Robotics

Spring 2023

Abhishek Gupta

TAs: Yi Li, Srivatsa GS

Most slides courtesy of Sidd Srinivasa (UW), Pieter Abbeel (UC Berkeley), Dieter Fox (UW)

Recap: Course Overview

- Filtering/Smoothing
- Localization
- Mapping
- SLAM

- Search
- Motion Planning
- TrajOpt
- Stability/Certification

- MDPs and RL
- Imitation Learning
- Solving POMDPs

Lecture Outline

$$\begin{tikzcd}
\text{LQR} \arrow[d] \\
\text{Optimal Control: Collocation and Shooting} \arrow[d] \\
\text{Lyapunov Stability}
\end{tikzcd}$$

Goal of Optimal Control

- Minimize sum of costs, subject to dynamics and other constraints

$$\min_{u_{1:T}} \sum_{t=1}^{T} g(x_t, u_t) + G(x_T, u_T)$$

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

Can be costs like smoothness, preferences, speed

Can be constraints like velocity/acceleration bounds

From Motion Planning to PID to Optimal Control

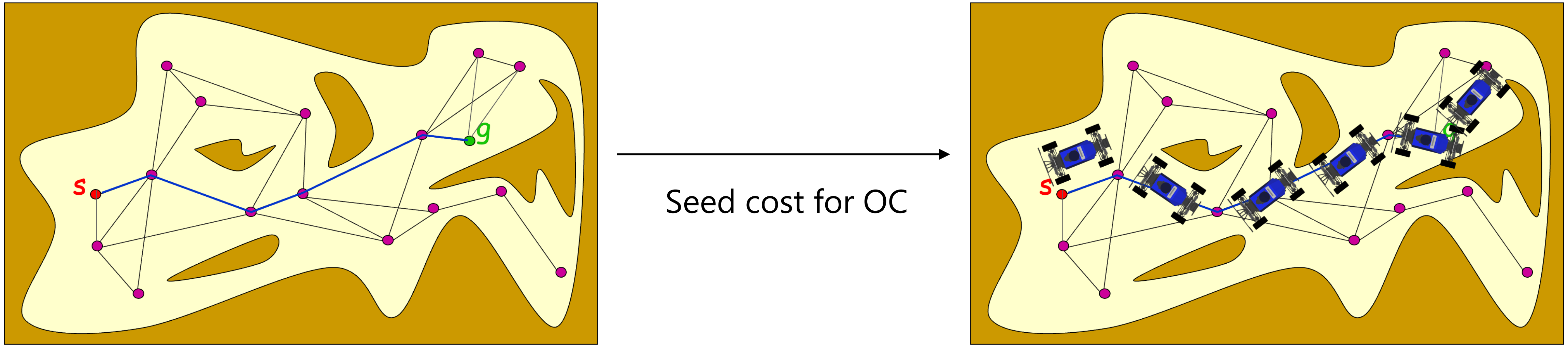
- Allows for integration of other costs/preferences
- Can be parameterized as closed loop

$$\begin{aligned}\min_{K_{1:T}} \quad & \sum_{t=0}^{T-1} g(x_t, u_t) + g(x_T, u_T) \\ \text{s.t.} \quad & x_{t+1} = f(x_t, u_t) \\ & u_t = K_t(x_t)\end{aligned}$$

- Allows for "optimal" closed-loop controllers, stable under disturbances

From Motion Planning to PID to Optimal Control

- Motion planning and optimal control are not orthogonal → complementary
- Can set costs for optimal control to track motion planning solution



Tractable Optimal Control Problem - LQR

- Optimal Control for Linear Dynamical Systems and Quadratic Cost (aka LQ setting, or LQR setting)
  - Very special case: can solve continuous state-space optimal control problem exactly and only requires performing linear algebra operations
  - Running time: O(H n^3)

Note 1: Great reference [optional] Anderson and Moore, Linear Quadratic Methods

Note2 : Strong similarity with Kalman filtering, which is able to compute the Bayes' filter updates exactly even though in general there are no closed form solutions and numerical solutions scale poorly with dimensionality.

Linear Quadratic Regulator (LQR)

The LQR setting assumes a linear dynamical system:

$$x_{t+1} = Ax_t + Bu_t,$$

x_t: state at time t
u_t: input at time t
It assumes a quadratic cost function:

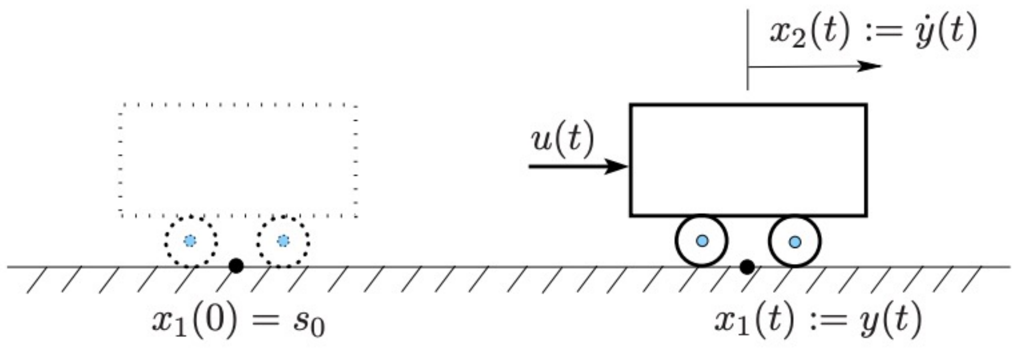
$$g(x_t, u_t) = x_t^\top Q x_t + u_t^\top R u_t$$

with $Q \succ 0, R \succ 0$.

For a square matrix X we have $X \succ 0$ if and only if for all vectors z we have $z^\top X z > 0$. Hence there is a non-zero cost for any state different from the all-zeros state, and any input different from the all-zeros input.

Examples of LQR systems

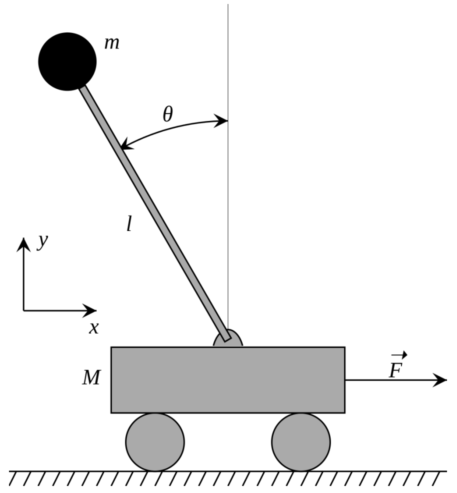
Double Integrator

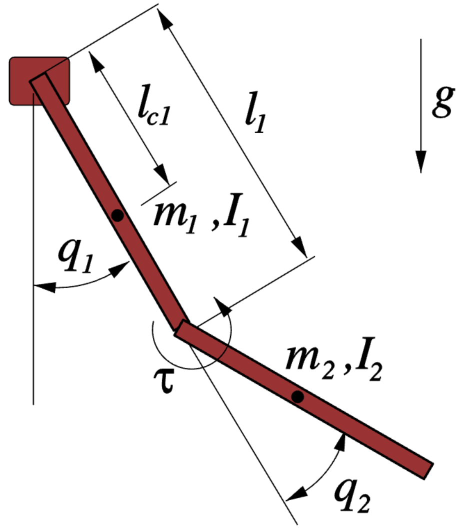


$$\ddot{q} = u(t) \qquad \dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mathbf{u}(t)$$

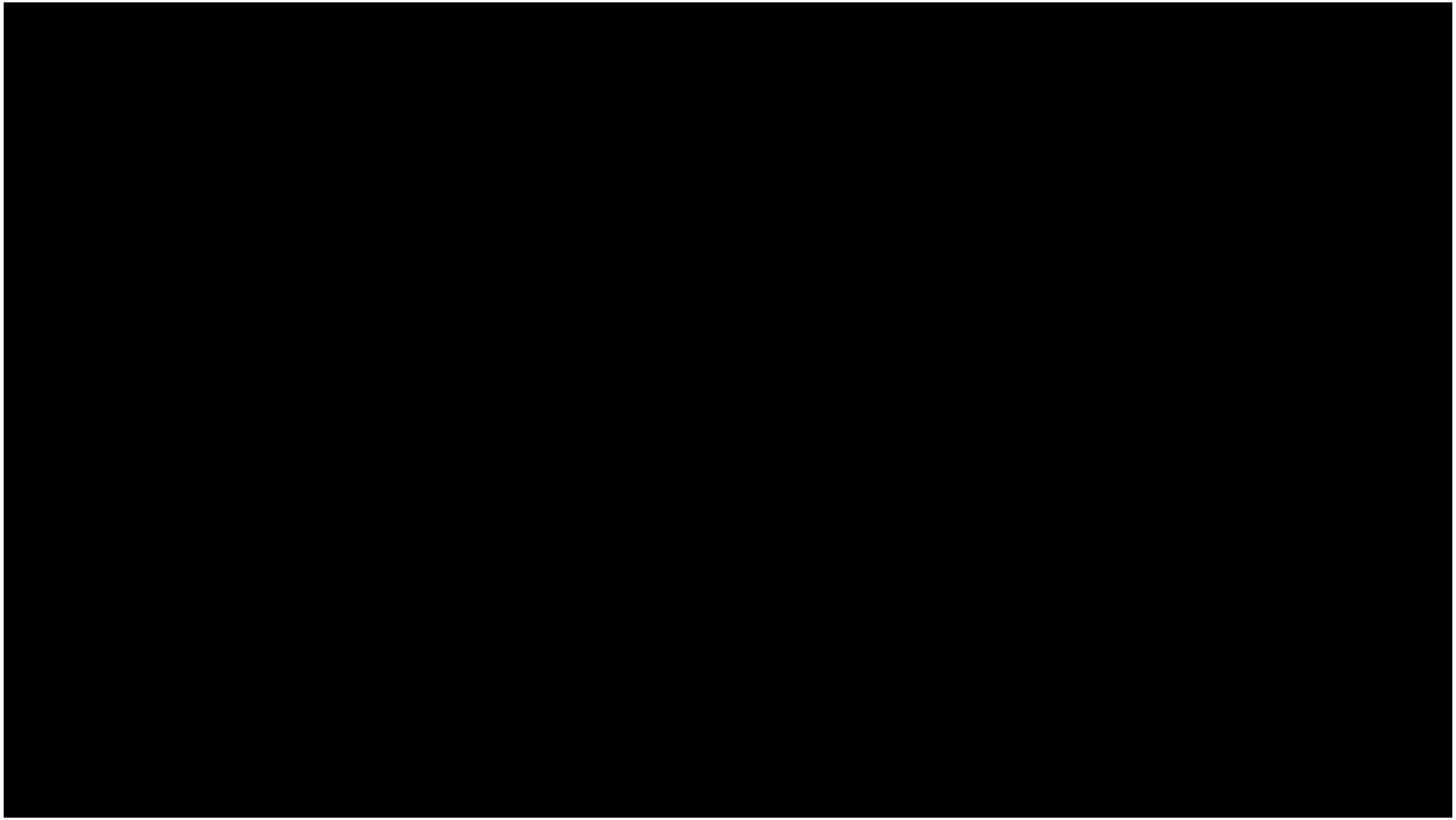
$$y = q(t) \qquad \mathbf{y}(t) = [1 \quad 0]\mathbf{x}(t).$$

Many systems can be easily linearized





Extension to Non-Linear Systems



LQR Problem Statement

- Minimize sum of costs subject to linear dynamics constraints

$$\min_{u_{1:T}} \sum_{t=1}^{T} g(x_t, u_t) + G(x_T, u_T)$$

$$\text{s.t.} \quad x_{t+1} = Ax_t + Bu_t$$

$$\min_{K_{1:t}} \sum_{t=1}^{T} g(x_t, u_t) + G(x_T, u_T)$$

$$\begin{aligned} \text{s.t.} \quad x_{t+1} &= Ax_t + Bu_t \\ u_t &= K_t(x_t) \end{aligned}$$

Recursive Solution to LQR Problem

- LQR:

$$J_{i+1}(x) = \min_u x^\top Qx + u^\top Ru + \sum_{x'=Ax+Bu} J_i(x')$$

$$= \min_u \left[x^\top Qx + u^\top Ru + J_i(Ax + Bu)\right]$$

LQR value iteration: J_1

$$J_{i+1}(x) \leftarrow \min_u \left[x^\top Q x + u^\top R u + J_i(Ax+Bu) \right]$$

Initialize $J_0(x) = x^\top P_0 x$.

$$\begin{aligned} J_1(x) &= \min_u \left[x^\top Q x + u^\top R u + J_0(Ax+Bu) \right] \\ &= \min_u \left[x^\top Q x + u^\top R u + (Ax+Bu)^\top P_0 (Ax+Bu) \right] \qquad (1) \end{aligned}$$

To find the minimum over u, we set the gradient w.r.t. u equal to zero:

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

$$\text{hence: } u = -(R + B^\top P_0 B)^{-1} B^\top P_0 A x \qquad (2)$$

$$\begin{aligned} \text{(2) into (1): } J_1(x) &= x^\top P_1 x \\ \text{for: } P_1 &= Q + K_1^\top R K_1 + (A + BK_1)^\top P_0 (A + BK_1) \\ K_1 &= -(R + B^\top P_0 B)^{-1} B^\top P_0 A. \end{aligned}$$

LQR value iteration: J_1 (ctd)

- In summary:

$$J_0(x) = x^\top P_0 x$$
$$x_{t+1} = Ax_t + Bu_t$$
$$g(x,u) = u^\top Ru + x^\top Qx$$

$$\begin{aligned} J_1(x) &= x^\top P_1 x \\ \text{for: } P_1 &= Q + K_1^\top RK_1 + (A + BK_1)^\top P_0 (A + BK_1) \\ K_1 &= -(R + B^\top P_0 B)^{-1} B^\top P_0 A. \end{aligned}$$

- $J_1(x)$ is quadratic, just like $J_0(x)$.

