Probabilistic Perspective for Interpretable AI Agents

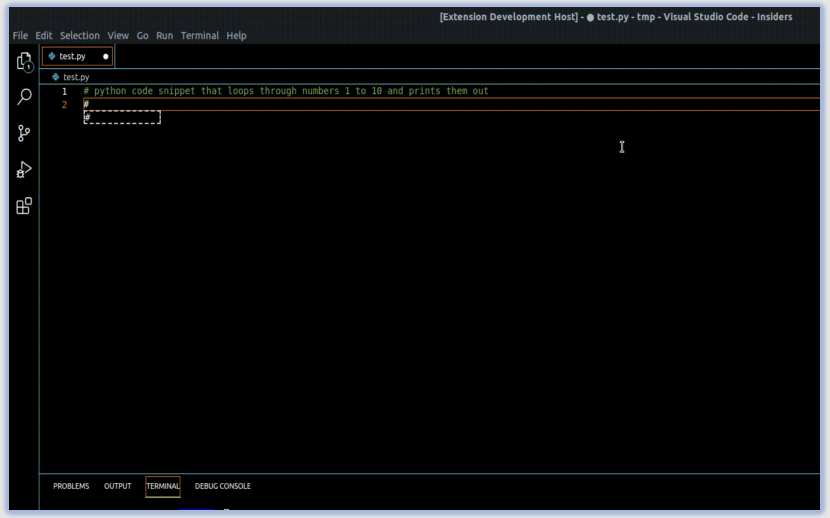
Darshan Gadginmath
PhD student
Mechanical Engineering

UC RIVERSIDE

AI agents in action

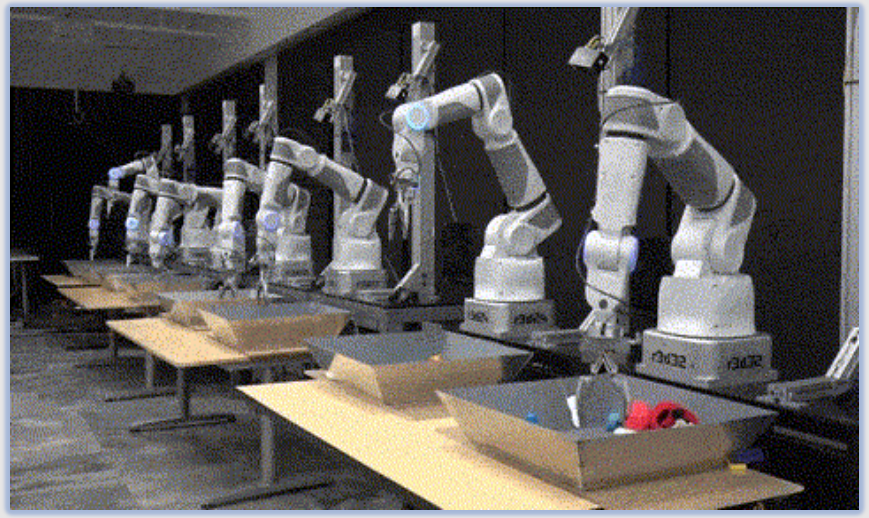
Content generation with diffusion models

- DALL-E
- Midjourney
- Sora



Code generation/completion with transformers

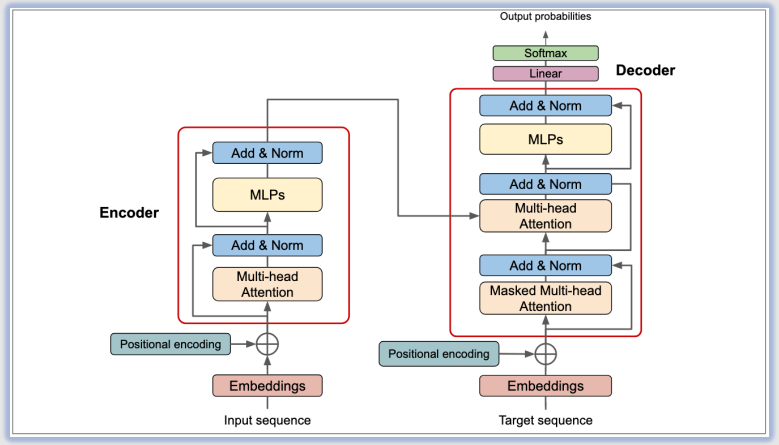
- Github Copilot
- Cursor.ai



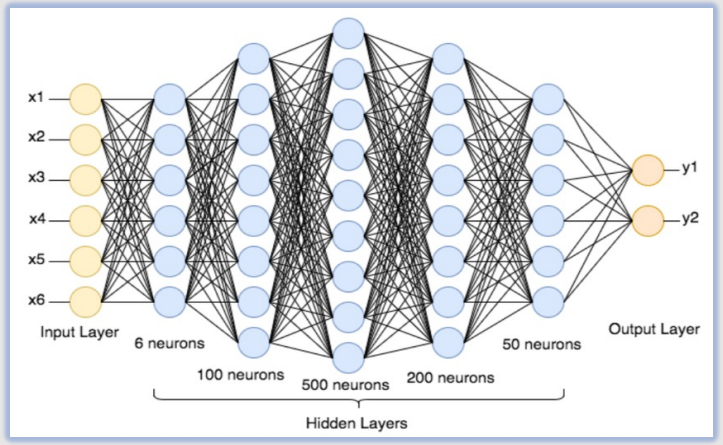
Foundation models with multi-task capability (active research)

Two issues: **Uninterpretability** and **no guarantees** in performance

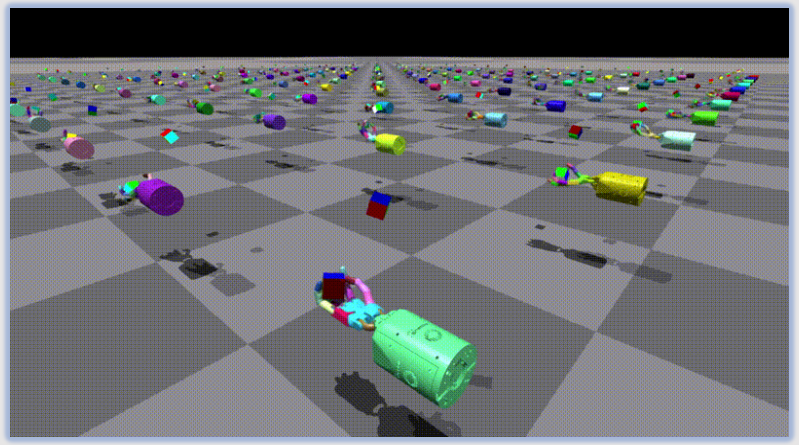
Uninterpretable nature of AI models



Transformers, SSMs



Feedforward NNs, RNNs, Convolutional NNs



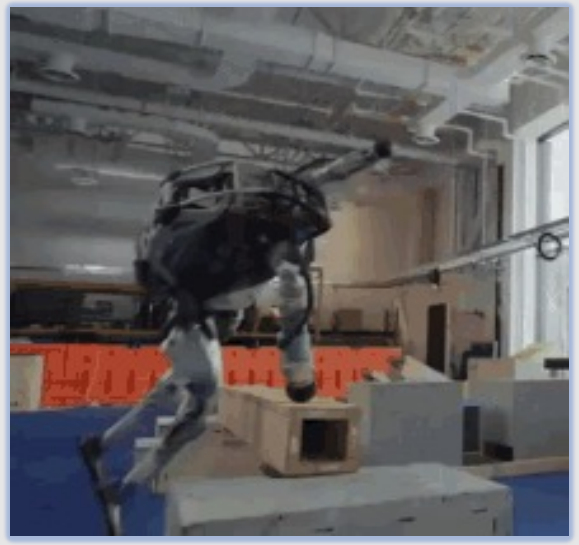
Massive models with complex architecture

- Billions of parameters
- Complex inter-linked structure
- Trained with massive datasets with biases and unclear statistics

Performance issues in AI models

Prompt: *"Generate a map of LA"*

Does not account for dynamics



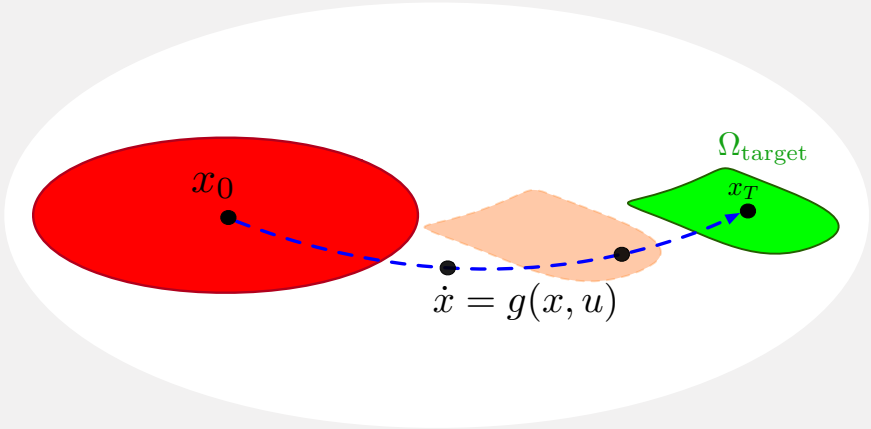
Failures in complex robotic tasks

Issues:

