PathFlow: A Normalizing Flow Generator that Finds Transition Paths

Abstract

Sampling from a Boltzmann distribution to calculate important macro statistics is one of the central tasks in the study of large atomic and molecular systems. Recently, a one-shot configuration sampler, the Boltzmann generator [Noé et al., 2019], is introduced. Though a Boltzmann generator can directly generate independent metastable states, it lacks the ability to find transition pathways and describe the whole transition process. In this paper, we propose PathFlow that can function as a one-shot generator as well as a transition pathfinder. More specifically, a normalizing flow model is constructed to map the base distribution and linear interpolated path in the latent space to the Boltzmann distribution and a minimum (free) energy path in the configuration space simultaneously. PathFlow can be trained by standard gradient-based optimizers using the proposed gradient estimator with a theoretical guarantee. PathFlow, validated with the extensively studied examples including a synthetic Müller potential and Alanine dipeptide, shows a remarkable performance.

1 INTRODUCTION

In the study of large atomic and molecular systems, the calculation of important macro statistics such as the total energy of the system or the folding probability of a protein is of fundamental importance [Tuckerman, 2010]. One may turn to Monte Carlo methods that require unbiased sampling of the equilibrium distribution. In many applications, the distribution can be expressed by the Boltzmann distribution:

$$p(\boldsymbol{r}) = \frac{1}{Z}\exp(-\mathcal{K}(\boldsymbol{r})),$$

where $\boldsymbol{r}$ is one configuration of the system, $\mathcal{K}(\boldsymbol{r})$ represents functions depending on the potential energy of the system e.g. the temperature and other thermodynamic quantities. The statistics are typically based on a sufficient observation of all important configurations. Whereas, the enumeration of these configurations is usually infeasible.

Recently, Noé et al. [2019] introduce a machine learning based Boltzmann distribution sampler, known as the Boltzmann generator. Following the idea of normalizing flows, Boltzmann generators seek an invertible mapping $F_{ZX}(z)$ from a latent space Z to the configuration space X which maps a simple Gaussian distribution to the targeted Boltzmann distribution. Unlike molecular dynamics (MD) sampling methods that require a long time simulation, Boltzmann generators can produce uncorrelated and low energy samples from different metastable states in one-shot.

Though the Boltzmann generator successfully repacks the high probability regions of the configuration space into a concentrated latent space density, its abilities to explore high energy regions and to find the transition pathways are not well justified. The synthetic experiments in Noé et al. [2019] report the feasibility of achieving transition pathways with low energy and high probabilities through mapping of the linear interpolated paths in latent space. However, there are neither theoretical results nor physical constraints to guarantee the physical meaning behind this observation. As an important concept in molecular dynamics, the transition path between metastable states provides an important description of the transition mechanism. For instance, the transition path can be used to evaluate the lowest energy barrier and the transition rate, where the rate is a good metric of materials in applications such as catalyst discovery. Meanwhile, the transition path, as an important guidance, can help to figure out the favorable condition for the transition of chemical reactions. The lack of physical interpretations of direct paths in the latent space limits the application of Boltzmann generators in transition path finding. To the best of our knowledge, however, there is no successful effort yet to improve the path finding ability of Boltzmann generators.

In this paper, an extended normalizing flow method, named

Submitted to the 38th Conference on Uncertainty in Artificial Intelligence (UAI 2022). **To be used for reviewing only**.

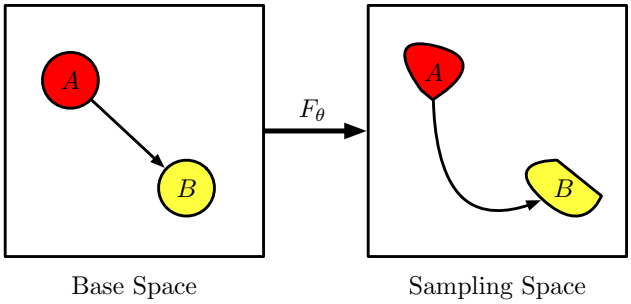

Figure 1: Illustration of PathFlow that maps the base distribution and a linear interpolated path to the Boltzmann distribution and a transition path simultaneously.

PathFlow, is proposed to improve the learning of transition paths. Beside retaining the feature of generating independent samples from the Boltzmann distribution, PathFlow further introduces physical constraints during training to regularize the mapping of linear interpolated paths between two metastable states to the *minimum energy path* (MEP) or the *minimum free energy path* (MFEP). A simple illustration of this mapping is provided in Figure 1.

Specifically, a system with two metastable states centered around A and B is considered. An invertible function F is learnt in two modes:

Learning on examples follows the general training of normalizing flows where we collect data of metastable states from MD and then train the model by minimizing the negative log-likelihood loss function L_{NF}.

Learning on paths is the main principle behind PathFlow. Following the physical definition of MEP and MFEP, another loss function L_{path} is designed to measure the ability of F mapping the linear interpolated path in the latent space to a transition path with physical meaning. On-the-fly estimators of physics quantities required in the calculation of L_{path} as well as its gradient ∇L_{path} are provided based on restraint dynamics [Maragliano et al., 2006, Maragliano and Vanden-Eijnden, 2006].

Therefore, unlike other path finding methods [Jónsson et al., 1998, Weinan et al., 2002], PathFlow can be trained by applying gradient-based methods to minimize the total loss:

$$L = w_{\text{NF}} L_{\text{NF}} + w_{\text{path}} L_{\text{path}}.$$

In the experiments based on extensively studied synthetic Müller potential and real-world Alanine dipeptide examples, a remarkable performance is achieved by PathFlow. Particularly, our contributions are summarized as below:

- Introduce physical constraints to normalizing flow which leads to a new machine learning model with knowledge of both high energy and low energy area of a system. This new model can serve as a data generator as well as a transition path finder.
- Design a loss function L_{path} to measure the performance of a transition path and provide its estimator based on restraint dynamics. Theoretical bounds of the estimation error are also provided.

2 RELATED LITERATURE

Molecular Dynamics. The first molecular dynamic simulations can be dated back to mid-20th century [Alder and Wainwright, 1957, McCammon et al., 1977]. Over the past several decades, with the fast development of computational sciences, MD has been successfully applied to physics, chemistry, biology, materials science, and several other fields. One of the greatest challenges of MD is to sample the rare events of state transitions. Enhanced sampling is thus needed to accelerate the dynamics. One line of research focuses on adding bias to the potential along predefined collective variables (CVs) to decrease the energy barrier. Such methods include, but are not limited to, the widely used umbrella sampling [Torrie and Valleau, 1977], adaptive biasing force method [Darve and Pohorille, 2001], metadynamics [Laio and Parrinello, 2002], and variational enhanced sampling [Valsson and Parrinello, 2014]. However, in many systems, proper CVs are not easily identified. Under such a situation, CV-free methods can be helpful. A number of such methods were proposed, such as parallel tempering [Swendsen and Wang, 1986], replica exchange of molecular dynamics [Sugita and Okamoto, 1999] and integrated tempering sampling [Gao, 2008].

Transition Path Finding. The study of the transition between metastable states is one of the most fundamental problems in chemistry. Existing literature such as transition state theory [Pechukas, 1981], transition path sampling [Dellago et al., 2002] and transition path theory [Vanden-Eijnden, 2006] establishes theoretical foundations to understand the mechanics of the transition. The well-known transition state theory states that the system has to navigate itself to the transition state, which is a saddle point on the potential energy surface. The most probable transition path for the reaction is the MEP. Popular methods for finding MEP include nudged elastic band (NEB) [Jónsson et al., 1998], string method [Weinan et al., 2002] and its variations [Weinan et al., 2007, Maragliano and Vanden-Eijnden, 2007, Pan et al., 2008]. Maragliano et al. [2006] extend the definition of MEP to the free energy space and modify the string method to find MFEP. After that, MFEP has been widely explored [Branduardi et al., 2007, Chen et al., 2013] and applied in different applications [Hu et al., 2007, Matsunaga et al., 2012].

Normalizing Flow. Normalizing flows (NF) are a family of generative models with tractable distributions where both sampling and density evaluation can be efficient and exact. It was popularised by Mohamed and Jimenez Rezende [2015] in the context of variational inference. Popular architectures include, but are not limited to, the planar flow, nonlinear

independent components estimation (NICE) [Dinh et al., 2014], real non-volume preserving (RealNVP [Dinh et al., 2017]), masked autoregressive flow (MAF, [Papamakarios et al., 2017]). Recent development on neural ordinary differential equations [Chen et al., 2018] extends discrete flow models to the continuous flow. Normalizing flows have been widely applied in different machine learning applications such as image generation [Ho et al., 2019], noise modelling [Abdelhamed et al., 2019], video generation [Kumar et al., 2019] and etc. Beside Boltzmann generators, normalizing flows also receive great attention in physics [Köhler et al., 2019, Kanwar et al., 2020, Wirnsberger et al., 2020, Wong et al., 2020, Wu et al., 2020]

3 MODEL

Consider a system in the NVT ensemble where the coordinates of D atoms are given by $\boldsymbol{r} = (\boldsymbol{r}_1, \boldsymbol{r}_2, ..., \boldsymbol{r}_{3D}) \in \mathbb{R}^{3D}$. The potential energy of the system is denoted by $V(\boldsymbol{r})$. It is known that $\boldsymbol{r}$ follows a Boltzmann distribution:

$$p(\boldsymbol{r}) = \frac{1}{Z}\exp(-\beta V(\boldsymbol{r})),$$

where $Z = \int_{\mathbb{R}^{3D}} \exp(-\beta V(\boldsymbol{r}))d\boldsymbol{r}$ is the partition function and $\beta = \frac{1}{\kappa_\beta T}$ is the inverse temperature. Here, κ_β is the Boltzmann constant and T is the temperature.

Suppose the system has two metastable states A and B, which, for instance, may represent the reactant and product states of a reaction. Based on MD simulation methods starting from A and B, the data $\{\boldsymbol{r}_A^i\}_{i=1}^n$ and $\{\boldsymbol{r}_B^i\}_{i=1}^n$ can be sampled. However, the transition between these two states can hardly be observed without any enhanced sampling technique, because of the high energy barrier presented in the potential energy landscape. In addition, long simulation trials are always required to achieve statistically independent samples for both metastable states.

This section describes the PathFlow model, avoiding the aforementioned challenges, to generate independent metastable states samples as well as the transition path. To achieve these two goals, the model will be trained in two modes: *learning on examples* and *learning on paths*.

3.1 LEARNING ON EXAMPLES

Given a target distribution X with probability density p_X, normalizing flows (NFs) target to find a learnable and invertible function $F_\theta : \mathbb{R}^d \mapsto \mathbb{R}^d$, usually represented by a neural network with parameter θ, that transforms a probability density Z to the target X, i.e.,$X = F_\theta(Z)$ and $Z = F_\theta^{-1}(X)$. Allowing the change of variable rule, we know that

$$p_X(x) = p_Z(F_\theta^{-1}(x))\left|\det(J_{F_\theta^{-1}}(x))\right|,$$

where $J_{F_\theta^{-1}}(x)$ is the Jacobian matrix of F_θ^{-1} at x. Given n realizations of the distribution X, $\{x_i\}_{i=1}^n$, NFs can be trained by minimizing the negative log-likelihood:

$$-\sum_{i=1}^n \log p_X(x_i) = -\sum_{i=1}^n \Big[\log p_Z(F_\theta^{-1}(x_i)) + \left|\det(J_{F_\theta^{-1}}(x_i))\right|\Big].$$

The base distribution Z is usually chosen as a uni-modal Gaussian distribution or uniform distribution. However, Cornish et al. [2020] point out that NFs can hardly map a unimodal base distribution to a multimodal distribution such as the Boltzmann distribution considered in this paper. To overcome this issue, we opt to use two separate base distributions Z_A and Z_B for states A and B, respectively. Different from Boltzmann generators using two mappings for two disconnected states, we will transform the two base distributions using the same mapping F_θ. We expect that:

$$Z_A = F_\theta^{-1}(\boldsymbol{r}_A) \quad \text{and} \quad Z_B = F_\theta^{-1}(\boldsymbol{r}_B).$$

The negative log-likelihood loss to find the best parameter θ can then be written as:

$$\begin{aligned} &L_{\mathrm{NF}}(\theta; w_A, w_B) \\ &= w_A L_{\mathrm{NF}}^A(\theta) + w_B L_{\mathrm{NF}}^B(\theta) \\ &= -w_A \sum_{i=1}^n \left[\log p_{Z_A}(F_\theta^{-1}(\boldsymbol{r}_A^i)) + \left|\det(J_{F_\theta^{-1}}(\boldsymbol{r}_A^i))\right|\right] \\ &\quad - w_B \sum_{i=1}^n \left[\log p_{Z_B}(F_\theta^{-1}(\boldsymbol{r}_B^i)) + \left|\det(J_{F_\theta^{-1}}(\boldsymbol{r}_B^i))\right|\right], \end{aligned} \tag{1}$$

where (w_A, w_B) are the weights of the two states.

