Robotics: Science and Systems 2026
Sydney, Australia, July 13–17, 2026

Tempered Sequential Monte Carlo for Trajectory and Policy Optimization with Differentiable Dynamics

Heng Yang
Harvard University
https://github.com/ComputationalRobotics/TSMC

***Abstract*—We propose a sampling-based framework for finite-horizon trajectory and policy optimization under differentiable dynamics by casting controller design as inference. Specifically, we minimize a KL-regularized expected trajectory cost, which yields an optimal "Boltzmann-tilted" distribution over controller parameters that concentrates on low-cost solutions as temperature decreases. To sample efficiently from this sharp, potentially multimodal target, we introduce *tempered sequential Monte Carlo* (TSMC): an annealing scheme that adaptively reweights and resamples particles along a tempering path from a prior to the target distribution, while using Hamiltonian Monte Carlo rejuvenation to maintain diversity and exploit *exact gradients* obtained by differentiating through trajectory rollouts. For policy optimization, we extend TSMC via (i) a deterministic empirical approximation of the initial-state distribution and (ii) an extended-space construction that treats rollout randomness as auxiliary variables. Experiments across trajectory- and policy-optimization benchmarks show that TSMC is broadly applicable and compares favorably to state-of-the-art baselines.**

I. Introduction

Consider a discrete-time dynamical system

$$x_{t+1} = f(x_t, u_t), \quad t \in \mathbb{N} \tag{1}$$

where $x_t \in \mathbb{R}^n$ is the state and $u_t \in \mathbb{R}^m$ is the control input. We focus on deterministic dynamics for clarity of exposition and discuss extensions to stochastic dynamics in Remark 3. For a finite horizon T, we denote a state-control trajectory by $\tau = (x_0, u_0, x_1, u_1, \ldots, x_{T-1}, u_{T-1}, x_T)$, with total cost

$$J(\tau) = \sum_{t=0}^{T-1} \ell_t(x_t, u_t) + \ell_T(x_T). \tag{2}$$

Here ℓ_t for $t = 0, \ldots, T-1$ are stage costs and ℓ_T is a terminal cost. Let μ denote a distribution over initial states x_0, and let $\theta \in \mathbb{R}^d$ parameterize a (state-feedback) controller $\pi_\theta : \mathbb{R}^n \to \mathbb{R}^m$. Under π_θ, the control input is $u_t = \pi_\theta(x_t)$, which together with the dynamics induces a trajectory rollout $\tau(x_0, \theta)$. For simplicity, we focus on deterministic controllers. We then consider the unified objective

$$\min_{\theta \in \mathbb{R}^d} \mathbb{E}_{x_0 \sim \mu} \left[J(\tau(x_0, \theta)) \right]. \tag{3}$$

Problem (3), henceforth referred to as *trajectory and policy optimization* (TPO), subsumes two fundamental settings in optimal control and reinforcement learning:

- **Trajectory Optimization (TO)** [60, 13, 18, 41, 57]: fix the initial state by setting $\mu = \delta_{x_0}$ (a Dirac measure), and choose an open-loop parameterization $\theta = (u_0, \ldots, u_{T-1})$ (equivalently, $\pi_\theta(x_t) \equiv u_t$). Then (3) reduces to optimizing a single control sequence for a given x_0.
- **Policy Optimization (PO)** [68, 63, 64, 39, 28]: let μ represent a distribution over initial conditions (*e.g.,* uniform over a region in the state space), and parameterize π_θ as a state-feedback policy (*e.g.,* with a neural network). Then (3) becomes the problem of minimizing expected trajectory cost over the initial-state distribution.

Motivated by the recent efforts in *differentiable* physics simulation—particularly contact solvers [45, 34, 65, 24, 56, 55, 33]—and in learning *differentiable* world models [58, 48, 82, 3, 5, 2, 80], we assume the dynamics $f(x_t, u_t)$ is differentiable in both x_t and u_t. This enables gradient-based approaches that "exploit" differentiability: in TO, nonlinear programming techniques (single or multiple shooting) optimize an open-loop control sequence [60, 7, 74]; in PO, recent "first-order" methods [17, 81, 5, 6, 77] leverage differentiability to compute policy gradients more efficiently and improve sample efficiency. A drawback, however, is that gradient-based optimization can get trapped in undesirable local minima, and gradients from differentiable simulators can be uninformative in the presence of nonsmooth dynamics [67]. In contrast, sampling-based methods such as *model predictive path integral control* (MPPI) [78, 79] "explore" the control space and use the dynamics model only to evaluate trajectory costs; they can be less sensitive to poor local minima but often require many samples. This motivates our central question: *can we design a unified and principled algorithm for TPO that combines the benefits of sampling and gradient-based optimization under differentiable dynamics?*

From Optimization to Inference. Towards this, instead of optimizing a single parameter θ in (3), we consider optimizing a distribution p over $\theta \in \mathbb{R}^d$. Let p_0 be a prior distribution with *full support* on $\mathbb{R}^d$ (*i.e.,* with density $p_0(\theta) > 0$ for all $\theta \in \mathbb{R}^d$; *e.g.,* a Gaussian). We study the KL-regularized problem

$$\min_{p} \mathbb{E}_{x_0 \sim \mu, \theta \sim p} \left[J(\tau(x_0, \theta)) \right] + \lambda D_{\mathrm{KL}}(p \| p_0), \tag{4}$$

where $D_{\mathrm{KL}}(p\|p_0) := \int_{\mathbb{R}^d} p(\theta) \log \frac{p(\theta)}{p_0(\theta)} d\theta$ is the Kullback–Leibler (KL) divergence and $\lambda > 0$ controls the strength of regularization (equivalently, a "temperature"). At $\lambda = 0$, problem (4) is equivalent to the TPO problem (3): they have the same optimal value and any distribution supported on the optimal solution set of (3) is optimal for (4). As $\lambda \to 0^+$, the optimal $p^\star$ concentrates its mass on minimizers of (3). A key advantage of (4) is that it admits a closed-form solution.

Theorem 1 (Optimal Distribution). *Assume $\ell_t(\cdot,\cdot) \geq 0$ for $t = 0, \ldots, T-1$ and $\ell_T(\cdot) \geq 0$. Define the energy function*

$$E(\theta) = \mathbb{E}_{x_0 \sim \mu}\left[J(\tau(x_0, \theta))\right], \tag{5}$$

then the optimal distribution $p^\star$ of (4) is given by

$$p^\star(\theta) = \frac{1}{Z} p_0(\theta) \exp\left(-\frac{1}{\lambda} E(\theta)\right), \tag{6}$$

where

$$Z = \int_{\mathbb{R}^d} p_0(\theta) \exp\left(-\frac{1}{\lambda} E(\theta)\right) d\theta \tag{7}$$

is the normalizing constant (partition function).

A proof is provided in the Supplementary Material. Theorem 1 reframes optimization as inference: the optimal solution of (4) is obtained by *Boltzmann-tilting* the prior p_0 with the energy $E(\theta)$, namely $p^\star(\theta) \propto p_0(\theta)\exp(-E(\theta)/\lambda)$ (a.k.a. the Gibbs posterior [51, 38]). Intuitively, this tilt exponentially *upweights* controller parameters θ that incur lower energy, *i.e.,* lower expected trajectory cost. Thus, if we can efficiently sample from the corresponding (unnormalized) density—especially at low temperature λ—then samples concentrate near minimizers of the original TPO objective (3). This "control-as-inference" characterization is not new: closely related forms appear in information-theoretic optimal control [71, 42, 79, 78] and reinforcement learning [63, 28, 47, 50].

While Theorem 1 gives a closed-form solution to the KL-regularized TPO problem, sampling from the Boltzmann-tilted distribution (6) remains challenging. At low temperature, the target distribution can be sharply concentrated and highly multimodal, reflecting the nonconvexity of the underlying TPO objective (3). To the best of our knowledge, sampling from such hard targets has received limited attention in the optimal control and reinforcement learning literature. Existing approaches are typically based on importance sampling (*e.g.,* MPPI and the cross-entropy method [20]) or on Markov chain Monte Carlo (MCMC) (*e.g.,* Bayesian motion planning [83, 43] and energy-based RL policies [27, 62]). For sharp, multimodal targets, the former often requires many samples, while the latter can suffer from long mixing times.

Contribution. We develop an algorithm for sampling from the Boltzmann-tilted distribution (6) that combines sampling-based exploration with gradient-based optimization, and applies to both TO and PO. Our approach builds on *tempered sequential Monte Carlo* (TSMC), a state-of-the-art tool for multimodal, low-temperature targets in Bayesian inference [53, 22, 21, 19]. TSMC introduces a *tempering path* that gradually deforms an easy-to-sample prior into the desired Boltzmann tilt (*cf.* Fig. 1); at each tempering level, it reweights and resamples particles to focus computation on low-energy regions, then applies Hamiltonian Monte Carlo (HMC) rejuvenation to restore diversity and enable local exploration while preserving the tempered target. We show that TSMC can be applied directly to TO, where $\nabla_\theta E(\theta)$ is available via automatic differentiation through rollouts. For PO, we introduce practical variants based on (i) a deterministic approximation of $E(\theta)$ and (ii) an extended-space construction that augments each particle with a batch of initial conditions. Experiments across TO and PO benchmarks demonstrate robust performance and improved results relative to existing methods.

