Differentiable Simulations for Enhanced Sampling of Rare Events

Martin Šípka [1 2 3] **Johannes C. B. Dietschreit** [1] **Lukáš Grajciar** [3] **Rafael Gómez-Bombarelli** [1]

Abstract

Simulating rare events, such as the transformation of a reactant into a product in a chemical reaction typically requires enhanced sampling techniques that rely on heuristically chosen collective variables (CVs). We propose using differentiable simulations (DiffSim) for the discovery and enhanced sampling of chemical transformations without a need to resort to preselected CVs, using only a distance metric. Reaction path discovery and estimation of the biasing potential that enhances the sampling are merged into a single end-to-end problem that is solved by path-integral optimization. This is achieved by introducing multiple improvements over standard DiffSim such as partial backpropagation and graph mini-batching making DiffSim training stable and efficient. The potential of DiffSim is demonstrated in the successful discovery of transition paths for the Muller-Brown model potential as well as a benchmark chemical system - alanine dipeptide.

1. Introduction

A chemical reaction can be viewed as a transition from one depression (reactant) on the potential energy surface (PES) to another (product). The most likely transition path(s) connecting the two basins define the reaction mechanism(s). The potential energy of a saddle point, through which the system has to pass, defines the reaction barrier and is the fundamental quantity when investigating reaction rates. The major obstacle in determining the reaction path lies in the high dimensionality of the molecular configuration space that can easily be spanned by thousands of degrees of freedom (DoF). Extensive sampling of configurations along candidate transition paths, characterized by comparatively high free energies(Chipot & Pohorille, 2007; Chipot, 2014) is needed, but standard unbiased sampling algorithms, e.g., molecular dynamics (MD) or Monte-Carlo (MC), often remain trapped in (meta)stable regions. Therefore, it is extremely inefficient to explore candidate paths in an unbiased way, and it is necessary to adopt heuristics to bias the exploration, which are often based on expert chemical intuition.

The problem has been commonly split into two seemingly easier sub-tasks. First, a dimensionality reduction from all DoFs down to the so-called collective variables (CVs) and second, enhanced sampling along those CVs (Torrie et al., 1977; Darve & Pohorille, 2001; Laio & Parrinello, 2002; Abrams & Bussi, 2013; Spiwok et al., 2015; Valsson et al., 2016). Even though widely used methods exist to solve the second problem, the first part - identifying collective variables - is still largely a manual task based on expert chemical intuition, with the commonly used CVs being not much more complex than simple linear combinations of manually chosen internal DoFs of the molecular systems in question. Recently, the task of identifying CVs has been partially automatized by multiple machine learning based tools.(Sultan & Pande, 2018; Mendels et al., 2018; Wehmeyer & Noé, 2018; Wang et al., 2019; Bonati et al., 2020; Wang & Tiwary, 2021; Sun et al., 2022; Šípka et al., 2022) The number of the CVs is typically limited to one to three due to the exponential growth of computational cost, known as the curse of dimensionality (Bellman, 1967; Köppen, 2000). The CVs should be based on those DoFs, which fully describe the rare transition event, and are thereby associated with the slowest motions. However, identification of the important DoFs *a priori* typically requires knowledge of the transition path that one is trying to discover in the first place, *i.e.*, one still ends up with the proverbial "chicken-and-egg problem" (Rohrdanz et al., 2013). This is a problem that previously proposed machine learning based tools cannot directly tackle. Additionally, once the CVs are chosen it is very difficult to correct them on-the-fly. Thus, one must be certain that the chosen CV function is properly defined and well behaved in all regions (i.e., the CV values for reactant and product basins do not overlap or it shows undefined behavior for unseen configurations). A hard task for tools

[1]Department of Materials Science and Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA [2]Mathematical Institute, Faculty of Mathematics and Physics, Charles University, Sokolovská 83, 186 75 Prague, Czech Republic [3]Department of Physical and Macromolecular 853 Chemistry, Faculty of Sciences, Charles University, 128 43 854 Prague 2, Czech Republic. Correspondence to: Rafael Gómez-Bombarelli <rafagb@mit.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

such as neural networks as they often extrapolate poorly when presented with unseen data. To solve these problems, iterative improvements of CVs have been proposed (Chen et al., 2018b; Belkacemi et al., 2022) where both, CV training algorithm and biasing method, are iterated until the final results of the biased dynamics is satisfactory. This can be slow as the enhanced sampling needs to be rerun for each iteration of the CV. Once equipped with a low-dimensional representation of the chemical reaction, the enhanced sampling algorithms usually introduce a biasing potential, which is a function of the identified CVs and modifies the original PES by lowering the reaction barrier. If CVs and enhanced sampling technique are chosen well, the biased simulation will significantly increase the occurrence of reactive events, and subsequent analysis will allow us to understand the reaction mechanism and to calculate reaction barrier and rate.

Traditionally, there exists an alternative to enhanced sampling, namely path sampling techniques(Dellago et al., 1998), including transition path sampling(Bolhuis et al., 2002), transition interface sampling(Van Erp et al., 2003), and multilevel splitting(Cérou et al., 2011). These algorithms try to sample reactive trajectories without modifying the PES itself, but by choosing optimal starting points that generate paths connecting reactant and product. In order to do so a rough measure of reaction progress is still needed, e.g., a CV or the committor (a function of all coordinates that gives the probability of a path originating there will visit the product well). This means that one either also needs a CV or has to have prior knowledge of the location of the transition state region.

Simulations that are fully differentiable have been developed for optimization, control, and learning of motion,(Degrave et al., 2016; de Avila Belbute-Peres et al., 2018; Hu et al., 2019; 2020) but also for the learning and optimization of quantities of interest in molecular dynamics (Wang et al., 2020; Ingraham et al., 2019; Greener & Jones, 2021). Differentiating through simulations comes naturally from the optimization of path-dependent quantities (the famous Brachistochrone curve problem is included in Appendix H for the novice reader). If the minimization of a loss function cannot be formulated separately for every point in the path, then optimization has to include the whole path leading up to it. While the results of DiffSims are often promising, it is well known (Metz et al., 2021) that naïvely backpropagated gradients may vanish or explode, and thus not lead to a useful parameter update. How to control their behaviour remains an open challenge. This problem of differentiable simulations is associated with the spectrum of the system's Jacobian(Metz et al., 2021; Galimberti et al., 2021) and closely connected to the chaotic nature of the simulated equations. Therefore, in order to employ path differentiation, one needs to find ways to produce well behaving and controllable gradients. In this contribution, the loss gradient behaviour is thoroughly investigated, and a mechanism to control its fluctuations and magnitude is proposed. Employing the improved DiffSims, we define a differentiable loss function that, when minimized, results in the robust training of a biasing potential, which enhances the sampling of reactive transitions without prior determination of CVs.

The manuscript is structured as follows. In Section 2 we define molecular dynamics simulations biased with a learnable potential, introduce a formalism to describe chemical reactions using path integrals, and outline the concept of differentiable simulations. We discuss the current challenges and limitations of DiffSims in Section 3 and propose novel techniques to resolve them. In section 4 we outline the practical implementation of our method. In Section 5, we demonstrate the usefulness of DiffSims in the context of chemical reactions by training the bias function promoting barrier crossing for the well-studied Muller-Brown potential as well as the alanine-dipeptide molecule.

2. Problem Definition

Molecular dynamics is commonly used to explore reaction processes on a atomistic level. Let the column vector $\boldsymbol{x} \in \mathbb{R}^N$ denote the mass-weighted coordinates of the system and $\boldsymbol{p}$ the conjugate momenta. The particle motion is simulated using Hamiltonian equations with potential energy function $U_0(\boldsymbol{x})$.

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{p}(t) \\ \dot{\boldsymbol{p}}(t) &= -\frac{\partial U_0(\boldsymbol{x}(t))}{\partial \boldsymbol{x}} \end{aligned} \tag{1}$$

These equations conserve energy and are purely reversible with respect to time. However, it is common in molecular modeling not to work with the micro-canonical ensemble but rather with the canonical ensemble that conserves temperature (Callen & Scott, 1998). This is realized by using a thermostat coupled to the system. In this work, we choose the Langevin thermostat because of its implementational simplicity and its favorable properties with respect to differentiating along the computational graph, as will be shown later (see Section 3.2). In Langevin dynamics, the thermostat is coupled to the system through the friction constant γ (3).

To increase the probability of the barrier crossing we modify the PES with a learnable bias term $B(\boldsymbol{x}, \theta)$

$$U(\boldsymbol{x}, \theta) = U_0(\boldsymbol{x}) + B(\boldsymbol{x}, \theta), \tag{2}$$

where the biasing function is parameterized by θ, which we aim to train to increase the frequency of reaction events.

The biased dynamics evolve according to

$$\begin{aligned}\dot{\boldsymbol{x}}(t) &= \boldsymbol{p}(t)\\ \dot{\boldsymbol{p}}(t) &= -\frac{\partial U(\boldsymbol{x}(t))}{\partial \boldsymbol{x}} - \gamma \boldsymbol{p}(t) + \sqrt{2\gamma k_B T}\boldsymbol{R}(t),\end{aligned} \tag{3}$$

where k_b is the Boltzmann constant, T the absolute temperature of the bath, and $\boldsymbol{R}(t)$ a Gaussian process.

2.1. Formal characterization of (chemical) reactions

By using the term "reaction" we mean to encompass any process that can be described as the transition between two depressions on the PES, this includes but is not limited to the rearrangement of chemical bonds, the exchange between conformers, and phase transitions. It is often suitable to use general curvilinear coordinates and not simply Cartesian or mass-weighted coordinates to describe reactions. Common are internal coordinates such as interatomic distances, angles, or dihedrals, as they are invariant with respect to system rotation and translation. These special coordinates are denoted with $\boldsymbol{\xi}(\boldsymbol{x}) \in \mathbb{R}^M$ and $M \leq N$.

