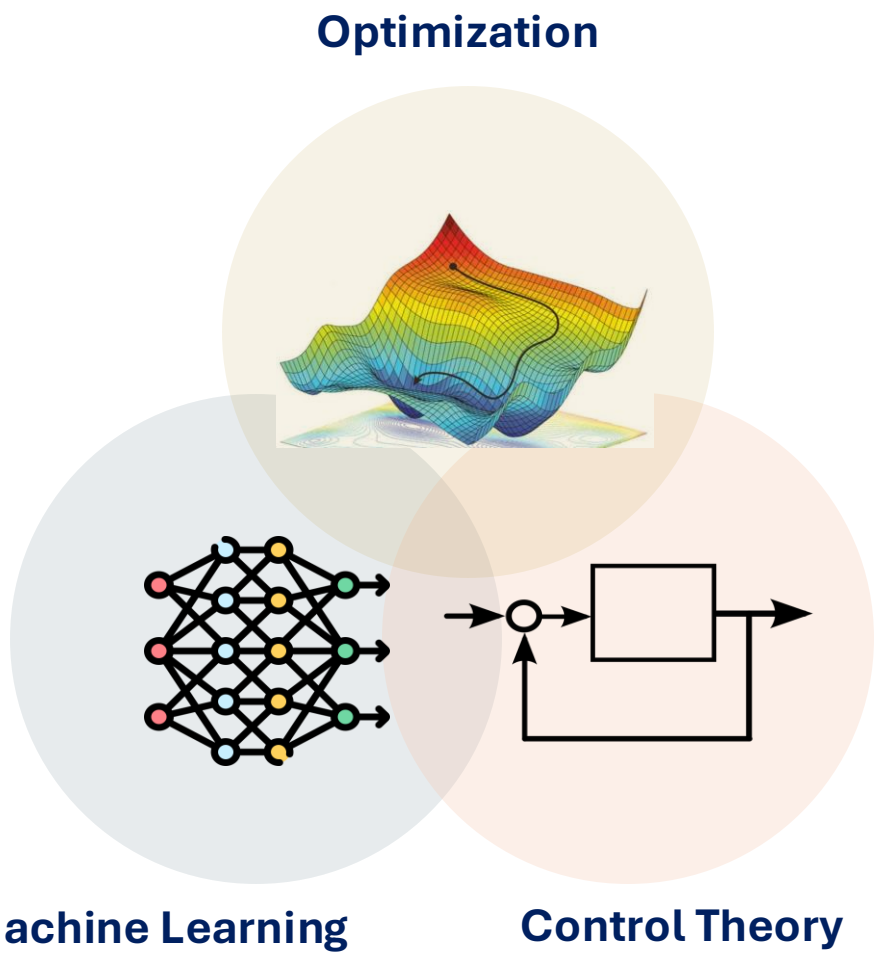
Fast Optimization via Flow Models

Augustinos D. Saravanos

Postdoctoral Researcher
Postdoc Advisor: Prof. Chuchu Fan
Reliable Autonomous Systems (REALM) Lab
MIT

JST ASPIRE “AI-Physical Systems” Kick-Off Meeting
March 2026

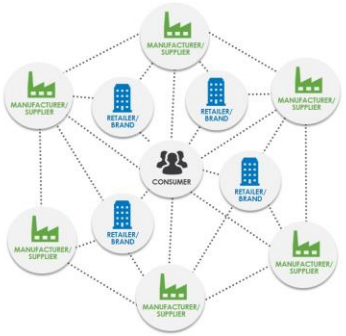
Research Scope & Background



Scalable and Reliable Decision-Making

Autonomy & Robotics

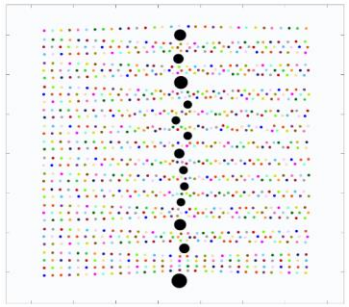
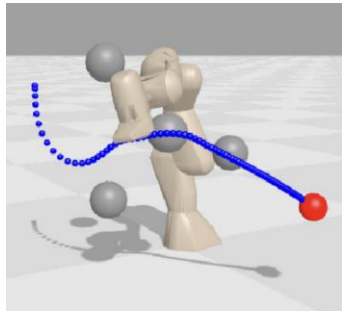
Transportation



