Autonomous Robotics

Winter 2025

Abhishek Gupta

TAs: Carolina Higuera, Entong Su, Bernie Zhu

Class Outline

State Estimation

- Robotic System Design
- Filtering
- Localization
- SLAM

Control

- Feedback Control
- PID Control
- MPC
- LQR

Planning

- Search
- Heuristic Search
- Motion Planning
- Lazy Search

Learning

- Imitation Learning
- Policy Gradient
- Actor-Critic
- Model-Based RL

Logistics

- Seeded Paper Discussion 2 - Wed Feb 19
- HW3 due Feb 20

- Post questions, discuss any issues you are having on Ed.
- Students with **no** access to 002, e-mail us with your student ID.
- Students that have not been added to the class, email abhgupta@cs.washington.edu with the subject-line "Waitlisted for CSE478"

Recap

LQR Riccati Equations

$$K_i = -(R + B^\top P_{i-1} B)^{-1} B^\top P_{i-1} A$$

$$P_i = Q + K_i^\top R K_i + (A + BK_i)^\top P_{i-1} (A + BK_i)$$

$$u = K_i x, \;\; J_i(x) = x^\top P_i x$$

Optimal controller is linear in x

Optimal cost is quadratic in x

RUNTIME: $O(H(n^3 + m^3))$

The LQR algorithm

```
Algorithm OptimalValueControl(A, B, Q, R, time-to-go):

    if time-to-go == 0:
        return 0, Q

    else:
        P_{i-1} = OptimalValueControl(A, B, Q, R, time-to-go - 1)
```

$$K_i = -(R + B^\top P_{i-1} B)^{-1} B^\top P_{i-1} A$$

$$P_i = Q + K_i^\top R K_i + (A + BK_i)^\top P_{i-1} (A + BK_i)$$

return K_i, P_i

Optimal controller is linear in x

Optimal cost is quadratic in x

Unpacking LQR intuitively

x^T $\left[P_i = Q + K_i^\top R K_i + (A + BK_i)^\top P_{i-1} (A + BK_i) \right]$ x

Current state cost

Current action cost

Optimal cost in the future based on dynamics

LQR Ext1: non-linear systems

Nonlinear system: $x_{t+1} = f(x_t, u_t)$

We can keep the system at the state x^* iff

$$\exists u^* \text{s.t.} \quad x^* = f(x^*, u^*)$$

Linearizing the dynamics around x^* gives:

$$x_{t+1} \approx f(x^*, u^*) + \underbrace{\frac{\partial f}{\partial x}(x^*, u^*)}_{A}(x_t - x^*) + \underbrace{\frac{\partial f}{\partial u}(x^*, u^*)}_{B}(u_t - u^*)$$

Equivalently:

$$x_{t+1} - x^* \approx A(x_t - x^*) + B(u_t - u^*)$$

Let $z_t = x_t - x^*$, let $v_t = u_t - u^*$, then:

$$z_{t+1} = Az_t + Bv_t, \qquad \text{cost} = z_t^\top Q z_t + v_t^\top R v_t$$ [=standard LQR]

$$v_t = K z_t \Rightarrow u_t - u^* = K(x_t - x^*) \Rightarrow u_t = u^* + K(x_t - x^*)$$

Lecture Outline

From LQR to MPC

↓

Lyapunov Stability

Why might this not be enough?

$$\min_{u_{1:T}} \sum_{t=1}^{T} c(x_t, u_t)$$

$$\text{s.t.} \quad x_{t+1} = f(x_t, u_t)$$

Non-linear

Non-quadratic

Use linear/quadratic Taylor expansion about current nominal states/actions

$$\sum_t x_t^\top Q x_t + u_t^\top R u_t$$

$$x_{t+1} = A x_t + B u_t$$

Might be a poor, local approximation!

May not be able to incorporate constraints

Let's revisit ideas from Bayesian filtering

	Linear Gaussian assumption	Sampling-based approximation
<u>Filtering</u>	Kalman Filtering	Particle Filtering
<u>Control</u>	LQR	Sampling based MPC

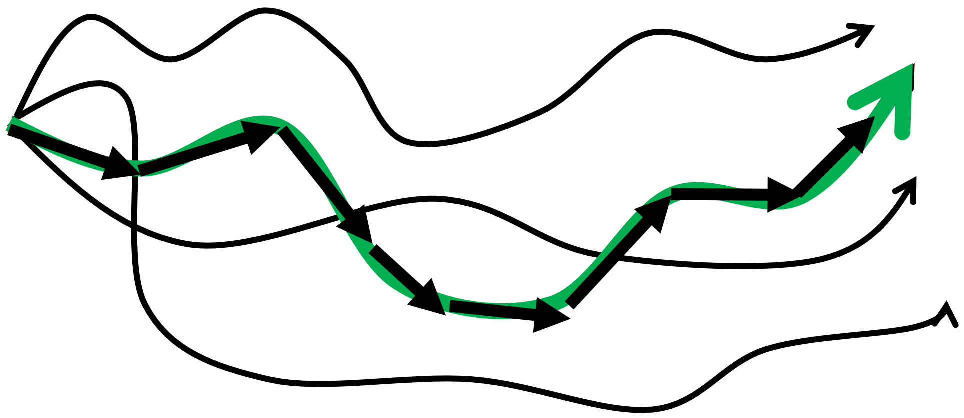
Solving Optimal Control with Sampling

$$\min_{u_{1:T}} \sum_{t=1}^{T} c(x_t, u_t)$$

$$\text{s.t.} \quad x_{t+1} = f(x_t, u_t)$$

1. Sample a set of K action trajectories of T steps from start state
2. Evaluate each K step action sequence through the model and get per trajectory cost
3. Choose minimum trajectory cost trajectory
4. Execute lowest cost actions

Random Sampling



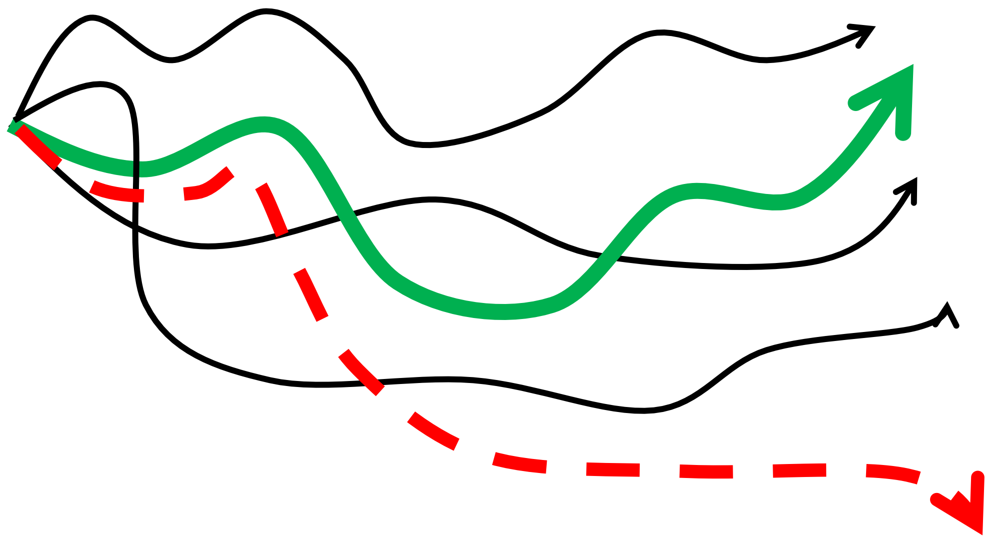
Can soften by taking softmin rather than argmin