3.2 LEARNING ON PATHS

To enable PathFlow to find physically meaningful transition pathways, we introduce physical constraints to the model training. Here, we are especially interested in finding the minimum energy path or the minimum free energy path.

3.2.1 Minimum Energy Path (MEP)

An MEP is a path that connects two minima of $V(\boldsymbol{r})$ via a saddle point and corresponds to the steepest descent path on $V(\boldsymbol{r})$ from this saddle point. More specifically, each point on the MEP is a local potential energy minimum on the hyperplane tangent to the path. This implies that the force $-\nabla V$ must be everywhere tangent to the MEP. Denote the MEP by a curve $\boldsymbol{r}(\alpha)$, where $\alpha \in [0, 1]$ is a parametrization of the path. We then have, for $\forall \alpha \in [0, 1]$,

$$\nabla V(\boldsymbol{r}(\alpha)) \text{ is parallel to } \frac{d\boldsymbol{r}(\alpha)}{d\alpha}, \tag{2}$$

or equivalently,

$$\nabla V(\boldsymbol{r}(\alpha)) - (\nabla V(\boldsymbol{r}(\alpha)) \cdot \hat{t})\hat{t} = 0, \tag{3}$$

where $\hat{t}$ is the unit tangent vector along the path at $\boldsymbol{r}(\alpha)$. Eq. (3) is yet a numerically efficient way to measure the performance of a path, due to the high computational cost to calculate the tangent vector. Olender and Elber [1997] instead prove that finding MEP is equivalent to solving the following variation optimization problem:

$$P_{\text{MEP}} = \operatorname*{argmin}_{P:A\to B} \int_P \|\nabla V\|_2 |dl|, \tag{4}$$

for a gradient system

$$\frac{d\boldsymbol{r}}{d\alpha} = -\nabla V(\boldsymbol{r}(\alpha)).$$

Suppose the path is divided into S segments $\{l_i\}_{i=1}^S$ with arc lengths $\{dl_i\}_{i=1}^S$. Let $-\nabla V_i$ be the force at the starting point of i-th path segment. The discretization of the optimization objective in Eq. (4) provides us an ideal loss function to measure the performance of a candidate path P:

$$L_{\text{MEP}}(P) = \sum_{i=1}^{S} \|\nabla V_i\|_2 |dl_i|. \tag{5}$$

3.2.2 Minimum Free Energy Path (MFEP)

Finding MEP has to deal with the difficulty caused by the extremely high dimensionality of the system and also the non-smoothness of the potential energy landscape. This difficulty can be reduced by the introduction of collective variables (CVs) and the mapping of MEP to the CV space (denoted as $\mathcal{X}$). Given N predefined CVs denoted by $\boldsymbol{x}(\boldsymbol{r}) = (x_1(\boldsymbol{r}), ..., x_N(\boldsymbol{r}))$, the free energy associated with $\boldsymbol{x}(\boldsymbol{r})$ is defined as follows:

$$U(\boldsymbol{z}) = -\beta^{-1} \ln \Big(Z^{-1} \int_{\mathbb{R}^{3D}} e^{-\beta V(\boldsymbol{r})} \times \prod_{i=1}^{N} \delta(x_i(\boldsymbol{r}) - z_i) d\boldsymbol{r} \Big), \forall \boldsymbol{z} \in \mathcal{X}, \tag{6}$$

where δ is the Dirac delta function. On the free energy surface, the path of our interest is the minimum free energy path (MFEP). Letting $\boldsymbol{z}(\alpha) = \boldsymbol{x}(\boldsymbol{r}(\alpha))$, Maragliano et al. [2006] show that MFEP $\boldsymbol{z}(\alpha)$ must satisfy

$$\frac{d\boldsymbol{z}(\alpha)}{d\alpha} \text{ is parallel to } M(\boldsymbol{z}(\alpha))\nabla_{\boldsymbol{z}} U(\boldsymbol{z}(\alpha)), \tag{7}$$

where

$$M_{ij}(\boldsymbol{z}) = Z^{-1} e^{\beta U(\boldsymbol{x})} \int_{\mathbb{R}^{3D}} \sum_k \frac{\partial x_i(\boldsymbol{r}(\alpha))}{\partial r_k} \frac{\partial x_j(\boldsymbol{r}(\alpha))}{\partial r_k} e^{-\beta V(\boldsymbol{r})} \prod_{i=1}^{N} (z_i - x_i(\boldsymbol{r})) d\boldsymbol{r}. \tag{8}$$

Maragliano et al. [2006] also prove that MFEP is the most likely path of transitions between A and B. Hence, it can greatly help us understand the underlying physical mechanism of the transition. Similar to Eq. (5), the following loss can be utilized to measure the performance of a candidate path on the free energy surface.

$$L_{\text{MFEP}}(P) = \sum_{i=1}^{S} \|M_i \nabla U_i\|_2 |dl_i|, \tag{9}$$

where P is a candidate path in $\mathcal{X}$ connecting A and B.

Remark 3.1. The minimum energy path can be viewed as a special case of the minimum free energy path. Specifically, if we choose $\boldsymbol{x}(\boldsymbol{r}) = \boldsymbol{r}$, i.e., an identity mapping, the free energy U is exactly the potential energy V and the transition matrix as defined in Eq. (8) is reduced to an identity matrix. Therefore, Eq. (9) is the same as Eq. (5).

3.3 TOTAL LOSS DESIGN

It is necessary to find an invertible mapping F_θ that 1) maps the base distribution to the target Boltzmann distribution and 2) maps a base path in the latent space to a transition path in the configuration or CV spaces. Given a base path P_{base}, the path after mapping is denoted as $F_\theta(P_{\text{base}})$. We can then use Eqs. (1), (5) and (9) to measure how good the parameter θ is to realize two targets. Denote the path loss as

$$L_{\text{path}}(\theta) = \begin{cases} L_{\text{MEP}}(F_\theta(P_{\text{base}})), \text{or} \\ L_{\text{MFEP}}(F_\theta(P_{\text{base}})). \end{cases} \tag{10}$$

Combining the path loss Eq. (10) with NF loss Eq. (1), we obtain the loss to train PathFlow:

$$L(\theta) = w_{\text{NF}} L_{\text{NF}}(\theta) + w_{\text{path}} L_{\text{path}}(\theta), \tag{11}$$

where w_{NF} and w_{path} are two hyper parameters to control the weight of two losses. To ease the training of our model, the base path P_{base} and base distribution Z_A, Z_B should be carefully selected. First, the end-points A and B of the transition path need to be determined. In some applications like the study of chemical reactions, the start and end of the transition path is already known. In other cases, A and B can be set in terms of the simulation data. For example, the end-points can be chosen as

$$\boldsymbol{\mu}_A, \boldsymbol{\mu}_B = \begin{cases} \frac{1}{n}\sum_i \boldsymbol{r}_A^i, \frac{1}{n}\sum_i \boldsymbol{r}_B^i, \text{ if in configuration space;} \\ \frac{1}{n}\sum_i \boldsymbol{x}(\boldsymbol{r}_A^i), \frac{1}{n}\sum_i \boldsymbol{x}(\boldsymbol{r}_B^i), \text{ if in CV space,} \end{cases} \tag{12}$$

the mean of samples from states A and B, respectively. Since the transition path must have A and B as its end-points, we require the base path starting from $F_\theta^{-1}(A)$ and ending at $F_\theta^{-1}(B)$. A natural choice of the whole base path is the linear interpolated path between these two points, i.e.,

$$P_{\text{base}}(\alpha) = (1-\alpha)F_\theta^{-1}(A) + \alpha F_\theta^{-1}(B).$$

In the sampling space, the simulation data should center around the points with minimal potential or free energy, which at the same time are the end-points of the transition path. Therefore, we prefer to set

$$Z_A \sim Gaussian(F_\theta^{-1}(A), \sigma \boldsymbol{I}),$$
$$Z_B \sim Gaussian(F_\theta^{-1}(B), \sigma \boldsymbol{I}).$$

Here, σ can be used to control the concentration of the base distribution. When σ is small, most of the linear interpolated path lies outside the concentration area of Z_A and Z_B. Hence, the model can focus on learning the path only. On the other hand, co-training of the path and the generator may be difficult around A and B, since F_θ has to minimize two losses at the same time. However, since we are most interested in the transition process that happens around the high energy barrier, we can avoid this conflict by reducing the weight of path samples near the end-points.

3.4 GRADIENT-BASED TRAINING

The gradient descent type algorithm is applied to update the model parameter θ to minimize $L(\theta)$. Notice that

$$\nabla_\theta L(\theta) = w_{\text{NF}} \nabla_\theta L_{\text{NF}}(\theta) + w_{\text{path}} \nabla_\theta L_{\text{path}}(\theta).$$

The gradient $\nabla_\theta L_{\text{NF}}(\theta)$ can be calculated by backpropagation that has already been implemented in popular deep learning frameworks such as Tensorflow [Abadi et al., 2016] and PyTorch [Paszke et al., 2019]. $\nabla_\theta L_{\text{path}}(\theta)$, however, involves the calculation of the potential mean force, the transition matrix and their gradients, and therefore cannot be calculated automatically. In the next section, we will provide efficient estimators of all the physics quantities appearing in $\nabla_\theta L_{\text{path}}(\theta)$ using restraint dynamics.

4 GRADIENT ESTIMATOR BY RESTRAINT DYNAMICS

In this section, we provide an estimator of the gradient $\nabla_\theta L_{\text{path}}(\theta)$ to facilitate the gradient-based training of our model. Since MEP can be viewed as a special case of MFEP as we mentioned in Remark 3.1, we only consider the case where $L_{\text{path}} = L_{\text{MFEP}}$. Suppose, under parameter θ, the candidate path is $P(\theta)$ and of arc length $l(P(\theta))$. We uniformly divide $P(\theta)$ into S segments, each with an arc length $|dl_i| = l(P(\theta))/S$ in Eq. (9). We then have:

$$L_{\text{MFEP}}(P) = l(P(\theta)) \frac{1}{S} \sum_{i=1}^{S} \|M_i \nabla U_i\|_2,$$

as well as the gradient:

$$\nabla_\theta L_{\text{path}}(\theta) = \nabla_\theta l(P(\theta)) \sum_{i=1}^{S} \|M_i \nabla U_i\|_2 + l(P(\theta)) \frac{1}{I} \sum_{i=1}^{S} \nabla_\theta \|M_i \nabla U_i\|_2. \quad (13)$$

The calculation of $\|M_i \nabla U_i\|_2$ and its gradient $\nabla_\theta \|M_i \nabla U_i\|_2$ can be done by regular backpropagation only if the free energy surface (FES) U or its analytical approximation are known. In practice, however, establishing FES requires a large number of simulations, which can be a task even harder than finding the path. Therefore, it would be more efficient to estimate these values on the fly at the given sample points on $P(\theta)$. We will adopt the approach of restrained dynamics [Maragliano et al., 2006, Maragliano and Vanden-Eijnden, 2006]. For a given point $\boldsymbol{z} = (z_1, ..., z_M)$ in the CV space, this method adds a harmonic restraint to the potential of the system to represent the effect of the spring forces between the configuration variables and the CVs:

$$V_k(\boldsymbol{r}; \boldsymbol{z}) = V(\boldsymbol{r}) + \frac{k}{2} \sum_{i=1}^{N} (x_i(\boldsymbol{r}) - z_i)^2, \quad (14)$$

where k is a parameter to control the restraint. The movement of particles in the CV space under this extended potential can then be characterized by the overdamped Langevin dynamics:

$$\dot{\boldsymbol{r}}(t) = -\nabla V_k(\boldsymbol{r}(t), \boldsymbol{z}) + \sqrt{2\kappa_\beta T} \eta_t, \quad (15)$$

where $\eta(t)$ is a white Gaussian noise with unit variance. It can been shown that Eq. (15) has the following Boltzmann-Gibbs density as its stationary distribution:

$$p_k(\boldsymbol{r}; \boldsymbol{z}) = \frac{1}{Z_k(\boldsymbol{z})} \exp(-\beta V_k(\boldsymbol{r}; \boldsymbol{z})),$$

where $Z_k(\boldsymbol{z}) = \int \exp(-\beta V_k(\boldsymbol{r}, \boldsymbol{z})) d\boldsymbol{r}$.