- Hallucination
- Complex physics need careful design
- Difficult to give guarantees on success rate

Dynamical systems perspective to AI agents

Agent dynamics: $\dot{x} = g(x, u)$



Compatible dynamics for

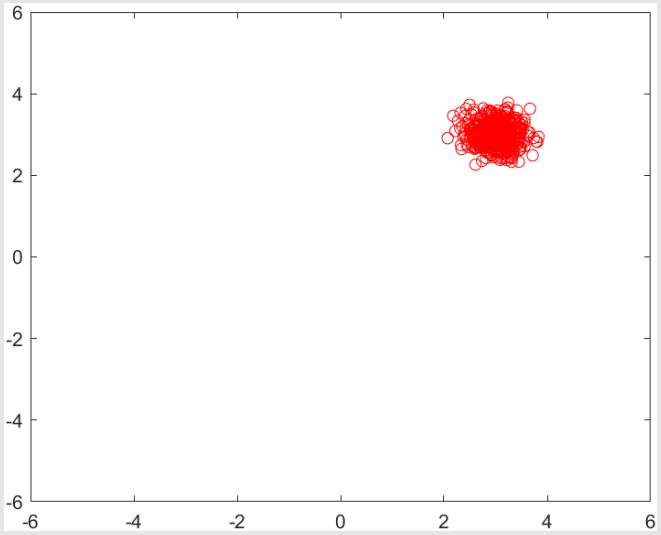
1. Physical systems: Robots, appliances
2. Generative models: Diffusion models, LLMs

Agent dynamics with black-box feedback: $\dot{x} = g(x, \mathtt{ML}_\theta(x))$

Probabilistic dynamics perspective

Dynamics with black-box feedback

$$\dot{x} = g(x, \mathtt{ML}_\theta(x))$$



Corresponding probabilistic dynamics

$$\partial_t \rho_t(x) = -\mathrm{div}\left[g(x, \mathtt{ML}_\theta(x)) \cdot \rho_t(x)\right]$$

$$\partial_t \rho_t(x) = \mathcal{L}\rho_t(x)$$

Liouville operator

Liouville and Perron-Frobenius operators

Liouville operator – differential:

$$\partial_t \rho_t(x) = \mathcal{L}\rho_t(x)$$

Perron-Frobenius operator – discrete:

$$\rho_{t+k}(x) = \mathcal{P}^k \rho_t(x)$$

PF operator is good to understand probabilistic behavior of autonomous systems

Properties:

- Propagates L^1 functions of state: $|f| = \int_X |f(x)| \mathrm{d}x < \infty$
  - Density
  - Risk functions
  - Probabilistic constraints
- Markov operator (positivity, preservation of probability)
- Adjoint of Koopman operator

Advantage of estimating the PF operator

1. Helps with **interpretability** by predicting short term behavior prediction

$$\hat{\rho}_{t+1} = \hat{\mathcal{P}}\rho_t$$

- Estimate probability of future states
- Reachability analysis

Invariant density $\rho^* = \mathcal{P}\rho*$

- Stable systems
- Oscillators

2. Verify alignment with human requirements - does the final state achieve the required task?

Estimating the PF operator - Data

Agent dynamics: $\dot{x} = g(x, \mathtt{ML}_\theta(x))$

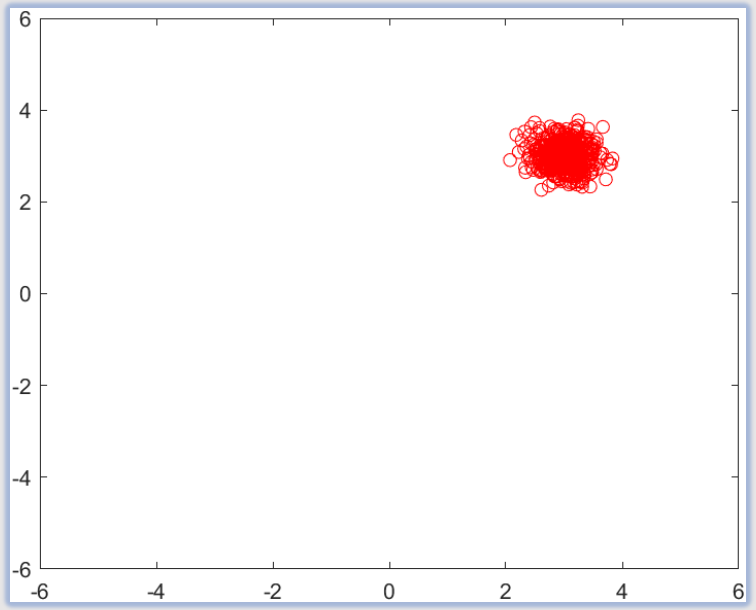
Experimental data – several individual trajectories

$$X_t = \begin{bmatrix} x_t^1 & x_t^2 & \cdots & x_t^N \end{bmatrix} \quad t \in \{0, 1, \cdots, T\}$$

Convert state data to probability densities

$$\begin{bmatrix} \rho_0 & \rho_1 & \rho_2 & \cdots & \rho_T \end{bmatrix}$$

- Kernel density estimation $\hat{\rho}_t(x) = \frac{1}{N} \sum_i \kappa(x, x_t^i)$
- Non-parametric ML methods



Estimating the PF operator – Existing methods

Popular with physicists

Ulam's method

- Discretize state-space into a grid
- Compute transition probability from box to box

EDMD-based method

- Dictionary of observables of state
- Compute a linear finite dimensional matrix to estimate evolution

Issues

- Discretization and linearization tend to perform poorly for complex systems
- Performance worsens with increase in state dimension
- Can only predict short-term behaviors
- Other existing methods tend to be computationally expensive

Spectral Decomposition Theorem

Let $\mathcal{P}$ be a constrictive PF operator (stable invariant density)

$$\rho* = \lim_{t\to\infty} \mathcal{P}\rho_t$$

There exists

- Two sequences of ℓ functions: $a_i(\rho)$ and $b_i(x)$
- Operator $\mathcal{Q} : L^1 \to L^1$

$$\rho_{t+1} = \mathcal{P}\rho_t = \sum_i^{\ell} a_i(\rho) b_i(x) + \mathcal{Q}\rho_t$$

- Properties
  1. Functions $b_i(x)$ are orthogonal: $b_i(x)b_j(x) = 0$
  2. $\mathcal{P}$ permutes $b_i(x)$: $\mathcal{P}b_i(x) = b_j(x)$
  3. Asymptotic property of $\mathcal{Q}$: $\lim_{t\to\infty} \mathcal{P}^t \mathcal{Q}\rho_t = 0$

Prediction Informed by Spectral-decomposition Algorithm (PISA)

1. Two neural networks - $\mathtt{A}_\theta$ and $\mathtt{B}_\gamma$, each with ℓ outputs:

2. Invariant density: $\rho^*(x) \approx \frac{1}{\ell} \sum_i^\ell \mathtt{B}_\gamma^i(x)$

3. Action of operator $\mathcal{Q}$: $\mathcal{Q}\rho_t = \rho_t - \rho*$

Estimate of PF operator: $\hat{\mathcal{P}}\rho_t = \rho_t(x) - \sum_{i=1}^{\ell} \left(\frac{1}{\ell} - \mathtt{A}_\theta^i \right) \mathtt{B}_\gamma^i$

$$\min_{\theta,\gamma} L = \sum_{t=0}^{T-1} KL(\rho_{t+1} || \hat{\mathcal{P}}\rho_t) + \lambda \sum_{i \neq j} \langle \mathtt{B}_\gamma^i, \mathtt{B}_\gamma^j \rangle + \mu \sum_{i=1}^{\ell} KL(\hat{\mathtt{B}}_\gamma^i || \mathtt{B}_\gamma^i)$$

