals can be obtained. Since the normalization constant is unknown, it is not possible to sample directly from it, so the MCMC method needs to be used.

In the rest of this section, We will specify the workflow of the above two steps of MCMC-PINNs

3.1 Construct the probability distribution of collocation points

To minimize the generalization error, the PDE error term of it should be minimized over $\Omega \times [0,T]$; at the same time, the PDE residual will significantly decrease in neibourhoods of the collocation points. For the above two reasons, more collocation points should be distributed in the area with larger residuals. As mentioned above, the unnormalized probability distribution has the following form

$$p(\boldsymbol{x},t) = F_p(r(\boldsymbol{x},t)) \tag{3.2}$$

It follows that F_p should be monotone increasing functions. Moreover, when the PDE error term of generalization is 0, the collocation points should be uniformly distributed.

For functions with lower bound $f(\boldsymbol{x})$, we can take it as the potential energy, the corresponding probability distribution can be constructed using the canonical distribution:

$$\pi_c(\boldsymbol{x}) = \frac{1}{Z}\exp(-\frac{f(\boldsymbol{x})}{T}). \tag{3.3}$$

where T is the temperature of the system and Z is the normalizing constant. It is obvious that $\pi(\boldsymbol{x})$ is a monotone decreasing function with respect to $f(\boldsymbol{x})$. However, we can construct a monotone increasing probability distribution with respect to $f(\boldsymbol{x})$ in the same way. Assume that the total energy of the system is E_{max} and treat $f(\boldsymbol{x})$ as the kinetic energy of the system. Then we have

$$F_p(\boldsymbol{x}) = \frac{1}{Z}\exp(\frac{f(\boldsymbol{x}) - E_{max}}{T}). \tag{3.4}$$

When the computation domain is a compact set, the absolute PDE residual $r(\boldsymbol{x},t)$ is bounded. Now, we can construct the distribution as

$$\pi(\boldsymbol{x},t) \propto p(\boldsymbol{x},t) = \exp(\frac{r(\boldsymbol{x},t) - E_{max}}{T}). \tag{3.5}$$

For each adaptivity iteration, we can take $E_{max} = \max\limits_{\boldsymbol{x},t\in\Omega\times[0,T]} r(\boldsymbol{x},t)$. When the generalization error is 0, $\pi(\boldsymbol{x},t) \propto \exp(\frac{r(\boldsymbol{x},t)-E_{max}}{T}) = 1$, which is the uniform distribution over the computation domain.

On the other hand, for any $0 < c < T$, the probability distribution mentioned above can be transformed into the following form,

$$\begin{aligned}
\pi(\boldsymbol{x},t) \propto p(\boldsymbol{x},t) &= \exp(\frac{r(\boldsymbol{x},t) - E_{max}}{T}) \\
&= \exp(\frac{-E_{max}}{T})\exp(\frac{r(\boldsymbol{x},t)}{T}) \\
&= \frac{1}{T}\exp(\frac{-E_{max}}{T})(T\exp(\frac{r(\boldsymbol{x},t)}{T}) - c) + \frac{c}{T}\exp(\frac{-E_{max}}{T})
\end{aligned} \tag{3.6}$$

Let Z be the normalization constant of 3.6, which can be splited into two terms $Z = Z_1 + Z_2$, where

$$Z_1 = \int_{\Omega\times[0,T]} T\exp(\frac{r(\boldsymbol{x},t)}{T}) - cd\boldsymbol{x}dt \qquad Z_2 = \int_{\Omega\times[0,T]} cd\boldsymbol{x}dt \tag{3.7}$$

The probability distribution $\pi(\boldsymbol{x},t)$ can represented as:

$$\begin{aligned} p(\boldsymbol{x},t) &= \frac{1}{T}\exp(\frac{-E_{max}}{T})(\frac{1}{Z}(T\exp\frac{r(\boldsymbol{x},t)}{T} - c) + c) \\ &= \frac{1}{T}\exp(\frac{-E_{max}}{T})(\frac{Z_1}{Z}\frac{T\exp\frac{r(\boldsymbol{x},t)}{T} - c}{Z_1} + \frac{Z_2}{Z}\frac{Tc}{Z_2}) \\ \pi(\boldsymbol{x},t) &= \frac{Z_1}{Z}\frac{T\exp\frac{r(\boldsymbol{x},t)}{T} - c}{Z_1} + \frac{Z_2}{Z}\frac{c}{Z_2}. \end{aligned} \tag{3.8}$$

It can be seen as a combination of two distributions, one of which is proportional to $T\exp(\frac{r(\boldsymbol{x},t)}{T}) - c$ and the other is a uniform distribution over the computation domain. For each adaptivity iteration, sample collocation points from $\pi(\boldsymbol{x},t)$ is equivalent to selecting $\frac{Z_1}{Z}$ of collocation points from the distribution proportional to $T\exp(\frac{r(\boldsymbol{x},t)}{T}) - c$ and $\frac{Z_2}{Z}$ collocation points uniformly distributed over the computation domain. This can also be seen as a strategy for adaptively adjusting the ratio of the two distributions, and c controls the maximum proportion of collocation points sampled from uniform distribution. The closer c is to T, the greater the maximum proportion of collocation points generated from uniform distribution and vice versa.

3.2 Sample collocation points from the probability distribution

As mentioned above, MCMC method is applied to sample collocation points from an unnormalized probability distribution. MH method is one of the most popular MCMC method, which is used in this paper. For the MH method, the choice of the proposal distribution significantly affects the efficiency of the method.

First of all, for many PDEs, the scales of various dimensions of the computation domain cannot keep consistent. The standard proposal distribution of MH method is an isotropic normal or uniform distribution. It is difficult to choose a good stepsize ϵ, and number of steps N_{MCMC}, such that each collocation point can efficiently explore the entire computation domain. So, in this case, changing the proposal distribution is an effective way to improve the efficiency of the MH method to explore the computation domain. For the general sampling problem, the adaptive MCMC[1] uses the covariance matrix of collocation points as the covariance matrix of the normal proposal distribution, so that all collocation points on a Markov chain can be distributed more often in the high probability region. But when we apply it to sample collocation points for PINNs, how to make all collocation points explore the whole computation domain efficiently is

the key problem. So, here we choose a diagonal matrix as the covariance matrix of the normal proposal distribution, and each element of the matrix is the square of the length of the computation domain in the corresponding dimension, i.e. $q(\boldsymbol{y}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}, \Sigma)$, $\Sigma = diag(\sigma_1, ..., \sigma_{d+1})$ and $\sigma_i = \Delta_i^2,\ i = 1, ..., d+1$, where Δ_i is the length of the computation domain in the i-th dimension.

On the other hand, the computation domain of PDE is often bounded, if the candidate colllocation point falls outside the computation domain, then it will be rejected. In general, the higher the acceptance probability, the higher the sampling efficiency for the same ϵ and the same N_{MCMC}. So how to reduce the cases where the collocation points are rejected because of falling outside the computation domain is another idea to improve the MH method.