Supply Chains

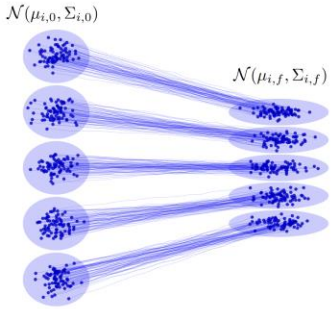
Power Systems

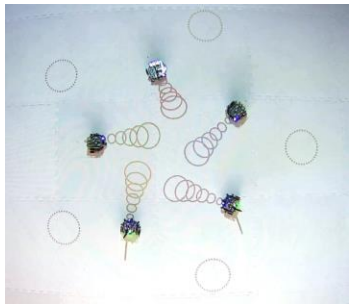
Research Scope & Background

Scalable Trajectory Optimization

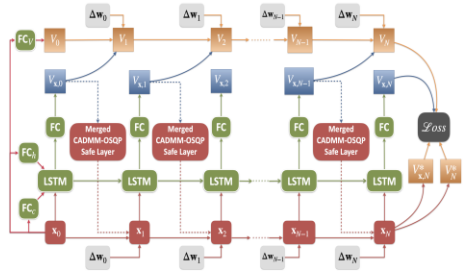
[T-RO 2023, IJRR 2026]

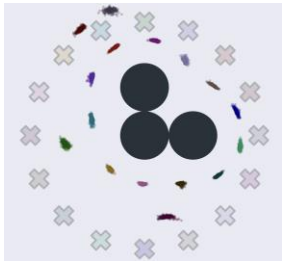




Distributional Control

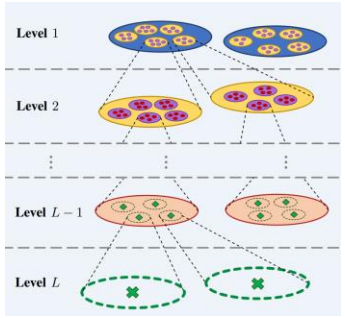
[RSS 2021, IROS 2024, TAC 2026 (UR)]

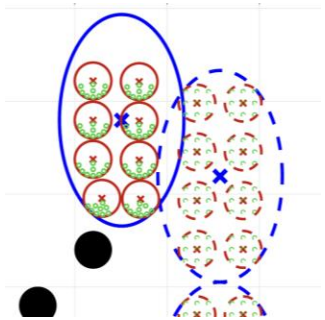




Deep Stochastic Control

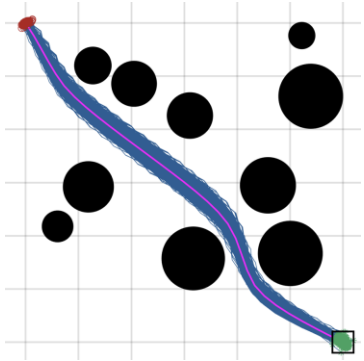
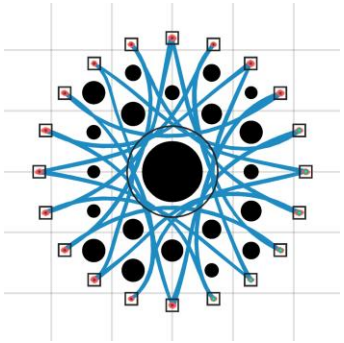
[RSS 2022]





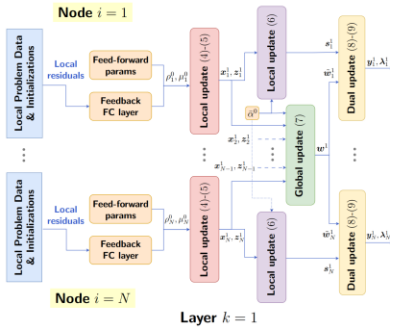
Hierarchical Optimization

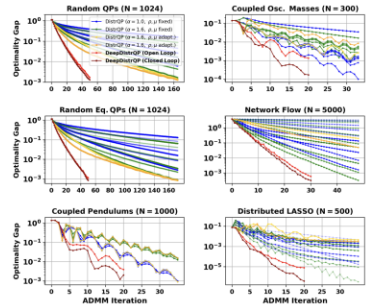
[RSS 2023]

Robust Optimization

[ACC 2025, CDC 2025, TAC 2026 (UR)]





Learning-to-Optimize

[ICLR 2025, ICLR 2026, TPAMI 2026 (UR)]

How to Optimize?

$$\min_{x \in \mathbb{R}^n} f(x)$$

Highly non-convex
High-dimensional

How can we blend the best of both worlds?

