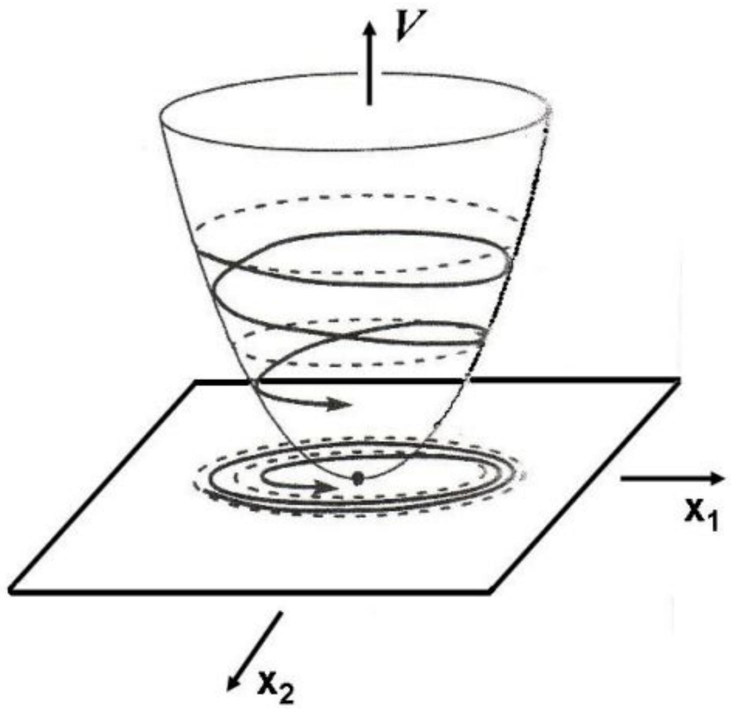
Combining Geometric Nonlinear Control with Reinforcement Learning-Enabled Control

Tyler Westenbroek
Dept EECS
University of California Berkeley

Adapted for EECS 290-005,
February 24, 2021

Geometric Nonlinear Control

- **Main idea:** exploit underlying structures in the system to systematically design feedback controllers
  - Explicitly connects ‘global’ and ‘local’ system structures
  - Gives fine-grain control over system behavior
  - Amenable to formal analysis
  - Difficult to learn to exploit non-parametric uncertainties

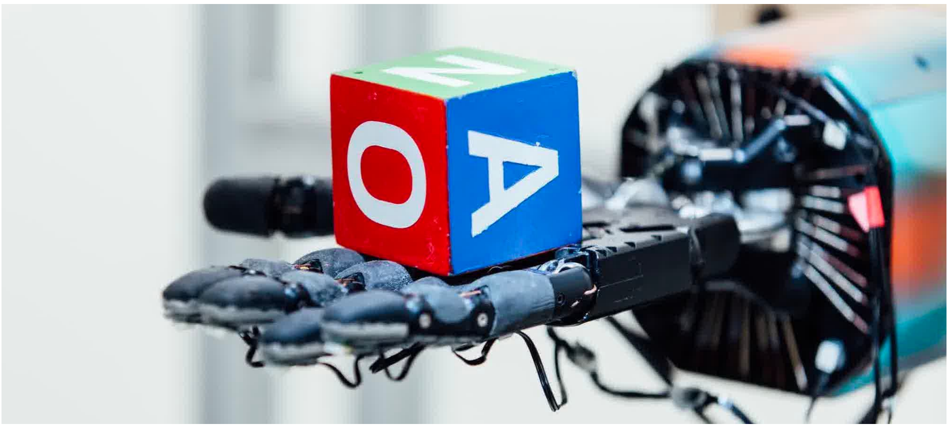


Deep Reinforcement Learning

- **Main idea:** sample system trajectories to find (approximately) optimal feedback controller
  - ‘Discovers’ connection between global and local structure
  - Automatically generates complex behaviors, but requires reward shaping
  - Effectively handles non-parametric uncertainty
  - Can require large amounts of data

$$V(s) = \mathbb{E}[R_{t+1} + \gamma V(S_{t+1}) | S_t = s]$$

[Levine et. al.](IJRR 2020)



[Open AI](2019)

Motivating Questions

- Can we design local reward signals with global structural information ‘baked in’ using geometric control?

- Can we use these structures to provide correctness and safety guarantees for the learning?

- Does reinforcement learning implicitly take advantage of these structures? What structures make a system ‘easy’ to control?

Thesis Proposal

Part 1: **Overcome non-parametric uncertainty by combining RL and geometric control**

$$\hat{u}_\theta(x) = \underbrace{u_m(x)}_{\text{MB Controller}} + \underbrace{\Delta u_\theta(x)}_{\text{Learned Correction}}$$

Tyler's PhD research

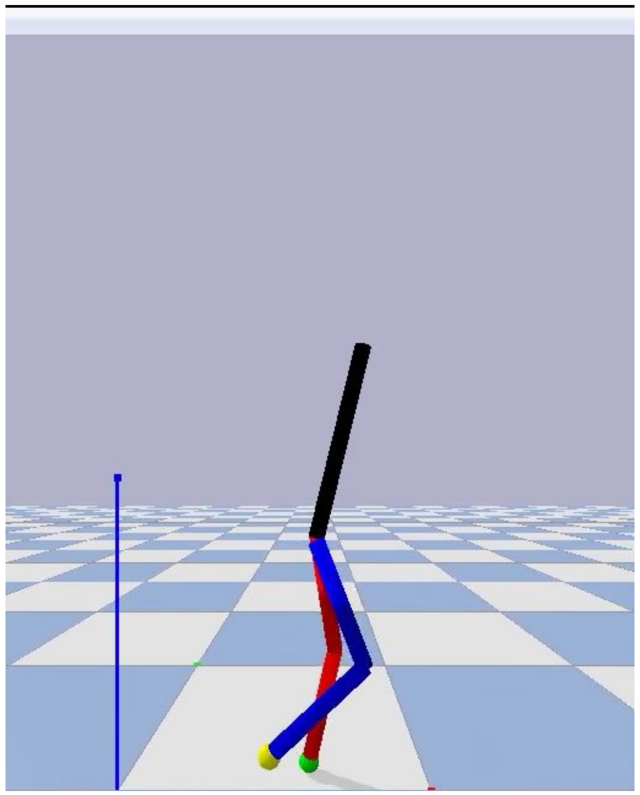
<u>Part 1</u>: **Overcome non-parametric uncertainty by combining RL and geometric control**

$$\underbrace{\hat{u}_\theta(x) = u_m(x)}_{} + \Delta u_\theta(x)$$

MB Controller

Learned Correction

Example: Learning a stable walking gait with ~20 seconds of data



Use structures from geometric control as a 'template' for the learning

Project Flow

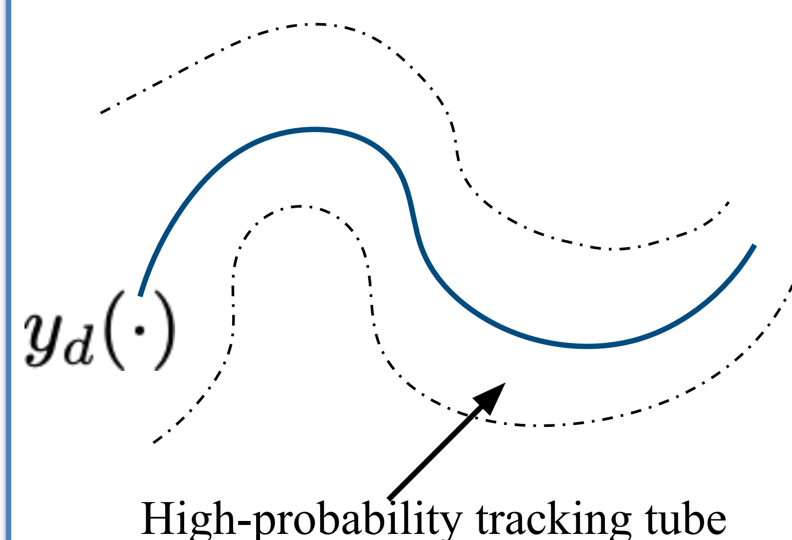
Part 1: **Overcome non-parametric uncertainty by combining RL and geometric control**

$$\hat{u}_\theta(x) = u_m(x) + \Delta u_\theta(x)$$

MB Controller

Learned Correction

Part 2: **Provide correctness and safety guarantees for specific learning algorithms:**



Part 3: **Future work:**

- **What makes a reward signal difficult to learn from?**
- **What makes a system fundamentally difficult to control?**
- **Where should geometric control be used in the long-run?**

Part 1 Outline

- Steps in design process
- Example control architectures
  - Feedback Linearization
  - Control Lyapunov Functions
  - Other architectures
- Trade-offs with ‘Model-based’ RL

Steps in Design Process

Step 1: Choose geometric control architecture which produces desired global behavior

$$\dot{x} = f_m(x) + g_m(x)u_m(x)$$

e.g. feedback linearizing controller

Step 2: Augment the nominal controller with a learned component:

$$\hat{u}_\theta(x) = u_m(x) + \Delta u_\theta(x)$$

learned augmentation

Step 3: Formulate reward which captures desired local behavior

$$\min_{\theta \in \Theta} \mathbb{E}_{x \sim X} \ell(x, \theta)$$

Minimize loss with RL

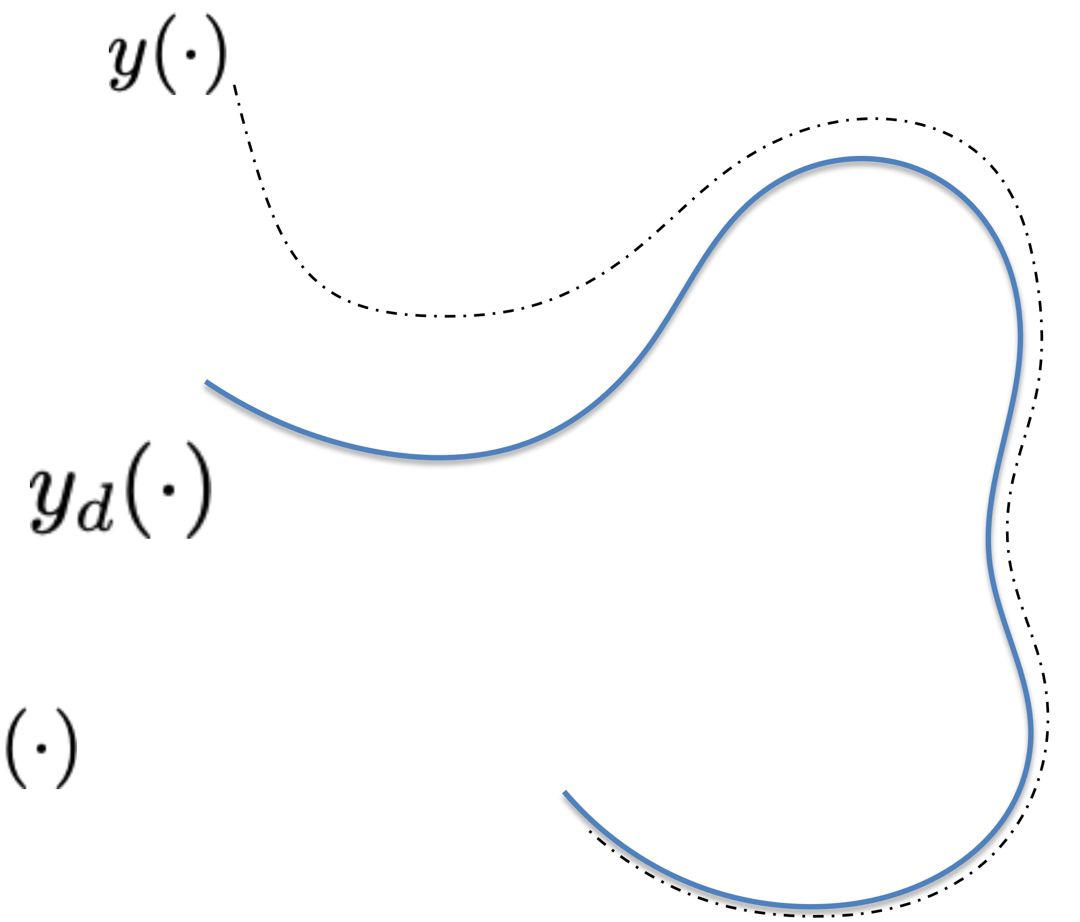
Feedback Linearization

Goal: Output Tracking

- Consider the system

$$\dot{x} = f(x) + g(x)u$$
$$y = h(x)$$

with $x \in \mathbb{R}^n$ the state, $u \in \mathbb{R}^q$ the input and $y \in \mathbb{R}^q$ the output.



- Goal: track any smooth reference $y_d(\cdot)$ with one controller

Calculating a Linearizing Controller

- For the time being assume $q = 1$. To obtain a direct relationship between the inputs and outputs we differentiate y:

$$\begin{aligned}\dot{y} &= \frac{d}{dx}h(x) \cdot [f(x) + g(x)u] \\ &= \frac{d}{dx}h(x) \cdot f(x) + \frac{d}{dx}h(x) \cdot g(x)u \\ &= b_1(x) + a_1(x)u\end{aligned}$$

- Now if $a_1(x) \neq 0$ for each $x \in \mathbb{R}^n$ then the controller

$$u(x, v) = \frac{1}{a_1(x)}[-b_1(x) + v]$$

yields

$$\dot{y} = v$$

Calculating a Linearizing Controller

- For the time being assume $q = 1$. To obtain a direct relationship between the inputs and outputs we differentiate y:

$$\begin{aligned}\dot{y} &= \frac{d}{dx}h(x) \cdot [f(x) + g(x)u] \\ &= \frac{d}{dx}h(x) \cdot f(x) + \frac{d}{dx}h(x) \cdot g(x)u \\ &= b_1(x) + a_1(x)u\end{aligned}$$