To achieve this goal, we want to construct a function $g : \mathcal{R}^{d+1} \to \Omega \times [0, T]$ which maps the candidate collocation point oustside the computation domain into it to avoid too many rejections. As mentioned above, detailed balance condition 2.1 can ensure that the stationary distribution is the target distribution. Assume that the MH transition kernel of $\pi(\boldsymbol{x}, t)$ is:

$$K(\boldsymbol{x}, t; \cdot) = q(\cdot|\boldsymbol{x}, t)\alpha(\boldsymbol{x}, t; \cdot) \tag{3.9}$$

and we have:

$$\pi(\boldsymbol{x}, t)q(\boldsymbol{x}', t'|\boldsymbol{x}, t)\alpha(\boldsymbol{x}, t; \boldsymbol{x}', t') = \pi(\boldsymbol{x}', t')q(\boldsymbol{x}, t|\boldsymbol{x}', t')\alpha(\boldsymbol{x}', t'; \boldsymbol{x}, t) \tag{3.10}$$

for any $(\boldsymbol{x}', t') \in \Omega \times [0, T]$. $q(\boldsymbol{x}', t'|\boldsymbol{x}, t)$ is the proposal distribution and $\alpha(\boldsymbol{x}, t; \boldsymbol{x}', t')$ is the acceptance. When the proposal distribution is symmetric over $\mathcal{R}^{d+1}$, i.e. $q(\boldsymbol{x}', t'|\boldsymbol{x}, t) = q(\boldsymbol{x}, t|\boldsymbol{x}', t')$ for any $(\boldsymbol{x}, t), (\boldsymbol{x}', t') \in \mathcal{R}^{d+1}$, the acceptance probability has very simple form, $\alpha(\boldsymbol{x}, t; \boldsymbol{x}', t') = \min(1, \frac{\pi(\boldsymbol{x}', t')}{\pi(\boldsymbol{x}, t)})$, which is very helpful for sampling. So we want the new proposal distribution can ensure that the stationary distribution is $\pi(\boldsymbol{x}, t)$ without changing the acceptance probability.

Theorem 3.1. *Assume that $q(\cdot|\boldsymbol{x}, t)$ is a symmetric proposal distribution and the transition kernel $q(\cdot|\boldsymbol{x}, t)\alpha(\boldsymbol{x}, t; \cdot)$ satisfies the detailed balance condition 2.1 for $\pi(\boldsymbol{x}, t)$. For any function $g : \mathcal{R}^{d+1} \to \Omega \times [0, T]$ satisfying $g(\boldsymbol{x}_1, t_1) = (\boldsymbol{x}_1, t_1)$ inside the computation domain and any point $(\boldsymbol{x}', t') \in \Omega \times [0, T] \backslash \mathcal{R}^{d+1}$, if there exists a one-to-one correspondence of $(\boldsymbol{x}', t')$ with $(\boldsymbol{x}'', t'')$ satisfying the following conditions:*

1. *$g(\boldsymbol{x}'', t'') = (\boldsymbol{x}_1, t_1)$ and $g(\boldsymbol{x}', t') = (\boldsymbol{x}_2, t_2)$;*
2. *$q(\boldsymbol{x}', t'|\boldsymbol{x}_1, t_1) = q(\boldsymbol{x}'', t''|\boldsymbol{x}_2, t_2)$.*

Then the proposal distribution

$$q^*(\cdot|\boldsymbol{x}_1, t_1) = \int_{\mathcal{R}^{d+1}} q(\boldsymbol{x}', t'|\boldsymbol{x}_1, t_1)\mathbb{I}_{g(\boldsymbol{x}', t')}(\cdot)d\boldsymbol{x}'dt' \tag{3.11}$$

can avoid rejections caused by candidate collocation points falling outside the computation domain and ensure that the stationary distribution is $\pi(\boldsymbol{x}, t)$ without changing the acceptance probability. $\mathbb{I}_{g(\boldsymbol{x}', t')}(\boldsymbol{x}, t)$ is an indicator function

with respect to $(\boldsymbol{x}, t)$*:*

$$\mathbb{I}_{g(\boldsymbol{x}',t')}(\boldsymbol{x},t) = \begin{cases} 1 & (\boldsymbol{x},t) = g(\boldsymbol{x}',t') \\ 0 & otherwise \end{cases} \tag{3.12}$$

The details of the proof can be seen in Appendix.

Now we can propose the modified MH method as below, For two-dimensional problems, when the computation domain is a rectangle, g can be choosen as the bounce function with respect to all the boundaries:

$$g(x) = Ub - |\Delta - (\boldsymbol{x} - Lb)\%2\Delta| \tag{3.13}$$

where Ub is the upper bound of the computation domain and Lb is the lower bound of the computation domain. Finally, we can propose the MCMC-PINNs method 3.2

For MCMC-PINNs we should note four points below:

- For MCMC-PINNs, when $Max(r(\boldsymbol{x}, t)) > \rho_{max}$, we choose $r(\boldsymbol{x}, t)$ as the unnormalized distribution of collocation points. This is because when $r(\boldsymbol{x}, t)$ is very large, $\exp(r(\boldsymbol{x}, t))$ may not be computed and make it more difficult for collocation points to move away the local maxima of $r(\boldsymbol{x}, t)$.
- Because Z_1 and Z_2 are defined by integration, it may be difficult to calculate in practice, so we can choose $c = \rho_u Z_1$. Then the proportion of collocation points equals to $\frac{\rho_u}{1+\rho_u}$.
- In practice we can also take $\pi(\boldsymbol{x}, t) = r(\boldsymbol{x}, t)$ for simplicy, because if we take $T = c$, $T \exp \frac{r(\boldsymbol{x},t)}{T} - c$ is an equivalent infinitesimal quantity of $r(\boldsymbol{x}, t)$. As training proceeds, for $r(\boldsymbol{x}, t)$ closed to zero, they are similiar to each other.

4 Convergence Analysis

In this section, we will give the convergence analysis of MCMC-PINNs. To this end, two assumption need to be given. Generally, the initial condition can be considered as part of the boundary condition.

Assumption 4.1. *Consider the general PDE with the following form*

$$\begin{aligned} \mathcal{L}(\boldsymbol{x}; u(\boldsymbol{x})) &= 0, \qquad \boldsymbol{x} \in \Omega \\ \mathcal{B}(\boldsymbol{x}; u(\boldsymbol{x})) &= 0, \qquad \boldsymbol{x} \in \partial\Omega \end{aligned} \tag{4.1}$$

where $\mathcal{L}$ *is a linear operator and* $\mathcal{L} : \mathcal{H} \to \mathcal{H}$*, where* $\mathcal{H}$ *is a Hilbert space on* Ω*. Assume that there exists positive constants* C_1, C_2 *indepedent of* ν *satisfy:*

$$C_1||\nu||_{2,\Omega} \leq ||\mathcal{L}\nu||_{2,\Omega} + ||b\nu||_{2,\partial\Omega} \leq C_2||\nu||_{2,\Omega} \tag{4.2}$$

Algorithm 2 Computation domain based modified Metroplis-Hasting algorithm for PDE

Require: Initial points $\{(\boldsymbol{x}_j^{(0)}, t_j^{(0)})\}_{j=1}^{N_{PDE}}$, computation domain upper bound Ub, computation domain lower bound Lb, number of samples N, function g, stepsize ϵ, number of steps N_{mcmc}.

for $i = 1 : N$ **do**
  for $j = 1 : N_{mcmc}$ **do**
    Generate a number from the uniform distribution, $u \sim \mathcal{U}[0, 1]$.
    Sample new intermediate state $(\boldsymbol{x}', t')$ from proposal distribution, $(\boldsymbol{x}', t') \sim \mathcal{N}((\boldsymbol{x}_j^{(i-1)}, t_j^{(i-1)}), \Sigma)$, where

$$\Sigma = diag(\sigma_1^2, ..., \sigma_{d+1}^2),\ \Delta = Ub - Lb,\ \epsilon\sigma_i = \Delta_i,\ i = 1, ...d + 1.$$

