

W

Autonomous Robotics

Spring 2026

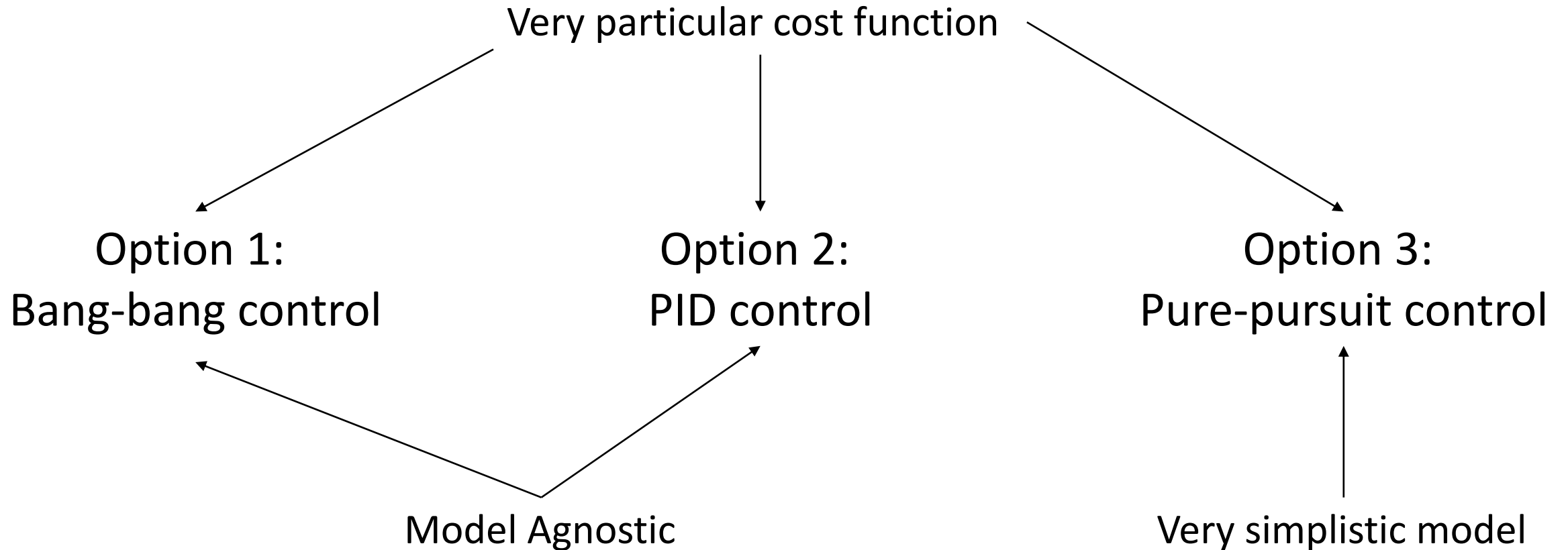
Abhishek Gupta, Siddhartha Srinivasa

TAs: Helen Wang, Rohan Baijal, Sidharth Talia, Christopher Tan



Recap

Controller Design Decisions



Bellman Equation for Dynamic Programming

- **Linear** system (model)
- **Quadratic** cost function to minimize

$$x_{t+1} = Ax_t + Bu_t$$
$$\sum_t x_t^\top Q x_t + u_t^\top R u_t$$

$$J^*(x_t) = \min_{u_t} x_t^\top Q x_t + u_t^\top R u_t + J^*(x_{t+1})$$

**MINIMUM COST,
STARTING FROM**

x_t

**IMMEDIATE
COST**

**MINIMUM FUTURE
COST, STARTING**

FROM x_{t+1}

Value Iteration (Horizon = 1)

$$J_1(x) = \min_u [x^\top Qx + u^\top Ru + (Ax + Bu)^\top P_0(Ax + Bu)]$$

$$\nabla_u[\cdot] = 2Ru + 2B^\top P_0(Ax + Bu) = 0$$

$$u = -(R + B^\top P_0 B)^{-1} B^\top P_0 Ax$$

$$J_1(x) = x^\top P_1 x$$

$$P_1 = Q + K_1^\top R K_1 + (A + B K_1)^\top P_0 (A + B K_1)$$

$$K_1 = -(R + B^\top P_0 B)^{-1} B^\top P_0 A$$

Turns into a recursion at time-to-go = i

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

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

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

Optimal controller is linear in x

Optimal cost is quadratic in x

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

Lecture Outline

Recap



Sampling-based MPC



Lyapunov Stability

Why might this not be enough?