  → Value iteration update is the same for all times and can be done in closed form for this particular continuous state-space system and cost!

$$\begin{aligned} J_2(x) &= x^\top P_2 x \\ \text{for: } P_2 &= Q + K_2^\top RK_2 + (A + BK_2)^\top P_1 (A + BK_2) \\ K_2 &= -(R + B^\top P_1 B)^{-1} B^\top P_1 A. \end{aligned}$$

Value iteration solution to LQR

```
Set $P_0 = 0$.
for $i = 1, 2, 3, \ldots$
```

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

The optimal policy for a i-step horizon is given by:

$$\pi(x) = K_i x$$

The cost-to-go function for a i-step horizon is given by:

$$J_i(x) = x^\top P_i x.$$

LQR assumptions revisited

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t \\ g(x_t, u_t) &= x_t^\top Q x_t + u_t^\top R u_t \end{aligned}$$

= for keeping a linear system at the all-zeros state while preferring to keep the control input small.

- Extensions make it more generally applicable:
  - Affine systems
  - Systems with stochasticity
  - Penalization for change in control inputs
  - Linear time varying (LTV) systems
  - Trajectory following for non-linear systems

LQR Ext0: Affine systems

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + c \\ g(x_t, u_t) &= x_t^\top Q x_t + u_t^\top R u_t \end{aligned}$$

- Optimal control policy remains linear, optimal cost-to-go function remains quadratic
- Two avenues to do derivation:
  - 1. Re-derive the update, which is very similar to what we did for standard setting
  - 2. Re-define the state as: $z_t = [x_t; 1]$, then we have:

$$z_{t+1} = \begin{bmatrix} x_{t+1} \\ 1 \end{bmatrix} = \begin{bmatrix} A & c \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ 1 \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u_t = A' z_t + B' u_t$$

LQR Ext1: stochastic system

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + w_t \\ g(x_t, u_t) &= x_t^\top Q x_t + u_t^\top R u_t \\ & w_t, t = 0, 1, \dots \text{are zero mean and independent} \end{aligned}$$

- Exercise: work through similar derivation as we did for the deterministic case, but which will now have expectations.
- Result:
  - Same optimal control policy
  - Cost-to-go function is almost identical: has one additional term which depends on the variance in the noise (and which cannot be influenced by the choice of control inputs)

LQR Ext2: 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]

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

LQR Ext3: Penalize for Change in Control Inputs

- Standard LQR:

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t \\ g(x_t, u_t) &= x_t^\top Q x_t + u_t^\top R u_t \end{aligned}$$

- When run in this format on real systems: often high frequency control inputs get generated. Typically highly undesirable and results in poor control performance.

- Why?

- Simple special case which works well in practice: penalize for change in control inputs. ---- How ??

LQR Ext3: Penalize for Change in Control Inputs

- Standard LQR:

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t \\ g(x_t, u_t) &= x_t^\top Q x_t + u_t^\top R u_t \end{aligned}$$

- How to incorporate the change in controls into the cost/reward function?
  - Idea 1: explicitly incorporate into the state by augmenting the state with the past control input vector, and the difference between the last two control input vectors.
  - Idea 2: change of control input variables.

LQR Ext3: Penalize for Change in Control Inputs

- Standard LQR:

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t \\ g(x_t, u_t) &= x_t^\top Q x_t + u_t^\top R u_t \end{aligned}$$

- Introducing change in controls Δu:

$$\underbrace{\begin{bmatrix} x_{t+1} \\ u_t \end{bmatrix}}_{x'_{t+1}} = \underbrace{\begin{bmatrix} A & B \\ 0 & I \end{bmatrix}}_{A'} \underbrace{\begin{bmatrix} x_t \\ u_{t-1} \end{bmatrix}}_{x'_t} + \underbrace{\begin{bmatrix} B \\ I \end{bmatrix}}_{B'} \underbrace{\Delta u_t}_{u'_t}$$

$$\text{cost} = -(x'^\top Q' x' + \Delta u^\top R' \Delta u) \qquad Q' = \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix}$$

$$R' = \text{penalty for change in controls}$$

[If R'=0, then "equivalent" to standard LQR.]

LQR Ext4: Linear Time Varying (LTV) Systems

$$\begin{aligned} x_{t+1} &= A_t x_t + B_t u_t \\ g(x_t, u_t) &= x_t^\top Q_t x_t + u_t^\top R_t u_t \end{aligned}$$

LQR Ext4: Linear Time Varying (LTV) Systems

Set $P_0 = 0$.
for $i = 1, 2, 3, \ldots$

$$\begin{aligned} K_i &= -(R_{H-i} + B_{H-i}^\top P_{i-1} B_{H-i})^{-1} B_{H-i}^\top P_{i-1} A_{H-i} \\ P_i &= Q_{H-i} + K_i^\top R_{H-i} K_i + (A_{H-i} + B_{H-i} K_i)^\top P_{i-1} (A_{H-i} + B_{H-i} K_i) \end{aligned}$$

The optimal policy for a i-step horizon is given by:

$$\pi(x) = K_i x$$

The cost-to-go function for a i-step horizon is given by:

$$J_i(x) = x^\top P_i x.$$

LQR Ext5: Trajectory Following for Non-Linear Systems

- A state sequence x_0^*, x_1^*, …, x_H^* is a feasible target trajectory if and only if

$$\exists u_0^*, u_1^*, \ldots, u_{H-1}^* \; : \; \forall t \in \{0, 1, \ldots, H-1\} \; : \; x_{t+1}^* = f(x_t^*, u_t^*)$$

- Problem statement:

$$\min_{u_0, u_1, \ldots, u_{H-1}} \sum_{t=0}^{H-1} (x_t - x_t^*)^\top Q (x_t - x_t^*) + (u_t - u_t^*)^\top R (u_t - u_t^*)$$

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

- Transform into linear time varying case (LTV):

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

$$x_{t+1} - x_{t+1}^* \approx A_t (x_t - x_t^*) + B_t (u_t - u_t^*)$$

LQR Ext5: Trajectory Following for Non-Linear Systems

- Transformed into linear time varying case (LTV):

$$\min_{u_0,u_1,\ldots,u_{H-1}} \sum_{t=0}^{H-1} (x_t - x_t^*)^\top Q (x_t - x_t^*) + (u_t - u_t^*)^\top R (u_t - u_t^*)$$

$$\text{s.t. } x_{t+1} - x_{t+1}^* = A_t(x_t - x_t^*) + B_t(u_t - u_t^*)$$

- Now we can run the standard LQR back-up iterations.
- Resulting policy at i time-steps from the end:

$$u_{H-i} - u_{H-i}^* = K_i(x_{H-i} - x_{H-i}^*)$$

- The target trajectory need not be feasible to apply this technique, however, if it is infeasible then there will an offset term in the dynamics:

$$x_{t+1} - x_{t+1}^* = f(x_t, u_t) - x_{t+1}^* + A_t(x_t - x_t^*) + B_t(u_t - u_t^*)$$

Most General Case

- How about this general optimal control problem?

$$\min_{u_0,\ldots,u_H} \sum_{t=0}^{H} g(x_t, u_t)$$

Iteratively Apply LQR

Initialize the algorithm by picking either (a) A control policy $\pi^{(0)}$ or (b) A sequence of states $x_0^{(0)}, x_1^{(0)}, \ldots, x_H^{(0)}$ and control inputs $u_0^{(0)}, u_1^{(0)}, \ldots, u_H^{(0)}$. With initialization (a), start in Step (1). With initialization (b), start in Step (2).

Iterate the following:

(1) Execute the current policy $\pi^{(i)}$ and record the resulting state-input trajectory $x_0^{(i)}, u_0^{(i)}, x_1^{(i)}, u_1^{(i)}, \ldots, x_H^{(i)}, u_H^{(i)}$.

(2) Compute the LQ approximation of the optimal control problem around the obtained state-input trajectory by computing a first-order Taylor expansion of the dynamics model, and a second-order Taylor expansion of the cost function.

(3) Use the LQR back-ups to solve for the optimal control policy $\pi^{(i+1)}$ for the LQ approximation obtained in Step (2).

(4) Set $i = i + 1$ and go to Step (1).

Iterative LQR in Standard LTV Format

Standard LTV is of the form:

$$\begin{aligned} z_{t+1} &= A_t z_t + B_t v_t \\ g(z,v) &= z^\top Q z + v^\top R v \end{aligned}$$

Linearizing f around $(x_t^{(i)}, u_t^{(i)})$ from the roll-out in iteration i of the iterative LQR algorithm gives us:

$$x_{t+1} \approx f(x_t^{(i)}, u_t^{(i)}) + \frac{\partial f}{\partial x}(x_t^{(i)}, u_t^{(i)})(x_t - x_t^{(i)}) + \frac{\partial f}{\partial u}(x_t^{(i)}, u_t^{(i)})(u_t - u_t^{(i)})$$

Keeping in mind that $x_{t+1}^{(i)} = f(x_t^{(i)}, u_t^{(i)})$ gives us:

$$x_{t+1} - x_{t+1}^{(i+1)} \approx \frac{\partial f}{\partial x}(x_t^{(i)}, u_t^{(i)})(x_t - x_t^{(i)}) + \frac{\partial f}{\partial u}(x_t^{(i)}, u_t^{(i)})(u_t - u_t^{(i)})$$

Hence we get the standard format if using:

$$\begin{aligned} z_t &= \begin{bmatrix} x_t - x_t^{(i)} & 1 \end{bmatrix}^\top \\ v_t &= (u_t - u_t^{(i)}) \\ A_t &= \begin{bmatrix} \frac{\partial f}{\partial x}(x_t^{(i)}, u_t^{(i)}) & 0 \\ 0 & 1 \end{bmatrix} \\ B_t &= \begin{bmatrix} \frac{\partial f}{\partial u}(x_t^{(i)}, u_t^{(i)}) \\ 0 \end{bmatrix} \end{aligned}$$

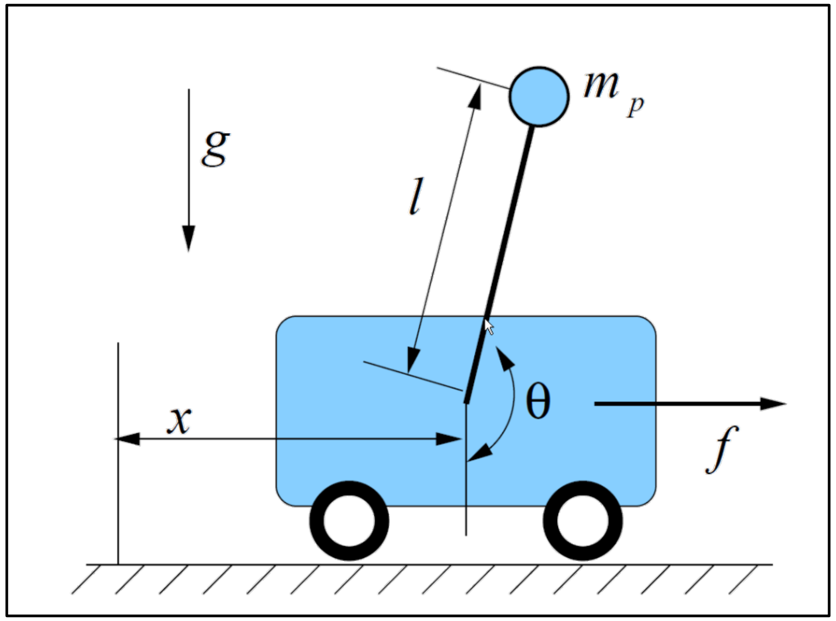
$$\begin{aligned} Q_t &= \begin{bmatrix} \frac{\partial^2 g}{\partial x^2}(x^{(i)}) & \frac{\partial g}{\partial x}(x^{(i)}) \\ \left(\frac{\partial g}{\partial x}(x^{(i)})\right)^\top & 2g(x^{(i)}) \end{bmatrix} \\ R_t &= \begin{bmatrix} \frac{\partial^2 g}{\partial u^2}(u^{(i)}) & \frac{\partial g}{\partial u}(u^{(i)}) \\ \left(\frac{\partial g}{\partial u}(u^{(i)})\right)^\top & 2g(u^{(i)}) \end{bmatrix} \end{aligned}$$

for simplicity and with some abuse of notation we assumed g(x,u) = g(x) + g(u)

Can We Do Even Better?