The wells W_α of reactant (-1) and product (1), divided by a reaction barrier, are characterized by the set of points Γ_α ($\alpha = -1, 1$), which correspond to the equilibrium configurations of reactants and products, i.e., we expect an unbiased simulation on the PES $U_0(\boldsymbol{x})$, to stay in these wells with a very high probability. We approximate the wells with a multivariate normal distribution. From short, unbiased simulations, we estimate mean $\boldsymbol{\mu}_\alpha$ and covariance matrix $\boldsymbol{\Sigma}_\alpha$. We consider a point to be part of a well if the probability of the point belonging to the distribution is above some chosen probability threshold.

$$W_\alpha = \left\{\boldsymbol{x} \mid (\boldsymbol{\xi}(\boldsymbol{x}) - \boldsymbol{\mu}_\alpha)^T \Sigma_\alpha^{-1} (\boldsymbol{\xi}(\boldsymbol{x}) - \boldsymbol{\mu}_\alpha) < \epsilon\right\} \ , \tag{4}$$

where epsilon can be obtained from χ^2 distribution. The indicator function for a well is

$$\mathbb{1}_\alpha(\boldsymbol{x}) = \begin{cases} 1 & \text{for} \quad \boldsymbol{x} \in W_\alpha \\ 0 & \text{for} \quad \boldsymbol{x} \notin W_\alpha. \end{cases} \tag{5}$$

In this manuscript, we only consider transitions between two wells, W_{-1} and W_1. Additional basins would be handled analogously. The (escape) probability p_α, within a specified time interval (t_0, t_e) of a transition $W_{-\alpha} \to W_\alpha$ is defined as

$$p_\alpha = P\left(\int_{t_0}^{t_e} \mathbb{1}_\alpha(\boldsymbol{x}(t))\,\mathrm{d}t > 0 \,\middle|\, \boldsymbol{x}(t_0) \in W_{-\alpha}\right) \ , \tag{6}$$

where t_0 is the start and t_e the end time of the trajectory $\boldsymbol{X}$. This can be understood as the probability of finding at least one point in W_α of a trajectory that has started in $W_{-\alpha}$. Our objective is to increase both p_1 and p_{-1} simultaneously to a level where both events can be observed frequently on a typical simulation time scale.

2.2. Optimizing the probability

The form of the probability in (6) is not usable for differentiable optimization and needs to be recast to a differentiable, continuous form. Under suitable regularity conditions, we can replace the expression of (6) with

$$p_\alpha = P\left(\sup_{t<t_e}(\mathbb{1}_\alpha(\boldsymbol{x}(t)) > 0 \,\middle|\, \boldsymbol{x}(t_0) \in W_{-\alpha}\right). \tag{7}$$

We can then define a soft loss function that is continuous everywhere and differentiable for any trajectory $\boldsymbol{X}$ with $\boldsymbol{x}(t_0) \in W_{-\alpha}$ as

$$L = L_{\boldsymbol{\xi}_\alpha} = \begin{cases} 0 & \text{if } \exists\, \boldsymbol{x}(t) \in W_\alpha \\ \min\limits_{t_0<t<t_e} (\boldsymbol{\xi}(\boldsymbol{x}(t)) - \boldsymbol{\xi}_\alpha)^2 & \text{otherwise} \end{cases}, \tag{8}$$

where for each trajectory a random single $\boldsymbol{\xi}_\alpha \in \Gamma_\alpha$ is selected for the loss function by running a short, unbiased simulation. The term $(\boldsymbol{\xi}(\boldsymbol{x}(t)) - \boldsymbol{\xi}_\alpha)^2$ is the distance metric used as measure of how close a trajectory got to the target. In general, we choose the metric to be the quadratic distance either in Cartesian coordinates or the descriptor space, depending on the type of the reaction. Other more complex metrics can be chosen. Choosing random targets is done to increase the configuration space the simulation is forced to cover, avoiding targeting a particular point, thus making the optimization more robust. Minimizing this loss function leads to a maximization of the probability (6) and can be seen as the minimization of a path-dependent integral. In the following Section, we will define a method that can be employed to minimize (8). Note that the loss function is defined only for one point of the trajectory and is influenced by the dynamics of every point that proceeds it.

2.3. Differentiable simulations

For the problem at hand, the parameters θ of the bias potential (2) have to be optimized such that the loss (8) is minimal. For a differentiable simulation, the information that we gain by differentiating the loss function at the point where it is defined can be used for optimization along the whole trajectory. While we could proceed by considering the simulation as a forward process, saving the computational graph for the entire path would be extremely memory-demanding. Instead, the optimization process can be conveniently reformulated using the adjoint equation and resulting adjoint vectors, using which the system dynamics can be run backwards to an arbitrary time, leading to memory-savings and the ability to adjust extent of backpropagation based on the sought-for dynamical scale (see Section 3). We employ the framework and notation adapted recently for neural networks(Chen et al., 2018a) from the original work by (Lev Semenovich Pontryagin et al., 1962).

We propagate the state $\boldsymbol{z}(t) = (\boldsymbol{x}(t), \boldsymbol{p}(t))$ using the biased

Langevin dynamics where the right side of (3) shall be denoted as $f(\boldsymbol{z}(t), \theta) = \dot{\boldsymbol{z}}(t)$. Propagating $\boldsymbol{z}(t)$ using $f(\boldsymbol{z}(t))$ is called the *forward process*. Notice that $f(\boldsymbol{z}(t))$ has no explicit time dependence (only through $\boldsymbol{z}(t)$). We define the adjoint vectors for this equations as

$$\boldsymbol{a}(t) = \frac{\partial L}{\partial \boldsymbol{z}(t)} \,. \tag{9}$$

To solve (9) we introduce the new time $\tau \in (0, \tau_e)$ such that $\boldsymbol{z}(\tau = 0) = \boldsymbol{z}(t = t_e)$ and $\boldsymbol{z}(\tau = \tau_e) = \boldsymbol{z}(t = 0)$. This backward flowing time reflects that the loss is not influenced by any points further in forward time.

In forward moving time t, the adjoint vectors obey the equations

$$\begin{aligned} \boldsymbol{a}(t_e) &= \frac{\partial L}{\partial \boldsymbol{z}(t_e)} \\ \dot{\boldsymbol{a}}(t) &= -\boldsymbol{a}(t)^T \frac{\partial f(\boldsymbol{z}(t), \theta)}{\partial \boldsymbol{z}}, \end{aligned} \tag{10}$$

or in backward going time τ, the equation

$$\begin{aligned} \boldsymbol{a}(\tau = 0) &= \frac{\partial L}{\partial \boldsymbol{z}(\tau = 0)} \\ \dot{\boldsymbol{a}}(\tau) &= \boldsymbol{a}(\tau)^T \frac{\partial f(\boldsymbol{z}(\tau), \theta)}{\partial \boldsymbol{z}}. \end{aligned} \tag{11}$$

The total gradient of the loss function with respect to bias parameters is then obtained by

$$\frac{\partial L}{\partial \theta} = \int_0^{\tau_e} \boldsymbol{a}(\tau)^T \frac{\partial f(\boldsymbol{z}(\tau), \theta)}{\partial \theta} \mathrm{d}\tau \,. \tag{12}$$

While solving (11), $\boldsymbol{z}(\tau)$ can be either saved or reconstructed by running dynamics (3) backward, depending on the memory and computational trade-off we would like to maintain. The algorithm for running the adjoint method for the dynamics that includes random noise is developed and analyzed in (Li et al., 2020).

3. Challenges and Solutions

3.1. Challenges

Ideally, one would simulate the biased dynamics (3), compute the loss (8), backpropagate by solving (11), and after a number of training epochs obtain the biasing potential that enhances transitions. However, differentiable simulations at their current state cannot be used out of the box. There exist several issues that need to be addressed.

1. **Gradient control**
   Significant effort has been devoted in the past years to understand the behavior of gradients that arise while optimizing neural network controlled differentiable simulations (Suh et al., 2022; Huang et al., 2021; Metz et al., 2021). Some of the main challenges in this respect are the explosion or the vanishing of gradients when training deep neural networks. A differentiable simulation can be arbitrarily deep, however, it can be challenging to backpropagate complex Hamiltonians in a controllable manner to such depths. In fact, it is possible to construct a simple Hamiltonian that gives rise to exploding gradients when using (3) (Galimberti et al., 2021).

2. **Multiscale Problem**
   The dynamics on $U_0(\boldsymbol{x})$ can include very high and very low frequency motions. However, only the slow dynamics should be controlled by the trainable bias, as those are associated with the sought-after chemical reactions. High frequency modes, e.g., hydrogen vibrations in the case of molecules, usually do not contribute to the reaction mechanism. Avoiding fitting such fast fluctuations is desirable as it reduces the noise in the gradients used for DiffSim training.

3. **Chaotic behaviour**
   One important property of some Hamiltonian systems is the emergence of chaos (Percival I, 1987). Small changes in initial conditions result in exponentially different trajectories. Great care must be taken to predict and control the behaviour of such systems.

4. **One large parameter update per trajectory**
   Differentiable simulations in their original formulation produce one update per trajectory. Obtaining a sufficient number of gradient updates can be very expensive when long trajectories are required.

All these challenges are addressed by the present work. We show how to efficiently learn slow dynamics necessary for the investigation of chemical reactions while keeping gradients under control.

3.2. Partial backpropagation

To reduce the complexity and level of detail in the equations, the computational graph is pruned such that backpropagation occurs only in the momenta. This is realized by adoption of the *.detach()* operator introduced, e.g., in Refs. (Foerster et al., 2018; Schulman et al., 2015; Zhang et al., 2019), which stops the flow of the gradient through $\boldsymbol{x}$. The backpropagation only in momenta can be reasoned as follows:

- For timescales $\Delta\tau$, typical for the slow dynamics in the system, we assume $\boldsymbol{x}$ to linearly approach the target value. This discards fast oscillations in $\boldsymbol{x}$. In other words, for timescales $\Delta\tau$ we expect the change in $\boldsymbol{x}$ to

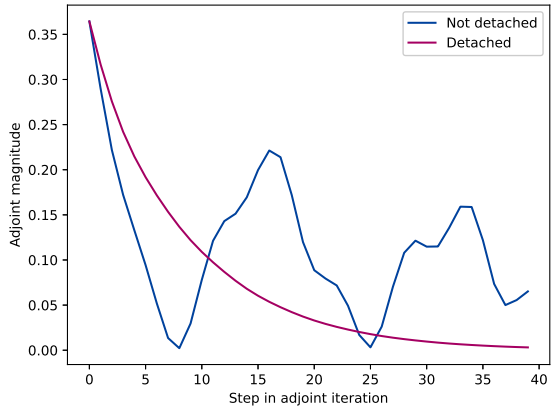

Figure 1: Comparison of the adjoint evolution for original and partially detached graphs for simulations on the 2D Müller-Brown potential with parameters given in Appendix G.

be linear in time

$$\frac{\partial \boldsymbol{x}(\tau)}{\partial \boldsymbol{p}(\tau)} = \Delta\tau \ , \tag{13}$$

which corresponds to the equation for $\boldsymbol{x}$ in the form

$$\boldsymbol{x}(\tau) = \boldsymbol{x}(\tau - \Delta\tau).detach() + \boldsymbol{p}(\tau)\Delta\tau \ . \tag{14}$$

- The use of the detach operator reduces the backpropagated ODE to first order in time, neglecting all higher order terms. Hence, the backpropagation follows a diffusion type equation, and any high frequency oscillations are removed. The change in adjoint dynamics can be seen in the Figure 1.