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

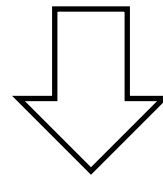
Non-linear

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

Non-quadratic

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

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



Idea 1: Shift the burden to brute-force computation

Idea 2: Convert into a locally linear system

Let's revisit ideas from Bayesian filtering

Linear Gaussian assumption

Sampling-based approximation

Filtering

Kalman Filtering

Particle Filtering

Control

LQR

Sampling based MPC

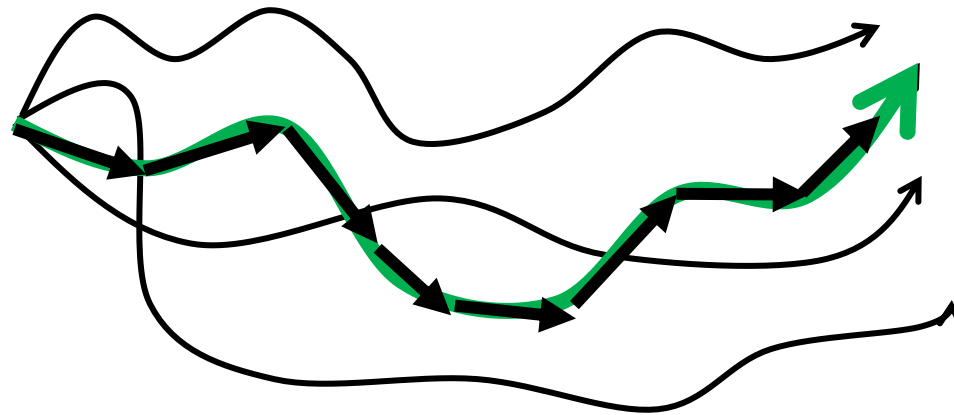
Solving Optimal Control with Sampling

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

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

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



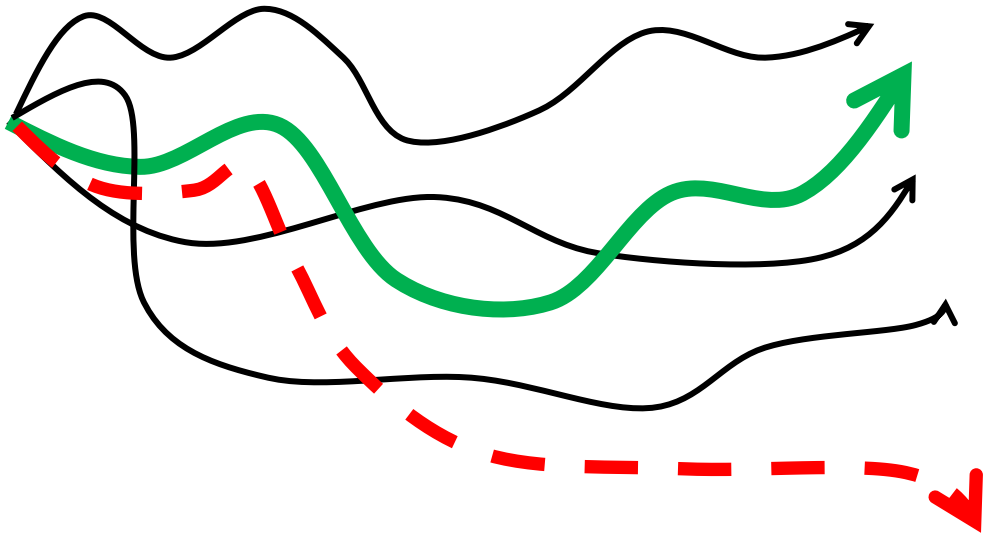
Can soften by taking softmin rather than argmin

Solving Optimal Control with Sampling – issues?

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

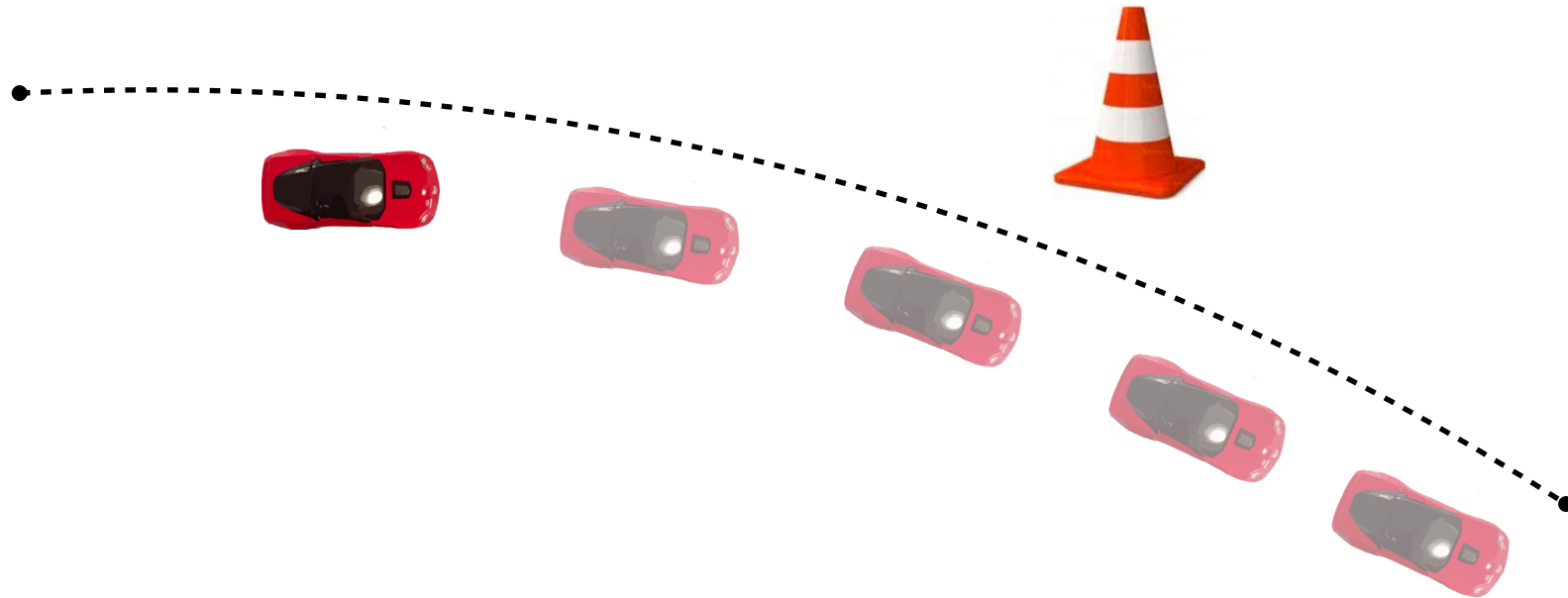
$$\text{s.t. } x_{t+1} = f(x_t, u_t)$$

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



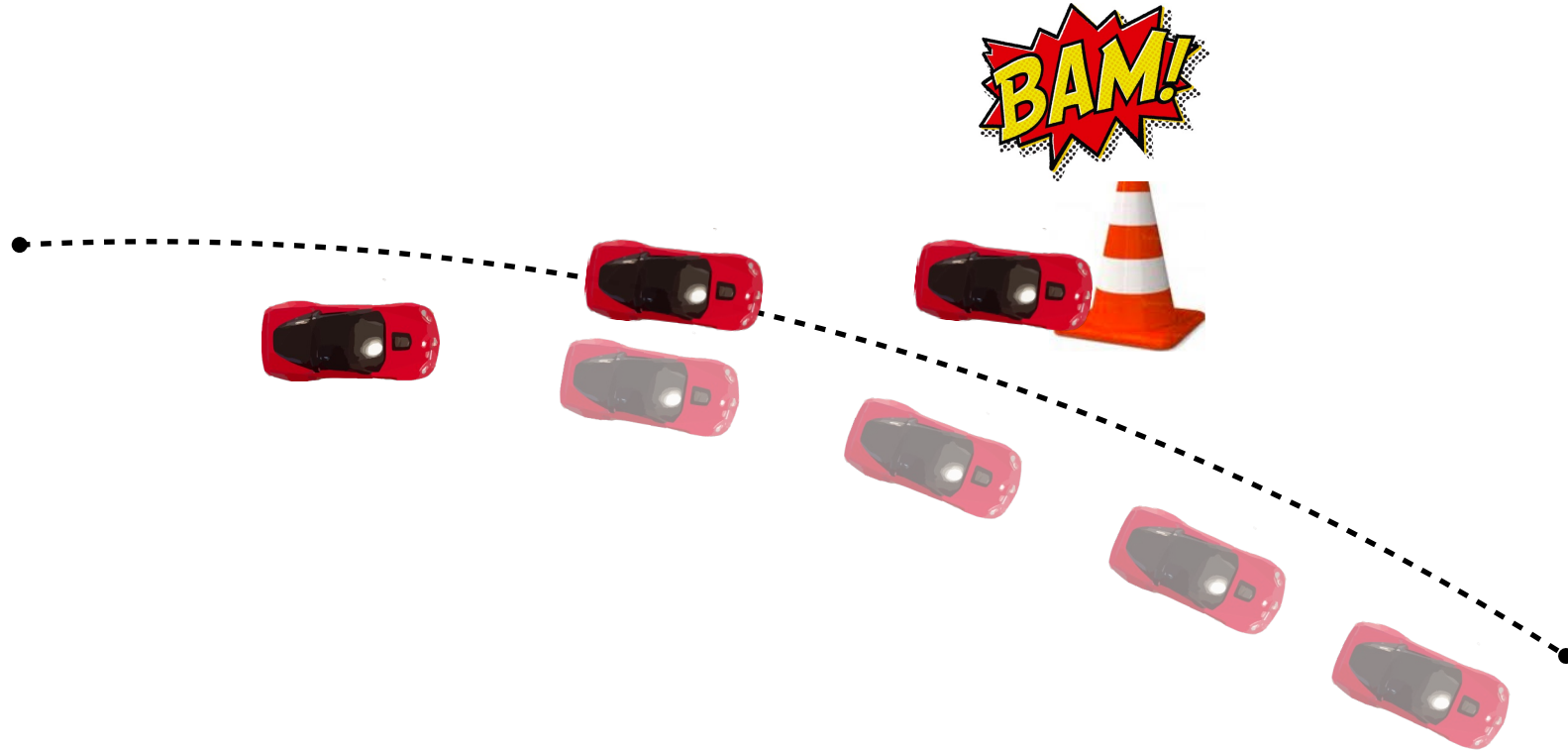
1. Open-loop controller may not be able to deal with unexpected events/divergences
2. Computation of full controller can be expensive:
→ Do it on the fly!
3. Model might be wrong, errors may accumulate
4. ...