Very hard to design **hand-crafted optimization rules** that **work well in general.**

Highly dependent on the **geometry of class of problems** we are interested in.

Gradient Descent, Newton, Momentum Methods
QP/CP, Interior Point, iLQR/DDP etc.

✓ Local derivative information
✗ Prone to local minima

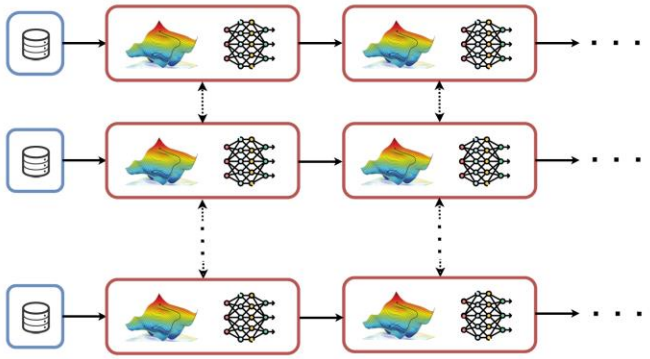
Cross-Entropy Method, Evolutionary Strategies,
Stochastic Search, MPPI, etc.

✓ Can avoid local minima
✗ Can't scale to high dimensions

Learning-to-Optimize via Deep Unfolded Flows
Augustinos D. Saravanos, Oswin So, H. M. Sabbir Ahmad and Chuchu Fan
Under Review at ICML 2026

