# Stochastic Optimal Control Matching

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

Thomas Mousseau

August 25, 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 $p(x)$ where the normalization constant $Z = \int p(x)dx$ is intractable to compute.

## 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)

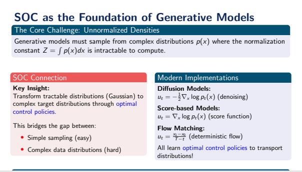
## Modern Implementations

**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?

## Dynamics (SDE)

$$dx_t = \underbrace{u_t dt}_{\text{drift coefficient}} + \underbrace{dw_t}_{\text{diffusion coefficient}} \tag{1}$$

## Cost Function

$$J(u) = \mathbb{E}\left[\int_0^T L(x_t, u_t, t)dt + \Phi(x_T)\right] \tag{2}$$

## Key Components

- **State Process:** $x_t \in \mathbb{R}^d$ (position in state space at time $t$)
- **Control Process:** $u_t \in \mathbb{R}^d$ (action/decision at time $t$)
- **Noise Process:** $w_t$ (random disturbances, typically Brownian motion)
- **Running Cost:** $L(x_t, u_t, t)$ (cost accumulated over time)
- **Terminal Cost:** $\Phi(x_T)$ (cost at the final time $T$)

## What is a Stochastic Control Problem?

**Dynamics (SDE)**

$$dx_t = \underbrace{u_t dt}_{\text{drift coefficient}} + \underbrace{dw_t}_{\text{diffusion coefficient}} \tag{1}$$

**Cost Function**

$$J(u) = \mathbb{E}\left[\int_0^T L(x_t, u_t, t)dt + \Phi(x_T)\right] \tag{2}$$

**Key Components**

- **State Process:** $x_t \in \mathbb{R}^d$ (position in state space at time $t$)
- **Control Process:** $u_t \in \mathbb{R}^d$ (action/decision at time $t$)
- **Noise Process:** $w_t$ (random disturbances, typically Brownian motion)
- **Running Cost:** $L(x_t, u_t, t)$ (cost accumulated over time)
- **Terminal Cost:** $\Phi(x_T)$ (cost at the final time $T$)

A stochastic control problem involves finding an optimal control policy to steer a dynamical system under uncertainty while minimizing the expected cost.

A voir si je rajouter une autre slide qui presente la solution avec HJB + Feynman-Kac approache.

# Classic Approach to Finding Optimal Control

## Optimal Control $u^*$

Find the control policy $u^*$ that minimizes the expected cost: $u^* = \arg\min_u J(u)$

## Hamilton-Jacobi-Bellman (HJB) Equation

The classical approach uses the HJB PDE to characterize the value function $V(x, t)$:

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

With boundary condition: $V(x, T) = \Phi(x)$ (terminal cost)

## The Curse of Dimensionality

Classical numerical methods (finite differences, grid-based) become computationally intractable in high dimensions due to exponential growth in grid size: $\mathcal{O}(N^d)$ where $d$ is dimension.

## Classic Approach to Finding Optimal Control

**Optimal Control $u^*$**

Find the control policy $u^*$ that minimizes the expected cost: $u^* = \arg\min_u J(u)$

**Hamilton-Jacobi-Bellman (HJB) Equation**

The classical approach uses the HJB PDE to characterize the value function $V(x,t)$:

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

With boundary condition: $V(x,T) = \Phi(x)$ (terminal cost)

**The Curse of Dimensionality**

Classical numerical methods (finite differences, grid-based) become computationally intractable in high dimensions due to exponential growth in grid size: $\mathcal{O}(N^d)$ where $d$ is dimension.

**HJB Equation:** The Hamilton-Jacobi-Bellman equation provides the theoretical foundation for solving stochastic optimal control problems by characterizing the value function $V(x,t)$ as the solution to a nonlinear PDE. The optimal control is then $u^*(x,t) = \arg\min_u[\cdots]$ from the HJB equation.

