Stochastic Optimal Control Matching

Carles Domingo-Enrich, Jiequn Han, Brandon Amos, Joan Bruna, Ricky T. Q. Chen

Thomas Mousseau

August 27, 2025

Overview

1. Setup and Preliminaries
2. Stochastic Optimal Control Matching
3. Experiments and results
4. Conclusion

Evolution of Generative Models

2020 **DDPM:** Denoising Diffusion Probabilistic Models interpret generation as reversing a discrete noise-adding process, learning to denoise at each step. They produced high-quality samples but required thousands of slow sampling steps.

2021 **Score-based Models:** Score-based generative models extended diffusion to continuous-time SDEs, learning the score function ($\nabla_x \log p_t(x)$) to reverse a stochastic diffusion process. This unified diffusion with stochastic control, allowed probability flow ODEs, and sped up sampling.

2023 **Flow Matching:** Flow matching views generation as learning a deterministic ODE vector field that directly transports a simple distribution (e.g., Gaussian) to data. This removed stochasticity and significantly improved efficiency compared to diffusion/score methods.

SOC as the Foundation of Generative Models

The Core Challenge: Unnormalized Densities

Generative models must sample from complex distributions $p_{\text{data}}(x) = \frac{1}{Z}\tilde{p}_{\text{data}}(x)$ where the normalization constant $Z = \int \tilde{p}_{\text{data}}(x)dx$ is intractable to compute. This intractability arises from the curse of dimensionality when integrating over high-dimensional spaces.

SOC Connection

Key Insight:
Transform tractable distributions (Gaussian) to complex target distributions through optimal control policies.

This bridges the gap between:

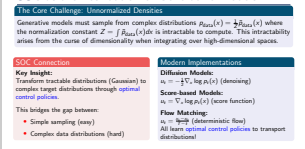
- Simple sampling (easy)
- Complex data distributions (hard)

Modern Implementations

Diffusion Models:
$u_t = -\frac{1}{2}\nabla_x \log p_t(x)$ (denoising)

Score-based Models:
$u_t = \nabla_x \log p_t(x)$ (score function)

Flow Matching:
$u_t = \frac{x_1 - x_0}{T - t}$ (deterministic flow)
All learn optimal control policies to transport distributions!



Unnormalized Densities: The fundamental challenge in generative modeling is sampling from distributions $p(x) = \frac{1}{Z}e^{-E(x)}$ where Z is unknown. SOC provides the mathematical framework to construct sampling procedures.

Historical Context: From Langevin dynamics to modern diffusion models, all major breakthroughs in generative modeling can be understood through the lens of stochastic optimal control theory.

What is a Stochastic Control Problem?

Control-Affine Stochastic Differential Equation

The general form of a controlled stochastic process:

$$dX_t^u = \big(b(X_t^u, t) + \sigma(t)u(X_t^u, t)\big)dt + \sqrt{\lambda}\sigma(t)dB_t \tag{1}$$

State Process: $X_t^u \in \mathbb{R}^d$ (system state under control u at time t)

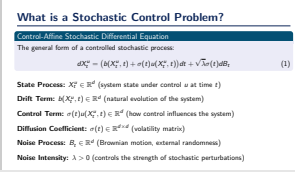
Drift Term: $b(X_t^u, t) \in \mathbb{R}^d$ (natural evolution of the system)

Control Term: $\sigma(t)u(X_t^u, t) \in \mathbb{R}^d$ (how control influences the system)

Diffusion Coefficient: $\sigma(t) \in \mathbb{R}^{d\times d}$ (volatility matrix)

Noise Process: $B_t \in \mathbb{R}^d$ (Brownian motion, external randomness)

Noise Intensity: $\lambda > 0$ (controls the strength of stochastic perturbations)



Control-Affine Structure: The "control-affine" property means the control u enters linearly (affinely) in the drift term. This is the most general practical form for controlled SDEs, encompassing most applications in finance, robotics, and machine learning.

Je dois aussi expliquer l'importance que le steering input ainsi que le noise sont tous les 2 multipliés par sigma(t) ce qui signifie que ...

The Optimal Control Objective

Cost Function to Minimize

Find the optimal control policy $u^* \in \mathcal{U}$ that minimizes:

$$\min_{u \in \mathcal{U}} \mathbb{E}\left[\int_0^T \left(\frac{1}{2}\|u(X_t^u, t)\|^2 + f(X_t^u, t)\right) dt + g(X_T^u)\right] \tag{2}$$

Control Effort: $\frac{1}{2}\|u(X_t^u, t)\|^2$ (penalizes large control actions)

Running Cost: $f(X_t^u, t)$ (ongoing cost during the process evolution)

Terminal Cost: $g(X_T^u)$ (final cost based on end state at time T)

Control Space: $\mathcal{U}$ (set of admissible control policies)

