Stochastic Optimal Control Matching

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

Thomas Mousseau

August 28, 2025

Overview

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.

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.
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

SOC Connection

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

This bridges the gap between:

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

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: $u(X_t^u, t) \in \mathbb{R}^d$ (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)

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)

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	Forward noising process	No natural drift (learnable)
Drift $b(X_t, t)$	Vehicle dynamics	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

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.

Solve SOC Problems in Low Dimension

Classical Approach: Hamilton-Jacobi-Bellman PDE

In small state spaces, discretize the state space and solve the HJB PDE directly. After applying dynamic programming and Bellman's optimality principle we get:

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

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 $d > 3$

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

Similar Trend in Generative Modeling

The Same Pattern: Non-convex $\longrightarrow$ 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

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**

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)

$v : \mathbb{R}^d$ 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

Presenting Matching Vector Field

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 (1/3) - 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 (1/3) - Path Integral II

We introduce a small step h so we can expand the cost-to-go function with dynamic programming recursion. Then using Itô's lemma (the stochastic version of the chain rule) this expansion produces the gradient and Hessian terms in the HJB. Shrinking h to zero converts the global Bellman principle into a local PDE.

$$V(x,t) = \min_u \mathbb{E}\left[\int_t^{t+h} \left(\frac{1}{2}\|u_s\|^2 + f(X_s,s)\right) ds + V(X_{t+h}, t+h) \,\middle|\, X_t = x\right]$$

Based on Itô's lemma

Expand $V(X_{t+h}, t+h)$ using Itô's lemma:

$$dV = \partial_t V\, ds + \nabla V^\top dX_s + \frac{1}{2}\mathrm{tr}[\sigma\sigma^\top \nabla^2 V]\lambda\, ds.$$

Matching Vector Field (1/3) - Path Integral III

Substituting $dX_s = b\, ds + \sigma u\, ds + \sqrt{\lambda}\sigma\, dB_s$, and taking expectation (the Brownian term vanishes):

$$\mathbb{E}[dV] = \left[\partial_t V + \nabla V \cdot (b + \sigma u) + \frac{\lambda}{2}\mathrm{tr}(\sigma^\top \nabla^2 V)\right] ds.$$

The expected increment equation plugged into the dynamic programming principle gave you the HJB PDE

$$\frac{\partial V}{\partial t} + f(x,t) + \nabla V \cdot (b + \sigma u) + \frac{\lambda}{2}\mathrm{tr}(\sigma^\top \nabla^2 V \sigma) + \frac{1}{2}\|u\|^2 = 0$$

Minimizing this gives us the **optimal control**:

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

Matching Vector Field (1/3) - Path Integral IV

And the reduced HJB using u^* is:

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

The HJB is nonlinear because of the quadratic gradient term. To linearize it, we do the Cole–Hopf transformation:

$$V(x, t) = -\lambda \log \phi(x, t).$$

Matching Vector Field (1/3) - Path Integral V

Plugging into the reduced HJB cancels the $-\frac{1}{2}\|\sigma^\top \nabla V\|^2$ term and gives a linear PDE for ϕ:

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

Here L is the infinitesimal generator of the uncontrolled diffusion:

$$L\phi = b \cdot \nabla \phi + \frac{\lambda}{2} \mathrm{tr}(\sigma\sigma^\top \nabla^2 \phi).$$

The Feynman–Kac theorem says: the solution to this linear PDE equals an expectation over uncontrolled trajectories:

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

Matching Vector Field (1/3) - Path Integral VI

Thus

$$V(x,t) = -\lambda \log \phi(x,t).$$

Remember

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

Substitute $V = -\lambda \log \phi$:

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

Matching Vector Field (1/3) - Path Integral VII

Define the "partition function" (unnormalized density):

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

Then the final path–integral representation is:

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

This is exactly Lemma 1 (Kappen 2005): the optimal control is the score of a

Path-integral Optimal Control

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

Matching Vector Field (1/3) - Path Integral VIII

Why This Matters

Key Achievement: We eliminate V to avoid solving a nonlinear PDE and reframe the control as a **score function** $\nabla_x \log \mathbb{E}[.|X_t = x]$.

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) — Optimal Control I

Step 2 — Rendering u^* Trackable using Pathwise reparameterization (under the uncontrolled law)

Path-integral Optimal Control

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

Reasons Why u^* is not Trackable

Computing this conditional expectation means integrating over the entire path space starting from $X_t = x$, which is intractable in high dimensions.