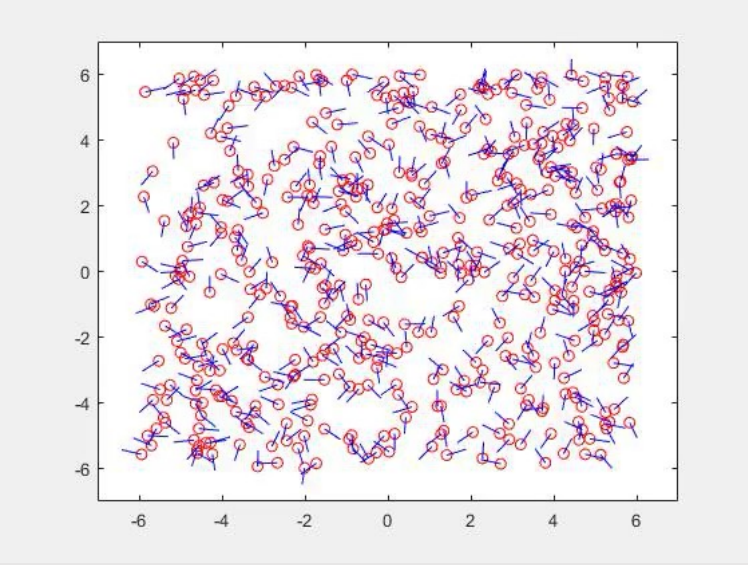
Density Estimation Orthogonality Permutation

Experimental results – nonlinear systems

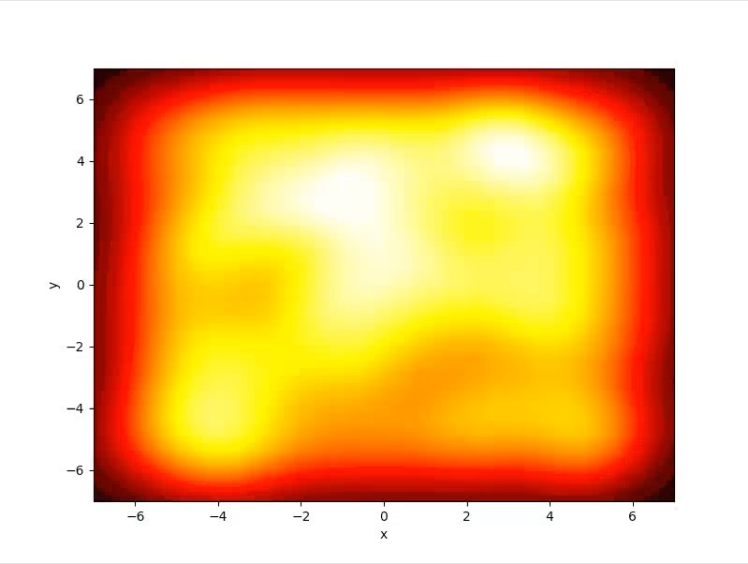
Unicycle model with NN controller

$$\dot{x} = u_1 \cos(\theta) \quad \dot{y} = u_1 \sin(\theta) \quad \dot{\theta} = u_2$$

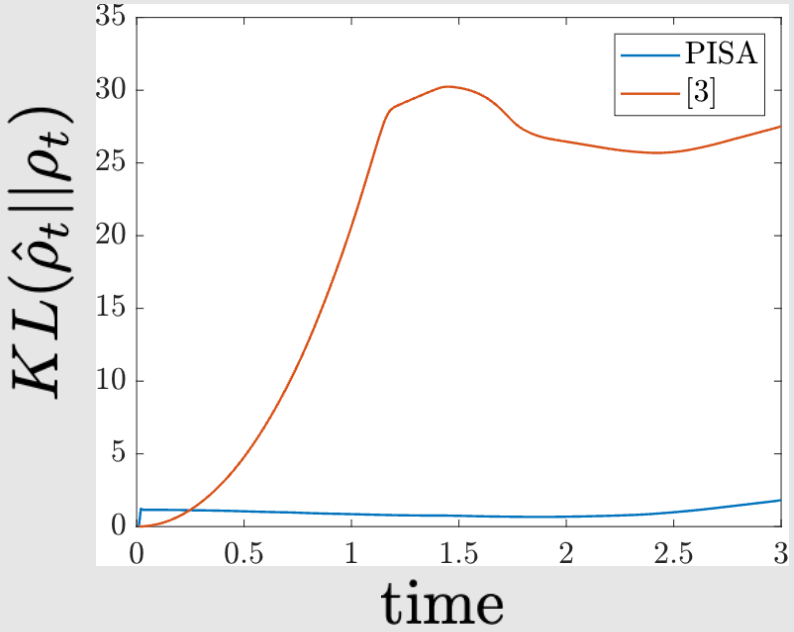
Trajectory data



Predicted density



Comparison



PISA's performs well for 3 seconds, SoTA rapidly deteriorates

Experimental results – generative models

Score-based diffusion model with differential equations (Y. Song, et al [2020]):

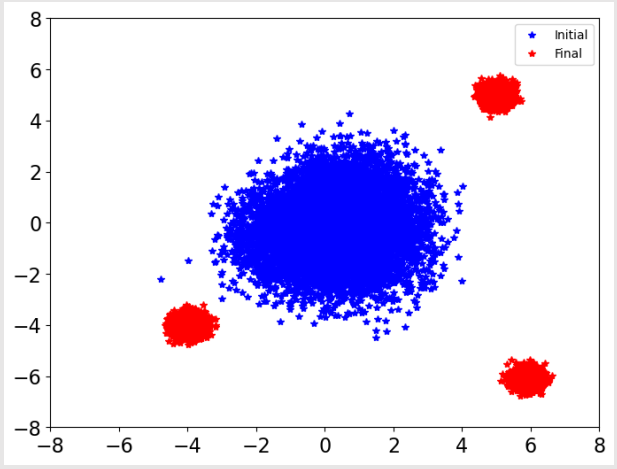
1. Sample from a gaussian: $x_0 \sim \mathcal{N}(0, I)$
2. Denoise using score-based model to sample from data distribution

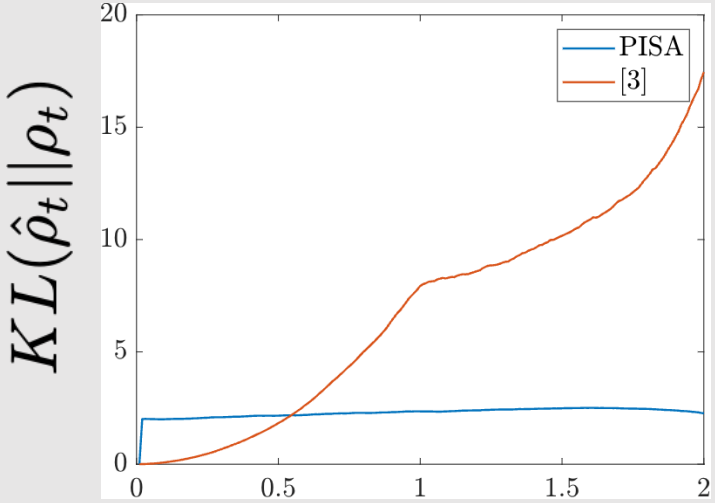
$$\dot{x} = Ax + B\mathrm{NN}_\beta(x)$$