- The modified dynamics are well behaved with respect to the magnitude of the gradients as shown in the theorems.

The introduction of the *.detach()* operator reduces the number of equations for which the adjoint is calculated and through which the loss is backpropagated. Position $\boldsymbol{x}(\tau)$ is no longer an independent variable only a function of $\boldsymbol{p}(\tau)$. The detached $\boldsymbol{x}$ from the previous timestep is treated as a constant in Equation (14). The adjoint dynamics is then calculated in only one variable $\boldsymbol{p}(\tau)$ and only the evolutionary equation for this variable is considered. To simplify the discussion of adjoints we will not discretize the backward equation, but keep it in the continuous form. The use of the *.detach()* operator simplifies the adjoint time derivative to

$$\begin{aligned}\dot{\boldsymbol{a}}(\tau) &= \boldsymbol{a}^T \frac{\partial}{\partial \boldsymbol{p}(\tau)} \left(-\frac{\partial U(\boldsymbol{x}(\tau))}{\partial \boldsymbol{x}} - \gamma \boldsymbol{p}(\tau) \right. \\ &\quad \left. + \sqrt{2\gamma k_B T}\boldsymbol{R}(\tau) \right) \\ &= -\boldsymbol{a}^T(\tau) \left(\frac{\partial^2 U(\boldsymbol{x}(\tau))}{\partial^2 \boldsymbol{x}} \frac{\partial \boldsymbol{x}(\tau)}{\partial \boldsymbol{p}(\tau)} - \gamma \mathbf{I} \right) \\ &= -\boldsymbol{a}^T(\tau) \frac{\partial^2 U(\boldsymbol{x}(\tau))}{\partial^2 \boldsymbol{x}} \Delta\tau - \gamma \boldsymbol{a}(\tau) \ .\end{aligned} \tag{15}$$

Let us now formulate the property of the adjoints that will be useful when designing numerical methods and also gives us some assurance of the non-diverging gradient dynamics. Consider a trajectory x_t generated by (3) with time t in a possibly infinite time interval $I \subset (-\infty, \infty)$. The loss (8) is defined for a point $\boldsymbol{x}_{t_L}$. To optimize $B(\boldsymbol{x}, \theta)$, we need to backpropagate the gradient of this loss through every point proceeding $\boldsymbol{x}_{t_L}$, using backwards flowing time τ. To summarize the notation and to set the stage for the proof of finite gradient update, we introduce the following definitions:

Definition 3.1 (Differentiable Trajectory)**.** A Differentiable Trajectory $\mathcal{T}$ is defined by the following quadruple $(\boldsymbol{z}(t), L(\boldsymbol{z}(t_L)), f(\boldsymbol{z}(t)), \tilde{f}(\boldsymbol{z}(t)))$: Let $\boldsymbol{z}(t) \in \Omega_{\boldsymbol{x}} \times \Omega_{\boldsymbol{p}}$ where $\Omega_{\boldsymbol{x}} \subset \mathbb{R}^N$ and $\Omega_{\boldsymbol{p}} \subset \mathbb{R}^N$ for $t \in (t_i, t_e)$, where $-\infty \leq t_i < t_e \leq \infty$ be the sequence of states generate by the dynamics $f(\boldsymbol{z}(t))$ from a certain initial state $\boldsymbol{z}(t_0), t_0 \in [t_i, t_e]$. We define a loss function $L(\boldsymbol{z}(t_L))$ in time t_L. The gradient dynamics of the loss function is guided by the dynamics $\tilde{f}(\boldsymbol{z}(t))$ that includes possible $.detach()$ operators. The backward dynamics is represented in the reverse flowing time τ starting from $\boldsymbol{z}(t_L) = \boldsymbol{z}(\tau = 0)$ to $\boldsymbol{z}(\tau_e) = \boldsymbol{z}(t_i)$.

And we define a Diffusive Differentiable Trajectory by

Definition 3.2 (Diffusive Differentiable Trajectory)**.** A Differentiable Trajectory $\mathcal{T}$ constructed by dynamics (3) equipped with a backward dynamics (15) and a loss function (8) is called a Diffusive Differentiable Trajectory, denoted by $\mathcal{T}_d$.

Property 1 (Finite gradient update)**.** *Let γ be sufficiently high. Let*

$$U(\boldsymbol{x}) \in C^2(\Omega_{\boldsymbol{x}}) \text{ and } \frac{\partial f(\boldsymbol{z}(\tau), \theta)}{\partial \theta} \text{ bounded} \tag{16}$$

Then the gradient update for a loss function in a Diffusive Differentiable Trajectory is finite for every (possibly infinite) τ_e.

The essence of the proof and specification of the sufficient conditions for γ are addressed in the following theorem.

Theorem 3.3 (Converging adjoints)**.** *$\mathcal{T}_d$ be a Diffusive Differentiable Trajectory. Let $U(\boldsymbol{x}) \in C^2(\Omega_{\boldsymbol{x}})$ and denote the*

spectrum of its hessian $\frac{\partial^2 U(\boldsymbol{x}(t))}{\partial^2 \boldsymbol{x}}$ *by* $\lambda_i(\boldsymbol{x}(t))$*. Define* λ_{min} *as*

$$\lambda_{min} = \inf_{\tau>0} \min_i \lambda_i(\boldsymbol{x}(\tau)). \tag{17}$$

Then for every γ *that fulfills:* $(\Delta\tau\lambda_{min} + \gamma) = \epsilon > 0$*, it holds:*

$$\forall \tau > 0 : \|\boldsymbol{a}(\tau)\|^2 \leq \|\boldsymbol{a}(0)\|^2 e^{-2\epsilon\tau} \tag{18}$$

We proof the theorem in the Appendix A. There is a useful corollary of the above

Corollary 3.4. *Under the assumptions of the theorem 3.3,* $\|\boldsymbol{a}(\tau)\| \in L^r(0, \tau_e),\ r \in [1, \infty]$.

Proof. Case $r = \infty$ is trivial as the square root of the upper bound (18) is still finite $\forall\tau$. Let us now consider only $r \in [1, \infty)$.

$$\begin{aligned}\|\boldsymbol{a}(\tau)\|^r_{L^r(0,\tau_e)} &= \int_0^{\tau_e} \|\boldsymbol{a}(\tau)\|^r \leq \|\boldsymbol{a}(0)\|^r \int_0^{\tau_e} e^{-\epsilon\, r\, \tau} \\ &= -\frac{\|\boldsymbol{a}(0)\|^r}{\epsilon\, r} \left[e^{-\epsilon\, r\, \tau}\right]_0^{\tau_e}\end{aligned} \tag{19}$$

Which is finite for every value of τ_e including ∞. □

Proof of the finite gradient update 1. Since we know that $\boldsymbol{a}(\tau) \in L^1(0, \tau_e)$ from the previous corollary and that $\frac{\partial f(\boldsymbol{z}(\tau),\theta)}{\partial\theta}$ bounded from the assumption, it is now trivial to show

$$\begin{aligned}\frac{\partial L(\boldsymbol{z}_{t_L})}{\partial\theta} &= \int_0^{\tau_e} \boldsymbol{a}(\tau) \frac{f(\boldsymbol{z}(\tau),\theta)}{\partial\theta} d\tau \\ &\leq \sup_{\boldsymbol{z}(\tau)\in T} \left\| \frac{\partial f(\boldsymbol{z}(\tau),\theta)}{\partial\theta} \right\| \int_0^{\tau_e} \boldsymbol{a}(\tau) d\tau,\end{aligned} \tag{20}$$

which is finite. □

This property allows us to backpropagate the dynamics without exploding gradients as long as γ is chosen large enough. The exponential scaling of the adjoints also indicates that once we identify the point where the loss function will be calculated, we only need to consider a handful of points before $\boldsymbol{a}(\tau)$ essentially vanishes. Any further adjoint propagation does not significantly contribute to the gradient update. This is intuitively desirable, as for the noisy equation (3) the loss function information becomes diluted as we backpropagate. Keeping only recent data points thus introduces a natural cutoff to the information we use for optimization.

The theorem also gives more insight into when such backpropagation may lead to exploding gradients. If the expression $(\Delta\tau\lambda_{min} + \gamma) = \epsilon < 0$, then the upper bound may not hold, and gradients can increase exponentially. Strongly negative λ_i of the hessian indicates a concave part in the potential landscape, which is generally problematic for control. However, with $\Delta\tau$ and γ, we have two robust dials to ensure non-exploding adjoints.

In practice, we assume $\Delta\tau$ to be equal to the forward timestep. Investigating the impact of setting $\Delta\tau$ to multiples of the timestep is beyond the scope of this paper.

3.3. Mini-batching the graph

One of the problems associated with differentiable simulation is the low number of updates. Usually, only one gradient step is taken per trajectory, making the gradients averaged across the entire path and necessitating rather large learning rates to train the network in just a few updates. The problem can be alleviated by a technique we call *graph mini-batching*. The idea is to calculate trajectory depended gradients first (the adjoints $\boldsymbol{a}$) in one pass and then split them to mini-batches. The adjoints are then used as vectors in Jacobi-vector products (12) during backpropagation of the bias function evaluated in batches. The approach stabilizes learning and allows for much lower learning rates, better suited for training neural networks. An example of a use case is more thoroughly discussed in Appendix C.

