Efficient and Unbiased Sampling of Molecular Energy Functions via Consistency Models

Fengzhe Zhang

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Master of Philosophy in Machine Learning and Machine Intelligence

Downing College October 2024

I would like to dedicate this thesis to my loving parents and grandparents.

Declaration

I, Fengzhe Zhang of Downing College, being a candidate for the MPhil in Machine Learning and Machine Intelligence, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

Software Declaration: In Section 4.2, the molecular dataset is generated using the code provided by Javier Antorán and Laurence Midgley. The implementation of the Equivariant Graph Neural Network is based on the EGNN repository. All other implementations are carried out using standard Python packages.

Word Count: 14981

Fengzhe Zhang
October 2024

Acknowledgements

I would like to begin by expressing my gratitude to Prof. José Miguel Hernández Lobato for making this project possible. The last four months have been incredibly instrumental to my growth, and I thoroughly enjoyed every discussion we had. I deeply appreciate the guidance and suggestions you provided.

I am also profoundly grateful to my co-supervisors: Jiajun He, Javier Antorán, and Laurence Midgley. Thank you for proposing and refining the central idea of this project. Special thanks to Jiajun for your expertise in Consistency Models, and to Javier and Laurence for your expertise in molecular experiments. I am sincerely thankful for your valuable suggestions throughout the project, your patience in answering my countless questions, and your unwavering support to ensure I fully understood every concept. It was truly a rewarding experience collaborating with you all, and you have each been a great inspiration to me.

I would like to extend a very special thanks to my girlfriend, Zhuoyue Huang. Thank you for your support over the past three years, from Jinan to London to Cambridge. Your brilliance as a mathematician and machine learning researcher has been a constant source of inspiration for me. I could not have achieved what I have today without you, and I look forward to the many more years we will share together.

I also want to express my heartfelt thanks to my friends in the MLMI cohort, Nanze Chen, Junyi Qian. You have brought so much color and joy to my life over the past year.

Finally, I would like to thank my parents and grandparents for their unwavering support throughout my journey. Thank you for giving me the courage to keep moving forward.

Abstract

Sampling from molecular energy functions has been a pivotal research area due to its potential to predict properties of new drugs or materials solely through computer simulations. However, this task is exceedingly challenging, as current sampling methods, particularly those using Markov Chain Monte Carlo (MCMC) (Hastings, 1970; Metropolis et al., 1953), are computationally expensive and time-consuming. A promising direction involves leveraging recently developed diffusion models (Ho et al., 2020; Sohl-Dickstein et al., 2015), which can be trained to learn distributions according to energy functions and subsequently generate samples. However, two significant challenges persist with this approach. First, generating samples from diffusion models remains **inefficient**, often requiring hundreds of time steps to produce high-quality samples. Second, there is an inherent **bias** in the generated samples. Due to the finite training data and limited model capacity, diffusion models typically learn an approximation of the true distribution rather than the exact one. As a result, predictions made using these biased samples can lead to incorrect and potentially misleading conclusions.

There are solutions to these challenges. Importance Sampling (IS) can correct the bias in the samples, while the newly developed Consistency Models (CMs) (Kim et al., 2023; Li and He, 2024; Song et al., 2023) significantly accelerate sampling from diffusion models. In this thesis, we propose a novel sampling method that strategically combines IS with CMs, allowing us to obtain **unbiased** samples using only **3-10** time steps. We validate our proposed method on a toy dataset and conduct experiments on the dynamics of supercooled water molecules, a notoriously challenging problem due to the complex interactions between atoms.

Table of contents

List of tables

List of figures

Chapter 1

Introduction

Sampling from molecular energy functions has been a key focus in the field of natural science. Efficient sampling would enable the prediction of properties for new materials and drugs solely through computational simulations, avoiding the need for expensive real-world experiments. However, sampling from molecular energy functions is extremely challenging due to their high dimensionality and multimodal nature. Traditional approaches, such as Monte Carlo Markov Chain (MCMC) (Hastings, 1970; Metropolis et al., 1953) methods, are extremely time-consuming for this task.

Recently, advancements in Diffusion Models (DMs) (Ho et al., 2020; Sohl-Dickstein et al., 2015) have opened a promising direction by training these models to learn the distribution of molecular energy functions. Once trained, samples can be generated from the diffusion models instead of running MCMC. However, this approach presents two significant issues. Firstly, generating samples from diffusion models is slow, often requiring hundreds of evaluations to produce high-quality samples. Secondly, estimates derived from these samples, such as integrals, are biased. This bias arises because it is impossible to achieve a perfect diffusion model that draws samples exactly matching the target energy functions, leading to a discrepancy between the generated sample distribution and the true distribution. This is particularly undesirable for molecular applications, where accurate predictions are crucial.

In this thesis, we aim to address these two problems associated with sampling molecular energy functions using DMs. To obtain unbiased samples, we can utilize a straightforward technique called Importance Sampling (IS). IS allows for the generation of unbiased samples when we can evaluate the target distribution (i.e., molecular energy functions) but cannot sample from it directly. Instead, we use a proposal distribution (i.e., diffusion models) from which we can draw samples and then reweight them to achieve unbiased samples from the target distribution. However, naively combining IS with diffusion models still requires hundreds of steps to achieve low variance in IS estimation.

Leveraging recent advancements in Consistency Models (CMs) (Kim et al., 2023; Li and He, 2024; Song et al., 2023), which are a new type of generative model trained through distillation from pre-trained diffusion models, enables sampling using only one step. This significantly accelerates the sampling process. However, the deterministic nature of CMs means they do not define a valid proposal distribution and thus cannot be straightforwardly combined with IS.

In this work, we develop an algorithm that carefully integrates CMs and IS, reducing the number of steps to between **3 and 10** while achieving the same IS variance that previously required hundreds of steps. This approach significantly speeds up the sampling process while maintaining the primary advantage of IS: obtaining unbiased samples.

1.1 Contributions

Our contributions can be summarized as follows:

- We provide a **comprehensive review** of diffusion models, with a particular focus on score-based diffusion models and the family of consistency models. We summarize their theoretical foundations and highlight the strengths and limitations of each approach.
- We develop a **novel sampling method** that combines importance sampling with consistency models. This approach significantly reduces the computational cost of obtaining unbiased samples from molecular energy functions. We present a rigorous mathematical formulation and validate its effectiveness through extensive experiments.
- Inspired by the formulation of Bidirectional Consistency Models (BCMs), we extend the training methodology of Consistency Trajectory Models (CTMs) to **develop Bidirectional CTMs (BCTMs)**. Furthermore, we enhance the family of Consistency Models (CMs) by introducing **E(3)-Equivariant CMs**, which improve performance on molecular datasets. This is achieved by integrating Equivariant Graph Neural Networks (EGNN) (Hoogeboom et al., 2022; Satorras et al., 2021) with CMs.
- We propose **potential improvements** to our algorithm and outline directions for future research.

1.2 Outline

This thesis is organized as follows:

- **Chapter 2** introduces the background knowledge required for our method. We first cover the basics of Importance Sampling, laying the foundation for later chapters and introducing the metrics we will use. Then, we provide background knowledge on diffusion models, focusing on score-based diffusion models. We further discuss the family of consistency models, which build upon and are closely related to score-based diffusion models.
- **Chapter 3** details both the baseline method and our novel approach. For the baseline method, we describe it in greater detail and provide mathematical formulations for combining diffusion models directly with importance sampling, discussing the potential limitations of this method. Then, we elaborate on our novel method, where we redesign the proposal and target distributions, and explain how consistency models accelerate the entire sampling process.
- **Chapter 4** is dedicated to experiments. We conduct two experiments: one with a toy dataset generated by a Gaussian Mixture Model (GMM) to verify our method using an analytical score function, and another using a trained model to approximate the score function. We provide a detailed comparison between our method and the baseline in different spaces and dimensions. Subsequently, we conduct experiments in a more practical setting, simulating the dynamics of water molecules at very cold temperatures, comparing our method with the baseline in this context.
- **Chapter 5** concludes the thesis, summarizing the experimental results, discussing the limitations of our method, and suggesting possible future directions.

Chapter 2

Background

In this chapter, we provide the necessary background and theoretical foundations for this dissertation. We present a detailed introduction to the methods and models briefly described in Chapter 1, laying the groundwork for the novel method presented in Chapter 3. Our discussion is structured as follows:

1. We begin with a comprehensive review of Importance Sampling, a crucial sampling technique that underpins our proposed algorithm.
2. We then focus on two key areas: Diffusion Models and the family of Consistency Models. For each, we explore core concepts, recent developments and state-of-the-art approaches, as well as strengths and limitations in current applications.

This thorough examination of existing methodologies serves as a foundation for understanding the novel contributions presented in subsequent chapters. By elucidating these fundamental concepts, we aim to provide readers with the necessary context to appreciate the innovations and advancements proposed in our work.

2.1 Importance Sampling

Consider the task of estimating integrals of the form $\mathbb{E}_{x\sim p}[\phi(x)]$ for some target distribution p that we can draw samples and a function ϕ that we can evaluate. One way to estimate this integral is through Monte Carlo (MC) estimation:

$$\mathbb{E}_{x\sim p}[\phi(x)] = \int \phi(x)p(x)dx \approx \frac{1}{N}\sum_{n=1}^{N} \phi(x^{(n)}) =: \hat{\phi}_{\mathrm{MC}}^{N}, \tag{2.1}$$

where $x^{(n)} \sim p$ and N is the total number of samples used. The MC estimator $\hat{\phi}^N_{\text{MC}}$ is unbiased (the expectation of the estimate equals the true value) and consistent (the variance of the estimate decreases to zero as the number of samples N increases to infinity).

However, in practice, sampling directly from p is often impossible, and we can only evaluate it. An alternative approach is to use Importance Sampling (IS). Consider a proposal distribution q from which we can sample and evaluate (and the support[1] of q contains the support of p). We can rewrite the integral as:

$$\int \phi(x)\frac{p(x)}{q(x)}q(x)dx = \mathbb{E}_{x\sim q}\left[\frac{p(x)}{q(x)}\phi(x)\right] \approx \frac{1}{N}\sum_{n=1}^{N}\frac{p(x^{(n)})}{q(x^{(n)})}\phi(x^{(n)}) =: \hat{\phi}^N_{\text{IS}}, \tag{2.2}$$

where $x^{(n)} \sim q$. Define $w_n := \frac{p(x^{(n)})}{q(x^{(n)})}$ for $n = 1,\dots,N$ as the importance weights. The IS estimator is still unbiased and consistent, but the variance of the IS estimator depends on both the target distribution p and the function ϕ. The main challenge of IS is designing the proposal distribution such that the IS variance is minimized. This will be a key focus of our later method.

In practice, to assess the effectiveness of the IS estimator, the metric called Effective Sample Size (ESS) is used. Intuitively, ESS indicates the number of samples from the proposal distribution that are equivalent to the samples from the target distribution for estimating the integral. ESS can be estimated as:

$$\widehat{\text{ESS}}^{\text{prop}} := \frac{(\sum_{n=1}^{N} w_n)^2}{\sum_{n=1}^{N} w_n^2}, \tag{2.3}$$

where $1 \leq \widehat{\text{ESS}}^{\text{prop}} \leq N$. This ESS is estimated using the samples from the proposal. There is another formulation to estimate ESS based on samples from the target:

$$\widehat{\text{ESS}}^{\text{tar}} := \frac{N}{\mathbb{E}_{x\sim p}\left[\frac{p(x)}{q(x)}\right]}. \tag{2.4}$$

The estimation $\widehat{\text{ESS}}^{\text{tar}}$ provides a more reliable assessment when the proposal distribution adequately covers[2] the target distribution. When ESS is very low, the variance of the estimate will be large. However, a high ESS does not necessarily imply a reliable IS estimate with

[1] The support of a probability density function q is defined as the set of all x for which $q(x) > 0$.

[2] Mathematically, "cover" means the support of the proposal distribution contains the support of the target distribution. While our Gaussian proposal used later theoretically has infinite support, in practice, samples aren't drawn from extreme tails. Thus, we define "covering" practically: the sample distribution from the proposal (using a large, finite number of samples) should completely encompass that from the target.

low variance, as it is still possible that the proposal distribution does not adequately cover the target distribution. In such cases, samples from the proposal may miss important regions of the target distribution. To assess the efficiency of the proposal, we need to test it using different ϕ to estimate the integral. If the variances of the estimates are consistently low across various ϕ, we can be confident that the proposal design is effective.

Another challenge for IS arises when the normalizing constant for the target distribution is unknown, which is the case for molecular energy functions. In such scenarios, we denote the unnormalized target distribution as $\bar{p}$ and employ Self-Normalizing Importance Sampling (SNIS). Rewriting the integral in terms of $\bar{p}$:

$$\frac{\int \phi(x) \frac{\bar{p}(x)}{q(x)} q(x) dx}{\int \frac{\bar{p}(x)}{q(x)} q(x) dx} \approx \frac{\frac{1}{N} \sum_{n=1}^{N} \frac{\bar{p}(x^{(n)})}{q(x^{(n)})} \phi(x^{(n)})}{\frac{1}{N} \sum_{n=1}^{N} \frac{\bar{p}(x^{(n)})}{q(x^{(n)})}} = \frac{\sum_{n=1}^{N} w_n \phi(x^{(n)})}{\sum_{n=1}^{N} w_n} = \sum_{n=1}^{N} \bar{w}_n \phi(x^{(n)}), \tag{2.5}$$

where $\bar{w}_n := w_n / \sum_{m=1}^{N} w_m$ are the normalized importance weights. It is important to note that SNIS is asymptotically unbiased and consistent, meaning the IS estimation becomes unbiased only as the number of samples approaches infinity.

IS is a crucial technique for obtaining asymptotically unbiased estimations of integrals. In the following section, we will review diffusion models. Subsequently, in Chapter 3, we will explore the combination of IS with diffusion models as a baseline method.

2.2 Diffusion Models

In this section, we will review diffusion models, including their theoretical foundations and recent developments. First, we will discuss the foundational concepts for diffusion models in general, including the definition of the diffusion process and the reverse process. Then, we will review recently developed score-based diffusion models, which form the foundation for the family of consistency models introduced later.

2.2.1 Denoising Diffusion Probabilistic/Implicit Models

Diffusion probabilistic models (DPMs) (Sohl-Dickstein et al., 2015) are powerful generative models that generate samples by progressively adding noise to perturb the data and then creating samples from noise via sequential denoising steps. In this section, we aim to cover the basic setup of diffusion models, including the theoretical foundations, which will be crucial for the subsequent chapter. We review the formulation of diffusion models following the settings in Kingma et al. (2021), including recent advancements like Denoising Diffusion Probabilistic Models (DDPM) (Ho et al., 2020) and Denoising Diffusion Implicit Models (DDIM) (Song et al., 2020a).

Diffusion Process

Given a data point sampled from the data distribution $x_0 \sim p_{\text{data}}(x)$, the diffusion process adds noise to the data, defined by a multivariate normal distribution for $t = 1, \dots, T$:

$$q(x_t|x_0) := \mathcal{N}(x_t; \alpha_t x_0, \sigma_t^2 I), \tag{2.6}$$

where $\alpha_t \in \mathbb{R}^+$ controls how much of the data is retained and $\sigma_t \in \mathbb{R}^+$ controls how much noise is added. Note that the diffusion process is Markovian and thus can be written as (for $s < t$):

$$q(x_t|x_s) = \mathcal{N}(x_t|\alpha_{t|s} x_s, \sigma_{t|s}^2 I), \tag{2.7}$$

where $\alpha_{t|s} = \alpha_t / \alpha_s$ and $\sigma_{t|s}^2 = \sigma_t^2 - \alpha_{t|s}^2 \sigma_s^2$. As a special case, DDPM (Ho et al., 2020) sets $\alpha_t = \sqrt{1 - \sigma_t}$ so that the diffusion process is variance-preserving. Conversely, DDIM (Song et al., 2020a), score-based diffusion models (Karras et al., 2022; Song and Ermon, 2019; Song et al., 2020b), and the family of consistency models (Kim et al., 2023; Li and He, 2024; Song et al., 2023) set $\alpha_t^2 = 1$ as a variance-exploding process. We will focus on the variance-preserving process for now, as we review models like DDPM. We will turn our attention to the variance-exploding process later, and all our experiments will be based on the variance-exploding process.

The entire diffusion process can be written as:

$$q(x_0, x_1, \dots, x_T) = q(x_0) \prod_{t=1}^{T} q(x_t|x_{t-1}), \tag{2.8}$$

where we sample from the data distribution to get $x_0 \sim q(x_0)$ and then progressively add noise to it, eventually resulting in x_T resembling a simple noise distribution.

Denoising Process

For the true denoising process, the posterior distribution conditioned on x_0 gives the inverse of the diffusion process:

$$q(x_s|x_0, x_t) = \mathcal{N}(x_s|\mu_{t \to s}(x_0, x_t), \sigma_{t \to s}^2 I), \tag{2.9}$$

with $\mu_{t\to s}(x_0,x_t)$ and $\sigma^2_{t\to s}$ defined as:

$$\mu_{t\to s}(x_0,x_t) = \frac{\alpha_{t|s}\sigma_s^2}{\sigma_t^2}x_t + \frac{\alpha_s\sigma_{t|s}^2}{\sigma_t^2}x_0, \quad \sigma_{t\to s} = \frac{\sigma_{t|s}\sigma_s}{\sigma_t}. \tag{2.10}$$

However, in practice, the true value of x_0 is unknown, so it is replaced by the model's prediction $\hat{x}_0 = \phi_\theta(x_t,t)$, given by the neural network ϕ_θ. Therefore, the denoising distribution is defined as:

$$p_\theta(x_s|x_t) = \mathcal{N}(x_s|\mu_{t\to s}(\hat{x}_0,x_t),\sigma^2_{t\to s}I). \tag{2.11}$$

Thus, the entire denoising process can be written as:

$$p_\theta(x_0,x_1,\ldots,x_T) = p_\theta(x_T)\prod_{t=1}^{T}p_\theta(x_{t-1}|x_t), \tag{2.12}$$

where $p_\theta(x_T)$ is the base distribution and it allows us to sample noise and then progressively denoise it according to the denoising distribution with the denoising model, eventually generating the sample x_0. Note that the hyperparameter T is important. A large T allows the diffusion process to add enough noise so that x_T follows a simple Gaussian distribution, from which we can easily draw samples. However, the denoising process described above needs to be performed sequentially, not in parallel, to obtain x_0. Therefore, sampling from diffusion models is time-consuming and much slower than other deep generative models, potentially limiting their applicability in tasks where computational resources are limited and latency is critical.

Training Objective

To train the diffusion model, we aim to minimize the negative log-likelihood. To do so, we aim to minimize the negative variational lower bound:

$$\begin{aligned}\mathbb{E}[-\log p_\theta(x_0)] &\leq \mathbb{E}_{x_{0:T}\sim q}\left[-\log\frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right] && (2.13)\\ &= \mathbb{E}_{x_{0:T}\sim q}\left[-\log p(x_T) - \sum_{t\geq 1}\log\frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}\right] := L && (2.14)\end{aligned}$$

Note that L can be further simplified as

$$L = \mathbb{E}_{x_{0:T} \sim q} \left[\underbrace{D_{\mathrm{KL}}(q(x_T|x_0) \| p(x_T))}_{L_T} + \sum_{t \geq 1} \underbrace{D_{\mathrm{KL}}(q(x_{t-1}|x_t, x_0) \| p_\theta(x_{t-1}|x_t))}_{L_t} \underbrace{- \log p_\theta(x_0|x_1)}_{L_0} \right] \tag{2.15}$$

Although the above training objective can be employed to train the model, Ho et al. (2020) identified some further improvements for the parameterization and training objective to enhance model performance. Initially, the network ϕ_θ was parameterized to directly predict $\hat{x}_0$, but Ho et al. (2020) found that the training process is more stable and easier if the model is parameterized to predict the noise at time t, i.e., $\hat{\varepsilon}_\theta(x_t, t) = \phi_\theta(x_t, t)$. If $x_t = \alpha_t x_0 + \sigma_t \varepsilon$, then the predicted $\hat{x}_0$ is given by:

$$\hat{x}_0 = \frac{1}{\alpha_t} x_t - \frac{\sigma_t}{\alpha_t} \hat{\varepsilon}_\theta(x_t, t) \tag{2.16}$$

This predicted $\hat{x}_0$ can then be used in the denoising distribution.

Ho et al. (2020) also found that the following training objective, a variant of the variational lower bound, leads to better performance and is easier to implement:

$$L_{\mathrm{simple}}(\theta) := \mathbb{E}_{t, x_0, \varepsilon} \left[||\varepsilon - \hat{\varepsilon}_\theta(\alpha_t x_0 + \sigma_t \varepsilon, t)||^2 \right] \tag{2.17}$$

where t is uniformly sampled between 1 and T. This objective is the same as that used in noise conditional score networks (Song and Ermon, 2019) based on score matching (Hyvärinen and Dayan, 2005; Vincent, 2011).

DDIM: Accelerated Sampling

Denoising Diffusion Implicit Models (DDIMs) (Song et al., 2020a) were proposed to accelerate the DDPM sampling process. One key observation that motivates the proposal of DDIM is that the training objective, like L_{simple}, only depends on the (conditional) marginal $q(x_t|x_0)$ and not directly on the (conditional) joint $q(x_{1:T}|x_0)$. Since many inference distributions (joints) have the same marginals, this motivates the exploration of alternative inference processes that are non-Markovian, leading to new generative processes. Remarkably, Song et al. (2020a) showed that these non-Markovian inference processes lead to the same surrogate objective function as DDPM.