Outline. Section II provides a tutorial-style introduction to TSMC. Sections III and IV apply TSMC to trajectory optimization and policy optimization, respectively, with experimental results in Sections V and VI. Section VII concludes with limitations and future directions. Supplementary Material contains proofs, additional results, and related work.

II. Tempered Sequential Monte Carlo

Recall that our goal is to sample from the Boltzmann-tilted distribution (6), where the energy function $E(\theta)$ is differentiable in θ. This section introduces the key ingredients of tempered sequential Monte Carlo (TSMC). For concreteness, we use the following running example.

Example 1 (Shekel-tilted Distribution). *Let $\theta \in \mathbb{R}^2$ and consider the energy function defined by the negative Shekel function [66] with three centers:*

$$E(\theta) = -\sum_{i=1}^{3} \frac{1}{\|\theta - c_i\|^2 + s_i}, \tag{8}$$

where $c_1 = (2,2)$, $c_2 = (-2,2)$, and $c_3 = (0,-2.5)$ are the centers, and $s_1 = s_2 = 0.5$, $s_3 = 1.2$ are the widths. This nonconvex landscape has three local minima at the three centers, with c_1 and c_2 being global minima. Let p_0 be the standard Gaussian, and define the Shekel-tilted distribution by

$$p(\theta) \propto \exp\left(-\left(\frac{1}{2}\|\theta\|^2 + \frac{1}{\lambda} E(\theta)\right)\right). \tag{9}$$

The lowest surface in Fig. 1(a) plots the negative log-density $\frac{1}{2}\|\theta\|^2 + \frac{1}{\lambda}E(\theta)$ with $\lambda = 0.1$, revealing multimodality of $p(\theta)$.

A. Tempered Importance Sampling

To address multimodality of the Boltzmann-tilted distribution (6), TSMC introduces a *tempering path* that gradually deforms the prior p_0 into the target $p^\star$. Concretely, choose an increasing sequence of scalars $0 = \beta_0 < \beta_1 < \cdots < \beta_K = 1$ and define intermediate distributions

$$p_k(\theta) \propto p_0(\theta) \exp\left(-\frac{\beta_k}{\lambda} E(\theta)\right), \quad k = 1, \ldots, K. \tag{10}$$

Clearly, p_0 coincides with the prior, and p_K coincides with the target $p^\star$. Fig. 1(a) plots the negative log-density of $p_k(\theta)$ for the Shekel-tilted distribution at different values of β. As β increases, the nonconvex Shekel term is weighted more heavily relative to the convex quadratic, smoothly deforming the landscape from convex to strongly nonconvex. To tractably sample from p_k, we maintain a weighted particle approximation $\{(\theta_k(i), w_k(i))\}_{i=1}^N$ with normalized weights $\sum_{i=1}^N w_k(i) = 1$. Here k indexes the tempering step and i indexes the i-th particle (and its weight).

Importance Sampling. Suppose we have a weighted particle approximation $\{(\theta_{k-1}(i), w_{k-1}(i))\}_{i=1}^N$ for p_{k-1}. To

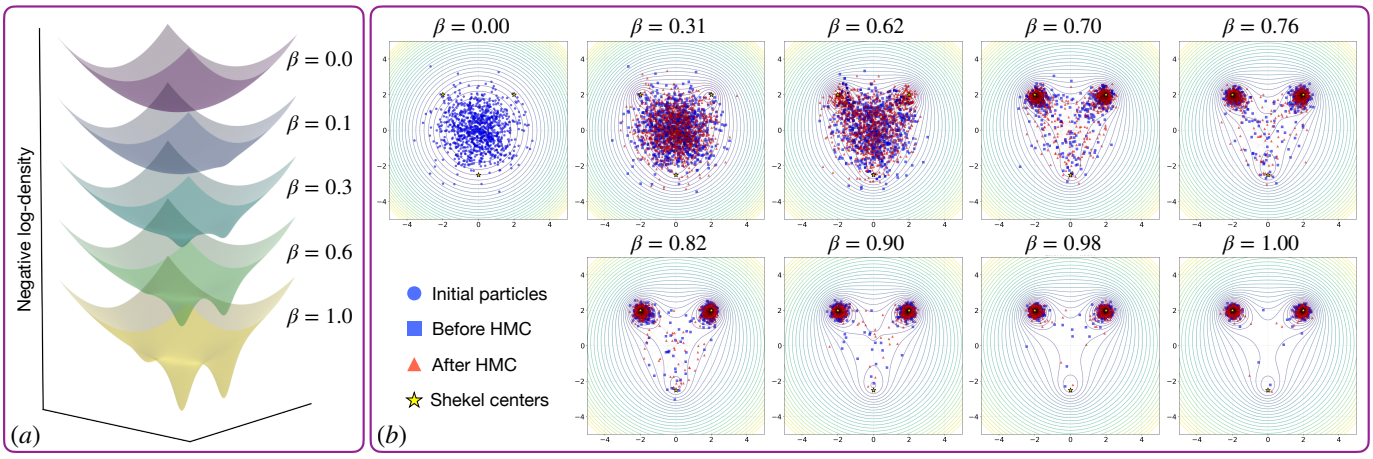

Fig. 1. Tempered Sequential Monte Carlo samples from the Shekel-tilted distribution in Example 1. (a) Negative log-density of the Shekel-tilted distribution at different tempering values. (b) Particle locations before and after HMC rejuvenation on contours of the Shekel-tilted negative log-density.

update it to approximate p_k, we reweight each particle by the *incremental likelihood ratio*

$$\Delta w_k(i) = \exp\Big(- \frac{\beta_k - \beta_{k-1}}{\lambda} E(\theta_{k-1}(i)) \Big) \tag{11}$$

and then renormalize,

$$w_k(i) = \frac{w_{k-1}(i) \cdot \Delta w_k(i)}{\sum_{j=1}^N w_{k-1}(j) \cdot \Delta w_k(j)}, \qquad i = 1, \dots, N. \tag{12}$$

The factor $\Delta w_k(i)$ is (up to a constant) the ratio $p_k(\theta(i))/p_{k-1}(\theta(i))$, so particles with lower energy $E(\theta)$ are exponentially favored (upweighted) as β increases.

Adaptive Tempering. As β_k increases, weight disparity grows and the particle approximation may degenerate, commonly tracked by the effective sample size (ESS),

$$\mathrm{ESS}_k \triangleq \frac{1}{\sum_{i=1}^N (w_k(i))^2} \in [1, N], \tag{13}$$

where the bounds correspond to complete degeneracy (one particle has all weight) and uniform weights ($w_k(i) = 1/N, i = 1, \dots, N$), respectively. In practice, we use ESS to choose the next temperature β_k adaptively: given particles from p_{k-1}, we pick $\beta_k \in (\beta_{k-1}, 1]$ (*e.g.,* using bisection) so that the ESS *after* reweighting by (12) hits a target level, such as $\mathrm{ESS}_k \approx \rho N$ for a fixed $\rho \in (0, 1)$. This yields an adaptive tempering schedule that prevents abrupt weight collapse. For the Shekel-tilted distribution, Fig. 1(b) shows the resulting adaptive schedule $\{\beta_k\}_{k=0}^K$ obtained by targeting $\mathrm{ESS}_k \approx \rho N$ with $\rho = 0.9$.

Resampling. After selecting β_k and computing the updated weights via (12), the particle approximation of p_k is *weighted*. Resampling converts this weighted set into an *approximately unweighted* one, which (i) prevents gradual accumulation of weight disparity across steps and (ii) focuses computation on particles that have non-negligible mass under p_k. Concretely, we draw ancestor indices $a_1, \dots, a_N$ i.i.d. from the categorical distribution with probabilities $(w_k(1), \dots, w_k(N))$, set $\theta_k(i) \leftarrow \theta_{k-1}(a_i)$, and reset weights to $w_k(i) = 1/N$.

Repeated reweighting and resampling can reduce particle diversity: the population may collapse onto a small number of distinct particles that carry most of the mass. Moreover, without additional moves, the particle locations remain restricted to the initial sample $\{\theta_0(i)\}_{i=1}^N$ and cannot explore new parameter values. Next, we incorporate MCMC transitions to rejuvenate the particle set and enable local exploration while preserving the tempered target distribution.

B. MCMC Rejuvenation

At each tempering level k and after resampling, we apply a Markov transition kernel $\mathcal{T}_k(\theta' \mid \theta)$ to each particle. The key requirement is that $\mathcal{T}_k$ leaves p_k *invariant*, meaning that if $\theta \sim p_k$ and $\theta' \sim \mathcal{T}_k(\cdot \mid \theta)$, then $\theta' \sim p_k$. Equivalently,

$$\int p_k(\theta)\, \mathcal{T}_k(\theta' \mid \theta)\, d\theta \;=\; p_k(\theta'). \tag{14}$$

Thus, rejuvenation can be viewed as a "move" step that improves diversity without biasing the particle approximation. For illustration, Fig. 1(b) overlays particle locations before and after MCMC rejuvenation on contours of the Shekel-tilted negative log-density. The rejuvenation step proposes distant moves, enabling exploration of the parameter space.