Behavior prediction of a **10-dimensional** diffusion model

Data samples

Noise



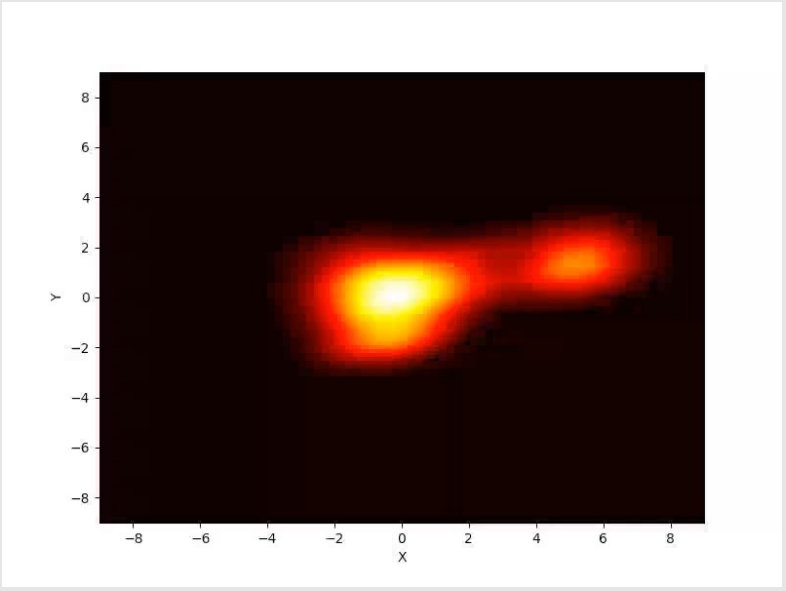


PISA's performs well up to 2 seconds

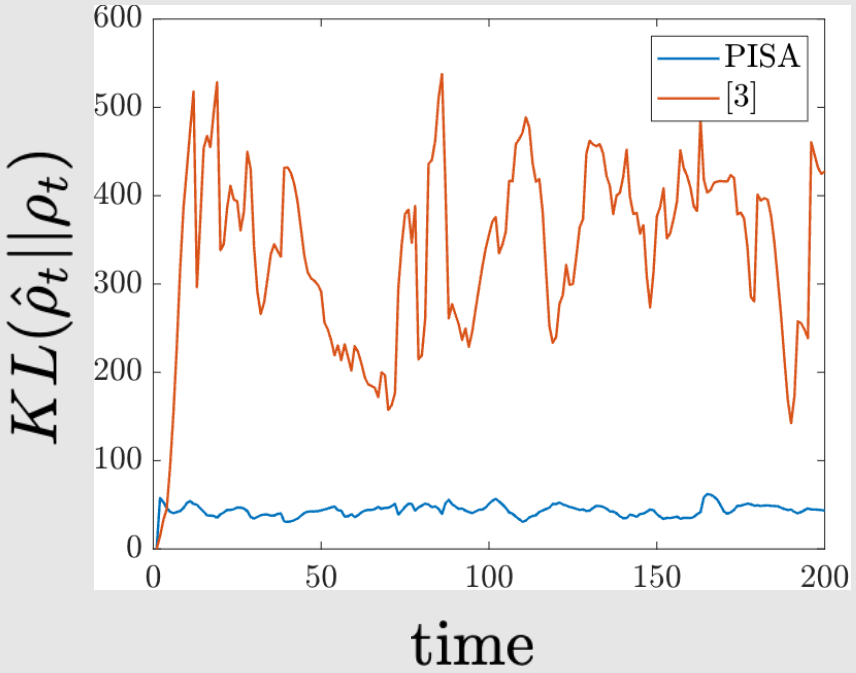
Experimental results – pedestrians

UCY pedestrian dataset

Predicted density



Comparison

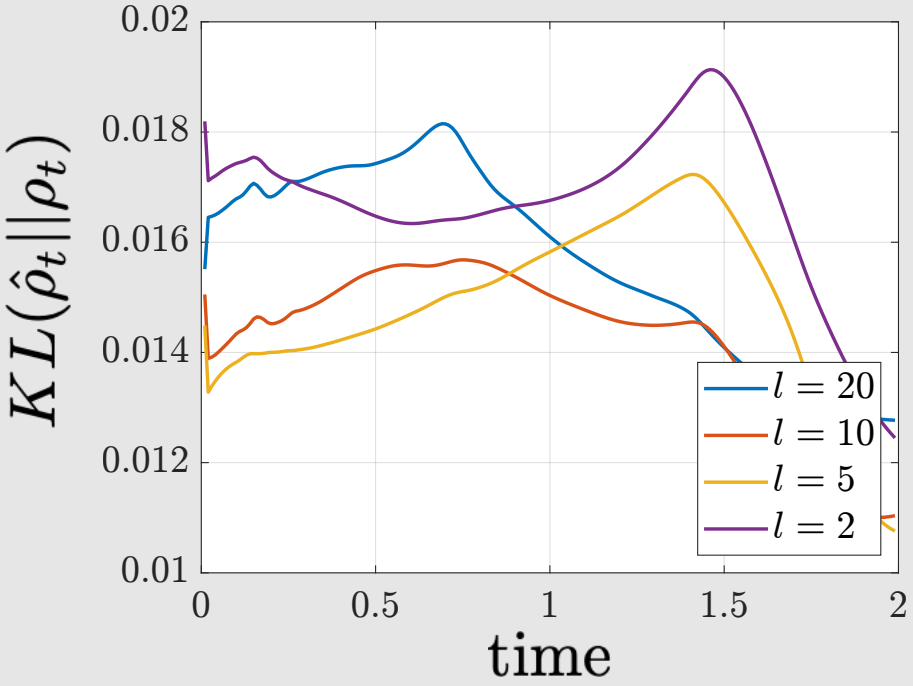


Assumptions:

1. All pedestrians are identical and have the same dynamics
2. No knowledge about interaction

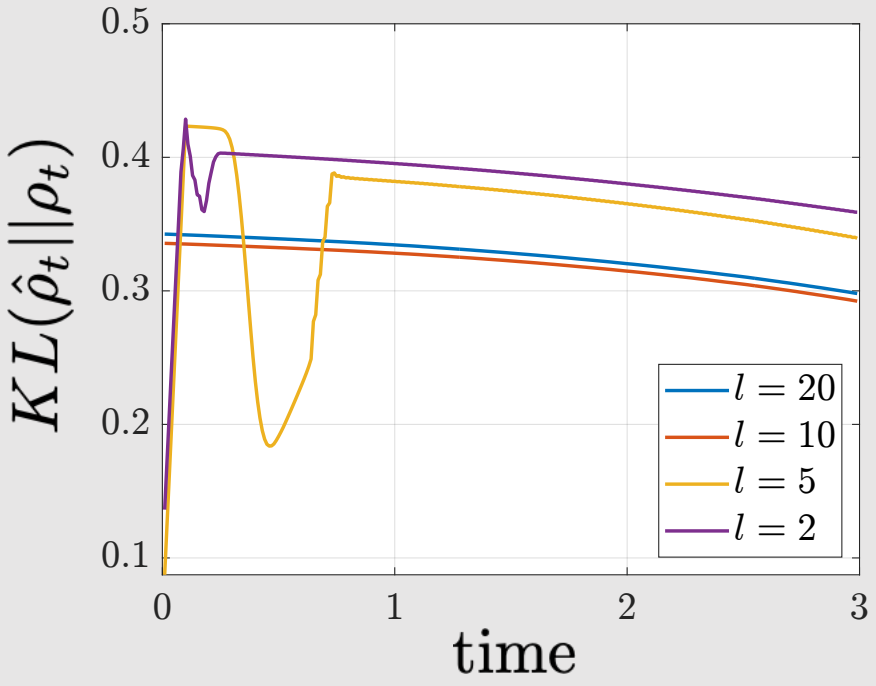
Experimental results – effect of choice of ℓ

Unicycle



Choice of ℓ matters little in low dimensions

Diffusion model



$\ell = 10$ seems to give good performance

Future directions

- Optimization of PISA
  - Careful analysis of choice of ℓ and better NN architectures
  - Replacing KDE with non-parametric density methods
  - Account for more realistic cases
  - Study of model complexity and computational efficiency

- Behavior cloning with an expert's PF operator: Only demonstrations, no knowledge of control actions

Nonlinear control with diffusion models

Capability of diffusion models

Futuristic robots working at USC **(ChatGPT)**

A litter of golden retrievers playing in the snow. Their heads popping out of snow **(Sora)**

Smooth soft R&B song with tender vocals, romantic piano and groovy, funky bass **(Noise2Music)**

Mechanism behind diffusion models

Data (unknown) → Forward (noising) → Noise (known)

Noise (known) → Reverse (denoising) → Data (unknown)

[Ho, et al.] – Denoising diffusion probabilistic models (discrete-time)
[Song, et al.] – Score-based diffusion models (SDE, continuous-time)

REVERSE-TIME DIFFUSION EQUATION MODELS*

Brian D.O. ANDERSON**
Department of Electrical Engineering, The University of Newcastle, N.S.W. 2308, Australia

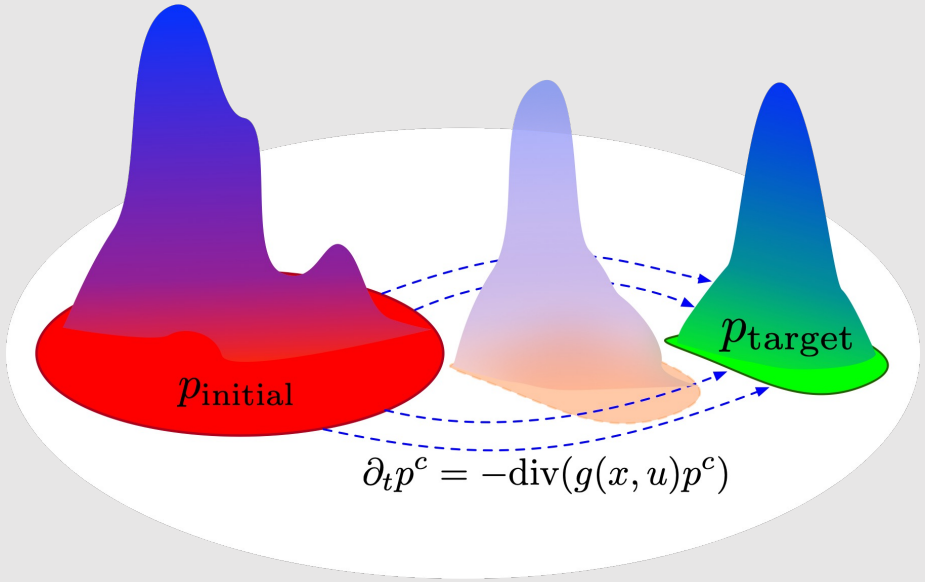
- DDPMs can generate high quality (mostly accurate) content
- Can they be used for control?