The Optimal Control Objective

Cost Function to Minimize

Find the optimal control policy $u^* \subset \mathcal{U}$ that minimizes:

$$\min_{u \in \mathcal{U}} \mathbb{E}\left[\int_0^T \left(\frac{1}{2}\|u(X_t^u, t)\|^2 + f(X_t^u, t)\right) dt + g(X_T^u)\right] \tag{2}$$

Control Effort: $\frac{1}{2}\|u(X_t^u, t)\|^2$ (penalizes large control actions)

Running Cost: $f(X_t^u, t)$ (ongoing cost during the process evolution)

Terminal Cost: $g(X_T^u)$ (final cost based on end state at time T)

Control Space: $\mathcal{U}$ (set of admissible control policies)

SOC in Practice

Aspect	Steering Actuator	Diffusion Models	Flow Models
State X_t	Vehicle position, velocity	Noisy data sample	Clean data sample
Control u_t	Steering angle, throttle	Denoising direction	Flow velocity field
Dynamics Source	Newtonian mechanics (vehicle dynamics)	Forward noising process	No natural drift (learnable)
Drift $b(X_t, t)$	Kinematic equations	Predetermined schedule	$b = 0$ (control learns drift)
Control Goal	Reach target safely	Reverse noise process	Transport distributions
Noise $\sqrt{\lambda}\sigma dB_t$	Road disturbances	Brownian motion	Optional stochasticity

SOC in Practice

Aspect	Steering Actuator	Diffusion Models	Flow Models
State X_t	Vehicle position, velocity	Noisy data sample	Clean data sample
Control u_t	Steering angle, throttle	Denoising direction	Flow velocity field
Dynamics Source	Newtonian mechanics (vehicle dynamics)	Forward noising process	No natural drift (learnable)
Drift $b(X_t, t)$	Kinematic equations	Predetermined schedule	$b = 0$ (control learns drift)
Control Goal	Reach target safely	Reverse noise process	Transport distributions
Noise $\sqrt{2}\sigma dB_t$	Road disturbances	Brownian motion	Optional stochasticity

Steering Actuator: The dynamics come from well-established physics - Newton's laws, kinematics, and vehicle dynamics. The control optimizes safety and efficiency.

Diffusion Models: The drift is determined by the forward noising process (e.g., $\beta(t)$), and control learns to reverse this predetermined corruption.

Flow Models: No natural drift exists - the control u_t directly becomes the drift term, learning the entire velocity field that transports distributions.

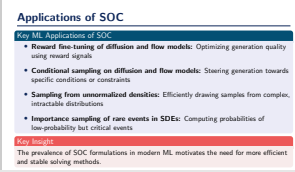
Applications of SOC

Key ML Applications of SOC

- **Reward fine-tuning of diffusion and flow models:** Optimizing generation quality using reward signals
- **Conditional sampling on diffusion and flow models:** Steering generation towards specific conditions or constraints
- **Sampling from unnormalized densities:** Efficiently drawing samples from complex, intractable distributions
- **Importance sampling of rare events in SDEs:** Computing probabilities of low-probability but critical events

Key Insight

The prevalence of SOC formulations in modern ML motivates the need for more efficient and stable solving methods.



Je dois etre capable de vraiment bien expliquer les applications du SOC dans le ML moderne pour pouvoir demontrer l'importance d'avoir pris ce papier

Solve SOC Problems in Low Dimension

Classical Approach: Hamilton-Jacobi-Bellman PDE

In small state spaces (1D, 2D, sometimes 3D), discretize the state space and solve the HJB PDE directly:

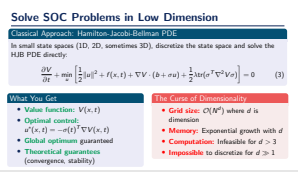
$$\frac{\partial V}{\partial t} + \min_u \left[\frac{1}{2}\|u\|^2 + f(x,t) + \nabla V \cdot (b + \sigma u) + \frac{1}{2}\lambda \mathrm{tr}(\sigma^T \nabla^2 V \sigma) \right] = 0 \tag{3}$$

What You Get

- **Value function:** $V(x,t)$
- **Optimal control:** $u^*(x,t) = -\sigma(t)^T \nabla V(x,t)$
- **Global optimum** guaranteed
- **Theoretical guarantees** (convergence, stability)

The Curse of Dimensionality

- **Grid size:** $\mathcal{O}(N^d)$ where d is dimension
- **Memory:** Exponential growth with d
- **Computation:** Infeasible for $d > 3$
- **Impossible** to discretize for $d \gg 1$



Works perfectly in low dimensions, but state space grows exponentially with dimension.

Classical HJB: In low dimensions, you can discretize the entire state space on a grid and solve the HJB PDE using finite difference methods. This gives you the exact solution but becomes computationally impossible as dimension increases.

Curse of Dimensionality: For a d-dimensional problem with N grid points per dimension, you need N^d total grid points. Even modest problems ($d = 10$, $N = 100$) require $100^{10} = 10^{20}$ grid points.