Estimation of $\|M \nabla U\|_2$. Define the effective free energy corresponding to $V_k(\boldsymbol{r}; \boldsymbol{z})$ as

$$U^{(k)}(\boldsymbol{z}) = -\beta^{-1} \ln \left(Z^{-1} \int_{\mathbb{R}^{3D}} \exp(-\beta V_k(\boldsymbol{r}; \boldsymbol{z})) d\boldsymbol{r} \right).$$

Maragliano et al. [2006] prove that when k is large,

$$\lim_{k \to \infty} \nabla U^{(k)}(\boldsymbol{z}) = \nabla U(\boldsymbol{z}),$$

where

$$\nabla_i U^{(k)} = \int_{\mathbb{R}^{3D}} k(z_i - x_i(\boldsymbol{r})) p_k(\boldsymbol{r}; \boldsymbol{z}) d\boldsymbol{r}, \ \ \forall i \leq N.$$

If we further assume the ergodicity of dynamics Eq. (15), we can obtain an estimator of the potential mean force:

$$\nabla_i U^{(T,k)}(\boldsymbol{z}) = \frac{k}{T} \int_0^T (z_i - x_i(\boldsymbol{r}(t))) dt. \quad (16)$$

Similar analysis can be done on M, and an estimator of $M_{ij}(\boldsymbol{z})$ can be derived from Eq. (8) as follows:

$$M_{ij}^{(T,k)}(\boldsymbol{z}) = \frac{1}{T}\int_0^T \sum_k \frac{\partial x_i(r(t))}{\partial r_k}\frac{\partial x_j(r(t))}{\partial r_k}dt. \quad (17)$$

Combining Eqs. (16) and (17), we obtain an estimation of $\|M\nabla U\|_2$ as $\|\nabla U^{(T,k)}M^{(T,k)}\|_2$.

Estimation of $\nabla_\theta\|M\nabla U\|_2$. To estimate $\nabla_\theta\|M\nabla U\|_2$ in the second term of Eq. (13), one naive approach is to use the finite difference method requiring at least $O(N)$ simulation trails under Eq. (15), which may be computationally challenged in practice. Instead, we propose a new estimator that can be obtained simultaneously with Eqs. (16) and (17). Specifically, we rewrite the gradient of $\|M\nabla U\|_2$ as follows:

$$\nabla_\theta\|M\nabla U\|_2 = \frac{J(M\nabla U)^\top M\nabla U}{\|M\nabla U\|_2},$$

where $J(\cdot)$ is the Jacobian matrix of a given function. Note that the Jacobian matrix can be further decomposed as

$$J(M\nabla U) = \nabla M\nabla U + M\nabla^2 U, \quad (18)$$

where $\nabla M\nabla U = [\nabla_{z_1}M\nabla U, ..., \nabla_{z_N}M\nabla U]$. Recall that the estimators Eqs. (16) and (17) can all be viewed as a time average estimation of the expectation of a function $f(\boldsymbol{r},\boldsymbol{z})$ over distribution $p_k(\boldsymbol{r};\boldsymbol{z})$, i.e., $\int_{\mathbb{R}^{3D}} f(\boldsymbol{r},\boldsymbol{z})p_k(\boldsymbol{r};\boldsymbol{z})d\boldsymbol{r}$. Specifically, for M_{ij}, $f(\boldsymbol{r},\boldsymbol{z})$ is taken as $\sum_k \frac{\partial x_i(r)}{\partial r_k}\frac{\partial x_j(r)}{\partial r_k}$ and for $\nabla_i U(\boldsymbol{z})$, $f(\boldsymbol{r},\boldsymbol{z})$ is taken as $k(z_i - x_i(\boldsymbol{r}))$. For this expectation, Maragliano et al. [2006] have proved that

$$\begin{aligned}&\lim_{k\to\infty}\int_{\mathbb{R}^{3D}} f(\boldsymbol{r},\boldsymbol{z})p_k(\boldsymbol{r};\boldsymbol{z})d\boldsymbol{r}\\ =&Z^{-1}e^{\beta U(\boldsymbol{z})}\int_{\mathbb{R}^{3D}} f(\boldsymbol{r},\boldsymbol{z})e^{-\beta V(\boldsymbol{r})}\prod_{i=1}^N \delta(z_i - x_i(\boldsymbol{r}))d\boldsymbol{r}.\end{aligned}$$

Under certain regularity conditions that we can change the order of derivative and limit, as well as the order of derivative and integration, the following equation is established.

$$\begin{aligned}&\lim_{k\to\infty}\int_{\mathbb{R}^{3D}}\frac{\partial f(\boldsymbol{r},\boldsymbol{z})p_k(\boldsymbol{r};\boldsymbol{z})}{\partial z_l}d\boldsymbol{r}\\ =&\lim_{k\to\infty}\frac{\partial\int_{\mathbb{R}^{3D}} f(\boldsymbol{r},\boldsymbol{z})p_k(\boldsymbol{r};\boldsymbol{z})d\boldsymbol{r}}{\partial z_l}\\ =&\frac{\partial Z^{-1}e^{\beta U(\boldsymbol{z})}\int_{\mathbb{R}^{3D}} f(\boldsymbol{r},\boldsymbol{z})e^{-\beta V(\boldsymbol{r})}\prod_{i=1}^N\delta(z_i - x_i(\boldsymbol{r}))d\boldsymbol{r}}{\partial z_l}.\end{aligned}$$

Estimating $\frac{\partial M}{\partial z_l}$ and $\nabla^2 U$ can both be generalized as how to use the simulation trajectory $\boldsymbol{r}(t)$ to estimate $\int_{\mathbb{R}^{3D}}\frac{\partial f(\boldsymbol{r},\boldsymbol{z})p_k(\boldsymbol{r};\boldsymbol{z})}{\partial z_l}d\boldsymbol{r}$. By some manipulation, we have

$$\begin{aligned}&\int_{\mathbb{R}^{3D}}\frac{\partial f(\boldsymbol{r},\boldsymbol{z})p_k(\boldsymbol{r};\boldsymbol{z})}{\partial z_l}d\boldsymbol{r} = \int_{\mathbb{R}^{3D}}\frac{\partial f(\boldsymbol{r},\boldsymbol{z})}{\partial z_l}p_k(\boldsymbol{r};\boldsymbol{z})d\boldsymbol{r}\\ &+\int_{\mathbb{R}^{3D}} f(\boldsymbol{r},\boldsymbol{z})\beta k(x_l(\boldsymbol{r}) - z_l)p_k(\boldsymbol{r};\boldsymbol{z})d\boldsymbol{r}\\ &-\int_{\mathbb{R}^{3D}} f(\boldsymbol{r},\boldsymbol{z})p_k(\boldsymbol{r};\boldsymbol{z})d\boldsymbol{r}\int_{\mathbb{R}^{3D}}\beta k(x_l(\boldsymbol{r}) - z_l)p_k(\boldsymbol{r};\boldsymbol{z})d\boldsymbol{r}.\end{aligned}$$

All terms are expectations under density $p_k(\boldsymbol{r};\boldsymbol{z})$. Therefore, with ergodicity, we can use time average to construct the estimator:

$$\begin{aligned}&\int_{\mathbb{R}^{3D}}\frac{\partial f(\boldsymbol{r},\boldsymbol{z})p_k(\boldsymbol{r};\boldsymbol{z})}{\partial z_l}d\boldsymbol{r} \approx \frac{1}{T}\int_{t=0}^T\frac{\partial f(\boldsymbol{r}(t),\boldsymbol{z})}{\partial z_l}dt\\ &+\frac{1}{T}\int_{t=0}^T f(\boldsymbol{r}(t))\beta k(x_l(\boldsymbol{r}(t)) - z_l)dt\\ &-\frac{1}{T}\int_{t=0}^T f(\boldsymbol{r}(t),z)dt\frac{1}{T}\int_{t=0}^T\beta k(x_l(\boldsymbol{r}(t)) - z_l)dt\\ &\triangleq \mathcal{F}_l(f(\boldsymbol{r},\boldsymbol{z}),T,k). \quad (19)\end{aligned}$$

Plugging $\sum_k \frac{\partial x_i(\boldsymbol{r})}{\partial r_k}\frac{\partial x_j(\boldsymbol{r})}{\partial r_k}$ or $k(z_j - x_j(\boldsymbol{r}))$ into $f(\boldsymbol{r},\boldsymbol{z})$, we get the estimators of $\nabla_l M_{i,j}(\boldsymbol{z})$ and $\nabla^2_{i,j}U(\boldsymbol{z})$.

$$\begin{aligned}\nabla_l M_{ij}^{(T,k)}(\boldsymbol{z}) &= \mathcal{F}_l\left(\sum_k\frac{\partial x_i(\boldsymbol{r})}{\partial r_k}\frac{\partial x_j(\boldsymbol{r})}{\partial r_k},T,k\right),\\ \nabla^2_{i,j}U^{(T,k)}(\boldsymbol{z}) &= \mathcal{F}_i\left(k(z_j - x_j(\boldsymbol{r})),T,k\right). \quad (20)\end{aligned}$$

Using Eqs. (16), (17) and (20), the approximation of the Jacobian matrix in Eq. (18) is established. We are ready to use gradient-based algorithm to find θ that optimizes $L(\theta)$.

4.1 ESTIMATION ERROR

The following theorem shows the estimation error of estimators Eqs. (16), (17) and (20).

Theorem 4.1. *Suppose the dynamics Eq.* (15) *is ergodic, for $\forall i,j,l \le N$ and $\boldsymbol{z}$ in $\mathcal{X}$, the estimation errors of $M_{ij}^{(T,k)}(\boldsymbol{z}), \nabla_i U^{(T,k)}(\boldsymbol{z}), \nabla_l M_{ij}^{(T,k)}(\boldsymbol{z}), \nabla^2_{ij}U^{(T,k)}(\boldsymbol{z})$ are as follows.*

$$\begin{aligned}&|M_{ij}^{(T,k)}(\boldsymbol{z}) - M_{ij}(\boldsymbol{z})| \le O(\frac{1}{k}) + O(\frac{1}{\sqrt{T}}),\\ &|\nabla_i U^{(T,k)}(\boldsymbol{z}) - \nabla_i U(\boldsymbol{z})| \le O(\frac{1}{k}) + O(\frac{k}{\sqrt{T}}),\\ &|\nabla_l M_{ij}^{(T,k)}(\boldsymbol{z}) - \nabla_l M_{ij}(\boldsymbol{z})| \le O(\frac{1}{k}) + O(\frac{k}{\sqrt{T}}),\\ &|\nabla^2_{ij}U^{(T,k)}(\boldsymbol{z}) - \nabla^2_{ij}U(\boldsymbol{z})| \le O(\frac{1}{k}) + O(\frac{k^2}{\sqrt{T}}).\end{aligned}$$

The proof of Theorem 4.1 can be found in Appendix A. To achieve an error of order ϵ, $M_{ij}^{(T,k)}(\boldsymbol{z})$, $\nabla_i U^{(T,k)}(\boldsymbol{z})$

and $\nabla_l M_{ij}^{(T,k)}(\boldsymbol{z})$ require at most $T = O(1/\epsilon^4)$, while $\nabla_{ij}^2 U^{(T,k)}(\boldsymbol{z})$ requires $T = O(1/\epsilon^6)$. This is consistent with our empirical observation that using $\nabla_{ij}^2 U^{(T,k)}(\boldsymbol{z})$ to estimate $\nabla^2 U$ can be statistically unstable which leads to the high variance of the whole Jacobian matrix estimation.