For DDPM, we assume the Markovian property in the diffusion and denoising processes. However, DDIM generalizes this assumption to be non-Markovian and derives the corresponding diffusion and denoising processes. They identified and proved a crucial property: models trained based on the training objective L_{simple} not only learn the generative process for the Markovian inference process considered in DDPM but also for many non-Markovian forward processes. Thus, we can essentially use pre-trained DDPM models as solutions to new objectives and focus on finding a generative process better suited to our needs.

While Song et al. (2020a) derived the DDIM denoising process for the variance-preserving process, we will focus on the variance-exploding process, as it will be utilized in later sections. Let us derive the DDIM denoising process for this case. Consider the variance-exploding process with $\alpha_t^2 = 1$. The diffusion process is given as $q(x_t|x_0) = \mathcal{N}(x_t; x_0, \sigma_t^2 I)$. The corresponding generative denoising process is given as:

$$p_\theta(x_{t-1}|x_t, \hat{x}_0) = \mathcal{N}\left(x_{t-1}; \frac{\sigma_{t-1}^2}{\sigma_t^2} x_t + \frac{\sigma_t^2 - \sigma_{t-1}^2}{\sigma_t^2} \hat{x}_0, \frac{(\sigma_t^2 - \sigma_{t-1}^2)\sigma_{t-1}^2}{\sigma_t^2} I\right) \tag{2.18}$$

Now consider a different generative denoising process given as:

$$q_\gamma(x_{t-1}|x_t, \hat{x}_0) = \mathcal{N}\left(x_{t-1}; x_t\sqrt{\frac{\sigma_{t-1}^2 - \gamma_{t-1}^2}{\sigma_t^2}} + \hat{x}_0\left(1 - \sqrt{\frac{\sigma_{t-1}^2 - \gamma_{t-1}^2}{\sigma_t^2}}\right), \gamma_{t-1}^2 I\right) \tag{2.19}$$

We define γ_{t-1} as:

$$\gamma_{t-1} = \eta\sqrt{\frac{(\sigma_t^2 - \sigma_{t-1}^2)\sigma_{t-1}^2}{\sigma_t^2}} \tag{2.20}$$

where η can be any value between 0 and 1. It can be shown via Bayes' rule that both denoising generative processes have the same forward marginal distribution $q(x_t|x_0) = \mathcal{N}(x_t; x_0, \sigma_t^2 I)$. Therefore, if a DDPM model is trained for the generative denoising process as in Eq. (2.18), then Eq. (2.19) is also a valid way to obtain samples from the trained diffusion model for any $\eta \in [0, 1]$.

Interestingly, when $\eta = 1$, Eq. (2.19) is exactly the DDPM denoising distribution in Eq. (2.18). When $\eta = 0$, the generative process becomes deterministic and is given by:

$$x_{t-1} = \frac{\sigma_{t-1}}{\sigma_t} x_t + \left(1 - \frac{\sigma_{t-1}}{\sigma_t}\right)\hat{x}_0 \tag{2.21}$$

This is exactly the first-order Euler solver of the Probability Flow Ordinary Differential Equation (Song et al., 2020b) in the case $\alpha_t^2 = 1$, which is known to produce high-quality samples using fewer time steps. We will introduce more about this in the next section.

2.2.2 Score-based Diffusion Models

In this section, we explore a different yet closely related perspective on diffusion models through score modeling. Generally, there are two primary approaches to corrupting clean data and subsequently training a model to reverse this process for generating new data:

1. The Denoising Diffusion Probabilistic Model (DDPM) approach, which we introduced in the previous section. This method trains a sequence of probabilistic models to reverse each step of the noise corruption process, using these models to draw samples starting from pure noise.
2. The Score Matching with Langevin Dynamics (SMLD) method (Song and Ermon, 2019). This approach estimates the score at each noise level, denoted as $\nabla_x \log p(x; \sigma_t)$. Sampling is then performed using Langevin dynamics to sample from a sequence of decreasing noise scales during the generation process.

It's worth noting that for continuous state spaces, the DDPM training objective implicitly computes the score at each noise level. Recognizing the underlying connections between these approaches, Song et al. (2020b) proposed a unifying framework by defining forward and reverse stochastic differential equations (SDEs). This unification provides a more comprehensive understanding of the relationship between these seemingly distinct methods.

Forward and Reverse SDE

The diffusion process is defined by the following forward SDE:

$$dx = f(x,t)dt + g(t)dw \tag{2.22}$$

Here, t is a continuous variable defined as $t \in [0,T]$, and w is the standard Wiener process. f and g are the drift and diffusion coefficients of x_t, respectively. As t increases, the data will be perturbed more and eventually become white noise. The score model $s_\theta(\cdot,\cdot)$ can be trained via score matching (Hyvärinen and Dayan, 2005; Song and Ermon, 2019). To draw samples, we follow the reverse SDE using the results from Anderson (1982):

$$dx = \left[f(x,t) - g(t)^2 \nabla_x \log p_t(x)\right] dt + g(t)d\bar{w} \tag{2.23}$$

where $\bar{w}$ is a standard Wiener process running backwards in time. In practice, $\nabla_x \log p_t(x)$ will be replaced by the modeled score s_θ to generate samples.

Probability Flow ODE

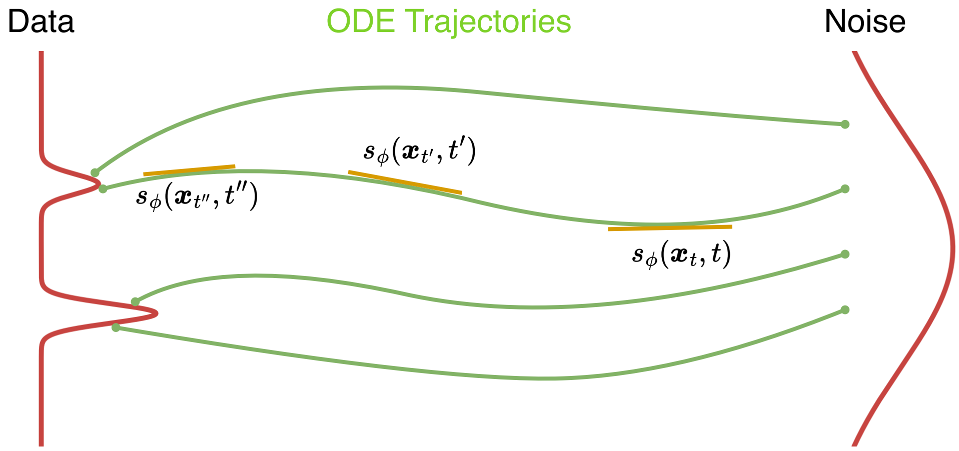

Data
ODE Trajectories
Noise
$s_\phi(\boldsymbol{x}_{t'}, t')$
$s_\phi(\boldsymbol{x}_{t''}, t'')$
$s_\phi(\boldsymbol{x}_t, t)$

Fig. 2.1 Illustration of the sampling process of score-based diffusion models. The green curves represent PF ODE trajectories, which establish a one-to-one correspondence between samples from the noise distribution and the data distribution. The actual score for the PF ODE is modeled by $s_\phi(x_t, t)$, trained via score matching. To generate samples, we start from noise sampled from the base distribution and then solve the PF ODE using the modeled score at each time step, gradually moving towards the data distribution.

Remarkably, Song et al. (2020b) proved that there exists a deterministic ordinary differential equation (ODE) corresponding to this reverse SDE, given as:

$$dx = \left[f(x,t) - \frac{1}{2} g(t)^2 \nabla_x \log p_t(x) \right] dt \tag{2.24}$$

This PF ODE possesses a unique property: moving along its trajectory forward in time adds noise, while moving backward in time removes noise. Importantly, any off-the-shelf ODE solver (e.g., 1st order Euler solver) can be employed to solve the PF ODE, generating samples at time 0 given the initial noise x_T at time T (see Figure 2.1).

It's crucial to note that the ODE itself is deterministic; the only source of randomness comes from the initial point. Song et al. (2020b) highlight that generating samples by solving the PF ODE not only produces high-quality samples but also allows for an explicit trade-off between accuracy and efficiency. Specifically, by increasing the error tolerance, we can solve the PF ODE with fewer discretization steps, thus achieving better computational efficiency.

In the next section, we will explore how, through extensive experiments conducted by Karras et al. (2022), carefully chosen architecture, preconditioning, and sampling methods can significantly enhance the performance of score-based diffusion models using the PF ODE approach.

2.2.3 Elucidating the Design Space of Score-based Diffusion Models

Building on the score-based diffusion models introduced in the previous subsection, Karras et al. (2022) identified several changes to the diffusion model architecture, sampling, and training process. These changes have led to significant improvements in model performance and are employed in consistency models, making them crucial to review here. We highlight the key changes and briefly introduce the motivations behind them. The settings used by Karras et al. (2022) will be utilized in all our subsequent experiments.

ODE Formulation Revisited

Recall the formulation of the PF ODE previously reviewed in Eq. (2.24). After extensive experiments, Karras et al. (2022) argue that the optimal choices for $f(x,t)$ and $g(t)$ are $f(x,t) = 0$ and $g(t) = \sqrt{2t}$. With this setting, the forward diffusion SDE can be written as

$$dx = \sqrt{2t}dw \tag{2.25}$$

where w is the standard Wiener process. Hence, for $s < t$, the forward diffusion process is defined as $q(x_t|x_s) = \mathcal{N}(x_t; x_s, t^2 - s^2)$. Simultaneously, with these choices, the PF ODE simplifies to

$$\frac{dx}{dt} = -t\nabla_x \log p_t(x) \tag{2.26}$$

for $t \in [\varepsilon, T]$, where $\varepsilon = 0.002$ and $T = 80$ as set by Karras et al. (2022). This PF ODE formulation will be used throughout the consistency models and our experiments.

Preconditioning and Training

Since the optimal σ_t found by Karras et al. (2022) is $\sigma_t = t$, we use σ_t and t interchangeably in our discussion. As previously mentioned, parameterizing the neural network to directly predict $\hat{x}_0$ is not ideal, because the input $x_t = x_0 + n$ is a combination of the clean signal x_0 and noise $n \sim \mathcal{N}(0, \sigma_t^2 I)$, whose magnitude varies significantly depending on the noise level. The common practice is to train a neural network F_θ, from which the denoiser D_θ can be derived to predict $\hat{x}_0$. For example, in DDPM, F_θ is trained to predict the noise scaled to unit variance. The denoiser D_θ is then constructed as $D_\theta(x,\sigma_t) = x - \sigma_t F_\theta(x,\sigma_t)$. However, this approach has a drawback: for large values of σ_t, the neural network F_θ must finely tune the output to cancel the noise, ensuring the prediction is at the correct scale.

Karras et al. (2022) propose parameterizing the denoiser as follows:

$$D_\theta(x,\sigma_t) = c_{\text{skip}}(\sigma_t)x + c_{\text{out}}(\sigma_t)F_\theta(c_{\text{in}}(\sigma_t)x, c_{\text{noise}}(\sigma_t)) \tag{2.27}$$

where F_θ is the neural network to be trained. The preconditioning functions are defined as:

$$c_{\text{skip}}(\sigma_t) = \frac{\sigma_{\text{data}}^2}{\sigma_t^2 + \sigma_{\text{data}}^2} \qquad c_{\text{out}}(\sigma_t) = \frac{\sigma_t \cdot \sigma_{\text{data}}}{\sqrt{\sigma_t^2 + \sigma_{\text{data}}^2}}$$

$$c_{\text{in}}(\sigma_t) = \frac{1}{\sqrt{\sigma_t^2 + \sigma_{\text{data}}^2}} \qquad c_{\text{noise}}(\sigma_t) = \frac{1}{4}\log(\sigma_t)$$

where c_{skip} modulates the skip connection and is chosen to amplify errors in F_θ as little as possible. c_{in} and c_{out} scale the input and output magnitudes such that inputs and training targets have unit variance. c_{noise} maps the noise level into a conditioning input for F_θ, and its formula is chosen empirically.

To train the diffusion model, the following training objective is used:

$$L(\theta) = \mathbb{E}_{\sigma_t \sim p_{\text{train}}, x_0 \sim p_{\text{data}}, n \sim \mathcal{N}(0,\sigma^2 I)}\left[\lambda(\sigma_t)\|D_\theta(x_0 + n, \sigma_t) - x_0\|_2^2\right] \tag{2.28}$$

where the noise distribution is defined as $\log\sigma_t \sim \mathcal{N}(P_{\text{mean}}, P_{\text{std}})$ (with $P_{\text{mean}} = -1.2$ and $P_{\text{std}} = 1.2$). This choice is based on experimental results which reveal that significant reduction is possible only at intermediate noise levels. Therefore, Karras et al. (2022) target the training efforts to the relevant range using a simple log-normal distribution for $p_{\text{train}}(\sigma_t)$. The loss weight $\lambda(\sigma_t)$ is defined as $\lambda(\sigma_t) = \frac{1}{c_{\text{out}}(\sigma_t)^2} = \frac{\sigma_t^2 + \sigma_{\text{data}}^2}{(\sigma_t \cdot \sigma_{\text{data}})^2}$, which greatly stabilizes the training process. These preconditioning and training settings will be employed in the consistency models and our subsequent experiments as well.

Sampling

To generate samples using score-based diffusion models, it is necessary to solve the PF ODE numerically, which requires approximating the true trajectory. At each step, a truncation error is introduced by the solver, which accumulates over N steps. The local error generally scales superlinearly with respect to the step size, and thus higher N will improve the accuracy of the approximation for the true trajectory. The commonly used first-order ODE solver is Euler's method, which has $\mathscr{O}(h)$ local error with respect to the step size h.

After extensive experiments, Karras et al. (2022) found that Heun's second-order method (Ascher and Petzold, 1998) provides an excellent trade-off between truncation error and neural function evaluations (NFE). It has $\mathscr{O}(h^3)$ local error at the cost of one additional evaluation of D_θ per step. However, we will not review the details of different ODE solvers as they will be replaced by consistency models introduced later. For experimental and illustration purposes in later chapters, we use the first-order Euler's method.

In addition to the ODE solver, it is also important to specify the time steps $\{t_i\}_{i=1}^N$ to determine how the step sizes, and thus truncation errors, are distributed between different noise levels. After extensive experiments, Karras et al. (2022) found the following optimal time step arrangement, assuming $\varepsilon = t_1 < t_2 < \cdots < t_N = T$:

$$t_i = \left(\varepsilon^{\frac{1}{\rho}} + \frac{i-1}{N-1} \left(T^{\frac{1}{\rho}} - \varepsilon^{\frac{1}{\rho}} \right) \right)^{\rho} \tag{2.29}$$

where $\rho = 7$. Intuitively, ρ controls how much the steps near ε are shortened at the expense of longer steps near T. Analysis in (Karras et al., 2022) shows that $\rho = 3$ nearly equalizes the truncation error at each step, but ρ values from 5 to 10 perform better for sampling images. In terms of combining importance sampling with diffusion models, we found that during experiments, the setting $\rho \to \infty$ generally gives the highest ESS. We will explain more about this time schedule in Section 3.1.3.

2.2.4 E(3) Equivariant Diffusion Models

Molecular datasets exist in physical 3D spaces and are subject to geometric symmetries such as translations, rotations, and possibly reflections. These symmetries are referred to as the Euclidean group in 3 dimensions, E(3). To effectively train models with molecular datasets, it is crucial to utilize these symmetries, as they significantly enhance the generalization ability of the model. In this subsection, we will discuss E(3) equivariant diffusion models, which will be utilized in our experiments involving water molecules in Chapter 4.

Equivariance

We first present the definition of equivariance. Suppose $T_g : X \to X$ is a set of transformations on X for the abstract group $g \in G$. A function $\phi : X \to Y$ is called equivariant to g if there exists an equivariant transformation $S_g : Y \to Y$ such that $\phi(T_g(x)) = S_g(\phi(x))$. In our case, we consider the Euclidean group E(3), which is generated by translations, rotations, and reflections. We use R, an orthogonal matrix, to represent rotations and reflections, and t to represent translations of coordinates. For example, we say ϕ is E(3) equivariant if $\phi(Rx+t) = R\phi(x)+t$.

Similar to Hoogeboom et al. (2022), we consider point clouds $x = (x_1, \ldots, x_M) \in \mathbb{R}^{M\times 3}$ with corresponding features $h = (h_1, \ldots, h_M) \in \mathbb{R}^{M\times \mathrm{nf}}$ where nf is the dimension of the node feature. The features h are invariant to group transformations, while the positions x are equivariant to rotations, reflections, and translations such that $Rx+t = (Rx_1+t, \ldots, Rx_M+t)$. We define the function $(z_x, z_h) = f(x,h)$ to be equivariant if for all R and t, we have

$$Rz_x+t, z_h = f(Rx+t, h) \tag{2.30}$$

Equivariant Graph Neural Networks

E(n) Equivariant Graph Neural Networks (EGNNs) (Satorras et al., 2021) are a special type of graph neural network that satisfy the equivariant constraints specified in (2.30). Considering the interactions between all atoms, we assume a fully connected graph $\mathscr{G}$ with nodes $v_i \in \mathscr{V}$. Each node v_i is associated with coordinates $x_i \in \mathbb{R}^3$ and features $h_i \in \mathbb{R}^d$. EGNN consists of a composition of multiple Equivariant Graph Convolutional Layers (EGCL), i.e., $x^{l+1}, h^{l+1} = \mathrm{EGCL}\left[x^l, h^l\right]$ which are defined as

$$m_{ij} = \phi_e(h_j^l, h_i^l, d_{ij}^2, a_{ij}) \tag{2.31}$$

$$h_i^{l+1} = \phi_h(h_i^l, \sum_{j\neq i} \tilde{e}_{ij} m_{ij}) \tag{2.32}$$

$$x_i^{l+1} = x_i^l + \sum_{j\neq i} \frac{x_i^l - x_j^l}{d_{ij}+1} \phi_x(h_i^l, h_j^l, d_{ij}^2, a_{ij}) \tag{2.33}$$

where l indexes the layers of EGCL, $d_{ij} = \|x_i^l - x_j^l\|$ is the Euclidean distance between nodes v_i and v_j, and a_{ij} are optional edge attributes (Hoogeboom et al. (2022); Satorras et al. (2021) set $a_{ij} = e_{ij}$ but they can also include additional edge information). The attention mechanism is used to infer a soft estimation of the edges $\tilde{e}_{ij} = \phi_{\mathrm{inf}}(m_{ij})$. All learnable functions (ϕ_e (edge operations), ϕ_h (node operations), and ϕ_{inf}) are parameterized by fully connected neural networks. Therefore, with this definition, the entire architecture of EGNN is composed of L EGCL layers, i.e., $\hat{x}, \hat{h} = \mathrm{EGNN}[x^0, h^0]$.

Equivariant Diffusion Models

Hoogeboom et al. (2022) propose E(3) Equivariant Diffusion Models (EDMs), which define the diffusion process on both node positions and features, and learn the generative denoising process using EGNN. We will review their definitions and model setups here, which are important for our supercooled water experiments.

Diffusion Process: Considering a single data pair (x_0, h_0), representing the positions and features respectively. Denote $[\cdot, \cdot]$ as concatenation. Using notations from previous sections, the diffusion process is defined as ($z_t = [x_t, h_t]$):

$$q(z_t|x_0, h_0) = \mathcal{N}_{xh}(z_t|\alpha_t[x_0, h_0], \sigma_t^2 I) \tag{2.34}$$

for $t = 1, \ldots, T$. $\mathcal{N}_{xh}$ denotes the product of two distributions, one for noised coordinates $\mathcal{N}_x$ and another for noised features $\mathcal{N}$, given as:

$$\mathcal{N}_x(x_t|\alpha_t x_0, \sigma_t^2 I) \cdot \mathcal{N}(h_t|\alpha_t h_0, \sigma_t^2 I) \tag{2.35}$$

For better readability, x_t, h_t, and z_t should be two-dimensional variables (one axis for point identifier and another axis for features), but they are treated as if flattened to one-dimensional. The noise distribution for h is defined as the conventional normal distribution. However, the noise distribution for x is defined as the normal distribution on a subspace defined by $\sum_i x_i = 0$ to preserve the equivariant property of the model (Xu et al., 2022).

Generative Denoising Process: To define the noise posterior, we replace the true x and h with network predictions $\hat{x}$ and $\hat{h}$:

$$p(z_s|z_t) = \mathcal{N}_{xh}(z_s|\mu_{t\to s}([\hat{x}, \hat{h}], z_t), \sigma_{t\to s}^2 I) \tag{2.36}$$

Similar to DDPM, Hoogeboom et al. (2022) use the network ϕ to predict the noise $\hat{\varepsilon} = [\hat{\varepsilon}^{(x)}, \hat{\varepsilon}^{(h)}]$ and predictions are given by:

$$[\hat{x}, \hat{h}] = \frac{z_t}{\alpha_t} - \hat{\varepsilon} \cdot \frac{\sigma_t}{\alpha_t} \tag{2.37}$$

However, in our experiments, we aim to train both equivariant diffusion models and equivariant consistency models on molecular datasets, and both types of models are based on the settings used by Karras et al. (2022). Hence, we will use a parameterization similar to Karras et al. (2022) but different from Hoogeboom et al. (2022). Similar to Eq. (2.27), we parameterize the model $D_\theta(z_t, \sigma_t)$ as:

$$D_\theta(z_t, \sigma_t) = c_{\text{skip}}(\sigma_t)z_t + c_{\text{out}}(\sigma_t)F_\theta(c_{\text{in}}(\sigma_t)z_t, c_{\text{noise}}(\sigma_t)) \tag{2.38}$$

The preconditioning functions are the same as in Karras et al. (2022). Note that F_θ will be parameterized using EGNN. In this way, D_θ will have all the equivariance properties needed for molecular applications. We will later demonstrate how a similar architecture can be used for consistency models to preserve equivariance properties. The same training objective as in Eq. (2.28) will be used to train this score-based equivariant diffusion model. We will assess the performance of this model in Chapter 4.