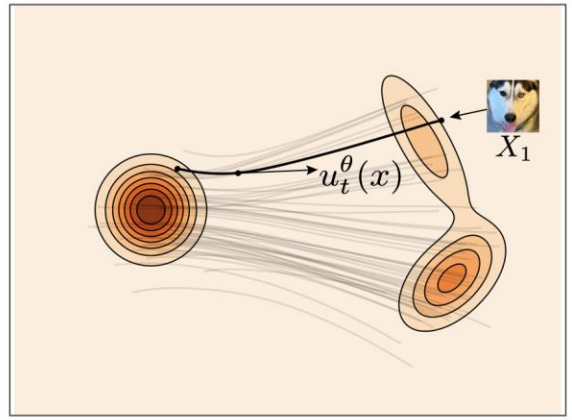
A Machine Learning Perspective

Learning-to-Optimize



Learn optimization updates or embed learnable components in optimizers

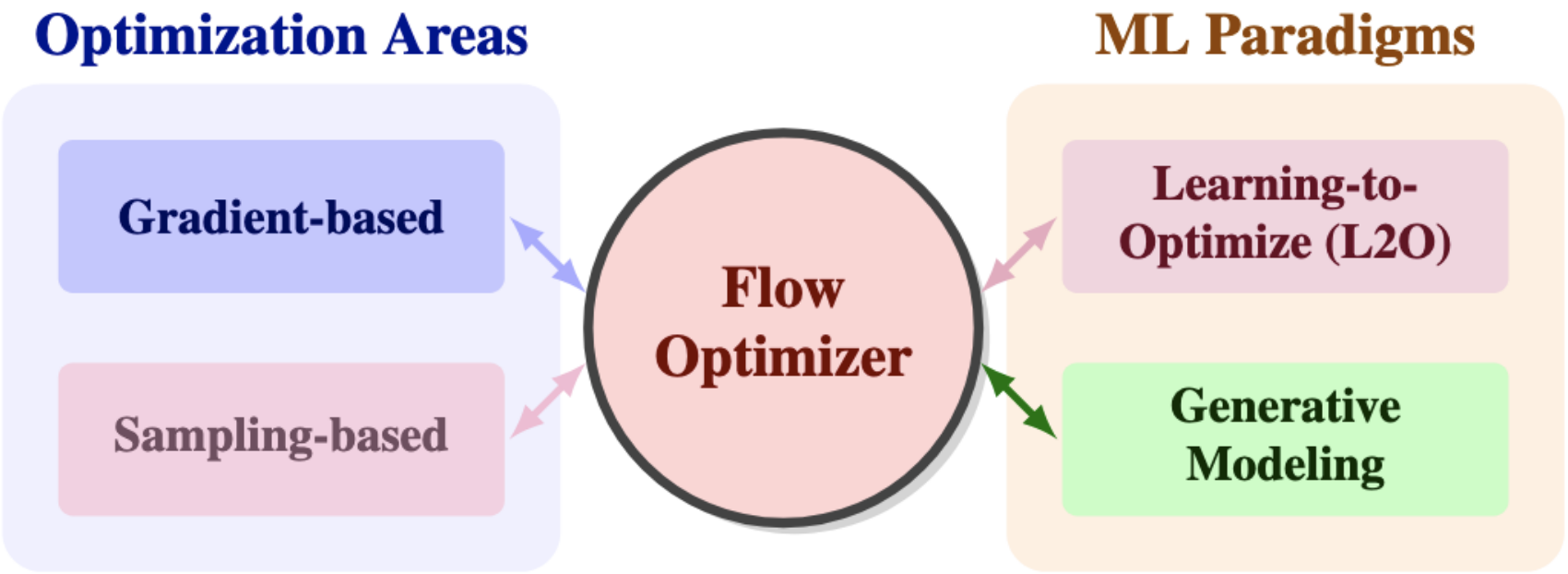
Generative Modeling



Learn transports from source to target distributions

Key insight: From a sampling-based optimization perspective, we can interpret strong optimization updates as learning transport maps to low-cost regions of the optimization landscape!