Matching Vector Field (2/3) — Optimal Control II

Ideal Loss Function

$$\tilde{\mathcal{L}}(u) = \mathbb{E}\left[\frac{1}{T}\int_0^T \|u(X_t, t) - u^*(X_t, t)\|^2 dt\right] \tag{11}$$

Let's expand the squared term $\|u(X_t, t) - u^*(X_t, t)\|^2$:

$$\tilde{\mathcal{L}}(u) = \mathbb{E}\left[\frac{1}{T}\int_0^T \left(\|u(X_t, t)\|^2 - 2\langle u(X_t, t), u^*(X_t, t)\rangle + \|u^*(X_t, t)\|^2\right) dt\right] \tag{12}$$

Matching Vector Field (2/3) — Optimal Control III

The Fundamental Challenge

We cannot compute $u^*(X_t, t)$ directly! This is exactly why we need SOCM's matching approach - to find a computable target w that serves as a proxy for u^*.

Why Focus on the Cross Term?

We concentrate on transforming the cross term $\langle u(X_t, t), u^*(X_t, t)\rangle$ because:

First term $\|u(X_t, t)\|^2$: Straightforward to compute - depends only on our learned control u

Third term $\|u^*(X_t, t)\|^2$: Constant with respect to u - doesn't affect the optimization

Cross term: The **only term** that requires the path-integral formulation since it involves both u and u^*

Matching Vector Field (2/3) — Optimal Control IV

Using the path-integral representation from Step 1, we can transform the problematic cross term:

Reminder: Path-integral Optimal Control

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

$$\begin{aligned}&\mathbb{E}\left[\frac{1}{T}\int_0^T \langle u(X_t,t), u^*(X_t,t)\rangle dt \exp\left(-\lambda^{-1}\int_0^T f(X_t,t)dt - \lambda^{-1}g(X_T)\right)\right] \\ &= \lambda\mathbb{E}\left[\frac{1}{T}\int_0^T \left\langle u(X_t,t), \sigma(t)^T\nabla_x\mathbb{E}\left[\exp\left(-\lambda^{-1}\int_t^T f(X_s,s)ds - \lambda^{-1}g(X_T)\right)\middle| X_t = x\right]\right\rangle\right] \end{aligned} \tag{13}$$

Matching Vector Field (2/3) — Optimal Control V

Next Challenge

The transformed cross term still contains an intractable expectation. This is where path-wise reparameterization trick (novel approach introduced by this paper) becomes essential to make this computable.

The Challenge: Intractable Conditional Expectation

From Step 2, we need to make this computable for training:

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

Matching Vector Field (2/3) — Optimal Control (continued) I

Fréchet-differentiable functional

In functional analysis, we often deal with functionals: mappings from a space of functions or paths into $\mathbb{R}$.
Here F takes a whole path $X = (X_s)_{s\in[0,T]}$ and outputs a number (total cost).

$$\nabla_x \mathbb{E}[e^{-F(X)}|X_0 = x] = \mathbb{E}\left[\left(-\nabla_z F(X+\psi)\big|_{z=0} + \cdots\right) e^{-F(X)} \,\middle|\, X_0 = x\right].$$

The Fréchet derivative $\nabla_z F(X+\psi)$ expresses how F changes under a small path shift ψ. This is what allows you to write the gradient of an expectation in terms of expectations over pathwise terms, instead of REINFORCE-style score-function estimators.

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

Matching Vector Field (2/3) — Optimal Control (continued) II

SOCM's Specific Choice

We assume our cost functional $F(X)$ is Fréchet differentiable, meaning it's smooth with respect to small perturbations of the path. This technical assumption lets us compute pathwise gradients instead of relying on high-variance score-function estimators.

The small perturbations are modeled by a matrix $M_t(s)$ which becomes a learnable parameter for variance reduction!

Matching Vector Field (2/3) — Optimal Control (continued) III

Path-wise Reparameterization Solution

Transform the intractable gradient into a computable expectation:

$$\begin{aligned}\nabla_x \mathbb{E}\left[\exp\left(-\lambda^{-1}\int_t^T f(X_s,s)ds - \lambda^{-1}g(X_T)\right)\middle| X_t = x\right] \\ = \mathbb{E}\Bigg[\Bigg(-\lambda^{-1}\int_t^T M_t(s)\nabla_x f(X_s,s)ds - \lambda^{-1}M_t(T)\nabla g(X_T) \\ +\lambda^{-1/2}\int_t^T (M_t(s)\nabla_x b(X_s,s) - \partial_s M_t(s))(\sigma^{-1})^T(s)dB_s\Bigg)\Bigg]\end{aligned} \tag{14}$$

So instead of gradient of expectation over all futures (intractable), you now have expectation of computable pathwise/local terms on the state space (tractable).