Solving Optimal Control with Sampling – issues?

$$\min_{u_{1:T}} \sum_{t=1}^{T} c(x_t, u_t)$$

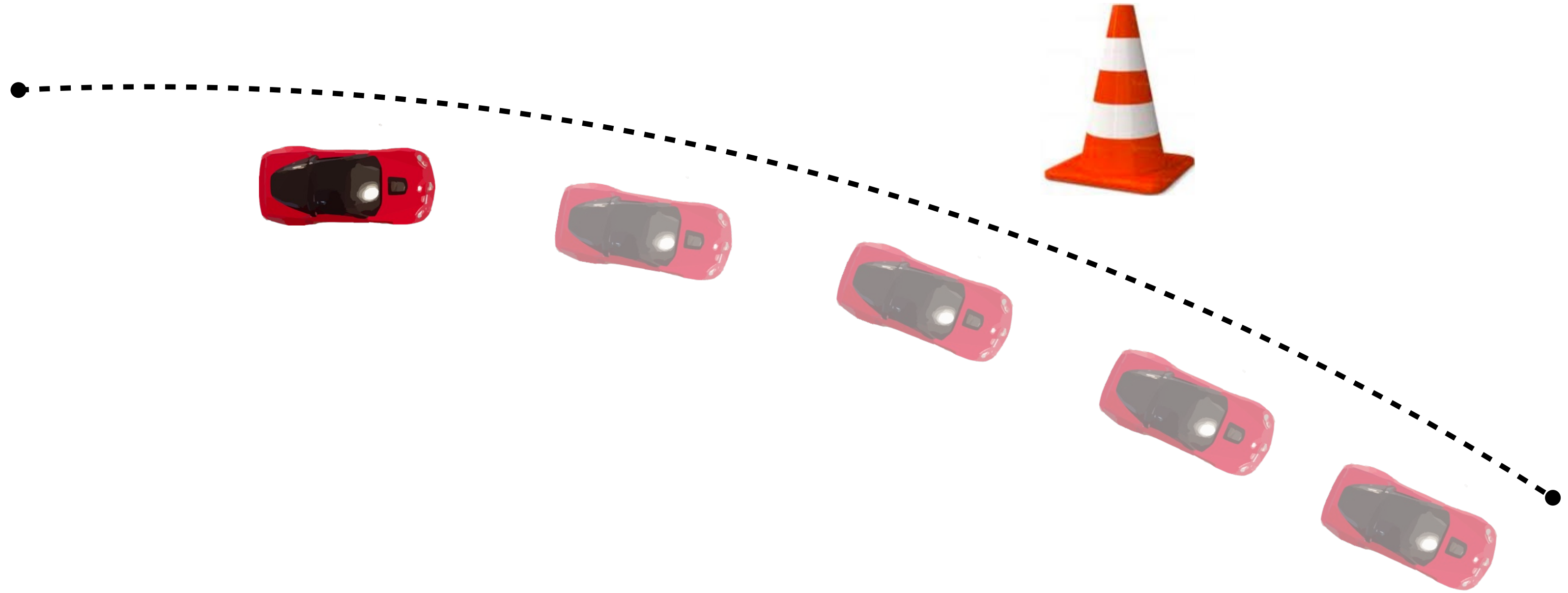
$$\text{s.t.} \quad x_{t+1} = f(x_t, u_t)$$

1. Sample a set of K action trajectories of T steps from start state
2. Evaluate each K step action sequence through the model and get per trajectory cost
3. Choose minimum trajectory cost trajectory
4. Execute lowest cost actions



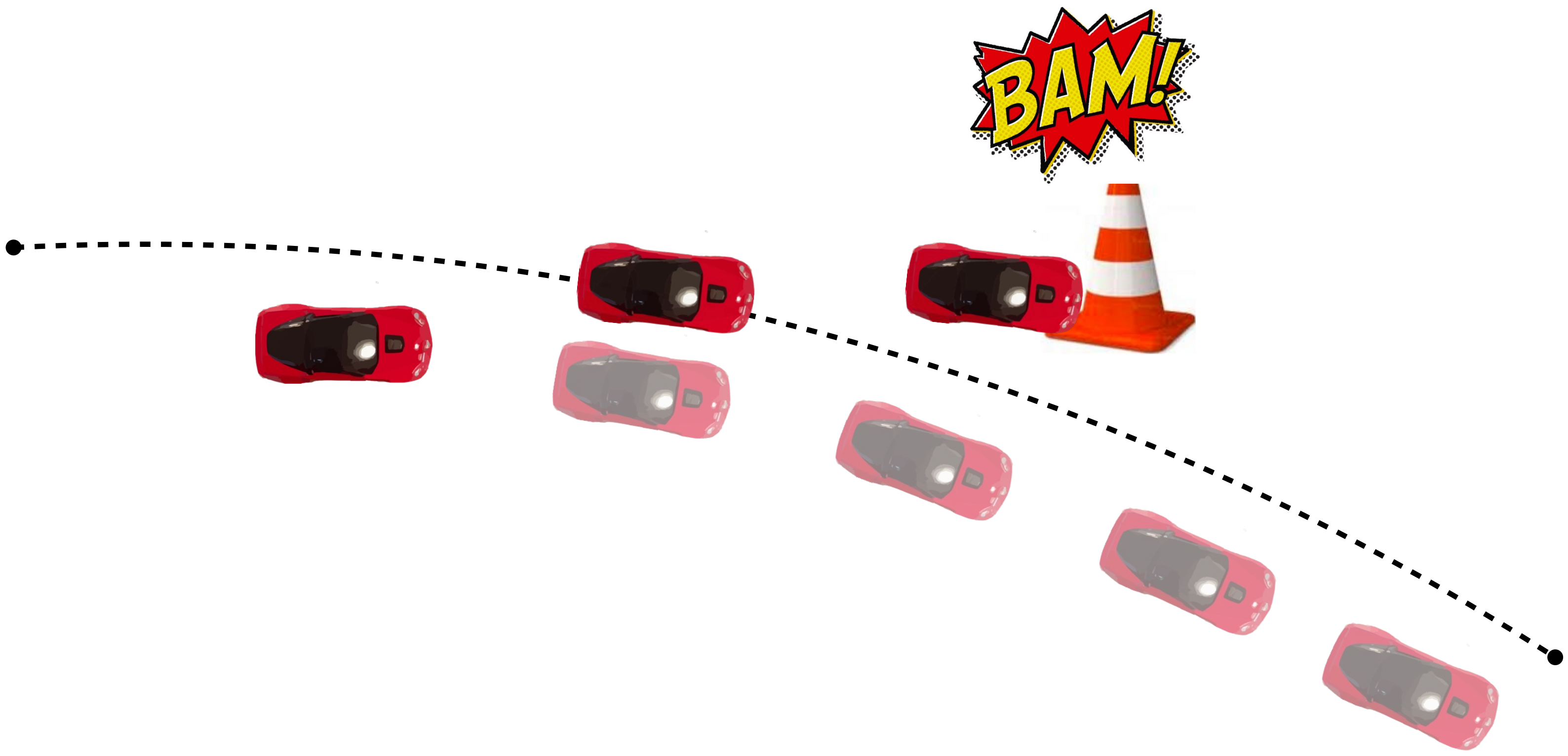
1. Open-loop controller may not be able to deal with unexpected events/divergences
2. Computation of full controller can be expensive: → Do it on the fly!
3. Model might be wrong, errors may accumulate
4. …

Why do we need to replan?