Conceptual Overview



Flow Model as an Iterative Optimizer

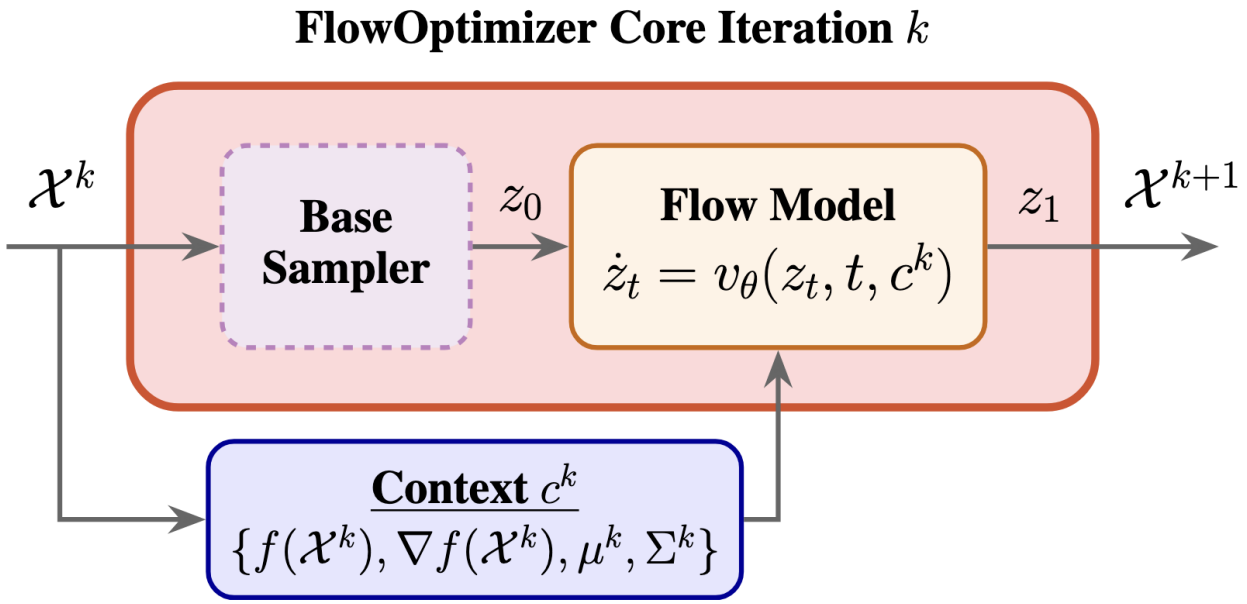
Learn a mapping

$$\mathcal{X}^{k+1} = \mathcal{T}(\mathcal{X}^k, c^k)$$

Next population

Current population

Context



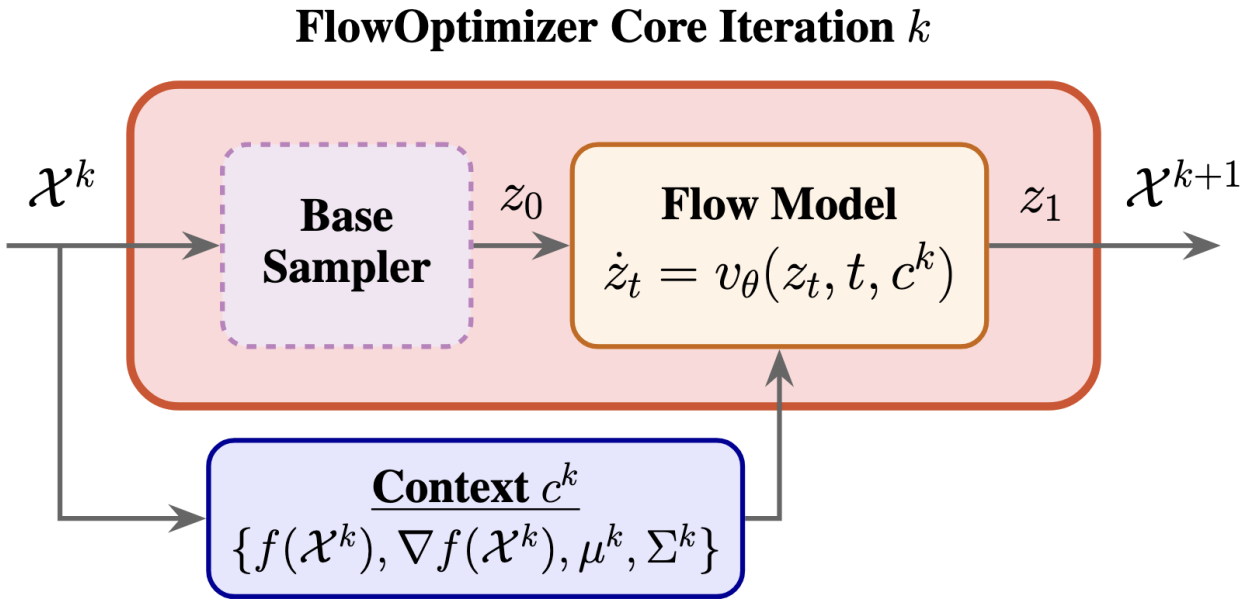
Represent as a flow model

$$\dot{z}_t = v_\theta(z_t, t, c^k), \quad z_{t=0} = \mathcal{X}^k$$

Context

Gradient information, population statistics, etc.

Pre-Training via Flow Matching



Construct pairs of populations

Sample base population points and improved population points

$$(z_0, z_1)$$

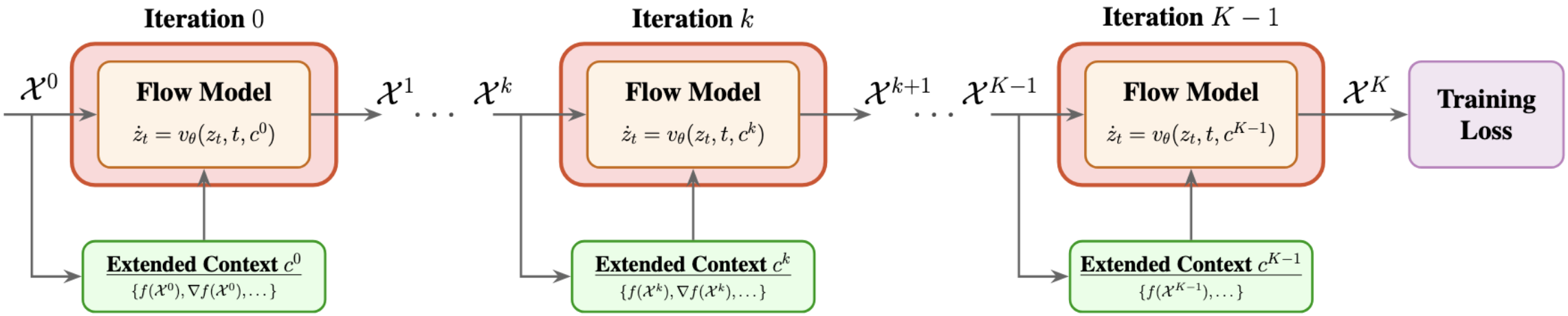
Train velocity field

Use flow matching loss to match population displacements

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{t,z_0,z_1} \left\| v_\theta(z_t, t, c) - \Delta z \right\|_2^2$$