3.4. Summary

The use of the Langevin thermostat with reasonable γ creates finite memory dynamics and therefore decaying adjoints. Employing also the *.detach()* operator ensures that the adjoints vanish smoothly, without high frequency oscillations, thus making them bounded (solving Item 1) and ignoring fast motion, helping with Item 2. Such a finite memory system is likely to be less chaotic, addressing Item 3. Splitting the loss gradient into random mini-batches obviously solves the point 4.

4. Practical implementation

It is important to promote the transition across the barrier equally. If only one direction is sampled, then one may end up with a "landslide" potential strongly tilted towards one minimum and not a diffusive behaviour. Therefore, we choose the following approach.

1. Create a batch of $2l$ starting configurations, with l in each well respectively.
2. Run all trajectories simultaneously for a fixed number of time steps.
3. Collect the loss 8 after all simulations have ended. After N initial steps, which serve as equilibration, we also calculate the minimal distance from the start to encourage the eventual return to the starting well and, thereby, true diffusive behavior. Thus, we have two

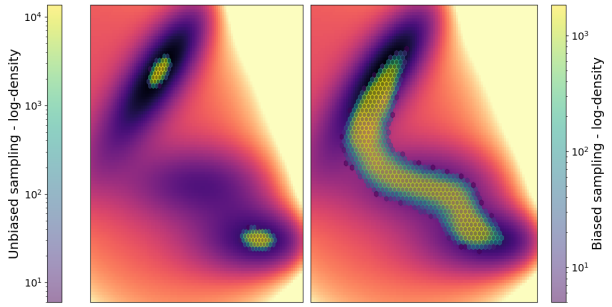

Figure 2: Log-density of simulated points before (*left*) and after the training (*right*) of bias function by differentiable simulations. The right plot shows how well all important regions are sampled after training. The background of the Figure is the $U_{\mathrm{MB}}(x, y)$, the underlying Muller-Brown potential.

losses: A forward loss $L_f(\boldsymbol{x}_{L_f})$ and start loss $L_s(\boldsymbol{x}_{L_s})$ that are summed together with equal weights.

4. Calculate adjoints and optimize the bias function $B(\boldsymbol{x}, \theta)$ using the graph mini-batching technique. Repeat from step 1 until convergence.

By running a large number of simulations concurrently, one can leverage the vectorization of the operations and reduce computational time.

For the performance monitoring we also define the **success rate**. This metric represents the percentage of trajectories that started in one basin and made it to the other over the course of a simulation.

5. Results

In this Section, we present the results of our novel DiffSim approach. First, we apply it to a commonly used two dimensional model PES, the Muller-Brown potential(Müller & Brown, 1979), where any linear combination of the Cartesian coordinates does not yield a good CV. Then we lift this example to five dimensions by introducing three noisy DoFs demonstrating the efficiency of the approach in a higher-dimensional setup. Second, we investigate the benchmark system for enhanced sampling in molecular systems, alanine dipeptide (amino acid alanine capped at both ends). The two collective variables describing the metastable states are well known in the biophysics community, the backbone dihedrals ϕ and ψ. We will assume no such knowledge and generate the enhanced sampling simulation from all backbone dihedral angles as candidates in an end-to-end process.

5.1. 2D Muller-Brown potential

The parameters of the commonly investigated 2D Muller-Brown PES (Müller & Brown, 1979; Sun et al., 2022) are given in the Appendix D. For the bias potential, $B(\boldsymbol{x}, \boldsymbol{h})$, we employ a grid of Gaussian functions, controlling their individual height. The biasing function is

$$B(\boldsymbol{x}, \boldsymbol{h}) = \sum_{i=1}^{n_g^2} h_i \exp\left(-\frac{\left(\boldsymbol{x} - \boldsymbol{x}_i^0\right)^2}{2\sigma^2}\right) \tag{21}$$

with trainable $\boldsymbol{h}$. Means $\boldsymbol{x}_i^0$ are evenly distributed in the computational domain. With n_g Gaussians along each dimension, only n_g^2 contributions to the total bias have to be calculated in two dimensions. After training the bias via DiffSim (parameters reported in Appendix G), we obtain biased dynamics that generate increasingly many successful transitions between reactants and products along the transition path (see the evolution of the loss function and success rate during the training shown in Figure 3). This leads to the log-density of the points along the transition path to even out significantly (see Figure 2).

We construct the CV by dimensionality reduction of frames from converged diffusive trajectories (well sampled transitions). To obtain a one dimensional CV describing the path, we use a Variational Autoencoder (Kingma & Welling, 2013) (architecture described in Appendix G). The resulting CV is visualized in Figure 3. The CV distinguishes well and interpolates smoothly between products and reactants. Using this CV, the unbiased and biased PES are plotted as averages along the CV. It is easy to see in Figure 3 how effectively the PES has been flattened by the bias function.

5.2. 5D Generalization of Muller-Brown potential

The situation is more complicated when additional harmonic degrees of freedom are included (see Appendix E for details). One may consider them to be, e.g., quickly oscillating hydrogen atoms that do not influence the reaction. As the ansatz of bias potential (21) scales exponentially with the dimensionality of the problem, it cannot be used with the 5D version of the potential (Appendix E). Instead, a fully connected neural network as a function of all five variables is employed, making training significantly harder. The results were postprocessed analogously as the two dimensional case, see Figure 7. Finally, the biased system converges to a success rate of 65 %. As before, we observe that the potential was relatively flattened and transitions occur with high probability. The results are not as good as in the 2D case, both due to the built-in noisiness of the dynamics (Appendix E) and because of a rather crude approximation of the biasing potential with a simple fully connected neural network, which is harder to train than the Gaussian grid used in a 2D case. However, even under such circumstances, the

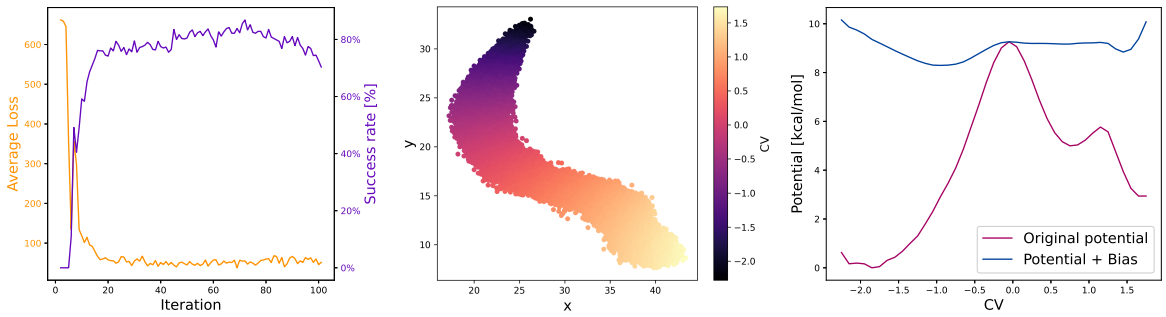

Figure 3: *left:* Loss functions and the probability of barrier crossing (success rate) during as the training progresses. *middle:* Variational Autoencoder producing a collective variable by training on a fully diffusive trajectory. *right:* Potential energy along the VAE collective variable with and without bias.

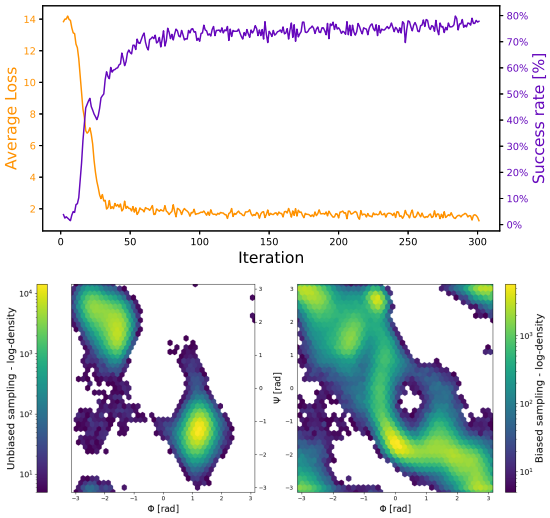

Figure 4: *top row*: Metrics for the alanine dipeptide run. Total loss function and transition success rate. *bottom row*: Log-density of simulated points before the training of bias by differentiable simulations (*left*) and after the training converged (*right*).

DiffSim approach managed to discover the transformation path and sample it with reasonably high probability.

5.3. Alanine dipeptide

Alanine dipeptide is a simple model system exhibiting typical protein dihedral dynamics. Therefore, it has become an important benchmark to test and verify free energy calculation methods. The collective variables, the dihedral angles ϕ and ψ, are well known and the PES is rather complex with relatively low barriers (Vymětal & Vondrášek, 2010; Mironov et al., 2019). We use this system to test the ability of our method to bias the dynamics along the important DoFs. The details about the location of minimas and biasing function inputs are reported in Appendix G.

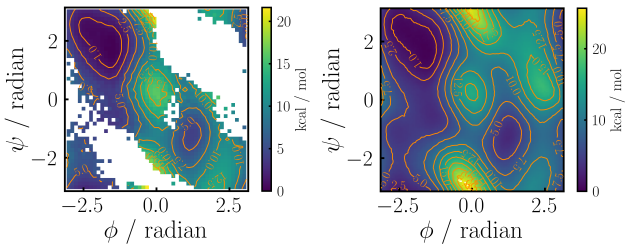

Figure 5: *left:* Directly reweighted PMF from simulations with the bias potential projected on the Ramachandran plane. White regions are without sufficient sampling. *right:* PMF of the ϕ-ψ-plane obtained from an eABF simulation.

The progress of the training and log-density of points are reported in Figure 4. After training, almost 80 % of trajectories show a transition within 10 ps. By comparing the potential of mean force (PMF) obtained from direct reweighting of 10 biased 1 ns runs with the fully trained potential with one obtained using the extended-Lagrangian adaptive biasing force (eABF, see Appendix F) we can see that they are identical in the sampled regions (see Figure 5). Whereas the eABF simulation explored the full $\phi - \psi$-space as intended by the algorithm, the simulations with the trained bias potential samples only that section of the same PMF, which is important for the $\beta \rightleftharpoons L_\alpha$ transitions . This demonstrates that our differentiable simulations can unravel the transition paths and reaction barriers in the same way as a collective variable based method would, except without requiring prior knowledge of ideal CVs.