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

Step 3 — Girsanov Theorem

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

SOCM Loss in Construction

From Steps 1-2, we have the theoretical loss function:

$$\begin{aligned}\tilde{\mathcal{L}}(u) = \mathbb{E}\Bigg[\frac{1}{T}\int_0^T \Bigg\| u(X_t, t) + \sigma(t)^T \Bigg(\int_t^T M_t(s)\nabla_x f(X_s, s) ds \\ + M_t(T)\nabla g(X_T) \\ - \lambda^{1/2} \int_t^T (M_t(s)\nabla_x b(X_s, s) \\ - \partial_s M_t(s))(\sigma^{-1})^T(s) dB_s \Bigg)\Bigg\|^2 dt\Bigg] \end{aligned} \quad (15)$$

Problem: This expectation is under the **uncontrolled** measure, but we need to sample using an **arbitrary control** v for practical training.

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

Girsanov Measure Change

By changing the drift of the system, we need to account for this by using the Girsanov theorem.

Introduces importance weight:

$$\alpha(v, X^v, B) = \exp\left(-\frac{1}{\sqrt{\lambda}}\int_0^T \langle v, dB_t\rangle - \frac{1}{2\lambda}\int_0^T \|v\|^2 dt\right)$$

Before Girsanov: the loss is under the uncontrolled measure $\rightarrow$ nice math, but intractable for training.

After Girsanov: we can sample under any arbitrary control v, but must reweight

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

Final SOCM Loss

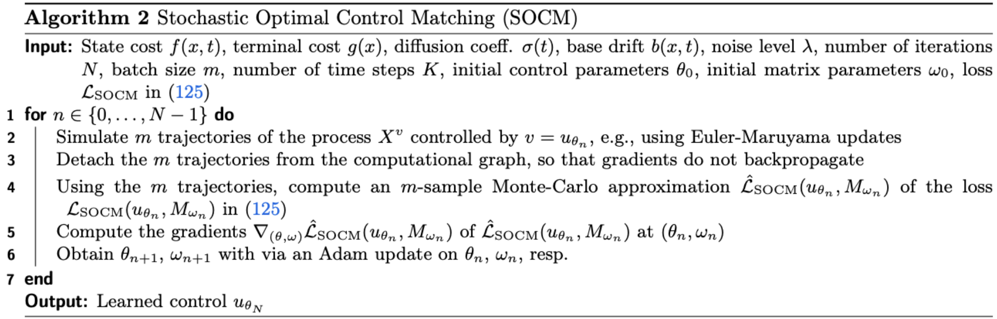
After Girsanov transformation, we get the practical SOCM loss:

$$\mathcal{L}_{SOCM}(u, M) = \mathbb{E}\left[\frac{1}{T}\int_0^T \|u(X_t^v, t) - w(t, v, X^v, B, M_t)\|^2 dt \times \alpha(v, X^v, B)\right] \tag{16}$$

where:

- w is the matching vector field (from the reparameterization terms)
- α is the complete importance weight including costs and control effort
- Trajectories X^v are generated using the practical control v