Hamiltonian Monte Carlo. There are many MCMC kernels that can be used for rejuvenation. Here we focus on Hamiltonian Monte Carlo (HMC) [23, 54, 11], which is a popular choice for its ability to propose distant moves (and hence improve particle diversity). For each tempering level k, define the (unnormalized) negative log-density (potential)

$$V_k(\theta) \;\triangleq\; \frac{\beta_k}{\lambda} E(\theta) \;-\; \log p_0(\theta), \tag{15}$$

so that $p_k(\theta) \propto \exp(-V_k(\theta))$ (*cf.* the definition in (10)). HMC augments θ with an auxiliary momentum $r \in \mathbb{R}^d$ drawn from a Gaussian $r \sim \mathcal{N}(0, M)$, where $M \succ 0$ is a mass matrix, and defines the Hamiltonian $H_k(\theta, r) \triangleq V_k(\theta) + 0.5 r^\top M^{-1} r$. HMC proposes distant moves by approximately simulating Hamiltonian dynamics under H_k using a symplectic leapfrog integrator [29]. In particular, given step size $\varepsilon > 0$ and number of steps L, a single HMC transition proceeds as follows:

- *Refresh momentum:* sample $r_0 \sim \mathcal{N}(0, M)$; set $\theta_0 = \theta$.

- *Leapfrog integration:* for $s = 0, \dots, L-1$,

$$\begin{aligned} r_{s+\frac{1}{2}} &= r_s - \frac{\varepsilon}{2}\nabla_\theta V_k(\theta_s), \\ \theta_{s+1} &= \theta_s + \varepsilon\, M^{-1} r_{s+\frac{1}{2}}, \\ r_{s+1} &= r_{s+\frac{1}{2}} - \frac{\varepsilon}{2}\nabla_\theta V_k(\theta_{s+1}). \end{aligned} \tag{16}$$

- *Metropolis correction:* Define $(\theta', r') = (\theta_L, -r_L)$ and accept the proposal with probability

$$\min\left\{1, \exp(-H_k(\theta', r') + H_k(\theta_0, r_0))\right\}. \tag{17}$$

Otherwise set $\theta' = \theta_0$. Return the new position θ'.

Intuitively, invariance follows because leapfrog defines a reversible, volume-preserving proposal map for the Hamiltonian dynamics [54], and the Metropolis–Hastings accept/reject step corrects the discretization error to enforce detailed balance with respect to the target density p_k [61]. Detailed balance implies that $\mathcal{T}_k$ leaves p_k invariant, *i.e.,* (14) holds.

Parameter Tuning. The HMC kernel has three key parameters: the mass matrix M, the step size ε, and the number of leapfrog steps L. Ideally, M matches the local scaling/-correlation structure of the target (so different coordinates evolve on comparable time scales), ε is as large as possible while maintaining a reasonable acceptance probability, and the trajectory length $L\varepsilon$ is long enough to make a substantive move without wasting computation by retracing the trajectory.

In our implementation, we set $M = I$ since estimating a well-conditioned, geometry-adapted mass matrix requires additional tuning/curvature information [31, 11, 25, 76]. We manually select ε, and use the NUTS sampler [31] to adaptively determine the effective number of integration steps L on the fly. Intuitively, this avoids hand-tuning L: it extends the Hamiltonian trajectory until further integration would start to reverse direction in parameter space, yielding long proposals when beneficial while preventing redundant trajectories.

Summary. TSMC tackles the multi-modal, low-temperature target (6) by introducing a tempering path that gradually deforms the easy-to-sample prior into the desired Boltzmann tilt. At each tempering level, we (i) reweight particles by importance sampling (using ESS to adaptively choose the next temperature), (ii) resample to control weight degeneracy, and (iii) apply HMC rejuvenation to restore diversity and explore the parameter space while preserving the tempered target distribution. Fig. 1(b) shows that TSMC successfully samples from the Shekel-tilted distribution: the final particles are well-distributed across the two global minima, with a small number of particles exploring the third local minimum. In Supplementary Material, we provide the asymptotic theoretical guarantees offered by TSMC. Next, we apply TSMC to trajectory optimization (TO) and policy optimization (PO).

III. TSMC for Trajectory Optimization

Recall from (5) the energy $E(\theta) = \mathbb{E}_{x_0\sim\mu}[J(\tau(x_0,\theta))]$. For trajectory optimization (TO), $\mu = \delta_{x_0}$, so $E(\theta) = J(\tau(x_0,\theta))$ and $\nabla_\theta E(\theta)$ can be computed exactly by differentiating through the trajectory rollout, using the adjoint method.

Theorem 2 (Exact Gradients for TO)**.** *Assume the dynamics $f(x,u)$ is differentiable in (x,u), the controller $\pi_\theta(x)$ is differentiable in (x,θ), the stage cost functions $\{\ell_t\}_{t=0}^{T-1}$ are differentiable in (x,u), and the terminal cost ℓ_T is differentiable in x. Starting from an initial state x_0, let $\tau = (x_0, u_0, \dots, x_T)$ be a trajectory rollout generated by the controller $\pi_\theta(x)$. Define*

$$\begin{aligned} A_t &\triangleq \frac{\partial f}{\partial x}(x_t,u_t) \in \mathbb{R}^{n\times n}, \quad B_t \triangleq \frac{\partial f}{\partial u}(x_t,u_t) \in \mathbb{R}^{n\times m}, \\ L_t &\triangleq \frac{\partial \pi_\theta}{\partial x}(x_t) \in \mathbb{R}^{m\times n}, \quad G_t \triangleq \frac{\partial \pi_\theta}{\partial \theta}(x_t) \in \mathbb{R}^{m\times d}, \\ \ell_{x,t} &\triangleq \frac{\partial \ell_t}{\partial x}(x_t,u_t) \in \mathbb{R}^{n}, \quad \ell_{u,t} \triangleq \frac{\partial \ell_t}{\partial u}(x_t,u_t) \in \mathbb{R}^{m}, \\ \ell_{x,T} &\triangleq \frac{\partial \ell_T}{\partial x}(x_T) \in \mathbb{R}^{n} \end{aligned} \tag{18}$$

and the costate (adjoint) recursion backward in time:

$$\begin{aligned} \varphi_T &= \ell_{x,T}, \quad \text{and for } t = T-1, \dots, 0: \\ g_t &= \ell_{u,t} + B_t^\top \varphi_{t+1}, \quad \varphi_t = \ell_{x,t} + L_t^\top g_t + A_t^\top \varphi_{t+1}. \end{aligned} \tag{19}$$

Then the gradient of the trajectory cost with respect to θ is

$$\nabla_\theta J(\tau(x_0,\theta)) = \sum_{t=0}^{T-1} G_t^\top g_t. \tag{20}$$

The result follows by repeated application of the chain rule (equivalently, reverse-mode automatic differentiation) and we omit the proof for brevity. In practice, when f, π_θ, $\{\ell_t\}_{t=0}^{T-1}$, and ℓ_T are implemented in modern autodiff frameworks (*e.g.,* PyTorch or JAX), the adjoint recursion is computed efficiently by backpropagation through the differentiable rollout.

In the TO setting with open-loop parameterization $\theta = (u_0, \dots, u_{T-1}) \in \mathbb{R}^{Tm}$ and $\pi_\theta(x_t) = u_t$ (no state feedback), $\frac{\partial \pi_\theta}{\partial x}(x_t) = 0$, hence $L_t = 0$. Moreover, $\frac{\partial \pi_\theta}{\partial \theta}(x_t)$ simply selects the t-th control block, *i.e.,* $G_t = [0, \dots, I_m, \dots, 0] \in \mathbb{R}^{m\times Tm}$, where the only nonzero block is the t-th $m \times m$ identity.

IV. TSMC for Policy Optimization

For policy optimization (PO), the energy $E(\theta)$ is an expectation over random rollouts (through $x_0 \sim \mu$), and is generally intractable to evaluate exactly under nonlinear dynamics. In practice, both $E(\theta)$ and $\nabla_\theta E(\theta)$ are approximated using Monte Carlo rollouts, yielding stochastic estimates. This raises a natural question: when only stochastic estimates of $E(\theta)$ and $\nabla_\theta E(\theta)$ are available, can we still use TSMC to sample from the optimal Boltzmann-tilted distribution $p^\star$ in (6)?

A first idea is to replace the deterministic HMC rejuvenation kernel from Section II with a stochastic-gradient variant (*e.g.,* stochastic gradient HMC [15]). While this can address the *move* (MCMC) step, it does not resolve the *weighting* step: the TSMC incremental weights (11) depend on factors of the form $\exp(-c\,E(\theta))$ (and ratios thereof) for a constant $c > 0$, and an unbiased estimator $\widehat{E}(\theta)$ does *not* in general yield an unbiased estimator of $\exp(-c\,E(\theta))$. A principled route would be to construct a *nonnegative unbiased estimator* $\widehat{W}(\theta) \geq 0$ such that $\mathbb{E}[\widehat{W}(\theta)] = \exp(-c\,E(\theta))$, and then plug $\widehat{W}(\theta)$ into the incremental weights. Unfortunately, nonnegative unbiased estimation of nonlinear functionals such as $\phi(z) = \exp(-z)$ is

highly nontrivial and, in general, impossible without additional structure; see [35]. Even when such estimators exist, they typically rely on specialized "factory" constructions and can require elaborate randomized procedures; see, *e.g.,* [44].

Due to these challenges, we use two practical alternatives: (i) a deterministic approximation of $E(\theta)$ and (ii) an extended-space TSMC variant.