    Map the intermediate state to the new proposal state,

$$(\boldsymbol{x}^*, t^*) = g(\boldsymbol{x}', t').$$

    Calculate the acceptance probability of the new state,

$$\alpha(\boldsymbol{x}_j^{(i-1)}, t_j^{(i-1)}; \boldsymbol{x}^*, t^*) = \min\{1, \frac{\pi(\boldsymbol{x}_j^{(i-1)}, t_j^{(i-1)})\mathcal{N}(\boldsymbol{x}^*, t^*; (\boldsymbol{x}_j^{(i-1)}, t_j^{(i-1)}), \Sigma)}{\pi(\boldsymbol{x}^*, t^*)\mathcal{N}(\boldsymbol{x}_j^{(i-1)}, t_j^{(i-1)}; (\boldsymbol{x}^*, t^*), \Sigma)}\}.$$

    if $u < \alpha(\boldsymbol{x}_j^{(i-1)}, t_j^{(i-1)}; \boldsymbol{x}^*, t^*)$ **then**
      Accept the new state, $(\boldsymbol{x}_j^{(i)}, t_j^{(i)}) = (\boldsymbol{x}^*, t^*)$;
    else
      Reject the new state, $(\boldsymbol{x}_j^{(i)}, t_j^{(i)}) = (\boldsymbol{x}_j^{(i-1)}, t_j^{(i-1)})$.
    end if
  end for
  Let $(\boldsymbol{x}^{(j)}, t^{(j)}) = (\boldsymbol{x}_j^{(N_{mcmc})}, t_j^{(N_{mcmc})})$
end for

Ensure: Samples $\{(\boldsymbol{x}^{(1)}, t^{(1)}), ..., (\boldsymbol{x}^{(N)}, t^{(N)})\}$

Algorithm 3 MCMC-PINNs

Require: Parameters of neural networks θ, initial condition training data $X_{ini} = \{(\boldsymbol{x}_i^{ini}, t_i^{ini})\}_{i=1}^{N_{ini}}$, boundary condition training data $X_{bou} = \{(\boldsymbol{x}_i^{bou}, t_i^{bou})\}_{i=1}^{N_{bou}}$, threshold for the mean of absolute PDE residuals, ρ, threshold for the maximum of absolute PDE residuals, ρ_{max}, maximum adaptivity iteration Max_{iter}, the temperature of the canonical distribution, T, c.

Sample the collocation points $X_{PDE} = \{(\boldsymbol{x}_i^{PDE}, t_i^{PDE})\}_{i=1}^{N_{PDE}}$ from a uniform distribution over the computation domain.
Apply algorithm 3.2 to update the collocation points $X_{PDE} = \{(\boldsymbol{x}_i^{PDE}, t_i^{PDE})\}_{i=1}^{N_{PDE}}$
while $Mean(r(\boldsymbol{x}, t)) > \rho$ and $N_{iter} < Max_{iter}$ **do**
    Train θ by minimizing the loss function with X_{ini}, X_{bou} and X_{PDE}.
    if $Max(r(\boldsymbol{x}, t)) > \rho_{max}$ **then**
        Let $\pi(\boldsymbol{x}, t) \propto r(\boldsymbol{x}, t)$;
    else
        Let $\pi(\boldsymbol{x}, t) \propto \exp(\frac{r(\boldsymbol{x},t)}{T})$.
    end if
    for $i = 1 : N_{PDE}$ **do**
        Apply algorithm 3.2 to generate new points $(\boldsymbol{x}_i^*, t_i^*)$ from $\pi(\boldsymbol{x}, t)$.
    end for
    Update collocation points $X_{PDE} = \{(\boldsymbol{x}_i^*, t_i^*)\}_{i=1}^{N_{PDE}}$
end while

Ensure: Parameters of neural network θ^*

The above condition is called the stability bound, which is essential to the existence and uniqueness of problem4.1.

Assumption 4.2. *The solution of neural network $\hat{u}(\boldsymbol{x};\theta)$ can be trained sufficiently to satisfy the condition, i.e. for any ϵ_b:*

$$||\mathcal{B}(u(\boldsymbol{x}) - \hat{u}(\boldsymbol{x};\theta))||_{2,\partial\Omega} \leq \epsilon_b \tag{4.3}$$

Theorem 4.1. *Assume the computation domain Ω is bounded, and let $\hat{u}(\boldsymbol{x};\theta)$ be the MCMC-PINNNs solution of 2.1. For a monotone increasing function $f(\boldsymbol{x})$, if we can find a positiv constant M such that $Mf(r(\boldsymbol{x})) > r(\boldsymbol{x})$. Take $\pi(\boldsymbol{x}) \propto Mf(r(\boldsymbol{x}))$ as the probability distribution of collocation points, and for any ϵ_r, if*

$$\mathbb{E}_{\pi(\boldsymbol{x})}[r(\boldsymbol{x})] < \epsilon, \tag{4.4}$$

we have

$$||u(\boldsymbol{x}) - \hat{u}(\boldsymbol{x},\theta)||_{2,\Omega} \leq \sqrt{2}C_1^{-1}(Mf(r_{max})S_\Omega\epsilon_r + \epsilon_b^2)^{1/2} \tag{4.5}$$

where

$$r_{max} = \max_{\boldsymbol{x}\in\Omega} |r(\boldsymbol{x})|, \tag{4.6}$$

S_Ω is the area of Ω.

Proof. Let $\nu(\boldsymbol{x}) = u(\boldsymbol{x}) - \hat{u}(\boldsymbol{x};\theta)$. According to Assumption 4.1, we have

$$\begin{aligned}
&||u(\boldsymbol{x}) - \hat{u}(\boldsymbol{x};\theta)||_{2,\Omega} \\
\leq& C_1^{-1}(||\mathcal{L}(\boldsymbol{x}, u(\boldsymbol{x}) - \hat{u}(\boldsymbol{x},\theta))||_{2,\Omega} + ||\mathcal{B}(u(\boldsymbol{x}) - \hat{u}(\boldsymbol{x};\theta))||_{2,\partial\Omega}) \\
=& C_1^{-1}(||r(\boldsymbol{x})||_{2,\Omega} + ||\mathcal{B}(u(\boldsymbol{x}) - \hat{u}(\boldsymbol{x};\theta))||_{2,\partial\Omega}) \\
\leq& \sqrt{2}C_1^{-1}(||r(\boldsymbol{x})||_{2,\Omega}^2 + ||\mathcal{B}(u(\boldsymbol{x}) - u(\boldsymbol{x};\theta))||_{2,\partial\Omega}^2)^{\frac{1}{2}}
\end{aligned} \tag{4.7}$$

From Assumption 4.2,

$$||u(\mathbf{x}) - u(\mathbf{x};\theta)||_{2,\Omega} \leq \sqrt{2}C_1^{-1}(||r(\mathbf{x};\theta)||_{2,\Omega}^2 + \epsilon_b^2)^{\frac{1}{2}} \tag{4.8}$$

Because the computation domain is bounded and neural network is a smooth function, there exists $r_{max} = \max_{\boldsymbol{x}\in\Omega} |r(\boldsymbol{x})|$. It follows that:

$$\int_\Omega f(r(\boldsymbol{x}))d\boldsymbol{x} \leq f(r_{max})S_\Omega. \tag{4.9}$$

Moreover,

$$\begin{aligned}
\mathbb{E}_{\pi(\boldsymbol{X})}[r(\boldsymbol{x})] &< \epsilon_r \\
\frac{\int_\Omega r(\boldsymbol{x})f(r(\boldsymbol{x}))dx}{\int_\Omega f(r(\boldsymbol{x}))dx} &< \epsilon_r \\
\int_\Omega r(\boldsymbol{x})Mf(r(\boldsymbol{x}))dx &< Mf(r_{max})S_\Omega\epsilon_r \\
\int_\Omega r(\boldsymbol{x})^2dx &< Mf(r_{max})S_\Omega\epsilon_r
\end{aligned} \tag{4.10}$$

Finnally we have

$$||u(\boldsymbol{X}) - u(\boldsymbol{X};\theta^*)||_{2,\Omega} \leq \sqrt{2}C_1^{-1}(Mf(r_{max})S_\Omega\epsilon + \epsilon_b^2)^{\frac{1}{2}}. \tag{4.11}$$