Why do we need to replan?



What happens if the controls are planned once and executed?

Why do we need to replan?

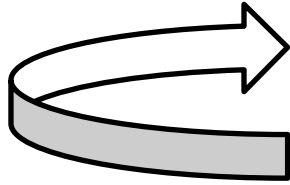


What happens if the controls are planned once and executed?

Solving Optimal Control with Sampling – issues?

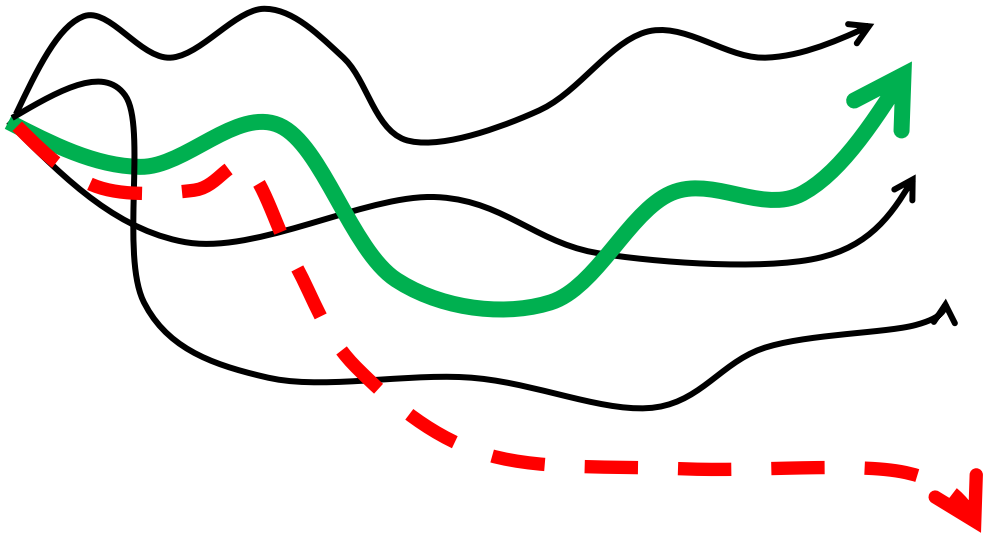
$$\min_{u_{1:T}} \sum_{t=1}^T c(x_t, u_t)$$