To overcome this issue, we propose a method that uses one more simulation trial to avoid estimation of $\nabla^2 U$. Note that by Eq. (18), $\nabla_\theta \|M\nabla U\|_2$ can be decomposed as

$$\nabla_\theta \|M\nabla U\|_2 = \frac{(\nabla M \nabla U)^\top M \nabla U}{\|M\nabla U\|_2} + \nabla^2 U \frac{M^\top M \nabla U}{\|M\nabla U\|_2}.$$

The second order term $\nabla^2 U$ appears in the second term in the form of a Hessian-vector product, which can be estimated directly with one additional simulation trial independently of N. Specifically, let $v = \frac{M^\top M \nabla U}{\|M\nabla U\|_2}$ and we have:

$$\nabla^2 U v \approx \frac{\nabla U(\boldsymbol{z} + \delta v) - \nabla U(\boldsymbol{z})}{\delta}.$$

Only one extra restraint simulation centered at $z + \delta v$ is required to get the estimate. Moreover, to increase stability, the product can also be estimated by central difference.

$$\nabla^2 U v \approx \frac{\nabla U(\boldsymbol{z} + \delta v) - \nabla U(\boldsymbol{z} - \delta v)}{2\delta},$$

By using Hessian-vector product trick, we obtain a new estimation of the second term.

$$\nabla^2 U \frac{M^\top M \nabla U}{\|M\nabla U\|_2} \approx \frac{\nabla U^{(T,k)}(\boldsymbol{z} + \delta v^{(T,k)}(\boldsymbol{z}))}{2\delta} - \frac{\nabla U^{(T,k)}(\boldsymbol{z} - \delta v^{(T,k)}(\boldsymbol{z}))}{2\delta},$$

where $v^{(T,k)}(\boldsymbol{z}) = \frac{(M^{(T,k)})^\top M^{(T,k)} \nabla U^{(T,k)}(\boldsymbol{z})}{\|M^{(T,k)} \nabla U^{(T,k)}(\boldsymbol{z})\|_2}$. Empirically, we find that using this trick can greatly stabilize the estimation with an acceptable simulation budget increment. For more detailed error estimation, please refer to Appendix B.

5 NUMERICAL EXAMPLE: MÜLLER POTENTIAL

We first illustrate PathFlow using a two-dimensional Müller potential that has metastable states separated by high energy barriers. The Müller potential has an explicit formulation:

$$V(x, y) = \sum_{k=1}^{4} A_k e^{B_k}, \tag{21}$$

where we take

$$B_k = a_k(x - x_k^0)^2 + b_k(x - x_k^0)(y - y_k^0) + c_k(y - y_k^0)^2.$$

Values of all parameters can be found in Appendix C. The two metastable states of Müller potential are located around

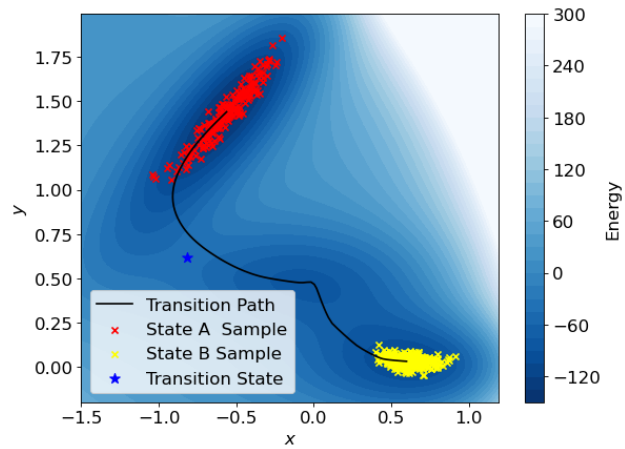

Figure 2: Experiment result on Müller Potential. PathFlow generates samples filling in two low energy regions. At the same time, the transition path found passes near the transition state. The energy barrier we found has energy of -38 which is very close to the ground-truth value -40.

$A = [-0.56, 1.44]$ and $B = [-0.05, 0.47]$, while the transition state is located around $C = [-0.82, 0.62]$. For simplicity, we consider finding the minimum energy path (MEP) starting from state A and ending at state B. We collect 100 data points using Markov Chain Monte Carlo starting from A and B respectively for learning on examples. Our normalizing flow is a masked autoregressive flow (MAF) model with 10 autoregressive layers and hidden units of shape $[256, 128, 64]$ with ReLU activation.

Given the explicit formulation of $V(x, y)$, there is no need of estimating the gradient of $L(\theta)$ using the proposed method in Section 4. All the gradients can be automatically obtained by backpropagation implemented in Tensorflow 2.3. We train the model by Adam optimizer. As shown in figure 2, PathFlow can learn the transition path and the sampler of metastable states at the same time. 1) In terms of path finding, PathFlow finds a transition path that passes the transition state C. The optimal energy barrier has energy around -40. The energy barrier we found is around -38 which is very close to the ground-truth. 2) In terms of sample generator, we can successfully generate data points for metastable states in one-shot.

6 NUMERICAL EXAMPLE: ALANINE DIPEPTIDE

In this section, we provide a practical example to illustrate the performance of our proposed models.

Experiment Setup We study the isomerization transition and sampling of Alanine dipeptide modeled by the CHARMM27 force field [Brooks et al., 2009] at 300 K in vacuum. This transition happens between two metastable

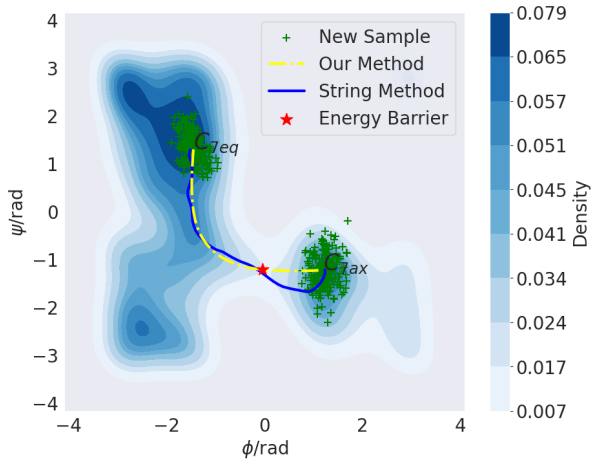

Figure 3: Experiment result on Alanine dipeptide in vacuum under room temperature 300K. The under-layer density plot is the kernel density estimation of the Boltzmann Distribution generated by Meta Dynamics. 1)Transition pathways found by PathFlow and string method overlap in most regions. The energy barrier with the energy of about 8.6 kcal/mol lies on both paths. 2) Our model can generate $C7_{eq}$ and $C7_{ax}$ data.

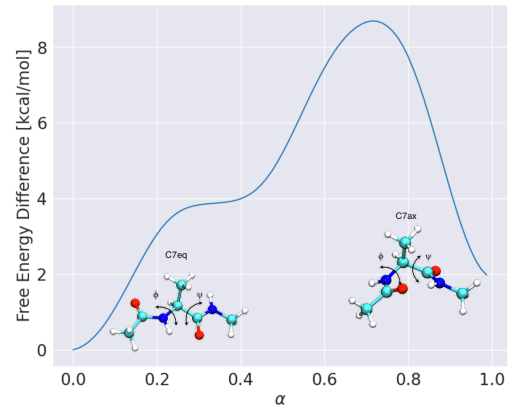

Figure 4: Free energy profile of the transition pathway found by PathFlow. Free energy in C_{7eq} ($\alpha = 0$) is set as 0. The configuration plots were made by Cuny et al. [2017].

states named C_{7eq} and C_{7ax}. We choose two torsion angles $\phi(C, N, C_\alpha, C)$ and $\psi(N, C_\alpha, C, N)$ as our CVs for this system, i.e., $\boldsymbol{z} = (\phi, \psi)$. All the MD simulations are performed by the package GROMACS 2021 [Lindahl et al., 2021] linked with Plumed 2.7 [Tribello et al., 2014]. To generate data in two metastable states, we run brute-force MD simulations starting from C_{7eq} and C_{7ax} for 100 picoseconds (ps), respectively. The CV values along the MD trajectories are computed and recorded in every 0.2 ps. We randomly select 100 data points for each state to train the sampler. On each candidate path in the CV space, we sample a point every 0.1 arc length. For each sample on the path, we run three restraint simulations with $k = 500$ kJ/mol/rad for 100 ps. The CV values along the trajectories are computed and recorded in every 0.01 ps to estimate the potential mean force, transformation matrix M, and their derivative.

We choose a masked autoregressive flow with 15 autoregressive layers and hidden units of shape $[512, 256, 128, 64]$ with ELU activation as our normalizing flow model. As a comparison, we also implement the string method with swarms of trajectory using a string with 40 images. Other setting-up follows that in Pan et al. [2008] Section III.1.

Numerical Results. Figure 3 plots the transition pathways found by the string method (average of 60-70 iterations) and PathFlow, as well as the new metastable states samples generated by PathFlow. We have the following observations. First, transition paths found by PathFlow and the string method overlap in most regions. They all pass the same energy barrier with free energy difference of 8.6 kcal/mol. The two pathways differ around C_{7ax} which may be caused by the conflict between L_{NF} and L_{path} during training. However, the free energy profile of our pathway in Figure 4 is almost consistent with that of the string method in Pan et al. [2008]. Second, in terms of generator, our newly generated samples locate in two low energy regions. Since the generator is trained using short simulation data which may not contain all metastable configurations, our generator can not sample these unknown low energy regions. This issue can be easily solved by using data from longer simulation trials. In terms of speed of path finding, PathFlow can find the path in 5 iterations requiring 60 ns restraint simulation, which is significantly faster than the string method that converges in about 60 iterations and takes 84 ns simulation (60 ns restraint simulation, 24 ns free simulation).

7 CONCLUSION AND PERSPECTIVE

In summary, PathFlow is a promising tool for generating Boltzmann samples and discovering transition paths to describe the transition mechanisms. Different from existing path finding algorithms (e.g.,NEB [Jónsson et al., 1998], string method [Weinan et al., 2002]), PathFlow is trained by the standard gradient-based optimizers associating with the efficient gradient estimator developed in section 4. Note that the estimator has the potential to be employed by other machine learning based path finding algorithms. In particular, as an independent research interest, it is empirically found that the gradient-based training leads to a faster path finding speed and fewer simulation trials. In addition, PathFlow can be viewed as one successful application of multitask learning to physics. We expect more multitask learning techniques will demonstrate their power in scientific research. Future research directions also include normalizing flows or other machine learning based methods in the transition tube [Vanden-Eijnden, 2006] sampling as well as CV discovery.

REFERENCES

Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016. URL `https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf`.

Abdelrahman Abdelhamed, Marcus A Brubaker, and Michael S Brown. Noise flow: Noise modeling with conditional normalizing flows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3165–3173, 2019.

B J Alder and T E Wainwright. Phase transition for a hard sphere system. *Journal of Chemical Physics (U.S.)*, 27, 11 1957. doi: 10.1063/1.1743957. URL `https://www.osti.gov/biblio/4322875`.

Davide Branduardi, Francesco Luigi Gervasio, and Michele Parrinello. From a to b in free energy space. *The Journal of chemical physics*, 126(5):054103, 2007.

Bernard R Brooks, Charles L Brooks III, Alexander D Mackerell Jr, Lennart Nilsson, Robert J Petrella, Benoît Roux, Youngdo Won, Georgios Archontis, Christian Bartels, Stefan Boresch, et al. Charmm: the biomolecular simulation program. *Journal of computational chemistry*, 30(10): 1545–1614, 2009.

Changjun Chen, Yanzhao Huang, Xiaofeng Ji, and Yi Xiao. Efficiently finding the minimum free energy path from steepest descent path. *The Journal of chemical physics*, 138(16):164122, 2013.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Rob Cornish, Anthony Caterini, George Deligiannidis, and Arnaud Doucet. Relaxing bijectivity constraints with continuously indexed normalising flows. In *International conference on machine learning*, pages 2133–2143. PMLR, 2020.

Jerome Cuny, Kseniia Korchagina, Chemseddine Menakbi, and Tzonka Mineva. Metadynamics combined with auxiliary density functional and density functional tight-binding methods: alanine dipeptide as a case study. *Journal of molecular modeling*, 23(3):1–8, 2017.

Eric Darve and Andrew Pohorille. Calculating free energies using average force. *The Journal of chemical physics*, 115(20):9169–9183, 2001.

Christoph Dellago, Peter Bolhuis, Phillip L Geissler, et al. Transition path sampling. *Advances in chemical physics*, 123(1), 2002.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. 2017. URL `https://arxiv.org/abs/1605.08803`.

Yi Qin Gao. An integrate-over-temperature approach for enhanced sampling. *The Journal of chemical physics*, 128 (6):064105, 2008.

Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International Conference on Machine Learning*, pages 2722–2730. PMLR, 2019.

Hao Hu, Zhenyu Lu, and Weitao Yang. Qm/mm minimum free-energy path: Methodology and application to triosephosphate isomerase. *Journal of chemical theory and computation*, 3(2):390–406, 2007.

Hannes Jónsson, Greg Mills, and Karsten W Jacobsen. Nudged elastic band method for finding minimum energy paths of transitions. 1998.

Gurtej Kanwar, Michael S Albergo, Denis Boyda, Kyle Cranmer, Daniel C Hackett, Sébastien Racaniere, Danilo Jimenez Rezende, and Phiala E Shanahan. Equivariant flow-based sampling for lattice gauge theory. *Physical Review Letters*, 125(12):121601, 2020.

Jonas Köhler, Leon Klein, and Frank Noé. Equivariant flows: sampling configurations for multi-body systems with symmetric energies. *arXiv preprint arXiv:1910.00753*, 2019.

Manoj Kumar, Mohammad Babaeizadeh, Dumitru Erhan, Chelsea Finn, Sergey Levine, Laurent Dinh, and Durk Kingma. Videoflow: A flow-based generative model for video. *arXiv preprint arXiv:1903.01434*, 2(5), 2019.

Alessandro Laio and Michele Parrinello. Escaping free-energy minima. *Proceedings of the National Academy of Sciences*, 99(20):12562–12566, 2002.

Lindahl, Abraham, Hess, and van der Spoel. Gromacs 2021 source code, January 2021. URL `https://doi.org/10.5281/zenodo.4457626`.

Luca Maragliano and Eric Vanden-Eijnden. A temperature accelerated method for sampling free energy and determining reaction pathways in rare events simulations. *Chemical physics letters*, 426(1-3):168–175, 2006.

Luca Maragliano and Eric Vanden-Eijnden. On-the-fly string method for minimum free energy paths calculation. *Chemical physics letters*, 446(1-3):182–190, 2007.

Luca Maragliano, Alexander Fischer, Eric Vanden-Eijnden, and Giovanni Ciccotti. String method in collective variables: Minimum free energy paths and isocommittor surfaces. *The Journal of chemical physics*, 125(2):024106, 2006.

Yasuhiro Matsunaga, Hiroshi Fujisaki, Tohru Terada, Tadaomi Furuta, Kei Moritsugu, and Akinori Kidera. Minimum free energy path of ligand-induced transition in adenylate kinase. *PLoS computational biology*, 8(6): e1002555, 2012.

J Andrew McCammon, Bruce R Gelin, and Martin Karplus. Dynamics of folded proteins. *nature*, 267(5612):585–590, 1977.

Shakir Mohamed and Danilo Jimenez Rezende. Variational information maximisation for intrinsically motivated reinforcement learning. *Advances in neural information processing systems*, 28, 2015.

Frank Noé, Simon Olsson, Jonas Köhler, and Hao Wu. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365 (6457):eaaw1147, 2019.

Roberto Olender and Ron Elber. Yet another look at the steepest descent path. *Journal of Molecular Structure: THEOCHEM*, 398:63–71, 1997.

Albert C Pan, Deniz Sezer, and Benoît Roux. Finding transition pathways using the string method with swarms of trajectories. *The journal of physical chemistry B*, 112(11): 3432–3440, 2008.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *Advances in neural information processing systems*, 30, 2017.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Philip Pechukas. Transition state theory. *Annual Review of Physical Chemistry*, 32(1):159–177, 1981.

Yuji Sugita and Yuko Okamoto. Replica-exchange molecular dynamics method for protein folding. *Chemical physics letters*, 314(1-2):141–151, 1999.

Robert H Swendsen and Jian-Sheng Wang. Replica monte carlo simulation of spin-glasses. *Physical review letters*, 57(21):2607, 1986.

Glenn M Torrie and John P Valleau. Nonphysical sampling distributions in monte carlo free-energy estimation: Umbrella sampling. *Journal of Computational Physics*, 23 (2):187–199, 1977.

Gareth A Tribello, Massimiliano Bonomi, Davide Branduardi, Carlo Camilloni, and Giovanni Bussi. Plumed 2: New feathers for an old bird. *Computer Physics Communications*, 185(2):604–613, 2014.

Mark Tuckerman. *Statistical mechanics: theory and molecular simulation*. Oxford university press, 2010.

Omar Valsson and Michele Parrinello. Variational approach to enhanced sampling and free energy calculations. *Physical review letters*, 113(9):090601, 2014.

Eric Vanden-Eijnden. Transition path theory. In *Computer Simulations in Condensed Matter Systems: From Materials to Chemical Biology Volume 1*, pages 453–493. Springer, 2006.

E Weinan, Weiqing Ren, and Eric Vanden-Eijnden. String method for the study of rare events. *Physical Review B*, 66(5):052301, 2002.

E Weinan, Weiqing Ren, and Eric Vanden-Eijnden. Simplified and improved string method for computing the minimum energy paths in barrier-crossing events. *Journal of Chemical Physics*, 126(16):164103, 2007.

Peter Wirnsberger, Andrew J Ballard, George Papamakarios, Stuart Abercrombie, Sébastien Racanière, Alexander Pritzel, Danilo Jimenez Rezende, and Charles Blundell. Targeted free energy estimation via learned mappings. *The Journal of Chemical Physics*, 153(14):144112, 2020.

Kaze WK Wong, Gabriella Contardo, and Shirley Ho. Gravitational-wave population inference with deep flow-based generative network. *Physical Review D*, 101(12): 123005, 2020.

Hao Wu, Jonas Köhler, and Frank Noé. Stochastic normalizing flows. *Advances in Neural Information Processing Systems*, 33:5933–5944, 2020.

A PROOF OF THEOREM 4.1

For any function $F \in \{M_{ij}, \nabla_i U, \nabla_l M_{ij}, \nabla^2_{ij} U\}$, the error of the estimator can be decomposed by triangle inequality as follows,

$$|F^{(T,k)}(\boldsymbol{z}) - F(\boldsymbol{z})| \leq |F^{(k)}(\boldsymbol{z}) - F(\boldsymbol{z})| + |F^{(T,k)}(\boldsymbol{z}) - F^{(k)}(\boldsymbol{z})|.$$

The first term is introduced by the finiteness of k. The second term is introduced by finiteness of the time T of restrained dynamics. The following two lemmas provide upper bounds for finite-k error and finite-T error respectively.

Lemma A.1 (Error by finite k)**.** *For any function $g(\boldsymbol{r})$, consider the two functionals $\mathcal{I}[g]$ and $\mathcal{I}^{(k)}[g]$ defined as follows*

$$\mathcal{I}[g](\boldsymbol{z}) = \int_{\mathbb{R}^{3D}} g(\boldsymbol{r}) e^{-\beta V(\boldsymbol{r})} \prod_{j=1}^{N} \delta(x_j(\boldsymbol{r}) - z_j) d\boldsymbol{r},$$
$$\mathcal{I}^{(k)}[g](\boldsymbol{z}) = (\frac{2\pi}{\beta k})^{N/2} \int_{\mathbb{R}^{3D}} g(\boldsymbol{r}) e^{-\beta(V(\boldsymbol{r}) + \frac{k}{2}\sum_{j=1}^{N}(x_j(\boldsymbol{r}) - z_j)^2)} d\boldsymbol{r}.$$

Their difference are bounded by

$$|\mathcal{I}[g](\boldsymbol{z}) - \mathcal{I}^{(k)}[g](\boldsymbol{z})| \leq \frac{1}{2\beta k} Tr[\nabla^2_z \mathcal{I}[g](\boldsymbol{z})].$$

Moreover, the difference of the derivatives of $\mathcal{I}[g](\boldsymbol{z})$ and $\mathcal{I}^{(k)}[g](\boldsymbol{z})$ are bounded by

$$|\frac{\partial^p}{\partial z_{i_1}, \ldots, \partial z_{i_p}} \mathcal{I}[g](\boldsymbol{z}) - \frac{\partial^p}{\partial z_{i_1}, \ldots, \partial z_{i_p}} \mathcal{I}^{(k)}[g](\boldsymbol{z})| \leq \frac{1}{2\beta k} \frac{\partial^p}{\partial z_{i_1}, \ldots, \partial z_{i_p}} Tr[\nabla^2_z \mathcal{I}[g](\boldsymbol{z})].$$

Proof. The proof of the upper bound of $|\mathcal{I}[g](\boldsymbol{z}) - \mathcal{I}^{(k)}[g](\boldsymbol{z})|$ follows Maragliano et al. [2006]. We generate the proof to the upper bound of the derivatives. Consider the Fourier transform of $\mathcal{I}[g](\boldsymbol{z})$ as follows

$$\hat{G}(\zeta) = \int_{\mathbb{C}^N} e^{-i\zeta\cdot\boldsymbol{z}} \int_{\mathbb{R}^{3D}} g(\boldsymbol{r}) e^{-\beta V(\boldsymbol{r})} \prod_{j=1}^{N} \delta(x_j(\boldsymbol{r}) - z_j) d\boldsymbol{r} d\boldsymbol{z} = \int_{\mathbb{R}^{3D}} g(\boldsymbol{r}) e^{-\beta V(\boldsymbol{r})} \int_{\mathbb{C}^N} e^{-i\zeta\cdot\boldsymbol{z}} \prod_{j=1}^{N} \delta(x_j(\boldsymbol{r}) - z_j) d\boldsymbol{z} d\boldsymbol{r}$$
$$= \int_{\mathbb{R}^{3D}} g(\boldsymbol{r}) e^{-\beta V(\boldsymbol{r})} e^{-i\zeta\cdot\boldsymbol{x}(\boldsymbol{r})} d\boldsymbol{r}.$$

The Fourier transform of $\mathcal{I}^{(k)}[g](\boldsymbol{z})$ is

$$\hat{G^{(k)}}(\zeta) = (\frac{2\pi}{\beta k})^{N/2} \int_{\mathbb{C}^N} e^{-i\zeta\cdot\boldsymbol{z}} \int_{\mathbb{R}^{3D}} g(\boldsymbol{r}) e^{-\beta(V(\boldsymbol{r}) + \frac{k}{2}\sum_{j=1}^{N}(x_j(\boldsymbol{r}) - z_j)^2)} d\boldsymbol{r} d\boldsymbol{z}$$
$$= (\frac{2\pi}{\beta k})^{N/2} \int_{\mathbb{R}^{3D}} g(\boldsymbol{r}) e^{-\beta V(\boldsymbol{r})} \int_{\mathbb{C}^N} \exp\{-i\zeta\cdot\boldsymbol{z} - \frac{\beta k}{2}\sum_{j=1}^{N}(x_j(\boldsymbol{r}) - z_j)^2\} d\boldsymbol{z} d\boldsymbol{r}$$
$$= (\frac{2\pi}{\beta k})^{N/2} \int_{\mathbb{R}^{3D}} g(\boldsymbol{r}) e^{-\beta V(\boldsymbol{r})} \int_{\mathbb{C}^N} \exp\{-\frac{\beta k}{2}\sum_{j=1}^{N}(z_j - x_j(\boldsymbol{r}) + \frac{i}{\beta k}\zeta_j)^2 - i\zeta\cdot\boldsymbol{x}(\boldsymbol{r}) - \frac{|\zeta|^2}{2\beta k}\} d\boldsymbol{z} d\boldsymbol{r}$$
$$= \int_{\mathbb{R}^{3D}} g(\boldsymbol{r}) e^{-\beta V(\boldsymbol{r})} e^{-i\zeta\cdot\boldsymbol{x}(\boldsymbol{r}) - \frac{|\zeta|^2}{2\beta k}} d\boldsymbol{r} = e^{-\frac{|\zeta|^2}{2\beta k}} \hat{G}(\zeta).$$

By applying reverse Fourier transformation, we have

$$|\mathcal{I}[g](\boldsymbol{z}) - \mathcal{I}^{(k)}[g](\boldsymbol{z})| = \Big| \int_{\mathbb{C}^N} e^{i\zeta\cdot\boldsymbol{z}} (\hat{G}(\zeta) - \hat{G^{(k)}}(\zeta)) d\zeta \Big| \leq \int_{\mathbb{C}^N} e^{i\zeta\cdot\boldsymbol{z}} \hat{G}(\zeta) |1 - e^{-\frac{|\zeta|^2}{2\beta k}}| d\zeta$$
$$\leq \int_{\mathbb{C}^N} e^{i\zeta\cdot\boldsymbol{z}} \hat{G}(\zeta) \frac{|\zeta|^2}{2\beta k} d\zeta = \frac{1}{2\beta k} Tr[\nabla_z \mathcal{I}^{(k)}[g](\boldsymbol{z})].$$