So far, we have reviewed all aspects of the diffusion model relevant to our methods and experiments. In the next section, we will focus on the family of consistency models, which will form an important part of our novel method.

2.3 Family of Consistency Models

One limitation of the diffusion models we introduced earlier is the slow sample generation process. Even with ODE solvers such as the first-order Euler's method or Heun's second-order method, generating samples with competitive results still requires evaluating the score model at each iteration, necessitating at least 10 evaluations. To address this issue, a new type of model, Consistency Models (CMs), was proposed by Song et al. (2023). These models learn the trajectory of the PF ODE, supporting single-step generation of samples. Additionally, they allow for iterative generation to enhance sample quality, offering a trade-off between sample quality and computational cost.

In this section, we will review three members of the family of consistency models: the original Consistency Models (CMs), which are trained to map points on the trajectory to the origin of the same trajectory; Consistency Trajectory Models (CTMs) (Kim et al., 2023), which map any point on the same trajectory to any point prior to it; and Bidirectional Consistency Models (BCMs) (Li and He, 2024), which map any point on the trajectory to any other point on the same trajectory.

2.3.1 Consistency Model

In this section, we will review the type of model that is central to the method introduced later: Consistency Models (CMs) (Song et al., 2023). The idea behind all types of consistency models is straightforward: learn the PF ODE trajectory to accelerate the sampling process.

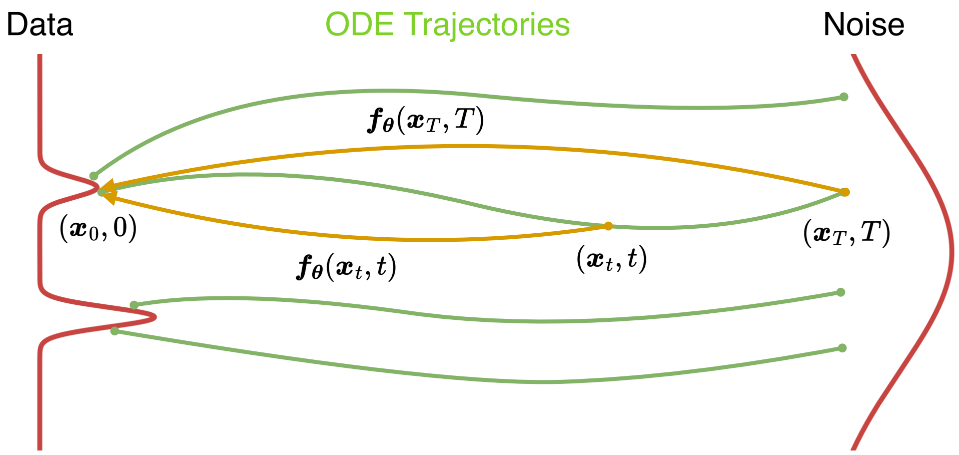


Fig. 2.2 Illustration of the sampling process of consistency models. Unlike score-based diffusion models, which model the score, consistency models directly learn the ODE trajectories between samples from the base distribution and the data distribution. To generate samples, a well-trained consistency model f_θ directly maps samples from the noise distribution or any point along the ODE trajectory to their corresponding samples in the data distribution (highlighted in orange).

Here, we will introduce the original CMs, covering their definition, parameterization, and two training methods—one for distillation and one trained in isolation. While the first CMs proposed have their limitations, which we will discuss, they are not used in subsequent experiments. However, their parameterization and training method significantly influenced the later proposed CTMs and BCMs, which are extensively used in our experiments.

Definition of Consistency Functions

The main motivation for CMs lies in the definition of consistency functions. Suppose $\{x_t\}_{t\in[\varepsilon,T]}$ is a solution trajectory of the PF ODE in Eq. (2.24), then a function defined as $f:(x_t,t)\to x_\varepsilon$ is called a consistency function if it satisfies the self-consistency property defined as:

$$f(x_t,t) = f(x_{t'},t'), \quad \forall t,t' \in [\varepsilon,T] \tag{2.39}$$

In other words, the consistency function will map any point on the same solution trajectory to the same starting point. The goal for CMs is to train a function f_θ according to the data to approximate f as accurately as possible.

Parameterization

As observed from the self-consistency property in Eq. (2.39), the constraint or boundary condition for consistency functions is $f(x_\varepsilon, \varepsilon) = x_\varepsilon$, meaning that $f(\cdot, \varepsilon)$ must be an identity function. To enforce this boundary condition, a simple modification of the parameterization proposed by Karras et al. (2022) can be employed. The consistency model is parameterized as:

$$f_\theta(x,t) = c_{\text{skip}}(t)x + c_{\text{out}}(t)F_\theta(x,t) \tag{2.40}$$

where c_{skip} and c_{out} are modified from Karras et al. (2022) and are defined as:

$$c_{\text{skip}}(t) = \frac{\sigma_{\text{data}}^2}{(t-\varepsilon)^2 + \sigma_{\text{data}}^2}, \quad c_{\text{out}}(t) = \frac{\sigma_{\text{data}}(t-\varepsilon)}{\sqrt{\sigma_{\text{data}}^2 + t^2}} \tag{2.41}$$

These functions satisfy $c_{\text{skip}}(\varepsilon) = 1$ and $c_{\text{out}}(\varepsilon) = 0$, which naturally enforces the boundary condition for f_θ.

Sampling from Consistency Models

Sampling from CMs is straightforward and efficient. Suppose we have a well-trained CM $f_\theta(\cdot,\cdot)$. First, we sample from the initial distribution $\hat{x}_T \sim \mathcal{N}(0, T^2 I)$, and then evaluate the consistency model to obtain $\hat{x}_\varepsilon = f_\theta(\hat{x}_T, T)$. This process requires only one evaluation of the model, enabling sample generation in a single step (see Figure 2.2). Importantly, the sample quality can be further improved by performing multiple denoising and noise injection steps. However, multiple-step sampling is not used in our method and subsequent experiments.

Training Consistency Models

CMs can be trained through two distinct approaches: Consistency Distillation (CD) and Consistency Training (CT). CD utilizes a pre-trained score model, while CT trains the model in isolation without relying on pre-trained models. Our preliminary experiments revealed that CT often requires significantly longer training time compared to CD, yet yields models with superior performance. However, as CMs are not utilized in our proposed algorithm or subsequent experiments, we have included the detailed training methods for both approaches in Appendix A.1.

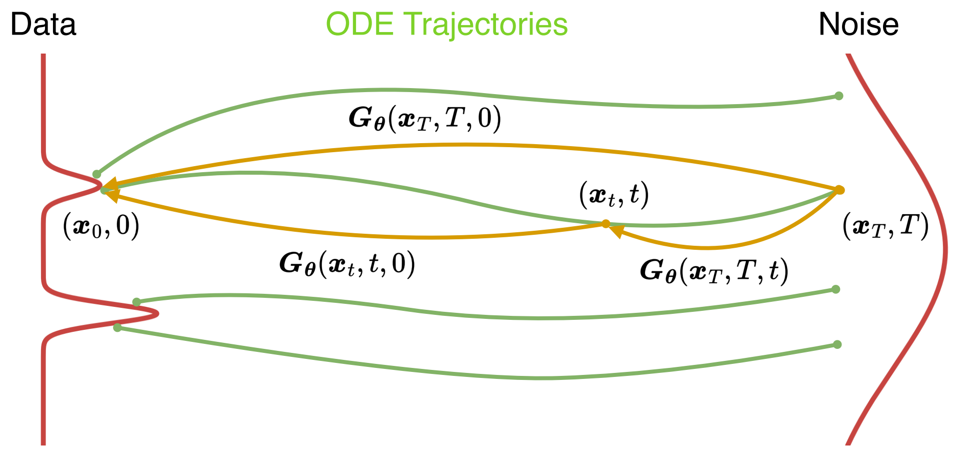


Fig. 2.3 Illustration of the sampling process of consistency trajectory models. Unlike consistency models, which can only map noise directly to samples in the data distribution, Consistency Trajectory Models can map a sample to an intermediate point t and then from t to time 0, producing samples in the data distribution (highlighted in orange). This approach offers a better trade-off between computational cost and sample quality.

2.3.2 Consistency Trajectory Models

One limitation of CMs is that they do not provide a satisfactory trade-off between accuracy and computational cost. For example, Song et al. (2023) found that using 3 Neural Function Evaluations (NFE) leads to the best sample quality when using consistency models. However, increasing the NFE beyond 3 does not continue to improve the quality of the samples. Kim et al. (2023) have demonstrated this limitation and proposed a new sampling method, γ-sampling, which leverages Consistency Trajectory Models (CTMs).

Stronger Consistency Functions

Recall that the consistency functions defined in Eq. (2.39) are functions that map any point on the same PF ODE trajectory to the origin of the PF ODE. Given the trajectory $\{x_t\}_{t\in[\varepsilon,T]}$, Kim et al. (2023) propose functions $G : (x_t, t, s) \to x_s$ with $s \leq t$ that satisfy stronger self-consistency properties defined as:

$$G(x_t, t, s) = G(x_{t'}, t', s), \quad \forall t, t' \in [\varepsilon, T] \text{ and } \forall s \in [\varepsilon, \min\{t, t'\}] \tag{2.42}$$

CTMs are functions G_θ that approximate G. Intuitively, a well-trained CTM can map any point on the same PF ODE trajectory to any point prior to it, offering greater flexibility than CMs. This flexibility is the key reason why CTMs can provide a better trade-off between sampling quality and computational cost, as we will discuss later.

Parameterization

Assume we use the same diffusion SDE and PF ODE as used by Karras et al. (2022); Song et al. (2023), shown in Eq. (2.25) and Eq. (2.26), respectively. Inspired by the Euler solver, Kim et al. (2023) propose the following parameterization of G, which is written as a mixture of x_t and an additional function g:

$$G(x_t,t,s) := x_t + \int_t^s \frac{x_u - \mathbb{E}[x|x_u]}{u} du = \frac{s}{t} x_t + \left(1 - \frac{s}{t}\right) g(x_t,t,s) \tag{2.43}$$

where $g(x_t,t,s) = x_t + \frac{t}{t-s} \int_t^s \frac{x_u - \mathbb{E}[x|x_u]}{u} du$. One benefit of using this parameterization is that in the limit of $s \to t$, we have:

$$\lim_{s \to t} g(x_t,t,s) = x_t + t \lim_{s \to t} \frac{1}{t-s} \int_t^s \frac{x_u - \mathbb{E}[x|x_u]}{u} du = \mathbb{E}[x|x_t] \tag{2.44}$$

which corresponds to the denoiser function. Therefore, estimating g not only captures the t-to-s-jump but also the denoiser function. In practice, g will be parameterized by g_θ, a neural network with a similar form to CMs, and is given as:

$$g_\theta(x_t,t,s) = c_{\text{skip}}(t) x_t + c_{\text{out}}(t) F_\theta(x_t,t,s) \tag{2.45}$$

where c_{skip} and c_{out} use the same preconditioning as in Karras et al. (2022). Then, the CTM model G_θ will be:

$$G_\theta(x_t,t,s) = \frac{s}{t} x_t + \left(1 - \frac{s}{t}\right) g_\theta(x_t,t,s) \tag{2.46}$$

Sampling from Consistency Trajectory Models

Due to the property of g_θ shown in Eq. (2.44), g_θ effectively models the score, thus supporting standard score-based sampling with ODE/SDE solvers. Additionally, Kim et al. (2023) propose a novel γ-sampling method for a sequence of sampling time steps $\varepsilon = t_0 < t_1 < \cdots < t_N = T$. The γ-sampling method proceeds as follows:

1. Denoise the sample x_{t_n} to some time $\tilde{t} < t_{n-1}$ using the CTM model G_θ.
2. Add noise according to the diffusion SDE to reach t_{n-1}.
3. Alternately perform steps 1 and 2 until reaching the final time point t_0[3].

The parameter $\gamma \in [0,1]$ controls the amount of noise added at each time step. Notably, when $\gamma = 0$, the entire process becomes deterministic (see Figure 2.3). Kim et al. (2023) found

[3] Our proposed method also draws inspiration from this sampling approach, see Figure 3.3.

that $\gamma = 0$ is the only setting that approximates the performance of the Heun's solver as the number of NFE increases.

Training Consistency Trajectory Models via Distillation

Unlike Consistency Models (CMs), the design choices for Consistency Trajectory Models (CTMs) in Kim et al. (2023) are suited only for consistency distillation and are not applicable for training in isolation[4]. Therefore, we focus on the scenario where a pre-trained diffusion model is available. Due to the distinct parameterization of CTMs compared to CMs, the training method also differs. The training loss for CTMs consists of two components: the soft consistency loss and an auxiliary loss to enhance training performance and facilitate the learning of the student model.

Soft Consistency Loss: Suppose we have an ODE solver, a pre-trained score model ϕ, a student model G_θ, and a teacher model G_{θ^-} where $\theta^- = \text{stopgrad}(\mu\theta^- + (1-\mu)\theta)$. Given a fixed time interval $[\varepsilon, T]$, we first sample a time point $t \in [\varepsilon, T]$ and then sample a time point $s \in [\varepsilon, t)$. We then sample another time point $u \in [s, t)$. Next, we sample $x_0 \sim p_{\text{data}}$ from the data distribution and add noise to x_0 to obtain x_t following the forward diffusion SDE.

For the teacher model, we use the ODE solver to move from t to u using the pre-trained score model. We then obtain the sample at time s using the teacher model, which is subsequently mapped to time $t_0 = \varepsilon$ to obtain x_{target}. The process is summarized as follows:

$$\tilde{x}_u = \text{Solver}(x_t, t, u; \phi) \tag{2.47}$$

$$\tilde{x}_s = G_{\theta^-}(\tilde{x}_u, u, s) \tag{2.48}$$

$$x_{\text{target}} = G_{\theta^-}(\tilde{x}_s, s, \varepsilon) \tag{2.49}$$

where $\varepsilon \leq s \leq u < t \leq T$.

For the student model, we directly map x_t from time t to time s and then map it again to time ε. The process is summarized as follows:

$$\tilde{x}_s = G_\theta(x_t, t, s) \tag{2.50}$$

$$x_{\text{est}} = G_\theta(\tilde{x}_s, s, \varepsilon) \tag{2.51}$$

[4] While the original paper mentioned a method for training CTM in isolation, our experiment results show that CTM's design is not suitable for training without a pre-trained diffusion model.

The loss function is defined as the distance between x_{target} and x_{est}:

$$\mathscr{L}_{\text{CTM}}(\theta;\phi) := \mathbb{E}_{t,s,u,x_0}\mathbb{E}_{x_t|x_0}\left[d(x_{\text{target}}, x_{\text{est}})\right] \tag{2.52}$$

Auxiliary Losses: In addition to the soft consistency loss, Kim et al. (2023) introduce two auxiliary losses to facilitate student learning: the Denoising Score Matching (DSM) loss and an adversarial loss. The DSM loss is used to enforce that $g_\theta(x_t,t,t)$ should act as a denoiser for any t. The DSM loss is given by:

$$\mathscr{L}_{\text{DSM}}(\theta) := \mathbb{E}_{x_0,t}\mathbb{E}_{x_t|x_0}\left[||x_0 - g_\theta(x_t,t,t)||_2^2\right] \tag{2.53}$$

In our experiments, we found that the adversarial loss did not significantly improve performance (it was mainly used to enhance image quality in the original paper), so we omit the adversarial loss and focus on the CTM and DSM losses.

Extending Consistency Trajectory Models to Bidirectional

In later experiments, inspired by Bidirectional Consistency Models (BCMs), which will be introduced in the next section, we found that it is not necessary to restrict s to be smaller than t. Instead, by relaxing this constraint and only requiring $s \neq t$, the CTM distilled from the pre-trained score model becomes a Bidirectional Consistency Trajectory Model (BCTM). This extension allows us to travel along the PF ODE trajectory not only backward in time but also forward in time using only 1 NFE. We will demonstrate the effectiveness of this extended model in the later experiment section.

2.3.3 Bidirectional Consistency Models

CTMs offer an improved balance between sample quality and computational efficiency through γ-sampling, allowing mapping from any point at time t to any earlier point at time $s < t$ along the PF ODE trajectory. However, despite the accelerated generation process in CMs and CTMs, one challenge remains in inversion tasks. For instance, a drawback of γ-sampling is that, while it enhances the quality of generated images, it can alter the content of the image. To unify the framework of generation and inversion, Li and He (2024) propose Bidirectional Consistency Models (BCMs), which learn a stronger, bidirectional consistency property that can map any point on the PF ODE trajectory to any other point, both forward and backward in time, providing ultimate flexibility.

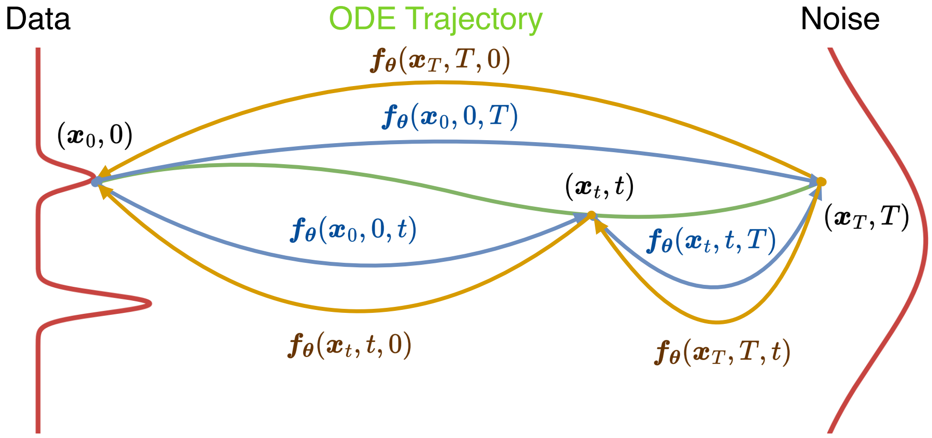


Fig. 2.4 Illustration of the sampling process of bidirectional consistency models. Bidirectional consistency models further improve upon CTMs by offering the ability to map any point on the sample PF ODE trajectory to any other point. This includes mapping in the denoising direction (highlighted in orange) and in the noising direction (highlighted in blue).

Bidirectional Consistency Functions

Suppose the solution trajectory is given by $\{x_t\}_{t\in[\varepsilon,T]}$. A function $f:(x_t,t,s)\to x_s$ satisfies the bidirectional consistency property if:

$$f(x_t,t,s) = f(x_{t'},t',s), \quad \forall t,t',s \in [\varepsilon,T] \tag{2.54}$$

BCMs, denoted as f_θ, are trained to approximate this function. For a well-trained function, any point on the same trajectory at any time can be mapped to any other time. In Chapter 3, we will explore how this property can be utilized in our method.

Parameterization

BCMs inherit the parameterization of CMs with slight modifications to the coefficients c_{skip} and c_{out}. The BCM is parameterized as:

$$f_\theta(x,t,s) = c_{\text{skip}}(t,s)x + c_{\text{out}}(t,s)F_\theta(x,t,s) \tag{2.55}$$

with c_{skip} and c_{out} defined as:

$$c_{\text{skip}}(t,s) = \frac{\sigma_{\text{data}}^2 + ts}{\sigma_{\text{data}}^2 + t^2}, \quad c_{\text{out}}(t,s) = \frac{\sigma_{\text{data}}(t-s)}{\sqrt{\sigma_{\text{data}}^2 + t^2}} \tag{2.56}$$

Notice that $c_{\text{skip}}(t,t) = 1$ and $c_{\text{out}}(t,t) = 0$, which enforce the boundary condition that $f_\theta(x_t,t,t) = x_t$.

Sampling from Bidirectional Consistency Models

Sampling from BCMs is similar to methods used in CTMs, but with a unique advantage. Exploiting the bidirectional consistency property of BCMs, Li and He (2024) introduce Zigzag sampling. This technique enhances sample quality using additional steps while preserving sample content. For a detailed algorithm, since this is not directly related to our work, we refer readers to the original paper by Li and He (2024).

Training Bidirectional Consistency Models in Isolation

The loss function for BCMs consists of two parts. The first part is the CT loss, similar to the one used in Eq. (A.3). The second part, specific to BCMs, is designed to train the model to follow the ODE trajectory both forward and backward in time. Suppose we have randomly sampled two time steps t and s such that $t \neq s$. We want to construct a model that learns the mapping from x_t to x_s. When $s < t$, the model learns to denoise, and when $s > t$, it learns to add noise. This dual capability allows BCMs to unify generative and inverse tasks within a single framework.

The training process is as follows: Given two sampled time points t and s, we first map x_t from time t to time s using the student model to obtain $x_s = f_\theta(x_t,t,s)$. Then, both x_s and x_t are mapped to time 0 using the teacher model to obtain x_{est} and x_{target}, respectively. The process is summarized as:

$$\tilde{x}_s = f_\theta(x_t,t,s) \tag{2.57}$$

$$x_{\text{est}} = f_{\theta^-}(\tilde{x}_s,s,\varepsilon) \tag{2.58}$$

$$x_{\text{target}} = f_{\theta^-}(x_t,t,\varepsilon) \tag{2.59}$$

The second loss function is then defined as:

$$\mathscr{L}^N_{\text{BCT}}(\theta) = \mathbb{E}_{z,x,t,s}\left[\lambda'(t,s)d(x_{\text{est}},x_{\text{target}})\right] \tag{2.60}$$

where the weight λ' is defined as $\lambda'(t,s) = \frac{1}{|t-s|}$. Thus, the total loss function used to train BCMs is given by $\mathscr{L}^N(\theta) = \mathscr{L}^N_{\text{BCT}}(\theta) + \mathscr{L}^N_{\text{CT}}(\theta)$, where $\mathscr{L}^N_{\text{CT}}(\theta)$ is given in Eq. (A.3).