Problem formulation

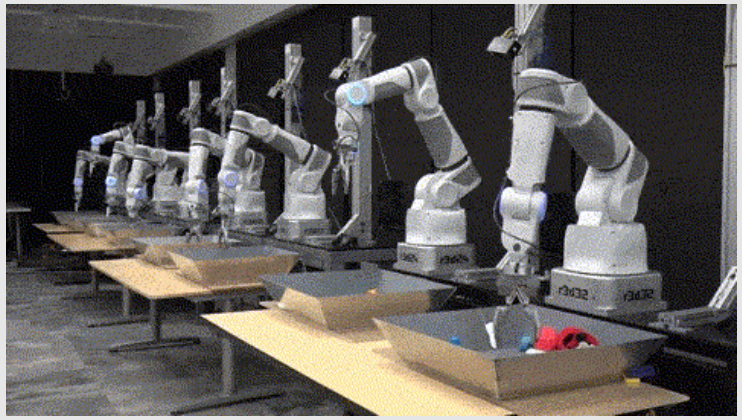
System: $\dot{x} = g(x, u)$

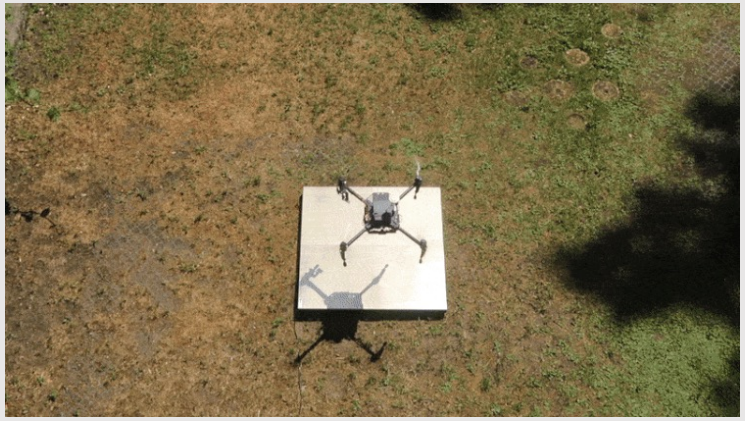
Data: M samples from an unknown density p_{target}

Objective: Design control u to take samples from p_{initial} to p_{target}



Practical challenges





- Target density is not known, only samples are available
- State is *sampled* at discrete-time instances
- *Limited computational resources* to learn controller

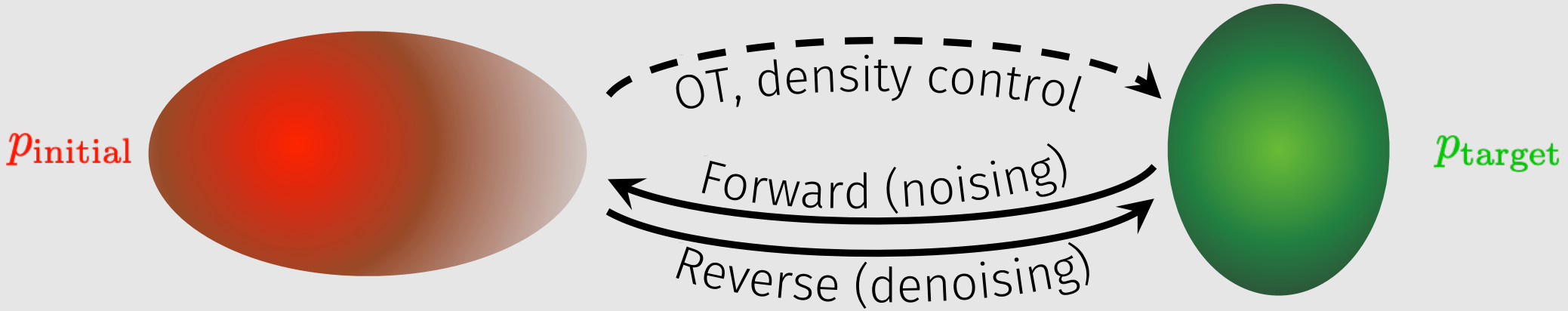
Comparison with traditional problems

Optimal transport (OT): $\min_u \int_0^T \int_\Omega L(x,u)\ \mathrm{d}x\ \mathrm{d}t$

s.t. $p(0) = p_{\text{initial}},\ p(T) = p_{\text{target}}$

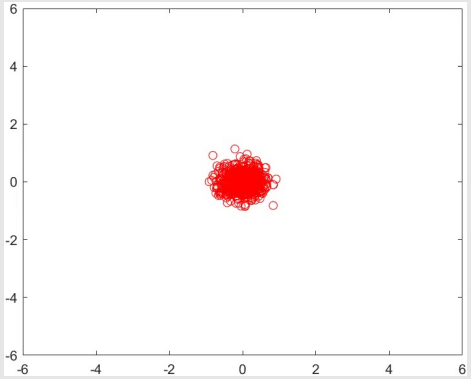
Density control: Design feedback u for system $\dot{x} = g(x,u)$

s.t. $p(0) = p_{\text{initial}},\ p(T) = p_{\text{target}}$



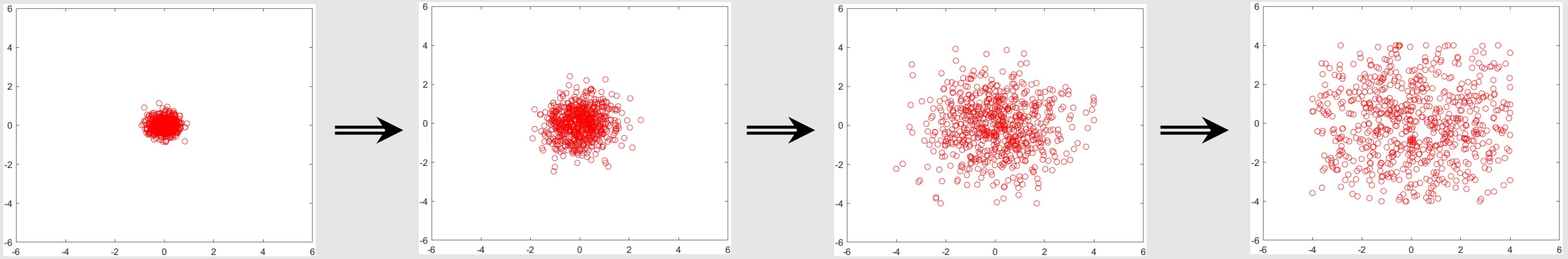
DDPMs for control – forward process

Data: Samples from $p_{\text{target}} = \mathcal{N}(0, I)$



Noising: $\dot{x} = h(x, \mathbf{w})$

$p^f(0) = p_{\text{target}}$



$p^f(T) = p_{\text{initial}}$

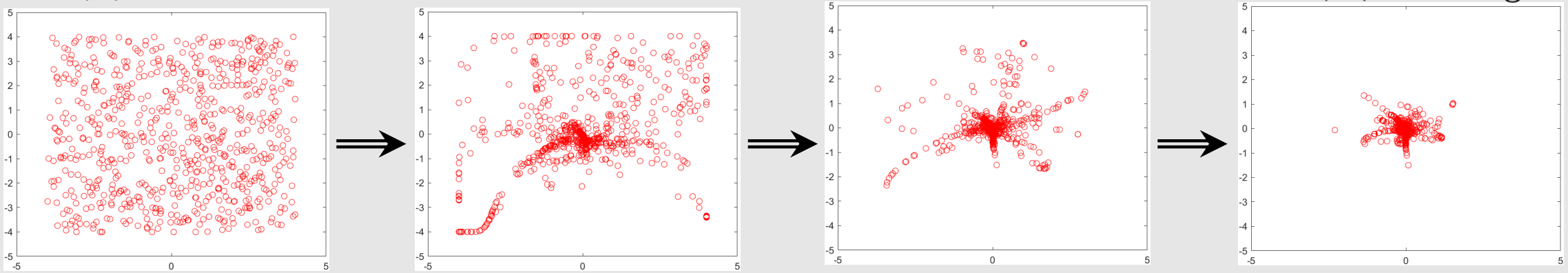
DDPMs for control – reverse process

Particle dynamics: $\dot{x} = g(x, u)$

Density dynamics: $\partial_t p^c(x) = -\mathrm{div}\left[p^c(x) \cdot g(x, u)\right]$

Example: unicycle model

$p^c(0) = p_{\text{initial}}$ $p^c(T) = p_{\text{target}}$



When can we exactly track the forward process in reverse?