- Yes!
- At convergence of iLQR, we end up with linearizations around the (state,input) trajectory the algorithm converged to
- In practice: the system could not be on this trajectory due to perturbations / initial state being off / dynamics model being off / …
- Solution: at time t when asked to generate control input ut, we could re-solve the control problem using iLQR over the time steps t through H
- Replanning entire trajectory is often impractical → in practice: replan over horizon h. = ***receding horizon control***
  - This requires providing a cost to go $J^{(t+h)}$ which accounts for all future costs. This could be taken from the offline iLQR run
  - More on this later!

Cart-pole



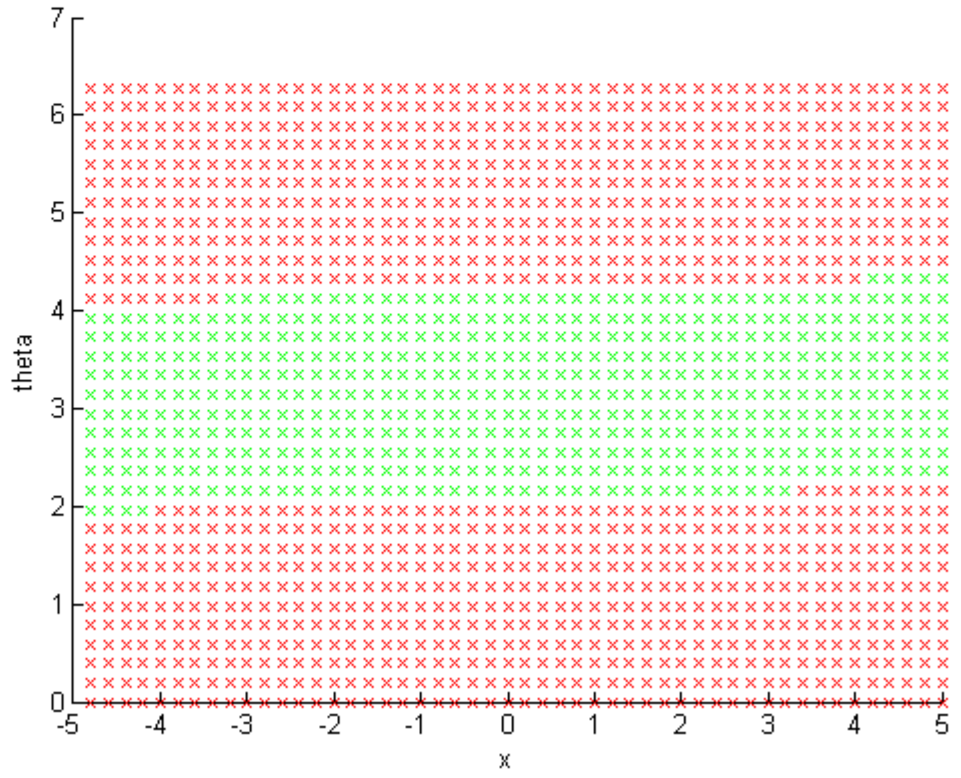
$$H(q)\ddot{q} + C(q,\dot{q}) + G(q) = B(q)u$$

$$
\begin{aligned}
H(q) &= \begin{bmatrix} m_c + m_p & m_p l\cos\theta \\ m_p l\cos\theta & m_p l^2 \end{bmatrix} \\
C(q,\dot{q}) &= \begin{bmatrix} 0 & -m_p l\dot{\theta}\sin\theta \\ 0 & 0 \end{bmatrix} \\
G(q) &= \begin{bmatrix} 0 \\ m_p g l\sin\theta \end{bmatrix} \\
B &= \begin{bmatrix} 1 \\ 0 \end{bmatrix}
\end{aligned}
$$

[See also Section 3.3 in Tedrake notes.]

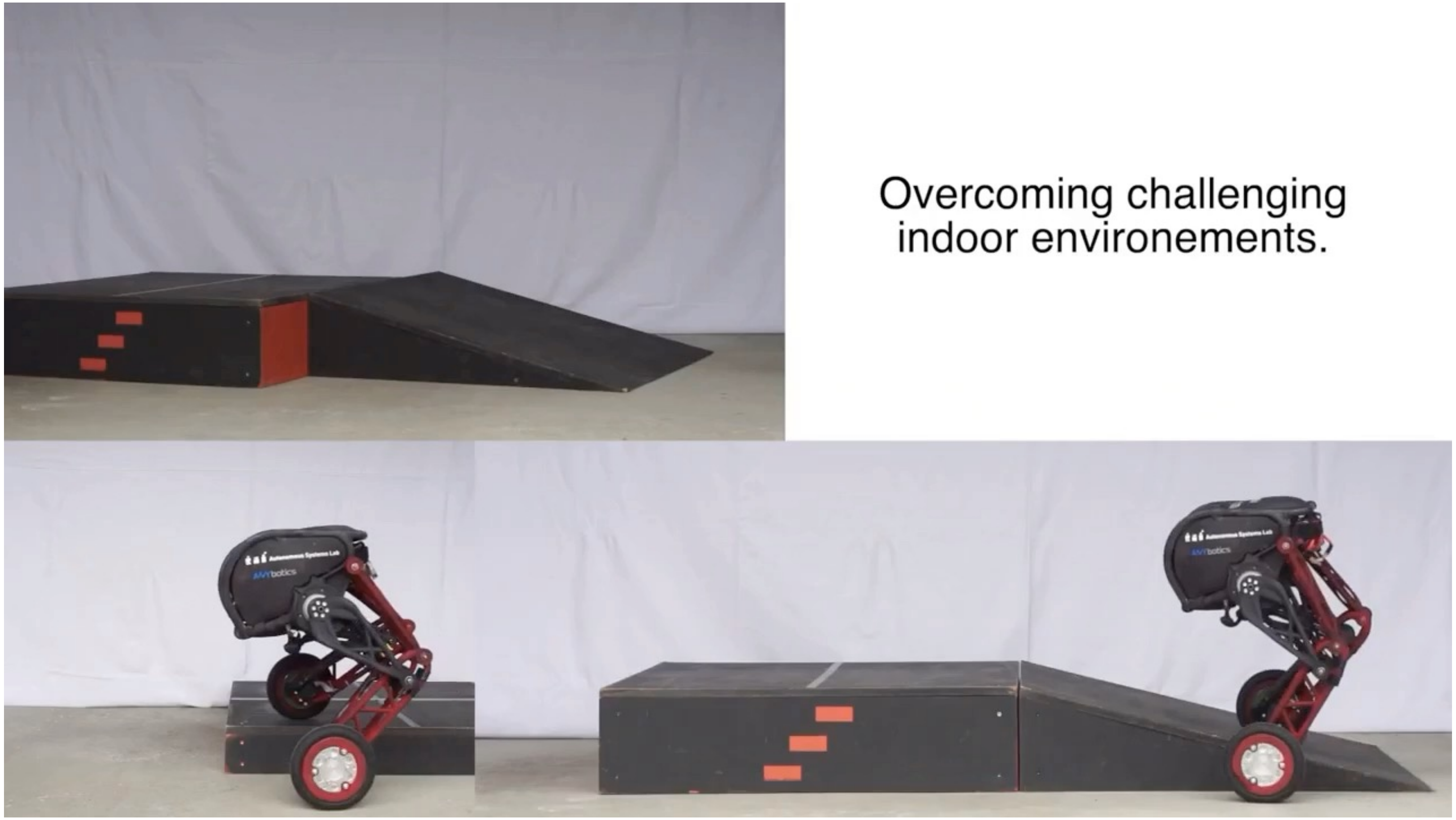
Cart-pole --- LQR

Results of running LQR for the linear time-invariant system obtained from linearizing around [0;0;0;0]. The cross-marks correspond to initial states. Green means the controller succeeded at stabilizing from that initial state, red means not.



Q = diag([1;1;1;1]); R = 0; [x, theta, xdot, thetadot]

LQR in Action



Overcoming challenging
indoor environements.

Klemm et al 2020

Learning Linear Dynamics Latent Spaces

- *Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images*
  Manuel Watter, Jost Tobias Springenberg, Joschka Boedecker, Martin Riedmiller
  https://arxiv.org/abs/1506.07365

- *Deep Spatial Autoencoders for Visuomotor Learning*
  Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, Pieter Abbeel
  https://arxiv.org/abs/1509.06113

- *SOLAR: Deep Structured Representations for Model-Based Reinforcement Learning*
  Marvin Zhang, Sharad Vikram, Laura Smith, Pieter Abbeel, Matthew J. Johnson, Sergey Levine
  https://arxiv.org/abs/1808.09105

Lecture Outline

$$\begin{tikzcd}
\textbf{LQR} \arrow[d] \\
\text{Optimal Control: Collocation and Shooting} \arrow[d] \\
\text{Lyapunov Stability}
\end{tikzcd}$$

Zooming out: Optimal Control

- LQR is a special case of optimal control with analytic solution
- Optimal control more generally solves this problem

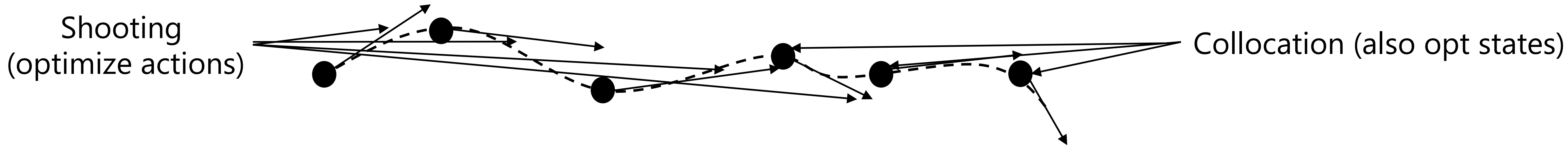
$$\min_{u_{1:T}} \sum_{t=1}^{T} g(x_t, u_t) + G(x_T, u_T)$$

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

- Challenge: often non-convex and challenging to optimize!

Two techniques for optimal control

- Shooting:
  - Optimize for actions such that cost is minimized and future states are implicitly defined by actions
    - LQR is a shooting method with linear dynamics and quadratic cost
- Collocation:
  - Optimize for actions **and** states such that cost is minimized and the dynamics are explicitly satisfied as a constaint

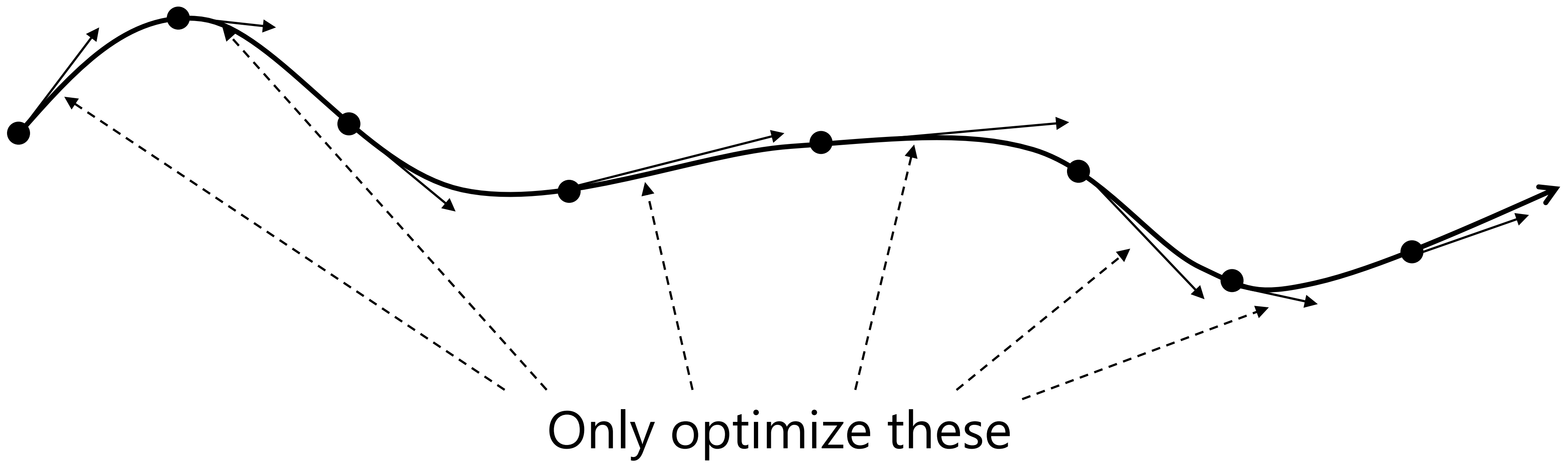


Optimal Control -- Approaches

	Return open-loop controls $u_0, u_1, \ldots, u_H$	Return feedback policy $\pi_\theta(\cdot)$ (e.g. linear or neural net)
shooting	$\min_{u_0,u_1,\ldots,u_H} c(x_0,u_0) + c(f(x_0,u_0),u_1) + c(f(f(x_0,u_0),u_1),u_2) + \ldots$	$\min_\theta c(x_0,\pi_\theta(x_0)) + c(f(x_0,\pi_\theta(x_0)),\pi_\theta(f(x_0,\pi_\theta(x_0)))) + \ldots$
collocation	$\min_{x_0,u_0,x_1,u_1,\ldots,x_H,u_H} \sum_{t=0}^H c(x_t,u_t)$ s.t. $x_{t+1} = f(x_t,u_t) \; \forall t$	$\min_{x_0,x_1,\ldots,x_H,\theta} \sum_{t=0}^H c(x_t,\pi_\theta(x_t))$ s.t. $x_{t+1} = f(x_t,\pi_\theta(x_t)) \; \forall t$ $\min_{x_0,u_0,x_1,u_1,\ldots,x_H,u_H,\theta} \sum_{t=0}^H c(x_t,u_t)$ s.t. $x_{t+1} = f(x_t,u_t) \; \forall t$, $u_t = \pi_\theta(x_t) \; \forall t$