6. Conclusion

This contribution presents advances in two distinct areas. First, it was described in detail how neural network controlled differential simulations (DiffSims) can be made robust and efficient. We have shown how the use of Langevin dynamics creates a finite memory horizon and therefore en-

forces decaying adjoints. The *.detach()* operator pruned the computational graph and thereby ensured that the adjoints vanish smoothly, removing any high frequency oscillations. The introduction of random mini-batches by breaking up the loss gradient made learning in the fashion of stochastic gradient descent possible, significantly stabilizing the training.

Second, the establishment of the robust and efficient neural network controlled DiffSims allowed us to successfully tackle an important open problem in computational chemistry - discovery and effective sampling of the rare event (chemical reaction) pathways. Initially, a path integral loss was defined to measure the success of a molecular dynamics trajectory with regard to a crossing of an energy barrier, i.e., with regard to exhibiting a rare event. This loss was then used to train a bias potential to discover and accelerate the chemical transitions without a need to guess a low-dimensional representation of the chemical reaction, i.e., the collective variable (CV), *a priori*. We showed the effectiveness of this approach by successfully biasing the dynamics on the Muller-Brown potential, which is a numerical benchmark for traditional enhanced sampling schemes and *a priori* CV determination algorithms. Our method worked without any previous knowledge of the good CV, however, from the the biased trajectories exhibiting transitions a reduced representation, i.e., a CV, can be constructed. The quality of biasing and subsequent CV identification was practically perfect for the 2D case, and even the challenging 5D case with a significant amount of noise in the added harmonic DoFs converged, exhibiting a high probability of observing a transition event. Finally, a realistic chemical system (alanine dipeptide) was investigated. The bias potential was constructed considering dihedral angles as candidate degrees of freedom, including not only the two dihedrals commonly used as CVs but also other dihedrals that noised the transition. Our method successfully generated biased trajectories, which exhibited sought-for transitions between the two target minima with high probability.

We have demonstrated that differentiable simulations with our innovations can handle not only model systems but also complex molecular motions. In the future, we intend to extend the tool to more challenging reactions with complicated transition paths, such as protein motion and chemical reactions with multiple intermediate steps or competing reaction paths.

Acknowledgements

M.S. was supported by project No. START/SCI/053 of Charles University Research program and GACR grant 23-07616S. J.C.B.D. is thankful for the support of the Leopoldina Fellowship Program, German National Academy of Sciences Leopoldina, grant number LPDS 2021-08. L.G. acknowledges the support of Primus Research Program of the Charles University (PRIMUS/20/SCI/004). R.G.-B. acknowledges support from the Jeffrey Cheah Career Development Chair. We thank Michal Pavelka for discussions regarding the nature of multiscale problems.

References

Abrams, C. and Bussi, G. Enhanced Sampling in Molecular Dynamics Using Metadynamics, Replica-Exchange, and Temperature-Acceleration. *Entropy 2014, Vol. 16, Pages 163-199*, 16(1):163–199, 12 2013. ISSN 1099-4300. doi: 10.3390/E16010163. URL https://www.mdpi.com/1099-4300/16/1/163/htmhttps://www.mdpi.com/1099-4300/16/1/163.

Belkacemi, Z., Gkeka, P., Lelièvre, T., and Stoltz, G. Chasing Collective Variables Using Autoencoders and Biased Trajectories. *Journal of Chemical Theory and Computation*, 18(1):59–78, 1 2022. ISSN 1549-9618. doi: 10.1021/acs.jctc.1c00415.

Bellman, R. Dynamic programming. *Mathematics in Science and Engineering*, 40(P1):101–137, 1967. ISSN 00765392. doi: 10.1016/S0076-5392(08)61063-2.

Bolhuis, P. G., Chandler, D., Dellago, C., and Geissler, P. L. Transition path sampling: Throwing ropes over rough mountain passes, in the dark. *Annu. Rev. Phys. Chem.*, 53:291–318, 2002.

Bonati, L., Rizzi, V., and Parrinello, M. Data-Driven Collective Variables for Enhanced Sampling. *The journal of physical chemistry letters*, 11(8):2998–3004, 4 2020. ISSN 1948-7185. doi: 10.1021/ACS.JPCLETT.0C00535. URL https://pubmed.ncbi.nlm.nih.gov/32239945/.

Boyer, C. and Merzbach Uta. *A History of Mathematics, Second Edition*. Wiley, 2 edition, 3 1991.

Callen, H. B. and Scott, H. L. *Thermodynamics and an Introduction to Thermostatistics, 2nd ed*, volume 66. 1998. doi: 10.1119/1.19071.

Cérou, F., Guyader, A., Lelièvre, T., and Pommier, D. A multiple replica approach to simulate reactive trajectories. *J. Chem. Phys.*, 134(5):054108, 2011. doi: 10.1063/1.3518708.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 2018-December, 2018a.

Chen, W., Tan, A. R., and Ferguson, A. L. Collective variable discovery and enhanced sampling using autoencoders: Innovations in network architecture and error

function design. *The Journal of Chemical Physics*, 149 (7):072312, 8 2018b. ISSN 0021-9606. doi: 10.1063/1.5023804.

Chipot, C. Frontiers in free-energy calculations of biological systems. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 4(1):71–89, 1 2014. ISSN 1759-0884. doi: 10.1002/WCMS.1157. URL https://onlinelibrary.wiley.com/doi/full/10.1002/wcms.1157https://onlinelibrary.wiley.com/doi/abs/10.1002/wcms.1157https://wires.onlinelibrary.wiley.com/doi/10.1002/wcms.1157.

Chipot, C. and Pohorille, A. *Free Energy Calculations*, volume 86 of *Springer Series in CHEMICAL PHYSICS*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. ISBN 978-3-540-38447-2. doi: 10.1007/978-3-540-38448-9. URL http://link.springer.com/10.1007/978-3-540-38448-9.

Darve, E. and Pohorille, A. Calculating free energies using average force. 2001. doi: 10.1063/1.1410978. URL http://jcp.aip.org/jcp/copyright.jsp.

de Avila Belbute-Peres, F., Smith, K., Allen, K., Tenenbaum, J., and Kolter, J. Z. End-to-End Differentiable Physics for Learning and Control. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/842424a1d0595b76ec4fa03c46e8d755-Paper.pdf.

Degrave, J., Hermans, M., Dambre, J., and wyffels, F. A Differentiable Physics Engine for Deep Learning in Robotics. *arXiv*, (1611.01652), 11 2016.

Dellago, C., Bolhuis, P. G., and Chandler, D. Efficient transition path sampling: Application to lennard-jones cluster rearrangements. *J. Chem. Phys.*, 108:9236, 1998.

Doerr, S., Majewski, M., Pérez, A., Krämer, A., Clementi, C., Noe, F., Giorgino, T., and De Fabritiis, G. TorchMD: A Deep Learning Framework for Molecular Simulations. *Journal of Chemical Theory and Computation*, 17(4): 2355–2363, 4 2021. ISSN 1549-9618. doi: 10.1021/acs.jctc.0c01343.

Foerster, J., Farquhar, G., Al-Shedivat, M., Rocktäschel, T., Xing, E. P., and Whiteson, S. DiCE: The Infinitely Differentiable Monte-Carlo Estimator. 2 2018.

Galimberti, C. L., Furieri, L., Xu, L., and Ferrari-Trecate, G. Hamiltonian Deep Neural Networks Guaranteeing Non-vanishing Gradients by Design. *arXiv*, (2105.13205), 5 2021.

Greener, J. G. and Jones, D. T. Differentiable molecular simulation can learn all the parameters in a coarse-grained force field for proteins. *PLOS ONE*, 16(9):e0256990, 9 2021. ISSN 1932-6203. doi: 10.1371/journal.pone.0256990.

Hu, Y., Li, T.-M., Anderson, L., Ragan-Kelley, J., and Durand, F. Taichi. *ACM Transactions on Graphics*, 38(6): 1–16, 12 2019. ISSN 0730-0301. doi: 10.1145/3355089.3356506.

Hu, Y., Anderson, L., Li, T.-M., Sun, Q., Carr, N., Ragan-Kelley, J., and Durand, F. DiffTaichi: Differentiable Programming for Physical Simulation. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=B1eB5xSFvr.

Huang, Z., Hu, Y., Du, T., Zhou, S., Su, H., Tenenbaum, J. B., and Gan, C. PlasticineLab: A Soft-Body Manipulation Benchmark with Differentiable Physics. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=xCcdBRQEDW.

Hulm, A., Dietschreit, J. C. B., and Ochsenfeld, C. Statistically Optimal Analysis of the Extended-system Adaptive Biasing Force (eABF) Method. *J. Chem. Phys.*, 157: 024110, 2022.

Ingraham, J., Riesselman, A., Sander, C., and Marks, D. Learning Protein Structure with a Differentiable Simulator. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Byg3y3C9Km.

Kingma, D. P. and Welling, M. Auto-Encoding Variational Bayes. *arXiv*, (1312.6114), 12 2013.

Köppen, M. The curse of dimensionality. 2000.

Laio, A. and Parrinello, M. Escaping free-energy minima. *Proceedings of the National Academy of Sciences of the United States of America*, 99(20):12562–12566, 10 2002. ISSN 00278424. doi: 10.1073/PNAS.202427399/ASSET/2381B9FD-FD4C-4ED9-9DCB-75FD02E8EA7E/ASSETS/GRAPHIC/PQ2024273003.JPEG. URL https://www.pnas.org/doi/abs/10.1073/pnas.202427399.

Lesage, A., Lelievre, T., Stoltz, G., and Henin, J. Smoothed biasing forces yield unbiased free energies with the extended-system adaptive biasing force method. *J. Phys. Chem. B*, 121(15):3676–3685, 2017.