Solve SOC Problems in High Dimension

Modern Approach: Adjoint Methods

Cannot discretize high-dimensional spaces, so use **adjoint methods** to optimize control directly:

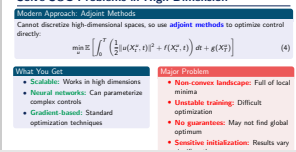
$$\min_u \mathbb{E}\left[\int_0^T \left(\frac{1}{2}\|u(X_t^u, t)\|^2 + f(X_t^u, t)\right) dt + g(X_T^u)\right] \tag{4}$$

What You Get

- **Scalable:** Works in high dimensions
- **Neural networks:** Can parameterize complex controls
- **Gradient-based:** Standard optimization techniques

Major Problem

- **Non-convex landscape:** Full of local minima
- **Unstable training:** Difficult optimization
- **No guarantees:** May not find global optimum
- **Sensitive initialization:** Results vary



Highly non-convex functional landscape makes optimization extremely challenging. Traditional SOC methods struggle with local minima and unstable training dynamics.

Adjoint Methods: These methods solve the forward SDE and then use the adjoint equation to compute gradients efficiently. This allows gradient-based optimization of the control parameters without discretizing the state space.

Non-convex Optimization: The major challenge is that the resulting optimization landscape is highly non-convex, leading to difficult optimization with many local minima and unstable training dynamics.

Similar Trend in Generative Modeling

The Same Pattern: Non-convex → Least-Squares

Generative modeling experienced the same transition from non-convex to convex optimization

Continuous Normalizing Flows

Method: Learn invertible transformations

$$\frac{dx}{dt} = f_\theta(x, t) \tag{5}$$

Problem:

- Use **adjoint methods** for gradients
- **Non-convex** optimization landscape
- Difficult training, unstable dynamics

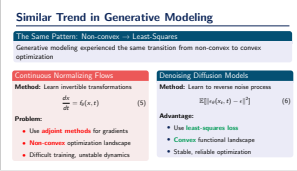
Denoising Diffusion Models

Method: Learn to reverse noise process

$$\mathbb{E}[\|\epsilon_\theta(x_t, t) - \epsilon\|^2] \tag{6}$$

Advantage:

- Use **least-squares loss**
- **Convex** functional landscape
- Stable, reliable optimization



Insight: Apply the same principle to SOC! Replace non-convex adjoint methods with **least-squares matching**, creating a **convex optimization landscape** for stochastic optimal control.

Historical Parallel: Continuous Normalizing Flows (CNFs) suffered from the same optimization challenges as traditional SOC - they used adjoint methods and had non-convex landscapes. DDPMs revolutionized generative modeling by reformulating the problem as least-squares regression.

SOCM's Contribution: SOCM brings the same insight to stochastic optimal control, replacing direct optimization with least-squares matching to achieve stable, convex optimization.

SOCM in Context: Optimization Landscapes

Task	Non-convex	Least Squares
Generative Modeling	Maximum Likelihood CNFs	Diffusion models and Flow Matching
Stochastic Optimal Control	Adjoint Methods	**Stochastic Optimal Control Matching**

SOCM Framework: From Classical SOC to SOCM

Dynamics (Same for Both)

$$dX_t^v = (b(X_t^v, t) + \sigma(t)v(X_t^v, t))dt + \sqrt{\lambda}\sigma(t)dB_t, \text{ with } X_0^v \sim p_0 \tag{7}$$

Original SOC Cost Function

$$\min_{u \in \mathcal{U}} \mathcal{L}_{SOC} := \mathbb{E}\left[\int_0^T \left(\frac{1}{2}\|u(X_t^u, t)\|^2 + f(X_t^u, t)\right) dt + g(X_T^u)\right] \tag{8}$$

SOCM Cost Function

$$\min_{u,M} \mathcal{L}_{SOCM}(u, M) := \mathbb{E}\left[\frac{1}{T}\int_0^T \|u(X_t^v, t) - w(t, v, X^v, B, M_t)\|^2 dt \times \alpha(v, X^v, B)\right] \tag{9}$$

Key Point: Same underlying stochastic dynamics, but SOCM uses a least-squares matching approach instead of direct minimization of the original cost.

SOCM Parameters Definition

$$\min_{u,M} \mathcal{L}_{SOCM}(u, M) := \mathbb{E}\left[\frac{1}{T}\int_0^T \|u(X_t^v, t) - w(t, v, X^v, B, M_t)\|^2 dt \times \alpha(v, X^v, B)\right]$$

Where:

$u : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$ is the **control** (policy being learned)

v is a **fixed arbitrary control**, X^v is the solution of the SDE with control v

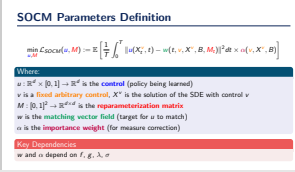
$M : [0,1]^2 \to \mathbb{R}^{d\times d}$ is the **reparameterization matrix**

w is the **matching vector field** (target for u to match)

α is the **importance weight** (for measure correction)

Key Dependencies

w and α depend on f, g, λ, σ



Control u: This is what we're trying to learn - the optimal policy that minimizes our cost function.