This describes the training process for BCMs in isolation. However, in practice, training BCMs in isolation requires a very long time for the parameters to converge and may fail to converge if the parameters are poorly initialized (as observed in the experiments with water molecules). Therefore, it is desirable to train BCMs via distillation to accelerate the training process. Inspired by the distillation process described for CTMs, we can extend the training process to BCMs and train them via distillation, given a pre-trained score-based diffusion model. To train the model, we replace the BCT loss with the CTM loss shown in Eq. (2.52). However, we found that this approach does not achieve the same performance as BCTMs. We leave further improvements as future work.

2.3.4 Summary of Family of Consistency Models

We summarize the properties of the family of consistency models in Table 2.1.

Table 2.1 Summary of Consistency Model Family Properties.

Property	**CM**	**BCM**	**CTM**	**BCTM**
Bidirectional	No	Yes	No	Yes
Training Method	CT/CD	CT	CD	CD
Time Constraint $f_\theta(x_t, t, s)$	$s = \varepsilon$	$s \neq t$	$s < t$	$s \neq t$
Special Sampling	–	Zigzag	γ-sampling	Zigzag or γ-sampling
Pre-trained Model Required	No*	No	Yes	Yes

* Not required for CT, required for CD.

2.4 Chapter Summary

In this chapter, we have provided a comprehensive review of three key areas: importance sampling, score-based diffusion models, and the family of consistency models. These concepts form the foundation for our subsequent work. In the next chapter, we will build upon this knowledge to formulate a baseline method that integrates importance sampling with score-based diffusion models. Furthermore, we will introduce our novel approach, which combines importance sampling with bidirectional consistency (trajectory) models. This novel method aims to leverage the strengths of both importance sampling and the latest advancements in generative modeling to achieve improved performance and efficiency.

Chapter 3

Importance Sampling with Consistency Models

In this chapter, we introduce the application of IS to correct biases in the sample distributions generated by diffusion models or consistency models. The chapter is divided into two main sections. The first section presents a baseline approach that integrates importance sampling with score-based diffusion models in a straightforward manner. Although this baseline method is theoretically sound, it has several practical limitations that hinder its feasibility for real-world deployment. The second section introduces our novel approach, which aims to achieve unbiased samples more efficiently than the baseline by leveraging the family of consistency models.

Throughout this chapter and in subsequent experiments, we adopt the settings used by Karras et al. (2022), where the diffusion SDE is given by Eq. (2.25) and the corresponding PF ODE by Eq. (2.26).

3.1 Importance Sampling with Diffusion Models

In this section, we formalize the application of IS within the context of diffusion models. To establish a valid IS framework, we must define both the target and proposal distributions. Broadly speaking, the target distribution we aim to sample from corresponds to the entire diffusion process governed by molecular energy functions, which can be evaluated but not directly sampled. The proposal distribution, conversely, corresponds to the denoising process of diffusion models, as defined by either DDPM or DDIM. Figure 3.1 provides an intuitive

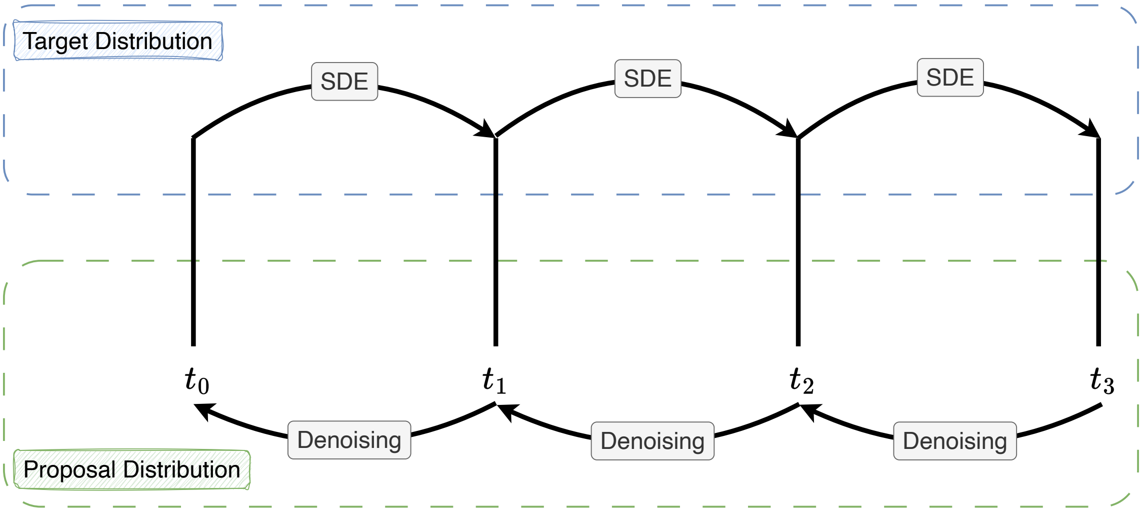


Fig. 3.1 Illustration of the target and proposal distributions for the baseline method. The target distribution is defined by the diffusion SDE for each time interval, while the proposal distribution is defined by the DDPM denoising distribution.

representation of this concept. We will elaborate on these distributions in the following subsections.

3.1.1 Importance Sampling Formulation

Consider the task of estimating an integral as described in Section 2.1. The target distribution from which we want to sample is denoted by π (e.g., molecular energy functions). In practice, we can only evaluate the unnormalized molecular energy functions, which we denote as $\bar{\pi}$. Our goal is to estimate $\mathbb{E}_{x\sim\pi}[\phi(x)]$ for some function ϕ.

Assume we have access to a well-trained diffusion model that allows us to draw samples and evaluate the log-likelihood, which serves as the proposal distribution. By employing Self-Normalized Importance Sampling (SNIS), we can effectively combine these elements to correct the bias (asymptotically, as the number of samples increases) in the sample distribution generated by the diffusion model.

To be specific, consider discretizing the time interval $[\varepsilon, T]$ as $\varepsilon = t_0 < t_1 < \cdots < t_N = T$. Here, $t_0 = \varepsilon$ is chosen so that the difference between the distribution $\pi(x_{t_0})$ and the true distribution $\pi(x)$ is negligible. Assume the proposal distribution of the diffusion model is $p_\theta(x_{t_{n-1}}|x_{t_n})$, and the noise distribution is $q(x_{t_n}|x_{t_{n-1}})$ for $n = 1, \ldots, N$. The expectation can then be written as:

$$\begin{aligned}
\mathbb{E}_{x\sim\pi}[\phi(x)] &\approx \mathbb{E}_{x_{t_0}\sim\pi}[\phi(x_{t_0})] && (3.1)\\
&= \int \phi(x_{t_0})\pi(x_{t_0})dx_{t_0} && (3.2)\\
&= \int \phi(x_{t_0})\pi(x_{t_0})\left(\prod_{n=1}^{N} q(x_{t_n}|x_{t_{n-1}})\right)dx_{t_{0:N}} && (3.3)\\
&= \int \phi(x_{t_0})\left(\frac{\pi(x_{t_0})\prod_{n=1}^{N} q(x_{t_n}|x_{t_{n-1}})}{p_\theta(x_N)\prod_{n=1}^{N} p_\theta(x_{t_{n-1}}|x_{t_n})}\right)p_\theta(x_{t_{0:N}})dx_{t_{0:N}} && (3.4)\\
&\approx \frac{1}{K}\sum_{k=1}^{K}\left(\frac{\pi(x_{t_0}^{(k)})\prod_{n=1}^{N} q(x_{t_n}^{(k)}|x_{t_{n-1}}^{(k)})}{p_\theta(x_N^{(k)})\prod_{n=1}^{N} p_\theta(x_{t_{n-1}}^{(k)}|x_{t_n}^{(k)})}\right)\phi(x_{t_0}) && (3.5)\\
&= \frac{1}{K}\sum_{k=1}^{K} w_k\phi(x_{t_0}) && (3.6)
\end{aligned}$$

where $x_{t_{0:N}}^{(k)}$ for $k = 1,\dots,K$ are samples from the diffusion model p_θ. The importance weights $\{w_k\}_{k=1}^{K}$ are defined as:

$$w_k = \frac{\pi(x_{t_0}^{(k)})\prod_{n=1}^{N} q(x_{t_n}^{(k)}|x_{t_{n-1}}^{(k)})}{p(x_N^{(k)})\prod_{n=1}^{N} p_\theta(x_{t_{n-1}}^{(k)}|x_{t_n}^{(k)})} \tag{3.7}$$

If we only have access to the unnormalized target distribution $\bar{\pi}$, according to Section 2.1, the estimation can be obtained using SNIS:

$$\mathbb{E}_{x\sim\bar{\pi}}[\phi(x)] \approx \sum_{k=1}^{K} \bar{w}_k\phi(x_{t_0}) \tag{3.8}$$

where $\bar{w}_k = w_k/\sum_{k=1}^{K} w_k$ for $k = 1,\dots,K$, and each w_k is computed using $\bar{\pi}$. So far, we have established a general framework for applying IS using diffusion models. However, we still need to specify the proposal distribution p_θ and target distribution q.

3.1.2 Noise and Denoising Distributions

Next, we define the noise and denoising distributions. According to the diffusion SDE defined in Eq. (2.25), the noise distribution is modeled as a Gaussian distribution:

$$q(x_{t_n}|x_{t_{n-1}}) = \mathcal{N}(x_{t_n}; x_{t_{n-1}}, (t_n^2 - t_{n-1}^2)I) \tag{3.9}$$

For the denoising distribution, we can define it in a manner similar to DDIM (Song et al., 2020a):

$$q_\sigma(x_{t_{n-1}}|x_{t_{n-1}},x_0) = \mathcal{N}\left(x_{t_{n-1}};x_{t_n}\sqrt{\frac{t_{n-1}^2-\sigma_{n-1}^2}{t_n^2}}+x_0\left(1-\sqrt{\frac{t_{n-1}^2-\sigma_{n-1}^2}{t_n^2}}\right),\sigma_{n-1}^2 I\right) \tag{3.10}$$

We define σ_{n-1} as follows:

$$\sigma_{n-1} = \eta\sqrt{\frac{(t_n^2-t_{n-1}^2)t_{n-1}^2}{t_n^2}} \tag{3.11}$$

where $\eta \in [0,1]$. When $\eta = 1$, this corresponds to the denoising distribution of DDPM (Ho et al., 2020), and when $\eta = 0$, the sampling becomes deterministic, corresponding to Euler's first method for solving the PF ODE in Eq. (2.26). Our empirical results indicate that $\eta = 1$ consistently yields the highest ESS and lowest IS variance. Therefore, we use the denoising distribution same to that in DDPM.

In practice, since x_0 is unknown, we replace it with the model prediction $\hat{x}_0$. Therefore, the denoising distribution is defined as:

$$p_\theta(x_{t_{n-1}}|x_{t_n}) = q_\sigma(x_{t_{n-1}}|x_{t_{n-1}},\hat{x}_0) = \mathcal{N}\left(x_{t_{n-1}};\frac{t_{n-1}^2}{t_n^2}x_{t_n}+\left(1-\frac{t_{n-1}^2}{t_n^2}\right)\hat{x}_0,\frac{(t_n^2-t_{n-1}^2)t_{n-1}^2}{t_n^2}I\right) \tag{3.12}$$

where $\hat{x}_0 = D_\phi(x_{t_n},t_n)$ is computed by the trained denoiser model. With the definitions for the proposal and target distributions established, we can compute the importance weights according to Eq. (3.5) and use these weights for integral estimation tasks. However, one remaining aspect that we have not yet specified is the selection of the time steps $\{t_n\}$.

3.1.3 Setting Time Steps

In our experiments, we adopt the same parameterization, preconditioning, and training methods as introduced by Karras et al. (2022). Consequently, it is natural to consider the time schedule specified by them, as stated in Eq. (2.29) with $\rho = 7$. However, during subsequent experiments, we observed that increasing the value of ρ results in a higher ESS. When $\rho \to \infty$, the time schedule approaches:

$$t_i = \varepsilon\left(\frac{T}{\varepsilon}\right)^{\frac{i-1}{N-1}} \tag{3.13}$$

This configuration is equivalent to arranging the time steps evenly in logarithmic space. We found that this approach yields the highest ESS across various target distributions when using the baseline method. Therefore, we employ this time schedule in all subsequent experiments related to the baseline method.

3.1.4 Strengths and Limitations

The baseline approach combining importance sampling with score-based diffusion models exhibits both strengths and limitations.

Strengths: The method is grounded in sound theoretical foundations and is straightforward to implement, requiring no modifications to a trained diffusion model. With a sufficient number of steps, it can achieve a high ESS, indicating excellent alignment between the proposal and target distributions. This results in low-variance importance sampling estimates, even in high-dimensional spaces.

Limitations: The primary weakness of the baseline approach lies in the number of steps required to achieve low-variance estimates. Our experiments reveal that even in one-dimensional space, approximately 100 steps may be necessary to obtain reasonable integral estimates. For higher-dimensional spaces, the required number of time steps increases exponentially, quickly becoming computationally infeasible.

This efficiency bottleneck severely limits the practical applicability of the baseline approach. Consequently, it motivates our search for an improved method that can apply importance sampling more efficiently, with significantly fewer steps.

3.2 Importance Sampling with Consistency Models

In this section, we introduce our improved importance sampling method. We summarize our motivation in Figure 3.2. The baseline method discussed previously is constrained by the necessity of a large number of time steps to achieve low variance in the IS estimate. Consistency models, which can efficiently traverse along the PF ODE, offer the potential to reduce the number of steps required for accurate sampling. However, one significant challenge is that once the initial point is fixed, ODEs are deterministic and, therefore, do not define a valid proposal distribution in the same way as the DDPM denoising distribution does.

To overcome this, we propose reintroducing stochasticity through an SDE component. For sampling from the proposal distribution, our method alternates between two steps: (1) moving

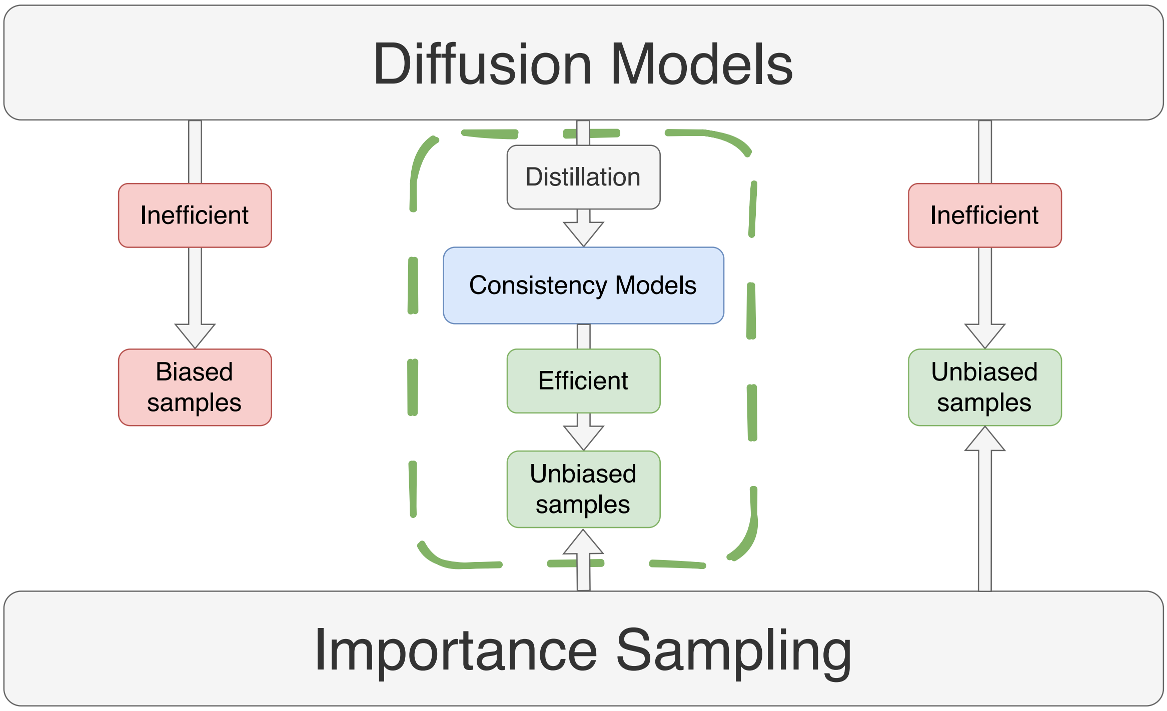


Fig. 3.2 Motivation for our proposed algorithm. We aim to carefully combine Importance Sampling and Consistency Models to efficiently obtain unbiased samples.

from a larger time to a smaller time via the ODE trajectory, and (2) moving forward in time following the diffusion SDE, adding Gaussian noise. This process continues until reaching the minimum time point[1].

A similar alternating ODE-SDE process is applied to the target distribution, enhancing its flexibility to better match the proposal distribution. Crucially, BCMs or BCTMs enable both forward and backward movement along the ODE trajectory. This bidirectional capability allows mapping between ODE trajectories using only one function evaluation (1 NFE), significantly improving computational efficiency.

This section is organized into several subsections. The first two subsections provide detailed and rigorous formulations regarding the design of the proposal and target distributions. In the subsequent subsection, we introduce the parameters involved in our method and discuss how to tune these parameters to achieve the lowest IS variance. Finally, we present the complete algorithm, with its effectiveness demonstrated in the next chapter.

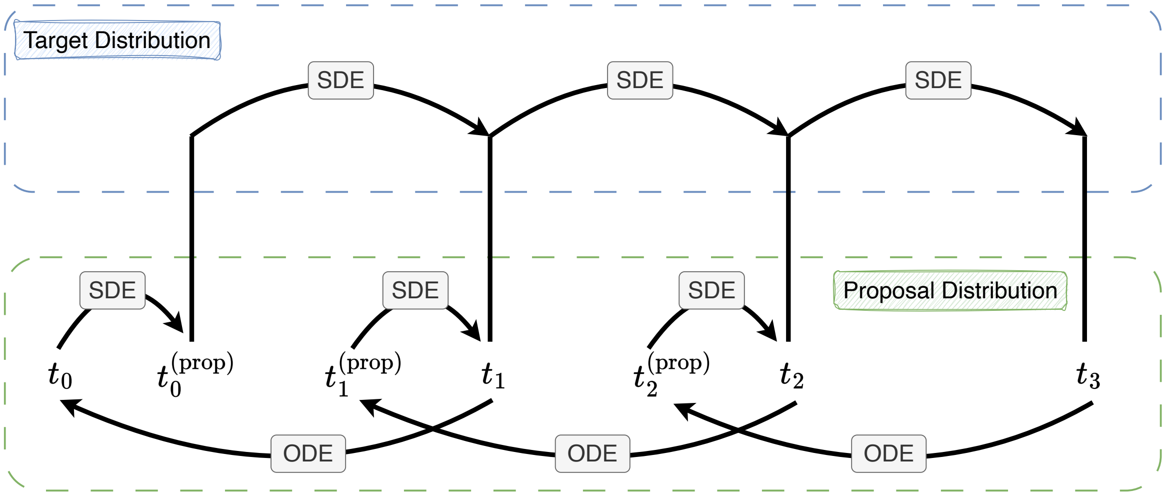


Fig. 3.3 Defining the proposal distribution. Starting with x_{t_3} at time t_3, we: (1) Map to $x_{t_2^{(\text{prop})}}$ at time $t_2^{(\text{prop})}$ via a deterministic ODE trajectory using Bidirectional Consistency (Trajectory) Models: $x_{t_2^{(\text{prop})}} = f_\theta(x_{t_3}, t_3, t_2^{(\text{prop})})$. (2) Add Gaussian noise to obtain $x_{t_2} \sim \mathcal{N}\left(x_{t_2^{(\text{prop})}}, \left(t_2^2 - \left(t_2^{(\text{prop})}\right)^2\right) I\right)$. This process is repeated for other time steps.

3.2.1 Designing the Proposal Distribution Using ODEs and SDEs

We begin by using the same notation for time steps as in the previous section: $\varepsilon = t_0 < t_1 < \cdots < t_N = T$. Note that the base distribution remains unchanged, with $p_\theta(x_{t_N}) = \mathcal{N}(x_{t_N}; 0, t_N^2 I)$. Our task is to design the proposal distribution $p_\theta(x_{t_{n-1}} | x_{t_n})$ for $n = 1, \ldots, N$. Our key goal is to leverage the ODE to minimize time steps in IS while forming a valid proposal distribution. Therefore, we define the proposal distribution as:

$$p_\theta(x_{t_{n-1}} | x_{t_n}) = \mathcal{N}\left(x_{t_{n-1}}; f_\theta\left(x_{t_n}, t_n, t_{n-1}^{(\text{prop})}\right), \left(t_{n-1}^2 - \left(t_{n-1}^{(\text{prop})}\right)^2\right) I\right) \tag{3.14}$$

where $\varepsilon \leq t_{n-1}^{(\text{prop})} < t_{n-1}$ for $n = 2, \ldots, N$, and f_θ represents the model that moves x_{t_n} at time t_n to time $t_{n-1}^{(\text{prop})}$ following the ODE trajectory. Then we follow the diffusion SDE to move forward in time to t_{n-1} by adding noise to form the proposal distribution as defined above.