Existence of controller

Recall notation

- Forward trajectory: $p^f(t)$, $p^f(0) = p_{\text{target}}$, $p^f(T) = p_{\text{initial}}$
- Reverse trajectory: $p^c(t)$

Theorem: Tracking forward process in reverse

- *Condition 1:* Driftless system
- *Condition 2:* Smoothness of domain
- *Condition 3:* Lie-bracket based controllability

$$p^c(t) = p^f(T - t)$$

K. Elamvazhuthi, D. Gadginmath, and F. Pasqualetti, 2024. Denoising Diffusion-Based Control of Nonlinear Systems, CDC 2024 (To appear)

Sketch of proof

Driftless system: $\dot{x} = g(x,u) = \sum_{i=1}^{m} g_i(x)u_i$

Differential operator $\mathcal{K}_i$ that evolves observables

Evolution of an observable of the state:

$$\dot{s} = \sum_{i=1}^{m} \mathcal{K}_i s = \partial_x s \sum_{i=1}^{m} g_i(x)u_i$$

The formal adjoint operator $\mathcal{K}_i^*$ evolves densities

$$\partial_t p^c = \sum_{i}^{m} \mathcal{K}_i^*(p^c u_i) = \sum_{i}^{m}\sum_{j=1}^{n} -\partial_{x_j}\left(g_i^j(x)u_i p^c\right)$$

Ability to track the forward process (in reverse) depends on invertiblity of $\sum_{i}^{m} \mathcal{K}_i^*\mathcal{K}_i$

Algorithm to design controller

Controller: $u = \texttt{NN}_\theta(x, t)$

1. Simulate the forward process, $\dot{x} = h(x, \mathbf{w})$, for M particles
2. Sample the states at times: $0 < t_1 \leq t_2 \leq \cdots \leq t_N = T$
3. Iteratively optimize controller to track forward process as

$$\dot{x} = g(x, \texttt{NN}_\theta(x, t))$$

$$\min_\theta \frac{1}{N} \sum_i^N KL\left(p^c(t_i) || p^f(T - t_i)\right)$$

$p^c(t_i)$ and $p^f(t_i)$ are estimated using Kernel Density Estimation as

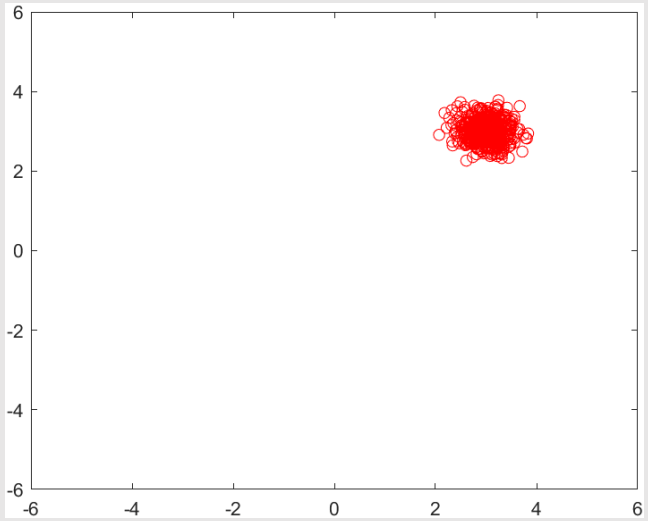
$$p(t_i) = \frac{1}{M} \sum_i^M \kappa(x, x_i)$$

Simulation results

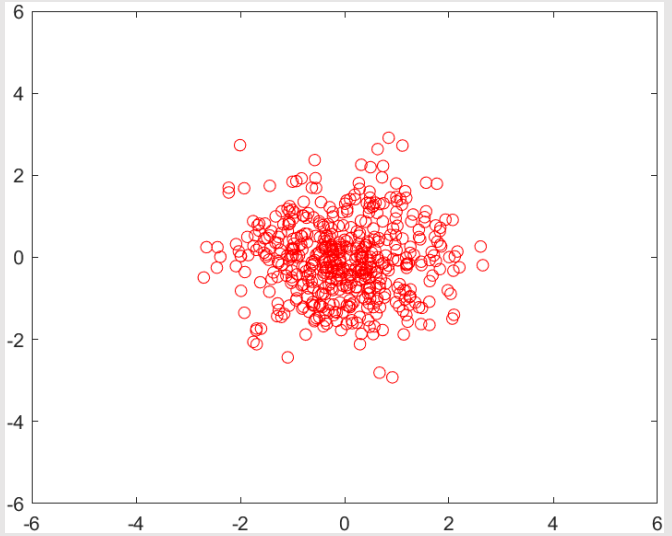
Unicycle robots:

$$\dot{x} = u_1 \cos(\theta) \quad \dot{y} = u_1 \sin(\theta) \quad \dot{\theta} = u_2$$

$$p_{\text{initial}} = \mathcal{N}(0, I) \qquad p_{\text{target}} = \mathcal{N}(31, \mathbf{0.5I})$$



Forward process: $\dot{x} = -x + 0.5\mathbf{w}$



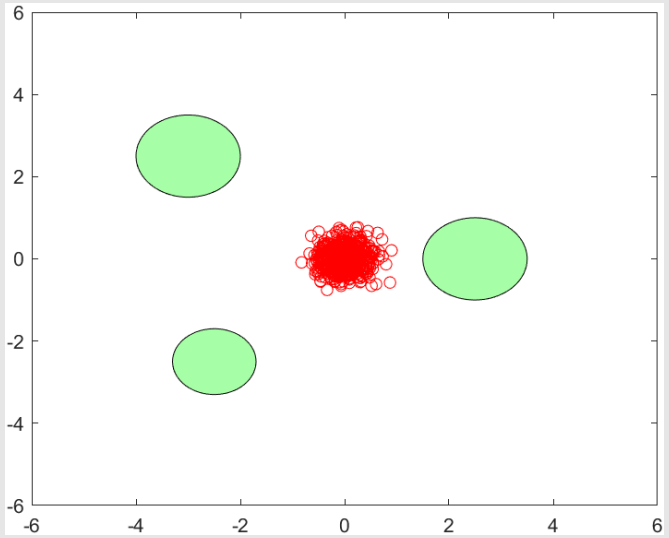
Reverse process: $\dot{x} = g(x, \mathtt{NN}_\theta(x, t))$

Simulation results – trajectory planning

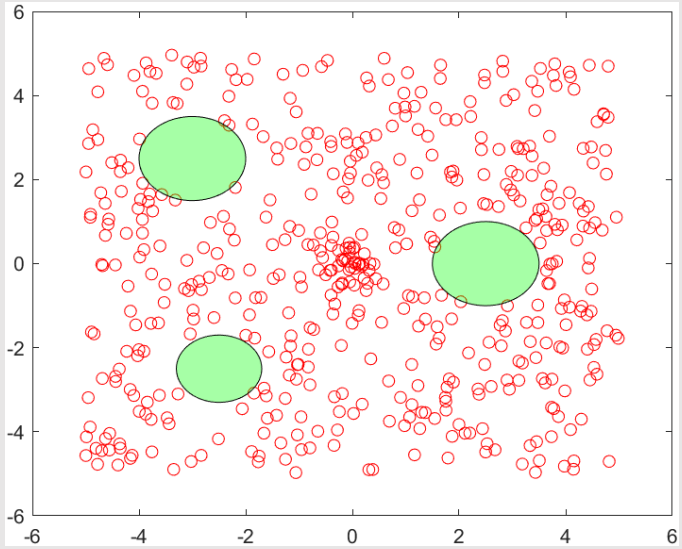
Unicycle robots:

$$\dot{x} = u_1 \cos(\theta) \quad \dot{y} = u_1 \sin(\theta) \quad \dot{\theta} = u_2$$

$$p_{\text{initial}} = \mathcal{U}(-5, 5)^3 \qquad p_{\text{target}} = \mathcal{N}(0, I)$$



(non-collision constraint)