Arbitrary Control v: Used to generate sample trajectories for training. The choice of v affects the efficiency but not the correctness of SOCM.

Reparameterization Matrix M: Jointly optimized with u to reduce variance in the gradient estimation through path-wise reparameterization.

Matching Vector Field (1/2)

Reparameterization Function $w(t, v, X^v, B, M_t)$

The **matching vector field** computed via path-wise reparameterization:

$$
\begin{aligned}
w(t, v, X^v, B, M_t) = \sigma(t)^\top \Bigg(& - \int_t^T M_t(s) \nabla_x f(X_s^v, s)\, ds \; - \; M_t(T) \nabla g(X_T^v) \\
& + \int_t^T \big(M_t(s) \nabla_x b(X_s^v, s) - \partial_s M_t(s)\big)(\sigma^{-1}(s))^\top v(X_s^v, s)\, ds \\
& + \sqrt{\lambda} \int_t^T \big(M_t(s) \nabla_x b(X_s^v, s) - \partial_s M_t(s)\big)(\sigma^{-1}(s))^\top dB_s \Bigg)
\end{aligned} \tag{10}
$$

Where $M_t(s)$ is the **reparameterization matrix** (learned jointly with u)

Matching Vector Field (2/2) - Path Integral I

Step 1 — Path-integral form of u^* (Kappen 2005)

What to Show

We want to eliminate the value function V from the optimal control formula and express u^* directly in terms of path integrals.

The value function is defined as the infimum over all controls:

$$V(x,t) = \inf_u J(u;x,t) = \inf_u \mathbb{E}\left[\int_t^T \left(\frac{1}{2}\|u(X_s^u,s)\|^2 + f(X_s^u,s)\right) ds + g(X_T^u) \,\middle|\, X_t^u = x\right]$$

Matching Vector Field (2/2) - Path Integral II

Assume $\sigma(t)$ is invertible and enters both drift (via σu) and diffusion, and control cost is quadratic. This enables the Cole-Hopf linearization.
The **infinitesimal generator** L of the uncontrolled diffusion $dX_t = b(x,t)\,dt + \sqrt{\lambda}\sigma(t)\,dB_t$ is:

$$L := \frac{\lambda}{2}\sum_{i,j=1}^{d}(\sigma\sigma^\top)_{ij}(t)\frac{\partial^2}{\partial x_i \partial x_j} + \sum_{i=1}^{d} b_i(x,t)\frac{\partial}{\partial x_i}$$

The Hamilton-Jacobi-Bellman equation becomes (minimize pointwise in (x,t)):

$$\frac{\partial V}{\partial t} + LV + \inf_u \left[\frac{1}{2}\|u\|^2 + \langle \sigma^\top \nabla V, u\rangle + f\right] = 0$$

Matching Vector Field (2/2) - Path Integral III

To find the infimum, we differentiate the quadratic expression with respect to u (complete the square):

$$\frac{\partial}{\partial u}\left[\frac{1}{2}\|u\|^2 + \langle \sigma^\top \nabla V, u\rangle\right] = u + \sigma^\top \nabla V = 0$$

This gives us the **optimal control**:

$$u^*(x,t) = -\sigma^\top(t)\nabla V(x,t)$$

Substituting back into the HJB equation:

$$\frac{\partial V}{\partial t} + LV - \frac{1}{2}\|\sigma^\top \nabla V\|^2 + f = 0, \quad V(x,T) = g(x)$$

Matching Vector Field (2/2) - Path Integral IV

Reminder (Linear Case)

If $\frac{\partial w}{\partial t} + Lw + q = 0$ with $w(x, T) = h(x)$, then Feynman-Kac gives:

$$w(x,t) = \mathbb{E}\left[h(X_T) - \int_t^T q(X_s, s)ds \,\middle|\, X_t = x\right]$$

The HJB for V is nonlinear; we apply Feynman-Kac to the linear PDE for ϕ, then set $V = -\lambda \log \phi$.

Cole-Hopf transformation: $V(x,t) = -\lambda \log \phi(x,t)$
Linearized PDE for ϕ:

$$\frac{\partial \phi}{\partial t} + L\phi + \frac{1}{\lambda} f\phi = 0, \quad \phi(x,T) = e^{-g(x)/\lambda}$$

Matching Vector Field (2/2) - Path Integral V

Feynman-Kac applied to ϕ:

$$\phi(x,t) = \mathbb{E}_0\left[\exp\left(-\frac{1}{\lambda}\int_t^T f(X_s,s)ds - \frac{1}{\lambda}g(X_T)\right)\middle| X_t = x\right] \text{ (uncontrolled SDE)}$$