It is important to carefully define the proposal distribution $p_\theta(x_{t_0} | x_{t_1})$. Since $t_0 = \varepsilon$ is the minimum time step, we cannot move further back in time and then add noise. In this case, we first move back to time t_0 and then add some noise to a time point $t_0^{(\text{prop})} > t_0$. If this noise is small, then $x_{t_0^{(\text{prop})}}$ should still lie in the high-probability region of π, and thus it will

[1] This process is similar to the γ-sampling method introduced in Section 2.3.2.

not significantly affect the IS process. The algorithm will then return $x_{t_0^{(\text{prop})}}$. Therefore, the proposal distribution at t_0 is given as:

$$p_\theta\left(x_{t_0^{(\text{prop})}}|x_{t_1}\right) = \mathcal{N}\left(x_{t_0^{(\text{prop})}}; f_\theta(x_{t_1}, t_1, t_0), \left(\left(t_0^{(\text{prop})}\right)^2 - t_0^2\right) I\right) \tag{3.15}$$

With this, we have defined all the proposal distributions. The visualization for the proposal distributions is shown in Figure 3.3. The function f_θ can be parameterized by BCMs or BCTMs, enabling mapping between different times on the same ODE trajectory using only 1 NFE, which is the key reason why our proposed method is efficient.

3.2.2 Designing the Target Distribution Using ODEs and SDEs

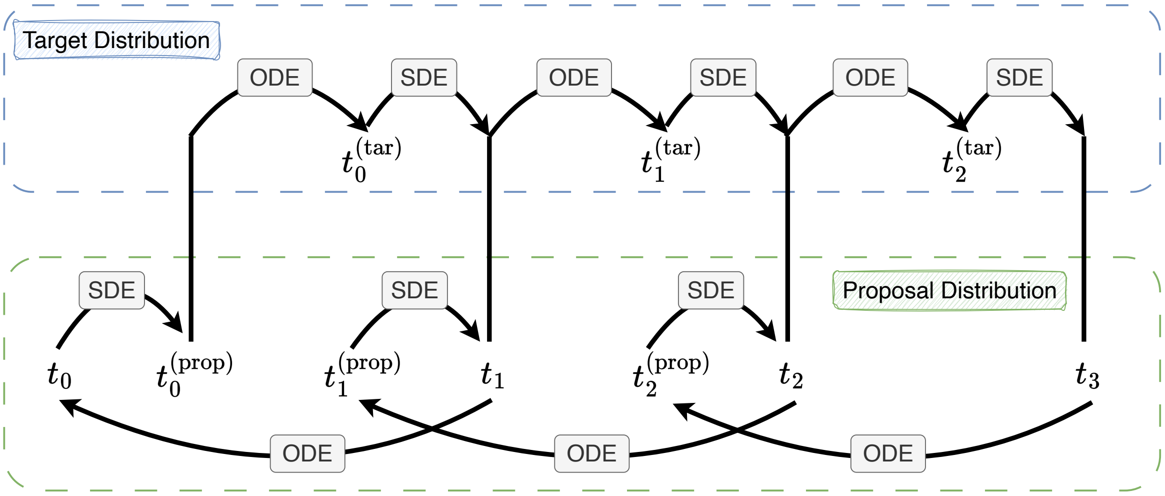


Fig. 3.4 Defining the target distribution using ODE-SDE combination. For example, starting with x_{t_2} at time t_2: (1) Map to $x_{t_2^{(\text{tar})}}$ at time $t_2^{(\text{tar})}$ via PF ODE: $x_{t_2^{(\text{tar})}} = f_\theta(x_{t_2}, t_2, t_2^{(\text{tar})})$. (2) Add Gaussian noise: $x_{t_3} \sim \mathcal{N}\left(x_{t_2^{(\text{tar})}}, \left(t_3^2 - \left(t_2^{(\text{tar})}\right)^2\right) I\right)$. This ODE-SDE approach, applied similarly at other time steps, offers greater flexibility in matching the proposal distribution.

Up to this point, we have assumed that the target distributions are defined by the diffusion SDE, given by:

$$q(x_{t_n}|x_{t_{n-1}}) = \mathcal{N}(x_{t_n}; x_{t_{n-1}}, (t_n^2 - t_{n-1}^2)I) \tag{3.16}$$

However, the target distribution offers some flexibility that we can leverage. Similar to the design of the proposal distribution, the target distribution can also be defined using ODEs and SDEs. Intuitively, given a sample at time t_n, we can map this sample forward in time along the ODE trajectory to some point $t_n^{(\text{tar})}$, and this target time point can be chosen such that the target distribution aligns better with the proposal distribution. In this case, the target distribution is defined as:

$$q_\theta(x_{t_n}|x_{t_{n-1}}) = \mathcal{N}\left(x_{t_n}; f_\theta\left(x_{t_{n-1}}, t_{n-1}, t_{n-1}^{(\text{tar})}\right), \left(t_n^2 - \left(t_{n-1}^{(\text{tar})}\right)^2\right) I\right) \tag{3.17}$$

for $n = 1, \ldots, N$, where $t_{n-1} \leq t_{n-1}^{(\text{tar})} < t_n$ and f_θ is a function that maps the sample $x_{t_{n-1}}$ forward in time along the PF ODE from time t_{n-1} to time $t_{n-1}^{(\text{tar})}$. Then, noise is added to $x_{t_{n-1}^{(\text{tar})}}$ according to the diffusion SDE, defining the distribution q_θ. Similar to the design of the proposal distribution, we must carefully handle the case for $n = 1$. The target distribution is defined as:

$$q_\theta\left(x_{t_1}|x_{t_0^{(\text{prop})}}\right) = \mathcal{N}\left(x_{t_1}; f_\theta\left(x_{t_0^{(\text{prop})}}, t_0^{(\text{prop})}, t_0^{(\text{tar})}\right), \left(t_1^2 - \left(t_0^{(\text{tar})}\right)^2\right) I\right) \tag{3.18}$$

Notably, BCMs or BCTMs can naturally serve as f_θ in this context, as they have the capability to map points on the same ODE trajectory to any other points both forward and backward in time using only one NFE. This ability significantly enhances the efficiency of the IS process by reducing the computational overhead.

So far, we have formalized a novel approach to performing IS using BCMs or BCTMs. However, there remain several important considerations. Specifically, how should we choose the time points t_n, $t_n^{(\text{tar})}$, and $t_n^{(\text{prop})}$ for each n?

3.2.3 Tuning the Time Steps

Recall that the goal in setting the time steps is to minimize the variance of the IS estimator. Therefore, the key to optimizing our algorithm lies in setting the time steps by minimizing certain objective functions (which we will introduce later) designed to reduce IS variance. To achieve this, we need to introduce a suitable parameterization for the time steps, and we will define three sets of parameters that determine these time steps. For a time interval $[\varepsilon, T]$, we reparameterize the three sets of time points as follows:

$$t_n = \begin{cases} T, & n = N \\ \mu_n(t_{n+1} - \varepsilon) + \varepsilon, & n = 2, \ldots, N-1 \\ \varepsilon, & n = 1 \end{cases} \tag{3.19}$$

$$t_n^{(\text{tar})} = (t_{n+1} - t_n)\eta_n + t_n, \quad n = 1, \ldots, N-1 \tag{3.20}$$

$$t_n^{(\text{prop})} = \begin{cases} \gamma_n(t_n - \varepsilon) + \varepsilon, & n = 2, \ldots, N-1 \\ \gamma_n(t_n^{(\text{tar})} - \varepsilon) + \varepsilon, & n = 1 \end{cases} \tag{3.21}$$

Note that in Eq. (3.19), the time steps are defined sequentially, with each time step at n falling between the minimum time point ε and the previous time point t_{n+1}. A similar approach is used in Eq. (3.20) to define the target time steps, where each target time point is positioned between the time points t_n and t_{n+1}. For the proposal time points in Eq. (3.21), each proposal time point is positioned between t_n and the minimal time point ε, except for the last proposal time point $n = 1$, which lies between $t_1^{(\text{tar})}$ and ε.

However, in later experiments, we found that the optimal proposal time points are best defined to ensure that the proposal variance equals the target variance whenever possible:

$$t_n^{(\text{prop})} = \begin{cases} \sqrt{\max\{t_n^2 + \left(t_n^{(\text{tar})}\right)^2 - t_{n+1}^2, \varepsilon^2\}}, & n = 2, \ldots, N-1 \\ \min\{\sqrt{\max\{t_{n+1}^2 - \left(t_n^{(\text{tar})}\right)^2 + t_n^2, \varepsilon^2\}}, t_n^{(\text{tar})}\}, & n = 1 \end{cases} \tag{3.22}$$

To ensure that the variance matches, we aim to define the proposal time points such that $t_n^2 - \left(t_{n-1}^{(\text{tar})}\right)^2 = t_{n-1}^2 - \left(t_{n-1}^{(\text{prop})}\right)^2$. Additionally, we enforce the constraint that the proposal time points must be greater than the minimal time point ε. The case for $n = 1$ differs slightly because of the way that proposal distribution defined for $n = 1$ (see Eq. (3.14)). Here, we apply the constraints that the last proposal time point lies between the minimal time ε and $t_n^{(\text{tar})}$. Notice that $\{\mu_n\}$ and $\{\eta_n\}$ are parameters within the range $[0, 1]$. During optimization, we can apply a sigmoid function to enforce this constraint and optimize the parameters to find the optimal values.

Now we introduce three possible objective functions that can be used to tune the time steps:

- **ESS estimated using samples from the proposal ($\widehat{\text{ESS}}^{\text{prop}}$):** Setting the parameters to maximize the $\widehat{\text{ESS}}^{\text{prop}}$ seems like an obvious choice since $\widehat{\text{ESS}}^{\text{prop}}$ is widely used as a metric to measure the variance of the IS estimator. The formula for computing is reviewed in Eq. (2.3). However, as we discussed before in Section 2.1, $\widehat{\mathrm{ESS}}^{\mathrm{prop}}$ is not a perfect metric for measuring variance. It is possible to have large error in the IS estimator even with a large $\widehat{\mathrm{ESS}}^{\mathrm{prop}}$, particularly when the proposal distribution only partially covers the target distribution. In later experiments, we found that optimizing parameters with respect to $\widehat{\mathrm{ESS}}^{\mathrm{prop}}$ only works well for very simple target distributions. For more complex target distributions in high-dimensional space, optimizing ESS can lead to integral estimations with large error.

- **ESS estimated using samples from the target ($\widehat{\mathbf{ESS}}^{\mathbf{tar}}$):** One reason for the limitations of optimizing ESS using samples from the proposal is that the ESS estimate itself may be biased. This bias can result in a high ESS but still lead to large variance in the integral estimation. This motivates the search for a better ESS estimation. Recall that we reviewed an alternative method for estimating ESS using samples from the target distribution, as shown in Eq. (2.4). The advantage of this method is that it can provide a more accurate ESS estimate when the proposal distribution does not fully cover the target distribution. In later experiments, it was demonstrated be a better metric than $\widehat{\mathrm{ESS}}^{\mathrm{prop}}$ and gradient-based approachs can be applied to optimize parameters.

- **Forward KL divergence:** Another potential method for tuning the parameters is to optimize the forward Kullback-Leibler (KL) divergence between the target distribution and the proposal distribution. It is known that by minimizing the forward KL divergence, the proposal distribution tends to be mean-seeking, meaning it is encouraged to cover the entire target distribution. In contrast, minimizing the reverse KL divergence encourages the proposal distribution to be mode-seeking, focusing on certain modes of the target distribution. For IS problems, we propose minimizing the forward KL divergence between the target and proposal distributions. Given the proposal p_θ and target q_θ, the forward KL is defined as:

$$D_{KL}(q_\theta || p_\theta) = \int q_\theta(x_{t_{0:N}}) \log\left(\frac{q_\theta(x_{t_{0:N}})}{p_\theta(x_{t_{0:N}})}\right) dx_{t_{0:N}} \approx \frac{1}{K}\sum_{k=1}^{K} \log\left(\frac{q_\theta(x^{(k)}_{t_{0:N}})}{p_\theta(x^{(k)}_{t_{0:N}})}\right) \quad (3.23)$$

where $x^{(k)}_{t_{0:N}} \sim q_\theta$ are samples from the target distribution. We propose using Eq. (3.23) as the objective function, with the time points $\{t_n^{(\mathrm{prop})}\}_{n=1}^{N-1}$, $\{t_n^{(\mathrm{tar})}\}_{n=1}^{N-1}$, and $\{t_n\}_{n=1}^{N}$ as the parameters to tune. In later experiments, it was demonstrated that a gradient-based approach can also be effectively applied using the forward KL metric.

3.2.4 Putting Them Together

After tuning the time points according to the above algorithm, we can now perform IS with the designed proposal and target distributions. We summarize our proposed IS algorithm in Algorithm 1.

Algorithm 1 Importance Sampling with Bidirectional Consistency Models

```
Require: Time steps $\{t_n\}_{n=1}^{N}$, proposal time steps $\{t_n^{(\text{prop})}\}_{n=1}^{N-1}$, target time steps $\{t_n^{(\text{tar})}\}_{n=1}^{N-1}$,
    Bidirectional Consistency Model $f_\theta$, target distribution $\bar{\pi}$, number of samples $K$
Ensure: Samples $\{x_0^{(k)}\}_{k=1}^{K}$
for each $k$ from 1 to $K$ do
    Sample $x_{t_N}^{(k)} \sim \mathcal{N}\left(0, t_N^2 I\right)$
    for each $n$ from $N$ to 2 do
        Compute $x_{t_{n-1}^{(\text{prop})}}^{(k)} \leftarrow f_\theta\left(x_{t_n}^{(k)}, t_n, t_{n-1}^{(\text{prop})}\right)$
        Sample $x_{t_{n-1}}^{(k)} \sim \mathcal{N}\left(x_{t_{n-1}^{(\text{prop})}}^{(k)}, \left(t_{n-1}^2 - \left(t_{n-1}^{(\text{prop})}\right)^2\right) I\right)$
        Compute $x_{t_{n-1}^{(\text{tar})}}^{(k)} \leftarrow f_\theta\left(x_{t_{n-1}}^{(k)}, t_{n-1}, t_{n-1}^{(\text{tar})}\right)$
    end for
    Compute $x_{t_1}^{(k)} \leftarrow f_\theta\left(x_{t_2}^{(k)}, t_2, t_1\right)$
    Sample $x_{t_1^{(\text{prop})}}^{(k)} \sim \mathcal{N}\left(x_{t_1}^{(k)}, \left(\left(t_1^{(\text{prop})}\right)^2 - t_1^2\right) I\right)$
    Compute $x_{t_1^{(\text{tar})}}^{(k)} \leftarrow f_\theta\left(x_{t_1}^{(k)}, t_1^{(\text{prop})}, t_1^{(\text{tar})}\right)$
    Set $x_0^{(k)} \leftarrow x_{t_1^{(\text{prop})}}^{(k)}$ {Return samples at time $t_1^{\text{prop}}$}
end for
return $\{x_0^{(k)}\}_{k=1}^{K}$
```

Notice that the importance weights w_k for $k = 1, \ldots, K$ can be computed as:

$$w_k = \frac{\bar{\pi}(x_0^{(k)}) \prod_{n=1}^{N} \mathcal{N}\left(x_{t_n}^{(k)}; x_{t_n^{(\text{tar})}}^{(k)}, \left(t_n^2 - \left(t_n^{(\text{tar})}\right)^2\right) I\right)}{\mathcal{N}\left(x_{t_1^{(\text{prop})}}^{(k)}; x_{t_1}^{(k)}, \left(\left(t_1^{(\text{prop})}\right)^2 - t_1^2\right) I\right) \prod_{n=2}^{N} \mathcal{N}\left(x_{t_n}^{(k)}; x_{t_n^{(\text{prop})}}^{(k)}, \left(t_n^2 - \left(t_n^{(\text{prop})}\right)^2\right) I\right)} \tag{3.24}$$

Similar as before, the normalized importance weights can be computed using $\bar{w}_k = w_k / \sum_{k=1}^{K} w_k$. The samples $\{x^{(k)}\}_{k=1}^{K}$, along with the normalized importance weights $\{\bar{w}_k\}_{k=1}^{K}$, can be used to apply IS for integral estimation tasks.

3.3 Chapter Summary

In this chapter, we introduced the mathematical formulation for the baseline method, which combines IS with DDPM, as well as the formulation for our proposed method, which combines IS with BCMs or BCTMs. Our approach involves redesigning the proposal and target distributions to offer better flexibility in matching each other, enabled by the introduction of additional parameters. We also discussed optimization metrics that can be used to tune these parameters effectively. With the theoretical foundations established in this chapter, we will proceed to the experimental validation in the next chapter, where we will test both the baseline method and our proposed method and compare their performances.

Chapter 4

Experiments and Results

In the previous chapter, we introduced the baseline method—importance sampling with DDPM—and our novel approach, which employs importance sampling with BCM or BCTM. In this chapter, we present experiments designed to evaluate and compare both methods. The primary objective is to assess the performance of our proposed method against the baseline in both controlled and practical settings. This chapter is organized into two sections:

1. In the first section, we experiment with a dataset generated by Gaussian Mixture Models (GMMs). A significant advantage of GMMs is that the score function can be derived analytically. Consequently, we first employ the analytical score to solve the ODE trajectory. This scenario is equivalent to having a perfect BCM or BCTM. The primary aim of using the analytical score in these experiments is to verify the plausibility of our method without considering model errors. Subsequently, we train the score model, BCM, and BCTM, and then evaluate and compare the performance of importance sampling with DDPM using the score model against our method utilizing BCM or BCTM.

2. In the second section, we transition to a more practical setting, employing a real-world dataset that simulates the dynamics of supercooled water molecules. This dataset comprises samples representing atomic configurations constrained within a simulation box. We will rigorously compare the performance of both our baseline and proposed algorithms on this challenging task.

4.1 Synthetic Dataset Using Gaussian Mixture Models

In this section, we explore the use of Gaussian Mixture Models (GMMs) to test our proposed method. A significant advantage of GMMs is that the score function can be derived analytically (shown in Appendix A.2). In subsequent subsections, we will utilize samples generated by GMMs to train score-based diffusion models and the family of consistency models. The goal of this section is to compare our method with the baseline in a relatively simple setting before applying it to more complex, practical problems involving molecular datasets.

4.1.1 Experiments with Analytical Score

In this experiment, we use the analytical score function to address the following key questions:

- Without considering errors in the score model or the BCM model, can our method reduce the number of steps required to achieve the same percentage of ESS as the baseline DDPM model? Additionally, how many steps can our method reduce in comparison?

The primary goal of this subsection is to answer these questions and demonstrate the effectiveness of our method in settings where visualization is possible. In the next subsection, we will apply our approach to trained models and scale up the method to higher-dimensional spaces, continuing to work within the context of data generated by GMMs.

Simple GMM with Two Components

We evaluate our method using the following GMM as the target distribution:

$$\pi(x) = \sum_{m=1}^{M} \omega_m \mathcal{N}(x; \mu_m, \sigma_m^2 I), \tag{4.1}$$

where $M = 2$, $\omega_1 = 2/3$, $\omega_2 = 1/3$, and $\sigma_1 = \sigma_2 = 0.15$. The mean vectors are $\mu_1 = \mathbf{1}$ and $\mu_2 = -2\mathbf{1}$, where $\mathbf{1}$ is a vector of ones with d dimensions. We explore dimensions $d \in \{1,2,3,5\}$.

Comparing ESS

As described earlier, our method uses ESS estimates or the forward KL divergence as the loss function to optimize the time points. However, when using the analytical score, the ODE trajectory must be solved using an ODE solver (e.g., Euler's method). During backpropagation, the gradient cannot propagate properly in this scenario. Therefore, we employ a greedy-based grid search to find the optimal values for the parameters $\{\eta_n\}$, $\{\gamma_n\}$, and $\{\mu_n\}$ that were specified earlier. The procedure involves fixing two of the parameter sets while performing a grid search for the third parameter over values in the range $[0,1]$. This process is repeated for all parameters, and the optimal set is selected based on the lowest loss.

It should be noted that this greedy grid-search algorithm may not perform as effectively as gradient-based optimization, particularly in higher dimensions, where parameter tuning becomes more challenging. Consequently, we limit our experiments to lower-dimensional spaces for now. In later sections, where trained BCMs or BCTMs are used, we will demonstrate that a gradient-based approach to optimizing the time steps is also feasible.

During our experiments, we found that the optimal $\{\gamma_n\}$ are always the values such that the proposal variance matches the target variance for each time interval. Therefore, we adopt the parameterization for the proposal time points as previously shown in Eq. (3.22). Consequently, during optimization, we only have two sets of parameters $\{\eta_n\}$ and $\{\mu_n\}$ to tune. We optimize the parameters of our algorithms using three metrics specified earlier, i.e., $\widehat{\text{ESS}}^{\text{prop}}$, $\widehat{\text{ESS}}^{\text{tar}}$, and forward KL divergence. We then use the optimal parameters found to estimate ESS using samples from the proposal distribution, which serves as a preliminary assessment of the IS algorithm's effectiveness. Further integral estimation tasks will be conducted to evaluate the method's performance more thoroughly.

For our experiments, we use 1 million samples to estimate the metric. For each tuning metric, we tune the parameters five times and evaluate the ESS for each set of parameters across a number of steps ranging from 3 to 10. We then visualize the mean ESS along with the 25% and 75% quantiles for the ESS across the five experiments. For the baseline method, since there are no parameters to tune, we evaluate the ESS five times and plot the mean along with the 25% and 75% quantiles for 20 different numbers of steps, ranging from 3 to 200, arranged evenly in log space. We summarize and visualize the results for $d \in \{1,2,3,5\}$ in Figure 4.1.

Integral Estimation Tasks

As we discussed previously, high ESS does not necessarily indicate low IS variance. Therefore, we need to perform integral estimation tasks to validate the reliability of the estimated ESS and to identify cases where the proposal distribution fails to effectively cover the target distribution. We consider three different test functions ϕ for the integral estimation task. To obtain the true value for $\mathbb{E}_\pi[\phi(x)]$, we sample a large number of samples (1 million) $x \sim \pi$ from the true target distribution and estimate the integral using standard MC methods.