Lev Semenovich Pontryagin, MishGamkrelidze RV, Boltyanskii VG, and Gamkrelidze RV. *The Mathematical Theory of Optimal Processes*. 1962.

Li, X., Wong, T.-K. L., Chen, R. T. Q., and Duvenaud, D. K. Scalable Gradients and Variational Inference for Stochastic Differential Equations. In *Proceedings of The 2nd Symposium on Advances in Approximate Bayesian Inference*, volume 118 of *Proceedings of Machine Learning Research*, pp. 1–28. PMLR, 9 2020. URL https://proceedings.mlr.press/v118/li20a.html.

Mendels, D., Piccini, G., and Parrinello, M. Collective Variables from Local Fluctuations. *Journal of Physical Chemistry Letters*, 9(11):2776–2781, 6 2018. ISSN 19487185. doi: 10.1021/ACS.JPCLETT.8B00733/ASSET/IMAGES/LARGE/JZ-2018-00733T{_}0005.JPEG. URL https://pubs.acs.org/doi/full/10.1021/acs.jpclett.8b00733.

Mertens, S. and Mingramm, S. Brachistochrones with loose ends. *European Journal of Physics*, 29(6):1191–1199, 11 2008. ISSN 0143-0807. doi: 10.1088/0143-0807/29/6/008.

Metz, L., Freeman, C. D., Schoenholz, S. S., and Kachman, T. Gradients are Not All You Need. *arXiv*, (2111.05803), 11 2021.

Mironov, V., Alexeev, Y., Mulligan, V. K., and Fedorov, D. G. A systematic study of minima in alanine dipeptide. *Journal of Computational Chemistry*, 40(2):297–309, 1 2019. ISSN 0192-8651. doi: 10.1002/jcc.25589.

Müller, K. and Brown, L. D. Location of saddle points and minimum energy paths by a constrained simplex optimization procedure. *Theoretica chimica acta*, 53(1):75–93, 1979. ISSN 1432-2234. doi: 10.1007/BF00547608. URL https://doi.org/10.1007/BF00547608.

Percival I. Chaos in hamiltonian systems. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 413(1844):131–143, 9 1987. ISSN 0080-4630. doi: 10.1098/rspa.1987.0105.

Rohrdanz, M. A., Zheng, W., and Clementi, C. Discovering Mountain Passes via Torchlight: Methods for the Definition of Reaction Coordinates and Pathways in Complex Macromolecular Reactions. *Annual Review of Physical Chemistry*, 64(1):295–316, 4 2013. ISSN 0066-426X. doi: 10.1146/annurev-physchem-040412-110006.

Schulman, J., Heess, N., Weber, T., and Abbeel, P. Gradient Estimation Using Stochastic Computation Graphs. *arXiv*, (1506.05254), 6 2015.

Schwantes, C. R. and Pande, V. S. Modeling Molecular Kinetics with tICA and the Kernel Trick. *Journal of Chemical Theory and Computation*, 11(2):600–608, 2 2015. ISSN 1549-9618. doi: 10.1021/ct5007357.

Šípka, M., Erlebach, A., and Grajciar, L. Understanding chemical reactions via variational autoencoder and atomic representations. *arXiv*, (2202.00817), 3 2022.

Spiwok, V., Sucur, Z., and Hosek, P. Enhanced sampling techniques in biomolecular simulations. *Biotechnology advances*, 33(6 Pt 2):1130–1140, 2015. ISSN 1873-1899. doi: 10.1016/J.BIOTECHADV.2014.11.011. URL https://pubmed.ncbi.nlm.nih.gov/25482668/.

Suh, H. J. T., Simchowitz, M., Zhang, K., and Tedrake, R. Do Differentiable Simulators Give Better Policy Gradients? 2 2022.

Sultan, M. M. and Pande, V. S. Automated design of collective variables using supervised machine learning. *The Journal of Chemical Physics*, 149(9):94106, 2018. doi: 10.1063/1.5029972. URL https://doi.org/10.1063/1.5029972.

Sun, L., Vandermause, J., Batzner, S., Xie, Y., Clark, D., Chen, W., and Kozinsky, B. Multitask Machine Learning of Collective Variables for Enhanced Sampling of Rare Events. *Journal of Chemical Theory and Computation*, 18(4):2341–2353, 2022. doi: 10.1021/acs.jctc.1c00143. URL https://doi.org/10.1021/acs.jctc.1c00143.

Tian, C., Kasavajhala, K., Belfon, K. A. A., Raguette, L., Huang, H., Migues, A. N., Bickel, J., Wang, Y., Pincay, J., Wu, Q., and Simmerling, C. ff19SB: Amino-Acid-Specific Protein Backbone Parameters Trained against Quantum Mechanics Energy Surfaces in Solution. *Journal of Chemical Theory and Computation*, 16(1):528–552, 1 2020. ISSN 1549-9618. doi: 10.1021/acs.jctc.9b00591.

Torrie, G. M., Valleau, J. P., Torrie, G. M., and Valleau, J. P. Nonphysical Sampling Distributions in Monte Carlo Free-Energy Estimation: Umbrella Sampling. *JCoPh*, 23(2):187–199, 1977. ISSN 0021-9991. doi: 10.1016/0021-9991(77)90121-8. URL https://ui.adsabs.harvard.edu/abs/1977JCoPh..23..187T/abstract.

Valsson, O., Tiwary, P., and Parrinello, M. Enhancing Important Fluctuations: Rare Events and Metadynamics from a Conceptual Viewpoint.

http://dx.doi.org/10.1146/annurev-physchem-040215-112229, 67:159–184, 5 2016. ISSN 0066426X. doi: 10.1146/ANNUREV-PHYSCHEM-040215-112229. URL https://www.annualreviews.org/doi/abs/10.1146/annurev-physchem-040215-112229.

Van Erp, T. S., Moroni, D., and Bolhuis, P. G. A novel path sampling method for the calculation of rate constants. *J. Chem. Phys.*, 118:7762, 2003.

Vymětal, J. and Vondrášek, J. Metadynamics As a Tool for Mapping the Conformational and Free-Energy Space of Peptides — The Alanine Dipeptide Case Study. *The Journal of Physical Chemistry B*, 114(16):5632–5642, 4 2010. ISSN 1520-6106. doi: 10.1021/jp100950w.

Wang, D. and Tiwary, P. State predictive information bottleneck. *The Journal of Chemical Physics*, 154(13):134111, 4 2021. ISSN 0021-9606. doi: 10.1063/5.0038198. URL https://aip.scitation.org/doi/abs/10.1063/5.0038198.

Wang, W., Axelrod, S., and Gómez-Bombarelli, R. Differentiable Molecular Simulations for Control and Learning. *arXiv*, (2003.00868), 2 2020.

Wang, Y., Ribeiro, J. M. L., and Tiwary, P. Past–future information bottleneck for sampling molecular reaction coordinate simultaneously with thermodynamics and kinetics. *Nature Communications 2019 10:1*, 10 (1):1–8, 8 2019. ISSN 2041-1723. doi: 10.1038/s41467-019-11405-4. URL https://www.nature.com/articles/s41467-019-11405-4.

Wehmeyer, C. and Noé, F. Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics. *The Journal of Chemical Physics*, 148 (24):241703, 2018. doi: 10.1063/1.5011399. URL https://doi.org/10.1063/1.5011399.

Zhang, S.-X., Wan, Z.-Q., and Yao, H. Automatic Differentiable Monte Carlo: Theory and Application. *arXiv*, (1911.09117), 11 2019.

A. Proof of adjoint convergence theorem

To prove Theorem 3.3 we need to state one more lemma.

Lemma A.1. *Let $\boldsymbol{x} \in \mathbb{R}^N$, $U(\boldsymbol{x})$ scalar, real, $C^2(\mathbb{R}^N)$ function. Consider a hessian computed at $\boldsymbol{x}_0$: $\frac{\partial^2 U(\boldsymbol{x}_0)}{\partial \boldsymbol{x}^2}$ with minimum and maximum eigenvalues λ_{min} and λ_{max} respectively. Then for any vector $\boldsymbol{v} \in \mathbb{R}^N$*

$$\lambda_{min} \left\|\boldsymbol{v}\right\|^2 \leq \boldsymbol{v}^T \cdot \frac{\partial U(\boldsymbol{x}_0^2)}{\partial \boldsymbol{x}^2} \boldsymbol{v} \leq \lambda_{max} \left\|\boldsymbol{v}\right\|^2. \tag{22}$$

Proof. We note that a hessian of a real continuous function is a symmetric matrix. Such a matrix is orthogonally diagonalizable and has real eigenvalues. The rest of the proof is a part of most standard linear algebra textbooks. □

We can now prove the Theorem 3.3.

Proof. We start by multiplying (15) by $2\boldsymbol{a}$. This yields

$$2\boldsymbol{a}(\tau) \cdot \dot{\boldsymbol{a}}(\tau) = 2\mathrm{d}\tau \boldsymbol{a}^T(\tau) \cdot \frac{\partial^2 U(\boldsymbol{x}(\tau))}{\partial^2 \boldsymbol{x}} \boldsymbol{a}(\tau) - 2\gamma \left\|\boldsymbol{a}(\tau)\right\|^2 \tag{23}$$

and can be recast using $2\boldsymbol{a}^T(\tau) \cdot \dot{\boldsymbol{a}}(\tau) = \dot{\overline{\left\|\boldsymbol{a}(\tau)\right\|^2}}$ (the norm is a standard vector 2-norm) to

$$\dot{\overline{\left\|\boldsymbol{a}(\tau)\right\|^2}} = -2\mathrm{d}\tau \boldsymbol{a}^T(\tau) \cdot \frac{\partial^2 U(\boldsymbol{x}(\tau))}{\partial^2 \boldsymbol{x}} \boldsymbol{a}(\tau) - 2\gamma \left\|\boldsymbol{a}(\tau)\right\|^2 \tag{24}$$

Using Lemma A.1 and, subsequently, the assumption of the theorem, we can estimate the upper bound of the time derivative as

$$\dot{\overline{\left\|\boldsymbol{a}(\tau)\right\|^2}} \leq -2\left(\mathrm{d}\tau\lambda_{min} + \gamma\right) \left\|\boldsymbol{a}(\tau)\right\|^2 = -2\epsilon \left\|\boldsymbol{a}(\tau)\right\|^2 \tag{25}$$

Using Gromwall lemma we can now estimate $\boldsymbol{a}(\tau)$ easily as

$$\left\|\boldsymbol{a}(\tau)\right\|^2 \leq \left\|\boldsymbol{a}(0)\right\|^2 \exp\left(-2\int_0^\tau \epsilon \,\mathrm{d}t\right) = \left\|\boldsymbol{a}(0)\right\|^2 e^{-2\epsilon\tau} \tag{26}$$

and since $\epsilon > 0$, the $\left\|\boldsymbol{a}(\tau)\right\|^2$ is bounded for all τ. □

B. Un-Detached formulation and convergence of adjoints

The proof of adjoint convergence after introduction of the *.detach()*-operator (Appendix A) raises the question how adjoints evolve for the full, un-detached set of equations. It appears that the adjoint decay cannot be ensured in this case, underlining the need for the use of detached dynamics when treating chaotic systems. In other words, there exists a concave potential that would, regardless of γ, result in exploding gradients.