A. Deterministic Approximation

A simple way to recover a deterministic energy is to approximate the initial-state distribution μ by an empirical measure supported on a *fixed* set of B initial conditions $\{x_0(b)\}_{b=1}^B$:

$$\mu_B \;=\; \frac{1}{B}\sum_{b=1}^{B}\delta_{x_0(b)}. \tag{21}$$

This yields the surrogate energy

$$E_B(\theta) := \mathbb{E}_{x_0\sim\mu_B}[J(\tau(x_0,\theta))] = \frac{1}{B}\sum_{b=1}^{B} J(\tau(x_0(b),\theta)) \tag{22}$$

and the corresponding Boltzmann-tilted target $p_B^\star(\theta) \propto p_0(\theta)\exp(-E_B(\theta)/\lambda)$ (and similarly for the intermediate tempered targets along the TSMC path). Crucially, once $\{x_0(b)\}_{b=1}^B$ is fixed, both $E_B(\theta)$ and $\nabla_\theta E_B(\theta)$ are deterministic and can be computed by differentiating through each rollout using Theorem 2 and averaging. Therefore, the TSMC algorithm from Section II does not need to be modified: incremental weights are evaluated using $E_B(\theta)$, and the HMC rejuvenation kernel uses $\nabla_\theta E_B(\theta)$ as usual.

B. Extended-space TSMC

Pseudo-marginal methods address intractable expectations by augmenting the state space with auxiliary random variables, and then sampling the desired marginal by running a standard sampler on the extended space [9, 8, 16, 4]. Here we adopt this idea for PO by treating the rollout randomness (initial conditions) as part of the particle state.

Extended-space Tempered Targets. Let each particle carry controller parameters $\theta \in \mathbb{R}^d$ and a batch of B i.i.d. initial conditions $x_0(1),\dots,x_0(B) \sim \mu$. We shorthand $\mathbf{x}_0 \triangleq (x_0(1),\dots,x_0(B))$. Define the empirical average rollout cost

$$\bar{J}_B(\theta,\mathbf{x}_0) \;\triangleq\; \frac{1}{B}\sum_{b=1}^{B} J(\tau(x_0(b),\theta)). \tag{23}$$

We then define a sequence of tempered distributions, for $k = 0,\dots,K$, on the extended space $(\theta,\mathbf{x}_0)\in\mathbb{R}^d\times(\mathbb{R}^n)^B$:

$$\tilde{p}_k(\theta,\mathbf{x}_0) \propto p_0(\theta)\prod_{b=1}^{B}\mu(x_0(b))\exp\!\left(-\frac{\beta_k}{\lambda}\,\bar{J}_B(\theta,\mathbf{x}_0)\right). \tag{24}$$

Running TSMC on $\{\tilde{p}_k\}$ yields samples $(\theta,\mathbf{x}_0)\sim\tilde{p}_K$; discarding $\mathbf{x}_0$ returns θ distributed according to the marginal

$$\tilde{p}_K(\theta) \propto p_0(\theta)\,\mathbb{E}_{\mathbf{x}_0\sim\mu^{\otimes B}}\left[\exp\!\left(-\frac{1}{\lambda}\,\bar{J}_B(\theta,\mathbf{x}_0)\right)\right]. \tag{25}$$

This marginal differs from the original Boltzmann tilt $p^\star(\theta)\propto p_0(\theta)\exp(-E(\theta)/\lambda)$: it is a *risk-sensitive* (exponential utility) objective [36], reflecting that in general $\mathbb{E}[\exp(-Z)] \neq \exp(-\mathbb{E}[Z])$. Indeed, write

$$\delta_B(\theta) := \log \mathbb{E}_{\mathbf{x}_0\sim\mu^{\otimes B}}\left[\exp\left(-\frac{1}{\lambda}\bar{J}_B(\theta,\mathbf{x}_0)\right)\right] + \frac{1}{\lambda}E(\theta), \tag{26}$$

we have, from (25),

$$\tilde{p}_K(\theta) \propto p_0(\theta)\exp\left(-\frac{1}{\lambda}E(\theta)\right)\exp(\delta_B(\theta)). \tag{27}$$

Thus the extended-space marginal is a multiplicative exponential-utility perturbation of the true Boltzmann tilt. Moreover, if $J(\tau(x_0,\theta))$ is sub-Gaussian under $x_0\sim\mu$ with variance proxy $\sigma^2(\theta)$ [75], then

$$0 \le \delta_B(\theta) \le \frac{\sigma^2(\theta)}{2B\lambda^2}. \tag{28}$$

Consequently, the mismatch decreases as $O(1/B)$, and is small when the rollout-cost variance across initial states is modest, the batch size B is large, and the temperature λ is not too small.

TSMC in the Extended Space. We maintain weighted particles $\{(\theta_k(i),\mathbf{x}_{0,k}(i),w_k(i))\}_{i=1}^N$ approximating $\tilde{p}_k$ as in (24), where $\mathbf{x}_{0,k}(i)\triangleq(x_{0,k}(i,1),\dots,x_{0,k}(i,B))$. The TSMC steps mirror those in Section II:

- *Initialize:* sample $\theta_0(i)\sim p_0$ and $x_{0,0}(i,b)\sim\mu$ i.i.d. for $b=1,\dots,B$; set $w_0(i)=1/N$.
- *Importance weight update:* given particles for $\tilde{p}_{k-1}$, update weights using the incremental ratio

$$\Delta w_k(i) = \exp\!\Big(-\frac{\beta_k-\beta_{k-1}}{\lambda}\,\bar{J}_B(\theta_{k-1}(i),\mathbf{x}_{0,k-1}(i))\Big), \tag{29}$$

  and normalize as in (12).
- *Resample:* resample ancestor indices according to $w_k(i)$ and reset weights to $1/N$, as in Section II.
- *HMC rejuvenation (move θ, hold $\mathbf{x}_0$ fixed):* for each particle, apply an HMC move targeting the conditional density $\tilde{p}_k(\theta\mid\mathbf{x}_0)\propto p_0(\theta)\exp(-\frac{\beta_k}{\lambda}\bar{J}_B(\theta,\mathbf{x}_0))$. Define the unnormalized negative log-density (potential)

$$\tilde{V}_k(\theta) \;\triangleq\; \frac{\beta_k}{\lambda}\bar{J}_B(\theta,\mathbf{x}_0) \;-\; \log p_0(\theta), \tag{30}$$

  and Hamiltonian $\tilde{H}_k(\theta,r)\triangleq\tilde{V}_k(\theta)+0.5r^\top M^{-1}r$, where $r\sim\mathcal{N}(0,M)$. The gradient of the potential is

$$\nabla_\theta\tilde{V}_k(\theta) = \frac{\beta_k}{\lambda B}\sum_{b=1}^{B}\nabla_\theta J(\tau(x_0(b),\theta)) - \nabla_\theta\log p_0(\theta), \tag{31}$$

  where each $\nabla_\theta J(\tau(x_0(b),\theta))$ is obtained by Theorem 2. One HMC move then proceeds by (i) sampling $r_0\sim\mathcal{N}(0,M)$, (ii) applying the leapfrog integrator (16) with V_k replaced by $\tilde{V}_k$ to obtain (θ_L,r_L), and (iii) accepting $\theta'=\theta_L$ with probability (17) where H_k is replaced by $\tilde{H}_k$ or otherwise keeping $\theta'=\theta_0$.
- *Refresh initial states (move $\mathbf{x}_0$, hold θ fixed):* to avoid restricting $\mathbf{x}_0$ to a finite set of initial samples, we update

the auxiliary batch using a Metropolis–Hastings refresh that targets $\tilde{p}_k(\mathbf{x}_0 \mid \theta)$. For each $b = 1, \ldots, B$, propose $x_0'(b) \sim \mu(\cdot)$ and accept with probability

$$\min\left\{1, \exp\left(-\frac{J(\tau(x_0'(b), \theta)) - J(\tau(x_0(b), \theta))}{\lambda B/\beta_k}\right)\right\}. \quad (32)$$

Otherwise, keep $x_0(b)$.

The rejuvenation kernel at level k is defined as the composition of (i) an HMC move on θ targeting $\tilde{p}_k(\theta \mid \mathbf{x}_0)$ with $\mathbf{x}_0$ held fixed, and (ii) an MH refresh of $\mathbf{x}_0$ targeting $\tilde{p}_k(\mathbf{x}_0 \mid \theta)$ with θ held fixed. Each sub-step leaves its corresponding conditional invariant, hence their composition leaves the joint extended target $\tilde{p}_k(\theta, \mathbf{x}_0)$ invariant.

Remark 3 (Extension to Stochastic Dynamics)**.** *A natural extension of our framework is to stochastic dynamics of the form $x_{t+1} = f(x_t, u_t) + w_t$, where w_t is a random disturbance. In that case, the energy becomes $E(\theta) = \mathbb{E}_{x_0, w_{0:T-1}}[J(\tau(x_0, \theta; w_{0:T-1}))]$. Thus, at the level of the target distribution, TSMC carries over directly by redefining the energy to average over both initial-state and disturbance randomness. Moreover, the extended-space construction also generalizes naturally by augmenting the auxiliary variables with sampled disturbance trajectories. The resulting θ-marginal is again a risk-sensitive surrogate of the exact Boltzmann target.*

V. Experiments: Trajectory Optimization

We evaluate TSMC for trajectory optimization against baseline methods. All experiments run on a Lambda workstation with an AMD Ryzen Threadripper PRO 5975WX (32 cores) and two NVIDIA Ada6000 GPUs.