Optimal Control: Shooting

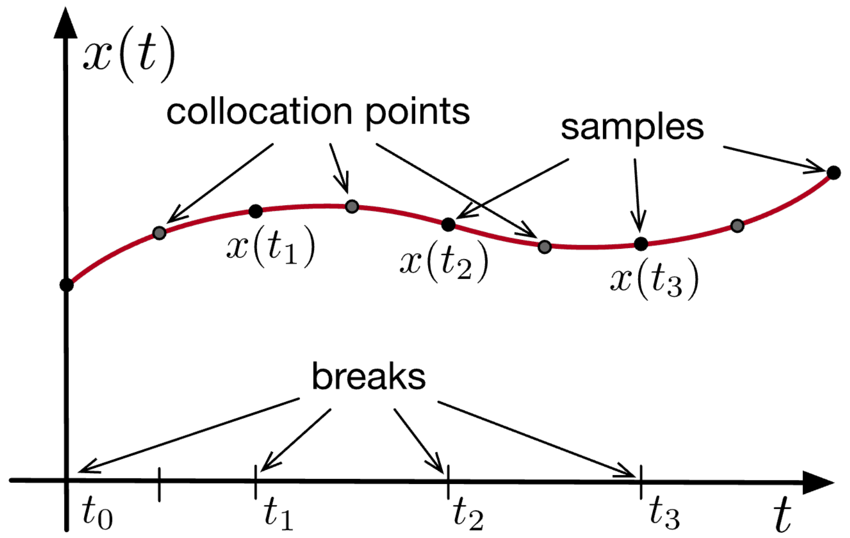
- Only optimize over actions to minimize cost



- Usually easier to specify valid domain, but worse conditioning

Optimal Control: Collocation

- Jointly optimize over both states and actions to minimize cost, subject to constraints



- Better numerical conditioning

(In)stability of Open-loop Shooting

- Let's reconsider:

$$\min_{u_0, u_1, \ldots, u_H} c(x_0, u_0) + c(f(x_0, u_0), u_1) + c(f(f(x_0, u_0), u_1), u_2) + \ldots$$

  - Rolling out $u_0, u_1, \ldots, u_H$ can often be unstable because small numerical errors (or noise) can amplify in case of unstable dynamics
  - In turn, this can make this formulation unstable to optimize

- Solutions:
  - During roll-out, use re-planning (model-predictive control)
  - **<u>Use collocation</u>**

Collocation versus Shooting

- Shooting:
  - Improve sequence of controls over time, at all times u (or pi) are meaningful
  - Often poorly conditioned (effect of early u so much higher than later u)
  - Not clear how to initialize in a way that nudges towards a goal state
- Collocation
  - Might converge to a local optimum that's infeasible, and until converged often not feasible
  - x provides decoupling between time-steps, making computation stable
  - Can initialize with simple linear interpolation or guess of good trajectory
- Iterative LQR?
  - Specific example of a shooting method, with linear controllers, and second order optimization

Let’s look a little closer

Solving Problems with Direct Shooting

Forward Shooting:

$$\min_{\mathbf{u}^0 \dots \mathbf{u}^T} \sum_t C^t(\mathbf{x}^t), \quad \mathbf{x}^{t+1} = f(\mathbf{x}^t, \mathbf{u}^t)$$

Solution 1:
Linearize and perform iLQR/DDQP

Solution 3:
Formulate as a convex optimization problem, use convex solver (gurobi/snopt)

Solution 2:
If differentiable, direct backpropagation through time

Visualization of Direct Shooting

Visualization of Direct Shooting

Forward Shooting:

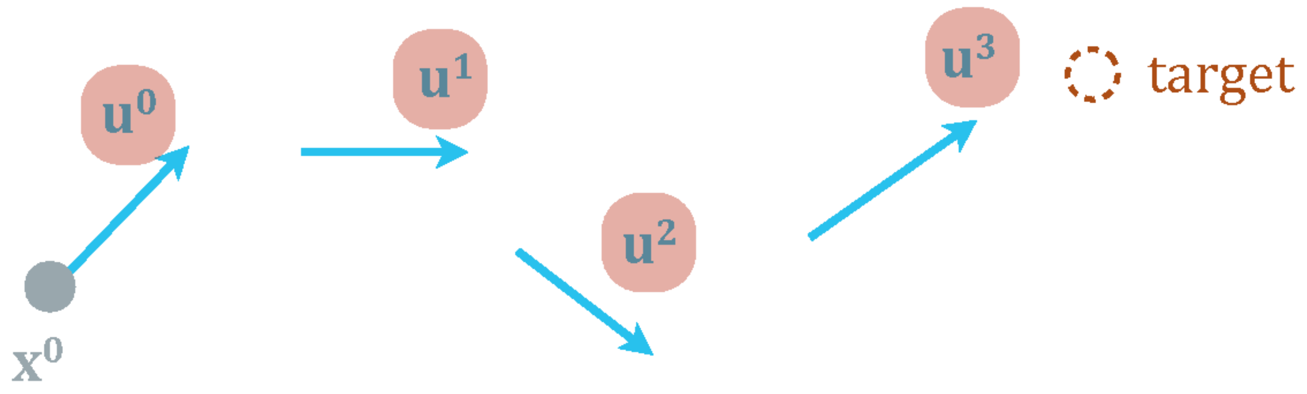
$$\min_{\mathbf{u}^0 \ldots \mathbf{u}^T} \sum_t C^t(\mathbf{x}^t), \quad \mathbf{x}^{t+1} = f(\mathbf{x}^t, \mathbf{u}^t)$$

target
$\mathbf{x}^0$

Visualization of Direct Shooting

Forward Shooting:

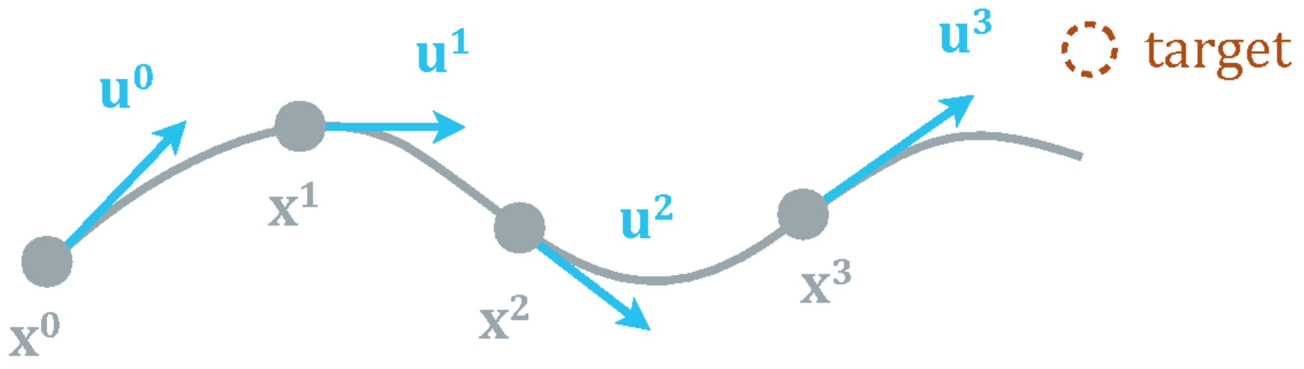
$$\min_{\mathbf{u}^0 \ldots \mathbf{u}^T} \sum_t C^t(\mathbf{x}^t), \quad \mathbf{x}^{t+1} = f(\mathbf{x}^t, \mathbf{u}^t)$$



Visualization of Direct Shooting

Forward Shooting:

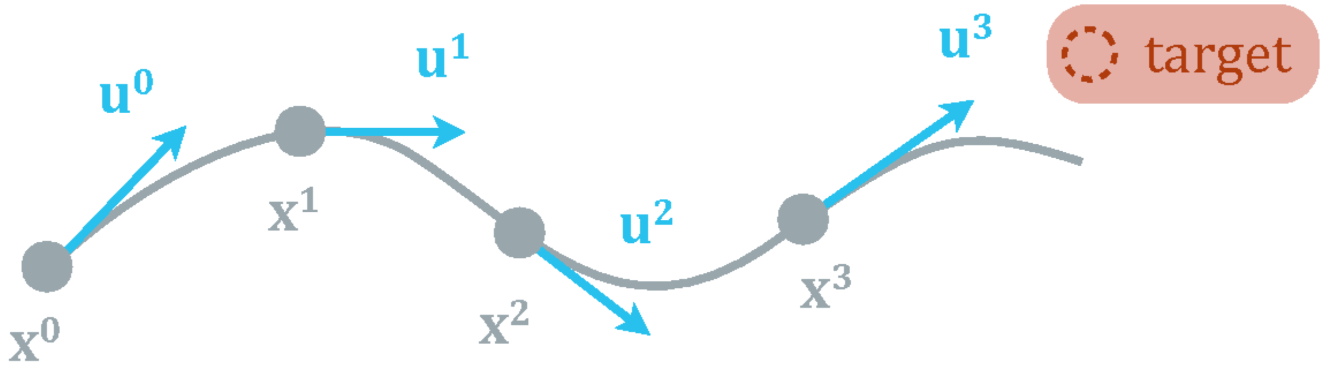
$$\min_{\mathbf{u}^0 \ldots \mathbf{u}^T} \sum_t C^t(\mathbf{x}^t), \qquad \mathbf{x}^{t+1} = f(\mathbf{x}^t, \mathbf{u}^t)$$



Visualization of Direct Shooting

Forward Shooting:

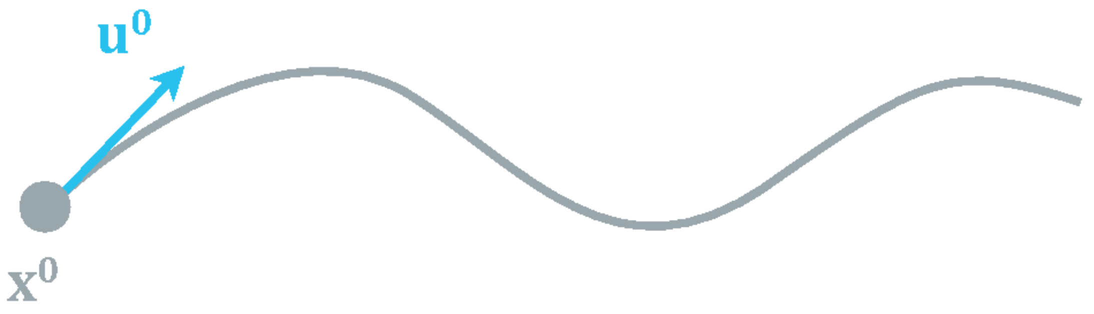
$$\min_{\mathbf{u}^0 \ldots \mathbf{u}^T} \sum_t C^t(\mathbf{x}^t), \qquad \mathbf{x}^{t+1} = f(\mathbf{x}^t, \mathbf{u}^t)$$



Poor Conditioning in Direct Shooting

Forward Shooting:

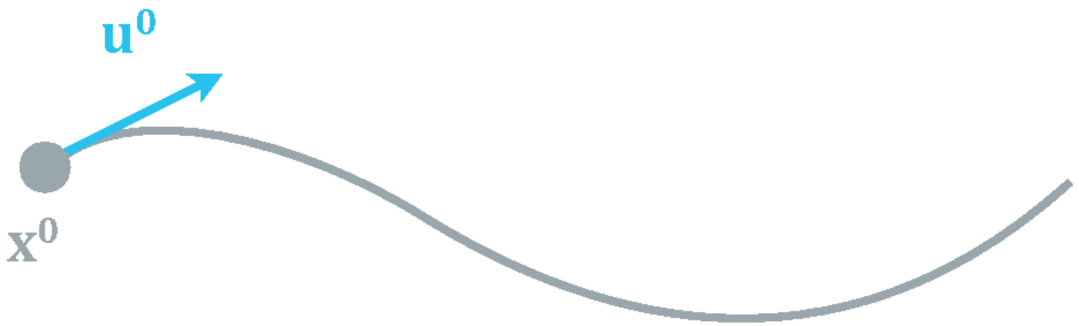
$$\min_{\mathbf{u}^0 \ldots \mathbf{u}^T} \sum_t C^t(\mathbf{x}^t), \quad \mathbf{x}^{t+1} = f(\mathbf{x}^t, \mathbf{u}^t)$$



Poor Conditioning in Direct Shooting

Forward Shooting:

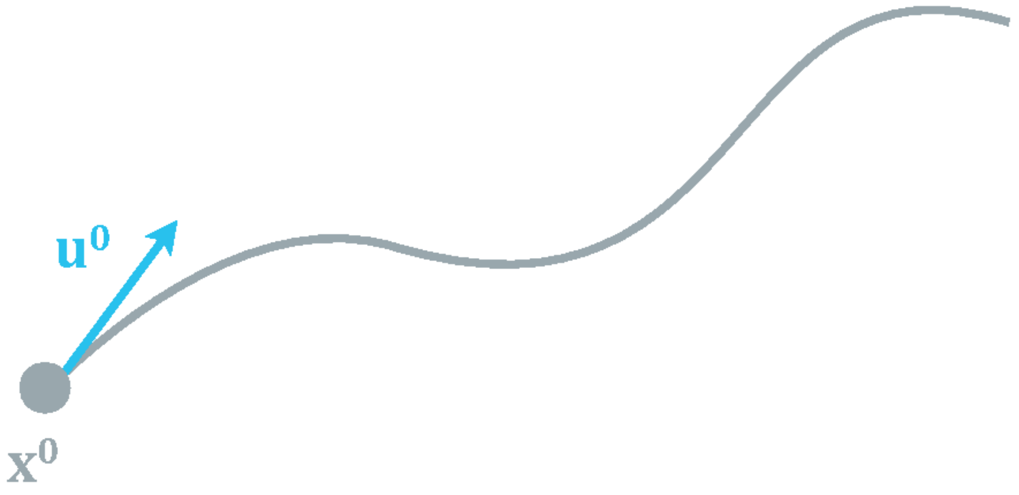
$$\min_{\mathbf{u}^0 \ldots \mathbf{u}^T} \sum_t C^t(\mathbf{x}^t), \quad \mathbf{x}^{t+1} = f(\mathbf{x}^t, \mathbf{u}^t)$$



Poor Conditioning in Direct Shooting

Forward Shooting:

$$\min_{\mathbf{u}^0 \dots \mathbf{u}^T} \sum_t C^t(\mathbf{x}^t), \quad \mathbf{x}^{t+1} = f(\mathbf{x}^t, \mathbf{u}^t)$$



Poor Conditioning in Direct Shooting

Forward Shooting:

$$\min_{\mathbf{u}^0\dots\mathbf{u}^T} \sum_t C^t(\mathbf{x}^t), \quad \mathbf{x}^{t+1} = f(\mathbf{x}^t, \mathbf{u}^t)$$

$$\Updownarrow$$

$$\min_{\mathbf{u}_1,\dots,\mathbf{u}_T} c(\mathbf{x}_1, \mathbf{u}_1) + c(f(\mathbf{x}_1, \mathbf{u}_1), \mathbf{u}_2) + \cdots$$

$$\cdots + c(f(f(\dots)\dots), \mathbf{u}_T)$$

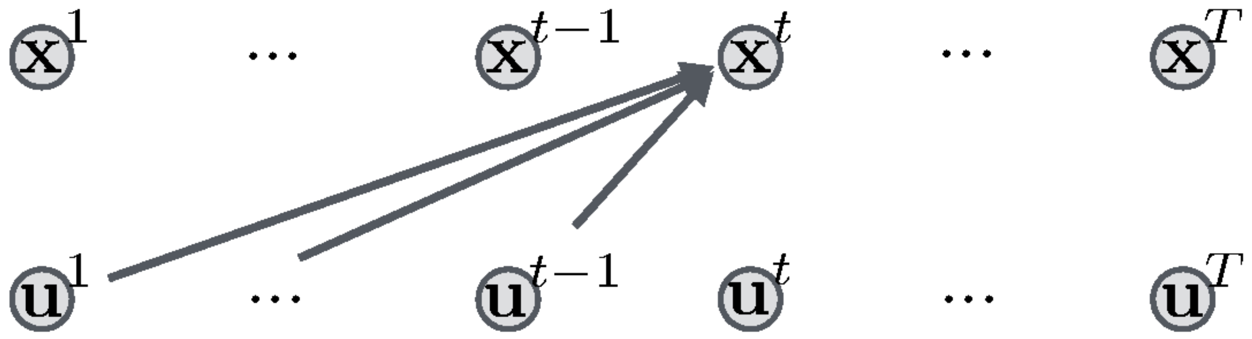
Poor Conditioning in Direct Shooting

Forward Shooting:

$$\min_{\mathbf{u}^0 \dots \mathbf{u}^T} \sum_t C^t(\mathbf{x}^t), \quad \mathbf{x}^{t+1} = f(\mathbf{x}^t, \mathbf{u}^t)$$

$$\Updownarrow$$

$$\min_{\mathbf{u}_1,\dots,\mathbf{u}_T} c(\mathbf{x}_1, \mathbf{u}_1) + c(f(\mathbf{x}_1, \mathbf{u}_1), \mathbf{u}_2) + \cdots$$
$$\cdots + c(f(f(\dots)\dots), \mathbf{u}_T)$$



Narrow Feasible Region

Forward Shooting:

$$\min_{\mathbf{u}^0 \dots \mathbf{u}^T} \sum_t C^t(\mathbf{x}^t), \quad \mathbf{x}^{t+1} = f(\mathbf{x}^t, \mathbf{u}^t)$$

Narrow Feasible Region

Forward Shooting:

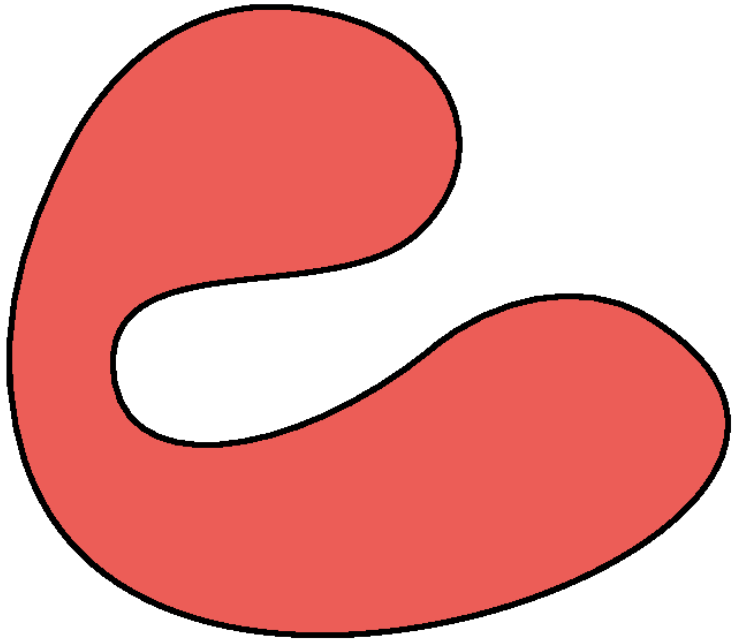
$$\min_{\mathbf{u}^0 \ldots \mathbf{u}^T} \sum_t C^t(\mathbf{x}^t), \qquad \mathbf{x}^{t+1} = f(\mathbf{x}^t, \mathbf{u}^t)$$

implicit hard constraint

Narrow Feasible Region

Forward Shooting:

$$\min_{\mathbf{u}^0 \ldots \mathbf{u}^T} \sum_t C^t(\mathbf{x}^t), \qquad \mathbf{x}^{t+1} = f(\mathbf{x}^t, \mathbf{u}^t)$$

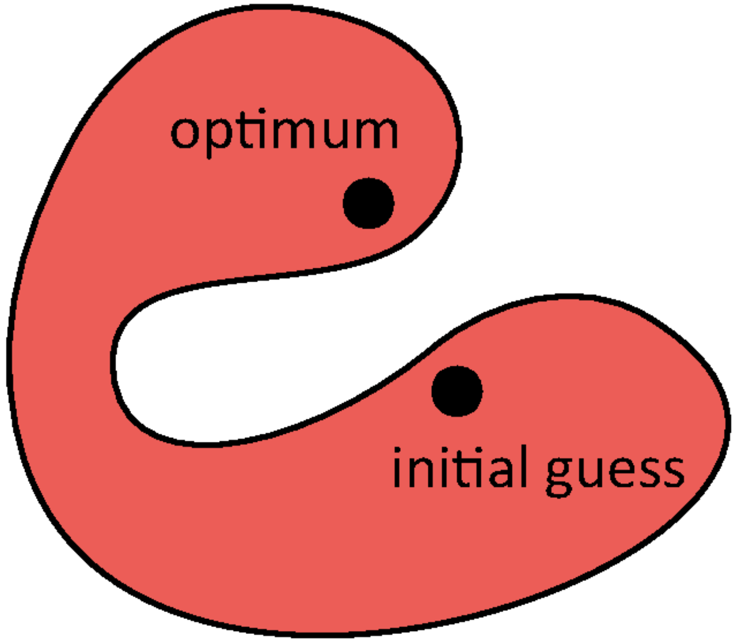


feasible region

Narrow Feasible Region

Forward Shooting:

$$\min_{\mathbf{u}^0 \ldots \mathbf{u}^T} \sum_t C^t(\mathbf{x}^t), \quad \mathbf{x}^{t+1} = f(\mathbf{x}^t, \mathbf{u}^t)$$

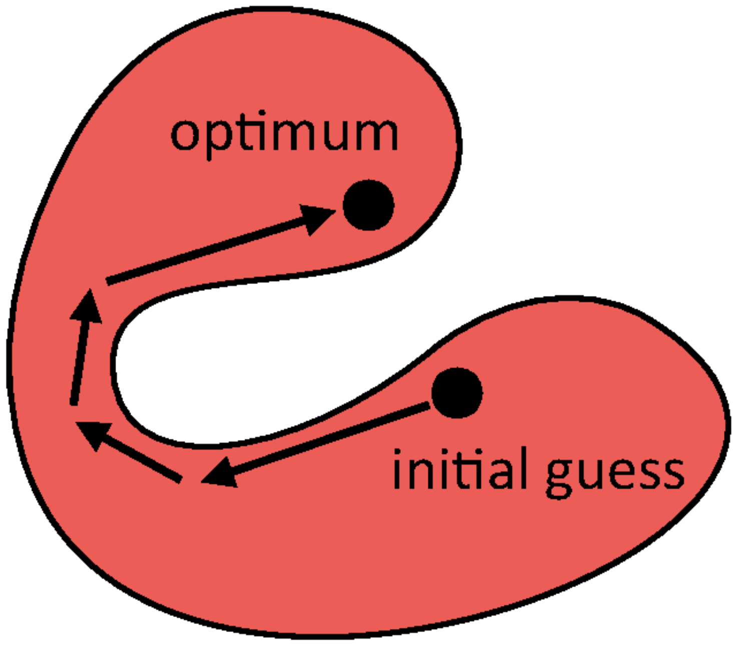


feasible region

Narrow Feasible Region

Forward Shooting:

$$\min_{\mathbf{u}^0\ldots\mathbf{u}^T} \sum_t C^t(\mathbf{x}^t), \quad \mathbf{x}^{t+1} = f(\mathbf{x}^t, \mathbf{u}^t)$$