- Now if $a_1(x) \neq 0$ for each $x \in \mathbb{R}^n$ then the controller

$$u(x, v) = \frac{1}{a_1(x)}[-b_1(x) + v]$$

If this is zero the controller is undefined

yields

$$\dot{y} = v$$

Calculating a Linearizing Controller

- Now if $a_1(x) \equiv 0$ then we differentiate y a second time and obtain and expression of the form

$$\ddot{y} = b_2(x) + a_2(x)u$$

- Now if $a_2(x) \neq 0$ for each $x \in \mathbb{R}^n$ then the control law

$$u(x, v) = \frac{1}{a_2(x)}[-b_2(x) + v]$$

yields

$$\ddot{y} = v$$

Calculating a Linearizing Controller

- In general, we can keep differentiating y until the input appears:

$$y^{\gamma} = \beta_{\gamma}(x) + \underbrace{\alpha_{\gamma}(x)}_{\neq 0} u$$

- At this point we can apply the control

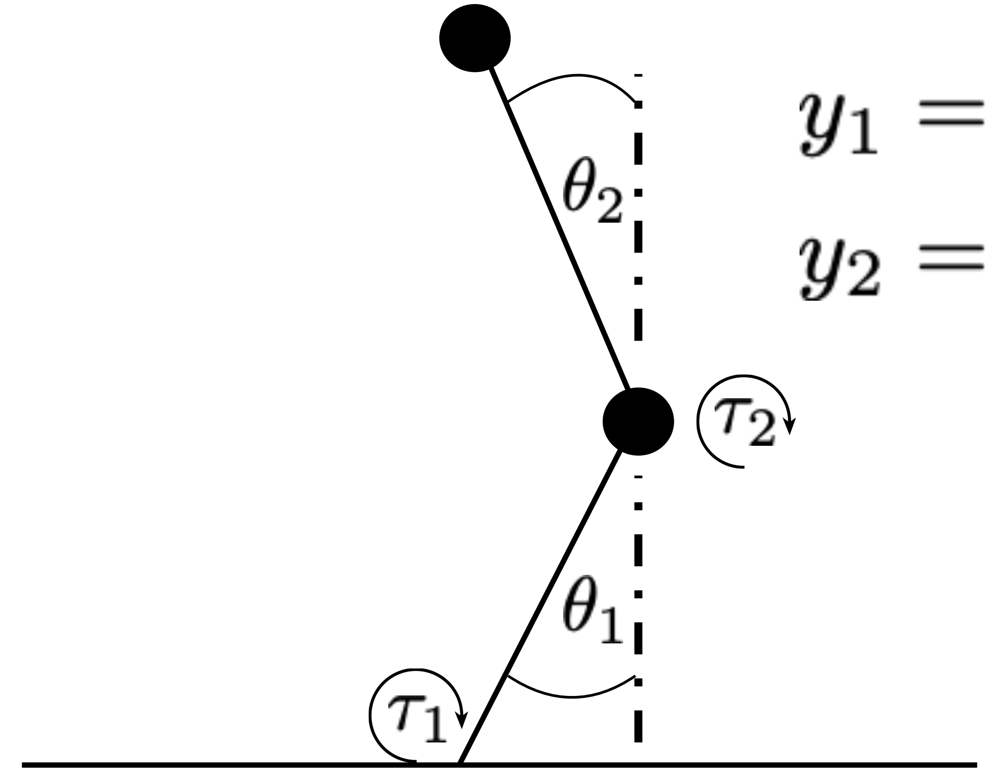
$$u(x,v) = \frac{1}{a_{\gamma}(x)}[-b_{\gamma}(x) + v]$$

which yields

$$y^{\gamma} = v$$

‘Inverting’ the Dynamics

- Take time derivatives of outputs to obtain an input-output relationship of the form



$$y_1 = \theta_1$$
$$y_2 = \theta_2$$

$$\begin{bmatrix} y_1^{(\gamma_1)} \\ \vdots \\ y_q^{(\gamma_q)} \end{bmatrix} = b(x) + A(x)u$$

- Applying the control law $u = A^{-1}(x)[-b(x) + v]$ yields

$$y^{(\gamma)} \triangleq \begin{bmatrix} y_1^{(\gamma_1)} \\ \vdots \\ y_q^{(\gamma_q)} \end{bmatrix} = \begin{bmatrix} v_1 \\ \vdots \\ v_q \end{bmatrix}$$

$$\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_q)$$

Vector Relative Degree

Normal Form

- Choose the outputs and their derivatives as new states for the system:

$$\xi = (y_1, \dot{y}_1, \ldots, y_1^{(\gamma_1 - 1)}, \ldots, y_q, \ldots, y_q^{(\gamma_q - 1)}) \in \mathbb{R}^{|\gamma|}$$

- If $|\gamma| < n$ we can 'complete the basis' by appropriately selecting $\eta \in \mathbb{R}^{n-|\gamma|}$ extra variables:

$$\dot{\xi} = A\xi + Bv \longleftarrow$$ Can track $y_d(\cdot)$ using linear control

$$\dot{\eta} = q(\xi, \eta) + p(\xi, \eta)v \longleftarrow$$ May become unstable!

$$\dot{\eta} = q(0, \eta) \longleftarrow$$ <u>Zero Dynamics</u> (Systems is <u>minimum-phase</u> if these are asymptotically stable)

Zero Dynamics

- We refer to the un-driven dynamics

$$\dot{\eta} = q(0, \eta)$$

as the **zero dynamics.**

- We say that the overall control system is **minimum-phase** if the zero dynamics are asymptotically stable

- We say that the system is **non-minimum-phase** if the zero dynamics are unstable

Tracking Desired Outputs

- To track the desired output

$$v = y_d^{(\gamma)} + K(\xi - \xi_d)$$

Feedforward Term **Feedback Term**

- If we design K such that $(A + BK)$ is Hurwitz then this control law drives $\xi(t) \to \xi_d(t)$ exponentially quickly

- However, the zero dynamics may not stay stable!

Model Mismatch

- Suppose we have an approximate dynamics model:

$$\dot{x} = f_m(x) + g_m(x)u \qquad \dot{x} = f_p(x) + g_p(x)u$$
$$y = h(x) \qquad y = h(x)$$

- Why not just learn the forward dynamics?

$$f_p(x) \approx \hat{f}_\theta(x)$$
$$g_p(x) \approx \hat{g}_\theta(x)$$

$$\Rightarrow \quad y^{(\gamma)} \approx \hat{b}_\theta(x) + \hat{A}_\theta(x)u$$

May be singular!

Directly Learning the Linearizing Controller

- We know the linearizing controllers are of the form

$$u_p(x,v) = \beta_p(x) + \alpha_p(x)v \qquad\qquad u_m(x,v) = \beta_m(x) + \alpha_m(x)v$$

- There is a "gap" between the two controllers:

$$u_p(x,v) = [\beta_m(x) + \Delta\beta(x)] + [\alpha_m(x) + \Delta\alpha(x)]v$$

- To overcome the gap we approximate

$$u_p(x,v) \approx \hat{u}_\theta(x,v) = [\beta_m(x) + \beta_{\theta_1}(x)] + [\alpha_m(x) + \alpha_\theta(x)]v$$

Model Based Components   Learned Components

"Feedback linearization for uncertain systems via RL" [**W**FMAPST] (2020)

Penalize Deviations from Desired Linear Behavior

- We want to find a set of learned parameter such that

$$y^{(\gamma)} = b_p(x) + A_p(x)\hat{u}_\theta(x,v) \approx v \qquad \forall x \in D, \forall v \in \mathbb{R}^q$$

- Thus, we define the point-wise loss

$$\ell(x,v,\theta) = \|(y_1^{\gamma_1}, \ldots, y_q^{\gamma_q})^T - (v_1, \ldots, v_q)^T\|_2^2$$

- We then define the optimization problem

$$\min_{\theta \in \Theta} \mathbb{E}_{x \sim X, v \sim V}[\ell(x,v,\theta)] \qquad (\mathbf{P})$$

Distribution of states

Distribution of virtual inputs

Solutions to the Problem

Theorem: [1] Assume that the learned controller is of the form

$$\beta_{\theta_1}(x) = \sum_{k=1}^{K_1} \theta_1^k \beta_k(x) \qquad \alpha_{\theta_2}(x) = \sum_{k=1}^{K_2} \theta_2^k \alpha_k(x)$$

where $\{\beta_k\}_{k=1}^{K_1}$ and $\{\alpha_k\}_{k=1}^{K_2}$ are linearly independent sets of features. Then the optimization problem $\mathbf{P}$ is strongly convex.

Corollary: Further assume that $u_p(x,v) \equiv \hat{u}_{\theta*}(x,v) \quad \forall x \in D, \forall v \in \mathbb{R}^q$ for some feasible $\theta^* \in \Theta$. Then θ^* is the unique optimizer for $\mathbf{P}$.

Remark: There are many known bases which can recover any continuous function up to a desired accuracy (e.g radial basis functions).

.

Discrete-Time Approximations with Reinforcement Learning

- In practice, we use a discretized version of the reward as a running cost in an RL problem:

Finite difference approximate to ℓ

$$\min_{\theta \in \Theta} \mathbb{E}_{x_0 \sim X, v_k \sim V, w_k \sim W} [\sum_{k=1}^{N} \bar{\ell}(x_k, v_k, u_k)]$$

$$x_{k+1} = x_k + \int_{t_k}^{t_{k+1}} f_p(x(t)) + g_p(x(t)) u_k dt$$

$$u_k = \hat{u}_\theta(x_k, v_k) + w_k$$

Gaussian noise added for exploration, enables use of policy gradient algorithms

12D Quadrotor Model

- Nominal dynamics model:

$$\ddot{x} = -\frac{u_1}{m}[\sin(\phi)\sin(\psi) + \cos(\phi)\cos(\psi)\cos(\theta)]$$

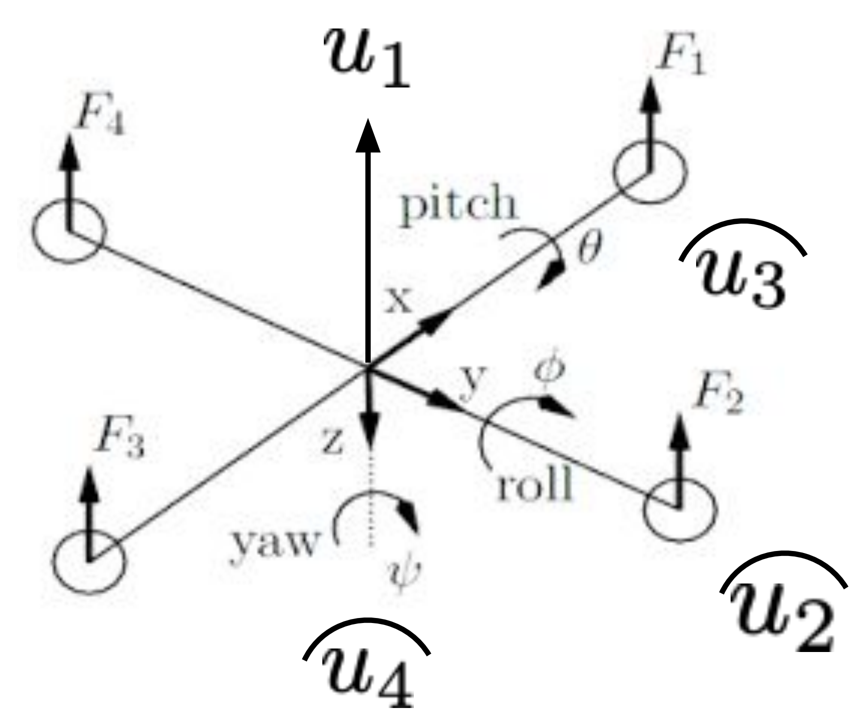
$$\ddot{y} = -\frac{u_1}{m}[\cos(\phi)\sin(\psi)\sin(\theta) - \cos(\psi)\sin(\phi)]$$

$$\ddot{z} = g - \frac{u_1}{m}[\cos(\phi)\cos(\theta)]$$

$$\ddot{\phi} = \frac{I_y - I_z}{I_x}\dot{\theta}\dot{\psi} + \frac{u_2}{I_x}$$

$$\ddot{\theta} = \frac{I_z - I_x}{I_y}\dot{\phi}\dot{\psi} + \frac{u_3}{I_y}$$

$$\ddot{\psi} = \frac{I_x - I_x}{I_z}\dot{\psi}\dot{\theta} + \frac{u_4}{I_z}$$



Choose Outputs

$$(x, y, z, \psi)$$

After Feedback Linearization:

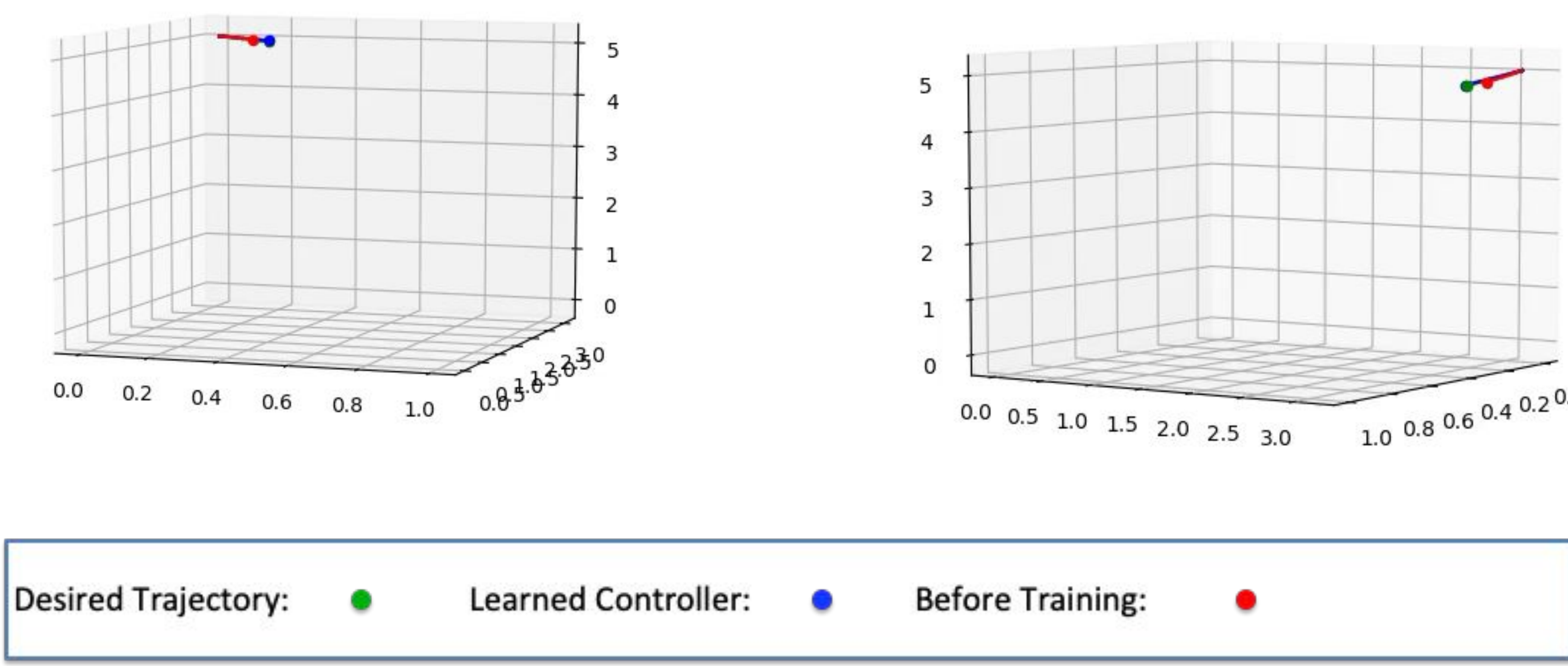
$$x^{(4)} = v_1$$

$$y^{(4)} = v_2$$

$$z^{(4)} = v_3$$

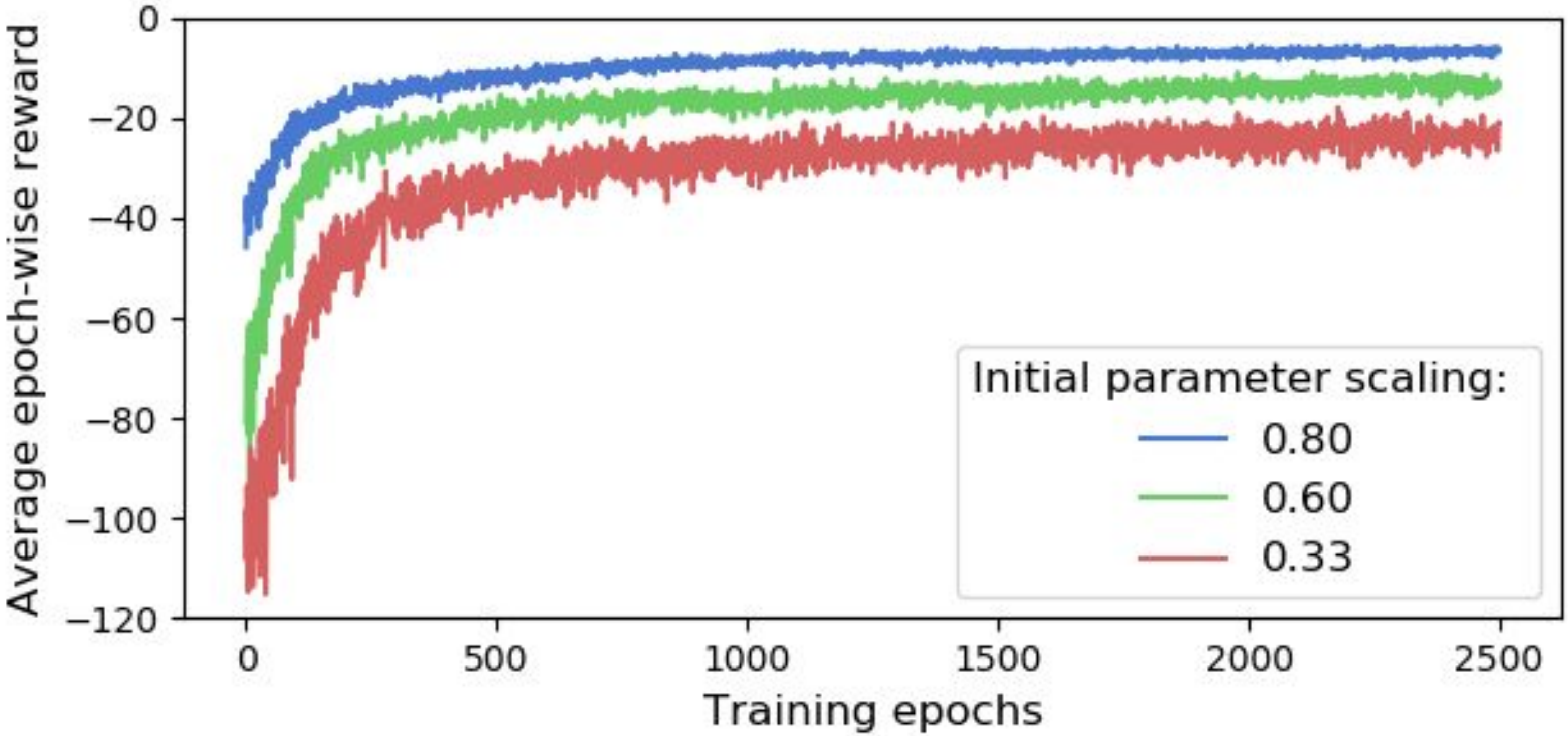
$$\psi^{(2)} = v_4$$

Improvement After ~1 Hour of Data

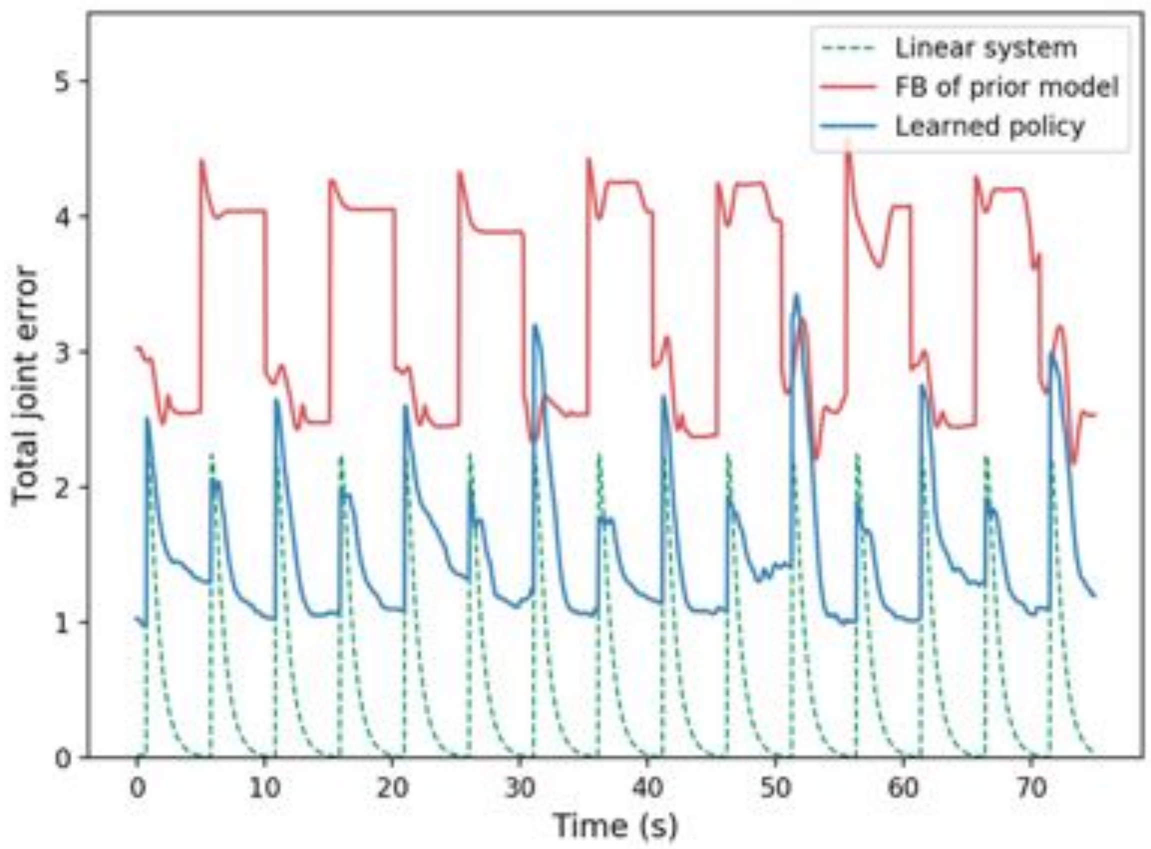


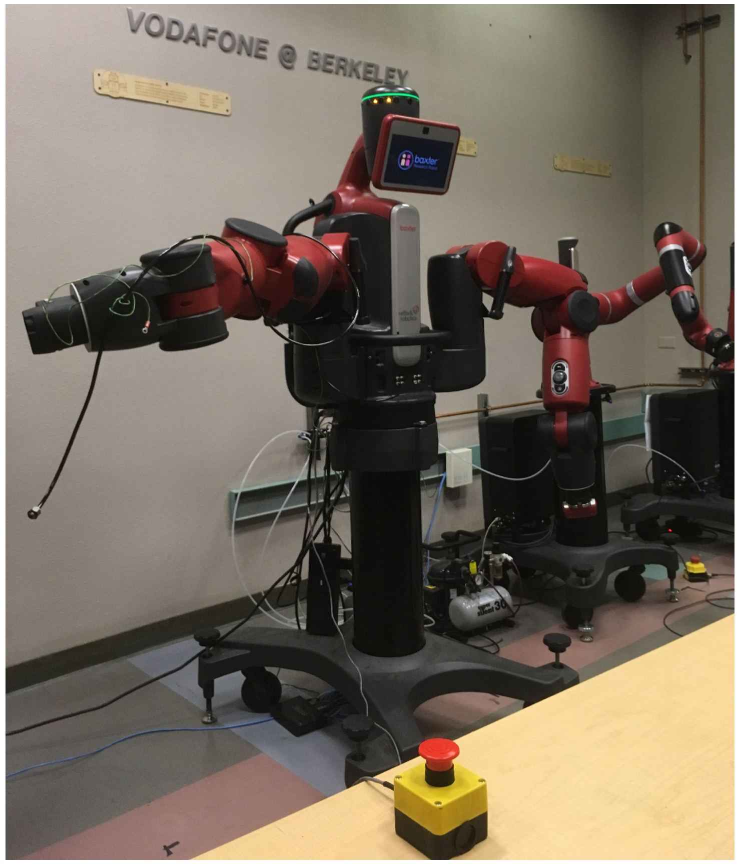
"Proximal Policy Optimization Algorithms" [Schulman et. al.] (2017)

Effects of Model Accuracy



7-DOF Baxter Arm After ~1 Hour of Data

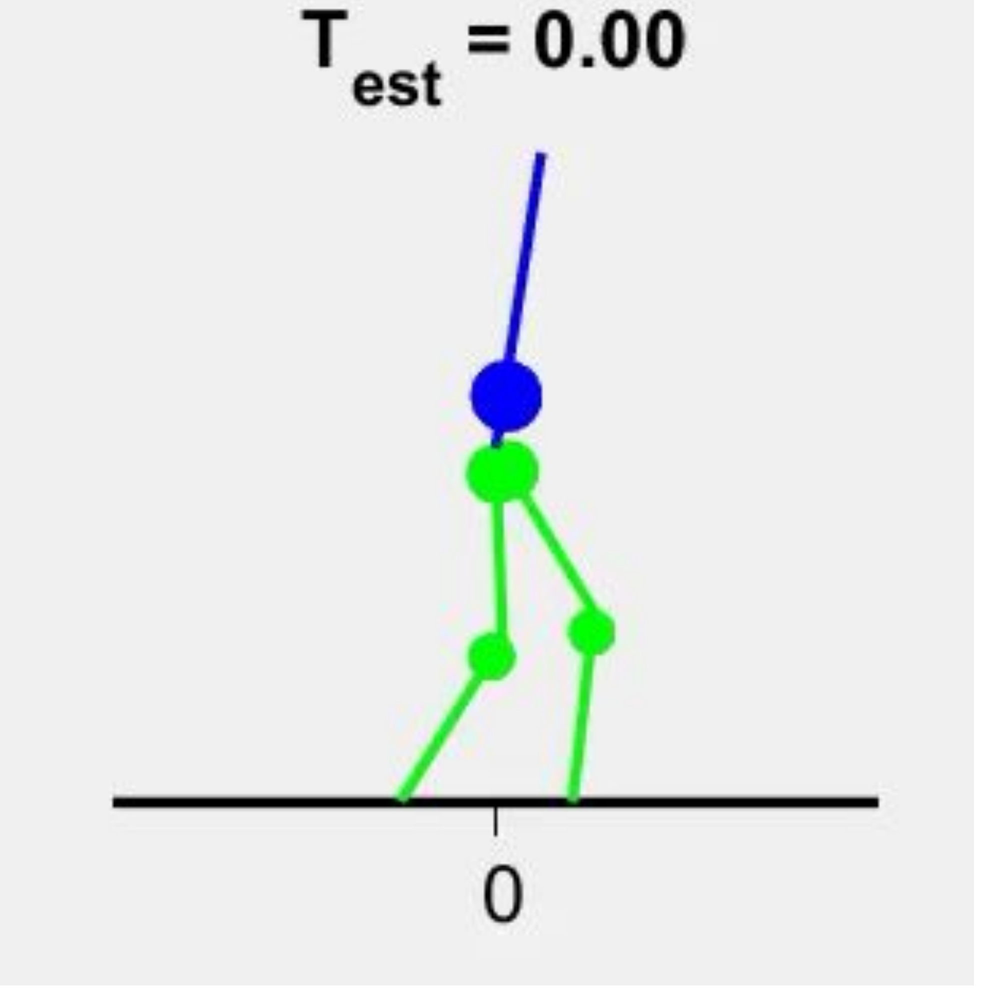




Learning a Stable Walking Gait in ~20 Minutes

- Feedback linearization is commonly used to design stable walking gates for bipedal robots

- Outputs are carefully designed so that zero dynamics generated a stable walking gate



"Improving I-O Linearizing Controllers for Bipedal Robots Via RL" [CWAWTSS] (2020)

"Continuous Control With Deep Reinforcement Learning" [Lillicrap et. al.] (2015)

Control Lyapunov Functions

Generalized ‘Energy’ Functions

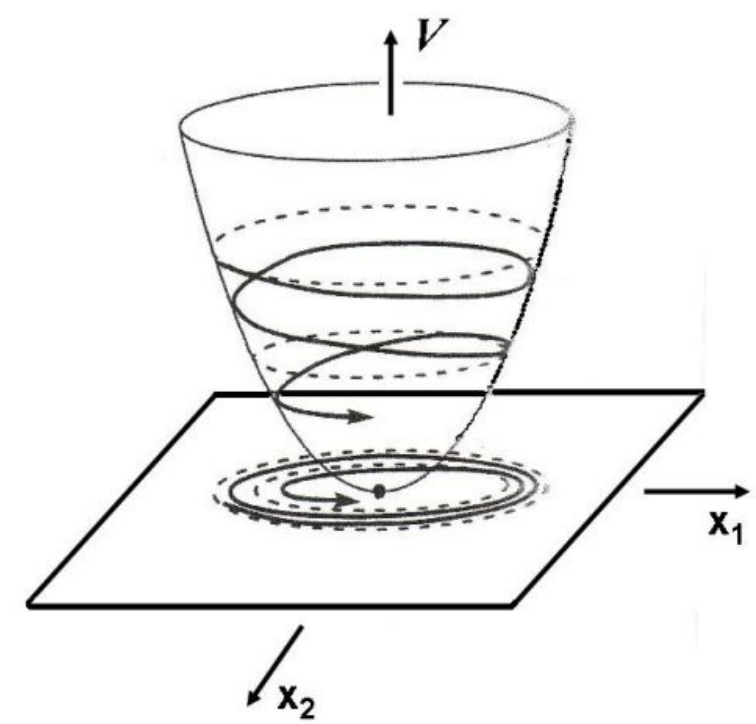
- Consider the plant

$$\dot{x} = f_p(x) + g_p(x)u$$

- We say that the positive definite function $V : \mathbb{R}^n \to \mathbb{R}$ is a **control Lyapunov function (CLF)** for the system if $\forall x \in \mathbb{R}^n$

$$\inf_{u \in U} \nabla V(x)[f_p(x) + g_p(x)u] \leq -\sigma(x)$$

User-specified energy dissipation rate



Learning Min-norm Stabilizing Controllers

- Given a Control Lyapunov Function $V : \mathbb{R}^n \to \mathbb{R}$ the associated min-norm controller for the plant is given by

$$u^*(x) = \min_{u \in U} \quad \|u\|_2^2$$
$$\text{s.t. } \nabla V(x)[f_p(x) + g_p(x)u] + \sigma(x) \leq 0$$