What happens if the controls are planned once and executed?

Why do we need to replan?

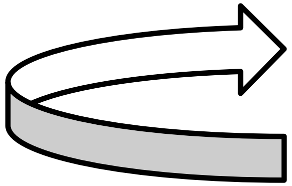


What happens if the controls are planned once and executed?

Solving Optimal Control with Sampling – issues?

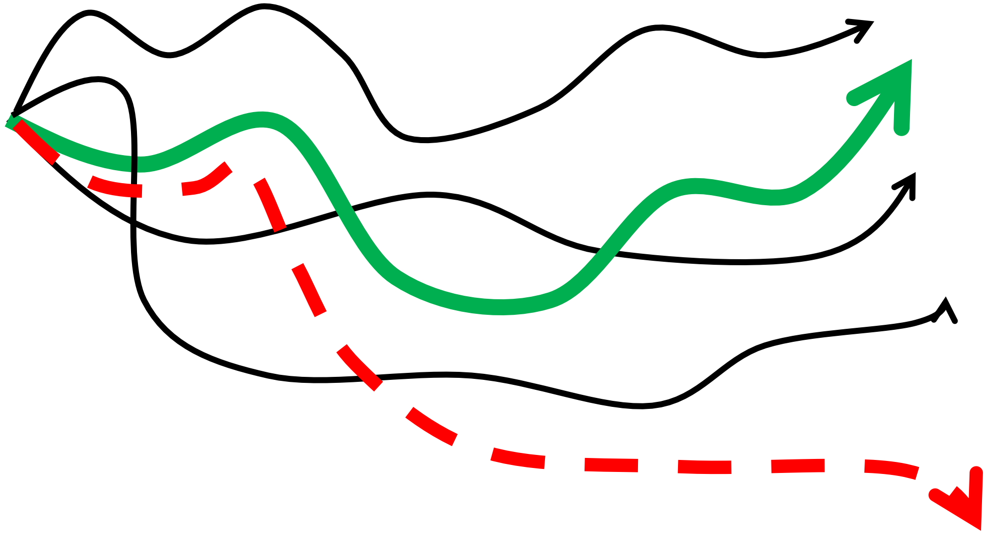
$$\min_{u_{1:T}} \sum_{t=1}^{T} c(x_t, u_t)$$