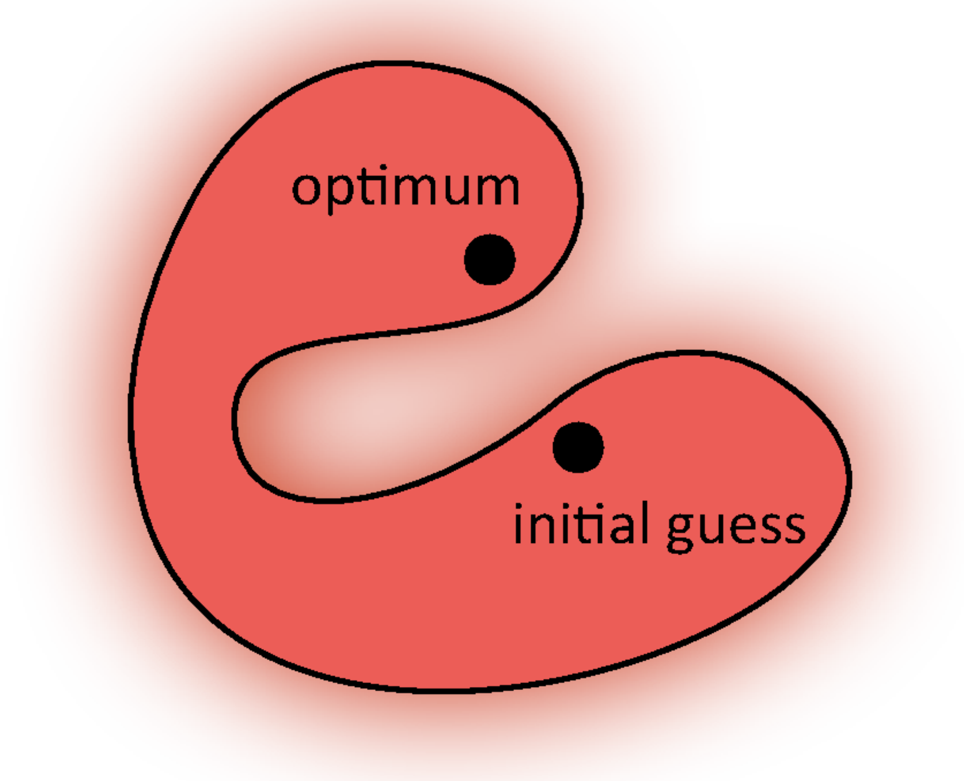
feasible region

Narrow Feasible Region

Forward Shooting:

$$\min_{\mathbf{u}^0 \ldots \mathbf{u}^T} \sum_t C^t(\mathbf{x}^t), \quad \boxed{\mathbf{x}^{t+1} = f(\mathbf{x}^t, \mathbf{u}^t)}$$

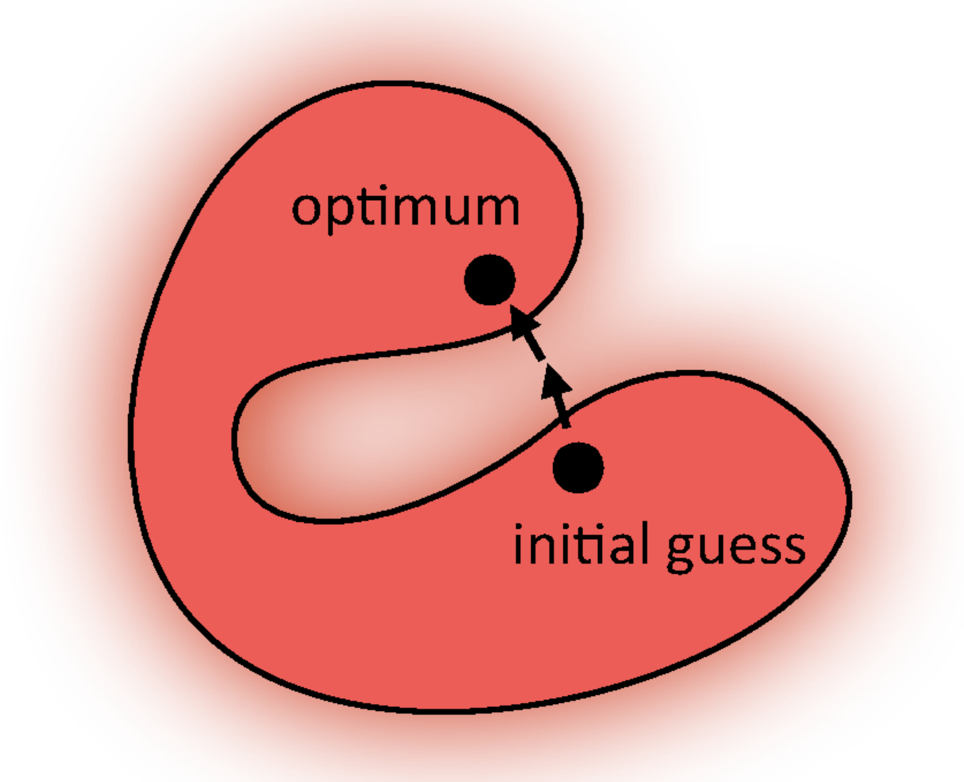
soft constraint



Narrow Feasible Region

Forward Shooting:

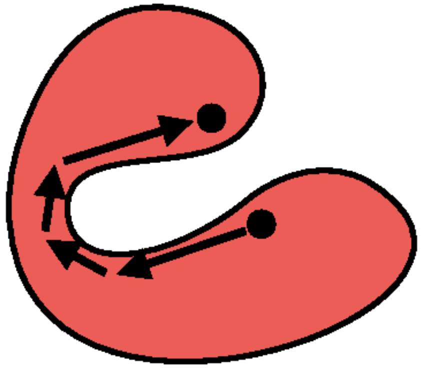
$$\min_{\mathbf{u}^0 \ldots \mathbf{u}^T} \sum_t C^t(\mathbf{x}^t), \quad \mathbf{x}^{t+1} = f(\mathbf{x}^t, \mathbf{u}^t)$$



Narrow Feasible Region

Forward Shooting:

$$\min_{\mathbf{u}^0 \ldots \mathbf{u}^T} \sum_t C^t(\mathbf{x}^t), \quad \mathbf{x}^{t+1} = f(\mathbf{x}^t, \mathbf{u}^t)$$



- Comes up as an issue in practice
  - collisions, falling down, etc…
- Prone to falling into local minima
- Makes solution sensitive to initial guess
- Initial guess from demonstrations and randomization helps

Comparison between Shooting and Collocation

shooting	$\min_{u_0,u_1,\ldots,u_H} c(x_0,u_0) + c(f(x_0,u_0),u_1) + c(f(f(x_0,u_0),u_1),u_2) + \ldots$
collocation	$\min_{x_0,u_0,x_1,u_1,\ldots,x_H,u_H} \sum_{t=0}^{H} c(x_t,u_t)$ s.t. $x_{t+1} = f(x_t,u_t) \;\; \forall t$

Comparison between Shooting and Collocation

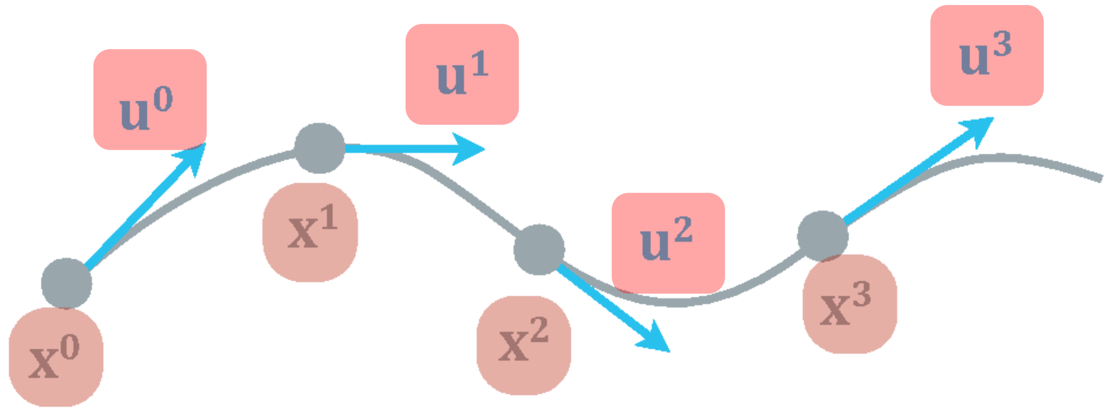
shooting	$\min_{u_0,u_1,\ldots,u_H} c(x_0,u_0) + c(f(x_0,u_0),u_1) + c(f(f(x_0,u_0),u_1),u_2) + \ldots$
collocation	$\min_{x_0,u_0,x_1,u_1,\ldots,x_H,u_H} \sum_{t=0}^{H} c(x_t,u_t)$ s.t. $x_{t+1} = f(x_t,u_t) \;\; \forall t$

Let’s think about collocation

shooting	$\min_{u_0,u_1,\ldots,u_H} c(x_0,u_0) + c(f(x_0,u_0),u_1) + c(f(f(x_0,u_0),u_1),u_2) + \ldots$
collocation	$\min_{x_0,u_0,x_1,u_1,\ldots,x_H,u_H} \sum_{t=0}^{H} c(x_t,u_t)$ s.t. $x_{t+1} = f(x_t,u_t) \ \forall t$

Solving Collocation Problems

$$\min_{x_0,u_0,x_1,u_1,\ldots,x_H,u_H} \quad \sum_{t=0}^{H} c(x_t,u_t)$$
$$\text{s.t.} \quad x_{t+1} = f(x_t,u_t) \quad \forall t$$



Idea 1: Direct constrained optimization through forward dynamics

Idea 2: constrained optimization through inverse dynamics

Collocation with Inverse Dynamics

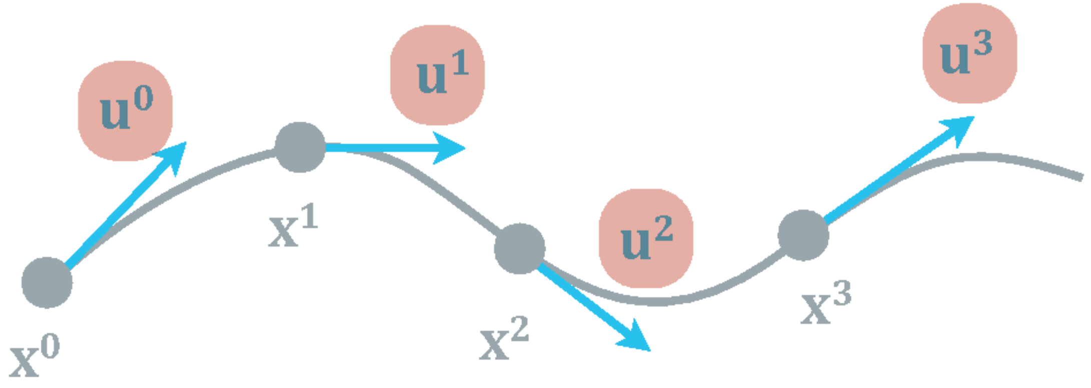
Forward Shooting:

$$\min_{\mathbf{u}^0\ldots\mathbf{u}^T} \sum_t C^t(\mathbf{x}^t), \quad \mathbf{x}^{t+1} = f(\mathbf{x}^t, \mathbf{u}^t)$$

inverse dynamics function

Direct Collocation:

$$\min_{\mathbf{x}^0\ldots\mathbf{x}^T} \sum_t C^t(\mathbf{x}^t), \;\; st \; \overset{\downarrow}{f^{-1}(\mathbf{x}^t, \mathbf{x}^{t+1}) = \mathbf{u}^t} \in \mathcal{U}$$



Collocation with Inverse Dynamics

Forward Shooting:

$$\min_{\mathbf{u}^0 \dots \mathbf{u}^T} \sum_t C^t(\mathbf{x}^t), \quad \mathbf{x}^{t+1} = f(\mathbf{x}^t, \mathbf{u}^t)$$

Direct Collocation:

$$\min_{\mathbf{x}^0 \dots \mathbf{x}^T} \sum_t C^t(\mathbf{x}^t), \;\; st \;\; f^{-1}(\mathbf{x}^t, \mathbf{x}^{t+1}) = \mathbf{u}^t \in \mathcal{U}$$

$$\begin{tikzcd}
\mathbf{x}^1 & \dots & \mathbf{x}^{t-1} \arrow[dl] \arrow[d] & \mathbf{x}^t \arrow[dl] \arrow[d] & \mathbf{x}^{t+1} \arrow[dl] \arrow[d] & \dots \arrow[dl] & \mathbf{x}^T \\
\mathbf{u}^1 & \dots & \mathbf{u}^{t-1} & \mathbf{u}^t & \mathbf{u}^{t+1} & \dots & \mathbf{u}^T
\end{tikzcd}$$

Advantages of Direct Collocation

Direct Collocation:

$$\min_{\mathbf{x}^0 \ldots \mathbf{x}^T} \sum_t C^t(\mathbf{x}^t), \;\; st \;\; f^{-1}(\mathbf{x}^t, \mathbf{x}^{t+1}) = \mathbf{u}^t \in \mathcal{U}$$

$$\begin{tikzcd}
\mathbf{x}^1 & \cdots & \mathbf{x}^{t-1} \arrow[dl] \arrow[d] & \mathbf{x}^t \arrow[dl] \arrow[d] & \mathbf{x}^{t+1} \arrow[dl] \arrow[d] & \cdots \arrow[dl] & \mathbf{x}^T \\
\mathbf{u}^1 & \cdots & \mathbf{u}^{t-1} & \mathbf{u}^t & \mathbf{u}^{t+1} & \cdots & \mathbf{u}^T
\end{tikzcd}$$