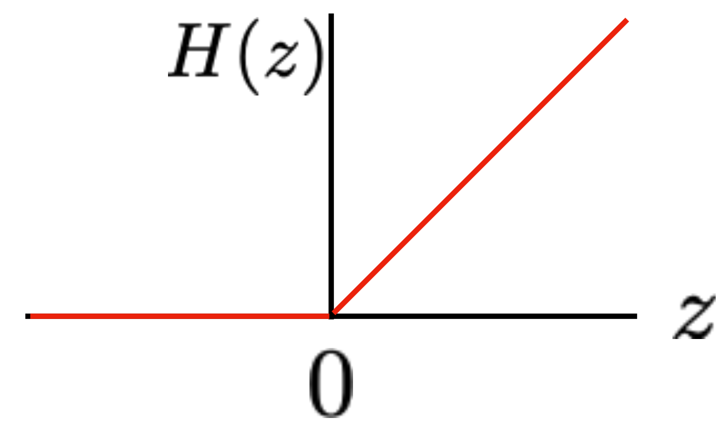
- To learn the min-norm controller we want to solve:

$$\min_{\theta \in \Theta} \quad \mathbb{E}_{x \sim X} \|\hat{u}_\theta(x)\|_2^2$$
$$\text{s.t. } \underbrace{\nabla V(x)[f_p(x) + g_p(x)\hat{u}_\theta(x)] + \sigma(x)}_{:=\Delta(x,\theta)} \leq 0, \ \forall x$$

Penalizing the Constraint

- To remove the constraint we add a penalty term to the cost:

$$(\mathbf{P}^{\lambda}): \min_{\theta \in \Theta} \mathbb{E}_{x \sim X} \left[\|\hat{u}_{\theta}(x)\|_2^2 + \underbrace{\lambda}_{\text{scaling parameter}} \underbrace{H(\Delta(x, \theta))}_{\text{penalty function}} \right]$$



$\lambda \geq 0$ $\qquad (\mathbf{P}^{\lambda})$

- If the controller in linear in its parameters $(\mathbf{P}^{\lambda})$ is strongly convex, under the additional assumption that $U = \mathbb{R}^q$

"Learning Min-norm Stabilizing Control Laws for systems with Unknown Dynamics" [**W**CASS] (CDC 2020, *To Appear*)

Learning the 'Forward' Terms

- Other approaches estimate the terms in the constraint [1][2]:

$$\underbrace{\nabla V(x) f_p(x)}_{\approx \hat{a}_\theta(x)} + \underbrace{\nabla V(x) g_p(x)}_{\approx \hat{b}_\theta(x)} u \leq \sigma(x)$$

- Then incorporate into QP:

$$u^*(x) \approx \arg\min_{u \in U} \ \|u\|_2^2$$
$$\text{s.t.} \quad \hat{a}_\theta(x) + \hat{b}_\theta(x) u \leq -\sigma(x)$$

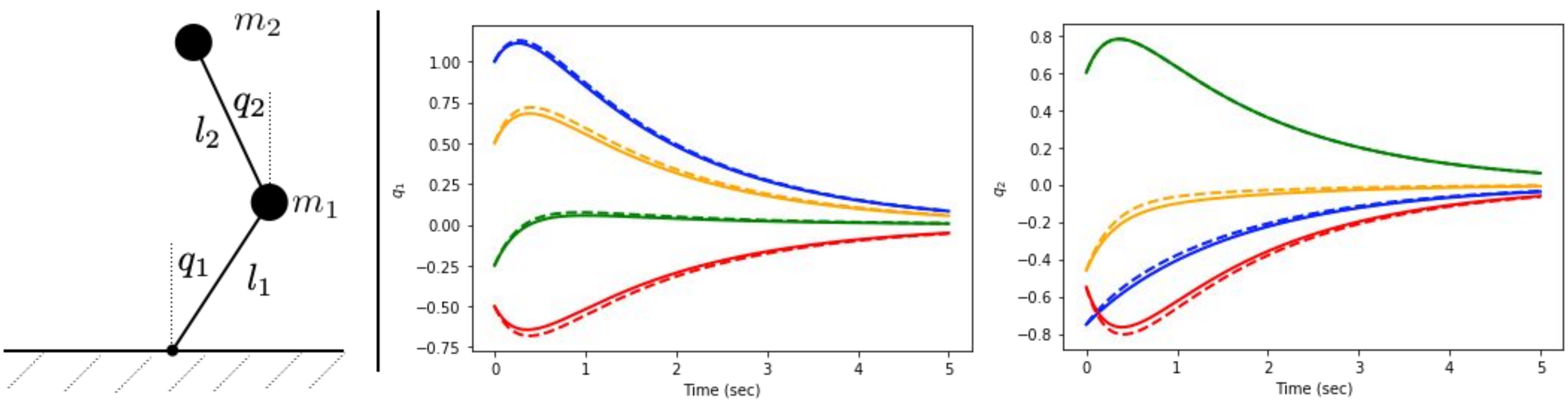
Advantages of our approach:

- Faster update rates for learned controller
- Learned controller always 'feasible'
- Does not require implicit 'inversion' of learned terms:

$$u^*(x) \approx \begin{cases} 0 & \text{if } \hat{a}_\theta(x) \leq -\sigma(x) \\ -\frac{[\hat{a}_\theta(x) + \sigma(x)](\hat{b}_\theta(x))^T}{\langle \hat{b}_\theta(x), \hat{b}_\theta(x) \rangle} & \text{else} \end{cases}$$

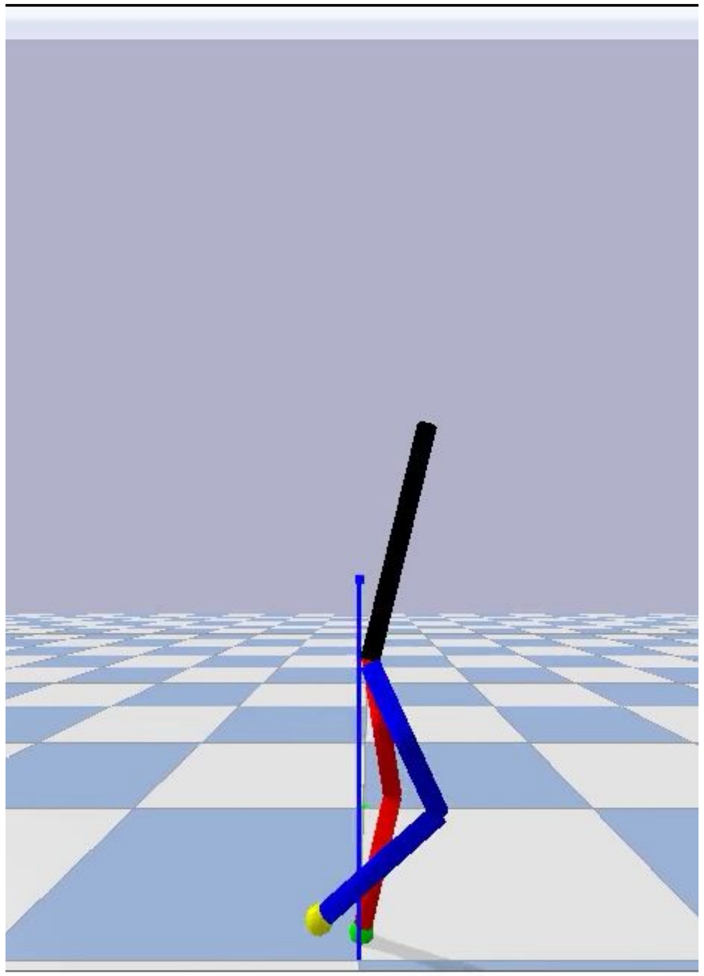
[Choi et. al] (2020) [Taylor et. al] (2019)

Double Pendulum ~ 4 Minutes of Data

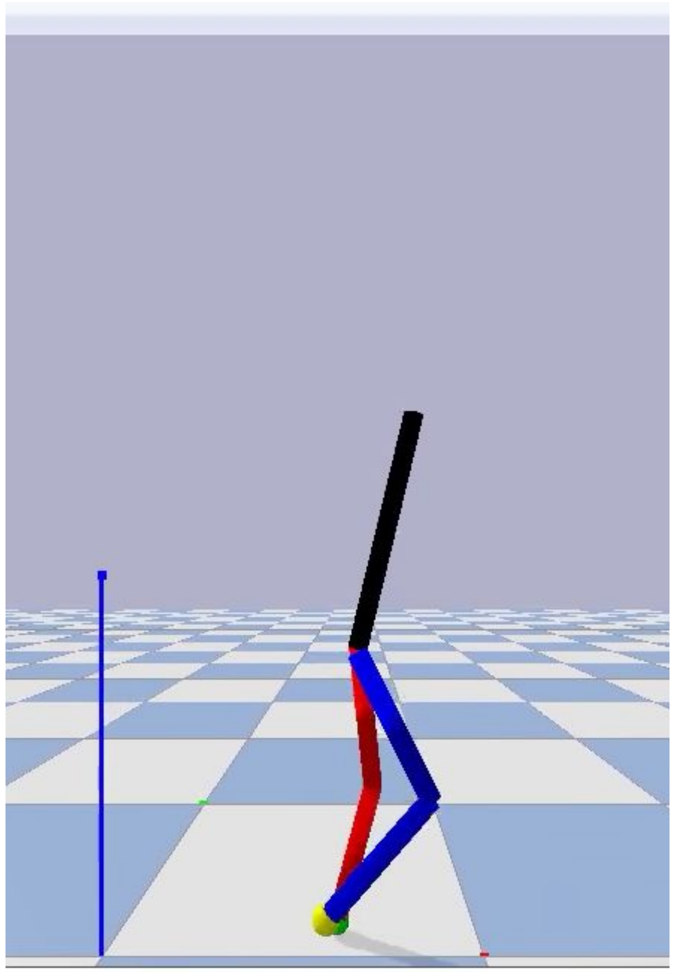


(a) (b)

Learning a Stable Walking Controller With ~20 Seconds of Data



Nominal Controller



Learned Controller

"Soft Actor-Critic: Off-Policy Maximum Entropy Deep RL with a Stochastic Actor: [Haarnoja et. al.] (2018)

Steps in Design Process

Step 1: Choose geometric control architecture which produces desired global behavior

$$\dot{x} = f_m(x) + g_m(x)u_m(x)$$

e.g. feedback linearizing controller

Step 2: Augment the nominal controller with a learned component:

$$\hat{u}_\theta(x) = u_m(x) + \Delta u_\theta(x)$$

learned augmentation

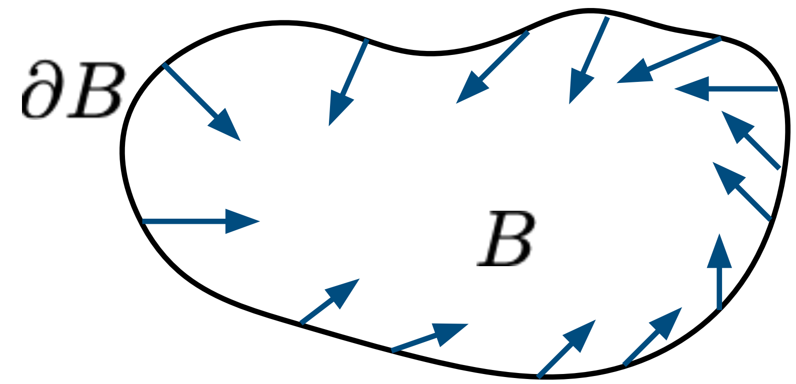
Step 3: Formulate reward which captures desired local behavior

$$\min_{\theta \in \Theta} \mathbb{E}_{x \sim X} \ell(x, \theta)$$

Minimize loss with RL

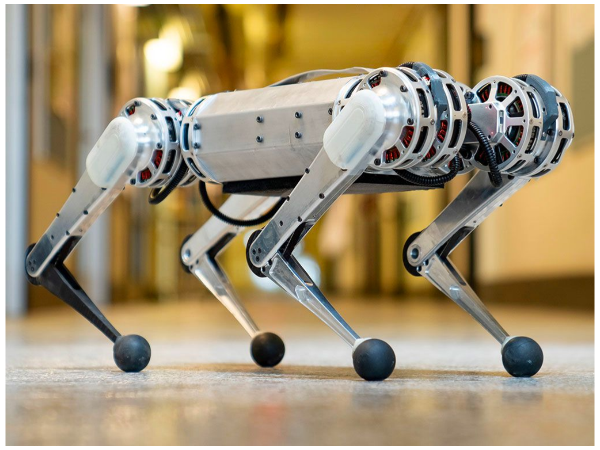
Specific Architectures

- Control Barrier Functions



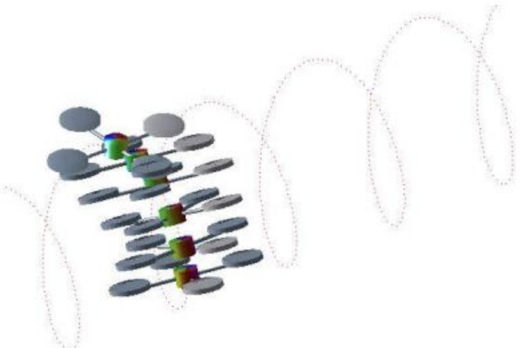
[Ames et. al.] (2019)

- Time Varying CLFs



[Kim et. al.] (2019)

- Geometric Controllers on $SE(3)$



[Lee et. al.] (2010)

Trade-offs With ‘Model-Based’ RL

- Mb-RL: learn a neural network dynamics model from scratch, use for online planning or training controllers offline with model-free RL

[Nagabandi et. al.] (2018)

Main Advantage of Mb-RL:

- Can be used when ‘ideal’ control architecture is not known

Advantages of our Approach:

- Fine grain control over system behavior

Key Take Aways

- Connecting local and global geometric structure allows us to efficiently overcome model uncertainty
- Learning a forward dynamics model may be incompatible with geometric control

Relevant Papers

- “Feedback Linearization for Uncertain Systems via Reinforcement Learning” [**W**FMAPST] (ICRA 2020)
- “Improving Input-Output Linearizing Controllers for Bipedal Robots Via Reinforcement Learning” [CWA**W**TSS] (L4DC 2020)
- “Learning Min-norm Stabilizing Control Laws for systems with Unknown Dynamics” [**W**CASS] (IEEE, CDC 2020, *Dec. 2020*)
- “Learning Feedback Linearizing Controllers with Reinforcement Learning” [**W**FPMST] (IJRR, *In Prep*)
- “Directly Learning Safe Controllers with Control Barrier Functions” (TBD)
- “Learning Time-based Stabilizing Controllers for Quadrupedal Locomotion” (TBD)

Current Work + Extensions

Can we use model-free policy optimization to overcome model mismatch in high dimensions for specific control architectures?

Can we use geometric control to systematically design rewards which are ‘easy’ to optimize over, and achieve the desired objective?