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

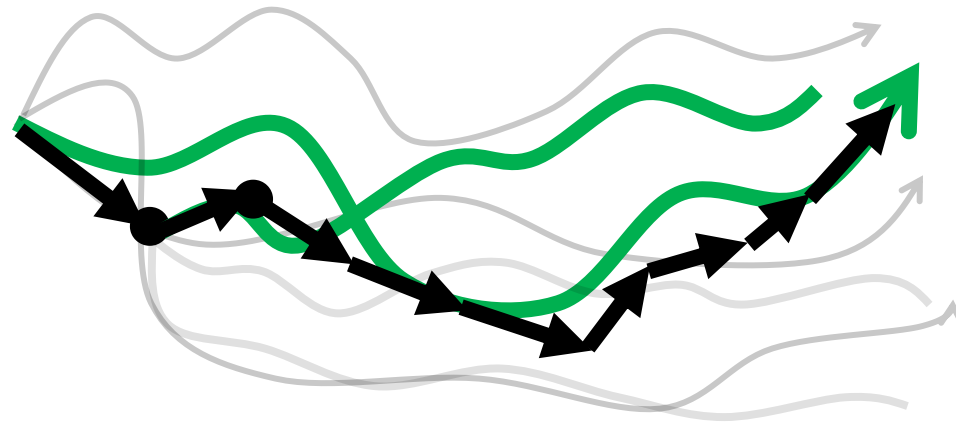


1. Plan with random shooting from s_t
2. Execute the first action a_0 and reach s_{t+1}

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

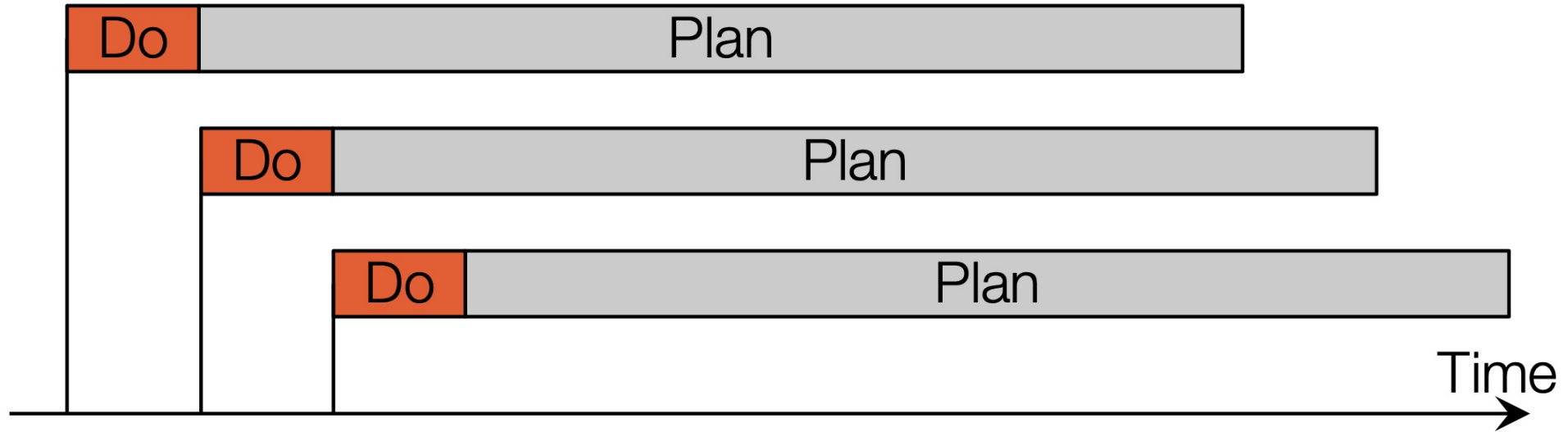


Replanning can help with divergence



Model-Predictive/Receding Horizon Control

General Replanning Framework - MPC

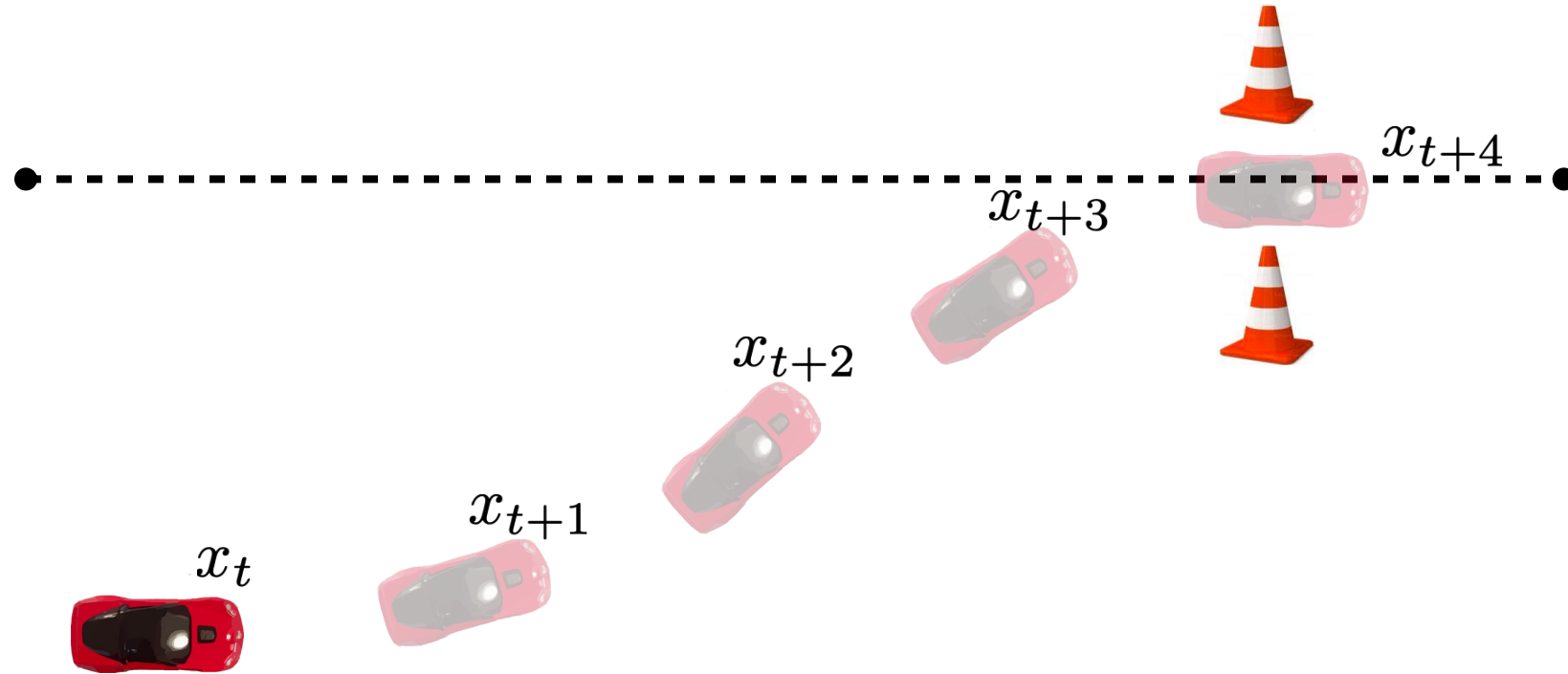


Step 1: Solve optimization problem to a horizon

Step 2: Execute the first control

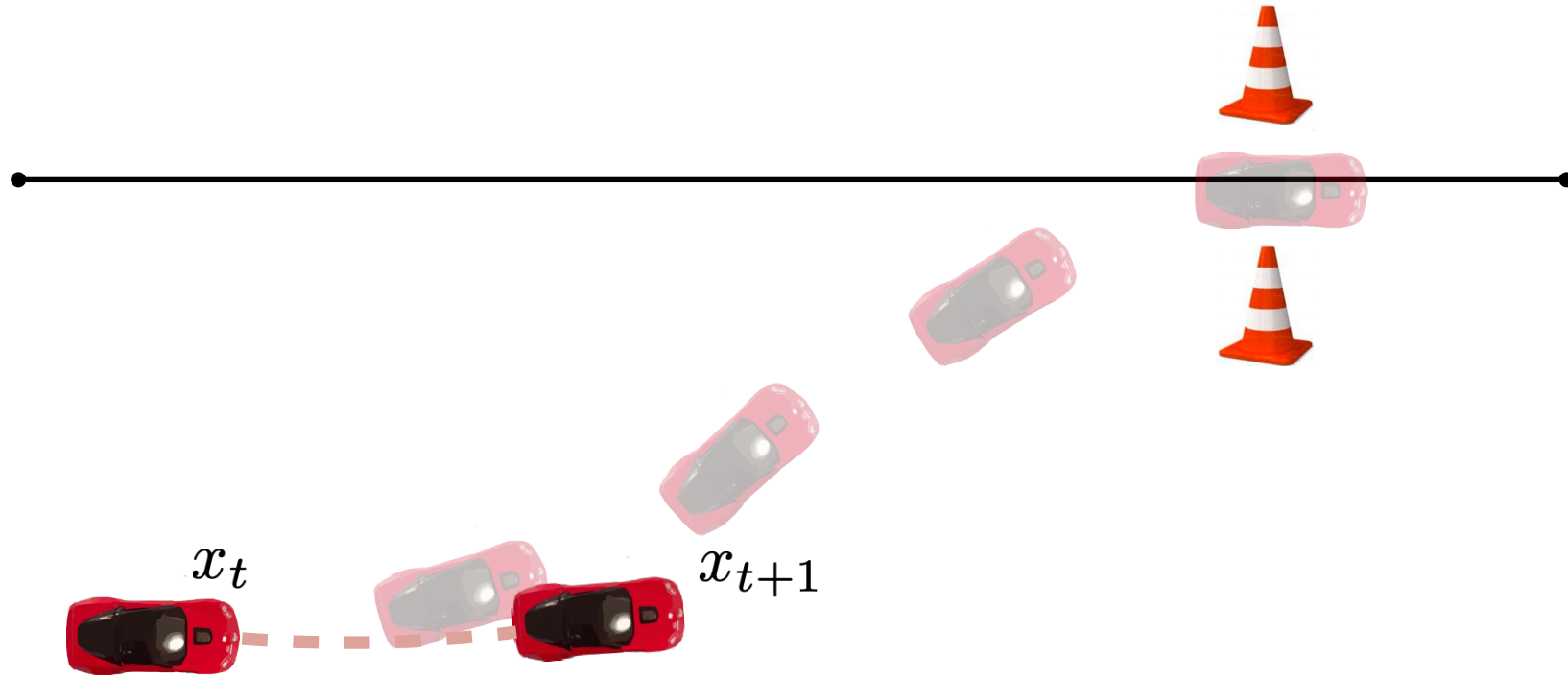
Step 3: Repeat!