Implementation. We implement the dynamical systems and TSMC in JAX [12] to leverage automatic differentiation and GPU acceleration. We use BlackJax [14] to implement TSMC; specifically, its adaptive TSMC framework with NUTS as the HMC rejuvenation kernel. We set the ESS threshold for selecting the next tempering parameter to $\rho = 0.8$.

Baselines. We compare TSMC with:

- Parallel-NUTS. TSMC starts from particles sampled from p_0. Parallel-NUTS uses the same initialization, but for each particle runs an independent MCMC using NUTS in BlackJax, targeting the optimal distribution $p^\star$ directly.
- Parallel-MALA. Same as Parallel-NUTS, but using the Metropolis-adjusted Langevin algorithm (MALA) [10] in BlackJax. Since MALA is cheaper per iteration than NUTS, we run MALA for $100\times$ more steps.
- Parallel-MPPI. Starting from the same initial particles as TSMC, we run 64 MPPI planning steps per particle. We treat each particle as the nominal control, sample 64 noise perturbations, evaluate their trajectory costs, and update by an $\exp(-\frac{1}{\lambda}\text{cost})$-weighted average.
- Parallel-IPOPT. Starting from the same initial particles, we run IPOPT using CasADi [7] with single shooting, initialized at each particle. We keep the CasADi dynamics consistent with the JAX implementation so the trajectory gradients with respect to controls match. Since IPOPT can be slow, we run it in parallel across CPU cores (25 workers). We set the maximum number of iterations to 200 and the stopping tolerance to 10^{-6}.

Comparing TSMC with Parallel-NUTS and Parallel-MALA isolates the role of tempering for sampling from sharp, multimodal targets. Comparing with Parallel-MPPI highlights the benefit of combining importance sampling with gradient-based optimization. Finally, Parallel-IPOPT serves as a simple hybrid baseline: gradient-based optimization from multiple random initializations. Across all methods, we use $N = 100$ particles.

Prior. We use a first-order autoregressive prior [30] over the open-loop control sequence to encourage smoothness: $u_t = \gamma u_{t-1} + \epsilon_t$, where $\epsilon_t \sim \mathcal{N}(0, \sigma^2 I)$, with $\gamma = 0.9$ and $\sigma = 0.3$.

A. Inverted Pendulum

Setup. The goal is to swing up the pendulum to the upright configuration under control limits. We discretize the dynamics using a variational integrator [46, 41, 69]. Thanks to its favorable energy behavior, this discretization permits a relatively large time step and a shorter horizon. We follow [41, Section E.1]; in particular, we use a time step $\Delta t = 0.1$ seconds and planning horizon $T = 30$. Since the control is one-dimensional, this yields $\theta \in \mathbb{R}^{30}$.

Certified Lower Bound. We use the semidefinite programming (SDP) relaxations in [41, 40] to compute a certified lower bound on the optimal cost, providing a reference for assessing whether TSMC (and baselines) reach global optimality.

Results. Fig. 2(a) shows boxplots of the trajectory costs across the final particles for each method, along with the certified lower bound. We make three observations. (i) TSMC, Parallel-NUTS, and Parallel-IPOPT achieve similarly strong performance: their final particles attain costs close to the SDP lower bound, suggesting near-global solutions. (ii) Parallel-MALA performs reasonably, but exhibits a heavier tail: while the best cost is near the lower bound, some particles remain far from optimality. (iii) Parallel-MPPI outperforms Parallel-MALA but underperforms TSMC, Parallel-NUTS, and Parallel-IPOPT, highlighting the benefit of combining importance sampling with gradient-based optimization under differentiable dynamics.

B. Acrobot

Setup. The goal is to swing up the acrobot (two-link pendulum) to the upright configuration using a single control at the elbow joint (Fig. 2(d)). We discretize the continuous-time dynamics using fourth-order Runge–Kutta (RK4) with time step $\Delta t = 0.025$ seconds and planning horizon $T = 200$. Since the control is one-dimensional, this yields $\theta \in \mathbb{R}^{200}$.

Results. Fig. 2(b) shows boxplots of the trajectory costs across the final particles for each method. Here, TSMC significantly outperforms all baselines. Fig. 2(d) visualizes the best-cost trajectories from each method (final state highlighted in thick black): TSMC is the only method that achieves a successful swing-up close to the upright configuration. Numerically, the best trajectory cost from TSMC is 2389, while the best costs from the baselines are 2967 for Parallel-NUTS, 2967 for Parallel-MALA, 2694 for Parallel-MPPI, and 2562 for Parallel-IPOPT.

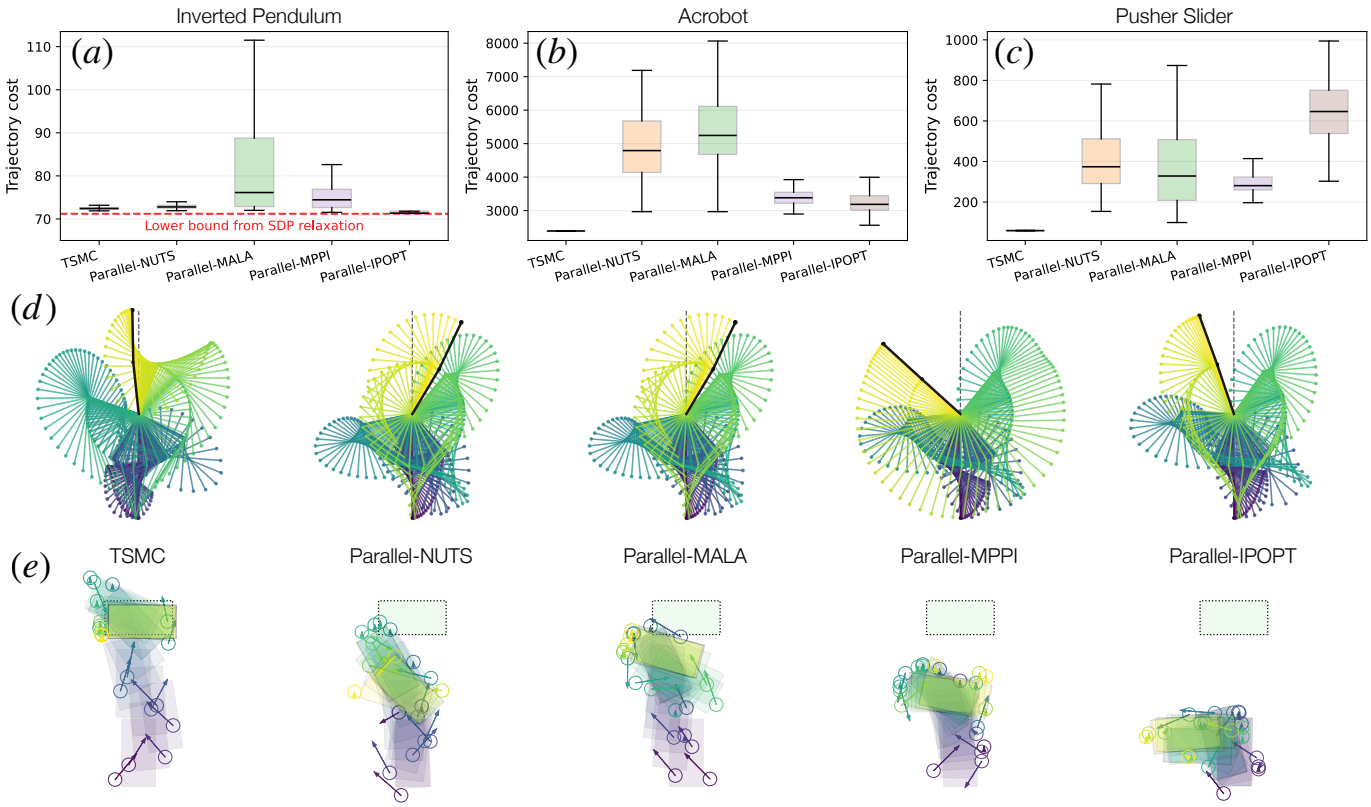

Fig. 2. Trajectory optimization results. (a)–(c): boxplots of final-particle trajectory costs for TSMC and baselines. (d)–(e): best-cost trajectories for Acrobot and pusher–slider, respectively. The colormap indicates progression from initial to final states.

C. Pusher Slider

Setup. We consider the planar pusher–slider problem, which features nonsmooth contact dynamics [52, 49, 32]. Inspired by [40], we model the pusher–slider dynamics only during contact between the pusher (a circular disc) and the slider (a rectangular box), see Fig. 2(e); any free-space motion of the pusher is assumed to be planned separately. The state is the planar pose of the slider in world coordinates, $x = [b_x, b_y, \alpha] \in \mathbb{R}^3$, where (b_x, b_y) is the box center and α is the box orientation. We use a 4D *contact-mode* control $u = [e, s, v_x, v_y] \in \mathbb{R}^4$: $e \in \{1, 2, 3, 4\}$ selects the contacted edge, $s \in [-s_{\max}, s_{\max}]$ selects the contact location along that edge (with $s = \pm 1$ corresponding to corners and $s_{\max} \in (0, 1]$ as a safety margin), and v_x, v_y are pusher velocity commands expressed in the slider frame and bounded componentwise by $|v_x|, |v_y| \leq u_{\max}$. Given (e, s, v_x, v_y), we use the quasi-static model in [49] to compute the slider twist and update the slider pose. This model is differentiable except for the discrete edge variable e. To obtain an end-to-end differentiable pipeline, we relax e by introducing a continuous variable $\hat{e} \in [1, 4]$, computing its distances to the integers in $\{1, 2, 3, 4\}$, mapping these distances to a categorical distribution via a softmax, and then using the Gumbel–Softmax trick [37] to sample e through a differentiable relaxation. The trajectory cost includes per-stage pose error to the goal pose and a terminal pose error, together with a temporal-smoothness penalty that discourages frequent contact-mode switches. We use a planning horizon $T = 200$, which yields $\theta \in \mathbb{R}^{800}$. For CASADI+IPOPT, we cannot use Gumbel–Softmax; instead, we use a soft mixture model that forms a contact normal direction as a weighted combination of per-edge normals using the softmax weights. Since this mixture is not physically realistic, after IPOPT we round the edge variables to the nearest integers and forward simulate to obtain physically valid trajectories for cost evaluation.