- Only pairwise dependencies
- Good conditioning
  - changing x^1 has similar effect as changing x^T
- No forward integration instability

Feasibility under Direct Collocation

Direct Collocation:

$$\min_{\mathbf{x}^0 \ldots \mathbf{x}^T} \sum_t C^t(\mathbf{x}^t), \; st \;\; f^{-1}(\mathbf{x}^t, \mathbf{x}^{t+1}) = \mathbf{u}^t \in \mathcal{U}$$

Explicit rather than implicit constraint

Feasibility under Direct Collocation

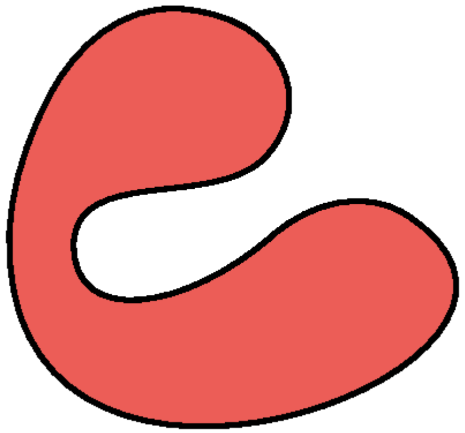
Direct Collocation:

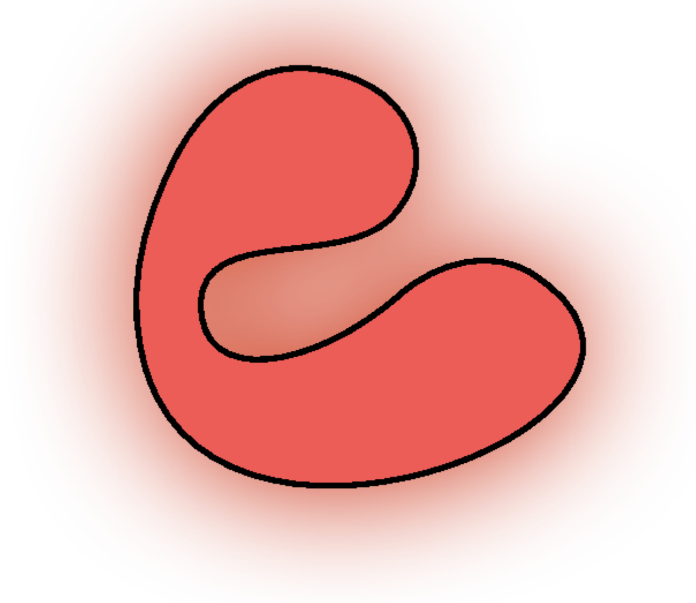
$$\min_{\mathbf{x}^0 \ldots \mathbf{x}^T} \sum_t C^t(\mathbf{x}^t), \quad st \ f^{-1}(\mathbf{x}^t, \mathbf{x}^{t+1}) = \mathbf{u}^t \in \mathcal{U}$$

Explicit rather than implicit constraint

Can be hard or soft

Less prone to local minima





Shooting vs Collocation

Forward Shooting:

$$\min_{\mathbf{u}^0 \ldots \mathbf{u}^T} \sum_t C^t(\mathbf{x}^t), \quad \mathbf{x}^{t+1} = f(\mathbf{x}^t, \mathbf{u}^t)$$

- Optimize over controls
- State trajectory is implicit
- Dynamics is an implicit constraint (always satisfied)

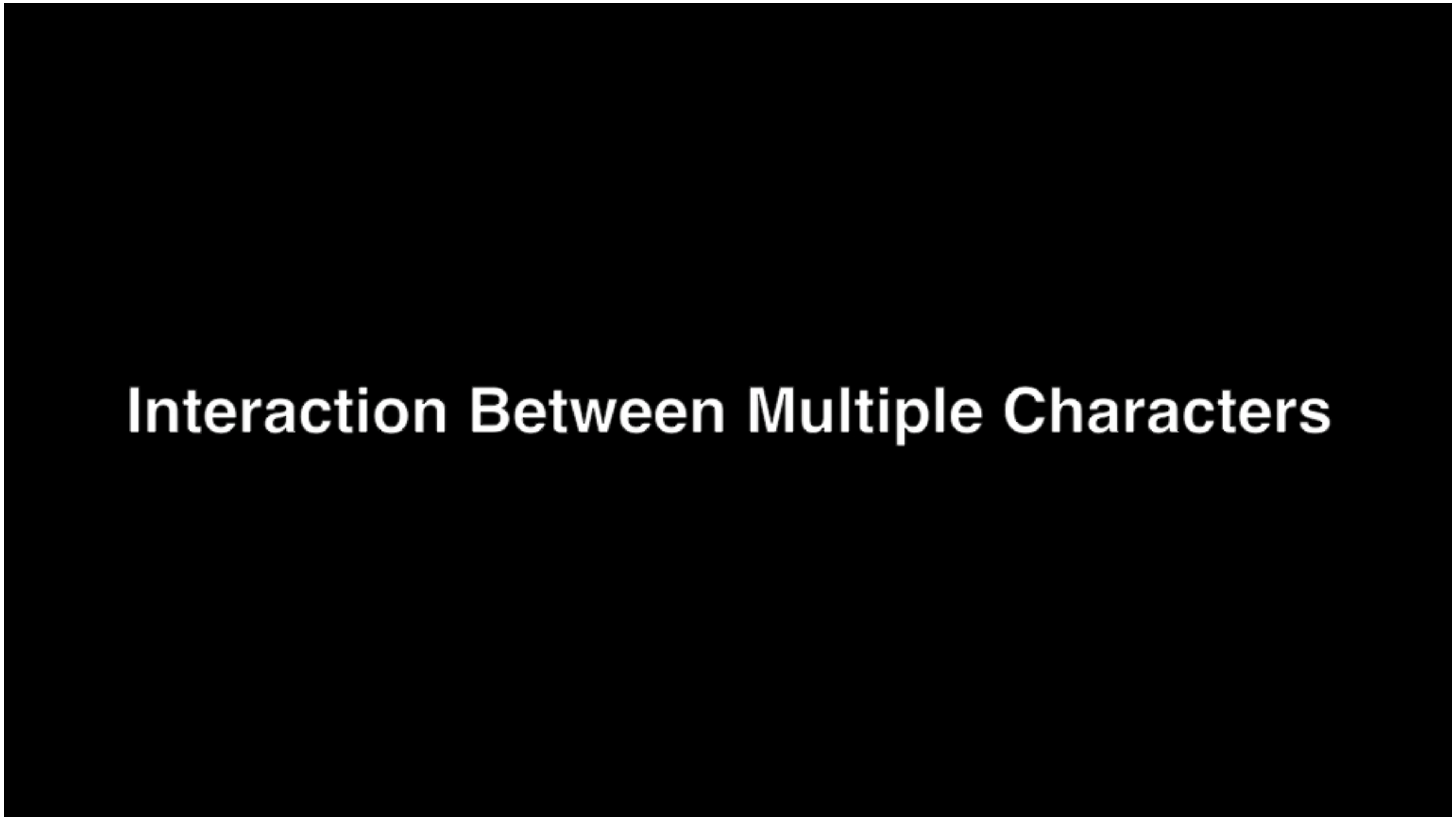
Direct Collocation:

$$\min_{\mathbf{x}^0 \ldots \mathbf{x}^T} \sum_t C^t(\mathbf{x}^t), \;\; st \;\; f^{-1}(\mathbf{x}^t, \mathbf{x}^{t+1}) = \mathbf{u}^t \in \mathcal{U}$$

- Optimize over states
- Controls and forces are implicit
- Dynamics is an explicit constraint (can be soft)

Optimal control methods in action

Optimal control methods in action



Interaction Between Multiple Characters

Optimal control methods in action

In-Hand Object Manipulation

Lecture Outline

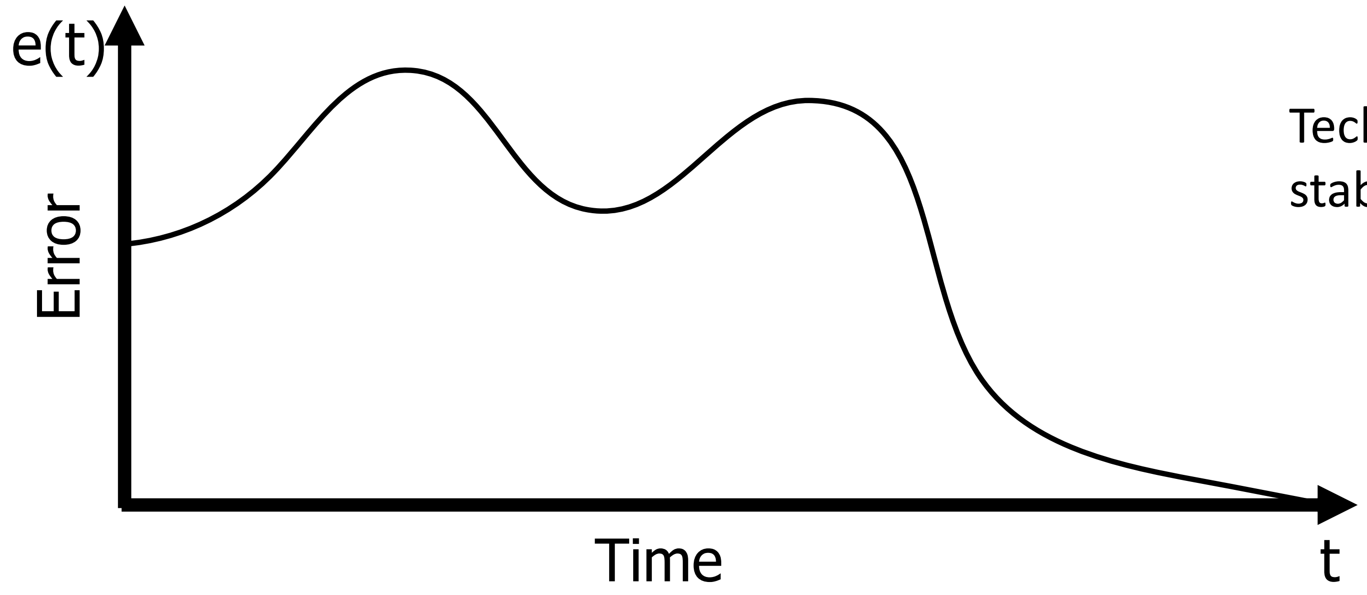
$$\begin{tikzcd}
\textbf{LQR} \arrow[d] \\
\textbf{Optimal Control: Collocation and Shooting} \arrow[d] \\
\text{Lyapunov Stability}
\end{tikzcd}$$

How can we prove that a controller is stable?

Lyapunov Stability

What is stability?

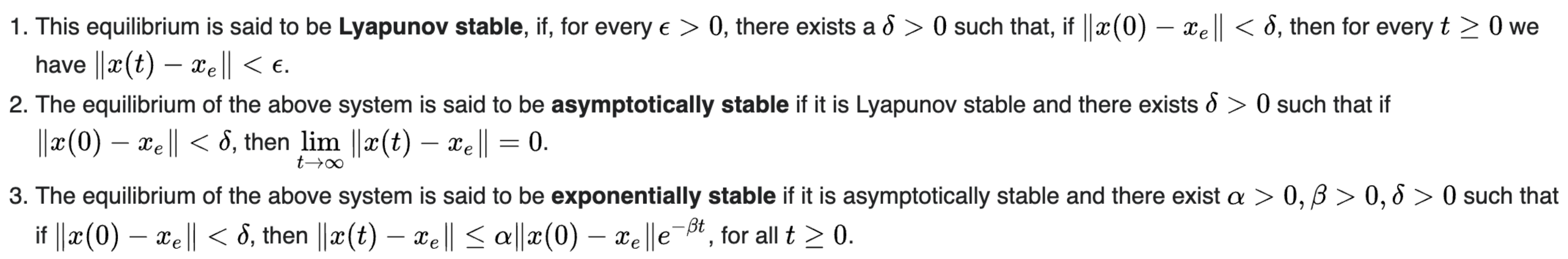
$$\lim_{t \to \infty} e(t) = 0$$



Technically: this is global asymptotic stability, there are other notions of stability