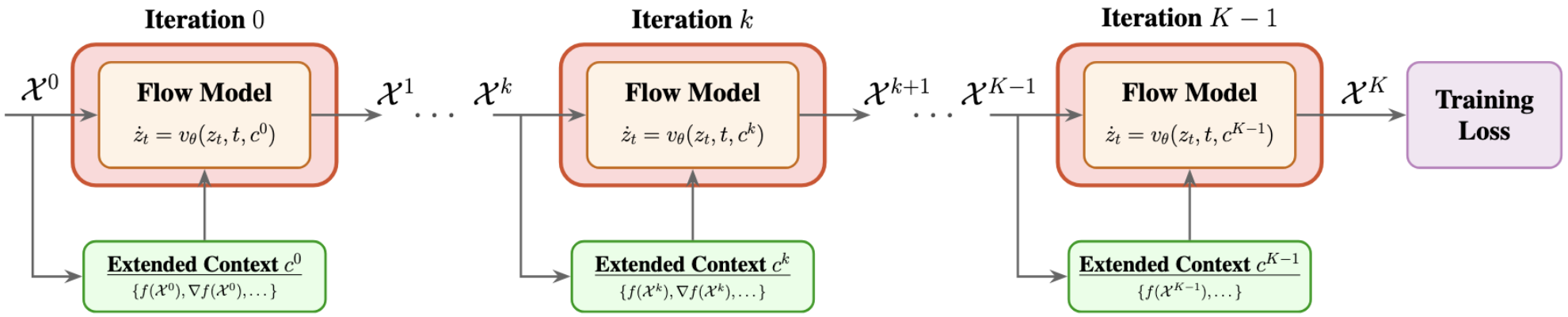
Deep Unfolded Flows

Unroll flow models as sequential iterations/layers of a deep learning architecture



Use extended context including history of best points, population statistics history, etc.

Fine-Tuning Phase via Objective Minimization



We fine-tune the deep unfolded optimizer directly minimizing the objective function values

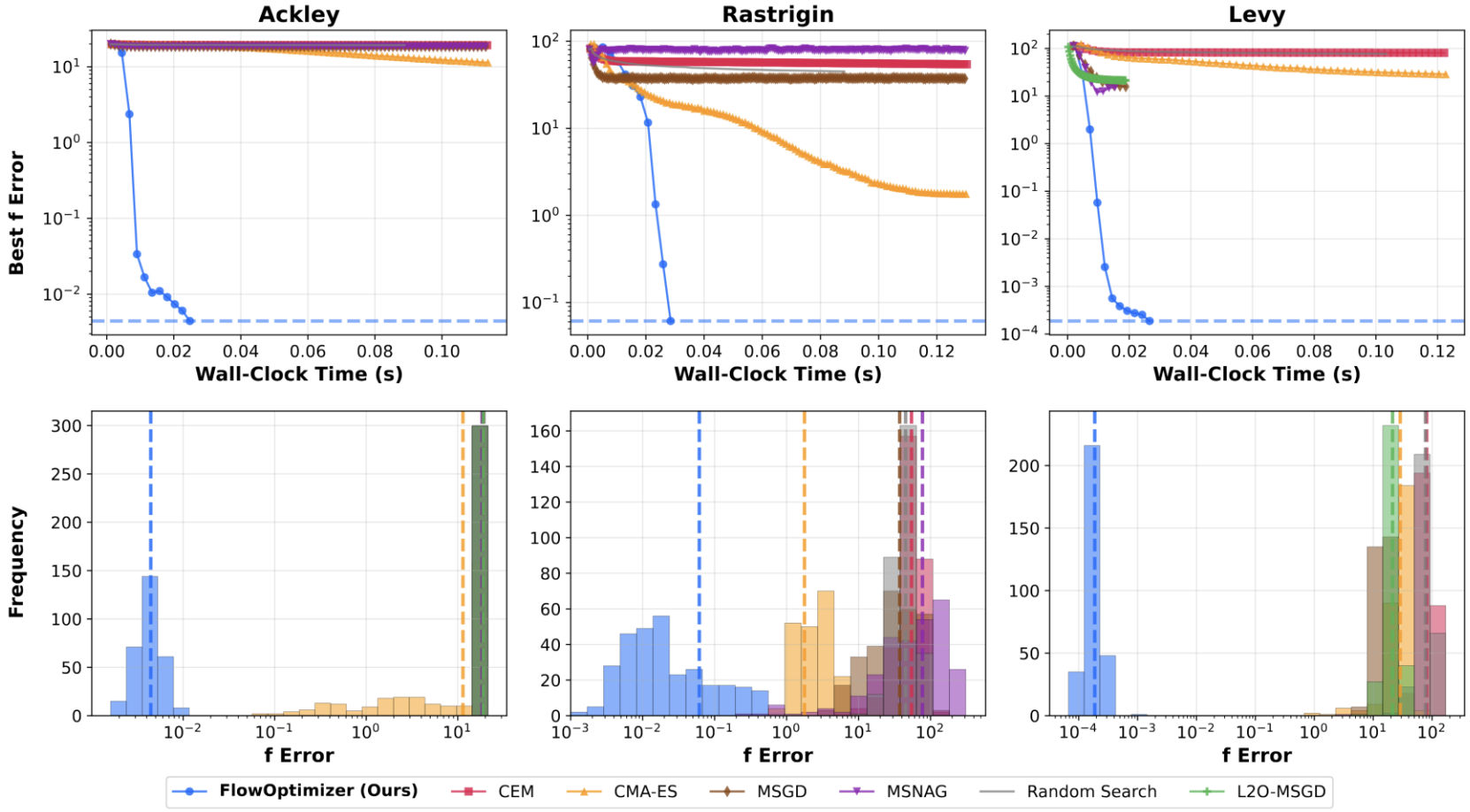
Fine-tuning training loss $$\mathcal{L}_{\mathrm{FT}} = \sum_{k=1}^{K} w_k \ell_k(\mathcal{X}^k)$$