$$\text{s.t.} \quad x_{t+1} = f(x_t, u_t)$$

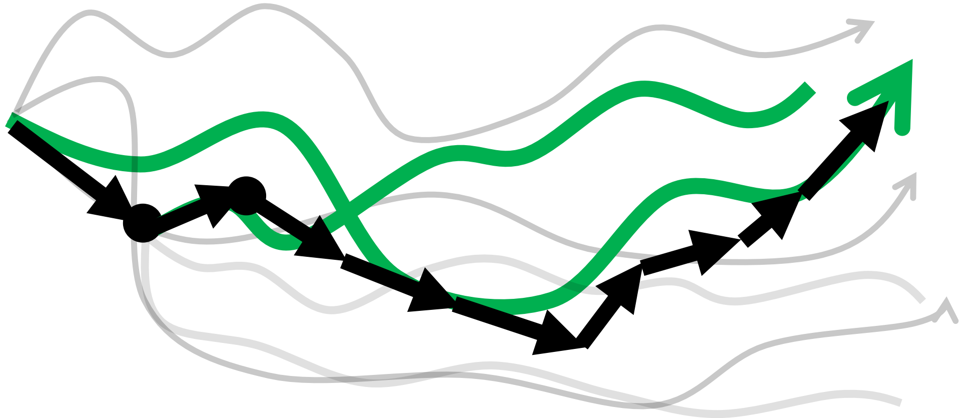


1. Plan with random shooting from s_t
2. Execute the first action a_0 and reach s_{t+1}

A stationary feedback controller may not be able to deal with unexpected events

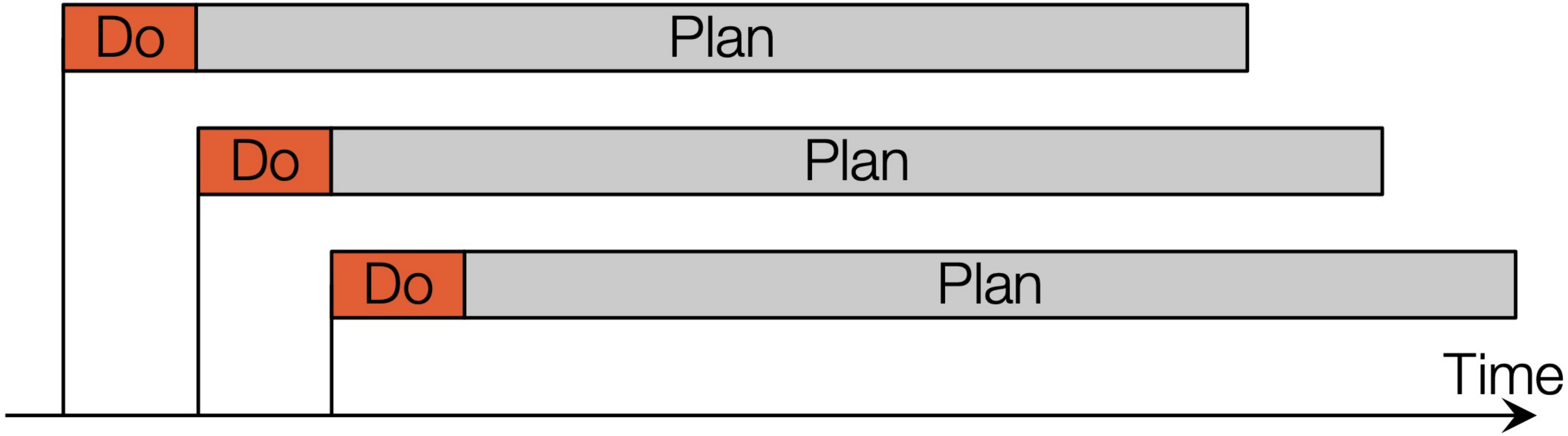


Replanning can help with divergence



Model-Predictive/Receding Horizon Control

General Replanning Framework - MPC

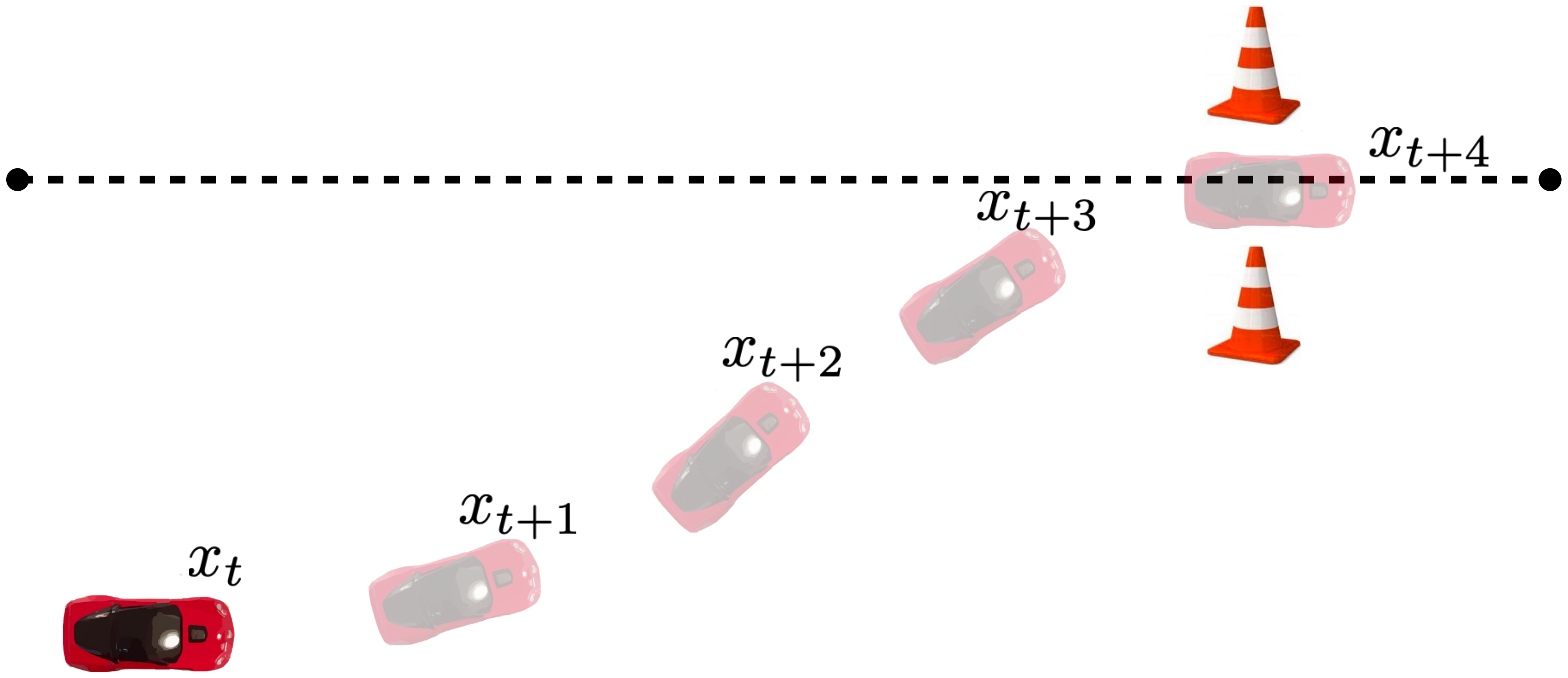


Step 1: Solve optimization problem to a horizon

Step 2: Execute the first control

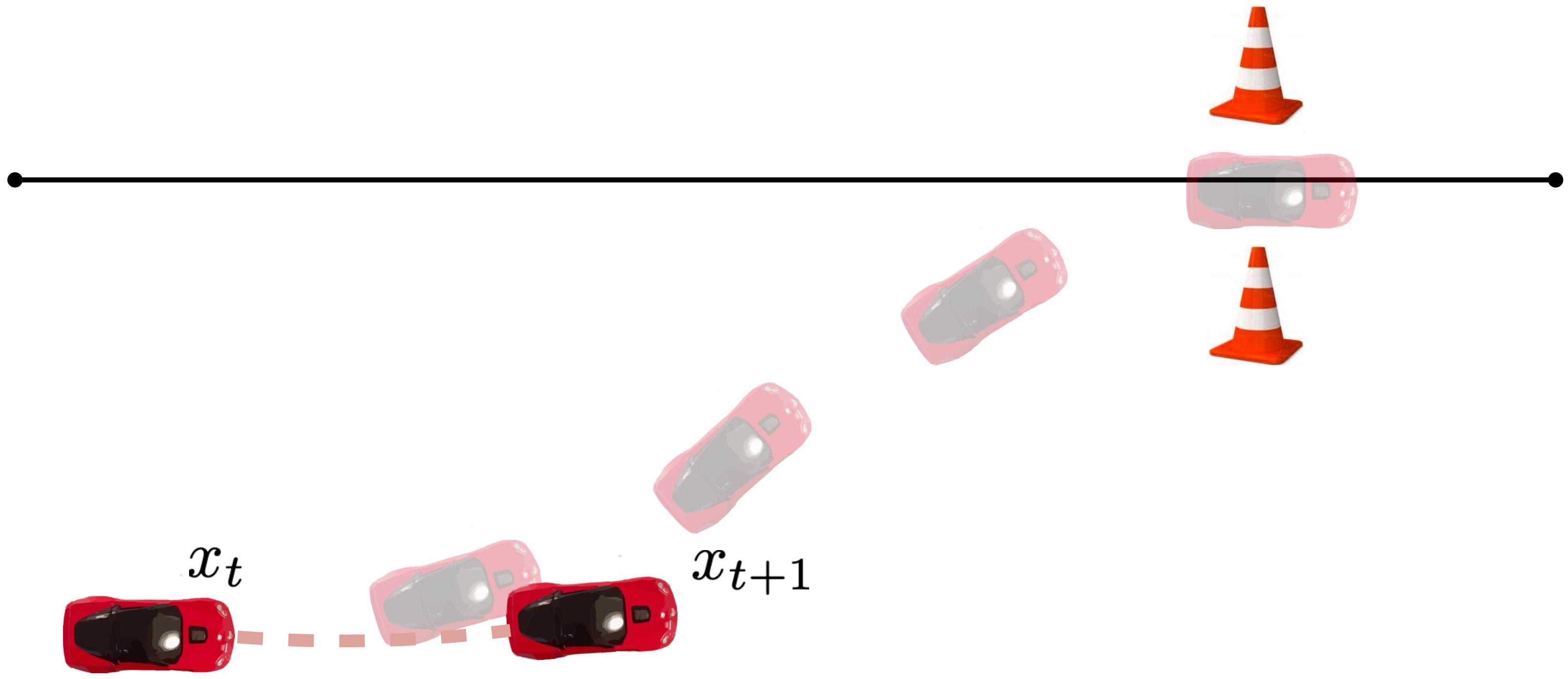
Step 3: Repeat!

How are the controls executed?