Let us consider a simple case where both $\boldsymbol{x}$ and $\boldsymbol{p}$ are one dimensional and the PES is the quadratic potential $U(\boldsymbol{x}) = \alpha\boldsymbol{x}^2$. In this case, the adjoint equation is that of a damped harmonic oscillator. The original equations:

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{p}(t) \\ \dot{\boldsymbol{p}}(t) &= -\beta\boldsymbol{x} - \gamma\boldsymbol{p}(t) + \sqrt{2\gamma k_B T}\boldsymbol{R}(t). \end{aligned} \tag{27}$$

with $\beta = 2\alpha$ and the adjoints:

$$\begin{aligned} \dot{\boldsymbol{a}}_{\boldsymbol{x}}(t) &= \boldsymbol{a}_{\boldsymbol{p}} \\ \dot{\boldsymbol{a}}_{\boldsymbol{p}}(t) &= -\beta\boldsymbol{a}_{\boldsymbol{x}} - \gamma\boldsymbol{a}_{\boldsymbol{p}} \end{aligned} \tag{28}$$

which is the classic damped oscillator. This equation can be solved analytically by transforming it to a second order equation

$$\ddot{\boldsymbol{a}}_{\boldsymbol{x}} = -\beta\boldsymbol{a}_{\boldsymbol{x}} - \gamma\dot{\boldsymbol{a}}_{\boldsymbol{x}} \tag{29}$$

This linear ODE system has a well-known solution in the form of a linear combination of two exponentials. The characteristic equation in terms of variable χ for this system is

$$\chi^2 + \gamma\chi + \beta = 0. \tag{30}$$

The general solution can be obtained as:

$$\chi = \frac{-\gamma \pm \sqrt{\gamma^2 - 4\beta}}{2}. \tag{31}$$

The solution depends on the sign of β and on the value under the square root. For this counterexample (i.e., uncontrollable adjoints) we are interested in a concave potential, represented by negative β. Then the roots are real, such that $\chi_1 > 0$ and $\chi_2 < 0$. This translates to a general solution

$$\boldsymbol{a_x}(\tau) = Ae^{\chi_1\tau} + Be^{\chi_2\tau} \tag{32}$$

It is possible to choose the constants A and B such that both are non-zero. In this case, no matter how small χ_1 is, the expression exponentially diverges with τ. In other words, no matter how large we choose γ, the negative β always pushes our solution to ∞. This proves that in the general un-detached setting, there are cases of potentials that produce diverging adjoints regardless of the damping.

Note 1: We do not discuss special solutions, like one that sets $A = 0$. This choice represents dynamics that come to rest at the maximum of the potential. It is easy to find initial conditions that do not result in this scenario.

Note 2: The entire derivation was based on well-known principles. The adjoint equation is by Pontryagin principle guided by the Legendre transform of the original Lagrangian. When the potential is quadratic, the resulting equations stay the same in adjoint formulation. From the damped harmonic oscilator in the original phase space we create a harmonic oscilator in the control variables.

C. Graph minibatching and adjoints

To better explain the graph minibatching technique, let us consider a simple differential equation with trainable parameters θ

$$\dot{z} = f(\boldsymbol{z}, \theta) \tag{33}$$

Let us discretize the equation using a simple Forward Euler method such that it becomes

$$z_{n+1} = z_n + dtf(z_n, \theta). \tag{34}$$

For simplicity consider a three step differentiable simulation (z_0, z_1, z_2) such that

$$\begin{aligned} z_2 &= z_1 + dtf(z_1, \theta) \\ z_1 &= z_0 + dtf(z_0, \theta) \end{aligned}$$

where a loss function is defined for the last point $L(z_2)$. Our goal is to find the gradient of $\frac{\partial L(z_2)}{\partial \theta}$. Let us derive

$$\begin{aligned} \frac{\partial L(z_2)}{\partial \theta} &= \frac{\partial L(z_2)}{\partial z_2}\frac{\partial z_2}{\partial \theta} \\ \frac{\partial z_2}{\partial \theta} &= \frac{\partial z_1}{\partial \theta} + dt\frac{\partial f(z_1,\theta)}{\partial \theta} = \frac{\partial z_1}{\partial \theta} + dt\left(\frac{\partial f(z_1,\theta)}{\partial z_1}\frac{\partial z_1}{\partial \theta} + \frac{\partial f(z_1,\theta)}{\partial \theta}\right) \\ \frac{\partial z_1}{\partial \theta} &= \frac{\partial z_0}{\partial \theta} + dt\frac{\partial f(z_0,\theta)}{\partial \theta} = dt\frac{\partial f(z_0,\theta)}{\partial \theta}. \end{aligned}$$

Put together,

$$\frac{\partial L(z_2)}{\partial \theta} = \frac{\partial L(z_2)}{\partial z_2}\left[dt\left(1 + dt\frac{\partial f(z_1,\theta)}{\partial z_1}\right)\frac{\partial f(z_0,\theta)}{\partial \theta} + dt\frac{\partial f(z_1,\theta)}{\partial \theta}\right]. \tag{35}$$

Meaning, when we optimize the biased function $f(z_n, \theta)$ We can split the derivative into two parts

$$\begin{aligned} &\left[\frac{\partial L(z_2)}{\partial z_2}dt\left(1 + dt\frac{\partial f(z_1,\theta)}{\partial z_1}\right)\right]\frac{\partial f(z_0,\theta)}{\partial \theta} \\ &\left[\frac{\partial L(z_2)}{\partial z_2}dt\right]\frac{\partial f(z_1,\theta)}{\partial \theta} \end{aligned} \tag{36}$$

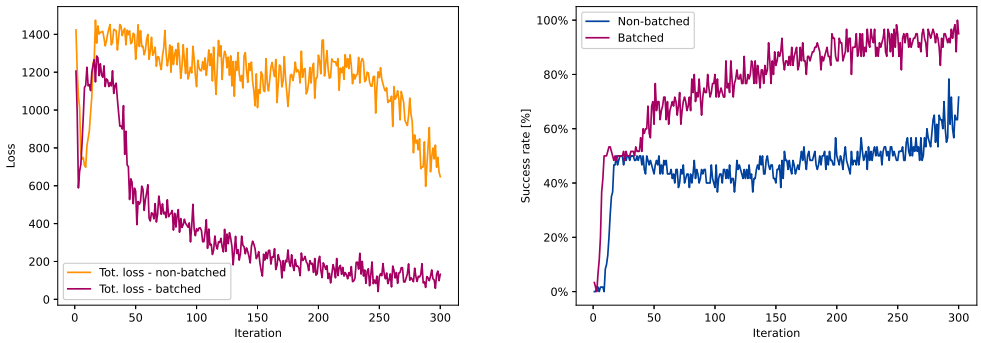

Figure 6: Comparison of the batched and one-time update of the weights in the 2D example from 5.1. The learning rate for the unbatched example was set approximately a number of batches times larger than for the batched run. The convergence is clearly more stable and even faster in the batched case. This was also observed for any other setting we tried during the development.

More steps can be obtained by continuing the iterations. One can easily see that the vectors we put into square brackets are actually the adjoints $a(z_n)$ from (11). By saving these vectors, we can then take z_n, feed forward through $f(z_n, \theta)$ and backpropagate using the vector jacobian product. This can be done in one gradient update, accumulating a gradient with respect to θ and updating it after going through all adjoints, or we can update weights in batches as it is common in neural network training. The latter is shown to be the more stable and faster converging of the methods (see Figure 6).

D. 2D Muller-Brown potential

The equation of the PES:

$$U_{\mathrm{MB}}(x, y) = B \sum_{i=1}^{4} A_i \exp\left[\alpha_i (x - x_0)^2 + \beta_i (x - x_0)(y - y_0) + \gamma_i (y - y_0)^2\right] \tag{37}$$

The parameters used in this work are:

i	A_i	α_i	β_i	γ_i	x_0	y_0
1	-1.73	0	-0.39	-3.91	48	8
2	-0.87	0	-0.39	-3.91	32	16
3	-1.47	4.3	-2.54	-2.54	24	31
4	0.13	0.23	0.273	0.273	16	24

The barrier parameter $B = 10$ kcal/mol

E. 5D Generalization

We consider a generalization of the Muller-Brown potential. By adding three harmonic DoFs we complicate the problem and make it necessary to use a general form of a biasing potential, dependent on all degrees of freedom, as we do not know which of them defines the reaction. The resulting potential has the form:

$$U_{5D}(x_1, x_2, x_3, x_4, x_5) = U_{\mathrm{MB}}(x_1, x_3) + \kappa(x_2^2 + x_4^2 + x_5^2) \tag{38}$$

The parameters for $U_{\mathrm{MB}}(x_1, x_3)$ are identical to the 2D-case, the new parameter $\kappa = 0.1$ The results for the 5D case are visualized in the figure Figure 3.