Forward process: $\dot{x} = \mathbf{w}$

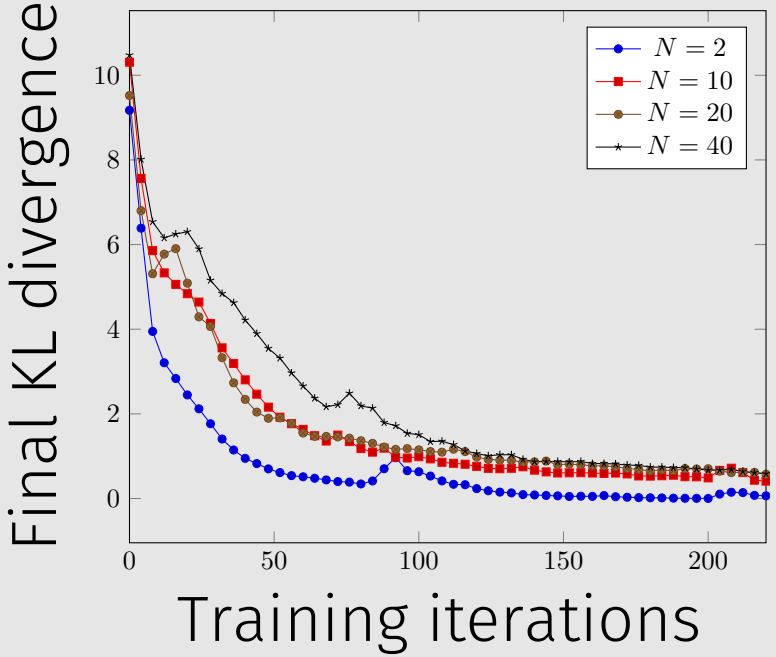
Reverse process: $\dot{x} = g(x, \mathtt{NN}_\theta(x, t))$

Simulation results – 5 dimensional system

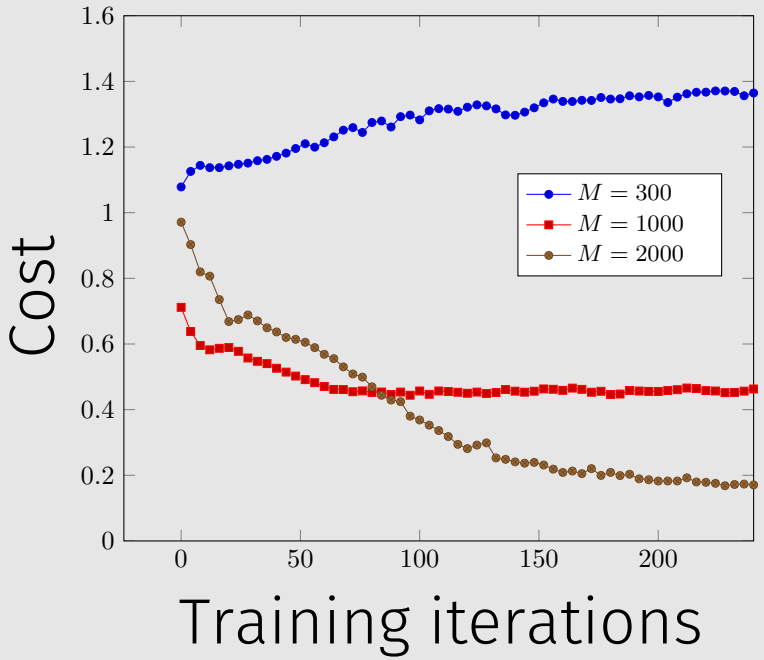
Particle dynamics:

$$\dot{x}_1 = u_1,\ \dot{x}_2 = u_2,\ \dot{x}_3 = x_2 u_1,\ \dot{x}_4 = x_3 u_1,\ \dot{x}_1 = x_4 u_1,$$

Controller existence verified

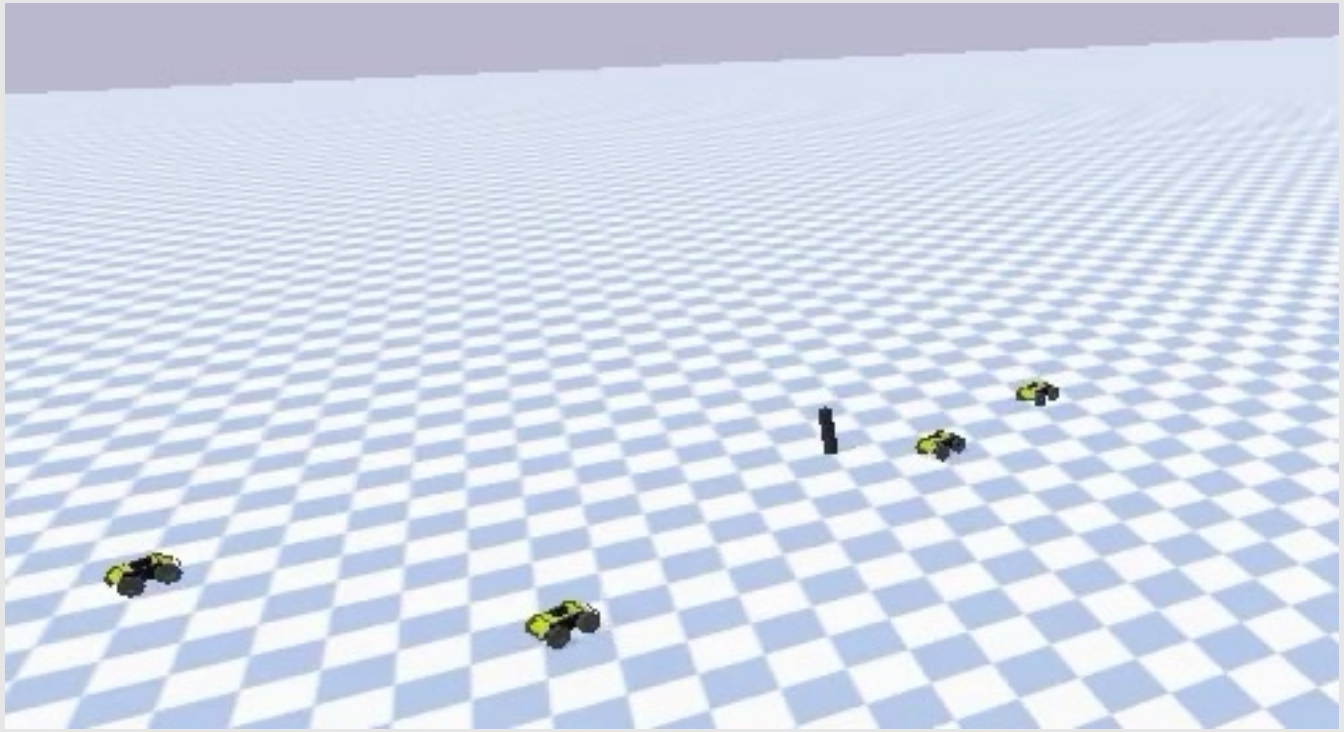


Data requirement



Simulation results – robotic system

Controller trained on unicycle model, and tested on Husky robot



Code: https://github.com/darshangm/diffusion-nonlinear-control

Future directions

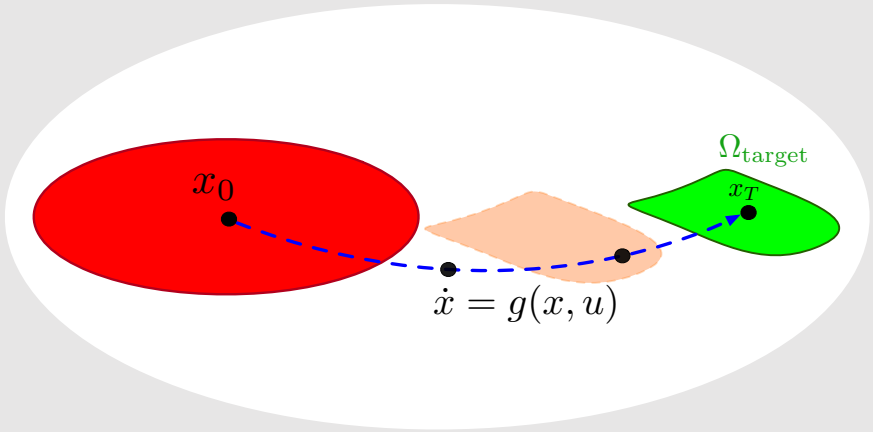
- Systems with drift
- Error bounds on tracking for general systems
- Arbitrary initial and target densities
- A numerical package for nonlinear control

Data-driven nonlinear control: Feedback linearization

Data-driven control problem

Unknown dynamical system: $\dot{x} = g(x, u)$

Experimental data: $X = [x_1 \; x_2 \cdots \; x_T] \quad U = [u_1 \; u_2 \cdots \; u_T]$



Linear system: $\dot{x} = Ax + Bu$

Learn controller: Persistently exciting U

$$\mathcal{H}(X, U) \; [\hat{X} \; \hat{U}]^\top$$

If the dynamics are nonlinear, when can we learn a stabilizing controller?

Feedback Linearization (model-based)

Nonlinear control system: $\dot{x} = f(x) + g(x)u$

Involutivity: $\Delta = \text{span}\{g, \text{ad}_f g, \text{ad}_f^2 g, \cdots, \text{ad}_f^{r-2} g\}$

Change of coordinates:

$$v = \alpha(x) + \beta(x)u\,, \quad z = H(x) = \begin{bmatrix} h(x) \\ L_f h(x) \\ \vdots \\ L^{(r-1)} h(x) \end{bmatrix} \longrightarrow \boxed{\dot{z} = Az + Bv}$$

r-dimensional

How do we use feedback linearization with data?