Control Architectures

- Feedback Linearization
- CLFs + CBFs
- Other Control Architectures

Combining Learning and Adaptive Control

- Probabilistic Safety Guarantees
- Choice of Learning Algorithm

Project Flow

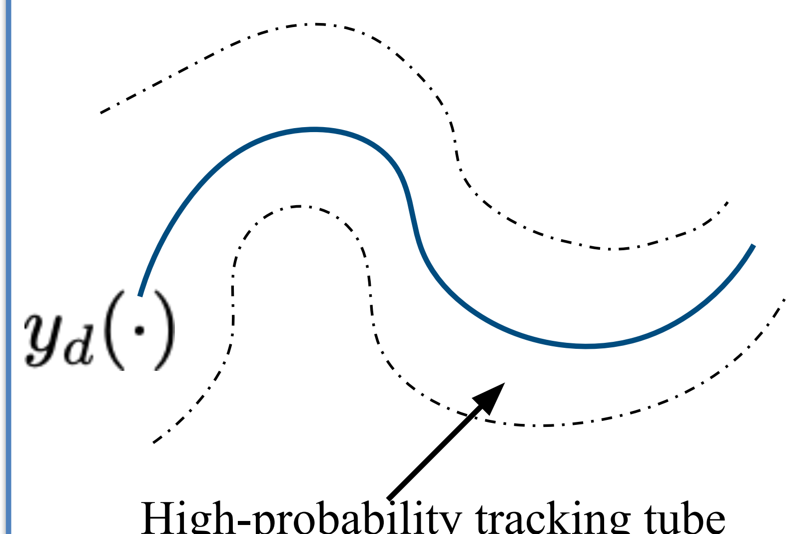
Part 1: **Overcome non-parametric uncertainty by combining RL and geometric control**

$$\hat{u}_\theta(x) = u_m(x) + \Delta u_\theta(x)$$

MB Controller

Learned Correction

Part 2: **Provide correctness and safety guarantees for specific learning algorithms:**



Part 3: **Future work:**

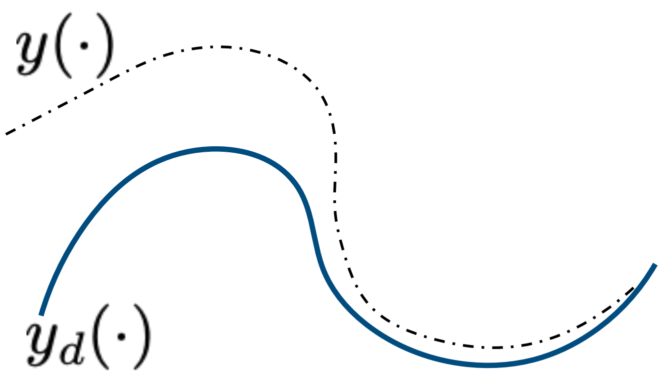
- **What makes a reward signal difficult to learn from?**
- **What makes a system fundamentally difficult to control?**
- **Where should geometric control be used in the long-run?**

Part 2 Outline

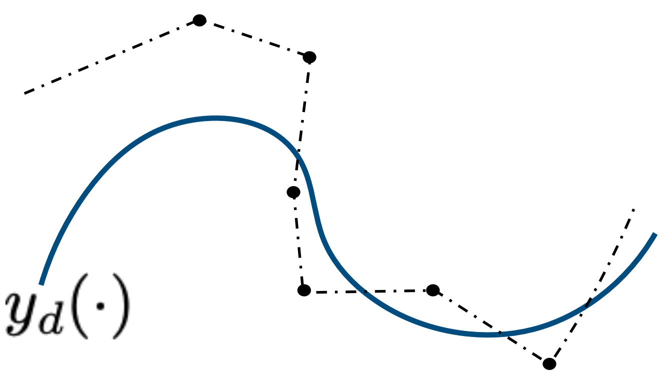
- Goal: show that we can safely learn a linearizing controller online using standard RL algorithms
  - Provide probabilistic tracking tracking bounds for overall learning system
    - Simple policy gradient algorithms
    - More sophisticated algorithms (Future Work)
  - Comparison with ‘model-based’ adaptive control

Analysis and Design Steps

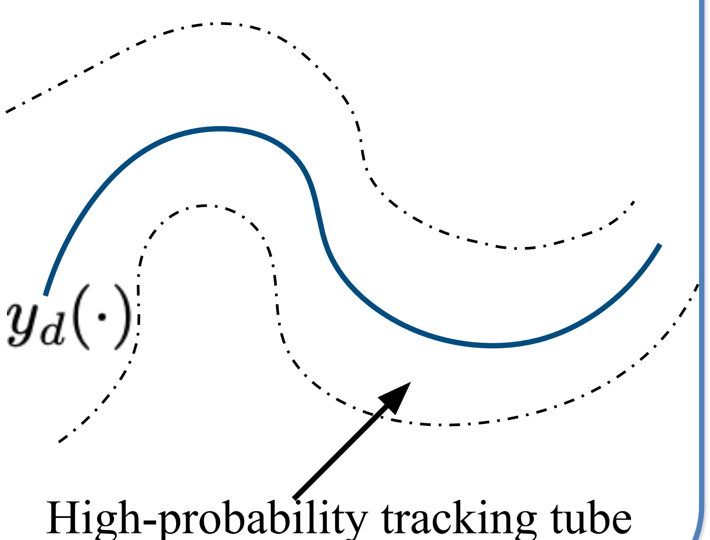
<u>Step 1:</u> Use our loss function from before to design an 'ideal' CT update rule



<u>**Step 2:**</u> Model DT model-free policy gradient algorithms as noisy discretization of the CT process



<u>Step 3:</u> Provide probabilistic safety guarantees for the overall learning system

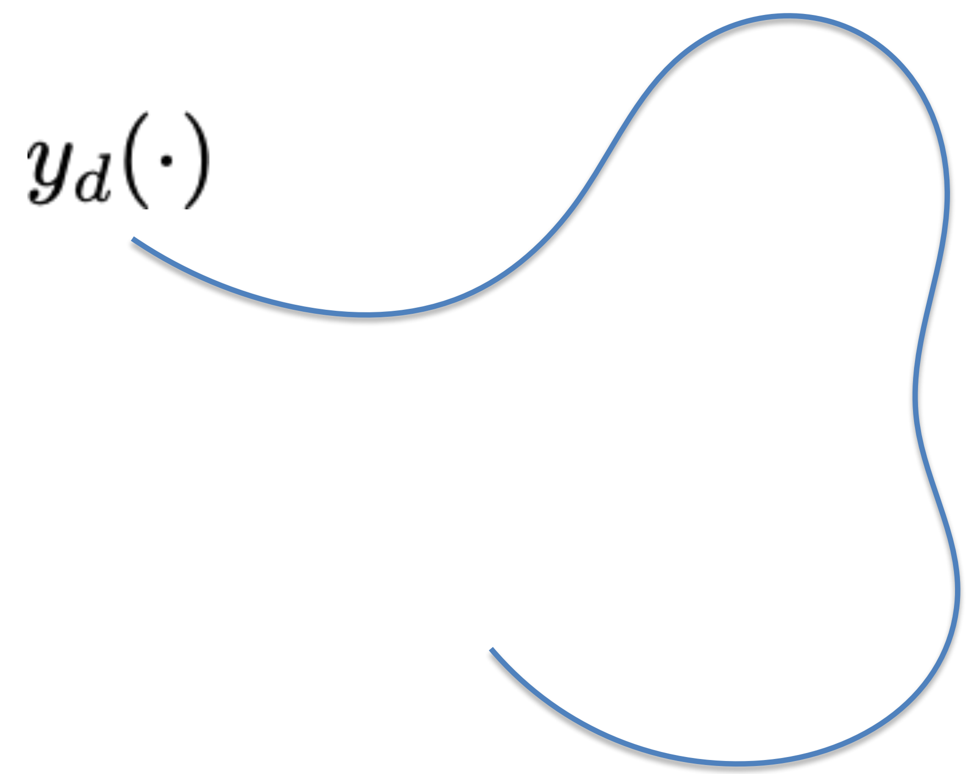


Modeling Learning as CT Process

- Goal: track a desired trajectory while improving estimated parameters

$y_d(\cdot)$

$\theta(\cdot)$



Modeling Learning as CT Process

- Goal: track a desired trajectory $y_d(\cdot)$ while improving estimated $\theta(\cdot)$ parameters

Recall the normal form:

$$\dot{\xi} = A\xi + Bv$$

$$\dot{\eta} = q(\xi, \eta) + p(\xi, \eta)v$$

- Apply estimated controller

$$u = \hat{u}_{\hat{\theta}}(x, y_d^{(\gamma)} + Ke)$$

$$(\xi - \xi_d)$$

Modeling Learning as CT Process

- Goal: track a desired trajectory $y_d(\cdot)$ while improving estimated $\theta(\cdot)$ parameters

Recall the normal form:

$$\dot{\xi} = A\xi + Bv$$
$$\dot{\eta} = q(\xi, \eta) + p(\xi, \eta)v$$

- Apply estimated controller

$$u = \hat{u}_{\hat{\theta}}(x, y_d^{(\gamma)} + Ke)$$

$$(\xi - \xi_d)$$

Assumption: Controller is linear is parameters: $\beta_{\theta_1}(x) = \sum_{k=1}^{K_1} \theta_1^k \beta_k(x)$ $\alpha_{\theta_2}(x) = \sum_{k=1}^{K_2} \theta_2^k \alpha_k(x)$

Modeling Learning as CT Process

- Goal: track a desired trajectory $y_d(\cdot)$ while improving estimated parameters $\hat{\theta}(\cdot)$

Recall the normal form:

$$\dot{\xi} = A\xi + Bv$$
$$\dot{\eta} = q(\xi, \eta) + p(\xi, \eta)v$$

- Apply estimated controller

$$u = \hat{u}_{\hat{\theta}}(x, y_d^{(\gamma)} + Ke)$$

$$(\xi - \xi_d)$$

Assumption: Controller is linear is parameters: $\beta_{\theta_1}(x) = \sum_{k=1}^{K_1} \theta_1^k \beta_k(x) \quad \alpha_{\theta_2}(x) = \sum_{k=1}^{K_2} \theta_2^k \alpha_k(x)$

Assumption: There exists a unique set of parameters $u_p(x, v) \equiv \hat{u}_{\theta*}(x, v)$

$$\forall x \in D, \forall v \in \mathbb{R}^q$$

Modeling Learning as CT Process

- Goal: track a desired trajectory $y_d(\cdot)$ while improving estimated parameters $\theta(\cdot)$

Recall the normal form:

$$\dot{\xi} = A\xi + Bv$$
$$\dot{\eta} = q(\xi, \eta) + p(\xi, \eta)v$$

- Apply estimated controller

$$u = \hat{u}_{\hat{\theta}}(x, y_d^{(\gamma)}) + Ke$$

$$(\xi - \xi_d)$$

- Tracking Error Dynamics:

$$\dot{e} = (A + BK)e + W(t)\phi$$

$$(\hat{\theta} - \theta^*)$$

Assumption: Controller is linear is parameters:

$$\beta_{\theta_1}(x) = \sum_{k=1}^{K_1} \theta_1^k \beta_k(x) \quad \alpha_{\theta_2}(x) = \sum_{k=1}^{K_2} \theta_2^k \alpha_k(x)$$

Assumption: There exists

$$u_p(x, v) \equiv \hat{u}_{\theta *}(x, v)$$

$$\forall x \in D, \forall v \in \mathbb{R}^q$$

Modeling Learning as CT Process

- Goal: track a desired trajectory $y_d(\cdot)$ while improving estimated $\theta(\cdot)$ parameters

Assumption: Controller is linear is parameters:

$$\beta_{\theta_1}(x) = \sum_{k=1}^{K_1} \theta_1^k \beta_k(x) \qquad \alpha_{\theta_2}(x) = \sum_{k=1}^{K_2} \theta_2^k \alpha_k(x)$$

- CT reward function:

$$R(t) = \frac{1}{2}\|W(t)\phi(t)\|_2^2$$

- Ideal CT update rule:

$$\dot{\hat{\theta}} = \dot{\phi} = -W(t)^T W(t)\phi$$

Modeling Online Learning as CT Process

- Goal: track $y_d(\cdot)$ while improving estimated parameters $\theta(\cdot)$ and using the estimated controller

Assumption 1: Controller is linear in parameters:

$$\beta_{\theta_1}(x) = \sum_{k=1}^{K_1} \theta_1^k \beta_k(x) \qquad \alpha_{\theta_2}(x) = \sum_{k=1}^{K_2} \theta_2^k \alpha_k(x)$$

Assumption 2: There exists a unique $\theta^* \in \Theta$ s.t.

$$u_p(x,v) \equiv \hat{u}_{\theta *}(x,v) \quad \forall x \in D, \forall v \in \mathbb{R}^q$$

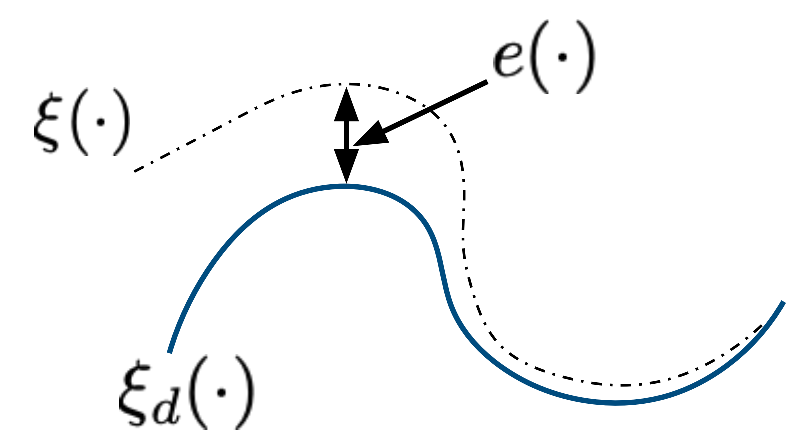
- We apply the 'ideal' update

$$\dot{\theta} = -\nabla_\theta \ell(x,v,\theta)$$

Least square loss from before

Define: $\phi(t) = \theta(t) - \theta^*$

- Under a persistency of excitation condition we show $\phi(t) \to 0$ exponentially quickly



Modeling Online Learning as CT Process

- Goal: track a desired trajectory $y_d(\cdot)$ while improving estimated $\theta(\cdot)$ parameters

Recall the normal form:

$$\dot{\xi} = A\xi + Bv$$
$$\dot{\eta} = q(\xi, \eta) + p(\xi, \eta)v$$

- Apply estimated controller

$$u = \hat{u}_{\hat{\theta}}(x, y_d^{(\gamma)}) + Ke$$
$$(\xi - \xi_d)$$

- Tracking Error Dynamics:

$$\dot{e} = (A + BK)e + W(t)\phi$$
$$(\hat{\theta} - \theta^*)$$

- CT reward function:

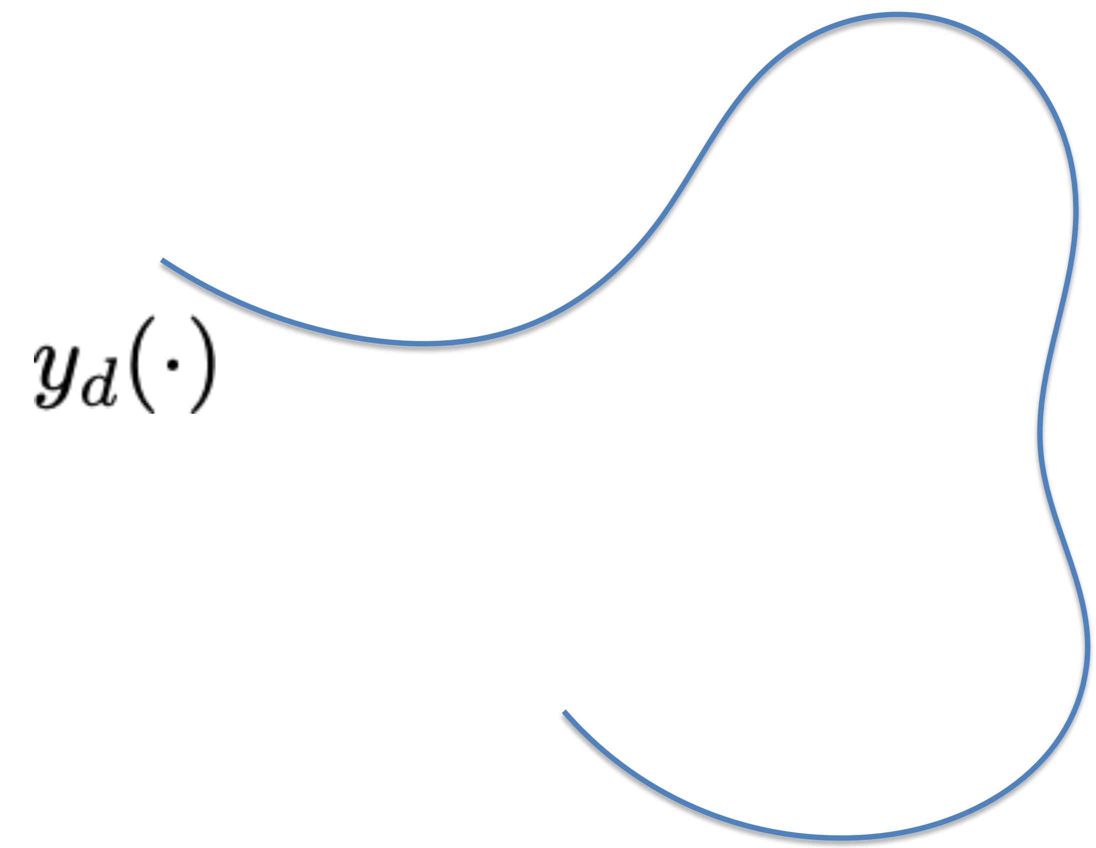
$$R(t) = \tfrac{1}{2}\|W(t)\phi(t)\|_2^2$$

- Ideal CT update rule:

$$\dot{\hat{\theta}} = \dot{\phi} = -W(t)^T W(t)\phi$$

Adaptive Control Approach

- Goal: track a desired trajectory while $y_d(\cdot)$ improving estimated parameters $\theta(\cdot)$



- Apply estimated tracking controller:

$$u = \hat{u}_{\hat{\theta}}(x, y_d^{(\gamma)} + Ke)$$

$$e = (\xi - \xi_d)$$

- Tracking error dynamics

CT Reward:

$$\dot{e} = (A + BK)e + W(t)\phi$$

$$\phi = (\hat{\theta} - \theta^*)$$

Persistency of Excitation

- We say that $W(t)$ is persistently exciting if

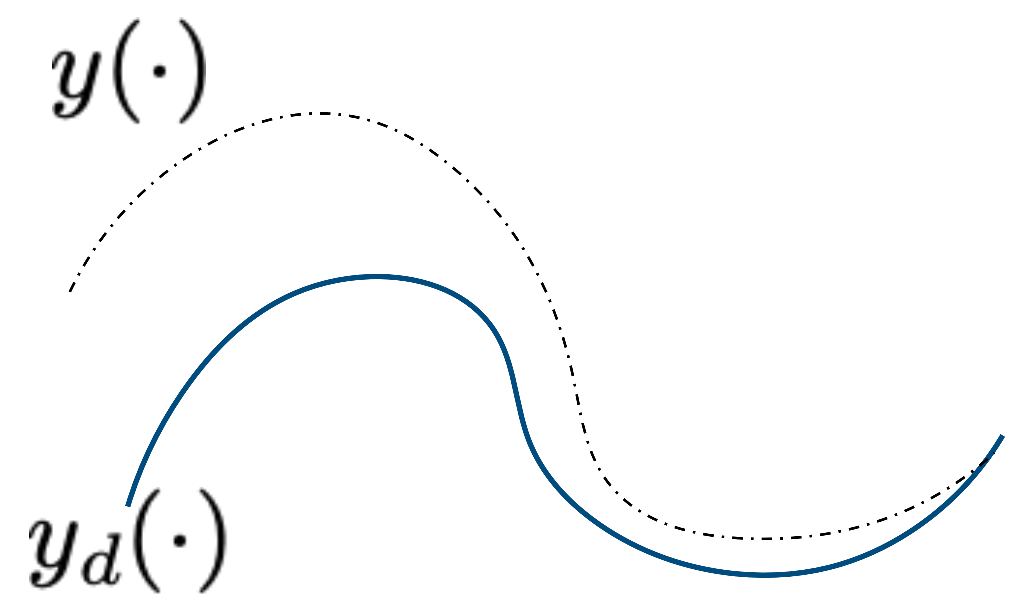
$$c_1 I < \int_t^{t+T} W(t)^T W(t)dt < c_2 I \qquad \forall t \geq 0$$

for some $c_1, c_2, T > 0$

- Under this condition we have

$$\phi(t) \to 0 \qquad\qquad e(t) \to 0$$

exponentially quickly as $t \to \infty$



Analyzing DT RL Algorithms

- On the interval $[t_k, t_{k+1})$ we apply the noisy control

$$u_k \sim \pi_k(\cdot|x_k, v_k, \theta_k) = \hat{u}_{\theta_k}(x_k, v_k) + W_k \qquad W_k \sim N(0, \sigma_k^2 I)$$

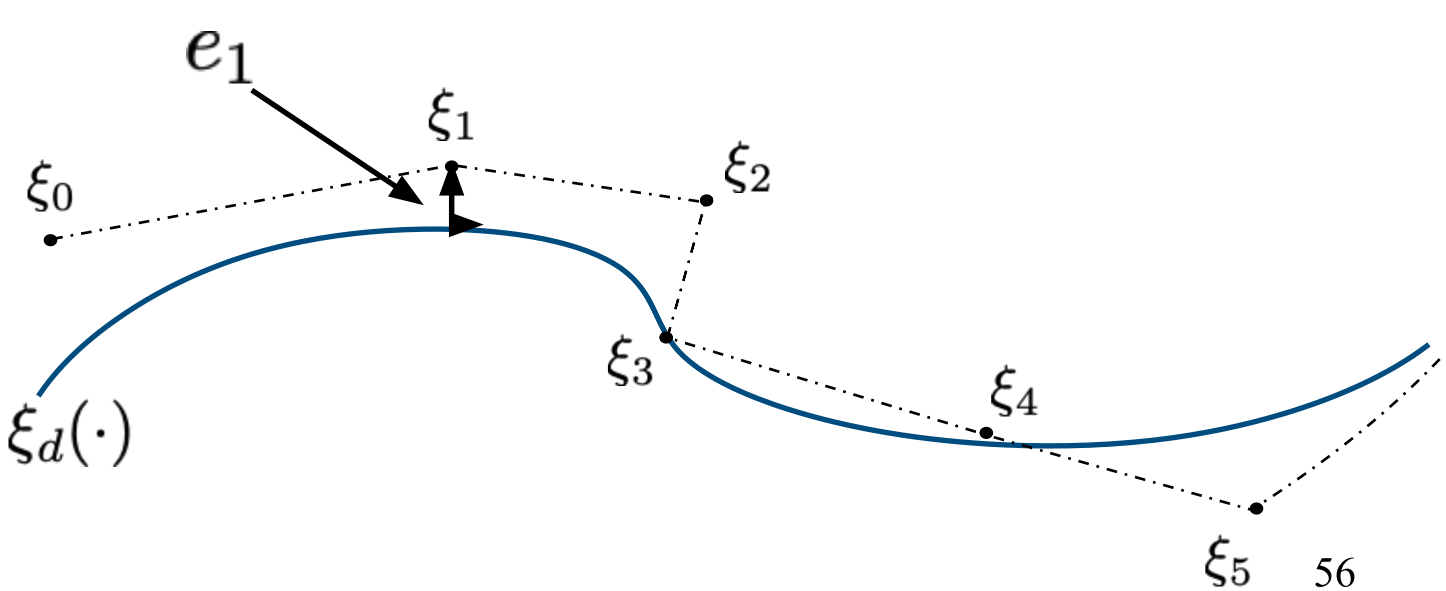
and apply noisy parameter updates of the form

$$\theta_{k+1} = \theta_k - \Delta t \alpha_k \hat{J}_k$$

Sampling Period

Learning Rate

Noisy Estimate of $\nabla_{\theta_k} \ell(x_k, v_k, \theta_k)$



$$e_k = \xi_k - \xi_d(t_k)$$
$$\phi_k = \theta_k - \theta^*$$

Implementable DT Stochastic Approximations

- Main idea: model standard policy gradient updates as (noisy) discretization of the ideal parameter update

- To explore the dynamics, $\forall t \in [t_k, t_{k+1})$ we apply the control

$$u_k \sim \pi_k(x_k, \theta_k) = \hat{u}_{\theta_k}(x_k, y_{d,k}^{(\gamma)} + Ke_k) + W_k, \quad W_k \sim N(0, \sigma_k^2 I)$$

Implementable DT Stochastic Approximations

- Main idea: model standard policy gradient updates as (noisy) discretization of the ideal parameter update
- To explore the dynamics, $\forall t \in [t_k, t_{k+1})$ we apply the control

$$u_k \sim \pi_k(x_k, \theta_k) = \hat{u}_{\theta_k}(x_k, y_{d,k}^{(\gamma)} + Ke_k) + W_k, \quad W_k \sim N(0, \sigma_k^2 I)$$

- This leads to a discrete time process of the form

$$e_{k+1} = e_k + \Delta t(A + BK)e_k + \Delta t W_k \phi_k + H_k(x_k, e_k, w_k)$$

$$\phi_{k+1} = \phi_k - \Delta t \alpha_k \hat{J}_k$$

Implementable DT Stochastic Approximations

- Main idea: model standard policy gradient updates as (noisy) discretization of the ideal parameter update
- To explore the dynamics, $\forall t \in [t_k, t_{k+1})$ we apply the control

$$u_k \sim \pi_k(x_k, \theta_k) = \hat{u}_{\theta_k}(x_k, y_{d,k}^{(\gamma)} + Ke_k) + W_k, \quad W_k \sim N(0, \sigma_k^2 I)$$

- This leads to a discrete time process of the form

$$e_{k+1} = e_k + \Delta t(A + BK)e_k + \Delta t W_k \phi_k + H_k(x_k, e_k, w_k)$$

$$\phi_{k+1} = \phi_k - \Delta t \alpha_k \hat{J}_k$$

Learning Rate

Estimate for gradient of $R(t_k)$

‘Vanilla’ Policy Gradient

- As a first step in analysis, we consider the simple policy gradient estimator:

$$\hat{J}_k = R_k \cdot \nabla_{\theta_k} \log(\pi_k(u_k|x_k, e_k, \theta_k))$$

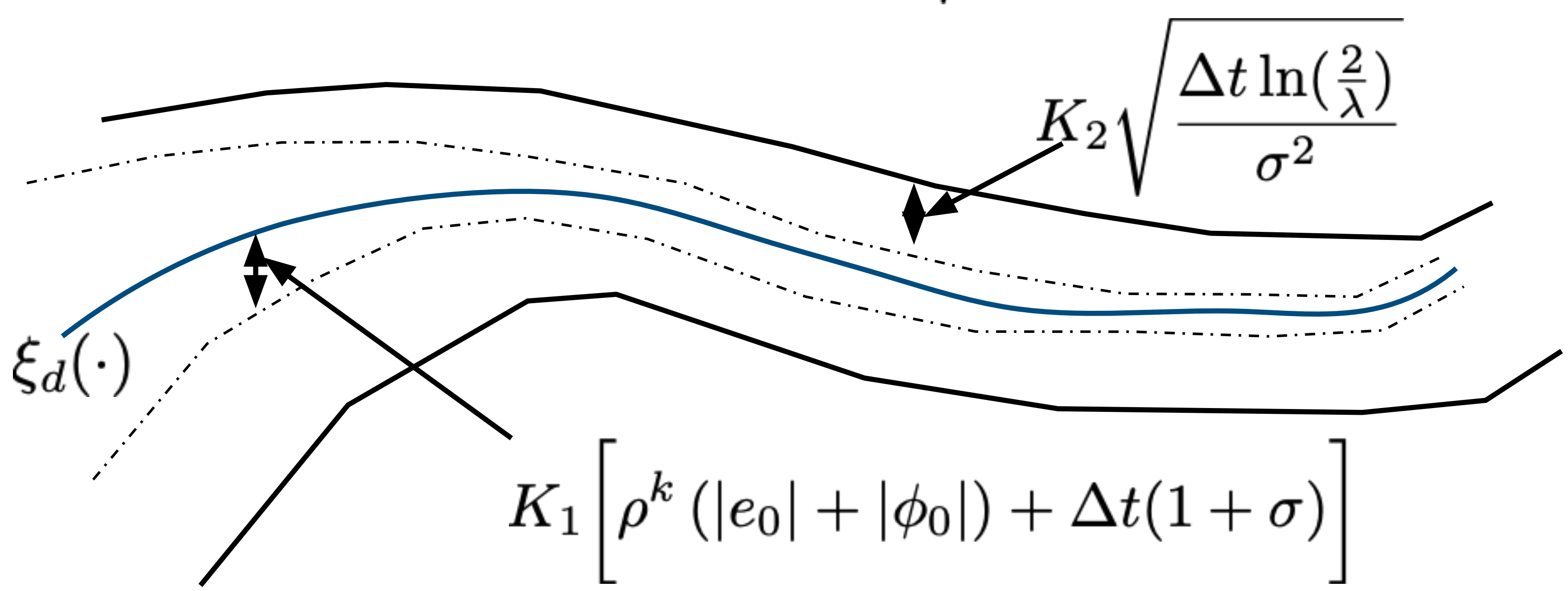
Convergence of ‘Vanilla Policy Gradient’

<u>Theorem:</u> For each $k \in \mathbb{N}$ put $\alpha_k = 1, \sigma_k = \sigma > 0$. Further assume the PE condition holds. Then there exists $0 < \rho < 1$ $K_1, K_2 > 0$ such that

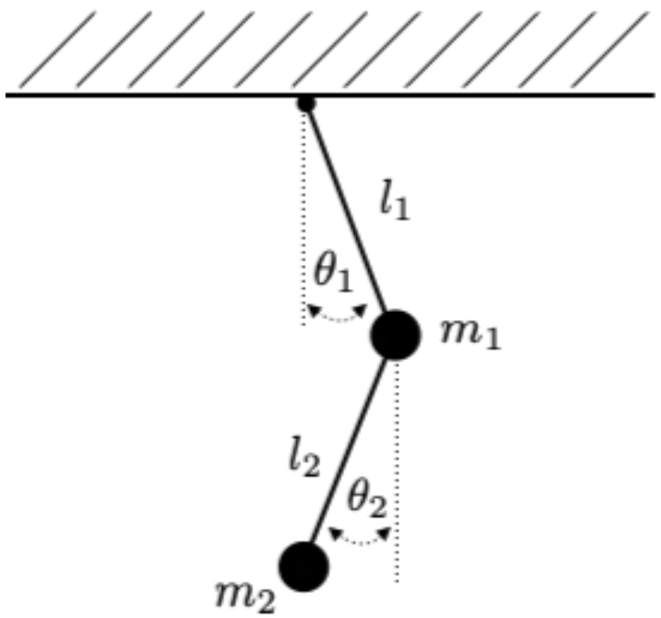
$$|\mathbb{E}[e_k]| \leq K_1 \left[\rho^k \left(|e_0| + |\phi_0|\right) + \Delta t(1+\sigma)\right]$$

and with probability $1 - \lambda$

$$|e_k - \mathbb{E}[e_k]| \leq K_2 \sqrt{\frac{\Delta t \ln(\frac{2}{\lambda})}{\sigma^2}}$$