Step 1: Solve optimization problem to a horizon

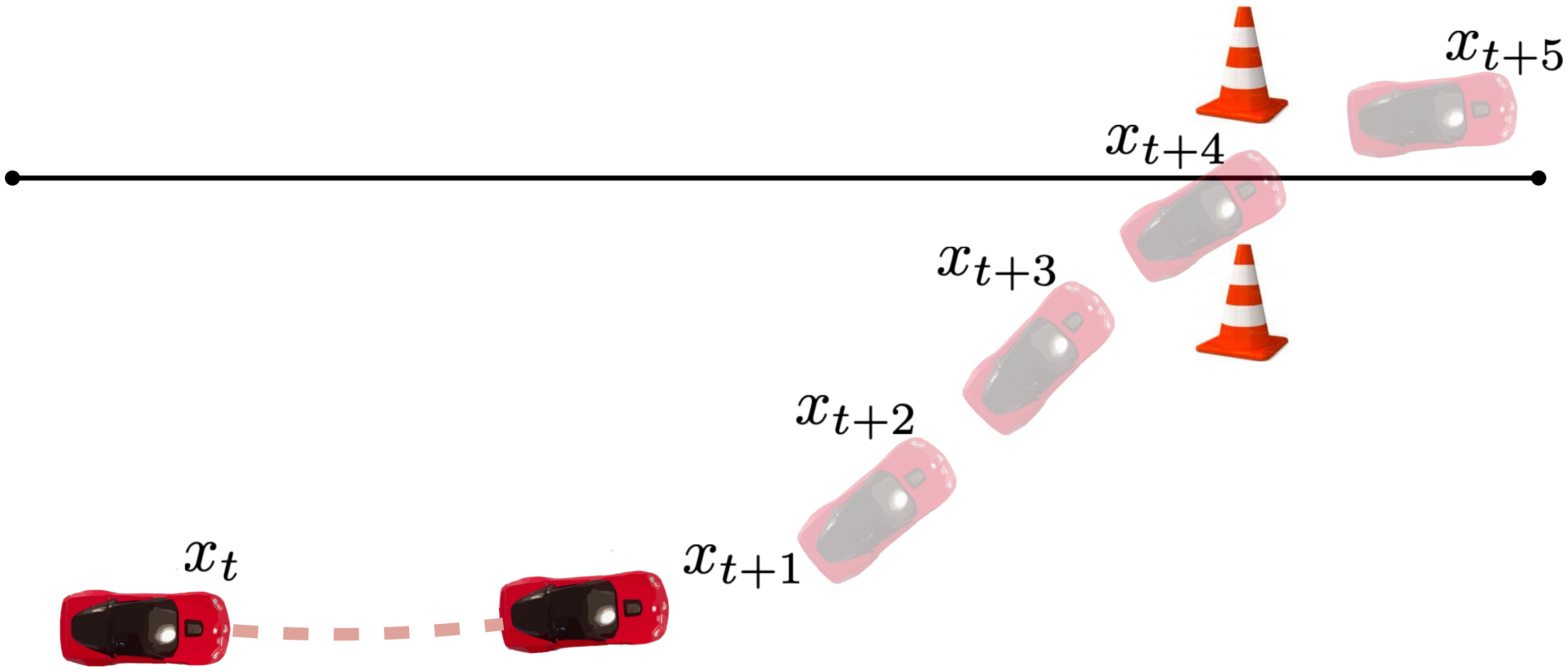
How are the controls executed?



Step 1: Solve optimization problem to a horizon

Step 2: Execute the first control

How are the controls executed?

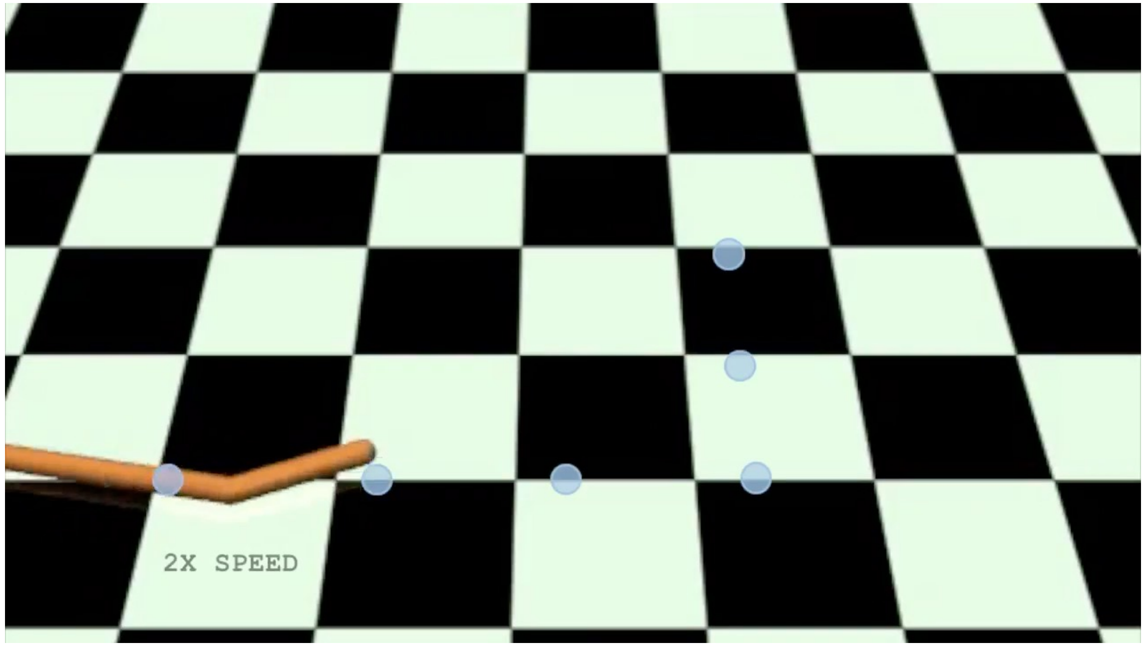


Step 1: Solve optimization problem to a horizon

Step 2: Execute the first control

Step 3: Repeat!

Does it work?



Nagabandi et al

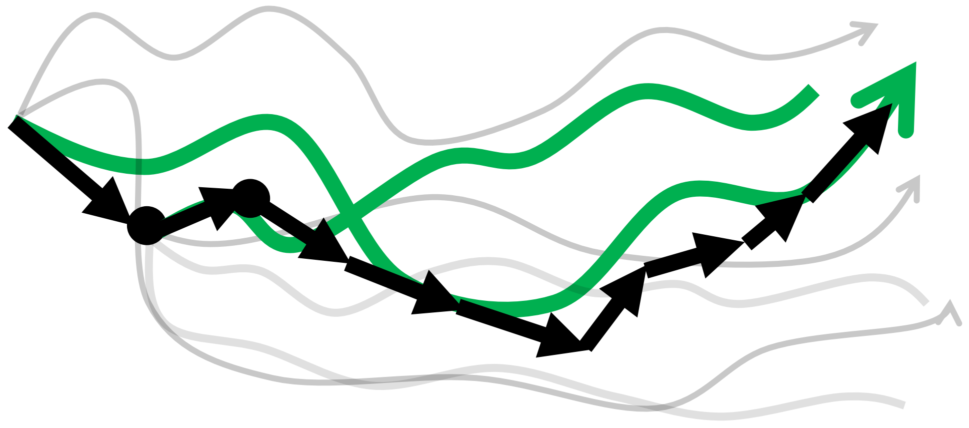
Why might this not work?

$$\min_{u_{1:T}} \sum_{t=1}^{T} c(x_t, u_t)$$

$$\text{s.t.} \quad x_{t+1} = f(x_t, u_t)$$

1. Sample a set of K action trajectories of T steps from start state
2. Evaluate each K step action sequence through the model and get per trajectory cost
3. Choose minimum trajectory cost trajectory
4. Execute lowest cost actions