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 $\lambda$, number of iterations $N$, batch size $m$, number of time steps $K$, initial control parameters $\theta_0$, initial matrix parameters $\omega_0$, loss $\mathcal{L}_{\mathrm{SOCM}}$ in (125)
1 for $n \in \{0, \dots, N-1\}$ do
2     Simulate $m$ trajectories of the process $X^v$ controlled by $v = u_{\theta_n}$, e.g., using Euler-Maruyama updates
3     Detach the $m$ trajectories from the computational graph, so that gradients do not backpropagate
4     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)
5     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 $(\theta_n, \omega_n)$
6     Obtain $\theta_{n+1}$, $\omega_{n+1}$ with via an Adam update on $\theta_n$, $\omega_n$, resp.
7 end
Output: Learned control $u_{\theta_N}$
```

Figure: Stochastic Optimal Control Matching (SOCM) Algorithm

Ornstein-Uhlenbeck Process Benchmarks I

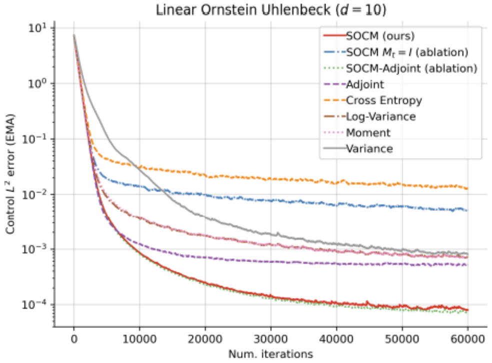
What's an Ornstein-Uhlenbeck (OU) Process?

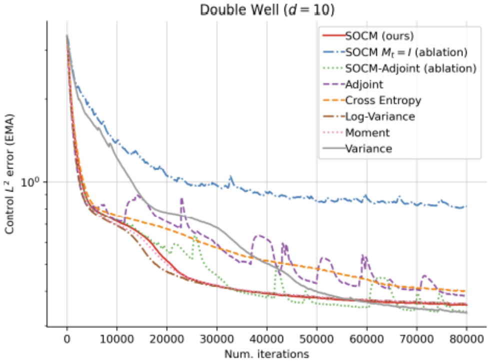
A **mean-reverting** stochastic differential equation with linear drift and Gaussian noise:

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

The Ornstein-Uhlenbeck process is a classic mean-reverting SDE with both drift and noise, simple enough to solve analytically but rich enough to test stochastic control methods. It serves as the ideal benchmark for evaluating SOCM's bias-variance tradeoff

Experimental Results (1/2)





Experimental Results (2/2)

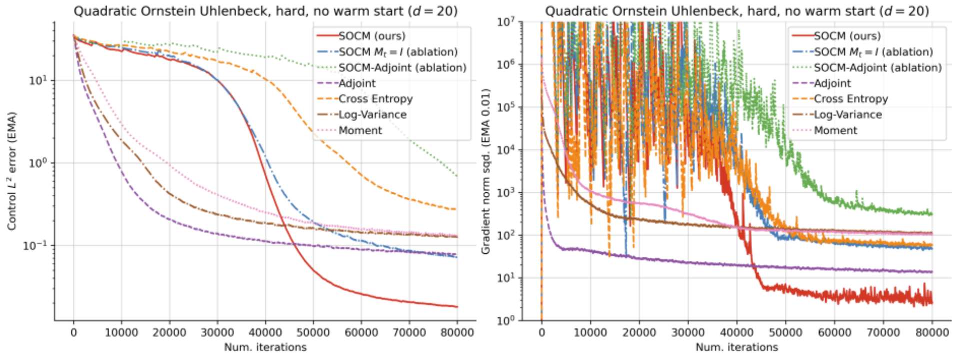


Figure 3 Plots of the L^2 error incurred by the learned control (*top*), and the norm squared of the gradient with respect to the parameters θ of the control (*bottom*), for the QUADRATIC ORNSTEIN UHLENBECK (HARD) setting and for each IDO loss. All the algorithms use a warm-started control (see Appendix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

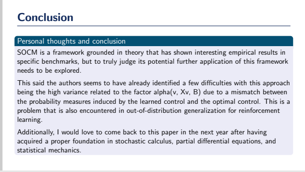
Conclusion

Personal thoughts and conclusion

SOCM is a framework grounded in theory that has shown interesting empirical results in specific benchmarks, but to truly judge its potential further application of this framework needs to be explored.

This said the authors seems to have already identified a few difficulties with this approach being the high variance related to the factor alpha(v, Xv, B) due to a mismatch between the probability measures induced by the learned control and the optimal control. This is a problem that is also encountered in out-of-distribution generalization for reinforcement learning.

Additionally, I would love to come back to this paper in the next year after having acquired a proper foundation in stochastic calculus, partial differential equations, and statistical mechanics.



Conclusion

Personal thoughts and conclusion

SOCM is a framework grounded in theory that has shown interesting empirical results in specific benchmarks, but to truly judge its potential further application of this framework needs to be explored.

This said the authors seems to have already identified a few difficulties with this approach being the high variance related to the factor alpha(v, Xv, B) due to a mismatch between the probability measures induced by the learned control and the optimal control. This is a problem that is also encountered in out-of-distribution generalization for reinforcement learning.

Additionally, I would love to come back to this paper in the next year after having acquired a proper foundation in stochastic calculus, partial differential equations, and statistical mechanics.

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).