□

Clearly $f(x) = x$ satisfies the conditions in Theorem 4.1. Moreover, we can prove that the unnormalized canonical distribution $\exp(\frac{x}{T})$ satisfies the conditions. If we take $M = T$, we have $T\exp(\frac{r(\boldsymbol{x})}{T}) > r(\boldsymbol{x})$, when $r(\boldsymbol{x}) = 0$ and $\frac{dT\exp(\frac{r(\boldsymbol{x})}{T})}{d\boldsymbol{x}} = \exp(\frac{r(\boldsymbol{x})}{T})r'(\boldsymbol{x}) > r'(\boldsymbol{x})$, for any $\boldsymbol{x} > 0$, then we have $T\exp(\frac{r(\boldsymbol{x})}{T}) > r(\boldsymbol{x})$ for any $r(\boldsymbol{x}) > 0$. It follows that no matter we choose $\pi(\boldsymbol{x}) \propto r(\boldsymbol{x})$ or $\pi(\boldsymbol{x}) \propto \exp(\frac{r(\boldsymbol{x})}{T})$, if $\mathbb{E}_{\pi(\boldsymbol{x})}[r(x)] < \epsilon_r$, the error between $u(\boldsymbol{x})$ and $\hat{u}(\boldsymbol{x}, t)$ can be bounded, i.e. MCMC-PINNs convergences.

5 Numerical Examples

In this section, we use three time-independent and two time-dependent PDEs to illustrate the performance of MCMC-PINNs algorithm.

5.1 Experimental Setup

Firstly, we give the basic settings of the three methods used for comparison in this paper.

- PINNs: The baseline PINNs method, updating collocation points with uniform distribution at every iterations.
- MH&PINNs: The method proposed in [6], updating collocation points by baseline MH methods.
- MCMC-PINNs: The method we proposed in this paper, updating collocation points by 3.2

Unless special mentioned, we will use the parameters below. $N_{PDE} = 5000, N_{bou} = 5000, N_{ini} = 2000$. And the network we used is a fully connected network with 6 hidden layers, 32 neurons in each layer and tanh as the activition function. The main optimizer of network is ADAM with learning rate $lr = 1e-4$ and iteration $n_{epoch} = 10000$. The maximum we update the parameter of network is $M = 10$. Because most numerical examples we test is zeor in most region of the computation domain, so we choose the mean square error(MSE) to illustrate the performance of each method.

$$MSE = \frac{1}{N_t}\sum_{i=1}^{N_t}(u(\boldsymbol{x}, t) - \hat{u}((\boldsymbol{x}, t);\theta))^2 \tag{5.1}$$

where N_t is the number of test data.

For MCMC we often choose stepsize $\epsilon = 0.1$, the number of steps $n_{MCMC} = 400$ and the proposal distribution as multidimensional independent distribution. For rectangle computation domain we take the rebounce function 3.13 as g for MCMC-PINNs. Furthermore, we$\rho_{max} = 1$, $Z_2 = 0.15Z_1$ and $T = c$.

5.2 One peak problem of two-dimensional elliptic equation

Consider the following two-dimensional elliptic equation:

$$\begin{aligned} -\Delta u(x,y) &= f(x,y) \qquad (\boldsymbol{x},y) \in \Omega \\ u(x,y) &= B(\boldsymbol{x},y) \quad (x,y) \in \partial\Omega \end{aligned} \tag{5.2}$$

where the coputation domain $\Omega = [-5,5]\times[-1,1]$, and the reference solution is

$$u(x,y) = \exp(-1000[(x-o_x)^2 + (y-o_y)^2]) \tag{5.3}$$

Let $o_x = 0.5$, $o_y = 0.5$, $f(x,y)$ and $B(x,y)$ be the right terms of PDE and boundary condition of 5.2 corresponding to the exact solution 5.3 respectively. The exact solution is closed to zero in most region of Ω. But, in a small region, the solution has a peak and large norm of gradient. So 5.3 can be regarde as a multi-scale solution. According to the norm of gradient, the neighborhood of (o_x, o_y), $O(o_x, o_y)$ can be considered as the small-scale reigion of solution. We expect to sample more collocation points in the small scale region, which is helpful for characterizing the small-scale characteristics of the solution.

For this problem the length of computation domain is 10 and 2, respectively. So the covariance matrix of the proposal distribution is choosen as $\Sigma = \mathrm{diag}(25, 1)$. Finally, 256×256 uniform meshgrid points over the computation domain are choosen as the test points, and use the mean square error(MSE) on the test points as the index to measure the accuracy of algorithms.

From Fig.1 we can see that PINNs cannot find the peak of the solution, because the small-scale region is too small to use collocation points from a uniform distribution to capture the small-scale characteristics. MH-PINNs and MCMC-PINNs both can capture the peak of the solution. But it is obvious that MCMC-PINNs can obtain a more accuracy prediction than MH-PINNs. And the mean test error of MCMC-PINNs is $\frac{2}{3}$ of the MH-PINNs after training. At the same time, during training, the test error of MCMC-PINNs is always smaller than MH-PINNs.

In Fig.2, we illustrate the residual contour and collocation points of MH-PINNs and MCMC-PINNs respectively. At the first adaptivity iteration, MCMC-PINNs can move more collocation points to the small-scale region. The reason is that the MCMC-PINNs gives the diagonal elements of the covariance matrix of the proposal distribution different values depending on the length of the interval. The anisotropic proposal distribution allows the collocation points to move farther in the x_1-direction. And as the training progrossece, MCMC-PINNs can reduce the residual over the computation domain faster than MH-PINNs.

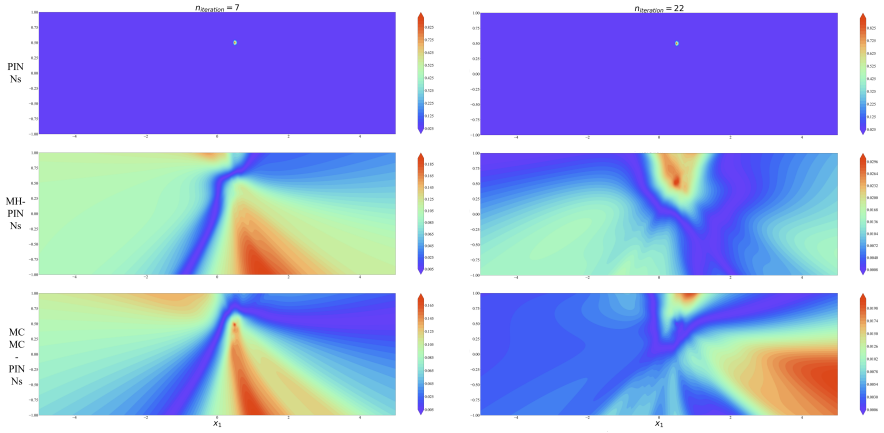

Figure 1: The distribution of collocation points obtained by two methods at three different iterations

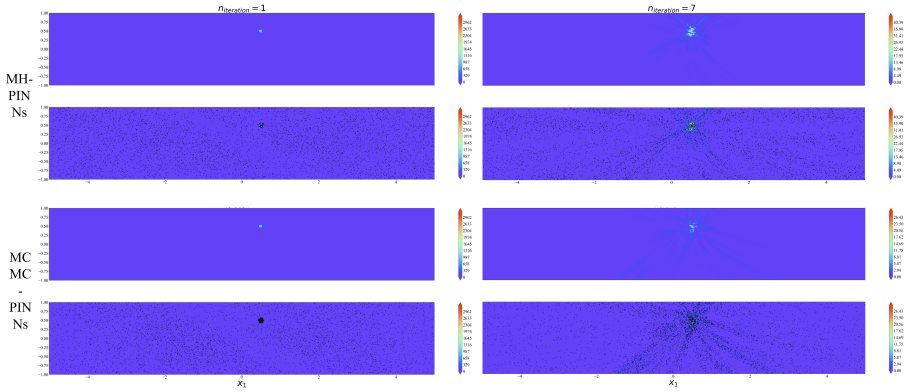

Figure 2: The abosute error and the predicted curves at $t = 1$obtained two methods