To generalize the upper bound to the derivatives, notice the Fourier transform of the derivatives $\frac{\partial^p}{\partial z_{i_1},\cdots,\partial z_{i_p}}\mathcal{I}[g](\boldsymbol{z})$ and $\frac{\partial^p}{\partial z_{i_1},\ldots,\partial z_{i_p}}\mathcal{I}^{(k)}[g](\boldsymbol{z})$ are $i^p\zeta_{i_1}\ldots\zeta_{i_p}\hat{G}(\zeta)$ and $i^p\zeta_{i_1}\cdots\zeta_{i_p}\hat{G}_k(\zeta)$, respectively. Similarly, applying reverse Fourier transformation

$$
\begin{aligned}
&|\frac{\partial^p}{\partial z_{i_1},\ldots,\partial z_{i_p}}\mathcal{I}[g](\boldsymbol{z})-\frac{\partial^p}{\partial z_{i_1},\ldots,\partial z_{i_p}}\mathcal{I}^{(k)}[g](\boldsymbol{z})|\leq\int_{\mathbb{C}^N}e^{i\zeta\cdot\boldsymbol{z}}i^p\zeta_{i1}\cdots\zeta_{i_p}\hat{G}(\zeta)|1-e^{-\frac{|\zeta|^2}{2\beta k}}|d\zeta\\
&\leq\int_{\mathbb{C}^N}e^{i\zeta\cdot\boldsymbol{z}}i^p\zeta_{i1}\cdots\zeta_{i_p}\hat{G}(\zeta)\frac{|\zeta|^2}{2\beta k}d\zeta=\frac{1}{2\beta k}\frac{\partial^p}{\partial z_{i_1},\ldots,\partial z_{i_p}}Tr[\nabla_z^2\mathcal{I}[g](\boldsymbol{z})].
\end{aligned}
$$

□

Lemma A.2 (Error by finite T [Maragliano et al., 2006])**.** *For any function $f(\boldsymbol{r},\boldsymbol{z})$, consider the true average functional and the time average estimator defined as follows*

$$
\begin{aligned}
\mathcal{A}^{(k)}[f](\boldsymbol{z})&=\int_{\mathbb{R}^{3N}}f(\boldsymbol{r},\boldsymbol{z})p_k(\boldsymbol{r},\boldsymbol{z})d\boldsymbol{r},\\
\mathcal{A}^{(T,k)}[f](\boldsymbol{z})&=\frac{1}{T}\int_0^T f(\boldsymbol{r}(t),\boldsymbol{z})dt.
\end{aligned}
$$

As $T\to\infty$, their difference is given by

$$
\mathcal{A}^{(T,k)}[f](\boldsymbol{z})-\mathcal{A}^{(k)}[f](\boldsymbol{z})->\sqrt{\frac{\tau_k[f](\boldsymbol{z})}{T}}\xi_k[f](\boldsymbol{z}),
$$

where $\xi_k[f](\boldsymbol{z})$ is a Gaussian variable with mean zero and variance

$$
Var[\xi_k[f](\boldsymbol{z})]=\int_{\mathbb{R}^{3N}}(f(\boldsymbol{r},\boldsymbol{z})-\mathcal{A}^{(k)}[f](\boldsymbol{z}))^2p_k(\boldsymbol{r},\boldsymbol{z})d\boldsymbol{r},
$$

and $\tau_k[f](\boldsymbol{z})$ is given by

$$
\tau_k[f](\boldsymbol{z})=\frac{1}{Var[\xi_k[f](\boldsymbol{z})]}\int_{t=0}^T\int_{\mathbb{R}^{3N}}\mathbb{E}[f(\boldsymbol{r}(t),\boldsymbol{z})-\mathcal{A}^{(k)}[f](\boldsymbol{z})](f(\boldsymbol{r},\boldsymbol{z})-\mathcal{A}^{(k)}[f](\boldsymbol{z}))^2p_k(\boldsymbol{r},\boldsymbol{z})d\boldsymbol{r}dt.
$$

Moreover, as k goes to infinity, $Var[\xi_k[f](\boldsymbol{z})]=Var[\xi[f](\boldsymbol{z})]+O(1/k)$ and $\tau_k[f](\boldsymbol{z})=\tau[f](\boldsymbol{z})+O(1/\sqrt{k})$, where $\xi[f]$ and $\tau[f]$ are defined by replacing $\mathcal{A}^{(k)}[f]$ by its limiting functional

$$
\mathcal{A}[f](\boldsymbol{z})=Z^{-1}e^{\beta U(\boldsymbol{z})}\int_{\mathbb{R}^{3N}}f(\boldsymbol{r},\boldsymbol{z})e^{-\beta V(\boldsymbol{r})}\prod_{j=1}^N\delta(z_j-x_j(\boldsymbol{r}))d\boldsymbol{r}.
$$

Now we are ready to given upper bounds for the errors of estimators of $\nabla U(\boldsymbol{z})$, $M(\boldsymbol{z})$, $\nabla M(\boldsymbol{z})$ and $\nabla^2U(\boldsymbol{z})$ by applying Lemma A.1 and A.2 respectively.

A.1 ERROR OF $M(\boldsymbol{z})$

Define $f_{ij}(\boldsymbol{r})=\sum_k\frac{\partial x_i(r(\alpha))}{\partial r_k}\frac{\partial x_j(r(\alpha))}{\partial r_k}$, and $\mathbf{1}(\boldsymbol{r})=1$. Then $M_{ij}(\boldsymbol{z})$ can be written as

$$
\begin{aligned}
M_{ij}(\boldsymbol{z})&=Z^{-1}e^{\beta U(\boldsymbol{z})}\int_{\mathbb{R}^{3D}}\sum_k\frac{\partial x_i(\boldsymbol{r}(\alpha))}{\partial r_k}\frac{\partial x_j(\boldsymbol{r}(\alpha))}{\partial r_k}e^{-\beta V(\boldsymbol{r})}\prod_{i=1}^N(z_i-x_i(\boldsymbol{r}))d\boldsymbol{r}\\
&=\frac{\int_{\mathbb{R}^{3D}}f_{ij}(\boldsymbol{r})e^{-\beta V(\boldsymbol{r})}\prod_{i=1}^N(z_i-x_i(\boldsymbol{r}))d\boldsymbol{r}}{\int_{\mathbb{R}^{3D}}e^{-\beta V(\boldsymbol{r})}\prod_{i=1}^N(z_i-x_i(\boldsymbol{r}))d\boldsymbol{r}}=\frac{\mathcal{I}[f_{ij}](\boldsymbol{z})}{\mathcal{I}[\mathbf{1}](\boldsymbol{z})}.
\end{aligned}
$$

Therefore, the finite-k error $|M_{ij}(\boldsymbol{z}) - M_{ij}^{(k)}(\boldsymbol{z})|$ is bounded by

$$
\begin{aligned}
&|M_{ij}(\boldsymbol{z}) - M_{ij}^{(k)}(\boldsymbol{z})| = |\frac{\mathcal{I}[f_{ij}](\boldsymbol{z})}{\mathcal{I}[\mathbf{1}](\boldsymbol{z})} - \frac{\mathcal{I}^{(k)}[f_{ij}](\boldsymbol{z})}{\mathcal{I}^{(k)}[\mathbf{1}](\boldsymbol{z})}| \\
&\leq \frac{1}{2\beta k}\frac{\mathcal{I}[f_{ij}](\boldsymbol{z})Tr[\nabla^2\mathcal{I}[\mathbf{1}](\boldsymbol{z})] + Tr[\nabla^2\mathcal{I}[f_{ij}](\boldsymbol{z})]\mathcal{I}[\mathbf{1}](\boldsymbol{z})}{(\mathcal{I}[\mathbf{1}](\boldsymbol{z}))^2} + O(\frac{1}{k^2}).
\end{aligned}
$$

The finite-T error is bounded by

$$
\begin{aligned}
&|M_{ij}^{(T,k)}(\boldsymbol{z}) - M_{ij}^{(k)}(\boldsymbol{z})| = |\frac{1}{T}\int_0^T f_{ij}(\boldsymbol{z})dt - \int_{\mathbb{R}^{3D}} f_{ij}(\boldsymbol{z})p_k(\boldsymbol{r},\boldsymbol{z})d\boldsymbol{r}| \\
&= |\mathcal{A}^{(T,k)}[f_{ij}](\boldsymbol{z}) - \mathcal{A}_k[f_{ij}](\boldsymbol{z})| \to \sqrt{\frac{\tau[f_{ij}](\boldsymbol{z})}{T}}\xi_k[f_{ij}](\boldsymbol{z}).
\end{aligned}
$$

A.2 ERROR OF $\nabla U(\boldsymbol{z})$

$\nabla_i U(\boldsymbol{z})$ can be written as

$$
\begin{aligned}
\nabla_i U(\boldsymbol{z}) &= -\beta^{-1}\nabla_i \ln\big(Z^{-1}\int_{\mathbb{R}^{3D}} \exp(-\beta V(\boldsymbol{r}))\prod_{j=1}^N \delta(x_j(\boldsymbol{r}) - z_j)d\boldsymbol{r}\big) \\
&= -\beta^{-1}\nabla_i \ln \mathcal{I}[\mathbf{1}](\boldsymbol{z}) = -\beta^{-1}\frac{\nabla_i\mathcal{I}[\mathbf{1}](\boldsymbol{z})}{\mathcal{I}[\mathbf{1}](\boldsymbol{z})}.
\end{aligned}
$$

The finite-k error is bounded by

$$
\begin{aligned}
&|\nabla_i U(\boldsymbol{z}) - \nabla_i U^{(k)}(\boldsymbol{z})| = \beta^{-1}|\frac{\nabla_i\mathcal{I}[\mathbf{1}](\boldsymbol{z})}{\mathcal{I}[\mathbf{1}](\boldsymbol{z})} - \frac{\nabla_i\mathcal{I}^{(k)}[\mathbf{1}](\boldsymbol{z})}{\mathcal{I}^{(k)}[\mathbf{1}](\boldsymbol{z})}| \\
&\leq \frac{1}{2\beta^2 k}\frac{|\nabla_i\mathcal{I}[\mathbf{1}](\boldsymbol{z})|Tr[\nabla^2\mathcal{I}[\mathbf{1}](\boldsymbol{z})] + |\nabla_i Tr[\nabla^2\mathcal{I}[\mathbf{1}](\boldsymbol{z})]|\mathcal{I}[\mathbf{1}](\boldsymbol{z})}{(\mathcal{I}[\mathbf{1}](\boldsymbol{z}))^2} + O(\frac{1}{k^2}).
\end{aligned}
$$

The finite-T error is bounded by

$$
\begin{aligned}
&|\nabla_i U^{(T,k)}(\boldsymbol{z}) - \nabla_i U^{(k)}(\boldsymbol{z})| = |\frac{k}{T}\int_0^T (z_i - x_i(\boldsymbol{r}(t)))dt - \int_{\mathbb{R}^{3D}} k(z_i - x_i(\boldsymbol{r}))p_k(\boldsymbol{r},\boldsymbol{z})d\boldsymbol{r}| \\
&= k|\mathcal{A}^{(T,k)}[z_i - x_i(\boldsymbol{r})](\boldsymbol{z}) - \mathcal{A}_k[z_i - x_i(\boldsymbol{r})](\boldsymbol{z})| \to k\sqrt{\frac{\tau[z_i - x_i(\boldsymbol{r})](\boldsymbol{z})}{T}}\xi_k[z_i - x_i(\boldsymbol{r})](\boldsymbol{z}).
\end{aligned}
$$

A.3 ERROR OF $\nabla M(\boldsymbol{z})$

$\nabla_l M_{ij}(\boldsymbol{z})$ can be written as

$$
\nabla_l M_{ij}(\boldsymbol{z}) = \nabla_l\left(\frac{\mathcal{I}[f_{ij}](\boldsymbol{z})}{\mathcal{I}[\mathbf{1}](\boldsymbol{z})}\right) = \frac{\nabla_l\mathcal{I}[f_{ij}](\boldsymbol{z})\mathcal{I}[\mathbf{1}](\boldsymbol{z}) - \mathcal{I}[f_{ij}](\boldsymbol{z})\nabla_l\mathcal{I}[\mathbf{1}](\boldsymbol{z})}{(\mathcal{I}[\mathbf{1}](\boldsymbol{z}))^2}.
$$