Results. Fig. 2(c) shows boxplots of trajectory costs across the final particles for each method. Here, TSMC significantly outperforms all baselines and is the only method that reaches the goal pose. Fig. 2(e) visualizes the best-cost trajectories.

VI. Experiments: Policy Optimization

We next turn to policy optimization benchmarks. We initially aimed to evaluate on MuJoCo locomotion tasks (*e.g.,* HalfCheetah and Ant), which involve contact dynamics. While both MuJoCo JAX (MJX) [72] and Brax [24] provide JAX-based implementations, we found MJX's contact solver relies on a `while` loop that prevents reverse-mode differentiation, as documented in GitHub issues [1, 70]. Brax supports differentiable contact, but for HalfCheetah we observed gradient norms as large as 10^9 to 10^{13}, which makes it unsuitable for our framework. We therefore benchmark on classical control tasks without contact—where we can already compare against strong RL baselines—and leave contact-rich tasks to future work as differentiable contact solvers in JAX mature.

Implementation. We use the same implementation details as in Section V, except that for the extended-space TSMC variant in Section IV-B we implement adaptive tempering from

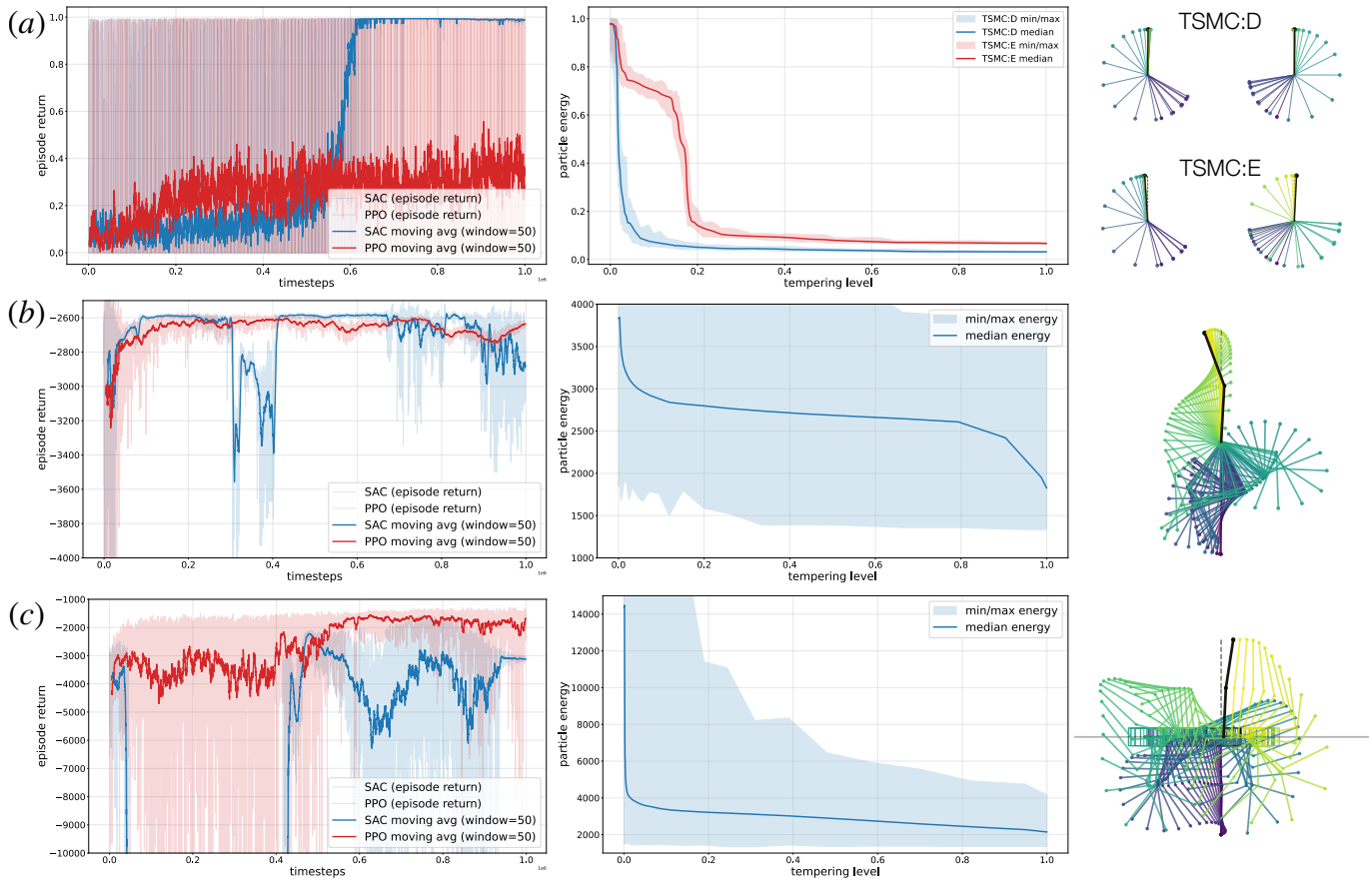

Fig. 3. Policy optimization results. Left to right: episode returns of PPO and SAC w.r.t. interaction steps; distribution of particle energies w.r.t. tempering levels in TSMC; trajectory rollouts of the policy learned by TSMC. (a) Inverted pendulum with sparse rewards; (b) Acrobot; (c) Double pendulum on a cart.

scratch, since BLACKJAX's built-in adaptive TSMC routine does not support alternating two block moves over θ and $\mathbf{x}_0$.

Policy Architecture. We parameterize π_θ as a multi-layer perceptron (MLP) with two hidden layers of 32 units each.

Baselines. We compare against two policy optimization algorithms for RL: proximal policy optimization (PPO) [64] and soft actor-critic (SAC) [28]. We wrap each dynamical system as a Gymnasium environment [73] and run PPO and SAC using Stable-Baselines3 [59] with default hyperparameters, 10^6 timesteps, and 32 parallel environments.

A. Inverted Pendulum with Sparse Rewards

Setup. Following Section V-A, we use a time step $\Delta t = 0.1$ seconds and planning horizon $T = 32$. To make the task challenging, we use a terminal cost only formulation. Let $(x_{T,\text{pos}}, x_{T,\text{vel}}) \in \mathbb{R}^4$ denote the pendulum's position and velocity at time T, and let $(x_{g,\text{pos}}, x_{g,\text{vel}})$ be the goal state. Define the goal distance $\text{dist} = \sqrt{\|x_{T,\text{pos}} - x_{g,\text{pos}}\|^2} + 0.5\sqrt{\|x_{T,\text{vel}} - x_{g,\text{vel}}\|^2}$, and set the terminal cost to $\ell_T = 1 - \sigma\left((\varepsilon - \text{dist})/\epsilon\right)$, where $\sigma(\cdot)$ is the sigmoid function, $\varepsilon = 0.1$, and $\epsilon = 0.02$ (the sparse reward is $1 - \ell_T$). The policy parameters satisfy $\theta \in \mathbb{R}^{1184}$. We set the initial-state distribution to be uniform over the 2D box $[-\pi, \pi] \times [-\pi, \pi]$. For both TSMC:D (deterministic approximation) and TSMC:E (extended-space), we use $B = 3000$ and $N = 32$.

Results. Fig. 3(a) (left) shows learning curves for PPO and SAC. PPO does not reliably improve within the allotted budget, whereas SAC attains strong returns, consistent with the benefits of off-policy learning and experience replay. Fig. 3(a) (middle) shows the distribution of particle energies across tempering levels in TSMC. Both variants achieve similar performance as SAC. Fig. 3(a) (right) visualizes representative rollouts.

B. Acrobot

Setup. We follow Section V-B and use a time step $\Delta t = 0.04$ seconds with planning horizon $T = 100$, which yields $\theta \in \mathbb{R}^{1184}$. We use a single-state initial distribution, *i.e.,* $\mu = \delta_{x_0}$, with x_0 corresponding to the bottom-right configuration. We focus on a single initial state because Acrobot is substantially more challenging; as we show below, this setting is already difficult for both PPO and SAC. Under $\mu = \delta_{x_0}$, the TSMC variants coincide; we use a large particle count $N = 16000$ for TSMC (smaller particle sets did not yield a successful policy).