5.3 Two peak problem of two-dimensional elliptic equation

Consider the following two-dimensional elliptic equation:

$$\begin{aligned} \nabla[u(x,y)\nabla(x^2+y^2)] + \nabla^2 u(x,y) &= f(x,y), \quad (x,y) \in \Omega \\ u(x,y) &= B(x,u) \quad (x,y) \in \partial\Omega, \end{aligned} \tag{5.4}$$

where the computation domain is

$$\Omega = [-5,5] \times [-1,1]$$

, the reference solution is

$$u(x,y) = \exp(-1000[(x-o_{x,1})^2+(y-o_{y,1})^2]) + \exp(-1000[(x-o_{x,2})^2+(y-o_{y,2})^2]) \tag{5.5}$$

Let $o_{x,1} = 0.5$, $o_{y,1} = 0.5, o_{x,2} = -0.5$, $o_{y,2} = -0.5$, $f(x,y)$ and $B(x,y)$ be the right terms of PDE and boundary condition of 5.4 corresponding to the exact solution 5.5. Different from 5.3, 5.5 has two peaks far away from the others, it is more difficult to capture the two peaks of the solution, simultaneously. Same as the numerical example 5.2, the covariance matraix is $\mathrm{diag}(25, 1)$ and we use the network with 64 neurons in each hidden layer. 256×256 uniform meshgrid points over the computation domainare choosen as the test points.

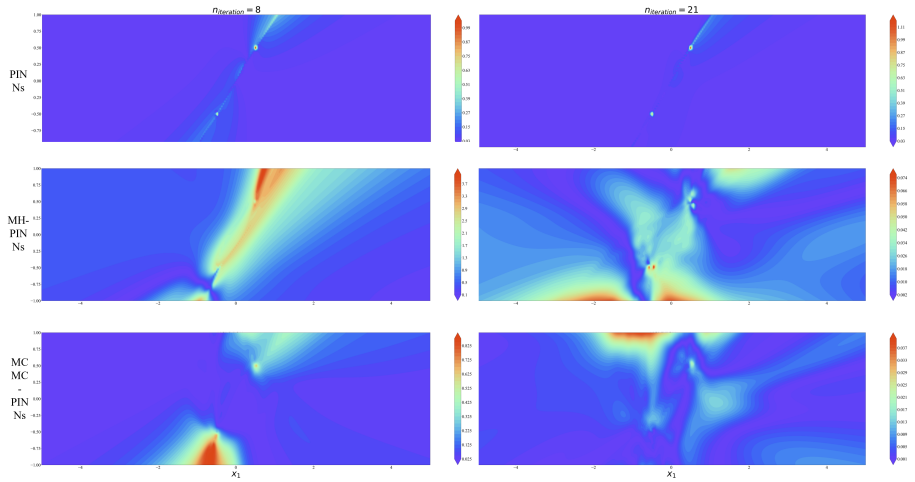

Figure 3: The distribution of collocation points obtained by two methods at three different iterations

Same as numerical exapmle 5.2, the Fig.3 shows the testerror contoure of PINNs, MH-PINNs and MCMC-PINNs at the 7-th adaptivity iteration and after training. We can see that the difference between the testerror of MCMC-PINNs and MH-PINNs is much greater than numerical example 5.2. MCMC-PINNs

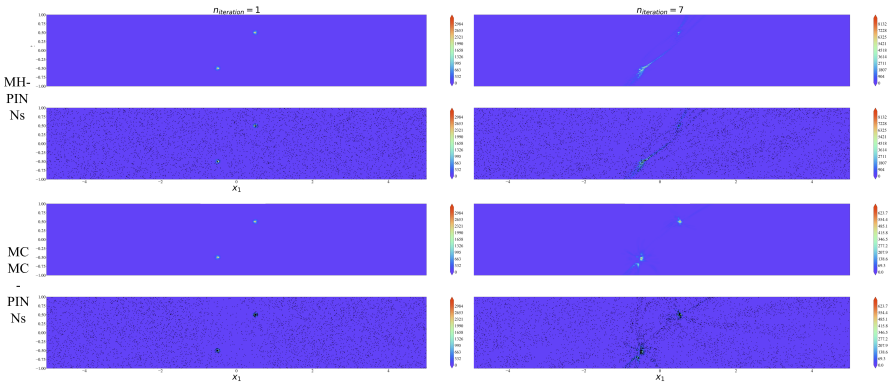

Figure 4: The abosute error and the predicted curves at $t = 1$obtained two methods

can achieve a prediction whose test error is half of MH-PINNs. During the training, the difference maybe even greater.

In Fig.4, same as Fig.2, the number of collocation points of MCMC-PINNs distributed in the small-scale region is much larger than MH-PINNs. Moreover, the residuals decreases much faster in the computation domain for MCMC-PINNs than for MH-PINNs.

5.4 Two-dimensional unbounded problem of elliptic equation

Consider the following two-dimensional elliptic equation

$$\begin{aligned} \Delta u(x,y) &= f(x,y), \qquad (x,y) \in \mathcal{R}^2 \backslash \Omega \\ u(x,y) &= B(x,u) \qquad (x,y) \in \partial\Omega, \end{aligned} \tag{5.6}$$

where the true solution is

$$u(x,y) = exp(-(x-4)^2 - (y-4)^2) \tag{5.7}$$

The boundary of Ω is defined as

$$\partial\Omega = (\cos(t) - \frac{\cos(5t)\cos(t)}{4}, \sin(t) - \frac{\sin(5t)\sin(t)}{4}) \tag{5.8}$$

We want to solve this problem in a rectangular computation domain. For any $(x,y) \in \mathcal{R}^2 \backslash \Omega$, We can use the following function $h : \mathcal{R}^2 \backslash \Omega \to [0, 2\pi] \times [0, +\infty)$ and its inverse to map (x,y) to polar coordinates or to map (t,r) back to the

rectangular coordinate system:

$$
\begin{cases}
t = \dfrac{\pi}{2} - sign(x)(\dfrac{\pi}{2} - \arctan(\dfrac{y}{x})) \\
r = \sqrt{x^2 + y^2 - (\cos(t) - \dfrac{\cos(5t)\cos(t)}{4})^2 - (\sin(t) - \dfrac{\sin(5t)\sin(t)}{4})^2}
\end{cases}
$$

$$
\begin{cases}
x = (r + \sqrt{(\cos(t) - \dfrac{\cos(5t)\cos(t)}{4})^2 + (\sin(t) - \dfrac{\sin(5t)\sin(t)}{4})^2})\cos t \\
y = (r + \sqrt{(\cos(t) - \dfrac{\cos(5t)\cos(t)}{4})^2 + (\sin(t) - \dfrac{\sin(5t)\sin(t)}{4})^2})\sin t
\end{cases}
\tag{5.9}
$$