Converting back: $V(x,t) = -\lambda \log \phi(x,t)$

Set $Z(x,t) := \phi(x,t)$, so the **partition function** is:

$$Z(x,t) = \mathbb{E}_0\left[\exp\left(-\frac{1}{\lambda}\int_t^T f(X_s,s)ds - \frac{1}{\lambda}g(X_T)\right)\middle| X_t = x\right]$$

Then the optimal control becomes:

Matching Vector Field (2/2) - Path Integral VI

Path-integral Optimal Control

$$u^*(x,t) = \lambda\, \sigma^\top(t)\, \nabla_x \log Z(x,t)$$

$$u^*(x,t) = \lambda\, \sigma^\top(t)\, \nabla_x \log \mathbb{E}_0 \left[\exp\left(-\frac{1}{\lambda} \int_t^T f(X_s, s) ds - \frac{1}{\lambda} g(X_T) \right) \middle| X_t = x \right]$$

∇_x is the gradient w.r.t. the initial condition $X_t = x$, not a terminal-state score.

Matching Vector Field (2/2) - Path Integral VII

Why This Matters

Key Achievement: We eliminate V to avoid solving a nonlinear PDE and reframe the control as a **score function** $\nabla_x \log Z(x,t)$.

We avoid solving a nonlinear PDE and instead target a score that admits a pathwise estimator (next slide). This sets up a score we can approximate with a **low-variance pathwise estimator** in Step 2, which will become the matching vector field w. This is the foundation for SOCM's **least-squares matching** approach.

Matching Vector Field (2/3) - Reparam I

Step 2 — Path-wise Reparameterization Trick

The Challenge

Direct estimation of the path-integral in Step 1 has high variance and is computationally intractable for complex problems.

From Step 1, we need to estimate:

$$u^*(x,t) = \lambda \sigma^T(t) \nabla_x \log Z(x,t)$$

where $Z(x,t) = \mathbb{E}_0[\exp(-\frac{1}{\lambda}\int_t^T f + \frac{1}{\lambda} g) | X_t = x]$
Direct approach: Estimate $\nabla_x \log Z(x,t)$ by:

Matching Vector Field (2/3) - Reparam II

- Sample paths from uncontrolled SDE
- Compute finite differences for gradients
- **High variance** due to exponential weights
- **Biased** finite difference approximations

Instead of estimating gradients of expectations, we use the **path-wise gradient identity**:

$$\nabla_x \mathbb{E}[h(X)] = \mathbb{E}[\nabla_x h(X) + h(X) \cdot \nabla_x \log p(X)]$$

where $p(X)$ is the path measure density.

This allows us to:

Matching Vector Field (2/3) - Reparam III

- Move the gradient **inside** the expectation
- Compute exact gradients along sample paths
- Avoid finite difference approximations
- Reduce variance significantly

To implement path-wise reparameterization effectively, we introduce a matrix $M_t(s)$ where:

- $M_t : [t, T] \to \mathbb{R}^{d \times d}$ is a **learnable matrix function**
- Acts as a "change of variables" for the gradient computation
- **Optimized jointly** with control u to minimize variance
- Enables efficient gradient flow through stochastic integrals

Matching Vector Field (2/3) - Reparam IV

After applying path-wise reparameterization, we get:

Path-wise Reparameterized Control

$$w(t, v, X^v, B, M_t) = \sigma(t)^\top \Bigg(- \int_t^T M_t(s) \nabla_x f(X_s^v, s)\, ds - M_t(T) \nabla g(X_T^v) + \int_t^T \big(M_t(s) \nabla_x b(X_s^v, s) - \partial_s M_t(s)\big)(\sigma^{-1}(s))^\top v(X_s^v, s)\, ds + \sqrt{\lambda} \int_t^T \big(M_t(s) \nabla_x b(X_s^v, s) - \partial_s M_t(s)\big)(\sigma^{-1}(s))^\top dB_s \Bigg) \tag{11}$$

Matching Vector Field (2/3) - Reparam V

Component breakdown:

- **Cost terms:** Direct path-wise gradients of running and terminal costs
- **Drift correction:** Accounts for arbitrary control v and matrix dynamics
- **Stochastic integral:** The key innovation - exact gradient through Brownian motion

Matching Vector Field (2/3) - Reparam VI

Why This Works

Key Achievement: We transform the high-variance score estimation problem into a **tractable regression target** w that can be:

- Computed exactly along sample paths
- Used as a target for least-squares matching: $\min_u \mathbb{E}[\|u(X_t^v, t) - w(\cdot)\|^2]$
- Optimized jointly with M_t to minimize variance

This sets up Step 3: using Girsanov theorem to handle the measure change from arbitrary control v to optimal control.

Matching Vector Field (3/3) - Girsanov Theorem I

Step 3 — Girsanov Theorem Application

The Goal

Transform expectations from the uncontrolled process to a controlled process using measure theory.

From Step 2, we have the matching vector field w computed under the **uncontrolled** process. But in practice, we need to train on trajectories we can actually generate using a reference control v.