Planning with Shooting + MPC



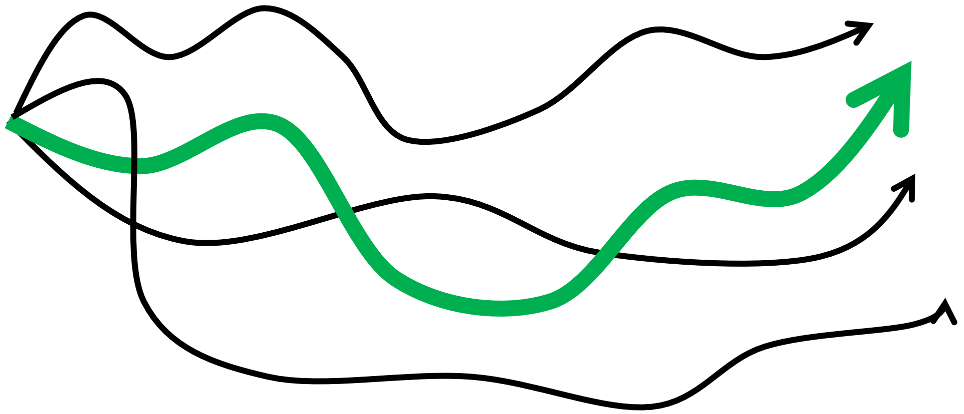
Searching for a needle in a haystack by random shooting, high variance!

Better Sampling Techniques for MPC

Sampled from stationary uniform/gaussian distribution

$$\arg \min_{u_0, u_1, \ldots, u_T} \sum_{t=1}^{T} c(x_t, u_t)$$

$$x_{t+1} = f(x_t, u_t)$$

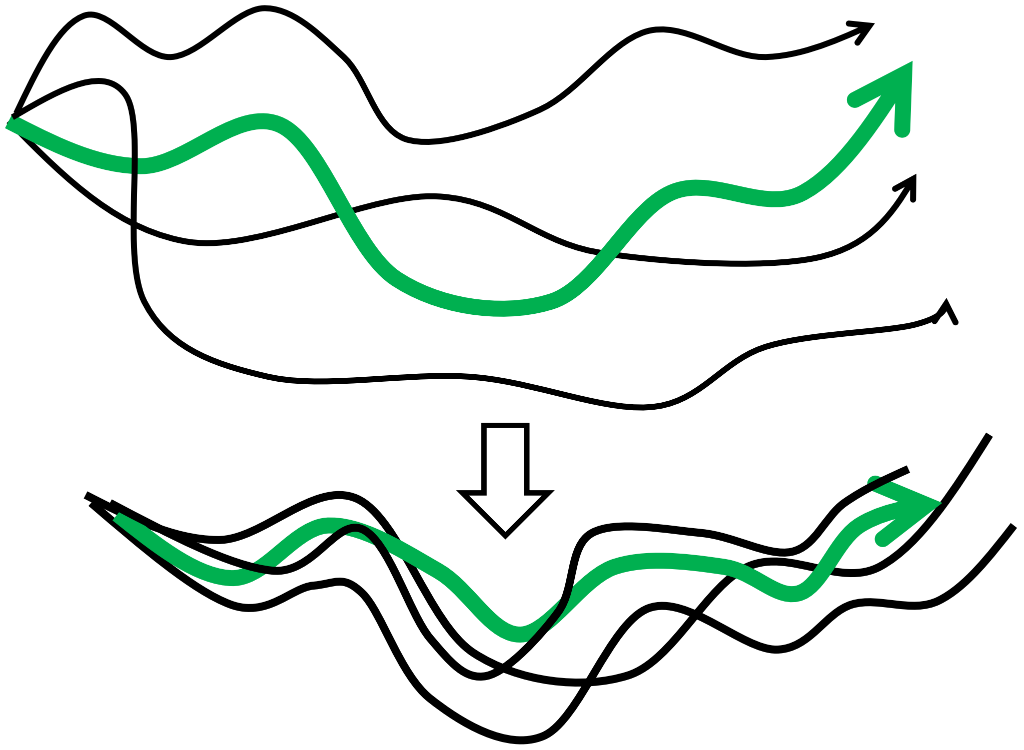


Can we inform the sampling function with the cost function?

Idea: Iteratively upweight sampling distribution around the things that are lower cost

Better Sampling Techniques for Shooting - MPPI

Idea: Iteratively upweight sampling distribution around the things that are lower costs



Referred to as **MPPI**, lower variance!

Sample N action sequences

$$(u_0^j, u_1^j, \ldots, u_T^j)_{j=1}^N \sim p(u)$$

Sample trajectories using these action sequences with the model

$$x_{t+1}^j = f(x_t^j, u_t^j)$$

Update action sampler by upweighting low cost actions

$$p(u) \leftarrow p(u) \frac{\exp \sum_{t=1}^T -c(x_t, u_t)}{Z}$$

Does it work?

Does it work?