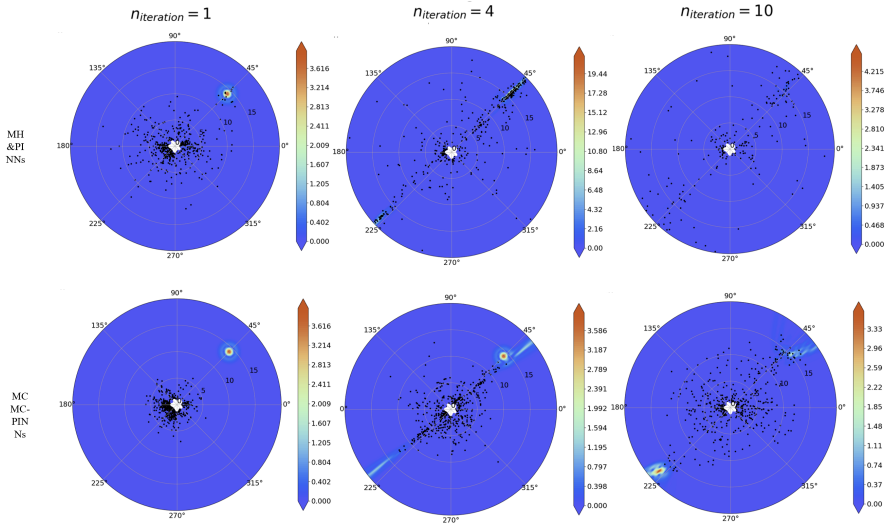

Figure 5: The distribution of collocation points obtained by two methods at three different iterations

In the polar coordinate, $g : \mathcal{R}^2 \to [0, 2\pi] \times [0, +\infty)$ can be:

$$
g(x, y) = \begin{cases}
(x, y) & (x, y) \in [0, 2\pi] \times [0, +\infty) \\
g(x, -y) & y < 0 \\
g(x\%2\pi, y) & x \notin [0, 2\pi]
\end{cases}
\tag{5.10}
$$

Fig.5 shows the residuals and collocation points sampled by two different methods in the same adaptivity iteration. Clearly, at the begining of adaptivity iteration, neither method can explore the peak of the solution due to the distance between the peak and the location of the initial collocation points. And MCMC-PINNs can distribute collocation points in a larger region and explore the calculation domain more efficiently. At the forth iteration, MCMC-PINNs has

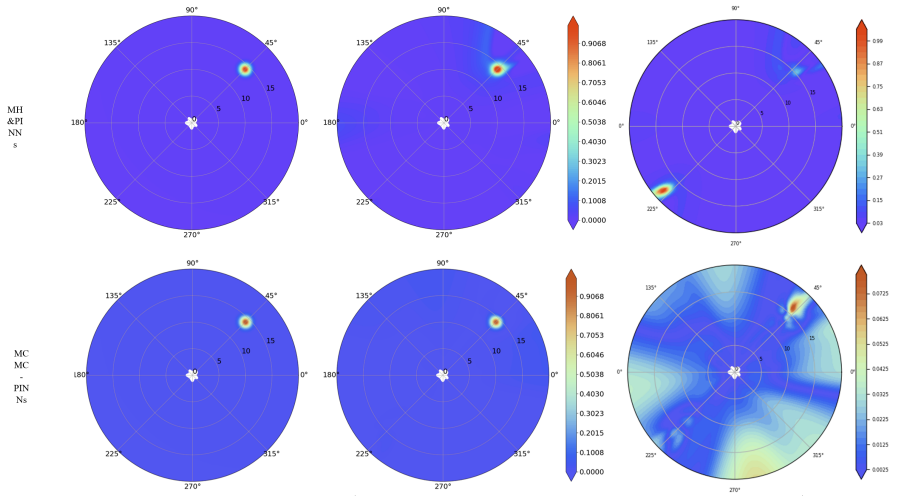

Figure 6: The distribution of collocation points obtained by two methods at three different iterations

already found the peak of the solution and apparently more collocation points are placed near the peak. Larger residuals attract more collocation points to the small-scale feature region of the solution. Eventually The collocation points of MCMC-PINNs tends to be uniformly distributed.

From Fig.6, it is obvious that the prediction error of $MCMC-PINNs$ is much smaller than the $MH\&PINNs$. In this numerical example, we have no knowledge about the location of the peak of the solution and the initial collocation points are far away from the peak, using $MCMC-PINNs$ can explore the computation domain with high speed as training progrosses without any human intervension. Finally we can get a better result than $MH\&PINNs$

5.5 Unbounde domain problem for a time-dependent PDE

The last numerical example we take a time-dependent PDE with unbounded domain into consideration:

$$\begin{aligned} \frac{\partial u(x,t)}{\partial t} &= \frac{\partial^2 u(x,t)}{\partial x^2} + f(x,t) \qquad (x,t) \in \mathcal{R} \times [0,1] \\ u(x,0) &= u_0(x) \qquad\qquad\qquad\qquad\quad x \in \mathcal{R} \end{aligned} \tag{5.11}$$

The exact solution of this problem is

$$u(x,t) = \frac{\exp -\frac{(x-10)^2}{4t+4}}{\sqrt{t+1}} \tag{5.12}$$

For this problem, the initial collocation points are uniformly generated in $[-6,0]\times[0,1]$, so the covariance matrix of $MCMC-PINNs$ is set to be $\mathrm{diag}(36,1)$.

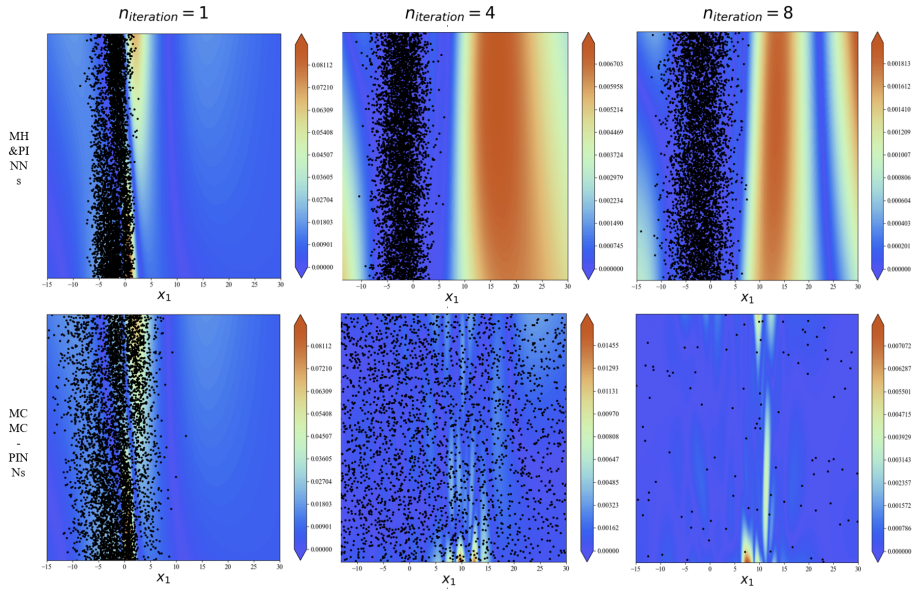

Figure 7: The distribution of collocation points obtained by two methods at three different iterations

And g is

$$g(x,t) = \begin{cases} (x,t) & t \in [0,1] \\ (x, Ub_t - |\Delta_t - (t - Lb_t)\%2\Delta_t|) & otherwise \end{cases} \tag{5.13}$$

Form Fig.7, we can see that for $MH\&PINNs$ with standard Gaussian proposal distribution, the collocation points cannot explore the computation domain effectivetly. They are more likely to be trapped in a localized region and cannot be distributed with $\pi(\boldsymbol{x}, t)$. On the other hand, $MCMC-PINNs$ with the modified proposal distribution can help collocation points fully explore the computational domain along the x-direction

Form Fig.8, we can also notice that the prediction solution of $MH\&PINNs$ cannot not capture the characteristics of the exact solution well, the maximum prediction error is nearly 1. While the prediction solution of $MCMC-PINNs$ has a relative small prediction error, the maximum prediction error is less than $1e-2$, and the distribution of collocation points tends to uniform distribution as training progrosses.