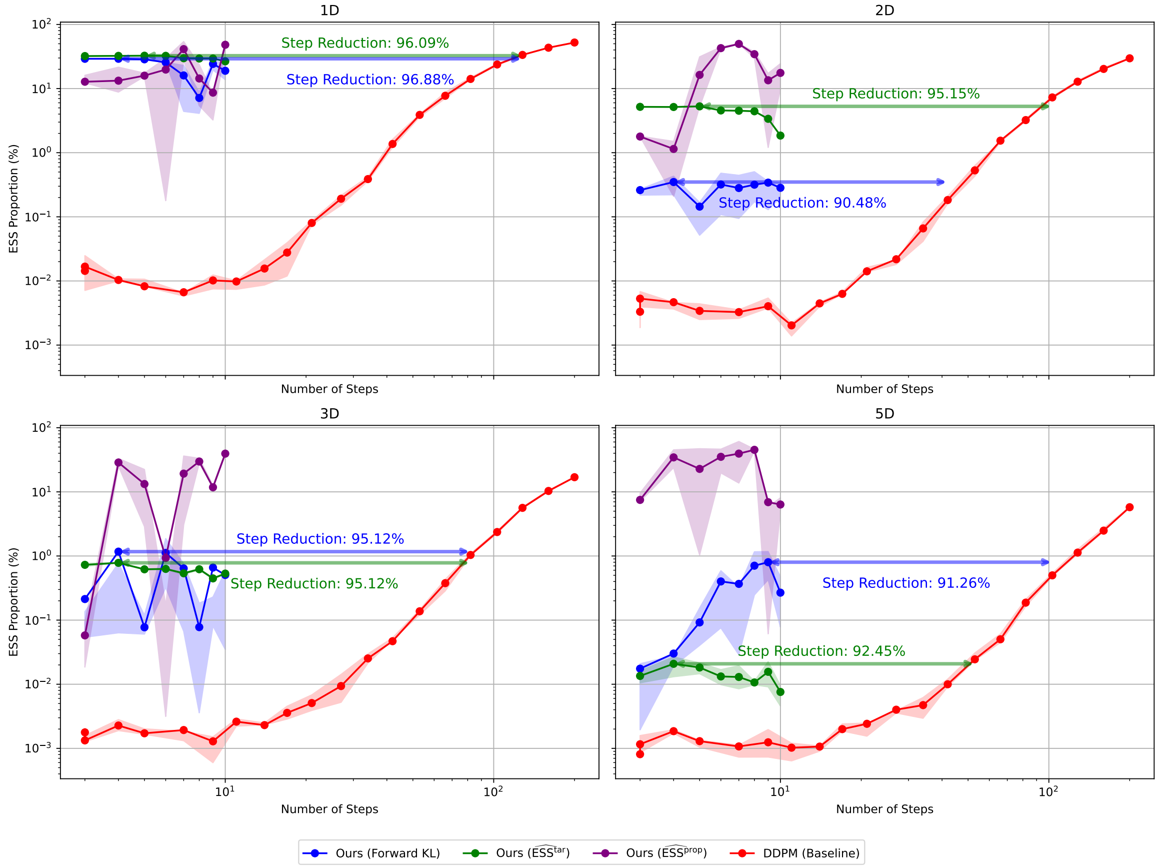


Fig. 4.1 Comparing ESS proportion between the proposed algorithm and the baseline algorithm. The ESS proportion is plotted against the number of time steps for our method with optimal parameters tuned via forward KL (blue curve), $\widehat{\mathrm{ESS}}^{\mathrm{prop}}$ (purple curve), and $\widehat{\mathrm{ESS}}^{\mathrm{tar}}$ (green curve), as well as for the baseline algorithm (red curve). All ESS proportions are estimated using 1 million samples from the proposal distribution.

We then proceed to estimate the integral using IS with both our method and the baseline method, comparing the results with the true value. This comparison serves as an additional metric, complementing the ESS, to provide a more comprehensive evaluation of IS performance. Before performing the integral estimation task, we need to fix the number of time steps for our method and for the baseline method. For the baseline DDPM method, as shown in Figure 4.1, 100 steps yield a comparable proportion of ESS to our method. Therefore, we will fix the number of steps for the baseline method at 100. For our proposed method, we choose the number of steps that correspond to the largest mean ESS. The number of samples we use in both our method and the baseline method will also be 1 million. We summarize the results in Table 4.1.

Visualizing Marginal Distributions

In addition to the integral estimation task for evaluating the effectiveness of our algorithm, we also provide visualizations to examine the coverage of the proposal distribution relative to the target distribution. Using the notations from Chapter 2, and given that we have access to the true target distribution π for this toy dataset, we draw samples $x_{t_{0:N}}^{(\mathrm{tar})} \sim q_\theta$ from the target distribution, as well as samples $x_{t_{0:N}}^{(\mathrm{prop})} \sim p_\theta$ from the proposal distribution. We choose $d = 1$ and $N = 6$ to visualize the samples using three sets of optimal parameters identified by three different metrics.

However, even in a one-dimensional space, visualizing samples from t_0 to t_5 together is still challenging. One possible approach to visualization is to plot samples for pairs of time points. We selected three pairs of time steps—specifically, (t_0, t_2), (t_1, t_2), and (t_1, t_5)—to visualize and support our subsequent discussion. The visualization is presented in Figure 4.2. In these visualizations, the blue points represent the sample distribution of the target, while the orange points represent the sample distribution of the proposal. Ideally, for a well-behaved IS algorithm, we would expect the proposal sample distribution to effectively cover the target sample distribution.

Result Discussion

We now discuss the results for the baseline method and our method tuned using three different metrics:

- **Baseline DDPM:** From Figure 4.1, we can observe that as the number of steps increases, the ESS proportion increases as well. Eventually, the ESS proportion will saturate at a certain number of steps, and further increasing the number of steps will not yield any significant improvement in ESS proportion (although this saturation

Table 4.1 Summary of integral estimation results for experiments using an analytical score. The task is performed with four different test functions: $\|x\|_2^2$, $\log\|x\|_1$, and $\cos\left(\|x\|_2\right)$. We evaluate five methods, including one using true samples from the target distribution, which is equivalent to performing IS with the proposal being the target distribution, ensuring an ESS of 100%. The estimation results using true samples serve as a reference (true value), against which other results are compared. We also present results for the baseline method and our proposed method, with optimal parameters tuned via three different metrics. The task is conducted in four different dimensional spaces using 1 million samples. Each experiment is repeated 5 times, with the mean and standard deviation reported. The number in parentheses indicates the number of time steps used to obtain the result.

Task $\mathbb{E}_\pi[\phi(x)]$	$\|x\|_2^2$	$\log\|x\|_1$	$\cos(\|x\|_2)$	ESS (%)
1D				
True Sample	2.150 ± 0.002	0.156 ± 0.000	0.206 ± 0.000	100
DDPM (100)	2.150 ± 0.003	0.156 ± 0.001	0.206 ± 0.001	22.9 ± 0.1
Ours $\left(\widehat{\text{ESS}}^{\text{prop}}, 6\right)$	2.189 ± 0.011	0.175 ± 0.008	0.194 ± 0.004	21.6 ± 21.0
Ours $\left(\widehat{\text{ESS}}^{\text{tar}}, 3\right)$	2.152 ± 0.004	0.156 ± 0.001	0.205 ± 0.001	32.2 ± 0.0
Ours (Forward KL, 3)	2.151 ± 0.003	0.156 ± 0.001	0.205 ± 0.001	29.2 ± 0.0
2D				
True Value	4.297 ± 0.004	0.894 ± 0.000	−0.233 ± 0.000	100
DDPM (100)	4.300 ± 0.010	0.894 ± 0.001	−0.234 ± 0.001	6.9 ± 0.3
Ours $\left(\widehat{\text{ESS}}^{\text{prop}}, 6\right)$	4.299 ± 0.014	0.905 ± 0.005	−0.237 ± 0.007	29.2 ± 18.1
Ours $\left(\widehat{\text{ESS}}^{\text{tar}}, 4\right)$	4.297 ± 0.009	0.894 ± 0.001	−0.233 ± 0.002	5.2 ± 0.0
Ours (Forward KL, 6)	4.271 ± 0.061	0.892 ± 0.007	−0.230 ± 0.010	0.4 ± 0.3
3D				
True Value	6.451 ± 0.002	1.310 ± 0.000	−0.442 ± 0.000	100
DDPM (100)	6.455 ± 0.027	1.311 ± 0.004	−0.441 ± 0.005	1.5 ± 0.6
Ours $\left(\widehat{\text{ESS}}^{\text{prop}}, 10\right)$	6.396 ± 0.003	1.311 ± 0.000	−0.438 ± 0.001	35.8 ± 2.5
Ours $\left(\widehat{\text{ESS}}^{\text{tar}}, 3\right)$	6.479 ± 0.058	1.313 ± 0.005	−0.444 ± 0.005	0.8 ± 0.0
Ours (Forward KL, 5)	6.468 ± 0.324	1.311 ± 0.023	−0.439 ± 0.017	0.9 ± 1.0
5D				
True Value	10.750 ± 0.006	1.830 ± 0.000	−0.498 ± 0.000	100
DDPM (100)	10.735 ± 0.053	1.829 ± 0.003	−0.498 ± 0.003	0.3 ± 0.2
Ours $\left(\widehat{\text{ESS}}^{\text{prop}}, 6\right)$	9.825 ± 0.01	1.789 ± 0.001	−0.538 ± 0.001	54.7 ± 11.5
Ours $\left(\widehat{\text{ESS}}^{\text{tar}}, 3\right)$	10.906 ± 0.279	1.836 ± 0.013	−0.489 ± 0.018	0.0 ± 0.0
Ours (Forward KL, 3)	10.752 ± 0.198	1.830 ± 0.008	−0.507 ± 0.012	0.9 ± 0.7

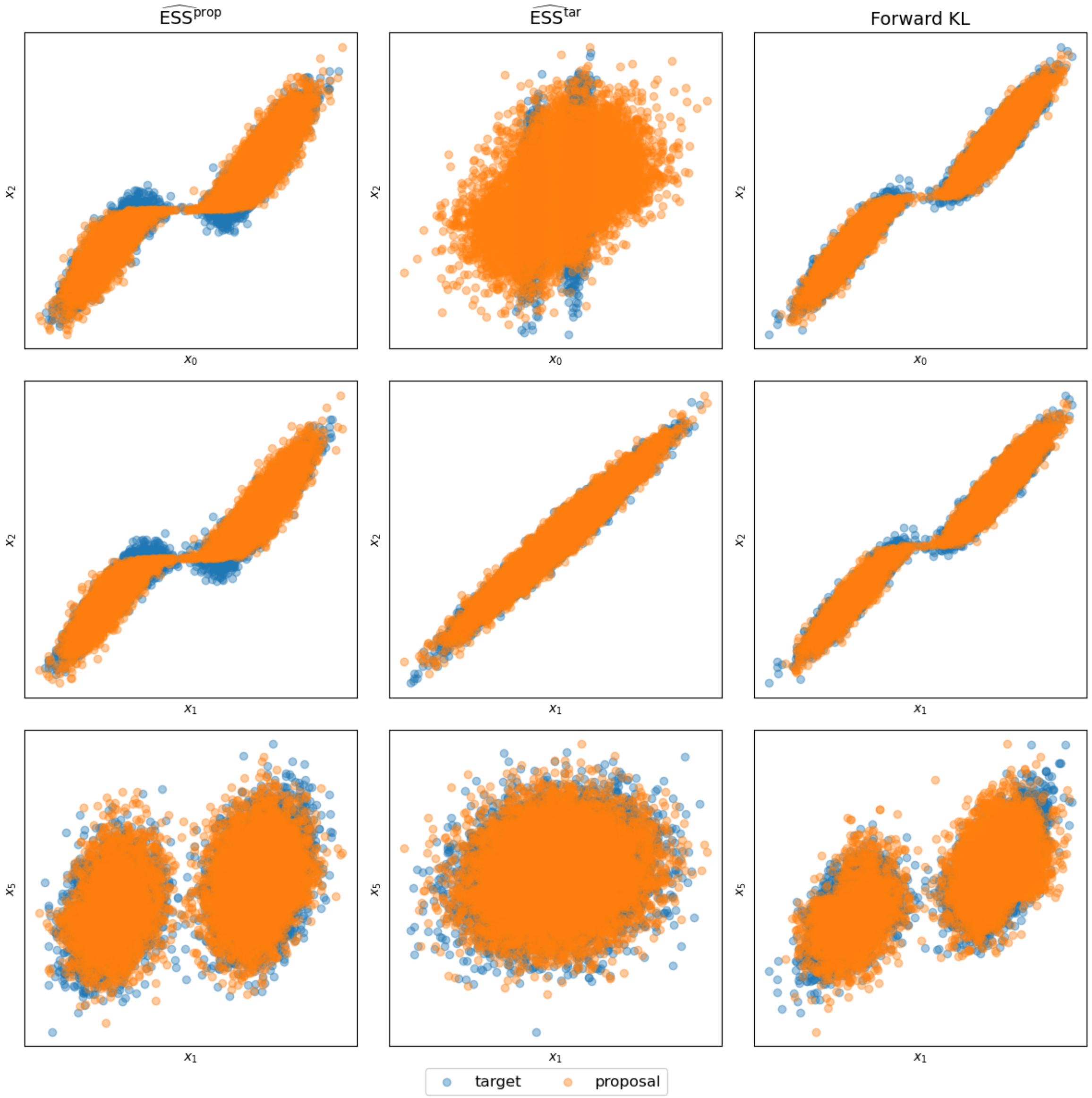


Fig. 4.2 Visualization of samples from the target distribution (blue points) and the proposal distribution (orange points) using optimal parameters tuned by three different metrics. Samples are visualized at three pairs of time points: (t_0, t_2), (t_1, t_2), and (t_1, t_5). Each column represents the visualization of the proposal and target distributions using optimal parameters tuned by each respective metric.

is not shown in our plot). For this particular problem, DDPM typically requires a large number of steps (at least 100) to achieve a reasonable ESS proportion. From Table 4.1, we can see that with a large number of steps, the DDPM method performs well, showing no noticeable error compared to the true value. This result validates the integration of IS with score-based diffusion models and suggests that enhancing efficiency remains crucial for applying this algorithm in practical scenarios.

- **Ours ($\widehat{\mathbf{ESS}}^{\mathbf{prop}}$):** For our algorithm, tuning based on $\widehat{\mathrm{ESS}}^{\mathrm{prop}}$ consistently results in high ESS across all dimensions, which is expected since this metric directly measures the performance of our IS algorithm. However, as shown in the integral estimation results in Table 4.1, the parameters found by tuning this metric consistently lead to large errors compared with the true values, even when a large number of samples are used to estimate the integral and the ESS proportion is high. The underlying reason is that the proposal distribution only partially covers the target distribution. This can be observed in the first column of Figure 4.2, which shows the visualization of the target and proposal sample distributions for parameters tuned using $\widehat{\mathrm{ESS}}^{\mathrm{prop}}$. The proposal sample distribution clearly fails to fully cover the target sample distribution, explaining why we observe a high ESS proportion but significant error in the integral estimation tasks. This confirms that tuning based on $\widehat{\mathrm{ESS}}^{\mathrm{prop}}$ is not a valid approach for our algorithm.
- **Ours ($\widehat{\mathbf{ESS}}^{\mathbf{tar}}$):** From Figure 4.1, we can observe impressive results for the parameters tuned using this metric: the highest ESS proportion is achieved with only three time steps, and adding more time steps does not further improve the ESS. From the integral estimation results in Table 4.1, the parameters found using this method perform equally well compared to the baseline DDPM method. However, a limitation of this metric becomes apparent as the dimension d increases—the ESS proportion drops significantly.

  The reason for this can be seen in the visualization of proposal and target sample distributions shown in the middle column of Figure 4.2. In the first plot for x_{t_2} against x_{t_0}, although the proposal distribution fully covers the target distribution, it is excessively wide compared to the target sample distribution, leading to a rapid decline in ESS as the dimension increases. The proposal distribution becomes too wide because, in our algorithm, the samples x_{t_1} are mapped to x_{t_0} according to the deterministic ODE, and then noise is added to form the final proposal distribution. For the optimal parameters found using $\widehat{\mathrm{ESS}}^{\mathrm{tar}}$, the suggestion is to add a large amount of noise at this step to form the proposal distribution.

We hypothesize that this occurs because, when using Gaussian distributions, the proposal and target distributions can match well if the ODE trajectories used to transform variables (see Figure 3.4) are linear. If the trajectories are nonlinear, aligning the proposal and target distributions becomes more challenging. Additionally, ODE trajectories are approximately linear when t is large, with most nonlinearity occurring at small t. To ensure that the proposal covers the target distribution, tuning $\widehat{\text{ESS}}^{\text{tar}}$ may result in the addition of large noise at the last step to bypass the nonlinear trajectories and form the proposal distribution. While this approach might be the simplest, it may not be the most effective, as it only works in low dimensional space.

We leave the verification of this hypothesis and potential improvements to this metric as future work.

- **Ours (Forward KL):** Finally, we discuss the performance of the algorithm using parameters tuned with forward KL. From Figure 4.1, we observe that the optimal parameters found by forward KL lead to a relatively low ESS proportion in low-dimensional spaces compared to the parameters found by $\widehat{\text{ESS}}^{\text{prop}}$, but it shows higher ESS proportions in higher-dimensional spaces. From Table 4.1, the integral estimation results of parameters found using forward KL show no significant discrepancies compared to the true values.

  In the visualization of proposal and target sample distributions in Figure 4.2, forward KL produces the best alignment between the proposal and target distributions compared to the other two methods. Ideally, this alignment should result in a high ESS proportion, which contradicts the experimental results. The reason tuning this metric leads to lower ESS is that the tails of the proposal and target distributions do not match well. Since we do not have access to the analytical form of the forward KL divergence, we must estimate it using samples from the target distribution. This estimation does not approximate the tails accurately, which leads to lower ESS when we use a large number of samples for IS. Samples from the tail of the proposal may have a disproportionately large influence on the variance of IS because the tails of the proposal and target do not match well.

We now summarize the properties of the three metrics we investigated in Table 4.2.

4.1.2 Experiments with a Family of Consistency Models

We have demonstrated that our method requires significantly fewer time steps than DDPM to achieve the same level of IS variance, given the availability of an analytical score. In

Table 4.2 Comparison of the properties of three metrics used for tuning the algorithm: $\widehat{\mathrm{ESS}}^{\mathrm{prop}}$, $\widehat{\mathrm{ESS}}^{\mathrm{tar}}$, and Forward KL. The asterisk (*) indicates that the Forward KL metric may have limitations in proposal coverage, particularly in the distribution tails.

	$\widehat{\mathrm{ESS}}^{\mathrm{prop}}$	$\widehat{\mathrm{ESS}}^{\mathrm{tar}}$	Forward KL
High ESS Proportion	✓	✗	✓
Proposal Coverage	✗	✓	✗*
Gradient-based Optimization	✗	✓	✓
Any Initialization	✓	✗	✓
Improvement with More Steps	✓	✗	✓

this subsection, we transition to a more practical setting. We begin by training a score-based diffusion model, followed by distilling the pre-trained score model into a BCTM. Additionally, we will train a BCM in isolation without using the pre-trained score. Our initial goal is to compare the performance of the BCTM and BCM. Subsequently, we will apply the better-performing model to our method and compare the results with the baseline DDPM model. In this experiment, we aim to address the following questions:

1. Can our method effectively utilize a trained BCTM or BCM? How many steps can it reduce compared to the baseline DDPM method?
2. How scalable is our method when using trained models? Up to how many dimensions can our method effectively scale?

The primary goal of this subsection is to answer these questions and demonstrate the effectiveness of our method in a more practical setting compared to previous experiments. We will first verify the performance of the BCTMs and BCMs, and then assess the overall effectiveness of our method.

Evaluation of BCM and BCTM

Experiment Setup and Results: In Section 2.3.2, we extended the training of CTM to BCTM. Before applying these models to our method, we first evaluate and compare the performances of BCM and BCTM. Note that BCM is trained in isolation, meaning it does not utilize the pre-trained score model during training. In contrast, BCTM is trained via distillation using the pre-trained score model.

After training both models, we first examine their denoising abilities. To do this, we sample $z \sim \mathcal{N}(0, T^2 I)$ with $T = 80$, and then use BCM and BCTM to denoise the same z to $t = \varepsilon = 0.002$ using only one NFE. We visualize the samples produced by both models separately and compare them with the true samples from the target distribution. The results are shown in Figure 4.3a.

Next, we test the noising abilities of both models. We begin by drawing samples $x_0 \sim \pi$ from the true target distribution. Then, we use the analytical score to solve the PF ODE, finding the marginal distribution at $t = 1$ using x_0. This represents the true marginal samples at $t = 1$. We then use the same x_0 to noise the samples using BCM and BCTM from $t = 0.002$ to $t = 1$ using only one NFE, and compare the results with the true marginal distribution. The comparison is visualized in Figure 4.3b.