# Neural PDE Solvers for SOC

## Core Innovation: Neural ODEs (Chen et al., 2018)

**Key Insight:** Replace discrete layers with continuous-time ODEs
$\frac{dh(t)}{dt} = f_\theta(h(t), t)$ where $h(t)$ represents hidden states evolving continuously

## Neural Network Approximation

**Value Function:**

$$V(x, t) \approx V_\theta(x, t) \tag{4}$$

**Control Policy:**

$$u(x, t) \approx u_\phi(x, t) \tag{5}$$

Both parameterized by deep neural networks

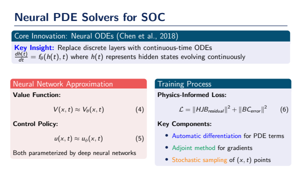
## Training Process

**Physics-Informed Loss:**

$$\mathcal{L} = \|HJB_{residual}\|^2 + \|BC_{error}\|^2 \tag{6}$$

**Key Components:**

- Automatic differentiation for PDE terms
- Adjoint method for gradients
- Stochastic sampling of $(x, t)$ points




**Neural ODEs:** Chen et al. (2018) showed that residual networks can be interpreted as discretizations of ODEs. This insight led to continuous-depth models and, crucially for our context, neural methods for solving differential equations.

**Physics-Informed Neural Networks:** The key is training networks to satisfy the HJB equation through the residual loss, making the physics constraints part of the optimization objective.

# Reasons behind SOCM (1/2)

Many fundamental tasks in machine learning can be naturally cast as stochastic optimal control problems, highlighting the importance of efficient SOC methods.

## 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

# Reasons behind SOCM (2/2)

Current SOC methods suffer from optimization challenges that limit their effectiveness.

### Current SOC Methods

- Use **adjoint methods** (like CNFs)
- Yield **non-convex** function landscapes
- Difficult optimization with local minima
- Unstable training dynamics

### Diffusion Models Success

- Use **least-squares loss**
- Create **convex** functional landscapes
- Stable and reliable optimization
- Excellent empirical performance

### SOCM's Innovation

**Goal:** Develop least-squares loss formulations for SOC problems, combining the expressiveness of stochastic control with the optimization stability of diffusion models.

# 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** |

# Introducing Stochastic Optimal Control Matching

SOCM offers a more principled, stable, and accurate way to learn generative dynamics by blending stochastic control theory with modern matching-based generative modeling.

## Key Novel Contributions

1. **Controlled Stochastic Process:** Views the generation process as a controlled stochastic process bridging a simple distribution to data.
2. **Least-Squares Matching:** Learning the control via least-squares matching, a stable and convex regression objective.
3. **Joint Optimization:** Optimizing control and variance-reducing reparameterization matrices simultaneously, for efficient learning.
4. **Path-wise Reparameterization:** Introducing a path-wise reparameterization trick, boosting gradient estimation quality.

# The SOCM Framework

## SOCM Loss Function

The Stochastic Optimal Control Matching objective is defined as:

$$\mathcal{L}_{SOCM}(u, M) := \mathbb{E}\left[\frac{1}{T}\int_0^T \|u(X_t^v, t) - w(t, v, X^v, B, M_t)\|^2 dt \times \alpha(v, X^v, B)\right] \tag{7}$$

## Where:

- $X^v$ is the process controlled by $v$:

$$dX_t^v = (b(X_t^v, t) + \sigma(t)v(X_t^v, t))dt + \sqrt{\lambda}\sigma(t)dB_t, \text{ with } X_0^v \sim p_0 \tag{8}$$

- $u(X_t^v, t)$ is the control policy being learned
- $w(t, v, X^v, B, M_t)$ is the target matching function
- $\alpha(v, X^v, B)$ is a weighting function

## The SOCM Framework

**SOCM Loss Function**

The Stochastic Optimal Control Matching objective is defined as:

$$\mathcal{L}_{SOCM}(u, M) := \mathbb{E}\left[\frac{1}{T}\int_0^T \|u(X_t^v, t) - w(t, v, X^v, B, M_t)\|^2 dt \times \alpha(v, X^v, B)\right] \quad (7)$$