6 Conclution

In this paper, we present a modified Markov chain Monte Carlo method for sampling collocation points of PINNs adaptively. The main idea of MCMC-PINNs is regarding the absolute residual error as kinetic energy of system and

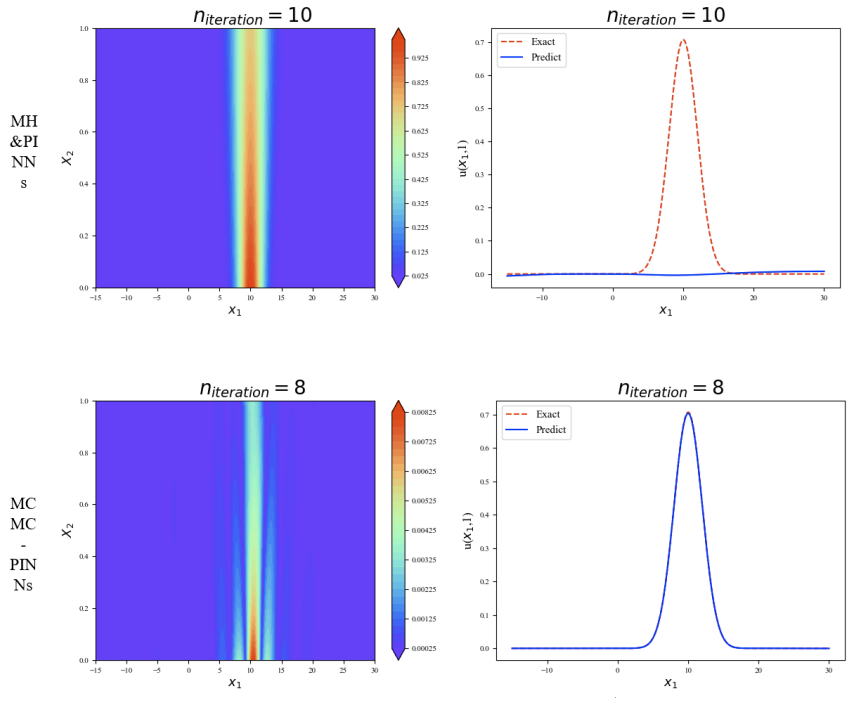

Figure 8: The abosute error and the predicted curves at $t = 1$obtained two methods

construct the canonical distribution of the potential energy as the probability distribution of collocation points. According to this distribution, we give a convergence analysis of MCMC-PINNs and modify the standard MCMC method by the length of each dimension in the computation domain and a function map points outside the computation domain to inside. We want to apply MCMC-PINNs to more complex problem ,especially complex time-dependent problem. To achieve this goal, we will study how to use collocation points describe the time evolution of PDEs. Also, we want to study constructing probability distribution with more indicators to make samples generated from it minimizing the generalization error more significantly. Work on these problems will be in our forthcoming papers.

References

[1] Christophe Andrieu and Johannes Thoms. "A tutorial on adaptive MCMC". In: *Statistics and computing* 18.4 (2008), pp. 343–373.

[2] Steve Brooks et al. *Handbook of markov chain monte carlo*. CRC press, 2011.

[3] Shengze Cai et al. "Physics-informed neural networks (PINNs) for fluid mechanics: A review". In: *Acta Mechanica Sinica* (2022), pp. 1–12.

[4] Arka Daw et al. "Rethinking the importance of sampling in physics-informed neural networks". In: *arXiv preprint arXiv:2207.02338* (2022).

[5] Xiaodong Feng, Li Zeng, and Tao Zhou. "Solving time dependent Fokker-Planck equations via temporal normalizing flow". In: *arXiv preprint arXiv:2112.14012* (2021).

[6] Wenhan Gao and Chunmei Wang. "Active Learning Based Sampling for High-Dimensional Nonlinear Partial Differential Equations". In: *arXiv preprint arXiv:2112.13988* (2021).

[7] Zhiwei Gao, Liang Yan, and Tao Zhou. "Failure-informed adaptive sampling for PINNs". In: *arXiv preprint arXiv:2210.00279* (2022).

[8] Ling Guo, Hao Wu, and Tao Zhou. "Normalizing field flows: Solving forward and inverse stochastic differential equations using physics-informed flow models". In: *Journal of Computational Physics* 461 (2022), p. 111202.

[9] Ling Guo et al. "Monte Carlo fPINNs: Deep learning method for forward and inverse problems involving high dimensional fractional partial differential equations". In: *Computer Methods in Applied Mechanics and Engineering* 400 (2022), p. 115523.

[10] Katsiaryna Haitsiukevich and Alexander Ilin. "Improved Training of Physics-Informed Neural Networks with Model Ensembles". In: *arXiv preprint arXiv:2204.05108* (2022).

[11] Ameya D Jagtap and George E Karniadakis. "Extended Physics-informed Neural Networks (XPINNs): A Generalized Space-Time Domain Decomposition based Deep Learning Framework for Nonlinear Partial Differential Equations." In: *AAAI Spring Symposium: MLPS*. 2021.

[12] Ameya D Jagtap, Ehsan Kharazmi, and George Em Karniadakis. "Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems". In: *Computer Methods in Applied Mechanics and Engineering* 365 (2020), p. 113028.

[13] Aditi Krishnapriyan et al. "Characterizing possible failure modes in physics-informed neural networks". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 26548–26560.

[14] Lu Lu et al. "DeepXDE: A deep learning library for solving differential equations". In: *SIAM Review* 63.1 (2021), pp. 208–228.

[15] Zhiping Mao, Ameya D Jagtap, and George Em Karniadakis. "Physics-informed neural networks for high-speed flows". In: *Computer Methods in Applied Mechanics and Engineering* 360 (2020), p. 112789.

[16] Nicholas Perrone and Robert Kao. "A general finite difference method for arbitrary meshes". In: *Computers & Structures* 5.1 (1975), pp. 45–57.

[17] Maziar Raissi and George Em Karniadakis. "Hidden physics models: Machine learning of nonlinear partial differential equations". In: *Journal of Computational Physics* 357 (2018), pp. 125–141.

[18] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational physics* 378 (2019), pp. 686–707.

[19] Junuthula Narasimha Reddy. *Introduction to the finite element method*. McGraw-Hill Education, 2019.

[20] Shashank Subramanian et al. "Adaptive Self-supervision Algorithms for Physics-informed Neural Networks". In: *arXiv preprint arXiv:2207.04084* (2022).

[21] Kejun Tang, Xiaoliang Wan, and Chao Yang. "DAS: A deep adaptive sampling method for solving partial differential equations". In: *arXiv preprint arXiv:2112.14038* (2021).

[22] Sifan Wang, Xinling Yu, and Paris Perdikaris. "When and why PINNs fail to train: A neural tangent kernel perspective". In: *Journal of Computational Physics* 449 (2022), p. 110768.

[23] E Weinan. "Machine learning and computational mathematics". In: *arXiv preprint arXiv:2009.14596* (2020).

[24] Colby L Wight and Jia Zhao. "Solving allen-cahn and cahn-hilliard equations using the adaptive physics informed neural networks". In: *arXiv preprint arXiv:2007.04542* (2020).

[25] Chenxi Wu et al. “A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks”. In: *Computer Methods in Applied Mechanics and Engineering* 403 (2023), p. 115671.

[26] Bing Yu et al. “The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems”. In: *Communications in Mathematics and Statistics* 6.1 (2018), pp. 1–12.