How are the controls executed?



Step 1: Solve optimization problem to a horizon

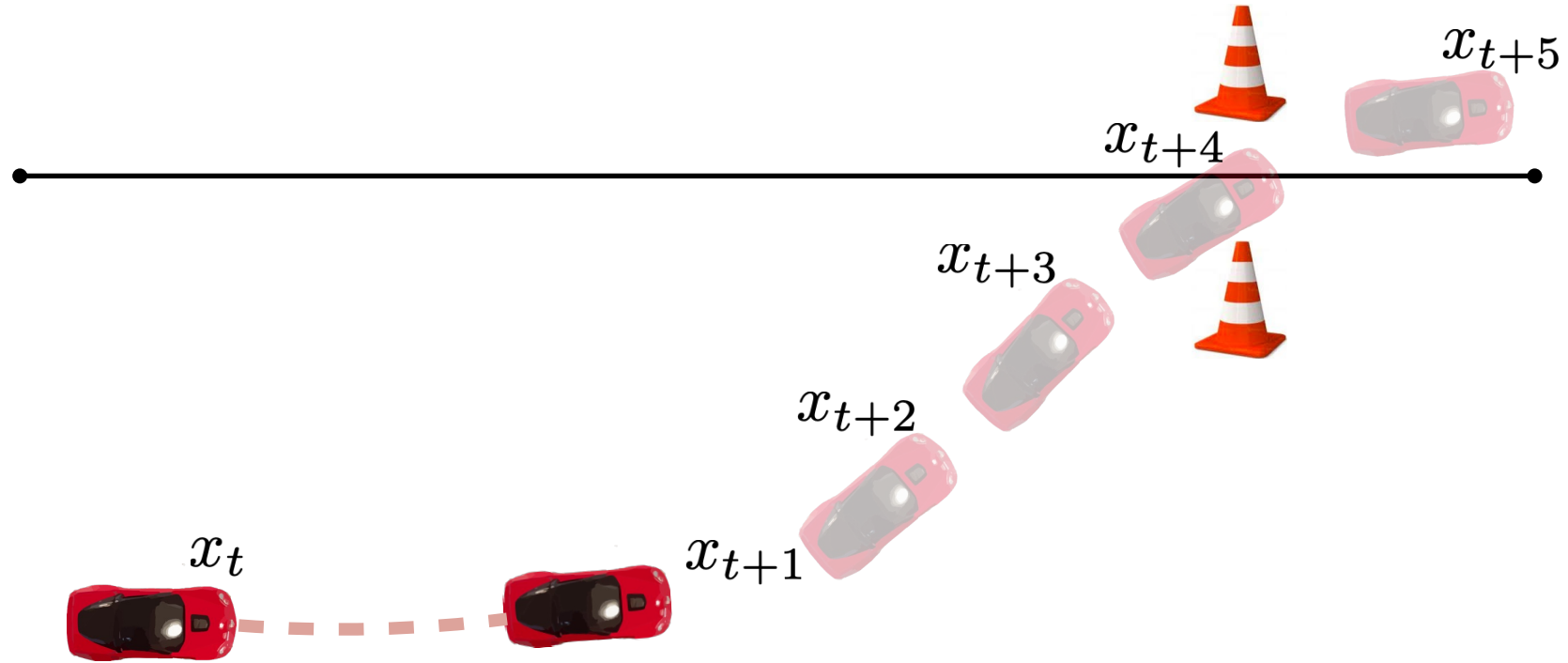
How are the controls executed?



Step 1: Solve optimization problem to a horizon

Step 2: Execute the first control

How are the controls executed?

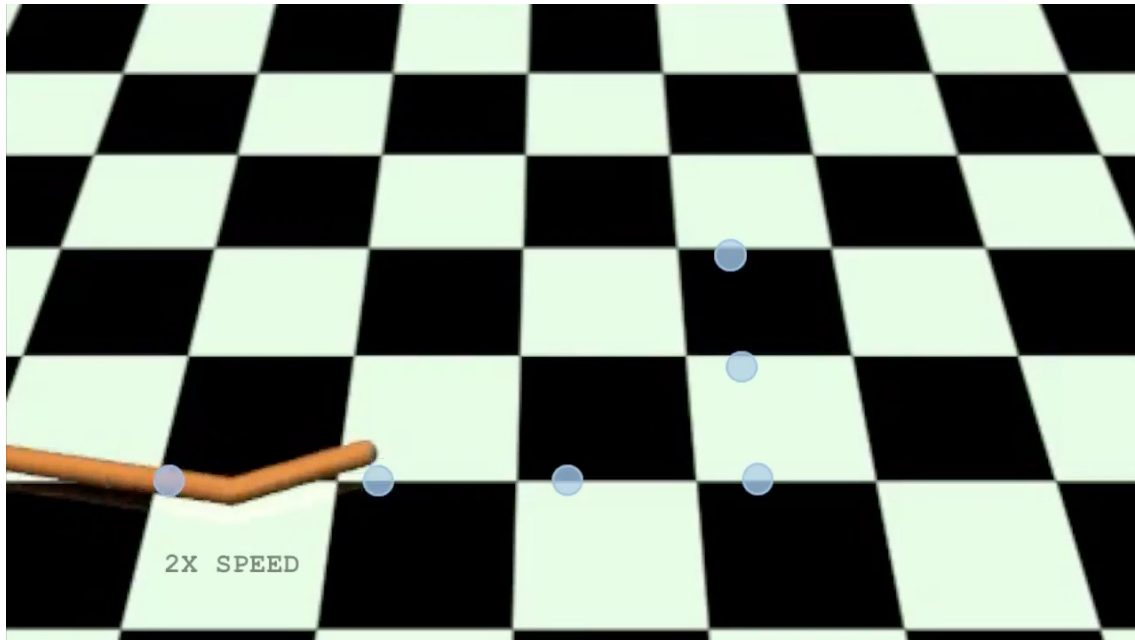


Step 1: Solve optimization problem to a horizon

Step 2: Execute the first control

Step 3: Repeat!

Does it work?



Why might this not work?