Controlled SDE: We simulate trajectories with reference control v:

$$dX_s^v = (b + \sigma v)(X_s^v, s)ds + \sqrt{\lambda}\sigma(s)dB_s$$

Matching Vector Field (3/3) - Girsanov Theorem II

Problem: Expectations are under different measures!

- Our theory assumes the uncontrolled measure
- Our training uses the controlled measure with v

The Girsanov theorem allows us to change measures from uncontrolled to controlled via an **importance weight**:

Matching Vector Field (3/3) - Girsanov Theorem III

Importance Weight $\alpha(v, X^v, B)$

$$\alpha(v, X^v, B) = \exp\Bigg(-\frac{1}{\lambda}\int_0^T f(X_t^v, t)\, dt - \frac{1}{\lambda} g(X_T^v) - \frac{1}{\sqrt{\lambda}}\int_0^T \langle v(X_t^v, t), dB_t \rangle - \frac{1}{2\lambda}\int_0^T \|v(X_t^v, t)\|^2 dt \Bigg) \tag{12}$$

Component breakdown:

- **Cost terms:** Running cost f and terminal cost g
- **Stochastic correction:** Accounts for the stochastic integral with control v

Matching Vector Field (3/3) - Girsanov Theorem IV

- **Control effort:** Quadratic penalty for control magnitude

Under the new measure, the Brownian motion transforms as:

$$dB_s = dB_s^{(v)} - \frac{1}{\sqrt{\lambda}} v(X_s^v, s) ds$$

This **Brownian shift** splits the stochastic term from Step 2 into:

$$\sqrt{\lambda} \int_t^T (M_t \nabla_x b - \partial_s M_t)(\sigma^{-1})^T dB_s = \underbrace{\sqrt{\lambda} \int_t^T (\cdots) dB_s^{(v)}}_{\text{martingale}} - \underbrace{\int_t^T (M_t \nabla_x b - \partial_s M_t)(\sigma^{-1})^T v \, ds}_{\text{deterministic}}$$

Matching Vector Field (3/3) - Girsanov Theorem V

Combining all three steps and multiplying by $\sigma^T(t)$ from Lemma 1:

Complete SOCM Matching Target

$$w(t,v,X^v,B,M_t) = \sigma(t)^T \Bigg(- \int_t^T M_t(s)\nabla_x f(X_s^v,s)\,ds - M_t(T)\nabla g(X_T^v) + \int_t^T \big(M_t(s)\nabla_x b(X_s^v,s) - \partial_s M_t(s)\big)(\sigma^{-1}(s))^T v(X_s^v,s)\,ds + \sqrt{\lambda}\int_t^T \big(M_t(s)\nabla_x b(X_s^v,s) - \partial_s M_t(s)\big)(\sigma^{-1}(s))^T dB_s^{(v)} \Bigg) \tag{13}$$

Matching Vector Field (3/3) - Girsanov Theorem VI

Why Girsanov is Essential

Key Achievement: We can now train on trajectories generated with reference control v, while the importance weight α corrects for the distributional mismatch.

This ensures the target remains **unbiased** for the true optimum, enabling practical implementation of SOCM with:

- Computable matching targets w from sample paths
- Proper measure correction via α
- Least-squares objective: $\min_{u,M} \mathbb{E}[\|u(X_t^v, t) - w(\cdot)\|^2 \times \alpha]$

Importance Weight (1/2)

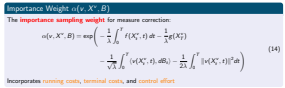
Importance Weight $\alpha(v, X^v, B)$

The **importance sampling weight** for measure correction:

$$\alpha(v, X^v, B) = \exp\Bigg(-\frac{1}{\lambda}\int_0^T f(X_t^v, t)\, dt - \frac{1}{\lambda} g(X_T^v) - \frac{1}{\sqrt{\lambda}}\int_0^T \langle v(X_t^v, t), dB_t\rangle - \frac{1}{2\lambda}\int_0^T \|v(X_t^v, t)\|^2 dt \Bigg) \tag{14}$$

Incorporates running costs, terminal costs, and control effort

Importance Weight (1/2)



Reparameterization Matrix M_t: This matrix enables path-wise reparameterization, a technique to reduce variance in gradient estimation by reparameterizing the stochastic process. It's optimized jointly with the control u to minimize the overall SOCM loss.

Importance Weight α: This exponential weight corrects for the mismatch between the learned control measure and the optimal control measure. However, it can have high variance when costs are large or in high dimensions, which is the main limitation of SOCM.