Therefore, the finite-k error is bounded by

$$
\begin{aligned}
&|\nabla_l M_{ij}(\boldsymbol{z}) - \nabla_l M_{ij}^{(k)}(\boldsymbol{z})| \\
&= |\frac{\nabla_l\mathcal{I}[f_{ij}](\boldsymbol{z})\mathcal{I}[\mathbf{1}](\boldsymbol{z}) - \mathcal{I}[f_{ij}](\boldsymbol{z})\nabla_l\mathcal{I}[\mathbf{1}](\boldsymbol{z})}{(\mathcal{I}[\mathbf{1}](\boldsymbol{z}))^2} - \frac{\nabla_l\mathcal{I}^{(k)}[f_{ij}](\boldsymbol{z})\mathcal{I}^{(k)}[\mathbf{1}](\boldsymbol{z}) - \mathcal{I}^{(k)}[f_{ij}](\boldsymbol{z})\nabla_l\mathcal{I}^{(k)}[\mathbf{1}](\boldsymbol{z})}{(\mathcal{I}^{(k)}[\mathbf{1}](\boldsymbol{z}))^2}| \\
&\leq |\frac{\nabla_l\mathcal{I}[f_{ij}](\boldsymbol{z})}{\mathcal{I}[\mathbf{1}](\boldsymbol{z})} - \frac{\nabla_l\mathcal{I}^{(k)}[f_{ij}](\boldsymbol{z})}{\mathcal{I}^{(k)}[\mathbf{1}](\boldsymbol{z})}| + |\frac{\mathcal{I}[f_{ij}](\boldsymbol{z})\nabla_l\mathcal{I}[\mathbf{1}](\boldsymbol{z})}{(\mathcal{I}[\mathbf{1}](\boldsymbol{z}))^2} - \frac{\mathcal{I}^{(k)}[f_{ij}](\boldsymbol{z})\nabla_l\mathcal{I}^{(k)}[\mathbf{1}](\boldsymbol{z})}{(\mathcal{I}^{(k)}[\mathbf{1}](\boldsymbol{z}))^2}| \\
&\leq \frac{1}{2\beta k}\Big(\frac{|\nabla_l\mathcal{I}[f_{ij}](\boldsymbol{z})|Tr[\nabla^2\mathcal{I}[\mathbf{1}](\boldsymbol{z})] + |\nabla_l Tr[\nabla^2\mathcal{I}[f_{ij}](\boldsymbol{z})]|\mathcal{I}[\mathbf{1}](\boldsymbol{z})}{(\mathcal{I}[\mathbf{1}](\boldsymbol{z}))^2} \\
&+ \frac{\mathcal{I}[f_{ij}](\boldsymbol{z})|\nabla_l Tr[\nabla^2\mathcal{I}[\mathbf{1}](\boldsymbol{z})]| + Tr[\nabla^2\mathcal{I}[f_{ij}](\boldsymbol{z})]|\nabla_l\mathcal{I}[\mathbf{1}](\boldsymbol{z})|}{(\mathcal{I}[\mathbf{1}](\boldsymbol{z}))^2} + \frac{\mathcal{I}[f_{ij}](\boldsymbol{z})|\nabla_l\mathcal{I}[\mathbf{1}](\boldsymbol{z})|Tr[\nabla^2\mathcal{I}[\mathbf{1}](\boldsymbol{z})]}{(\mathcal{I}[\mathbf{1}](\boldsymbol{z}))^3}\Big) + O(\frac{1}{k^2}).
\end{aligned}
$$

The finite-T error is bounded by

$$
\begin{aligned}
&|\nabla_l M_{ij}^{(T,k)}(\boldsymbol{z}) - \nabla_l M_{ij}^{(k)}(\boldsymbol{z})| \leq |\mathcal{A}^{(T,k)}[\frac{\partial f_{ij}(\boldsymbol{z})}{z_l}](\boldsymbol{z}) - \mathcal{A}^{(k)}[\frac{\partial f_{ij}(\boldsymbol{z})}{z_l}](\boldsymbol{z})| \\
&+ \beta k|\mathcal{A}^{(T,k)}[f_{ij}(\boldsymbol{z})(z_l - x_l(\boldsymbol{r}))](\boldsymbol{z}) - \mathcal{A}^{(k)}[f_{ij}(\boldsymbol{z})(z_l - x_l(\boldsymbol{r}))](\boldsymbol{z})| \\
&+ \beta k|\mathcal{A}^{(T,k)}[f_{ij}(\boldsymbol{z})](\boldsymbol{z})\mathcal{A}^{(T,k)}[(z_l - x_l(\boldsymbol{r}))](\boldsymbol{z}) - \mathcal{A}^{(k)}[f_{ij}](\boldsymbol{z})\mathcal{A}^{(k)}[(z_l - x_l(\boldsymbol{r}))](\boldsymbol{z})| \\
\to &\sqrt{\frac{\tau[\frac{\partial f_{ij}(\boldsymbol{z})}{z_l}](\boldsymbol{z})}{T}}\xi[\frac{\partial f_{ij}(\boldsymbol{z})}{z_l}](\boldsymbol{z}) + \beta k\sqrt{\frac{\tau[f_{ij}(\boldsymbol{z})(z_l - x_l(\boldsymbol{r}))](\boldsymbol{z})}{T}}\xi[f_{ij}(\boldsymbol{z})(z_l - x_l(\boldsymbol{r}))](\boldsymbol{z}) \\
&+ \beta k\sqrt{\frac{\tau[f_{ij}(\boldsymbol{z})]}{T}}\mathcal{A}^{(k)}[(z_l - x_l(\boldsymbol{r}))](\boldsymbol{z})\xi[f_{ij}(\boldsymbol{z})](\boldsymbol{z}) \\
&+ \beta k\sqrt{\frac{\tau[(z_l - x_l(\boldsymbol{r}))](\boldsymbol{z})}{T}}\mathcal{A}^{(k)}[f_{ij}](\boldsymbol{z})\xi[(z_l - x_l(\boldsymbol{r}))](\boldsymbol{z}) + O(\frac{1}{T}).
\end{aligned}
$$

A.4 ERROR OF $\nabla^2 U(\boldsymbol{z})$

$\nabla_{ij}^2 U(\boldsymbol{z})$ can be written as

$$
\nabla_{ij}^2 U(\boldsymbol{z}) = \nabla_j\left(-\beta^{-1}\frac{\nabla_i \mathcal{I}[\mathbf{1}](\boldsymbol{z})}{\mathcal{I}[\mathbf{1}](\boldsymbol{z})}\right) = -\beta^{-1}\frac{\nabla_{ij}^2\mathcal{I}[\mathbf{1}](\boldsymbol{z})\mathcal{I}[\mathbf{1}](\boldsymbol{z}) - \nabla_i\mathcal{I}[\mathbf{1}](\boldsymbol{z})\nabla_j\mathcal{I}[\mathbf{1}](\boldsymbol{z})}{(\mathcal{I}[\mathbf{1}](\boldsymbol{z}))^2}.
$$

Therefore, the finite-k error is bounded by

$$
\begin{aligned}
&|\nabla_{ij}^2 U(\boldsymbol{z}) - \nabla_{ij}^2 U^{(k)}(\boldsymbol{z})| \\
&= \beta^{-1}|\frac{\nabla_{ij}^2\mathcal{I}[\mathbf{1}](\boldsymbol{z})\mathcal{I}[\mathbf{1}](\boldsymbol{z}) - \nabla_i\mathcal{I}[\mathbf{1}](\boldsymbol{z})\nabla_j\mathcal{I}[\mathbf{1}](\boldsymbol{z})}{(\mathcal{I}[\mathbf{1}](\boldsymbol{z}))^2} - \frac{\nabla_{ij}^2\mathcal{I}^{(k)}[\mathbf{1}](\boldsymbol{z})\mathcal{I}^{(k)}[\mathbf{1}](\boldsymbol{z}) - \nabla_i\mathcal{I}^{(k)}[\mathbf{1}](\boldsymbol{z})\nabla_j\mathcal{I}^{(k)}[\mathbf{1}](\boldsymbol{z})}{(\mathcal{I}^{(k)}[\mathbf{1}](\boldsymbol{z}))^2}| \\
&\leq \beta^{-1}\left(|\frac{\nabla_{ij}^2\mathcal{I}[\mathbf{1}](\boldsymbol{z})}{\mathcal{I}[\mathbf{1}](\boldsymbol{z})} - \frac{\nabla_{ij}^2\mathcal{I}^{(k)}[\mathbf{1}](\boldsymbol{z})}{\mathcal{I}^{(k)}[\mathbf{1}](\boldsymbol{z})}| + |\frac{\nabla_i\mathcal{I}[\mathbf{1}](\boldsymbol{z})\nabla_j\mathcal{I}[\mathbf{1}](\boldsymbol{z})}{(\mathcal{I}[\mathbf{1}](\boldsymbol{z}))^2} - \frac{\nabla_i\mathcal{I}^{(k)}[\mathbf{1}](\boldsymbol{z})\nabla_j\mathcal{I}^{(k)}[\mathbf{1}](\boldsymbol{z})}{(\mathcal{I}^{(k)}[\mathbf{1}](\boldsymbol{z}))^2}|\right) \\
&\leq \frac{1}{2\beta^2 k}\Big(\frac{|\nabla_{ij}^2\mathcal{I}[\mathbf{1}](\boldsymbol{z})|Tr[\nabla^2\mathcal{I}[\mathbf{1}](\boldsymbol{z})] + |\nabla_{ij}^2 Tr[\nabla^2\mathcal{I}[\mathbf{1}](\boldsymbol{z})]|\mathcal{I}[\mathbf{1}](\boldsymbol{z})}{(\mathcal{I}[\mathbf{1}](\boldsymbol{z}))^2} \\
&+ \frac{|\nabla_i\mathcal{I}[\mathbf{1}](\boldsymbol{z})\nabla_j Tr[\nabla^2\mathcal{I}[\mathbf{1}](\boldsymbol{z})]| + |\nabla_i Tr[\nabla^2\mathcal{I}[\mathbf{1}](\boldsymbol{z})]\nabla_j\mathcal{I}[\mathbf{1}](\boldsymbol{z})|}{(\mathcal{I}[\mathbf{1}](\boldsymbol{z}))^2} + \frac{|\nabla_i\mathcal{I}[\mathbf{1}](\boldsymbol{z})\nabla_j\mathcal{I}[\mathbf{1}](\boldsymbol{z})|Tr[\nabla^2\mathcal{I}[\mathbf{1}](\boldsymbol{z})]}{(\mathcal{I}[\mathbf{1}](\boldsymbol{z}))^3}\Big) + O(\frac{1}{k^2}).
\end{aligned}
$$