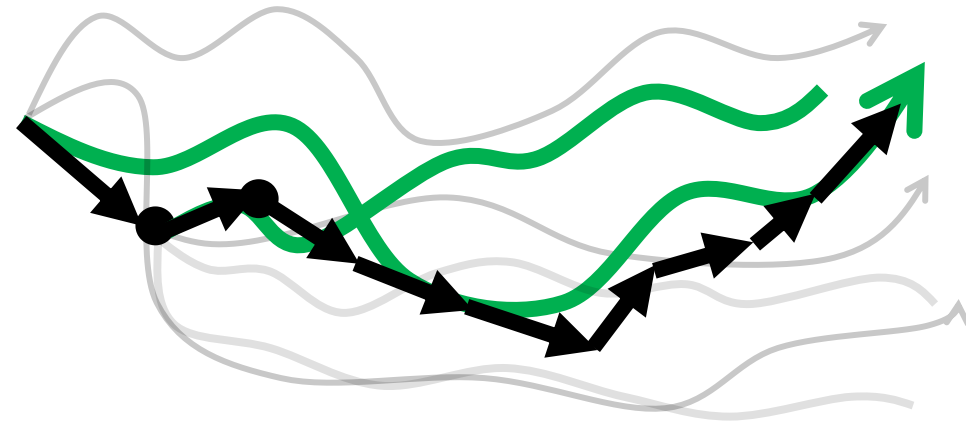
$$\min_{u_{1:T}} \sum_{t=1}^T c(x_t, u_t)$$

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

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

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

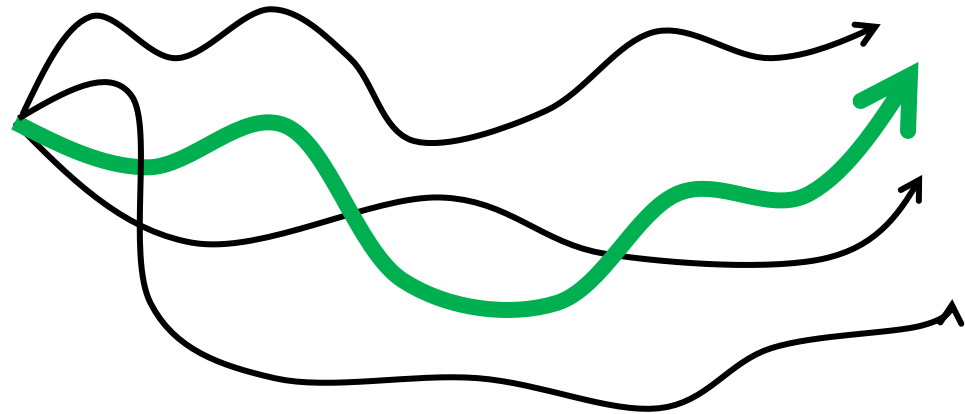
Planning with Shooting + MPC



Better Sampling Techniques for MPC

Sampled from stationary uniform/gaussian distribution

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



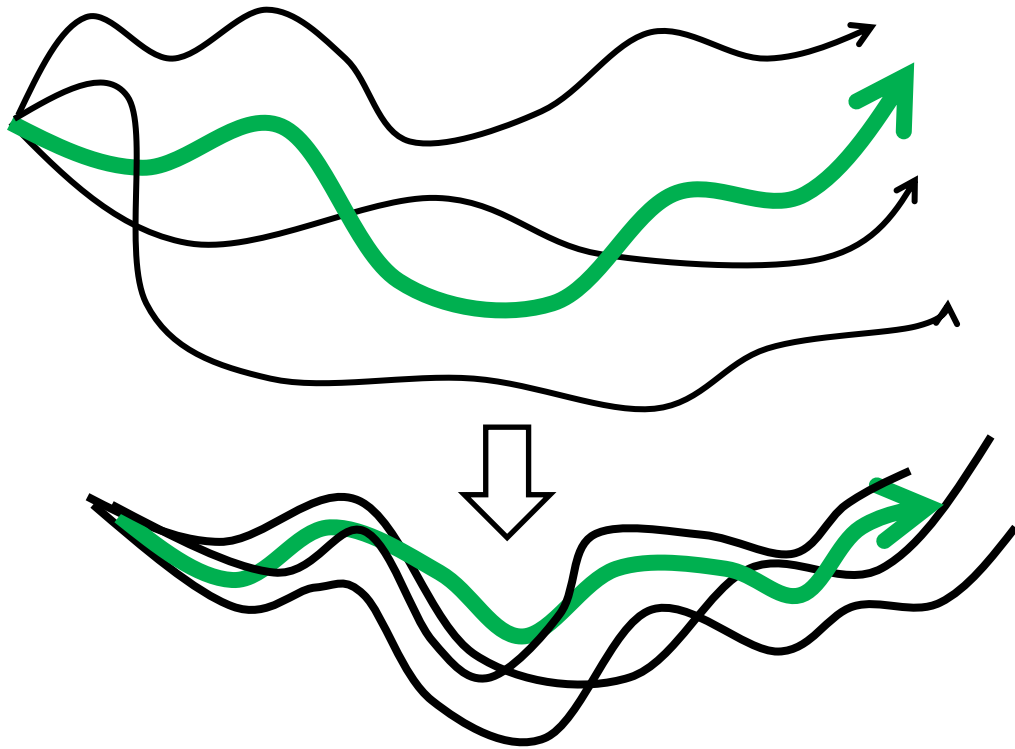
Can we inform the sampling function with the cost function?



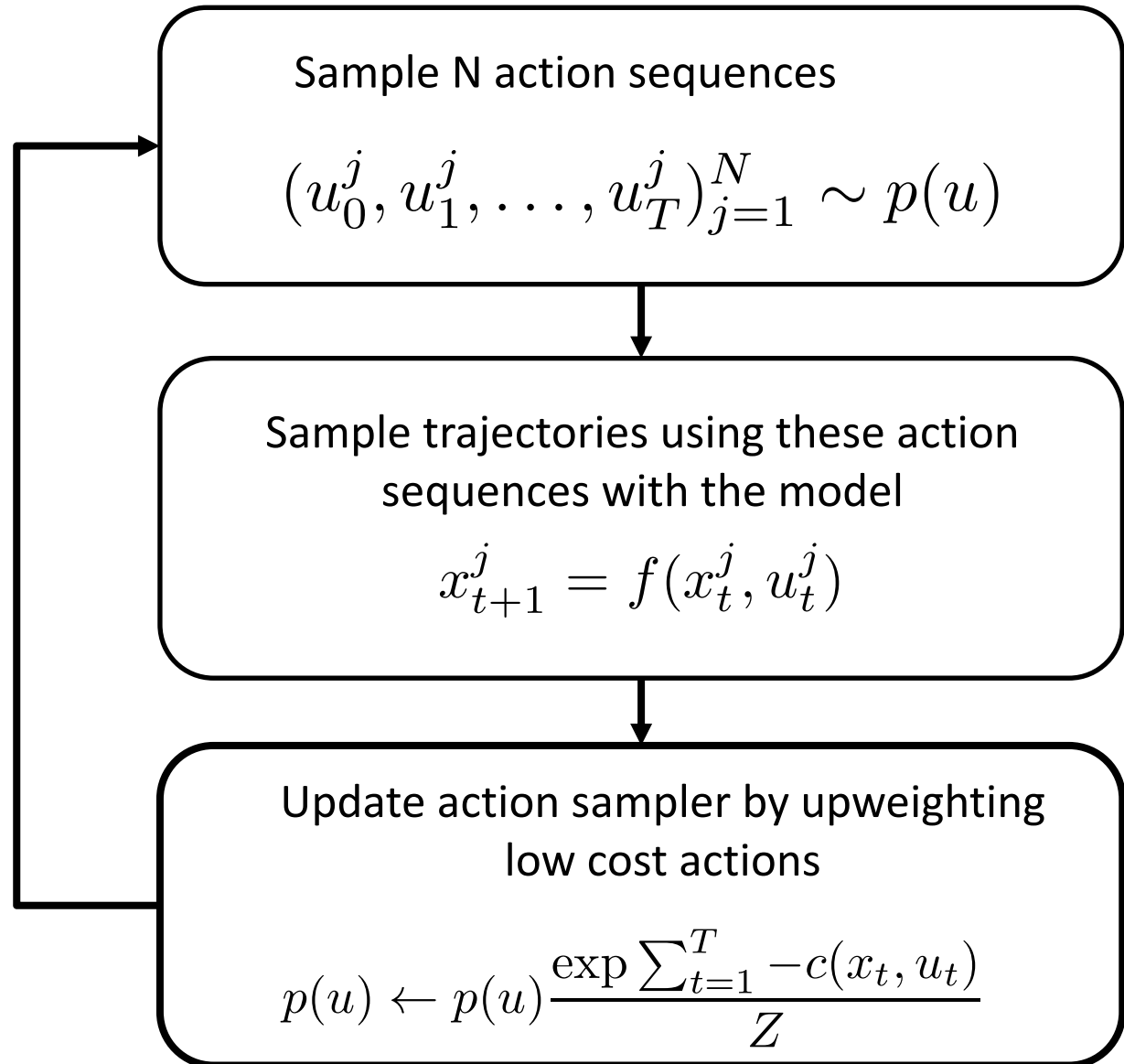
Idea: Iteratively upweight sampling distribution around the things that are lower cost