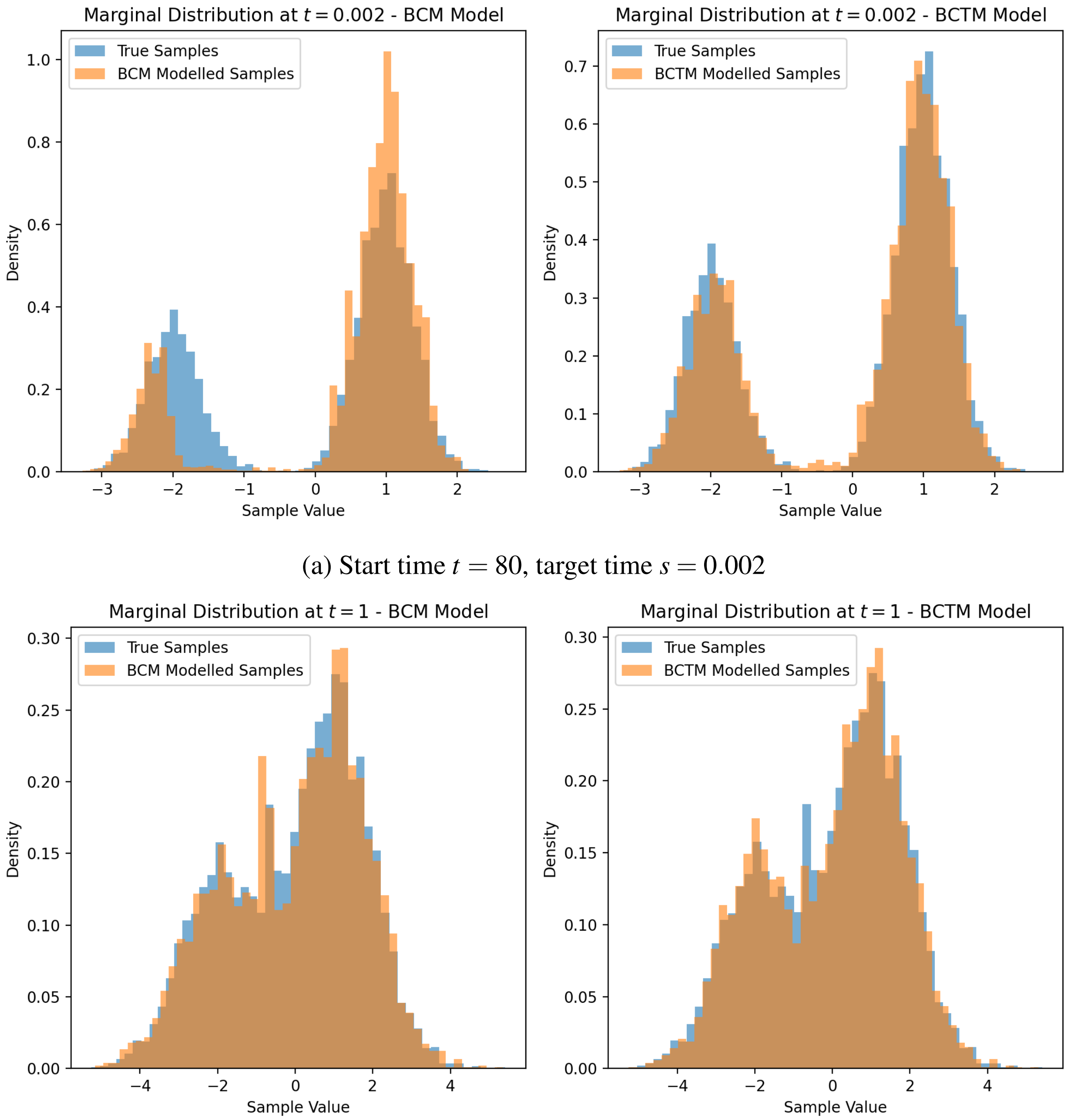
Results Discussion: From Figure 4.3, we observe that both models are capable of solving the PF ODE in both forward and backward directions. Notably, even though the training times for BCM and BCTM are approximately the same, BCTM demonstrates superior performance compared to BCM in terms of denoising. With just one NFE, BCTM produces a sample distribution that is nearly identical to the true sample distribution, whereas the sample distribution generated by BCM with one NFE still exhibits noticeable differences from the true distribution (although this can be improved through multi-step sampling).

Beyond this evaluation, we will further assess the performance of these two models when they are incorporated into our proposed IS algorithm.

Evaluation of Our Method

Comparing ESS: The experimental setup will be similar to that of the previous section, except that we will now use gradient-based optimization to tune the parameters instead of a greedy-based grid search.

As discussed earlier, there are three possible metrics that can be used to tune the parameters of our algorithm. However, $\widehat{\text{ESS}}^{\text{prop}}$ has been empirically shown to lead to results with large errors, and during experiments, we found that gradient-based optimization with this metric is unstable, so we will not use it here. Another metric is $\widehat{\text{ESS}}^{\text{tar}}$, which is compatible with gradient-based optimization. However, we found that optimizing with respect to this metric is highly sensitive to the initialization of the parameters. The parameters must be carefully initialized to ensure that the proposal fully covers the target distribution (even if it is allowed to be wider than the target) to stabilize the optimization process. This initialization depends on the target distribution, and one way to obtain such initialization is to first run a



(a) Start time $t = 80$, target time $s = 0.002$

(b) Start time $t = 0.002$, target time $s = 1$

Fig. 4.3 Visualization of denoising and noising performances of BCM and BCTMs. The first pair of plots visualizes the sample distributions produced by BCM and BCTM using 1 step, given the same starting noise, to examine the denoising ability of both models. The second plot shows the marginal distributions at $t = 1$ produced by BCM and BCTM using 1 step, given the same sample from the target distribution, to examine the noising ability of both models.

greedy-based grid search (for one loop through all parameters) and then use the results as the initialization for gradient-based optimization. Despite this, we found that the optimal parameters still struggle to scale up to high-dimensional spaces. Due to the complexity of obtaining suitable initialization and the scalability issues, we will not use this metric either. We leave the improvement of this metric for future work. Therefore, the metric we will use is the forward KL divergence, estimated using samples from the target distribution.

Similar to before, we will visualize the proportion of ESS against the number of time steps for our algorithm and for the baseline DDPM in $d \in \{1,3,5,10,20,50\}$. We will tune the parameters five times, evaluate the ESS for each set of optimal parameters found, and plot them against the number of steps for BCM and BCTM. We will also plot the ESS against the number of steps for the baseline DDPM model using the trained score model. We will use 100,000[1] samples to test our algorithm as well as the baseline DDPM method. We visualize the ESS comparison results in Figure 4.4.

Integral Estimation Tasks: Additionally, we will perform the integral estimation task for our algorithm here. Previously, we used integral estimation to check whether the tuning metric led to results with large errors compared to the true value. Here, the goal of the integral estimation task is to determine whether IS can correct the bias present in the original model without IS and to confirm our proposed algorithm with BCTM or BCM does not result in large errors.

Since BCTM performed better than BCM in the previous section, we will perform the integral estimation task using BCTM. For our proposed method, we choose the number of steps that correspond to the largest mean ESS according to Figure 4.4. We will estimate the integral using two sets of samples: the first set of samples will be directly sampled from BCTM, with the integral estimated using MC; the second set of samples will be obtained using our algorithm, with the integral estimated using IS.

For the baseline DDPM model, we will fix the number of steps to 150. The integral will be estimated using one set of samples obtained from DDPM. The integral will be estimated using MC and IS in two ways. This approach will test whether IS can correct the bias in the sample distribution and whether our proposed algorithm is more efficient than the baseline. We summarize the results in Table 4.3.

[1] We reduced the sample size from the previously used 1 million due to a limitation in our method when optimizing the forward KL divergence. This optimization results in a mismatch between the proposal and target distributions in the tail regions. With a large number of samples, the tail samples significantly degrade the IS estimation. We acknowledge this limitation and plan to address it in future work.

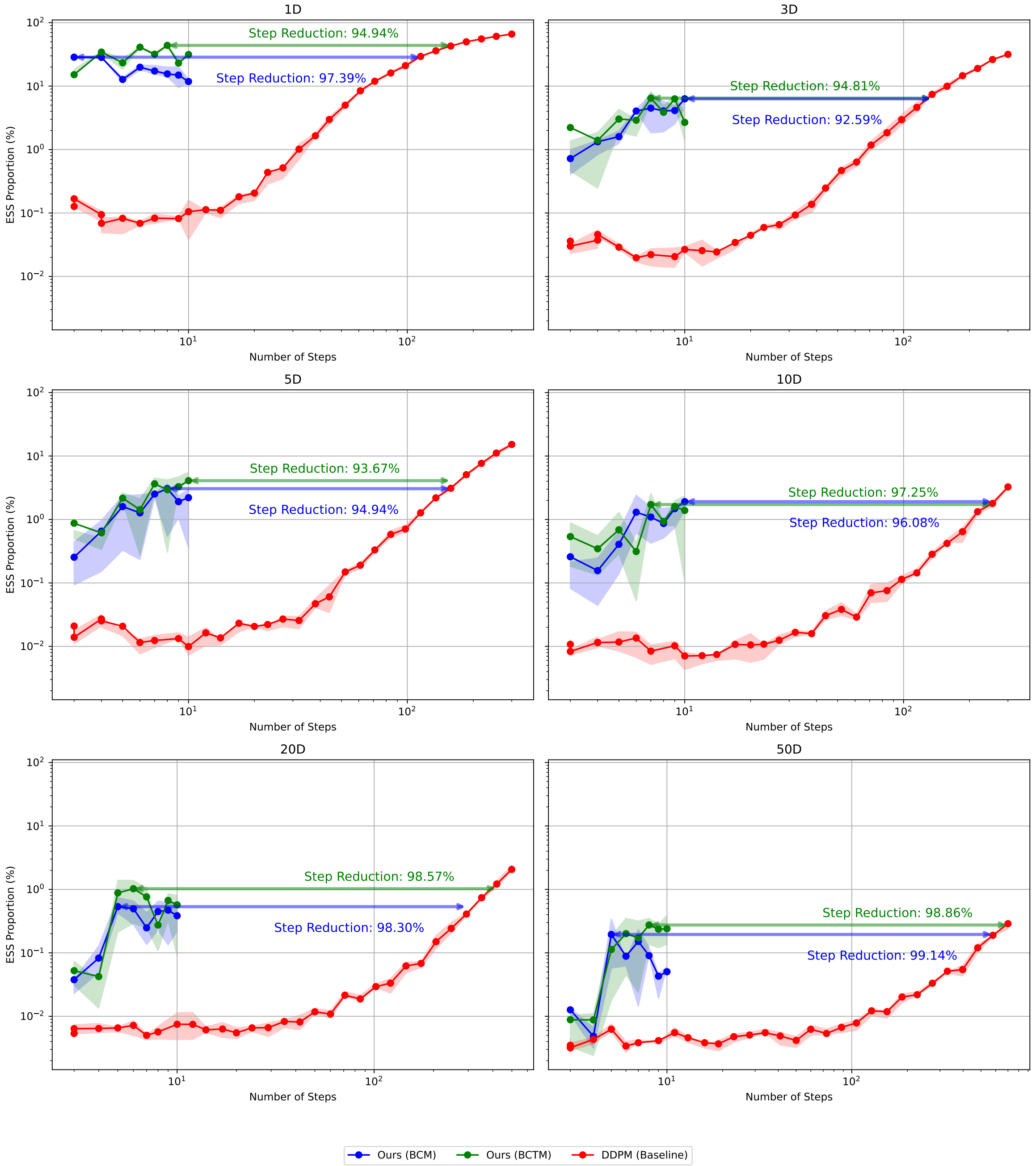


Fig. 4.4 Comparison of ESS proportion between the proposed algorithm and the baseline algorithm. For our proposed algorithm, we use BCM (blue curve) and BCTM (green curve), applying a gradient-based approach to minimize the forward KL divergence between the target and proposal distributions. The ESS proportion is estimated using 100,000 samples from the proposal distribution. Our proposed algorithm consistently shows a significant reduction in the number of time steps required to achieve the same ESS proportion as the baseline method (red curve).

Table 4.3 Summary of the integral estimation task using trained score models, BCM, and BCTM. We first estimate the integral's value using samples from the target distribution via MC as a reference. Then, we test the baseline DDPM method and our proposed method using MC and IS, comparing the results with the true values. The experiments are conducted in four different dimensional spaces, with each integral estimated using 100,000 samples. Each experiment is repeated 5 times, with the mean and standard deviation reported. The number in parentheses indicates the number of time steps used to obtain the result.

Task $\mathbb{E}_\pi[\phi(x)]$	$\|x\|_2^2$	$\log \|x\|_1$	$\cos(\|x\|_2)$	ESS (%)
1D				
True Samples + MC	2.151 ± 0.001	0.156 ± 0.000	0.205 ± 0.001	100
DDPM (150) + MC	2.096 ± 0.001	0.145 ± 0.000	0.220 ± 0.000	N/A
DDPM (150) + IS	2.151 ± 0.003	0.156 ± 0.001	0.205 ± 0.001	40.5 ± 0.3
BCTM (1) + MC	2.013 ± 0.002	0.094 ± 0.001	0.247 ± 0.001	N/A
BCTM (8) + IS	2.151 ± 0.004	0.156 ± 0.001	0.205 ± 0.001	34.1 ± 8.3
3D				
True Samples + MC	6.448 ± 0.002	1.311 ± 0.000	−0.441 ± 0.000	100
DDPM (150) + MC	6.344 ± 0.003	1.305 ± 0.000	−0.435 ± 0.000	N/A
DDPM (150) + IS	6.448 ± 0.012	1.310 ± 0.001	−0.441 ± 0.001	8.8 ± 1.0
BCTM (1) + MC	6.146 ± 0.003	1.273 ± 0.000	−0.401 ± 0.000	N/A
BCTM (9) + IS	6.467 ± 0.020	1.313 ± 0.002	−0.444 ± 0.002	1.5 ± 0.9
5D				
True Samples + MC	10.751 ± 0.005	1.830 ± 0.000	−0.498 ± 0.000	100
DDPM (150) + MC	10.589 ± 0.008	1.825 ± 0.000	−0.500 ± 0.000	N/A
DDPM (150) + IS	10.804 ± 0.042	1.833 ± 0.002	−0.497 ± 0.002	1.9 ± 0.5
BCTM (1) + MC	10.258 ± 0.005	1.803 ± 0.000	−0.497 ± 0.000	N/A
BCTM (7) + IS	10.652 ± 0.061	1.826 ± 0.003	−0.503 ± 0.002	0.8 ± 0.3
10D				
True Samples + MC	21.494 ± 0.012	2.529 ± 0.000	−0.299 ± 0.000	100
DDPM (150) + MC	21.222 ± 0.015	2.525 ± 0.000	−0.310 ± 0.001	N/A
DDPM (150) + IS	21.618 ± 0.196	2.532 ± 0.005	−0.295 ± 0.009	0.2 ± 0.1
BCTM (1) + MC	20.171 ± 0.008	2.496 ± 0.000	−0.321 ± 0.001	N/A
BCTM (9) + IS	21.261 ± 0.427	2.521 ± 0.011	−0.317 ± 0.024	0.3 ± 0.2
20D				
True Samples + MC	42.986 ± 0.024	3.225 ± 0.000	−0.251 ± 0.000	100
DDPM (300) + MC	42.768 ± 0.078	3.223 ± 0.001	−0.257 ± 0.002	N/A
DDPM (300) + IS	42.982 ± 2.063	3.226 ± 0.023	−0.246 ± 0.029	0.4 ± 0.1
BCTM (1) + MC	39.414 ± 0.101	3.182 ± 0.001	−0.253 ± 0.003	N/A
BCTM (6) + IS	38.737 ± 1.584	3.175 ± 0.019	−0.193 ± 0.015	0.6 ± 0.3

Results Discussion: From Figure 4.4, we observe that across all dimensional spaces, our proposed algorithm consistently outperforms the baseline DDPM method in terms of the number of time steps required to achieve the same ESS proportion. Our method consistently reduces the number of time steps by at least 90% compared to the baseline method, indicating a significant improvement in efficiency. Notably, our method maintains an ESS proportion of nearly 1% in 20-dimensional space using only 6 time steps, significantly outperforming the baseline method. This result demonstrates the potential scalability of our proposed approach to higher-dimensional problems. Within our algorithm, BCTM slightly outperforms BCM in terms of ESS proportion, which aligns with our previous observation that BCTM has better denoising ability than BCM.

Regarding the integral estimation task, both directly sample from DDPM and BCTM lead to biased integral estimates compared to the true value. However, after applying IS with either DDPM or BCTM, the integral estimation results are clearly corrected to be closer to the true value, demonstrating the effectiveness of IS. It is important to note that our proposed algorithm with IS exhibits large error compared to the true value in 20-dimensional space. As the dimensionality increases, the mismatch between the tails of the proposal and target distributions becomes increasingly problematic, severely impacting model performance. We have observed similar results in 50-dimensional space.

4.1.3 Summary

To summarize, we have conducted two experiments with the toy dataset, one using an analytical score and the other with a trained score. The key takeaway is that our proposed IS algorithm consistently outperforms the DDPM model in terms of efficiency, reducing the required time steps by at least 90% to achieve unbiased integral estimation results. The parameters of our proposed algorithm can be tuned by minimizing the forward KL divergence between the target and proposal distributions, though thisleads to mismatches in the tails, especially in high-dimensional spaces, which we aim to address in future work.

4.2 Simulation of Supercooled Water Molecules

We have demonstrated the effectiveness of our method using the toy dataset generated by GMMs. In this section, we shift our focus to a real-world problem involving the simulation of water molecules at extremely low temperatures, specifically supercooled water. The dataset comprises the 3D coordinates of each atom in a water molecule, which includes one oxygen atom and two hydrogen atoms.

We initially use MCMC methods to draw samples and construct the dataset. This dataset will then be used to train diffusion models and consistency models. Additionally, we have access to the true unnormalized energy function, which allows us to compute the unnormalized log probability of the target distribution.

The primary goal of this experiment is to evaluate the performance of our method in this real-world scenario and to assess its applicability to the complex dynamics of supercooled water molecules.

4.2.1 Modification of Model Parameterization and Sampling Process

Equivariant Diffusion Models and Consistency Models

As introduced in Section 2.2.4, we modified the parameterization of the original Equivariant Diffusion Model (EDM) based on DDPM, incorporating the parameterization introduced by Karras et al. (2022) based on score-based diffusion models. We applied preconditioning and adjusted the training method accordingly. This allows us to obtain a score model that preserves the desirable equivariance property, making it suitable for learning from molecular datasets.

For the family of consistency models, recall that all types are based on the parameterization:

$$D_\theta(x;t) = c_{\text{skip}}(t)x + c_{\text{out}}(t)F_\theta(c_{\text{in}}(t)x, t), \tag{4.2}$$

where $c_{\text{skip}}(t)$, $c_{\text{out}}(t)$ and $c_{\text{in}}(t)$ are defined as in Karras et al. (2022). To achieve the desired symmetry property, we use an Equivariant Graph Neural Network (EGNN) to parameterize F_θ. This ensures that all types of consistency models preserve the equivariance necessary for molecular datasets.

The EGNN model, as used by Hoogeboom et al. (2022); Satorras et al. (2021), accepts two inputs: the 3D coordinates of atoms, x, and the atom features, h. In our experiment, the atom features include the types of atoms (oxygen and hydrogen). The EGNN produces two outputs: the predicted 3D coordinates, $\hat{x}$, and the predicted atom features, $\hat{h}$. However, in our experiments, we only require the prediction of the 3D coordinates of the atoms, so the output of the predicted features is disregarded for both score-based diffusion models and consistency models.

Sampling Process for Equivariant Models

As introduced previously in Section 2.2.4, to preserve the equivariance property, the noising and denoising distributions of x must be defined in a subspace where the center of gravity is zero. We denote such a normal distribution as $\mathcal{N}_x(x;\mu,\sigma^2 I)$, where $\sum_i x_i = 0$. Sampling from a normal distribution with zero center of gravity is straightforward. As shown by Xu et al. (2022), one can sample from a conventional normal distribution to obtain $\tilde{x}$, and then project $\tilde{x}$ onto the subspace where the center of gravity is zero, resulting in $x \sim \mathcal{N}_x$. Note that this sampling process is only applicable to isotropic normal distributions. The log-likelihood for such a normal distribution can be computed as:

$$\mathcal{N}_x(x;\mu,\sigma^2 I) = (\sqrt{2\pi}\sigma)^{-(M-1)n} \exp\left(-\frac{1}{2\sigma^2}\|x-\mu\|^2\right). \tag{4.3}$$

where both x and μ are in the subspace with zero gravity. To apply our algorithm to this specific experiment, it is necessary to replace the standard normal distribution $\mathcal{N}$ in Algorithm 1 with the distribution $\mathcal{N}_x$, where the center of gravity is zero.

4.2.2 Experiments with a Family of Consistency Models

The main goal of this experiment is to evaluate how our method performs compared to the baseline DDPM method, to determine how much it can outperform the baseline, and to understand the dimensionality our method can effectively scale to. Given that this dataset is invariant to rotation, translation, and permutation, it is advantageous to use an E(3) equivariant diffusion model and extend the consistency models, BCTMs, and BCMs to be E(3) equivariant. We will begin by setting up the equivariant score-based diffusion models, as well as the E(3) equivariant BCTM and BCM. Following this, we will assess the performance of the models and then compare our method with the baseline method, evaluating their respective performances.

Evaluation of BCM and BCTM

Experiment Setup and Results: We aim to train both score-based diffusion models and consistency models on the water molecule dataset. To generate the dataset[2], the number of water molecules must be specified, after which the dynamics of the water molecules will be simulated, and samples of atomic configurations will be drawn via MCMC. Note that the dataset generation process is time-consuming, so we generate only 10,000 samples and

[2] The code to generate the supercooled water molecule dataset is provided by Javier Antorán and Laurence Midgley.