Bias-Variance Decomposition of the SOCM Loss I

Key Insight: Clean Separation

The SOCM loss decomposes cleanly into two independent components:

- **Bias term:** Depends only on control u, minimized at u^*
- **Variance term:** Depends only on reparameterization matrices M_t

SOCM Loss Decomposition (Proposition 2)

$$\underbrace{\mathcal{L}_{SOCM}(u, M)}_{\text{what we minimize}} = \underbrace{\mathbb{E}\left[\frac{1}{T}\int_0^T \|u(X_t^{u^*}, t) - u^*(X_t^{u^*}, t)\|^2 dt\, e^{-V(X^{u^*},0)/\lambda}\right]}_{\text{Bias of } u \ (= \text{ CE landscape})} + \underbrace{\text{CondVar}(w; M)}_{\text{Variance of } w \text{ (tuned by } M)} \tag{15}$$

Bias-Variance Decomposition of the SOCM Loss II

Same Optimization Landscape in u (Lemma 2)

The **bias term** in SOCM is **exactly** the cross-entropy loss landscape:

- **Same optimum:** Both minimize at u^*
- **Same landscape:** Identical curvature and critical points in u
- **Different formulation:** SOCM uses least-squares regression to target w

What M_t Does

- **Variance reduction only:** Does not change the optimum u^*
- **Stabilizes gradients:** Reduces noise in the matching target w
- **Improves convergence:** Better sample efficiency and stability

What M_t Does NOT Do

- **Does not change bias:** u^* remains the same
- **Does not alter optimum:** Same solution as CE methods
- **Pure variance control:** Only affects the conditional variance $\mathrm{CondVar}(w; M)$

Bias-Variance Decomposition of the SOCM Loss III

SOCM = Cross-Entropy + Variance Reduction

Practical Interpretation:

- SOCM keeps the **same objective** and **optimum** as cross-entropy methods
- But reformulates it as a **least-squares regression** problem: $\min_u \mathbb{E}[\|u - w\|^2]$
- Introduces explicit **variance reduction** via learnable matrices M_t
- Results in **better gradients**, **faster convergence**, and **more stable training**

Bias-Variance Decomposition of the SOCM Loss IV

Why This Matters

This decomposition explains SOCM's empirical success:

1. **Theoretical equivalence:** Same optimum as proven CE methods
2. **Practical advantage:** Explicit variance control improves optimization
3. **Best of both worlds:** CE's guarantees + diffusion models' stability

SOCM Algorithm

Algorithm 2 Stochastic Optimal Control Matching (SOCM)

Input: State cost $f(x,t)$, terminal cost $g(x)$, diffusion coeff. $\sigma(t)$, base drift $b(x,t)$, noise level λ, number of iterations N, batch size m, number of time steps K, initial control parameters θ_0, initial matrix parameters ω_0, loss $\mathcal{L}_{\mathrm{SOCM}}$ in (125)

for $n \in \{0, \dots, N-1\}$ **do**

- Simulate m trajectories of the process X^v controlled by $v = u_{\theta_n}$, e.g., using Euler-Maruyama updates
- Detach the m trajectories from the computational graph, so that gradients do not backpropagate
- Using the m trajectories, compute an m-sample Monte-Carlo approximation $\hat{\mathcal{L}}_{\mathrm{SOCM}}(u_{\theta_n}, M_{\omega_n})$ of the loss $\mathcal{L}_{\mathrm{SOCM}}(u_{\theta_n}, M_{\omega_n})$ in (125)
- Compute the gradients $\nabla_{(\theta,\omega)} \hat{\mathcal{L}}_{\mathrm{SOCM}}(u_{\theta_n}, M_{\omega_n})$ of $\hat{\mathcal{L}}_{\mathrm{SOCM}}(u_{\theta_n}, M_{\omega_n})$ at (θ_n, ω_n)
- Obtain θ_{n+1}, ω_{n+1} with via an Adam update on θ_n, ω_n, resp.

end

Output: Learned control u_{θ_N}

Figure: Stochastic Optimal Control Matching (SOCM) Algorithm

Ornstein-Uhlenbeck Process Benchmarks I

What's an Ornstein-Uhlenbeck (OU) Process?

A **mean-reverting** stochastic differential equation with linear drift and Gaussian noise:

$$dX_t = \theta(\mu - X_t)dt + \sigma dB_t \quad \text{(1D case)}$$

For multi-dimensional benchmarks: $dX_t = Axdt + \sigma dB_t$ with linear drift $b(x,t) = Ax$

Quadratic OU (LQR Class)

Dynamics: OU with quadratic costs

$$f(x,t) = x^T Px, \quad g(x) = x^T Qx$$

Key Feature: This is a **Linear-Quadratic Regulator** problem

Optimal Control (Known):

Linear OU

Dynamics: Linear drift $b(x,t) = Ax$

Costs:

- No running cost: $f = 0$
- Linear terminal: $g(x) = \langle \gamma, x \rangle$

Optimal Control: Also available in **closed**

Ornstein-Uhlenbeck Process Benchmarks II

Double-Well (Challenging Nonlinear Case)