A The proof of Theorem 3.1

Proof. Consider 3.11,

$$\begin{aligned}
&\int_{\Omega\times[0,T]}\int_{\mathcal{R}^{d+1}} q(\boldsymbol{x}',t'|\boldsymbol{x}_1,t_1)\mathbb{I}_{g(\boldsymbol{x}',t')}(\boldsymbol{x}'',t'')d\boldsymbol{x}'dt'd\boldsymbol{x}''dt'' \\
=&\int_{\mathcal{R}^{d+1}} q(\boldsymbol{x}',t'|\boldsymbol{x}_1,t_1)\int_{\Omega\times[0,T]}\mathbb{I}_{g(\boldsymbol{x}',t')}(\boldsymbol{x}'',t'')d\boldsymbol{x}''dt''d\boldsymbol{x}'dt' \\
=&\int_{\mathcal{R}^{d+1}} q(\boldsymbol{x}',t'|\boldsymbol{x}_1,t_1)d\boldsymbol{x}'dt' \\
=&1,
\end{aligned} \tag{A.1}$$

where the second equation follows from the fact that the range of g is $\Omega\times[0,T]$. Also, it is obvious instead of rejecting the candidate collocation points outside the computation domain directly, the function g maps them into the computation domain and decide whether accept them or not.

Since $q(\cdot|\boldsymbol{x},t)$ is a symmetric proposal distribution and the transition kernel $q(\cdot|\boldsymbol{x},t)\alpha(\boldsymbol{x},t;\cdot)$ satisfies the detailed balance condition for $\pi(\boldsymbol{x},t)$, we have $\alpha(\boldsymbol{x},t;\boldsymbol{x}',t')=\min(1,\frac{\pi(\boldsymbol{x}',t')}{\pi(\boldsymbol{x},t)})$. Now if we want to prove that the proposal distribution $q^*(\cdot|\boldsymbol{x}_1,t_1)$ guarantees that $\pi(\boldsymbol{x},t)$ is still the stationary distribution of the Markov chains keeping the acceptance probabilities invariant we only need prove the transition kernel $\int_{\mathcal{R}^{d+1}} q(\boldsymbol{x}',t'|\boldsymbol{x}_1,t_1)\mathbb{I}_{g(\boldsymbol{x}',t')}(\cdot)d\boldsymbol{x}'dt'\min(1,\frac{\pi(\cdot)}{\pi(\boldsymbol{x}_1,t_1)})$ holds detailed balance condition. It is obvious that

$$\begin{aligned}
&\pi(\boldsymbol{x}_1,t_1)\int_{\mathcal{R}^{d+1}} q(\boldsymbol{x}',t'|\boldsymbol{x}_1,t_1)\mathbb{I}_{g(\boldsymbol{x}',t')}(\boldsymbol{x}_2,t_2)d\boldsymbol{x}'dt'\min(1,\frac{\pi(\boldsymbol{x}_2,t_2)}{\pi(\boldsymbol{x}_1,t_1)}) \\
=&\pi(\boldsymbol{x}_1,t_1)\int_{\Omega\times[0,T]} q(\boldsymbol{x}',t'|\boldsymbol{x}_1,t_1)\mathbb{I}_{g(\boldsymbol{x}',t')}(\boldsymbol{x}_2,t_2)d\boldsymbol{x}'dt'\min(1,\frac{\pi(\boldsymbol{x}_2,t_2)}{\pi(\boldsymbol{x}_1,t_1)}) \\
&+\pi(\boldsymbol{x}_1,t_1)\int_{\Omega\times[0,T]\setminus\mathcal{R}^{d+1}} q(\boldsymbol{x}',t'|\boldsymbol{x}_1,t_1)\mathbb{I}_{g(\boldsymbol{x}',t')}(\boldsymbol{x}_2,t_2)d\boldsymbol{x}'dt'\min(1,\frac{\pi(\boldsymbol{x}_2,t_2)}{\pi(\boldsymbol{x}_1,t_1)}).
\end{aligned} \tag{A.2}$$

Since $g(\boldsymbol{x}_1,t_1)=(\boldsymbol{x}_1,t_1)$, for any $(\boldsymbol{x}_1,t_1)\in\Omega\times[0,T]$, then we have

$$\begin{aligned}
&\pi(\boldsymbol{x}_1,t_1)\int_{\Omega\times[0,T]} q(\boldsymbol{x}',t'|\boldsymbol{x}_1,t_1)\mathbb{I}_{g(\boldsymbol{x}',t')}(\boldsymbol{x}_2,t_2)d\boldsymbol{x}'dt'\min(1,\frac{\pi(\boldsymbol{x}_2,t_2)}{\pi(\boldsymbol{x}_1,t_1)}) \\
=&\pi(\boldsymbol{x}_1,t_1)q(\boldsymbol{x}_2,t_2|\boldsymbol{x}_1,t_1)\min(1,\frac{\pi(\boldsymbol{x}_2,t_2)}{\pi(\boldsymbol{x}_1,t_1)}) \\
=&\pi(\boldsymbol{x}_2,t_2)q(\boldsymbol{x}_1,t_1|\boldsymbol{x}_2,t_2)\min(1,\frac{\pi(\boldsymbol{x}_1,t_1)}{\pi(\boldsymbol{x}_2,t_2)}) \\
=&\pi(\boldsymbol{x}_2,t_2)\int_{\Omega\times[0,T]} q(\boldsymbol{x}',t'|\boldsymbol{x}_2,t_2)\mathbb{I}_{g(\boldsymbol{x}',t')}(\boldsymbol{x}_1,t_1)d\boldsymbol{x}'dt'\min(1,\frac{\pi(\boldsymbol{x}_1,t_1)}{\pi(\boldsymbol{x}_2,t_2)})
\end{aligned} \tag{A.3}$$

According to the condition 1 and 2, for any given $(\boldsymbol{x}_2,t_2)\in\mathcal{R}^d\times[0,T]$ and any candidate collocate point $(\boldsymbol{x}',t')$ outside the computation domain generated by q, there exists only one point $(\boldsymbol{x}'',t'')$ outside the computation domain

corresponding to $(\boldsymbol{x}', t')$ satisfies:

$$\mathbb{I}_{g(\boldsymbol{x}',t')}(\boldsymbol{x}_2, t_2) = 1 \quad \mathbb{I}_{g(\boldsymbol{x}'',t'')}(\boldsymbol{x}_1, t_1) = 1 \quad q(\boldsymbol{x}'', t''|\boldsymbol{x}_1, t_1) = q(\boldsymbol{x}', t'|\boldsymbol{x}_2, t_2) \tag{A.4}$$

It follows that

$$\begin{aligned}
&\pi(\boldsymbol{x}_1, t_1) \int_{\Omega\times[0,T]\setminus\mathcal{R}^{d+1}} q(\boldsymbol{x}', t'|\boldsymbol{x}_1, t_1)\mathbb{I}_{g(\boldsymbol{x}',t')}(\boldsymbol{x}_2, t_2) d\boldsymbol{x}' dt' \min(1, \frac{\pi(\boldsymbol{x}_2, t_2)}{\pi(\boldsymbol{x}_1, t_1)}) \\
=&\pi(\boldsymbol{x}_1, t_1) \int_{\Omega\times[0,T]\setminus\mathcal{R}^{d+1}} q(\boldsymbol{x}'', t''|\boldsymbol{x}_2, t_2)\mathbb{I}_{g(\boldsymbol{x}'',t'')}(\boldsymbol{x}_1, t_1) d\boldsymbol{x}' dt' \min(1, \frac{\pi(\boldsymbol{x}_2, t_2)}{\pi(\boldsymbol{x}_1, t_1)}) \\
=&\pi(\boldsymbol{x}_2, t_2) \int_{\Omega\times[0,T]\setminus\mathcal{R}^{d+1}} q(\boldsymbol{x}'', t''|\boldsymbol{x}_2, t_2)\mathbb{I}_{g(\boldsymbol{x}'',t'')}(\boldsymbol{x}_1, t_1) d\boldsymbol{x}' dt' \min(1, \frac{\pi(\boldsymbol{x}_1, t_1)}{\pi(\boldsymbol{x}_2, t_2)})
\end{aligned} \tag{A.5}$$

Combining A.3 and A.5, we prove the detailed balance condition for the transition kernel q^* with respect to π □