Results. Fig. 3(b) (left) shows learning curves for PPO and SAC. Neither method reliably reaches a successful swing-up. Fig. 3(b) (middle) shows the distribution of particle energies across tempering levels in TSMC. The final-step best-cost particle attains a trajectory cost of 1338.15, which improves over the average trajectory cost over the final 50 episodes of 2311.14 for PPO and 2482.44 for SAC. Fig. 3(b) (right) visualizes the resulting swing-up trajectory from TSMC. To investigate whether the performance gap is due to limited

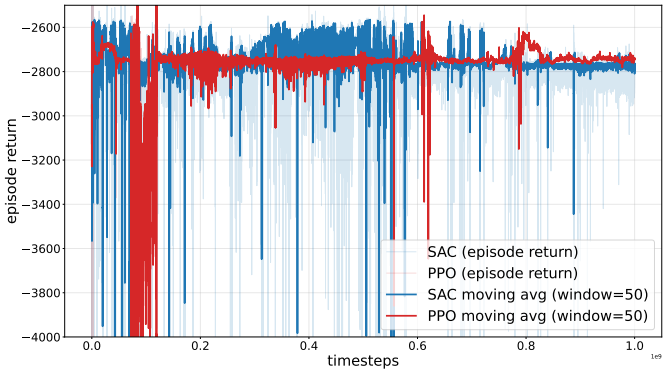

Fig. 4. Running PPO and SAC on Acrobot for 10^9 environment steps.

training budget, we run PPO and SAC for 10^9 environment steps. Fig. 4 shows the learning curves. Neither method reaches a successful swing-up.

C. Double Pendulum on a Cart

Setup. We consider swinging up a two-link double pendulum mounted on a cart to the upright configuration [26]. This task is similar to Acrobot but has a higher-dimensional state, $x \in \mathbb{R}^6$. We discretize the continuous-time dynamics using RK4 with a time step $\Delta t = 0.06$ seconds and planning horizon $T = 100$, yielding $\theta \in \mathbb{R}^{1248}$. We use a single-state initial distribution, with x_0 corresponding to the bottom-right configuration. We use $N = 14000$ particles for TSMC.

Results. Fig. 3(c) (left) shows learning curves for PPO and SAC. The average trajectory costs over the final 50 episodes are 1713.00 for PPO and 3121.77 for SAC. Fig. 3(c) (middle) shows particle energy distributions across tempering levels in TSMC; the best cost among the final particles is 1350.50, improving over both PPO and SAC. Fig. 3(c) (right) visualizes the resulting swing-up trajectory from TSMC.

VII. Conclusion

We introduced TSMC, a sampling-based framework for trajectory and policy optimization that exploits differentiable dynamics, and demonstrated its robustness and versatility across TO and PO benchmarks.

Limitations. While TSMC is naturally parallelizable across particles, it can be computationally and memory intensive: differentiating through rollouts can be slow, and maintaining many particles can be demanding on GPU memory, potentially requiring multi-GPU training.

Future work. (i) Integrate TSMC with differentiable contact solvers and evaluate it on training RL policies for contact-rich tasks. (ii) Accelerate TSMC via customized low-level GPU kernels and hardware acceleration.

Acknowledgments

We thank Sitan Chen for discussions on sampling from the Boltzmann-tilted distribution, and Shucheng Kang and Haoyu Han for helpful conversations throughout the project and for proofreading early drafts. We acknowledge funding from the Office of Naval Research grant N00014-25-1-2322.

References

[1] *[MJX] `jax.lax.while_loop` in `solver.py` prevents computation of backward gradients (#2259)*. https://github.com/google-deepmind/mujoco/issues/2259. 2024.

[2] Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. "Cosmos world foundation model platform for physical ai". In: *arXiv preprint arXiv:2501.03575* (2025).

[3] Bo Ai, Stephen Tian, Haochen Shi, Yixuan Wang, Tobias Pfaff, Cheston Tan, Henrik I Christensen, Hao Su, Jiajun Wu, and Yunzhu Li. "A review of learning-based dynamics models for robotic manipulation". In: *Science Robotics* 10.106 (2025).

[4] Johan Alenlöv, Arnaud Doucet, and Fredrik Lindsten. "Pseudo-Marginal Hamiltonian Monte Carlo". In: *Journal of Machine Learning Research* 22.141 (2021), pp. 1–45.

[5] Joseph Amigo, Rooholla Khorrambakht, Elliot Chane-Sane, Nicolas Mansard, and Ludovic Righetti. "First order model-based rl through decoupled backpropagation". In: *Conference on Robot Learning (CoRL)*. 2025.

[6] Brandon Amos, Samuel Stanton, Denis Yarats, and Andrew Gordon Wilson. "On the model-based stochastic value gradient for continuous reinforcement learning". In: *Annual Learning for Dynamics and Control Conference (L4DC)*. PMLR. 2021, pp. 6–20.

[7] Joel AE Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. "CasADi: a software framework for nonlinear optimization and optimal control". In: *Mathematical Programming Computation* 11.1 (2019), pp. 1–36.

[8] Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. "Particle Markov Chain Monte Carlo Methods". In: *Journal of the American Statistical Association* 105.491 (2010), pp. 1541–1554.

[9] Christophe Andrieu and Gareth O. Roberts. "The Pseudo-Marginal Approach for Efficient Monte Carlo Computations". In: *The Annals of Statistics* 37.2 (2009), pp. 697–725.

[10] Julian Besag. "Comments on "Representations of knowledge in complex systems" by U. Grenander and M. I. Miller". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* (1994).

[11] Michael Betancourt. "A conceptual introduction to Hamiltonian Monte Carlo". In: *arXiv preprint arXiv:1701.02434* (2017).

[12] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. *JAX: composable transformations of Python+NumPy programs*. Version 0.3.13. 2018.

[13] Arthur E. Bryson and Yu-Chi Ho. *Applied Optimal Control: Optimization, Estimation, and Control*. Waltham, MA: Blaisdell Pub. Co., 1969.

[14] Alberto Cabezas, Adrien Corenflos, Junpeng Lao, Rémi Louf, Antoine Carnec, Kaustubh Chaudhari, Reuben Cohn-Gordon, Jeremie Coullon, Wei Deng, Sam Duffield, et al. "BlackJAX: composable Bayesian inference in JAX". In: *arXiv preprint arXiv:2402.10797* (2024).

[15] Tianqi Chen, Emily Fox, and Carlos Guestrin. "Stochastic gradient hamiltonian monte carlo". In: *International Conference on Machine Learning (ICML)*. PMLR. 2014, pp. 1683–1691.

[16] Nicolas Chopin, Pierre E. Jacob, and Omiros Papaspiliopoulos. "SMC^2: An Efficient Algorithm for Sequential Analysis of State Space Models". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 75.3 (2013), pp. 397–426.

[17] Ignasi Clavera, Violet Fu, and Pieter Abbeel. "Model-augmented actor-critic: Backpropagating through paths". In: *International Conference on Learning Representations (ICLR)*. 2020.

[18] Bruce A Conway. *Spacecraft trajectory optimization*. Vol. 29. Cambridge University Press, 2010.

[19] Chenguang Dai, Jeremy Heng, Pierre E Jacob, and Nick Whiteley. "An invitation to sequential Monte Carlo samplers". In: *Journal of the American Statistical Association* 117.539 (2022), pp. 1587–1600.

[20] Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. "A tutorial on the cross-entropy method". In: *Annals of operations research* 134.1 (2005), pp. 19–67.

[21] Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. "Sequential Monte Carlo samplers". In: *Journal of the Royal Statistical Society: Series B* 68.3 (2006), pp. 411–436.

[22] Arnaud Doucet, Nando De Freitas, Neil James Gordon, et al. *Sequential Monte Carlo methods in practice*. Vol. 1. 2. Springer, 2001.

[23] Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. "Hybrid monte carlo". In: *Physics letters B* 195.2 (1987), pp. 216–222.

[24] C Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. "Brax–a differentiable physics engine for large scale rigid body simulation". In: *arXiv preprint arXiv:2106.13281* (2021).

[25] Mark Girolami and Ben Calderhead. "Riemann manifold Langevin and Hamiltonian Monte Carlo methods". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 73.2 (2011), pp. 123–214.

[26] Knut Graichen, Michael Treuer, and Michael Zeitz. "Swing-up of the double pendulum on a cart by feedforward and feedback control with experimental validation". In: *Automatica* 43.1 (2007), pp. 63–71.

[27] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. "Reinforcement learning with deep energy-based policies". In: *International Conference on Machine Learning (ICML)*. PMLR. 2017, pp. 1352–1361.

[28] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *International Conference on Machine Learning (ICML)*. 2018, pp. 1861–1870.

[29] Ernst Hairer, Christian Lubich, and Gerhard Wanner. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*. 2nd ed. Vol. 31. Springer Series in Computational Mathematics. Springer, 2006.

[30] James D Hamilton. *Time series analysis*. Princeton university press, 2020.

[31] Matthew D. Hoffman and Andrew Gelman. "The No-U-Turn Sampler: adaptively setting path lengths in Hamiltonian Monte Carlo". In: *Journal of Machine Learning Research* 15.1 (2014), pp. 1593–1623.

[32] François Robert Hogan and Alberto Rodriguez. "Feedback control of the pusher-slider system: A story of hybrid and underactuated contact dynamics". In: *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*. Springer. 2020, pp. 800–815.

[33] Taylor A Howell, Simon Le Cleach, J Zico Kolter, Mac Schwager, and Zachary Manchester. "Dojo: A differentiable simulator for robotics". In: *arXiv preprint arXiv:2203.00806* 9.2 (2022), p. 4.

[34] Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. "Difftaichi: Differentiable programming for physical simulation". In: *International Conference on Learning Representations (ICLR)*. 2019.