Lecture Outline

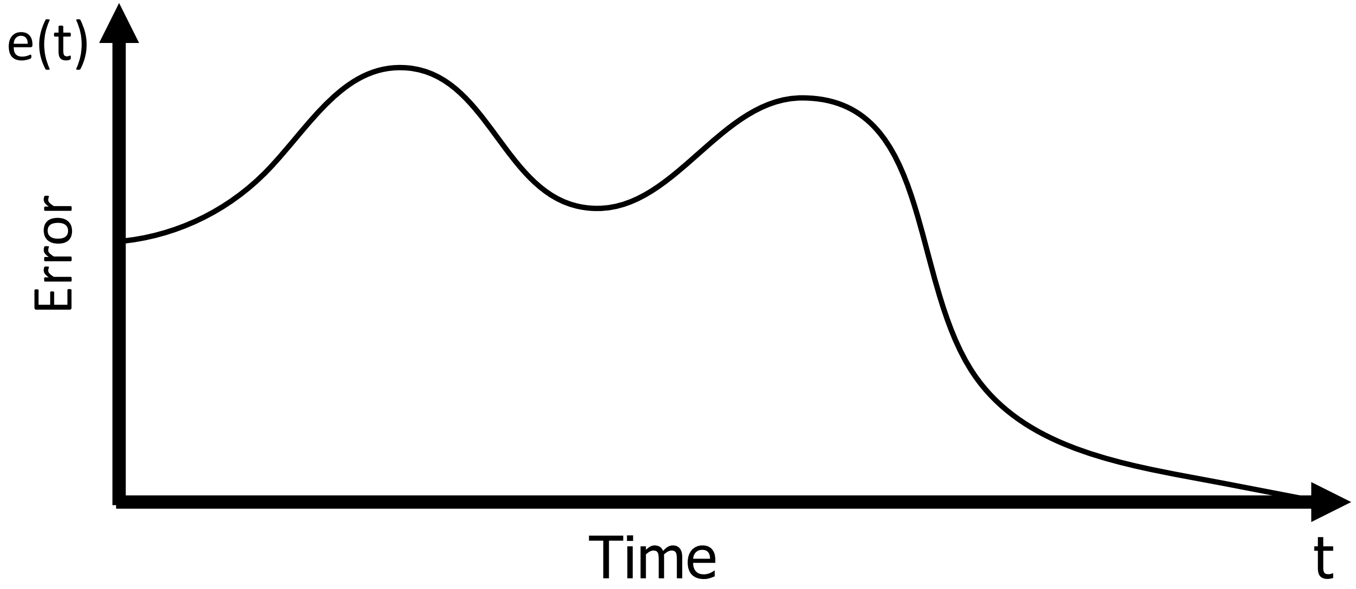
From LQR to MPC

↓

Lyapunov Stability

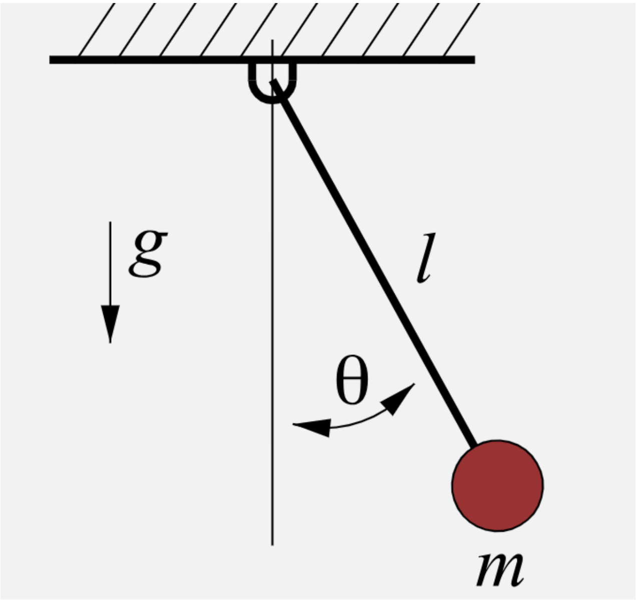
What is stability?

$$\lim_{t \to \infty} e(t) = 0$$



So we want both $e(t) \to 0$ and $\dot{e}(t) \to 0$

Detour: How do we make a pendulum stable?



$$ml^2\ddot{\theta} + mgl\sin\theta = u$$

What control law should we use to stabilize the pendulum, i.e.

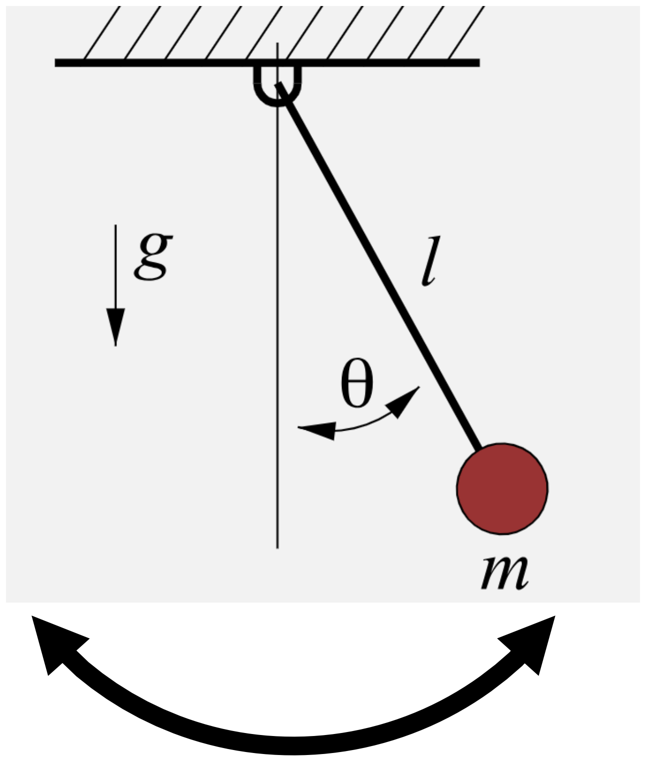
Choose $u = \pi(\theta, \dot{\theta})$ such that $\theta \rightarrow 0$

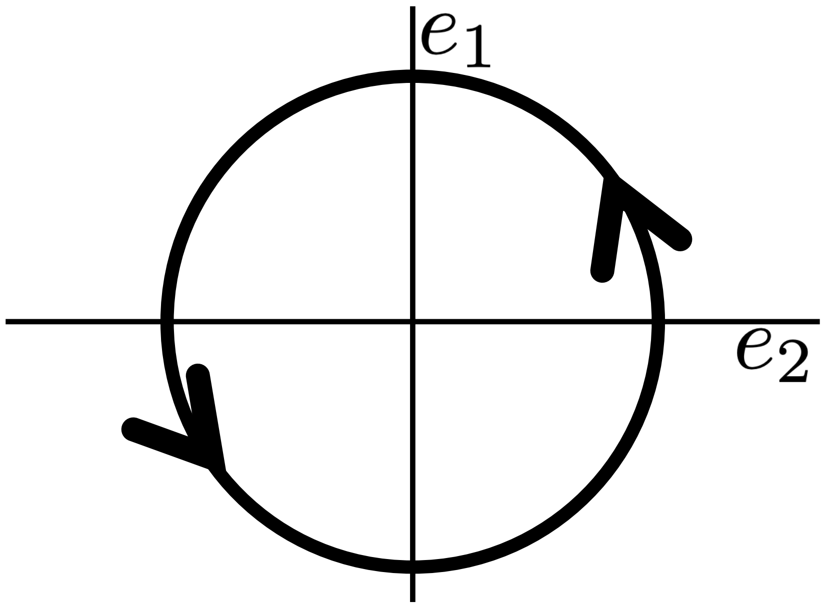
$\dot{\theta} \rightarrow 0$

How does the passive error dynamics behave?

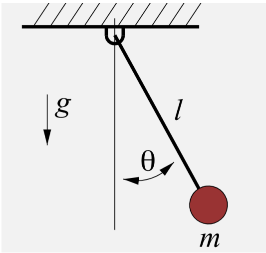
$$e_1 = \theta - 0 = \theta \qquad e_2 = \dot{\theta} - 0 = \dot{\theta}$$

Set u=0. Dynamics is not stable.





How do we verify if a controller is stable?



$$ml^2\ddot{\theta} + mgl\sin\theta = u$$

Lets pick the following law:

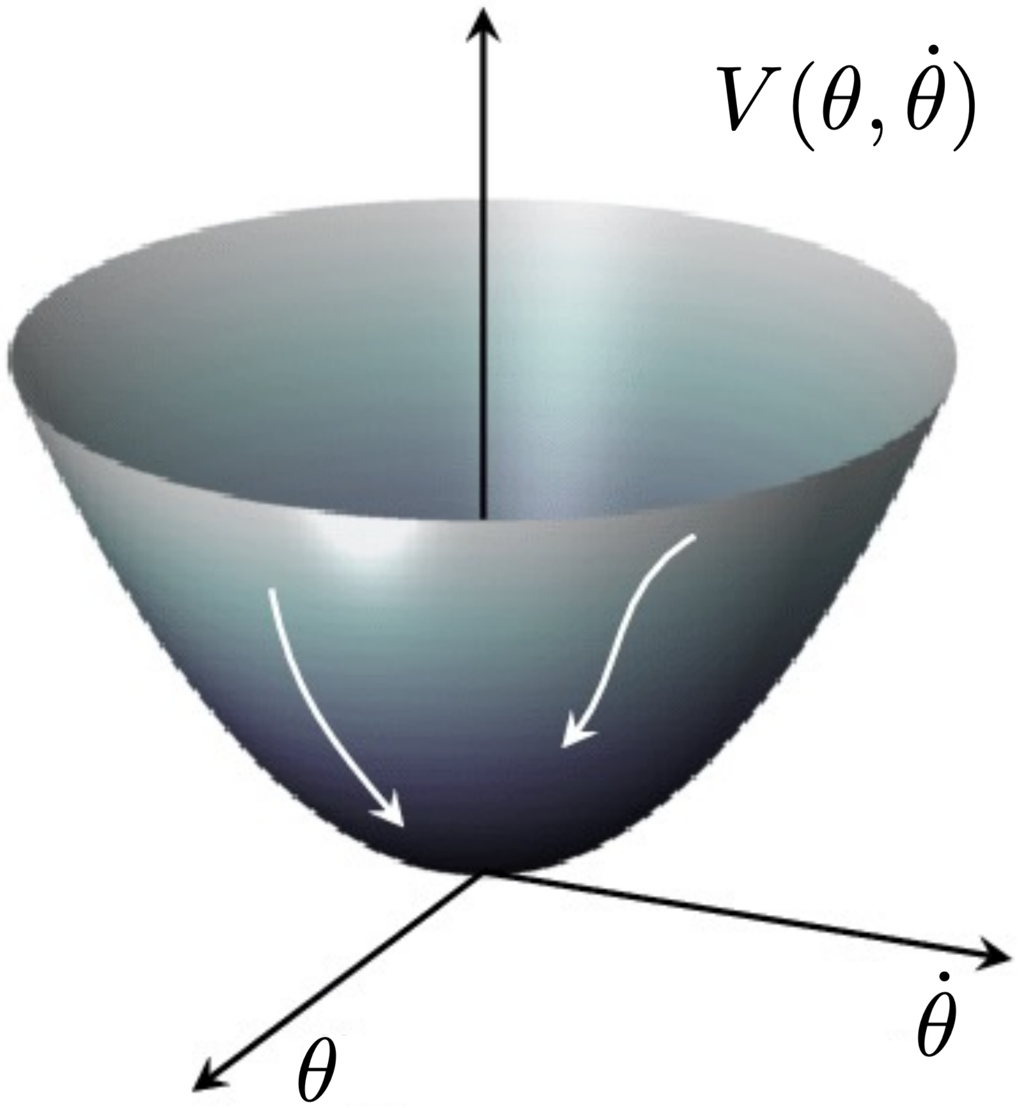
$$u = -K\dot{\theta}$$

Is this stable? How do we know?

We can simulate the dynamics from different start point and check….

but how many points do we check? what if we miss some points?

Key Idea: Think about energy!



Make energy decay to 0 and stay there

$$V(\theta, \dot{\theta}) = \frac{1}{2} m l^2 \dot{\theta}^2 + mgl(1 - \cos\theta)$$

$$> 0$$

$$\dot{V}(\theta, \dot{\theta}) = m l^2 \dot{\theta}\ddot{\theta} + mgl(\sin\theta)\dot{\theta}$$

$$= \dot{\theta}(u - mgl\sin\theta) + mgl(\sin\theta)\dot{\theta}$$

$$= \dot{\theta} u$$

Choose a control law $u = -k\dot{\theta}$

$$\dot{V}(\theta, \dot{\theta}) = -k\dot{\theta}^2 < 0$$

Lyapunov function:

A generalization of energy

Lyapunov function for a closed-loop system

1. Construct an energy function that is always positive

$$V(x) > 0, \forall x$$

Energy is only 0 at the origin, i.e. $V(0) = 0$

2. Choose a control law such that this energy always decreases

$$\dot{V}(x) < 0, \forall x$$

Energy rate is 0 at origin, i.e. $\dot{V}(0) = 0$

No matter where you start, energy will decay and you will reach 0!

Let's get provable control for our car!

Dynamics of the car

$$\dot{x} = V \cos\theta$$
$$\dot{y} = V \sin\theta$$
$$\dot{\theta} = \frac{V}{B} \tan u$$

Let's get provable control for our car!

Let's define the following Lyapunov function

$$V(e_{ct}, \theta_e) = \frac{1}{2} k_1 e_{ct}^2 + \frac{1}{2} \theta_e^2 \qquad > 0$$

Compute derivative

$$\dot{V}(e_{ct}, \theta_e) = k_1 e_{ct} \dot{e_{ct}} + \theta_e \dot{\theta_e}$$

$$\dot{V}(e_{ct}, \theta_e) = k_1 e_{ct} V \sin \theta_e + \theta_e \frac{V}{B} \tan u$$

Let's get provable control for our car!

$$\dot{V}(e_{ct}, \theta_e) = k_1 e_{ct} V \sin\theta_e + \theta_e \frac{V}{B} \tan u$$

Trick: Set u intelligently to get this term to always be negative

$$\theta_e \frac{V}{B} \tan u = -k_1 e_{ct} V \sin\theta_e - k_2 \theta_e^2$$

$$\tan u = -\frac{k_1 e_{ct} B}{\theta_e} \sin\theta_e - \frac{B}{V} k_2 \theta_e$$

$$u = \tan^{-1}\left(-\frac{k_1 e_{ct} B}{\theta_e} \sin\theta_e - \frac{B}{V} k_2 \theta_e\right)$$

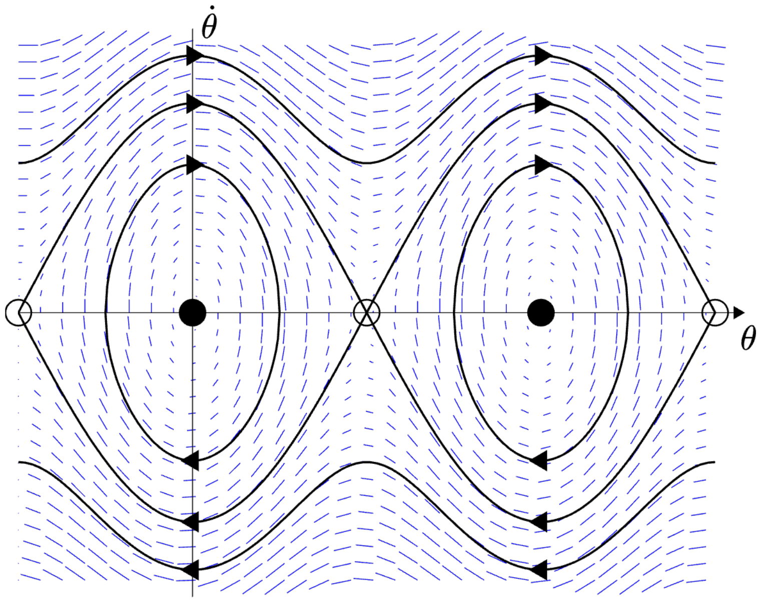
So what's the point of Lyapunov theory?

<u>Option 1:</u>

Use Lyapunov theory to <u>**construct**</u> stable controllers

<u>Option 2:</u>

Use Lyapunov theory to <u>**verify**</u> controllers for stability



Lecture Outline

Recap + iLQR

↓

Sampling-based Optimal Control

↓

Lyapunov Stability

Class Outline

State Estimation

- Robotic System Design
- Filtering
- Localization
- SLAM

Control

- Feedback Control
- PID Control
- MPC
- LQR

Planning

- Search
- Heuristic Search
- Motion Planning
- Lazy Search

Learning

- Imitation Learning
- Policy Gradient
- Actor-Critic
- Model-Based RL