The finite-T error is bounded by

$$
\begin{aligned}
&|\nabla_{ij}^2 U^{(T,k)}(\boldsymbol{z}) - \nabla_{ij}^2 U^{(k)}(\boldsymbol{z})| \leq k|\mathcal{A}^{(T,k)}[\frac{\partial(z_j - x_j(\boldsymbol{r}))}{z_i}](\boldsymbol{z}) - \mathcal{A}^{(k)}[\frac{\partial(z_j - x_j(\boldsymbol{r}))}{z_i}](\boldsymbol{z})| \\
&+ \beta k^2|\mathcal{A}^{(T,k)}[(z_j - x_j(\boldsymbol{r}))(z_i - x_i(\boldsymbol{r}))](\boldsymbol{z}) - \mathcal{A}^{(k)}[(z_j - x_j(\boldsymbol{r}))(z_i - x_i(\boldsymbol{r}))](\boldsymbol{z})| \\
&+ \beta k^2|\mathcal{A}^{(T,k)}[(z_j - x_j(\boldsymbol{r}))](\boldsymbol{z})\mathcal{A}^{(T,k)}[(z_i - x_i(\boldsymbol{r}))](\boldsymbol{z}) - \mathcal{A}^{(k)}[(z_j - x_j(\boldsymbol{r}))](\boldsymbol{z})\mathcal{A}^{(k)}[(z_i - x_i(\boldsymbol{r}))](\boldsymbol{z})| \\
\to\ &k\sqrt{\frac{\tau[\frac{\partial(z_j - x_j(\boldsymbol{r}))}{z_i}](\boldsymbol{z})}{T}}\xi[\frac{\partial(z_j - x_j(\boldsymbol{r}))}{z_i}](\boldsymbol{z}) + \beta k^2\sqrt{\frac{\tau[(z_j - x_j(\boldsymbol{r}))(z_i - x_i(\boldsymbol{r}))](\boldsymbol{z})}{T}}\xi[(z_j - x_j(\boldsymbol{r}))(z_i - x_i(\boldsymbol{r}))](\boldsymbol{z}) \\
&+ \beta k^2\sqrt{\frac{\tau[(z_i - x_i(\boldsymbol{r}))](\boldsymbol{z})}{T}}\mathcal{A}^{(k)}[(z_j - x_j(\boldsymbol{r}))](\boldsymbol{z})\xi[(z_i - x_i(\boldsymbol{r}))](\boldsymbol{z}) \\
&+ \beta k^2\sqrt{\frac{\tau[(z_j - x_j(\boldsymbol{r}))](\boldsymbol{z})}{T}}\mathcal{A}^{(k)}[(z_i - x_i(\boldsymbol{r}))](\boldsymbol{z})\xi[(z_j - x_j(\boldsymbol{r}))](\boldsymbol{z}) + O(\frac{1}{T}).
\end{aligned}
$$

B CONVERGENCE RATE OF HESSIAN-VECTOR PRODUCT ESTIMATOR

Here we discuss the convergence rate of using Hessian vector product estimator

$$
\frac{\nabla U(\boldsymbol{z} + \delta v) - \nabla U(\boldsymbol{z} - \delta v)}{2\delta},
$$

compared to the direct estimator $\nabla^2 U v$. Here $v = \frac{M^T M \nabla U}{\|M\nabla U\|}$. By Theorem 4.1, we know that

- For any $\boldsymbol{z}$, the error of estimating ∇U is bounded by $|\nabla_i U(\boldsymbol{z}) - \nabla_i U^{(T,k)}(\boldsymbol{z})| \leq O(\frac{1}{k} + \frac{k}{\sqrt{T}})$,
- For any $\boldsymbol{z}$, the error of estimating $v = \frac{M^T M \nabla U}{\|M \nabla U\|}$ is bounded by $|v_i(\boldsymbol{z}) - v_i^{(T,k)}(\boldsymbol{z})| \leq O(\frac{1}{k} + \frac{k}{\sqrt{T}})$,
- For any $\boldsymbol{z}$, the error of estimating $\nabla^2 U$ is bounded by $|\nabla^2_{ij} U(\boldsymbol{z}) - \nabla^2_{ij} U^{(T,k)}(\boldsymbol{z})| \leq O(\frac{1}{k} + \frac{k^2}{\sqrt{T}})$.

First we consider the error of directly estimating $\nabla^2 U v$.

$$
\begin{aligned}
&|(\nabla^2 U v)_i - (\nabla^2 U^{(T,k)} v^{(T,k)})_i| = |\sum_j \nabla^2_{ij} U(\boldsymbol{z}) v_j(\boldsymbol{z}) - \nabla^2_{ij} U^{(T,k)}(\boldsymbol{z}) v_j^{(T,k)}(\boldsymbol{z})| \\
&\leq \sum_j |\nabla^2_{ij} U(\boldsymbol{z}) v_j(\boldsymbol{z}) - \nabla^2_{ij} U^{(T,k)}(\boldsymbol{z}) v_j^{(T,k)}(\boldsymbol{z})| \\
&\leq \sum_j \Big(|\nabla^2_{ij} U(\boldsymbol{z}) - \nabla^2_{ij} U^{(T,k)}(\boldsymbol{z})| v_j(\boldsymbol{z}) + |v_j(\boldsymbol{z}) - v_j^{(T,k)}(\boldsymbol{z})| \nabla^2 U_{ij}(\boldsymbol{z}) + |\nabla^2_{ij} U(\boldsymbol{z}) - \nabla^2_{ij} U^{(T,k)}(\boldsymbol{z})| |v_j(\boldsymbol{z}) - v_j^{(T,k)}(\boldsymbol{z})| \Big) \\
&\leq O(\frac{1}{k} + \frac{k^2}{\sqrt{T}}).
\end{aligned}
$$

So the convergence rate of the error is $O(\frac{1}{k} + \frac{k^2}{\sqrt{T}})$, due to the contribution of $|\nabla^2_{ij} U(\boldsymbol{z}) - \nabla^2_{ij} U^{(T,k)}(\boldsymbol{z})|$.

Now let's consider the Hessian-vector product estimator. The error can be decomposed into three terms as follows

$$
\begin{aligned}
&|\frac{\nabla_i U^{(T,k)}(\boldsymbol{z} + \delta v^{(T,k)}) - \nabla_i U^{(T,k)}(\boldsymbol{z} - \delta v^{(T,k)})}{2\delta} - (\nabla^2 U v)_i| \\
&\leq |\frac{\nabla_i U^{(T,k)}(\boldsymbol{z} + \delta v^{(T,k)}) - \nabla_i U^{(T,k)}(\boldsymbol{z} - \delta v^{(T,k)})}{2\delta} - \frac{\nabla_i U(\boldsymbol{z} + \delta v^{(T,k)}) - \nabla_i U(\boldsymbol{z} - \delta v^{(T,k)})}{2\delta}| \\
&+ |\frac{\nabla_i U(\boldsymbol{z} + \delta v^{(T,k)}) - \nabla_i U(\boldsymbol{z} - \delta v^{(T,k)})}{2\delta} - (\nabla^2 U v^{(T,k)})_i| \\
&+ |(\nabla^2 U v^{(T,k)})_i - (\nabla^2 U v)_i|.
\end{aligned}
$$

The first term comes from the error of estimating ∇U, which can be upper bounded by

$$
\begin{aligned}
&|\frac{\nabla_i U^{(T,k)}(\boldsymbol{z} + \delta v^{(T,k)}) - \nabla_i U^{(T,k)}(\boldsymbol{z} - \delta v^{(T,k)})}{2\delta} - \frac{\nabla_i U(\boldsymbol{z} + \delta v^{(T,k)}) - \nabla_i U(\boldsymbol{z} - \delta v^{(T,k)})}{2\delta}| \\
&\leq \frac{1}{2\delta} \Big(|\nabla_i U^{(T,k)}(\boldsymbol{z} + \delta v^{(T,k)}) - \nabla_i U(\boldsymbol{z} + \delta v^{(T,k)})| + |\nabla_i U^{(T,k)}(\boldsymbol{z} - \delta v^{(T,k)}) - \nabla_i U(\boldsymbol{z} - \delta v^{(T,k)})| \Big) \\
&\leq O(\frac{1}{\delta}(\frac{1}{k} + \frac{k}{\sqrt{T}})).
\end{aligned}
$$

The second term comes from finite difference estimate of $\nabla^2 U$. Using Taylor expansion,

$$
\begin{aligned}
&\nabla_i U(\boldsymbol{z} + \delta v^{(T,k)}) - \nabla_i U(\boldsymbol{z} - \delta v^{(T,k)}) \\
&= \left(U_i(\boldsymbol{z}) + \delta v^{(T,k)} \nabla \nabla_i U(\boldsymbol{z}) + \frac{\delta^2}{2} (v^{(T,k)})^T \nabla^2 \nabla_i U(\boldsymbol{z}) v^{(T,k)} \right) \\
&- \left(U_i(\boldsymbol{z}) - \delta v^{(T,k)} \nabla \nabla_i U(\boldsymbol{z}) + \frac{\delta^2}{2} (v^{(T,k)})^T \nabla^2 \nabla_i U(\boldsymbol{z}) v^{(T,k)} \right) + O(\delta^3) \\
&= 2\delta (\nabla^2 U v^{(T,k)})_i + O(\delta^3).
\end{aligned}
$$

Therefore, the error is bounded by

$$
|\frac{\nabla_i U(\boldsymbol{z} + \delta v^{(T,k)}) - \nabla_i U(\boldsymbol{z} - \delta v^{(T,k)})}{2\delta} - (\nabla^2 U v^{(T,k)})_i| = O(\delta^2).
$$

The third term comes from the error of estimating v, which can be upper bounded by

$$
|(\nabla^2 U v^{(T,k)})_i - (\nabla^2 U v)_i| \leq \sum_j \nabla^2_{ij} U(\boldsymbol{z}) |v_j(\boldsymbol{z}) - v_j^{(T,k)}(\boldsymbol{z})| \leq O(\frac{1}{k} + \frac{k}{\sqrt{T}}).
$$

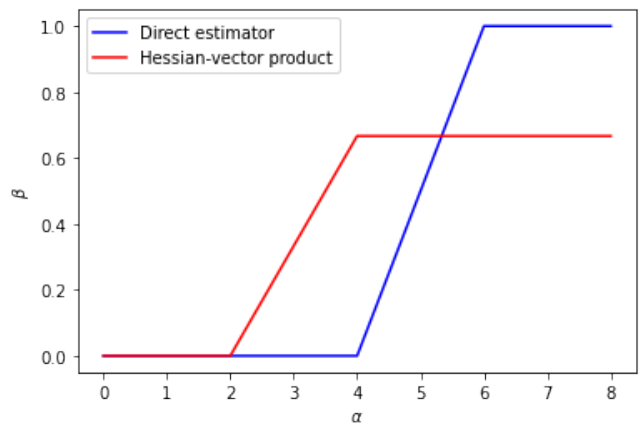

Figure 5: Relation between the convergence rate $\beta = -\lim_{k\to\infty} \frac{\log|\nabla^2 U^{(T,k)} v^{(T,k)} - \nabla^2 U v|}{\log k}$ and the rate $\alpha = \lim_{k\to\infty} \frac{\log T}{\log k}$.

The combined convergence rate of the total estimation error is bounded by

$$
\begin{aligned}
&|\frac{\nabla_i U^{(T,k)}(\boldsymbol{z} + \delta v^{(T,k)}) - \nabla_i U^{(T,k)}(\boldsymbol{z} - \delta v^{(T,k)})}{2\delta} - (\nabla^2 U v)_i| \\
&\leq O(\frac{1}{\delta}(\frac{1}{k} + \frac{k}{\sqrt{T}})) + O(\delta^2) + O(\frac{1}{k} + \frac{k}{\sqrt{T}}).
\end{aligned}
$$

The upper bound contains two terms of δ which are $O(\frac{1}{\delta}(\frac{1}{k} + \frac{k}{\sqrt{T}}))$ and $O(\delta^2)$. The optimal δ is $O((\frac{1}{k} + \frac{k}{\sqrt{T}})^{1/3})$ which makes the rate of the error be $O((\frac{1}{k} + \frac{k}{\sqrt{T}})^{2/3})$. The third term is always not the leading term.

Remark: The benifit of Hessian-vector product estimator depends on how fast T grows as k grows. Let $\alpha = \lim_{k\to\infty} \frac{\log T}{\log k}$.

- If $\alpha \leq 2$, neither direct estimator nor Hessian-vector product estimator converges.
- If $2 < \alpha \leq 4$, direct estimator does not converge, but Hessian-vector product estimator converges at a rate of $O(k^{-(\alpha-2)/3})$.
- If $4 < \alpha \leq 6$, both estimators converge. The direct estimator converges at a rate of $O(k^{-(\alpha-4)/2})$ and the Hessian-vector product estimator converges at a rate of $O(k^{-2/3})$. If $\alpha \leq 16/3$, the Hessian-vector product estimator converges faster and if $\alpha > 16/3$, the direct estimator converges faster.
- If $\alpha > 6$, both estimators converge. The direct estimator converges at a rate of $O(k^{-1})$ and the Hessian-vector product estimator converges at a rate of $O(k^{-2/3})$. The direct estimator converges faster.

The relation between the convergence rate and α are shown in Figure 5.

C MÜLLER POTENTIAL PARAMETERS

This section provides the detailed parameters in Eq.(21).

$$
\begin{aligned}
&A = (-200, -100, -170, 15), \quad a = (-1, -1, -6.5, 0.7), \\
&b = (0, 0, 11, 0.6), \quad c = (-10, -10, -6.5, 0.7), \\
&x^0 = (1, 0, -0.5, -1), y^0 = (0, 0.5, 1.5, 1).
\end{aligned}
$$