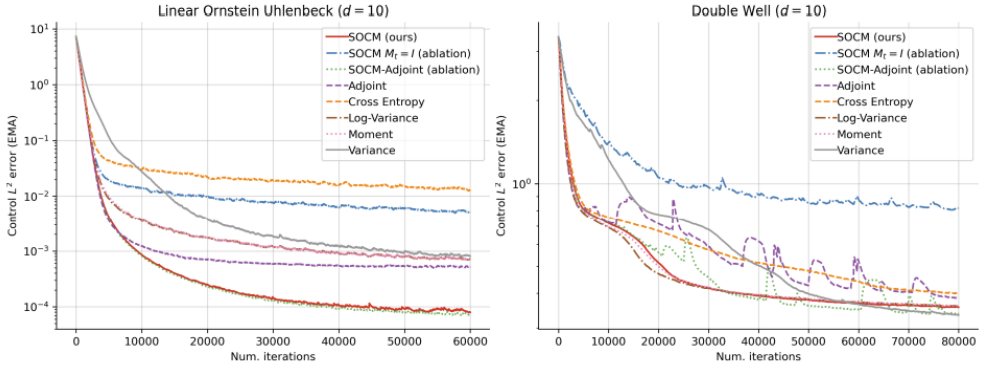
Nonlinear drift from quartic potential: $\Psi(x) = \sum_i \kappa_i (x_i^2 - 1)^2$
Multimodal terminal cost: $g(x)$ has 2^d modes (exponentially many!)
Highly challenging: Tests algorithm stability in complex, multimodal landscapes

Ornstein-Uhlenbeck Process Benchmarks III

Why OU-type Problems are Standard SOC Testbeds

- **Ground truth available:** Analytic u^* for LQR/linear cases $\rightarrow$ direct L_2 control error measurement
- **Controlled difficulty:** Can dial complexity via matrix scaling (A, P, Q parameters)
- **Stress test variance:** Reveals importance weight and gradient signal-to-noise issues
- **Multimodal landscapes:** Double-Well tests stability in challenging optimization scenarios

Experimental Results (1/2)



Experimental Results (2/2)

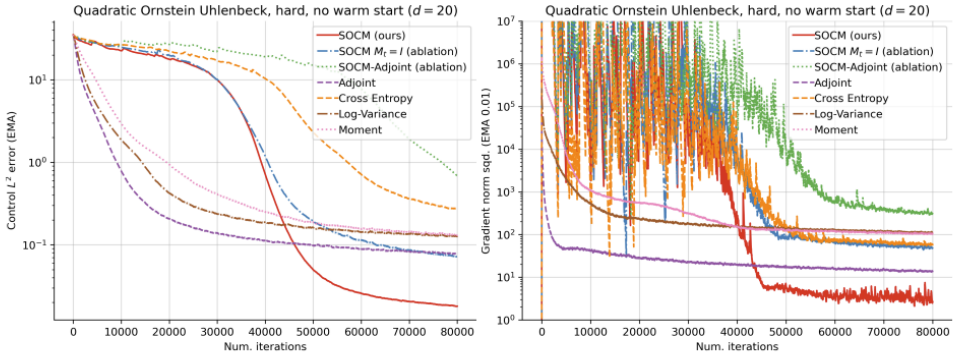


Figure 3 Plots of the L^2 error incurred by the learned control (*top*), and the norm squared of the gradient with respect to the parameters θ of the control (*bottom*), for the QUADRATIC ORNSTEIN UHLENBECK (HARD) setting and for each IDO loss. All the algorithms use a warm-started control (see Appendix D).

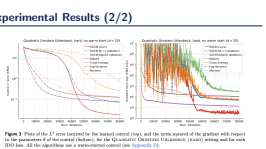


Figure 1 Plots of the L^2 error incurred by the learned control (top), and the norm squared of the gradient with respect to the parameters θ of the control (bottom), for the Q1-scholastic Ornstein Uhlenbeck (train) setting and for each IDO loss. All the algorithms use a warm-started control (see Appendix D).

At the end of training, SOCM obtains the lowest L2 error, improving over all existing methods by a factor of around ten. The two SOCM ablations come in second and third by a substantial difference, which underlines the importance of the path-wise reparameterization trick.

JE DOIS COMPRENDRE CE QUE EST UN ORNSTEIN UHLENBECK PROCESS

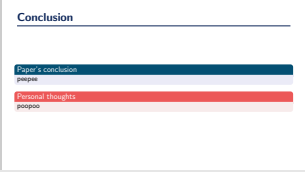
Conclusion

Paper's conclusion

peepee

Personal thoughts

poopoo



The main roadblock when we try to apply SOCM to more challenging problems is that the variance of the factor alpha(v, Xv, B) explodes when f and/or g are large, or when the dimension d is high. The control L2 error for the SOCM and cross-entropy losses remains high and fluctuates heavily due to the large variance of alpha The large variance of alpha is due to the mismatch between the probability measures induced by the learned control and the optimal control. Similar problems are encountered in out-of-distribution generalization for reinforcement learning, and some approaches may be carried over from that area (Munos et al., 2016).

References