[35] Pierre E. Jacob and Alexandre H. Thiéry. "On Nonnegative Unbiased Estimators". In: *The Annals of Statistics* 43.2 (2015), pp. 769–784.

[36] David H. Jacobson. "Optimal Stochastic Linear Systems with Exponential Performance Criteria and Their Relation to Deterministic Differential Games". In: *IEEE Transactions on Automatic Control* 18.2 (1973), pp. 124–131.

[37] Eric Jang, Shixiang Gu, and Ben Poole. "Categorical reparameterization with gumbel-softmax". In: *International Conference on Learning Representations (ICLR)*. 2017.

[38] E. T. Jaynes. "Information Theory and Statistical Mechanics". In: *Phys. Rev.* (1957).

[39] Sham M. Kakade. "A Natural Policy Gradient". In: *Conference on Neural Information Processing Systems (NeurIPS)*. 2001.

[40] Shucheng Kang, Guorui Liu, and Heng Yang. "Global Contact-Rich Planning with Sparsity-Rich Semidefinite Relaxations". In: *Robotics: Science and Systems (RSS)*. 2025.

[41] Shucheng Kang, Xiaoyang Xu, Jay Sarva, Ling Liang, and Heng Yang. "Fast and certifiable trajectory optimization". In: *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*. 2024.

[42] Hilbert J Kappen. "Path integrals and symmetry breaking for optimal control theory". In: *Journal of statistical mechanics: theory and experiment* 2005.11 (2005), P11011.

[43] Jovana Kondic. "Monte Carlo Methods for Motion Planning and Goal Inference". PhD thesis. Massachusetts Institute of Technology, 2024.

[44] Jere Koskela, Toni Karvonen, et al. "A debiased Bernoulli factory and unbiased estimation of a probability". In: *arXiv preprint arXiv:2510.01941* (2025).

[45] Quentin Le Lidec, Louis Montaut, Yann de Mont-Marin, Fabian Schramm, and Justin Carpentier. "Highly-Efficient Differentiable Simulation for Robotics". In: *arXiv preprint arXiv:2409.07107* (2025).

[46] Taeyoung Lee. "Computational geometric mechanics and control of rigid bodies". PhD thesis. University of Michigan, 2008.

[47] Sergey Levine. "Reinforcement learning and control as probabilistic inference: Tutorial and review". In: *arXiv preprint arXiv:1805.00909* (2018).

[48] Chenhao Li, Andreas Krause, and Marco Hutter. "Robotic world model: A neural network simulator for robust policy optimization in robotics". In: *arXiv preprint arXiv:2501.10100* (2025).

[49] Kevin M Lynch and Matthew T Mason. "Stable pushing: Mechanics, controllability, and planning". In: *International Journal of Robotics Research (IJRR)* 15.6 (1996), pp. 533–556.

[50] Haitong Ma, Tianyi Chen, Kai Wang, Na Li, and Bo Dai. "Efficient Online Reinforcement Learning for Diffusion Policy". In: *International Conference on Machine Learning (ICML)*. 2025.

[51] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[52] Matthew T Mason. "Mechanics and planning of manipulator pushing operations". In: *International Journal of Robotics Research (IJRR)* 5.3 (1986), pp. 53–71.

[53] Radford M Neal. "Annealed importance sampling". In: *Statistics and computing* 11.2 (2001), pp. 125–139.

[54] Radford M. Neal. "MCMC Using Hamiltonian Dynamics". In: *Handbook of Markov Chain Monte Carlo*. Ed. by Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. Chapman and Hall/CRC, 2011. Chap. 5.

[55] Rhys Newbury, Jack Collins, Kerry He, Jiahe Pan, Ingmar Posner, David Howard, and Akansel Cosgun. "A review of differentiable simulators". In: *IEEE Access* (2024).

[56] Anselm Paulus, A René Geist, Pierre Schumacher, Vít Musil, and Georg Martius. "Hard Contacts with Soft Gradients: Refining Differentiable Simulators for Learning and Control". In: *arXiv preprint arXiv:2506.14186* (2025).

[57] Michael Posa, Cecilia Cantu, and Russ Tedrake. "A direct method for trajectory optimization of rigid bodies through contact". In: *The International Journal of Robotics Research* 33.1 (2014), pp. 69–81.

[58] Han Qi, Haocheng Yin, Aris Zhu, Yilun Du, and Heng Yang. "Strengthening generative robot policies through predictive world modeling". In: *arXiv preprint arXiv:2502.00622* (2025).

[59] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. "Stable-baselines3: Reliable reinforcement learning implementations". In: *Journal of machine learning research* 22.268 (2021), pp. 1–8.

[60] James Blake Rawlings, David Q Mayne, Moritz Diehl, et al. *Model predictive control: theory, computation, and design*. Vol. 2. Nob Hill Publishing Madison, WI, 2020.

[61] Christian P Robert and George Casella. *Monte Carlo statistical methods*. Vol. 2. Springer, 1999.

[62] Brian Sallans and Geoffrey E Hinton. "Using free energies to represent Q-values in a multiagent reinforcement learning task". In: *Conference on Neural Information Processing Systems (NeurIPS)*. Vol. 13. 2000.

[63] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. "Trust region policy optimization". In: *International Conference on Machine Learning (ICML)*. PMLR. 2015, pp. 1889–1897.

[64] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[65] Clemens Schwarke, Victor Klemm, Joshua Bagajo, Jean Pierre Sleiman, Ignat Georgiev, Jesus Tordesillas Torres, and Marco Hutter. "Learning Deployable Locomotion Control via Differentiable Simulation". In: *9th Annual Conference on Robot Learning*. 2025.

[66] Jacob Shekel. "Test functions for multimodal search techniques". In: *Fifth Annual Princeton Conf. on Information Science and Systems*. 1971.

[67] Hyung Ju Suh, Max Simchowitz, Kaiqing Zhang, and Russ Tedrake. "Do differentiable simulators give better policy gradients?" In: *International Conference on Machine Learning (ICML)*. PMLR. 2022, pp. 20668–20696.

[68] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

[69] Sangli Teng, Ashkan Jasour, Ram Vasudevan, and Maani Ghaffari. "Convex geometric motion planning on lie groups via moment relaxation". In: *Robotics: Science and Systems (RSS)*. 2023.

[70] *The CG Solver in MJX doesn't support reverse-mode differentiation (Issue #1182)*. https://github.com/google-deepmind/mujoco/issues/1182. Nov. 2023.

[71] Emanuel Todorov. "Linearly-solvable Markov decision problems". In: *Conference on Neural Information Processing Systems (NeurIPS)*. Vol. 19. 2006.

[72] Emanuel Todorov, Tom Erez, and Yuval Tassa. "Mujoco: A physics engine for model-based control". In: IEEE. 2012, pp. 5026–5033.

[73] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. "Gymnasium: A standard interface for reinforcement learning environments". In: *Conference on Neural Information Processing Systems (NeurIPS)*. 2025.

[74] Robin Verschueren, Gianluca Frison, Dimitris Kouzoupis, Jonathan Frey, Niels van Duijkeren, Andrea Zanelli, Branimir Novoselnik, Thivaharan Albin, Rien Quirynen, and Moritz Diehl. "acados—a modular open-source framework for fast embedded optimal control". In: *Mathematical Programming Computation* 14.1 (2022), pp. 147–183.

[75] Martin J Wainwright. *High-dimensional statistics: A non-asymptotic viewpoint*. Vol. 48. Cambridge university press, 2019.

[76] Ziyu Wang, Shakir Mohamed, and Nando Freitas. "Adaptive hamiltonian and riemann manifold monte carlo". In: *International Conference on Machine Learning (ICML)*. PMLR. 2013, pp. 1462–1470.

[77] Nina Wiedemann, Valentin Wüest, Antonio Loquercio, Matthias Müller, Dario Floreano, and Davide Scaramuzza. "Training efficient controllers via analytic policy gradient". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2023.

[78] Grady Williams, Andrew Aldrich, and Evangelos Theodorou. "Model predictive path integral control using covariance variable importance sampling". In: *arXiv preprint arXiv:1509.01149* (2015).

[79] Grady Williams, Andrew Aldrich, and Evangelos A Theodorou. "Model predictive path integral control: From theory to parallel computation". In: *Journal of Guidance, Control, and Dynamics* 40.2 (2017), pp. 344–357.

[80] Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. "Daydreamer: World models for physical robot learning". In: *Conference on Robot Learning (CoRL)*. PMLR. 2023, pp. 2226–2240.

[81] Jie Xu, Viktor Makoviychuk, Yashraj Narang, Fabio Ramos, Wojciech Matusik, Animesh Garg, and Miles Macklin. "Accelerated policy learning with parallel differentiable simulation". In: *International Conference on Learning Representations (ICLR)*. 2022.

[82] Kaifeng Zhang, Baoyu Li, Kris Hauser, and Yunzhu Li. "Particle-Grid Neural Dynamics for Learning Deformable Object Models from RGB-D Videos". In: *Robotics: Science and Systems (RSS)*. 2025.

[83] Tan Zhi-Xuan, Jovana Kondic, Stewart Slocum, Joshua B Tenenbaum, Vikash K Mansinghka, and Dylan Hadfield-Menell. "Bayesian Inverse Motion Planning for Online Goal Inference in Continuous Domains". In: *ICRA 2023 Workshop on Cognitive Modeling in Robot Learning*. 2023.