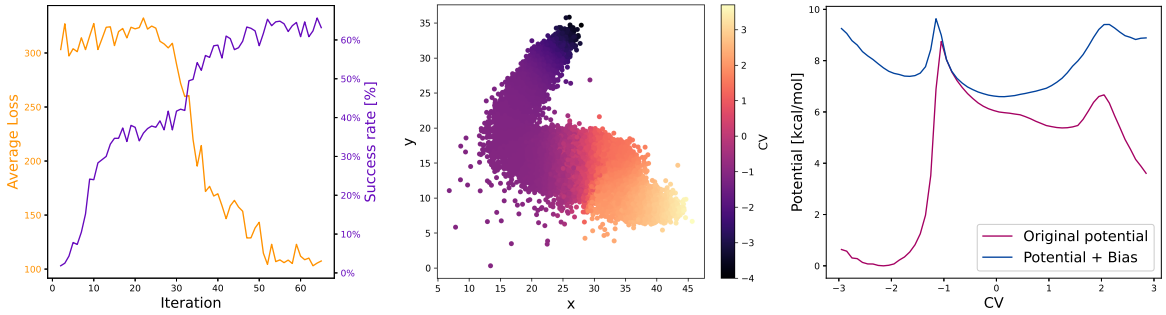

Figure 7: Results for the 5D extension of the Muller-Brown potential. *left:* Evolution of loss value and probability of barrier crossing during the training progresses. *middle:* CV determined with a Variational Autoencoder trained on a fully diffusive trajectory. The collective variables are not sharp around the transition region due to the high variance of the other noisy DoFs. This could be improved by more data, and more refined dimensionality reduction techniques that include temporal data such as e.g TiCA (Schwantes & Pande, 2015) or time-lagged autoencoders (Wehmeyer & Noé, 2018). *right:* Average potential energy along the VAE collective variable with and without bias. The barriers were lowered to the level where they could be crossed with high probability.

F. Alanine dipeptide simulation settings

The $\phi - \psi$-PMF of alanine dipeptide was obtained by means of eABF simulations(Lesage et al., 2017). Both angles were coupled to independent particles with a mass of 25 a.m.u. with a coupling width of $\sigma = 5°$. The ABF bias was collected on a grid with a bin width of σ and the bias was downscaled with a linear ramp function until at least 25 samples were collected in a bin. The simulation time step was 1 fs and the temperature was kept at 300 K with the Langevin thermostat with a friction constant of 1 ps^{-1} for both the real and the extended system. The total simulation time was 10 ns. The unbiased Boltzmann weights were recovered with MBAR(Hulm et al., 2022), the convergence criterion was set to 10^{-6}.

G. Differentiable simulation parameters

The equations we simulate are (3), discretized by the Leapfrog algorithm. The method is symplectic and conserves energy. The constants and parameters of the method were chosen as follows:

case	m [g/mol]	γ [ps^{-1}]	T [K]	dt [fs]	timesteps	epochs
2D	0.1	0.1	10	1	6×10^3	101
5D	0.01	1.0	300	1	2×10^4	66
Ala2	-	0.1	300	1	10^4	301

The column "timesteps" lists for how many steps we propagate a single simulation in each epoch. The update of parameters then represents an epoch. For the backward dynamics, we use 190 adjoints directly before the point where the loss function is calculated.

Batches of Parallel MDs These batches refer to the number of replicas that are simulated simultaneously. Using GPUs, we can parallelize the computation of forces and time step integration and thus are able to run 600 systems at once with a similar speed of running just one. Accumulating the simulated data from so many systems allows us to increase the number of adjoints obtained and enables us to use a lower learning rate, making the training more stable.

The setup of the bias function differed for every test case:

2D Muller-Brown: In this case, we use the setup described in (21) with 50 times 50 basis functions.

5d Muller-brown: We use a fully connected network with all five degrees of freedom used as five continuous input neurons. The network has four hidden layers, each 150 neurons with SiLU as activation functions. The final layer has a single output neuron - the bias - and no activation.

Alanine dipeptide: In the case of real molecules, the bias function gets more complicated. We define the Gaussian basis set

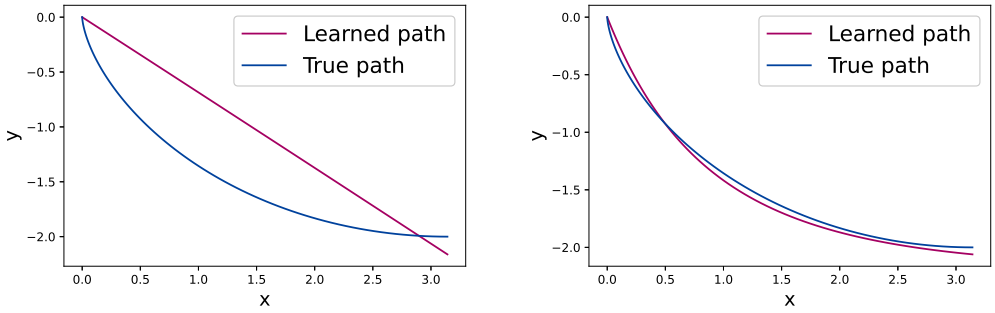

Figure 8: *left* Initial state. The path is initialized as an almost straight path. *right* After 200 iterations of differentiable simulations training, the path approximates the true path. The difference between the true curve and the one obtained by training is likely in the numerical scheme used to evaluate the integral.

in every single degree of freedom represented by a basis vector $\mathbf{e}_j$ and form a vector

$$\boldsymbol{v}(\boldsymbol{x}) = \sum_{j=1}^{n_{dof}} \sum_{i=1}^{n_g} \exp\left(-\frac{(\boldsymbol{x} - \boldsymbol{x}_{ij}^0)^2}{2\sigma^2}\right) \mathbf{e}_j. \tag{39}$$

n_{dof} represents the total number of candidate CVs or degrees of freedom considered. In our case, this was 5. n_g is the total number of basis functions defined separately for every candidate CV. In our case, this was 50 and since we described dihedral angles, centers $\boldsymbol{x}_{ij}^0$ were distributed uniformly from 0 to 2π, respecting the periodicity of dihedrals. The flattened vector $\boldsymbol{v}(\boldsymbol{x})$ with size $n_{dof} \cdot n_g$ is then used as an input to a fully connected neural network with three hidden layers, each 150 neurons with SiLU as an activation function. The final layer has one output neuron without an activation function.

As reactant and product we use the minima β ($\phi \approx -2,\ \psi \approx 2$) and $\mathrm{C7_{ax}}$ ($\phi \approx 1,\ \psi \approx -1.5$), respectively (compare right panel in Figure 5). As candidate DoFs, we choose the four dihedral angles along the backbone of alanine dipeptide, denoted as θ_1, ϕ, ψ, θ_2 in the Figure 1 from the paper (Mironov et al., 2019). To make thinks more complicated, we add one dihedral involving the side-chain methyl group that is expected to be correlated with ϕ, defined by atoms C_1, N_1, C_α, C_β using the notation from the same Figure. In each dihedral angle, a Gaussian basis set accounting for periodicity is defined Appendix F. These expanded dihedrals are then input to a fully connected network that calculates the bias function. The detailed settings are listed in Appendix G. As ϕ and ψ are known, the results are reported as Ramachandran plots. No additional dimensionality reduction via VAEs is performed in this example.

The numerical tool used for the DiffSim of alanine dipeptide was partially based on components from the TorchMD library (Doerr et al., 2021). The simulations were carried out with the Amber ff19SB forcefield (Tian et al., 2020) in vacuum.

Graph-Minibatching batch size This batch size refers to the mini-batching of the computational graph illustrated in Appendix C. Here we split the accumulated adjoints into smaller batches and train the network sequentially. The mini-batch of 120 was used for all systems. We use the learning rate as a learning factor divided by the number of replicas to make it independent of the number of systems simulated simultaneously. The learning factor is chosen as 20 for the 2D case, 6 for the 5D case and 3 for the Alanine dipeptide. This, with 300 replicas running from reactant to the product and 300 the other way, gives us learning rates on the orders 10^{-2} to 10^{-3}. We use Adam optimizer for all our cases.

For the CV construction via Variational Autoencoder a simple setup was employed with a two hidden layer encoder and a two hidden layer decoder with 50 neurons and a Softplus activation function for each hidden layer.

H. Brachistochrone curve

Here we exemplify how one can employ differentiable simulations and their capabilities to optimize path dependent integrals and solve the Brachistochrone problem. The problem is formulated as follows: Given a mass freely sliding on a curve $y = y(x)$ in the gravitational field g, find the curve from point **A** to lower point **B** for which the sliding time is the shortest. We assume no friction or air resistance and assume that **B** does not lie directly below **A**. For simplicity, we choose **A** to be the origin of the coordinate system. The solution, the cyclone curve, of this famous problem was obtained by Leibniz,

L'Hospital, Newton, and Bernoulli brothers (Boyer & Merzbach Uta, 1991). A modified version, where we allow for an arbitrary difference in height between the two points and only prescribe their horizontal distance Δx was solved by Lagrange and much later summarized and written in the modern language of variational formalism by (Mertens & Mingramm, 2008). In this case, a solution is also a cyclone with some parameters fixed. We prescribe the horizontal Δx to be π and search for a solution using differentiable simulations. A simple fully connected neural network $f(x)$ serves as a derivative of the curve $f(x) = \frac{dy(x)}{dx}$, so that $y(x)$ is then obtained by the path integration of the neural network. After integration, we numerically evaluate the time from the simulated path

$$t = \int_0^l \frac{ds}{v(x)} = \int_0^\pi \sqrt{\frac{1 + \left(\frac{dy(x)}{dx}\right)^2}{-2gy(x)}} dx \tag{40}$$

and minimize it. In the first integrat, l represents the length of the curve. The formula can be easily derived from the conservation of kinetic energy and from a Pythagorean expression $ds^2 = dx^2 + dy^2$. For the path construction and backpropagation we employ the *torchdiffeq* python package shipped with the paper (Chen et al., 2018a).

In this example, we present a problem that could not be solved with just a point-wise neural network optimization but requires consideration of a full path. The results for a short training are in Figure 8.

I. Data availability

The source code for all examples (2D and 5D Muller-Brown, Alanine Dipeptide) are available online on Github `https://github.com/martinsipka/rarediffsim`

The Google Colab notebook with the Brachistochrone example is available at: `https://colab.research.google.com/drive/1YjIMTFQA0L9oLMkNpV7E2jOxbbzO6PnM?usp=sharing`