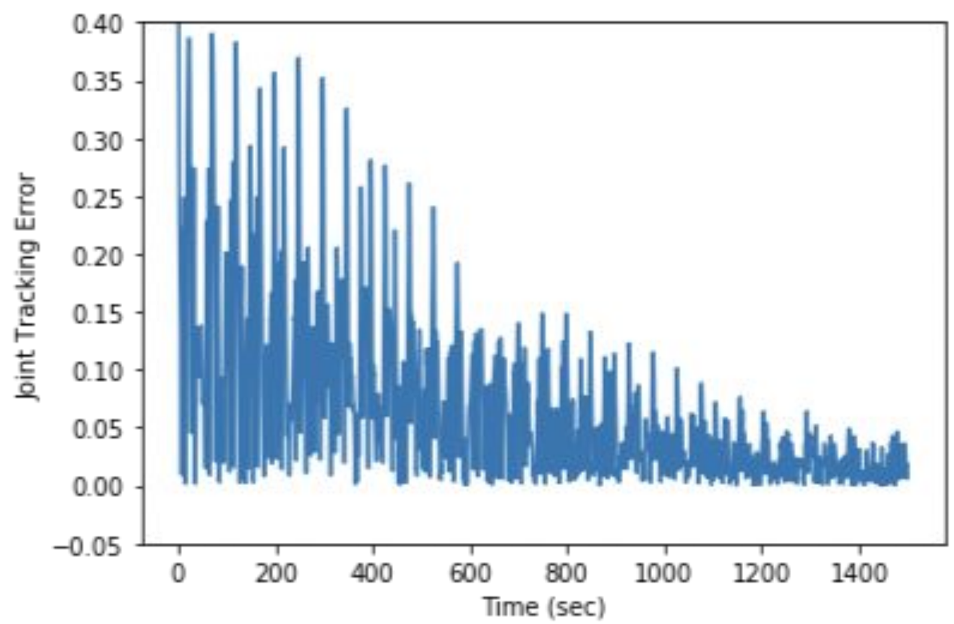


Double Pendulum



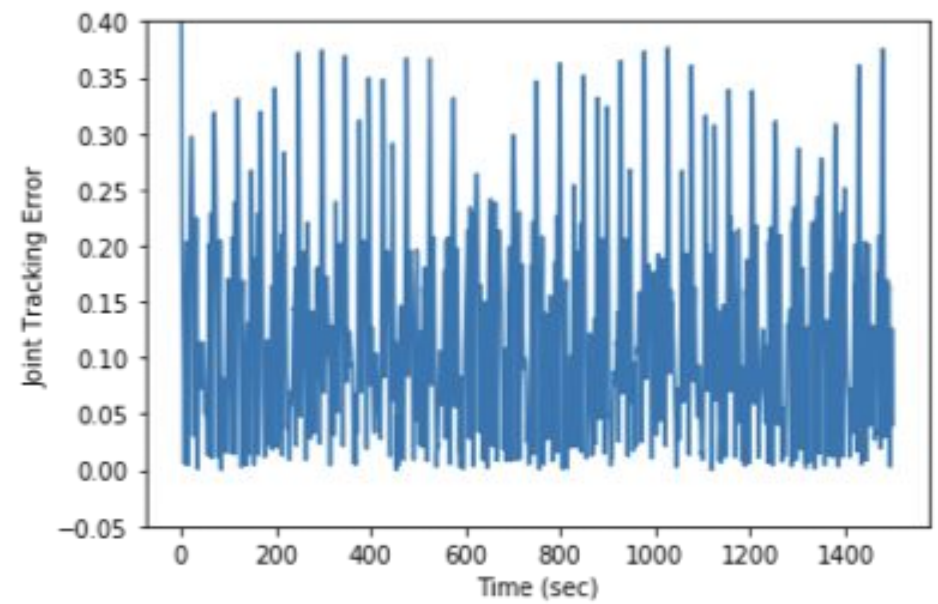
(a)

Tracking With Learning



(b)

Tracking Without Learning



(c)

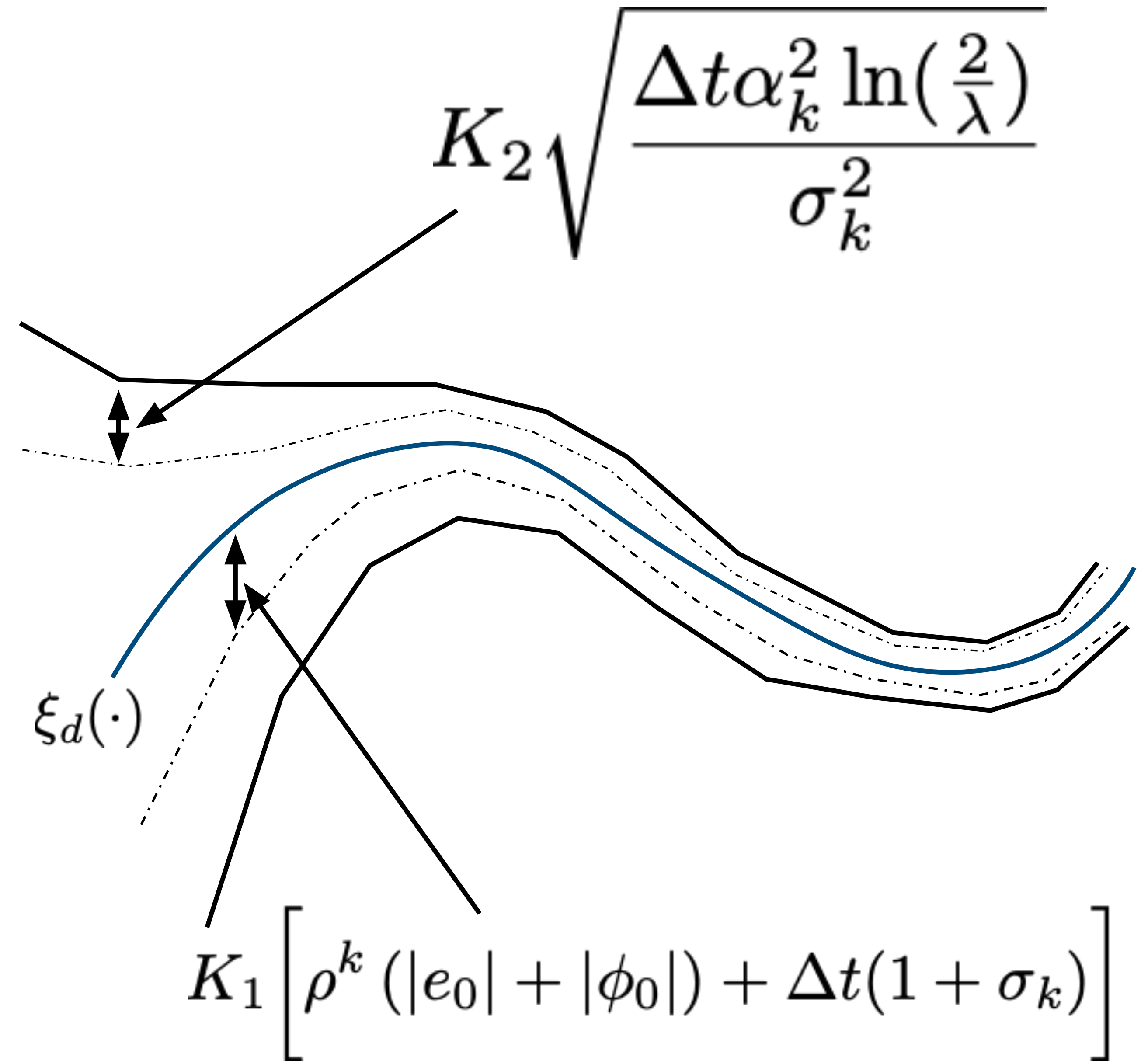
Step-size Selection

- Many convergence results from the ML literature require:

$$\sum_{k=0}^{\infty} \alpha_k = \infty, \qquad \alpha_k \to 0$$

- In a forthcoming article, we will show that the learning 'converges' if we take

$$\alpha_k \to 0,\ \ \sigma_k \to 0,\ \ \frac{\alpha_k}{\sigma_k} \to 0$$



Trade-offs with Model-Based Adaptive Control

- Advantages:
  - Can deal with non-parametric uncertainty
  - More freedom in choosing function approximator
- Disadvantages:
  - Generally slower
  - Loss of deterministic guarantees

L1 Adaptive Control

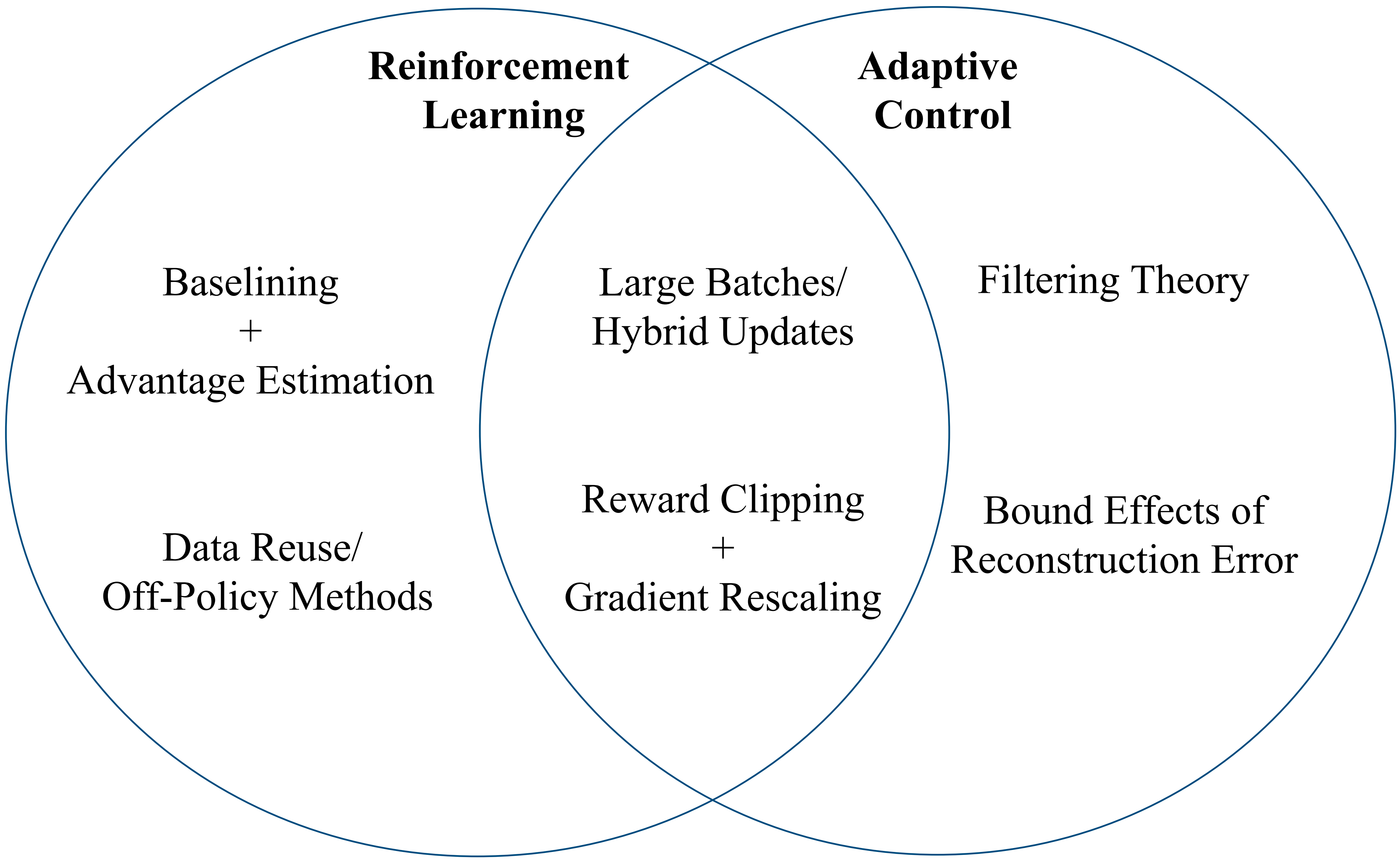
- Model unknown nonlinearities as a disturbance to be identified:

$$\dot{x}(t) = Ax(t) + b(wu(t) + f(x(t), t))$$
$$y(t) = cx(t)$$

Estimate with $\hat{w}(t)$ Estimate with $\hat{\delta}(t)$

- Control size of tracking by using fast adaptation for $\hat{\delta}(t)$

(Near) Future Work: More Sophisticated Algorithms



Thesis Proposal

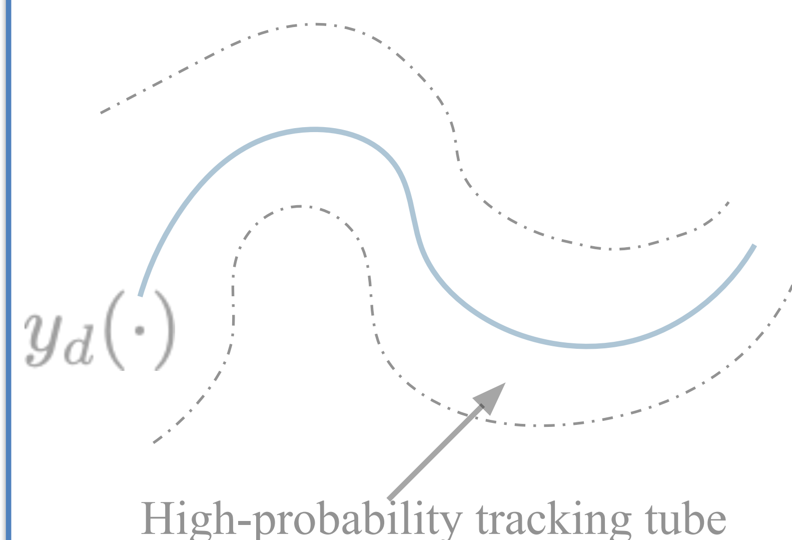
Part 1: Overcome non-parametric uncertainty by combining RL and geometric control

$$\hat{u}_\theta(x) = u_m(x) + \Delta u_\theta(x)$$

MB Controller

Learned Correction

Part 2: Provide correctness and safety guarantees for specific learning algorithms:



Part 3: Future work:

- **What makes a reward signal difficult to learn from?**
- **What makes a system fundamentally difficult to control?**
- **Where should geometric control be used in the long-run?**

Relavent Papers

- "Adaptive Control for Linearizable Systems Using On-Policy Reinforcement Learning" [**W**MFPTS] (CDC 2020, *To Appear*)

- "Reinforcement Learning for the Adaptive Control of Linearizable Systems" [**W**SMFTS] (Transaction on Automatic Control, *In Prep*)