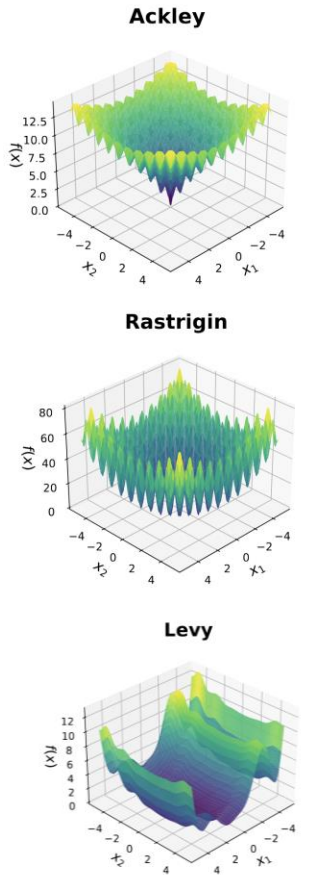
$$\ell_k(\mathcal{X}^k) = \alpha \mathrm{Best}(f(\mathcal{X}_k)) + (1-\alpha)\mathrm{Mean}(f(\mathcal{X}_k))$$

Training happens in a **self-supervised manner**
(we don't require known solutions!)

Optimization Benchmarks

Evaluation on non-convex optimization benchmarks: Train and test on 20D problems.
FlowOptimizer significantly outperforms classical/learned gradient and sampling-based optimizers!





Real-World Problems

Applying on real-world problems:
1) Robotic Arm Inverse Kinematics,
2) Power Grid Economic Load Dispatch
3) Supply Chain Multi-Source Weber Problem

Again, **FlowOptimizer** outperforms classical/learned optimization baselines!