Better Sampling Techniques for Shooting - MPPI

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



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



Does it work?



Does it work?



Lecture Outline

Recap



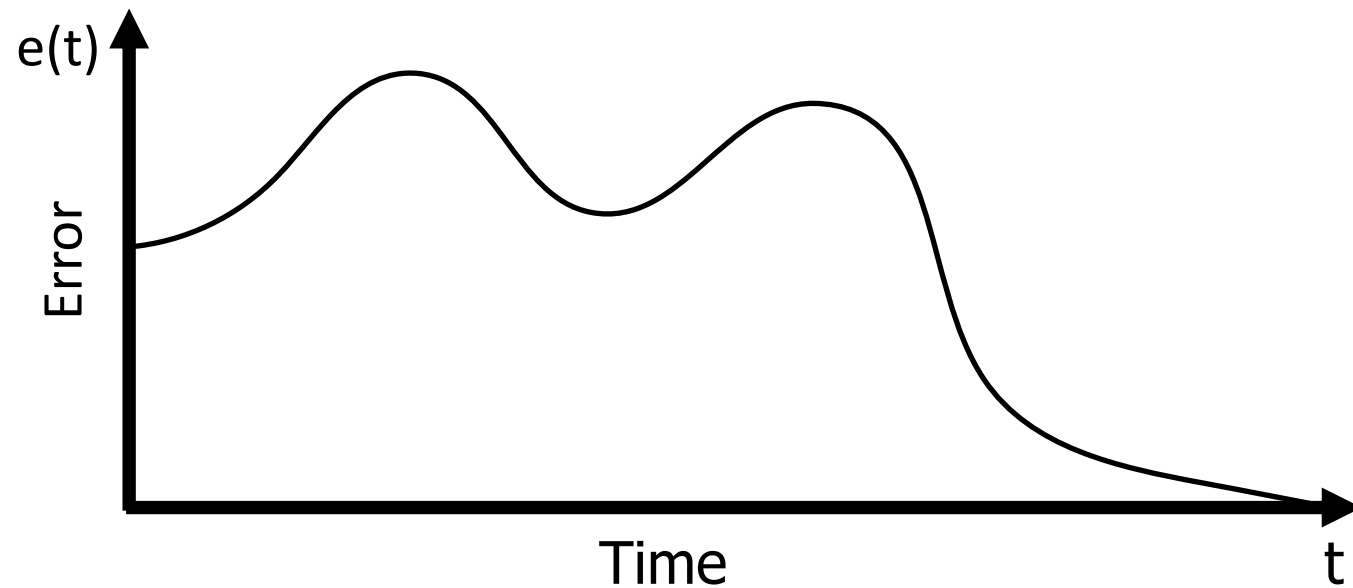
Sampling-based MPC



Lyapunov Stability

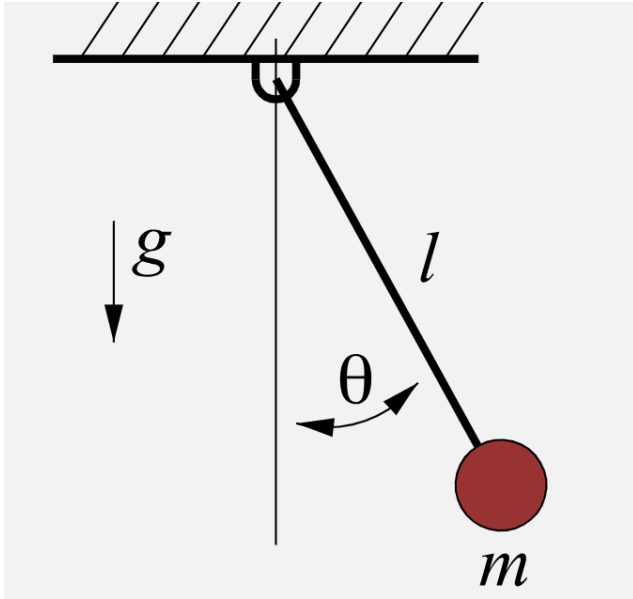
What is stability?

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



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

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.