Need to learn: $H(x),\ \alpha(x),\ \beta(x)$

Koopman operator and EDMD

Dynamical system: $\dot{x} = f(x)$

Koopman operator: $\mathcal{K}^{\tau} h(x(t)) = h(x(t+\tau))$

Infinitesimal generator: $L_f h(x) = \dot{h}(x)$

How do we learn the Koopman operator/generator from data?

Experimental data: $X = [x_1 \; x_2 \cdots \; x_T]$, dictionary of functions $\psi(x)$

Extended Dynamic Mode Decomposition: $\Psi(X) = [\psi(x_1) \; \psi(x_2) \; \cdots \; \psi(x_T)]$

$$\dot{\psi}(x_t) \approx \hat{A}\psi(x_t)$$

Connection between Koopman and FL

Involutivity: $\Delta = \mathrm{span}\{g, \mathrm{ad}_f g, \mathrm{ad}_f^2 g, \cdots, \mathrm{ad}_f^{r-2} g\}$

Theorem: Koopman generator and feedback linearization

There exist an observable $h(x)$ and a feedback $\alpha(x)$ such that $L_{f+g\alpha}$ is nilpotent to a degree r at the observable $h(x)$.

$$L_{f+g\alpha}^r h(x) = 0$$

- Still need to find observable $h(x)$ and feedback $\alpha(x)$ and $\beta(x)$
- We can use Koopman-based ideas with dictionaries

Data-driven feedback linearization

Experimental data: $X = [x_1\ x_2 \cdots\ x_T] \quad U = [u_1\ u_2 \cdots\ u_T]$

Transformation with dictionaries: $z = D(x)K,\ v = G\theta(x) + J\gamma(x)u$

Learning combination of dictionaries

$$\min_{K,G,J} \sum_{t}^{T-1} \left|\left| \frac{D(x_{t+1}) - D(x_t)}{\tau} K - AD(x_t) - Bv_t \right|\right|^2$$

$$\text{s.t. } v_t = G\theta(x) + J\gamma(x), A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

Experimental results – Van der pol oscillator

System: $\dot{x}_1 = x_2$

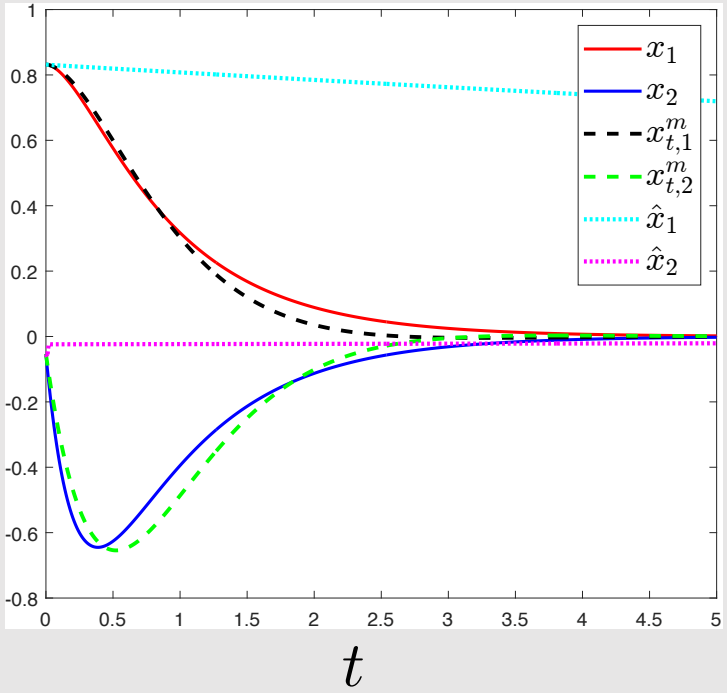
$$\dot{x}_2 = -x_1 + 0.5(1 - x_1^2)x_2 + (1 - x_2^2)u$$

Model-based: – – –

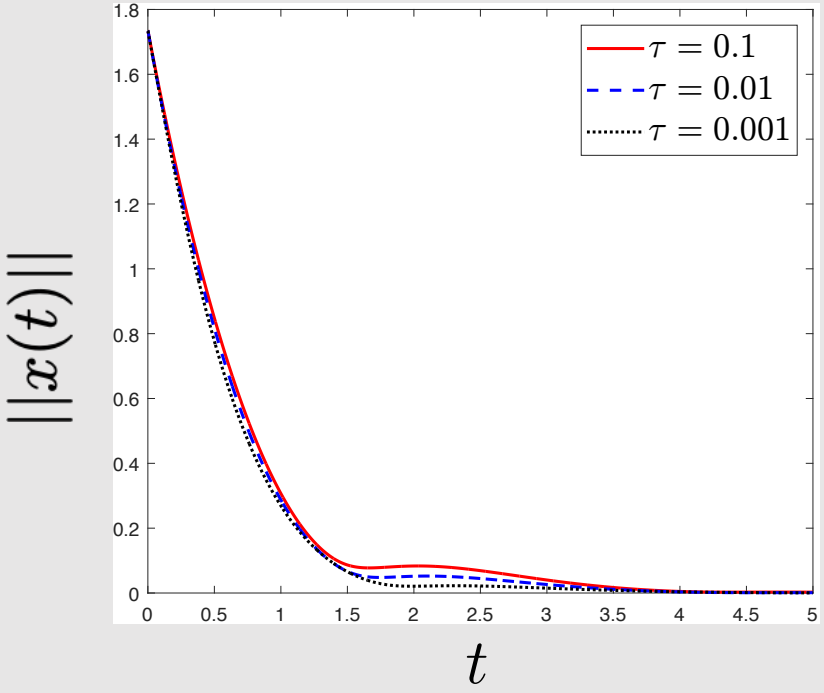
Naïve: ·····

KGFL: ——

Comparison



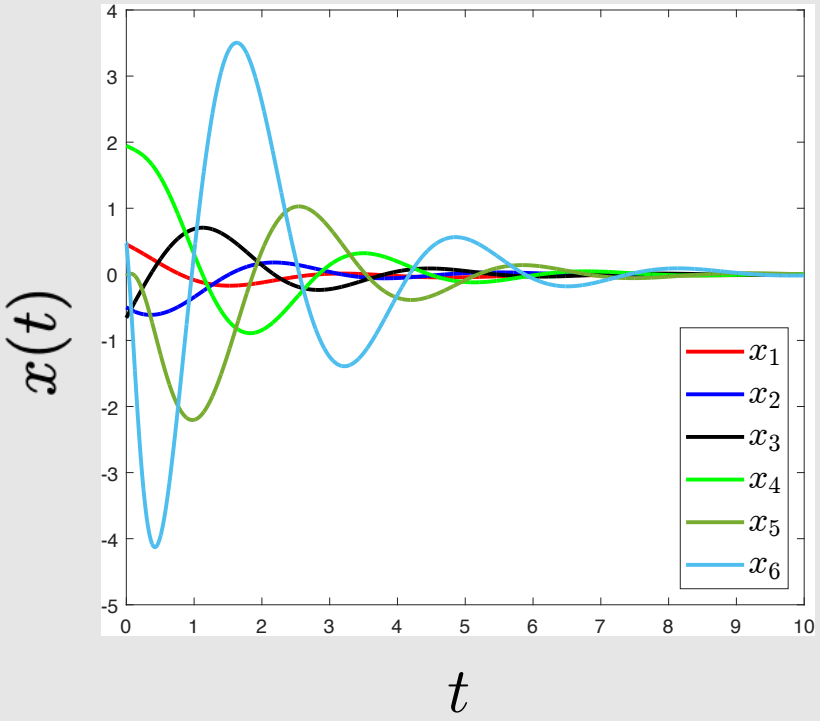
Effect of sampling



Experimental results – six dimensional

System: $\dot{x}_1 = x_2,\ \dot{x}_2 = x_3 + x_2,\ \dot{x}_3 = x_4 + \sin(x_1),$

$\dot{x}_4 = x_5 + \cos(x_2),\ \dot{x}_5 = x_6 + x_1,\ \dot{x}_6 = -x_1^2 + \sin(x_1) + u$



Conclusion

- Probabilistic perspective
  - Good for understanding behavior of black box systems
  - Computationally efficient method for nonlinear control and transport
  - Operator theoretic methods naturally incorporate dynamics
- Data-driven nonlinear control- feedback linearization gives accurate linearized control

Vishaal Krishnan

Karthik Elamvazhuthi

Shiqi Zhang

Prof. Fabio Pasqualetti