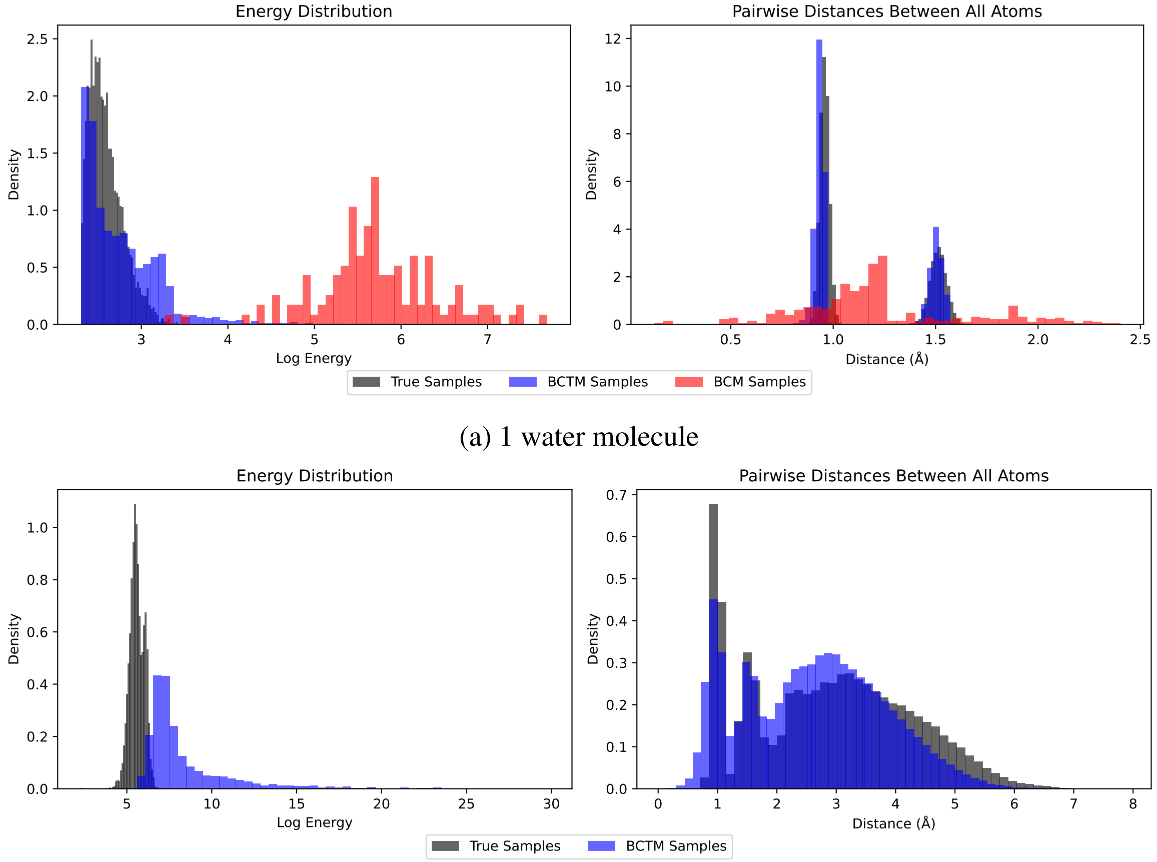
use these samples to train both the score-based diffusion model and the consistency models. During training, the parameter settings are the same as in previous experiments, except that ε is adjusted to 0.0002 instead of 0.002. This adjustment is necessary because the positions of atoms are extremely sensitive, and even slight mismatches can result in unrealistic energy values. Therefore, we choose a smaller starting time to ensure that the denoising process reaches a time step where the noise on atomic positions is negligible for energy computation.

We generate two datasets: one with 1 water molecule and another with 3 water molecules. After training, to evaluate the performance of BCM and BCTM, we first draw samples from both BCM and BCTM using only one NFE, given the same noise sampled from the base distribution. We then compute the energy of each sample and plot the energy histogram against the true energy histogram of the actual samples. Additionally, we plot the pairwise distances between all atoms and compare them with the true pairwise distances. The results are summarized in Figure 4.5.

Discussion: From Figure 4.5, we can clearly observe that BCTM outperforms BCM in both energy distribution and pairwise distance distribution. Learning the distribution of the energy function without access to a pre-trained score may be too challenging for BCM. Unlike the toy dataset used previously, the ODE trajectories here are much more complex due to the inter-dependencies between the positions of the atoms. In this scenario, having a pre-trained score greatly stabilizes the learning process and yields better performance, as demonstrated by BCTM. Therefore, in the evaluation of our method, we will use only BCTM, as it provides significantly better performance than BCM.

Additionally, we observe that for the dataset with 3 water molecules, the sample energy histogram produced by BCTM fails to fully cover the true energy histograms. The same issue is also present in the pre-trained score-based diffusion model. This highlights the challenge of accurately learning the true distribution using diffusion or consistency models. The underlying reason is that energy computation is extremely sensitive to the positions of the atoms. When dealing with 3 water molecules (i.e., 9 atoms), even a slight mismatch in the positions of two atoms can result in unrealistic energy values (and consequently very low log probability).

This raises concerns about applying IS to correct bias since our samples from the model do not fully cover the high-density regions of the true target distribution, leading to high variance in IS estimation. Therefore, for the evaluation of our method, we will focus only on the dataset containing 1 water molecule.



(a) 1 water molecule

(b) 3 water molecules

Fig. 4.5 Visualization of BCM and BCTM performance in terms of sampling quality. The two plots correspond to simulations with 1 water molecule and 3 water molecules. We visualize the energy histogram and pairwise distances between all atoms using 5,000 samples drawn from BCM and BCTM in 1 step, comparing them with the true distributions. BCM failed to converge during training on the dataset containing 3 water molecules; consequently, we omit these results from our analysis.

Evaluation of Our Method

Experiment Setup: To evaluate our proposed algorithm, the experiment setup will be similar to the previous experiments. We perform experiments using the models trained on the datasets with 1 water molecule corresponding to $d = 9$ dimensional space, respectively. To tune the parameters of our proposed algorithm, we optimize by minimizing the forward KL divergence between the target distribution and the proposal distribution.

We first visualize the ESS proportion of our proposed algorithm compared to the baseline DDPM method. An additional caveat is that computing the exact energy (and thus the exact log probability) is time-consuming, which may limit our scalability with a large number of samples (improving energy computation is considered future work beyond the scope of this project). Therefore, we use 10,000 samples to evaluate the ESS proportion. The ESS proportion is visualized in Figure 4.6. As before, we also provide the results for integral estimation tasks using 10,000 samples generated by the diffusion model or consistency models, and compare them with the true samples. The results are summarized in Table 4.4.

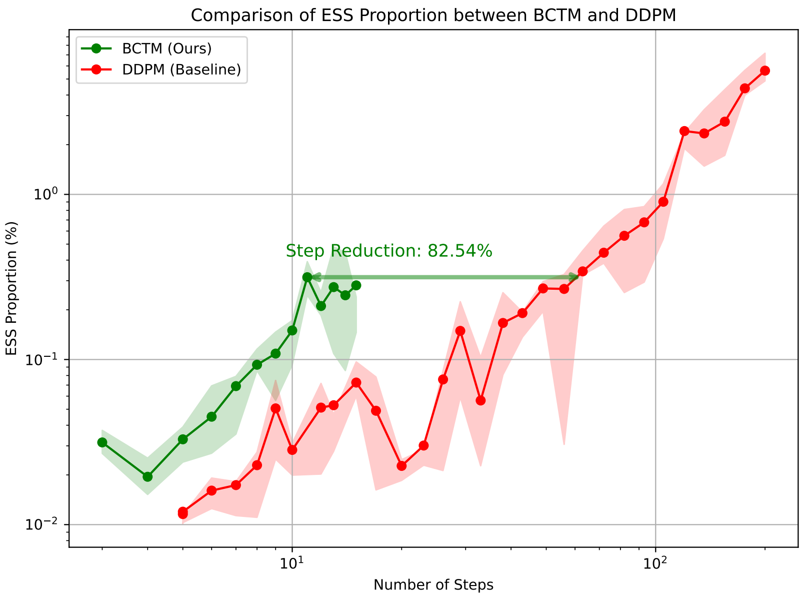


Fig. 4.6 Comparison of ESS proportion between the proposed algorithm (green curve) and the baseline algorithm (red curve) for the water molecule experiments with 1 water molecule. Even in this more challenging task, our proposed algorithm demonstrates a significant reduction in the number of steps required.

Results Discussion: From Figure 4.6, we observe that both methods exhibit lower ESS proportions compared to the toy dataset. Additionally, our proposed IS algorithm still requires fewer time steps to achieve the same level of ESS proportion, although the margin is smaller

Table 4.4 Summary of the integral estimation task for molecular experiments with 1 water molecule. The number in parentheses indicates the number of time steps used to obtain the result. Each integral and ESS proportion is estimated using 10,000 samples. Each experiment is performed 5 times, with the mean and standard deviation reported.

Task $\mathbb{E}_{\pi}[\phi(x)]$	$100\|x\|_2^2$	$\log\|x\|_1$	$\cos(100\|x\|_2)$	ESS (%)
True samples + MC	1.377 ± 0.000	-1.229 ± 0.001	0.652 ± 0.000	100
DDPM (80) + MC	1.374 ± 0.000	-1.230 ± 0.001	0.646 ± 0.001	N/A
DDPM (80) + IS	1.402 ± 0.042	-1.201 ± 0.021	0.717 ± 0.100	0.3 ± 0.3
DDPM (200) + MC	1.375 ± 0.001	-1.23 ± 0.000	0.648 ± 0.002	N/A
DDPM (200) + IS	1.377 ± 0.001	-1.227 ± 0.003	0.654 ± 0.004	6.1 ± 1.1
BCTM (11) + MC	1.334 ± 0.000	-1.250 ± 0.001	0.513 ± 0.001	N/A
BCTM (11) + IS	1.377 ± 0.014	-1.233 ± 0.016	0.650 ± 0.038	0.3 ± 0.1

compared to the toy dataset. This is understandable, given that the ODE trajectories for the water molecule dataset are significantly more complex than those of the toy dataset.

Furthermore, from Table 4.4, we see that with a low computing budget (around 80 time steps), our method performs roughly equal to the baseline method, using only 11 time steps in terms of ESS proportion. Both models, when incorporated with IS, can correct the bias of MC estimation that directly samples from the model. However, when a higher computing budget is available (i.e., more than 200 time steps), DDPM performs much better than our proposed algorithm. Notably, increasing the computing budget for our method, by increasing the number of time steps, does not result in a corresponding increase in ESS proportion, unlike what is observed for the baseline DDPM. This suggests a potential limitation of our proposed algorithm.

4.2.3 Summary

To summarize, in the water molecule experiment, we trained both BCM and BCTM on the molecular datasets. We observed that BCTM, trained via distillation using a pre-trained score model, performs much better than BCM trained in isolation, consistent with our observations from the toy dataset. During the evaluation of our proposed algorithm, we found that it still reduces the number of steps required by the baseline DDPM and verified that both the baseline and our algorithm can be used to correct the bias in the sample distribution. However, this experiment also highlighted a potential limitation of our algorithm: the lack of a satisfactory trade-off between compute budget and IS performance. We suggest possible ways to overcome this limitation in Section 5.2.

4.3 Chapter Summary

In this chapter, we presented the results of two experiments—one with a toy dataset and one with a molecular dataset—and discussed the implications of the results, pointing out possible limitations of our proposed algorithm. To summarize, our proposed algorithm consistently shows efficiency improvements across all tasks, demonstrating significant potential for practical applications. However, it is also limited by the tuning metric used to optimize the parameters, which impacts the scalability of the algorithm. Additionally, our proposed algorithm does not exhibit performance increases with a higher computing budget, indicating a lack of an explicit trade-off between compute and performance. We leave the improvement of our algorithm as future work.

Chapter 5

Conclusions and Future Work

In this work, we proposed a sampling algorithm that combines IS and Bidirectional Consistency (Trajectory) Models, enabling the acquisition of unbiased samples with significantly fewer time steps than when using IS and DDPM in isolation. Our algorithm also has the potential to scale to high-dimensional spaces, overcoming the limitations of the baseline method. We presented the mathematical formulation of both the baseline DDPM method and our proposed algorithm, providing a solid theoretical foundation. Additionally, we introduced the parameters of our algorithm, offering flexibility in deploying it to various target distributions. While this flexibility is advantageous, it also presents the challenge of parameter tuning. We proposed and tested three potential metrics for tuning, each with its own advantages and disadvantages. Among them, minimizing the forward KL divergence between the target distribution and the proposal distribution showed the most promising results, especially in scaling to high-dimensional spaces.

We conducted extensive experiments to evaluate the performance of our proposed algorithm against the baseline. In all cases, our algorithm consistently outperformed the baseline in terms of the number of time steps required to achieve comparable performance. The experiment involving supercooled water molecules was particularly challenging due to the difficulties in training diffusion and consistency models. By extending BCTM and integrating it with EGNN to incorporate appropriate equivariance properties, both the score model and BCTM demonstrated promising results in sampling from molecular energy functions, although there is room for further improvement.

Our proposed algorithm also outperformed the baseline in challenging experiment of water molecules, albeit by a smaller margin than in the toy dataset experiments. However, both our algorithm and the baseline struggled to generate unbiased samples for datasets containing two or more water molecules, likely due to the complex dependencies between atoms, which complicate the learning of ODE trajectories.

In summary, our proposed algorithm is theoretically grounded and has shown promising results in the experiments conducted. With further refinement, it has the potential for practical application, offering significant benefits in terms of efficiency and accuracy in predictions.

5.1 Limitations

The limitations of our work are as follows:

- The tuning process is currently constrained by the chosen metric. Although the forward KL divergence metric used for parameter tuning shows promise, it is not flawless. Specifically, it does not ensure a good match between the proposal and target distributions in the tails, which limits the scalability of our algorithm. Other potential metrics, such as $\widehat{\mathrm{ESS}}^{\mathrm{prop}}$ and $\widehat{\mathrm{ESS}}^{\mathrm{tar}}$, also have their own shortcomings and require further refinement.
- We did not observe a consistent relationship between IS performance and the number of time steps. Unlike the DDPM baseline method, our method does not exhibit a clear benefit from increasing the number of steps. Ideally, we would like to establish an explicit trade-off between computational budget and the algorithm's performance.
- The supercooled molecule experiments we conducted may have been too challenging. Training a score-based diffusion model or consistency models on this dataset is difficult, which significantly affects the evaluation of our proposed algorithm's performance. This challenge prevents us from making accurate comparisons between our algorithm and the baseline.

5.2 Future Work

We now outline potential future directions that could further enhance the performance of our proposed algorithm.

- In practice, the baseline DDPM method performs well with a very large number of steps (potentially thousands), achieving reasonable ESS proportions even in high-dimensional spaces. However, as discussed in the limitations section, our method does not exhibit clear performance improvements with an increased number of time steps. To combine the strengths of both methods, we propose reparameterizing our algorithm within the BCTM framework, possibly by introducing additional parameters. This modification could allow the new algorithm to automatically interpolate between the baseline DDPM model and our current approach. Consequently, increasing the number of time steps would yield clear benefits, while still requiring significantly fewer steps than the DDPM baseline.

- It is important to note that our algorithm is not restricted to molecular datasets. In fact, any application requiring unbiased samples could potentially benefit from our proposed algorithm. Therefore, we suggest extending our experiments to more general datasets, possibly in domains beyond molecular simulations.

References

Anderson, B. D. (1982). Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326.

Ascher, U. M. and Petzold, L. R. (1998). *Computer methods for ordinary differential equations and differential-algebraic equations*. SIAM.

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.

Hoogeboom, E., Satorras, V. G., Vignac, C., and Welling, M. (2022). Equivariant diffusion for molecule generation in 3d. In *International conference on machine learning*, pages 8867–8887. PMLR.

Hyvärinen, A. and Dayan, P. (2005). Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4).

Karras, T., Aittala, M., Aila, T., and Laine, S. (2022). Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577.

Kim, D., Lai, C.-H., Liao, W.-H., Murata, N., Takida, Y., Uesaka, T., He, Y., Mitsufuji, Y., and Ermon, S. (2023). Consistency trajectory models: Learning probability flow ode trajectory of diffusion. *arXiv preprint arXiv:2310.02279*.

Kingma, D., Salimans, T., Poole, B., and Ho, J. (2021). Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707.

Li, L. and He, J. (2024). Bidirectional consistency models. *arXiv preprint arXiv:2403.18035*.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092.

Satorras, V. G., Hoogeboom, E., and Welling, M. (2021). E (n) equivariant graph neural networks. In *International conference on machine learning*, pages 9323–9332. PMLR.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR.

Song, J., Meng, C., and Ermon, S. (2020a). Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*.

Song, Y. and Dhariwal, P. (2023). Improved techniques for training consistency models. *arXiv preprint arXiv:2310.14189*.

Song, Y., Dhariwal, P., Chen, M., and Sutskever, I. (2023). Consistency models. *arXiv preprint arXiv:2303.01469*.

Song, Y. and Ermon, S. (2019). Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2020b). Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*.

Vincent, P. (2011). A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674.

Xu, M., Yu, L., Song, Y., Shi, C., Ermon, S., and Tang, J. (2022). Geodiff: A geometric diffusion model for molecular conformation generation. *arXiv preprint arXiv:2203.02923*.

Appendix A

Extended Background

A.1 Training Consistency Models

Training Consistency Models via Distillation

Following Song et al. (2023), CMs are proposed based on the score-based diffusion model used by Karras et al. (2022). Therefore, for both of the training methods, we will use the exact same diffusion SDE as in Eq. (2.25) and the corresponding PF ODE as in Eq. (2.26).

Assume a pre-trained score model $s_\phi(x,t)$ is available. Consider discretizing the time as $\varepsilon = t_1 < t_2 < \cdots < t_N = T$. The exact arrangement of the time steps will follow Karras et al. (2022) as shown in Eq. (2.29). With N large enough, an accurate estimation of x_{t_n} can be obtained from $x_{t_{n+1}}$ by running one discretization step of a numerical ODE solver. This estimate, $\hat{x}^\phi_{t_n}$, according to the ODE in Eq. (2.26), is given by:

$$\hat{x}^\phi_{t_n} := x_{t_{n-1}} - (t_n - t_{n+1}) t_{n+1} s_\phi(x_{t_{n+1}}, t_{n+1}) \tag{A.1}$$

Note that due to our specific choices for f and g for the diffusion SDE, given $x_0 \sim p_{\text{data}}$, $x_{t_{n+1}}$ can be obtained by sampling from $\mathcal{N}(x_{t_{n+1}}; x_0, t^2_{n+1} I)$. Therefore, to train the consistency models, after sampling $n \sim \mathcal{U}[1, N-1]$ (a uniform categorical distribution on $1, \ldots, N-1$), we can first diffuse a data sample x_0 to $x_{t_{n+1}}$ and then run one step of Euler's method to get $\hat{x}^\phi_{t_n}$. The goal is to minimize the output differences on the pair $(\hat{x}^\phi_{t_n}, x_{t_{n+1}})$. The following Consistency Distillation (CD) loss will be used as the optimization objective:

$$\mathcal{L}^N_{\text{CD}}(\theta, \theta^-; \phi) := \mathbb{E}_{n,x,x_{t_{n+1}}} \left[\lambda(t_n) d(f_\theta(x_{t_{n+1}}, t_{n+1}), f_{\theta^-}(\hat{x}^\phi_{t_n}, t_n)) \right] \tag{A.2}$$

where $\lambda(\cdot) \in \mathbb{R}^+$ is a positive weighting function, θ^- denotes the running average of past values during optimization, and d is the metric function..

Training Consistency Models in Isolation

If a pre-trained model is not available for training the consistency models, an alternative approach is needed. Given $x_{t_{n+1}} = x + t_{n+1}z$ where $x \sim p_{\text{data}}$ and $z \sim \mathcal{N}(0, I)$, Song et al. (2023) used the approximation $\hat{x}_{t_n}^{\phi} \approx x + t_n z$, using the same x and z, to replace the previous value. They also proved that the following Consistency Training (CT) loss can be used to train the consistency models from scratch:

$$\mathscr{L}_{\text{CT}}^{N}(\theta, \theta^-) := \mathbb{E}_{z,x,t_n}\left[\lambda(t_n) d(f_{\theta}(x + t_{n+1}z, t_{n+1}), f_{\theta^-}(x + t_n z, t_n))\right] \tag{A.3}$$

According to Song and Dhariwal (2023), several improved techniques were proposed to enhance the performance of CT.

A.2 Derivation of Analytical Score of GMM

Derivation of Analytical Score

For a given GMM, denoted as π, with M components in d-dimensional space, the log-likelihood can be expressed as:

$$\log \pi(x) = \log \sum_{m=1}^{M} \omega_m \mathcal{N}(x; \mu_m, \sigma_m^2 I), \tag{A.4}$$

where $\sum_{m=1}^{M} \omega_m = 1$ represents the weights for each component, and $\mu_m \in \mathbb{R}^d$ and $\sigma_m^2 \in \mathbb{R}^+$ for $m = 1, \ldots, M$ are the mean and variance of each component, respectively. In this context, we are assuming a simplified variance that is isotropic across all components.

According to the diffusion SDE outlined in Eq. (2.25), the noise distribution at time $t \in [\varepsilon, T]$ can be expressed as:

$$\log p_t(x) = \log \sum_{m=1}^{M} \omega_m \mathcal{N}(x; \mu_m, (\sigma_m^2 + t^2) I). \tag{A.5}$$

The score, $\nabla_x \log p_t(x)$, can be obtained by differentiating the above expression with respect to x. In practice, this can be computed through backpropagation in any deep learning framework. With the analytical score in hand, we can accurately trace the PF ODE trajectory with minimal error, provided the number of steps is sufficiently large. This setup allows us to test whether our method is effective in scenarios analogous to having a perfectly trained BCM or BCTM.

As previously described, the baseline method in our experiments will be the DDPM sampler. Given the analytical score, the estimate $\hat{x}_0$ can be obtained using Tweedie's formula:

$$\hat{x}_0 = x_t + t^2 \nabla_x \log p_t(x). \tag{A.6}$$

This estimated $\hat{x}_0$ will be used in the proposal distribution as shown in Eq. (3.12), while the target distribution will remain consistent with that presented in Eq. (3.9).