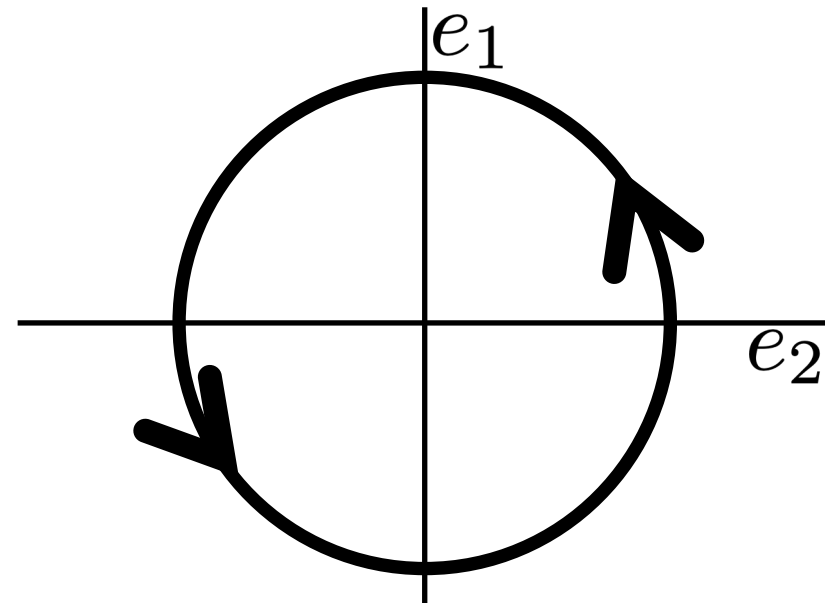
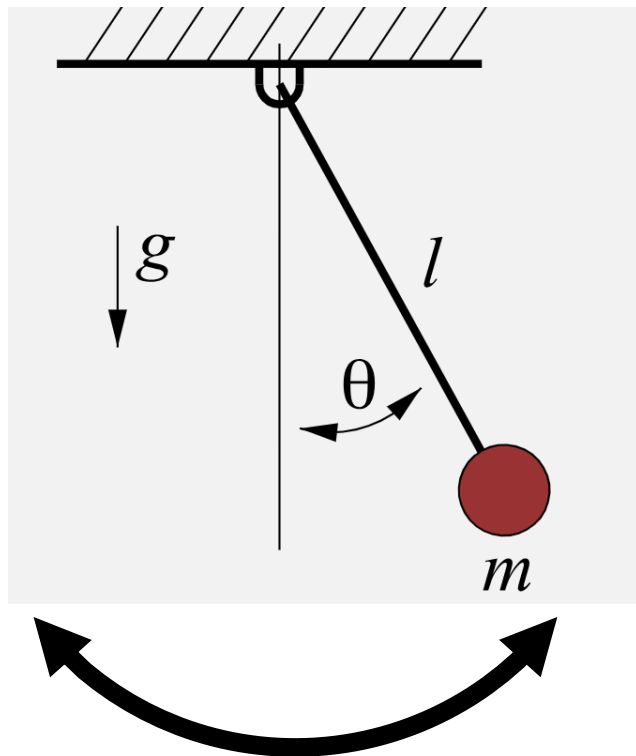
Choose $u = \pi(\theta, \dot{\theta})$ such that $\theta \rightarrow 0$
 $\dot{\theta} \rightarrow 0$

How does the **passive** error dynamics behave?

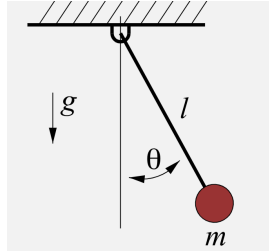
$$e_1 = \theta - 0 = \theta$$

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

Set $u=0$. Dynamics is not stable.



How do we verify if a controller is stable?



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

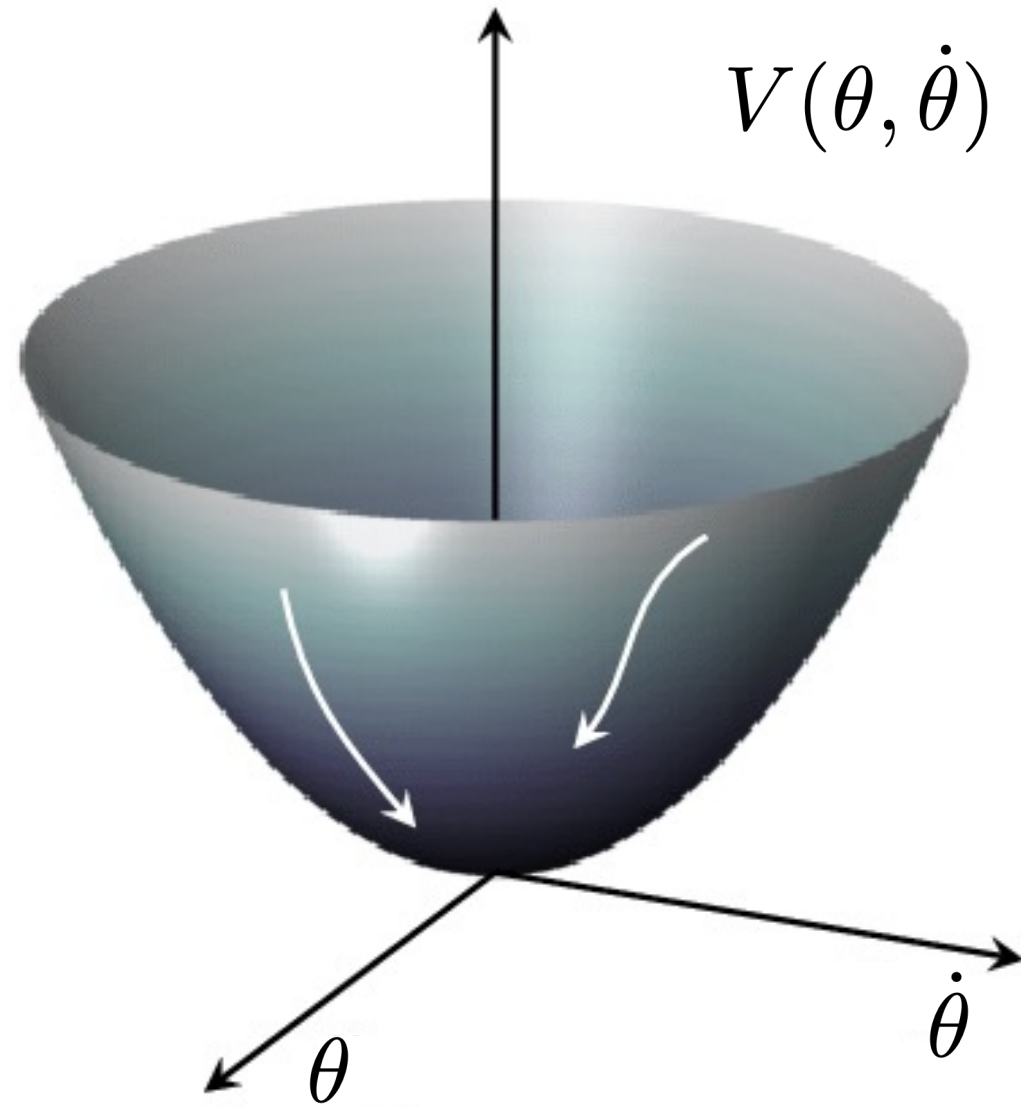
Lets pick the following law:

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

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

$$V(\theta, \dot{\theta}) = \frac{1}{2}ml^2\dot{\theta}^2 + mgl(1 - \cos \theta)$$
$$> 0$$

$$\dot{V}(\theta, \dot{\theta}) = ml^2\dot{\theta}\ddot{\theta} + mgl(\sin \theta)\dot{\theta}$$
$$= \dot{\theta}(u - mgl \sin \theta) + mgl(\sin \theta)\dot{\theta}$$
$$= \dot{\theta}u$$

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

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

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?

Option 1:

Use Lyapunov theory to **construct** stable controllers

Option 2:

Use Lyapunov theory to **verify** controllers for stability