- "Data-Efficient Off-Policy Reinforcement Learning for Nonlinear Adaptive Control" [**W**SMFTS] (TBD)

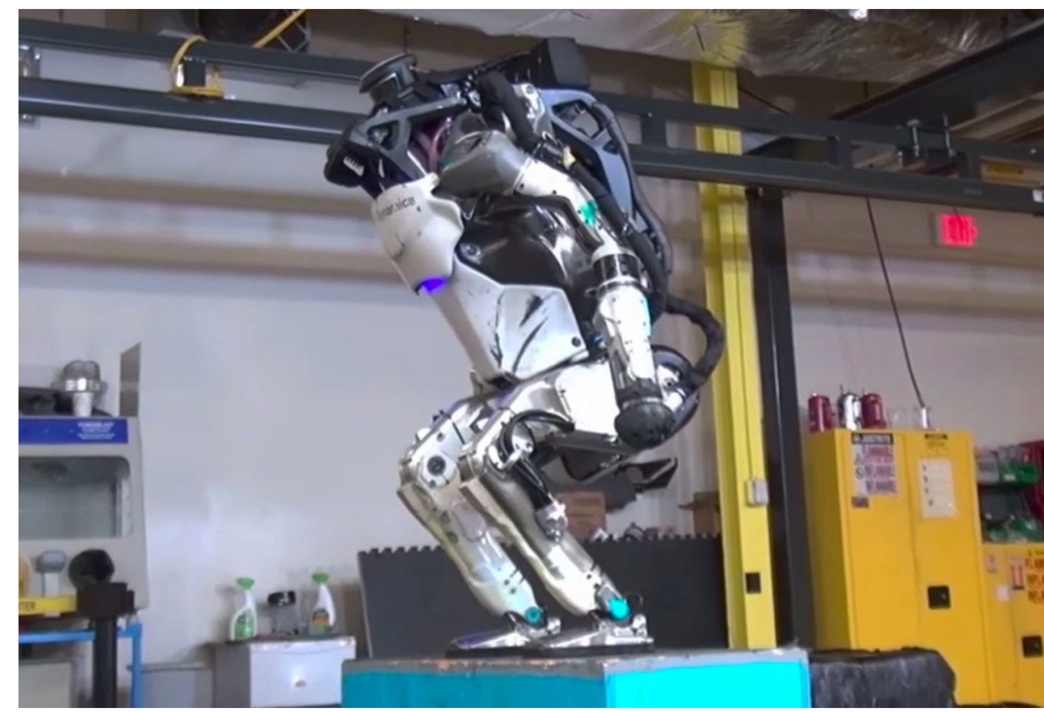
Where do these techniques fit in?

- Can we use geometric control to **partially** reduce the complexity of learning more difficult tasks?

- Can we combine our approach with techniques such as meta-learning?

[Finn et. al.] (2017)

- Can we automatically synthesize rewards for families of tasks?



Understanding Geometric 'Templates'

- So far: use geometric structures as 'templates' for learning
  - Can we formalize what makes a local reward signal 'compatible' with the global structure of the problem?
  - Can we quantify the difficulty of a RL problem in terms of how much global information the reward contains?
  - **Does reinforcement learning implicitly take advantage of the structures we've identified?**
  - Can we apply general structural results from geometric control?

What makes a system difficult to control?

- Control theory has many colloquial ways to describe what makes a system difficult to control

- Can we use sample complexity to make these notions rigorous? [1][2]

  - Can we use ideas from geometric control to separate out different ‘complexity classes’ of problems?

  - For example, Minimum-Phase << Non-Minimum Phase?

[1] Dean, Mania, Matni, Recht, Tu (2018) [2] Fazel, Ge, Kakade, Mesbahi(2018)

Non-Minimum Phase Tracking Control

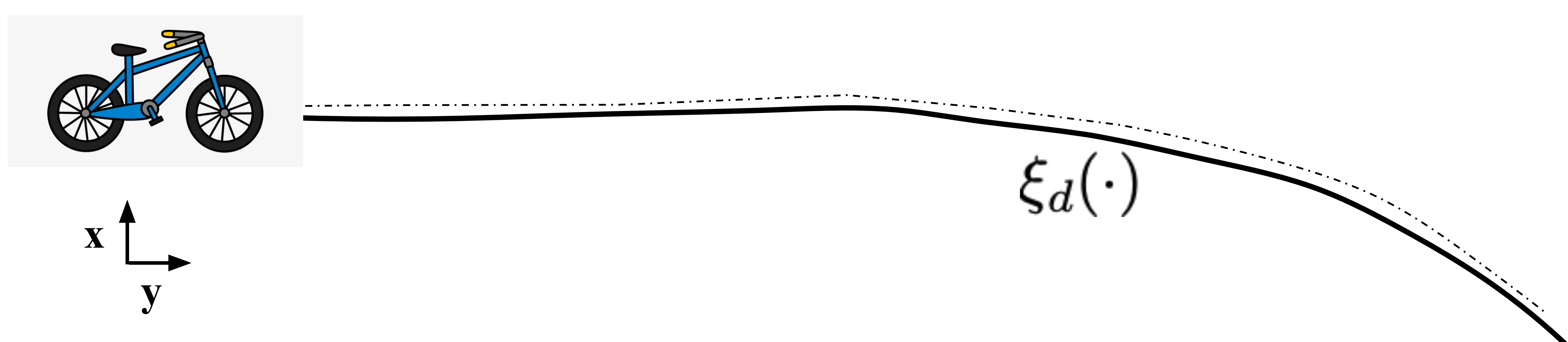
- When zero dynamics are NMP we cannot ‘forget’ them

- Example: steering a bike

Recall the normal form:

$$\dot{\xi} = A\xi + Bv$$

$$\dot{\eta} = q(\xi, \eta) + p(\xi, \eta)v$$



Non-Minimum Phase Tracking Control

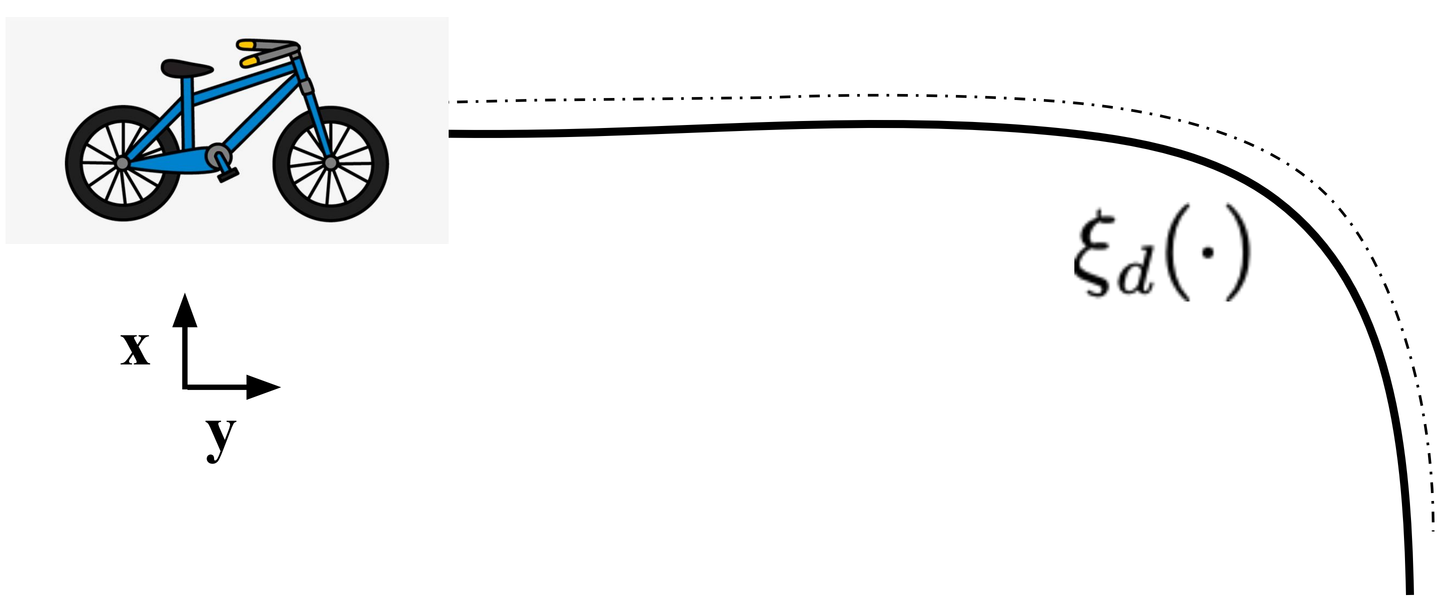
- When zero dynamics are NMP we cannot 'forget' them

- Example: steering a bike

Recall the normal form:

$$\dot{\xi} = A\xi + Bv$$

$$\dot{\eta} = q(\xi, \eta) + p(\xi, \eta)v$$



Non-Minimum Phase Tracking Control

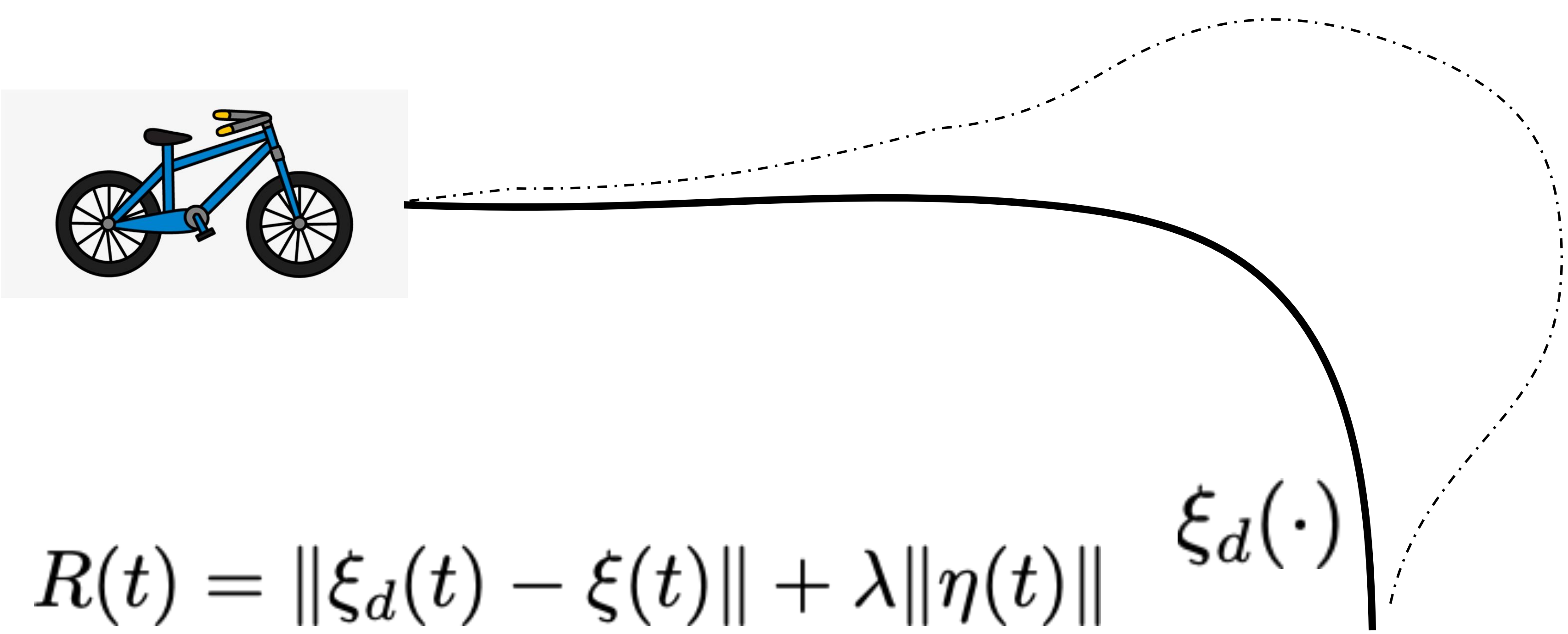
- When zero dynamics are NMP we cannot ‘forget’ them

- Example: steering a bike

Recall the normal form:

$$\dot{\xi} = A\xi + Bv$$
$$\dot{\eta} = q(\xi, \eta) + p(\xi, \eta)v$$

“Counter-Steering”



- **Can we learn these behaviors?**

$$R(t) = \|\xi_d(t) - \xi(t)\| + \lambda\|\eta(t)\|$$

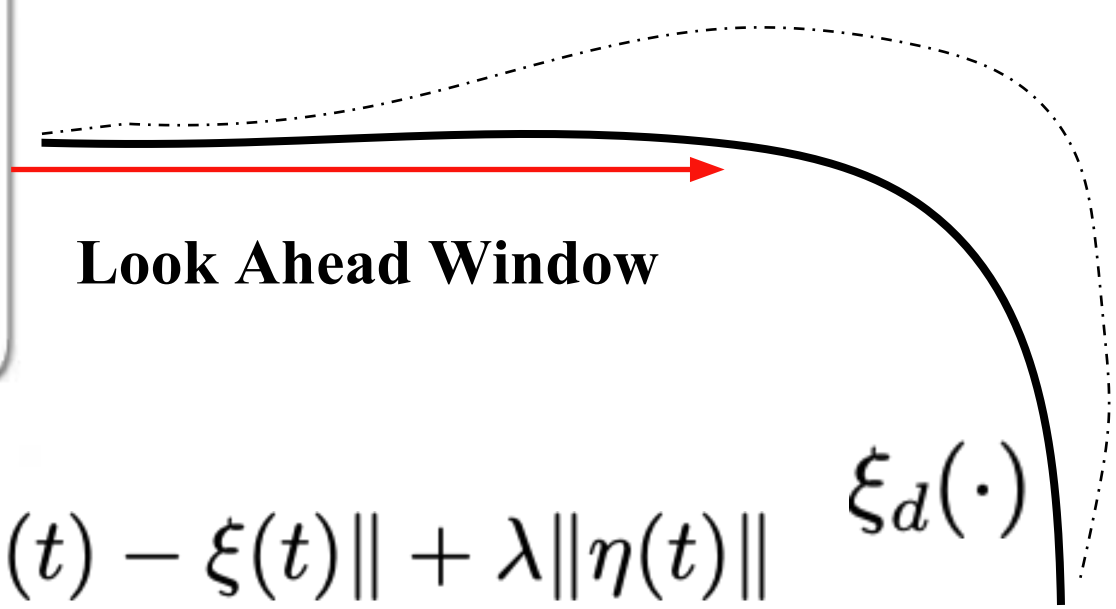
Non-Minimum Phase Tracking Control

- When zero dynamics are NMP we cannot 'forget' them
- Example: steering a bike

Recall the normal form:

$$\dot{\xi} = A\xi + Bv$$
$$\dot{\eta} = q(\xi, \eta) + p(\xi, \eta)v$$

"Counter-Steering"



$$R(t) = \|\xi_d(t) - \xi(t)\| + \lambda\|\eta(t)\|$$

- **Can we learn these behaviors?**
- **Can we quantify what makes it difficult to learn these behaviors?**
- **How much preview do we need to learn?**
- **Can we learn safely?**

[Devasia et. al.] (1999)

Questions?