**Where:**

- $X^v$ is the process controlled by $v$:

$$dX_t^v = (b(X_t^v, t) + \sigma(t)v(X_t^v, t))dt + \sqrt{\lambda}\sigma(t)dB_t, \text{ with } X_0^v \sim p_0 \quad (8)$$

- $u(X_t^v, t)$ is the control policy being learned
- $w(t, v, X^v, B, M_t)$ is the target matching function
- $\alpha(v, X^v, B)$ is a weighting function

Ici, je dois presenter en details la matrice Mt, B ainsi que w. De plus, je veux expliquer l'intuition du path-wise reparameterization trick, a novel technique to obtain low-variance estimates of the gradient of the conditional expectation of a functional of a random process with respect to its initial value

# 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)

**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 $(\theta_n, \omega_n)$

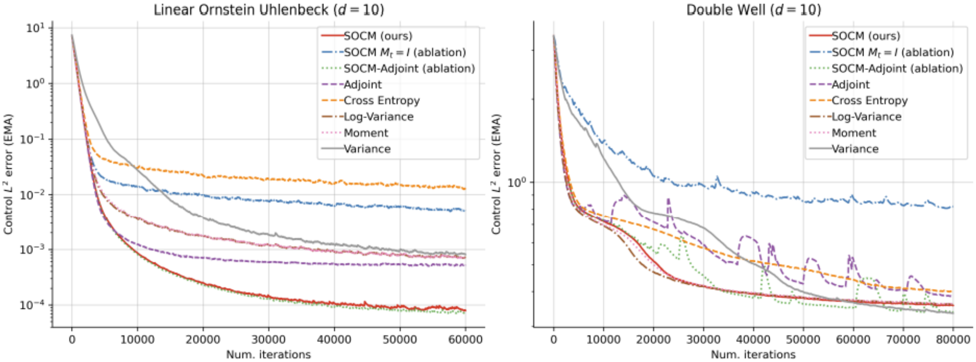
  Obtain $\theta_{n+1}$, $\omega_{n+1}$ with via an Adam update on $\theta_n$, $\omega_n$, resp.

**end**

**Output:** Learned control $u_{\theta_N}$

Figure: Stochastic Optimal Control Matching (SOCM) Algorithm

# Experimental Results (1/2)



# Experimental Results (2/2)

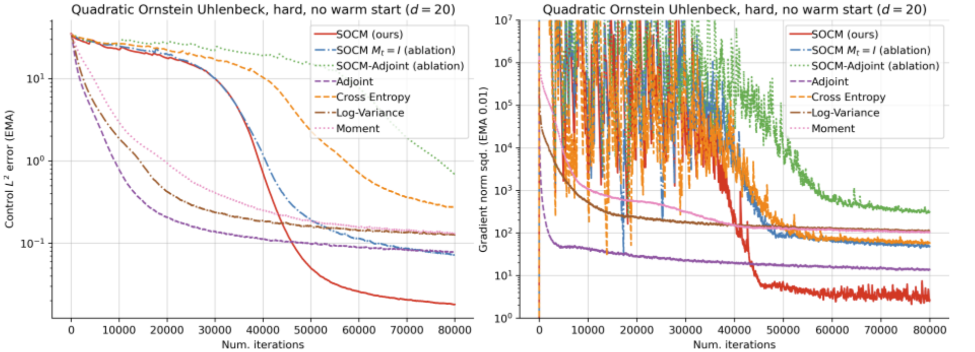



**Figure 3** Plots of the $L^2$ error incurred by the learned control (*top*), and the norm squared of the gradient with respect to the parameters $\theta$ of the control (*bottom*), for the QUADRATIC ORNSTEIN UHLENBECK (HARD) setting and for each IDO loss. All the algorithms use a warm-started control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

## Experimental Results (2/2)

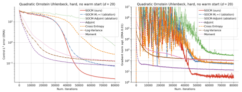



**Figure 3** Plots of the $L^2$ error incurred by the learned control (*top*), and the norm squared of the gradient with respect to the parameters $\theta$ of the control (*bottom*), for the Quadratic Ornstein Uhlenbeck (hard) setting and for each IDO loss. All the algorithms use a warm-started control (see Appendix D).

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

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