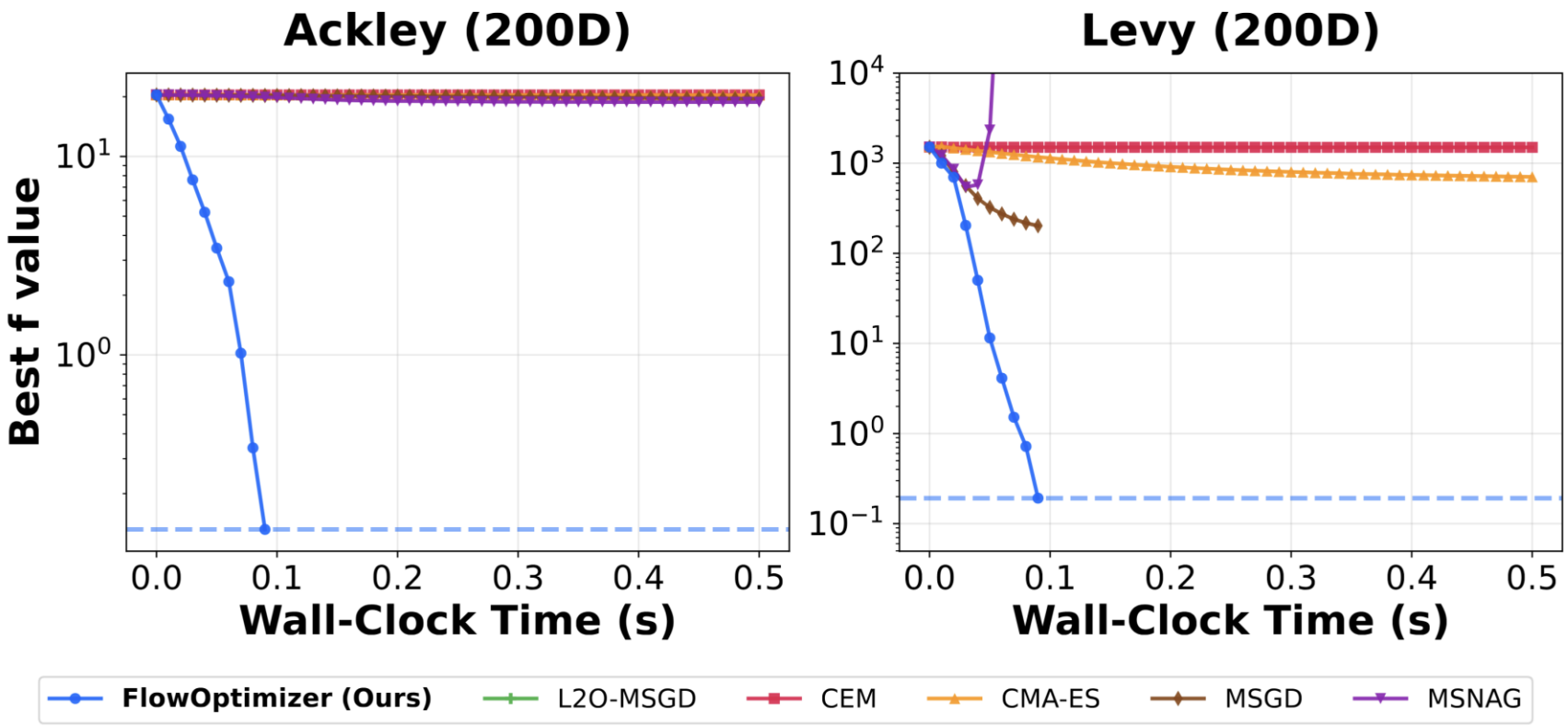
Problem Class	Dim. n	RS	CEM	CMA-ES	MSGD	MSNAG	L2O-GD	**FlowOpt (ours)**
Robotic Arm	10	0.18	0.09	0.008	0.014	0.021	0.012	**0.003**
	30	0.31	0.18	0.04	0.18	0.17	0.12	**0.007**
Power Grid	20	0.53	0.14	0.03	0.05	0.03	0.03	**5e-4**
	50	0.83	0.31	0.11	0.23	0.17	0.08	**1.2e-3**
Supply Chain	40	0.47	0.09	0.05	0.07	0.03	0.013	**0.007**
	80	0.68	0.14	0.06	0.09	0.03	0.027	**0.013**

Sinha, N., Chakrabarti, R., and Chattopadhyay, P. K. **Evolutionary programming techniques for economic load dispatch**. IEEE Transactions on evolutionary computation, 7(1):83–94, 2003.
Brimberg, J., Hansen, P., Mladenovic, N., and Salhi, S. **A survey of solution methods for the continuous location-allocation problem.** International Journal of Operations Research, 5(1):1–12, 2008.

Generalization to Higher Dimensions

Dimension-Agnostic Variation: Train on 20D problems and test on 200D problems.

Flow models learned in low-dimensions can still be effective in much higher-dimensional problems!



What makes **FlowOptimizer** stronger?

✪ Expressiveness of flow models

Highly expressive parameterization of optimization updates, enabling a rich representation of complex, nonlinear transport maps compared to hand-designed updates.

✪ Learning-to-optimize from problem geometry

Learns geometry patterns of the underlying problem class, such as curvature, multi-modality, and scale, which are difficult to capture with generic optimizers.

✪ Learning population-level interactions

Unlike pointwise update rules, it operates on a population of candidate solutions and explicitly learns interactions among its members, which allows for implicitly coordinating the landscape exploration.

Ongoing/Future Directions

- ✪ **Fast Constrained Optimization via Flows**
- ✪ **Trajectory Optimization / MPC via Flow Optimizers**
- ✪ **Neural Combinatorial Optimization**
- ✪ **Distributed Optimization via Flow Models Coordination**

Thank you! Questions?