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

More technically precise definitions



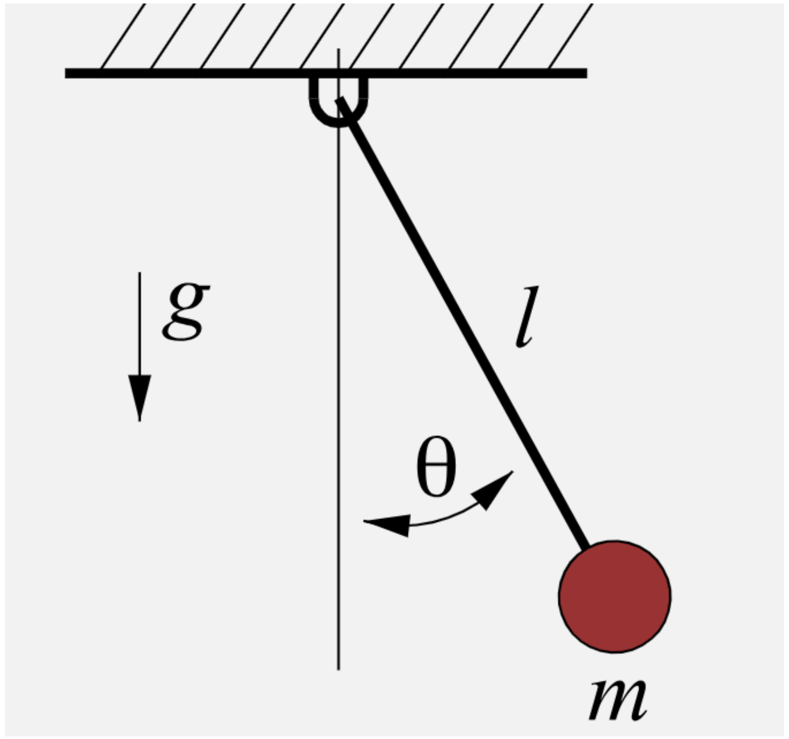
1. This equilibrium is said to be **Lyapunov stable**, if, for every $\epsilon > 0$, there exists a $\delta > 0$ such that, if $\|x(0) - x_e\| < \delta$, then for every $t \geq 0$ we have $\|x(t) - x_e\| < \epsilon$.
2. The equilibrium of the above system is said to be **asymptotically stable** if it is Lyapunov stable and there exists $\delta > 0$ such that if $\|x(0) - x_e\| < \delta$, then $\lim_{t \to \infty} \|x(t) - x_e\| = 0$.
3. The equilibrium of the above system is said to be **exponentially stable** if it is asymptotically stable and there exist $\alpha > 0, \beta > 0, \delta > 0$ such that if $\|x(0) - x_e\| < \delta$, then $\|x(t) - x_e\| \leq \alpha \|x(0) - x_e\| e^{-\beta t}$, for all $t \geq 0$.

Stable i.s.l: doesn't go unbounded

Asymptotically stable → converges

Exponentially stable → converges fast

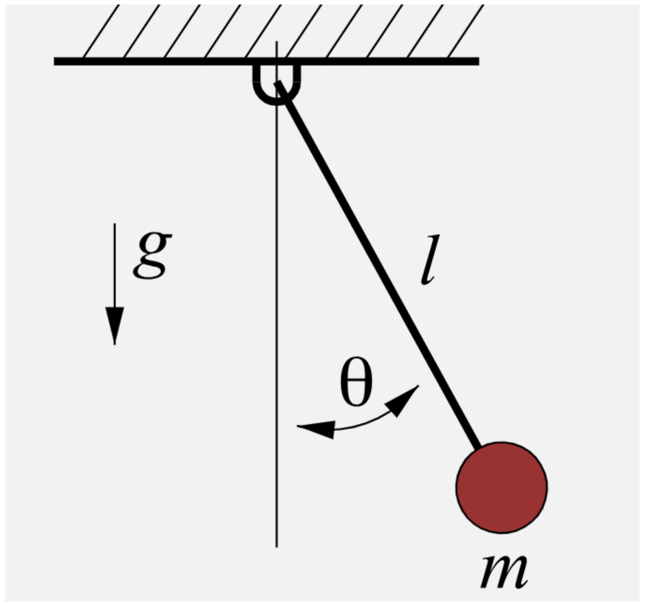
Let’s start with a simple system



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

Can derive via method of Lagrange or newtons laws of motion

Is a pendulum stable?



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

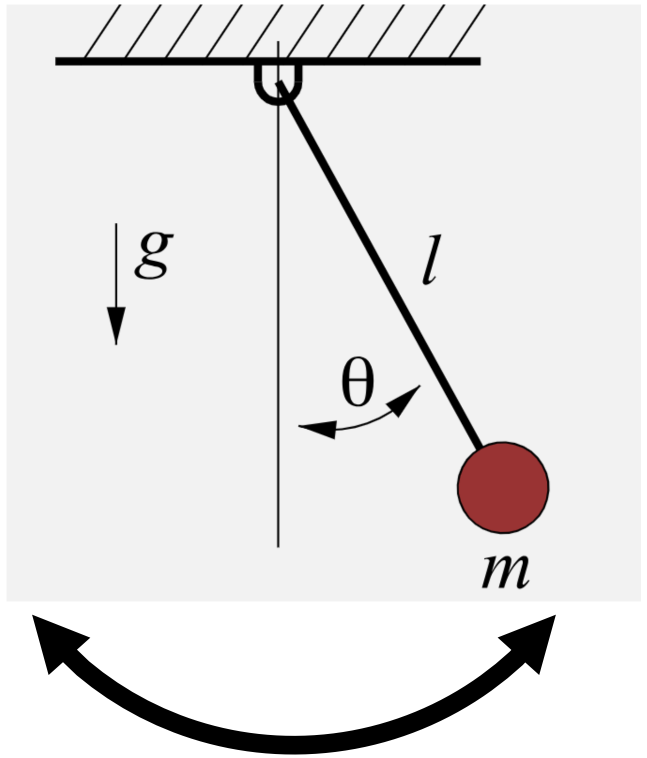
What control law should we use to stabilize the pendulum, i.e.

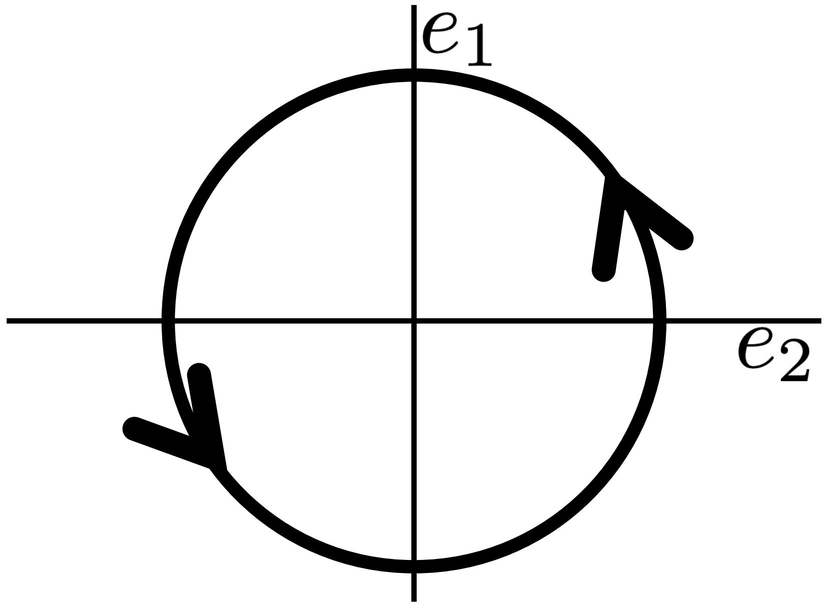
Choose $u = \pi(\theta, \dot{\theta})$ such that $\theta \to 0$

$$\dot{\theta} \to 0$$

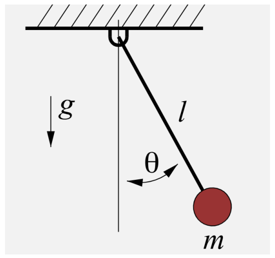
How does the passive error dynamics behave?

Set u=0. Dynamics is not stable, pendulum keeps oscillatin





How do we verify if a controller is stable?



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

Lets pick the following law:

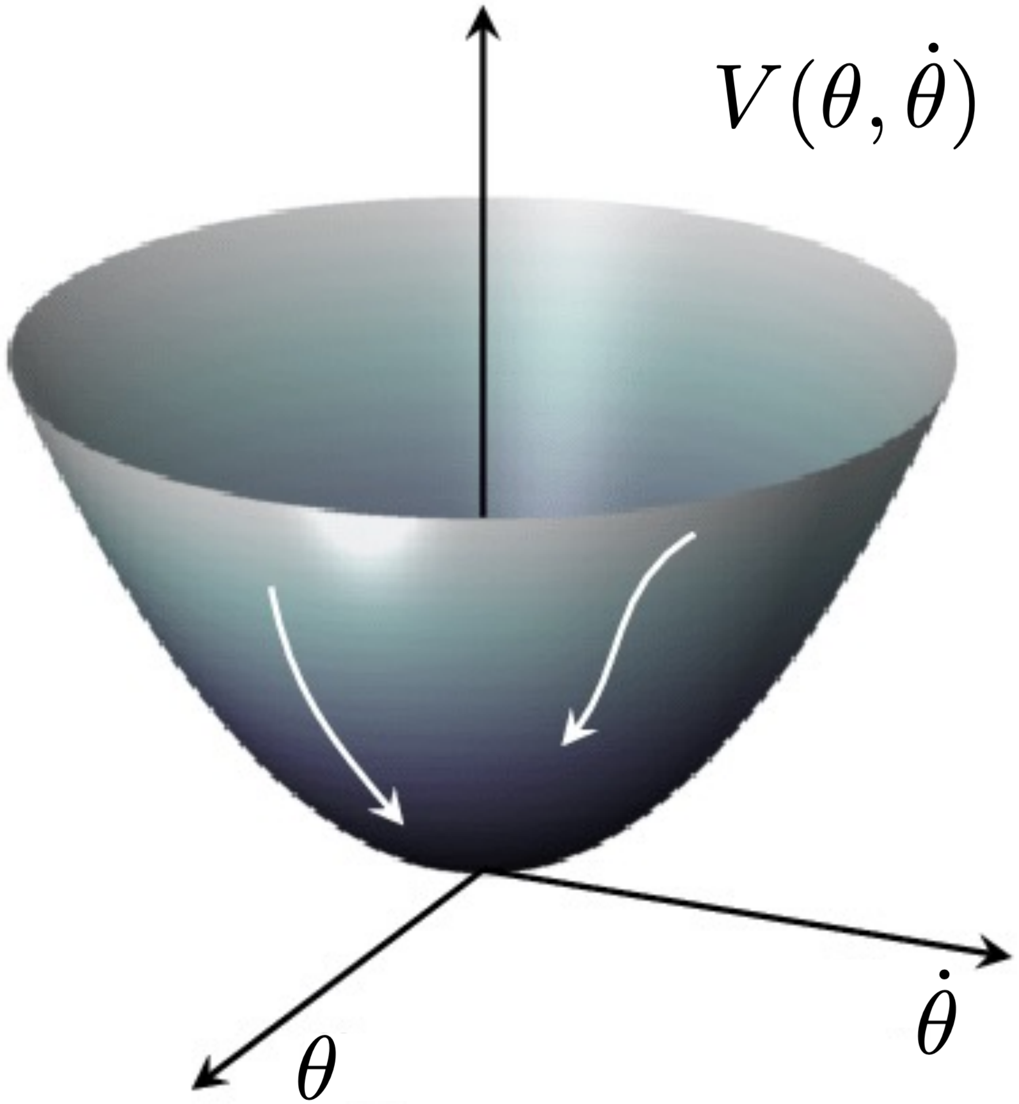
$$u = -K\dot{\theta}$$

Is this stable? How do we know?

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

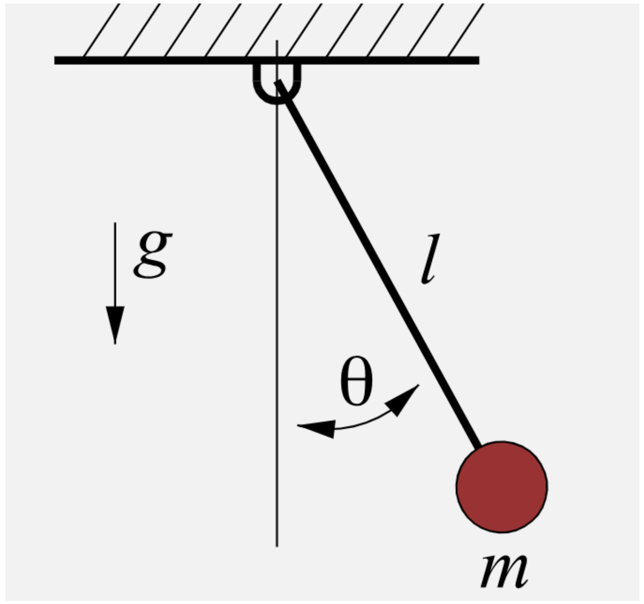
but how many points do we check? what if we miss some points?

Key Idea: Think about energy!



Make energy decay to 0 and stay there

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



$$\begin{aligned}\dot{V}(\theta, \dot{\theta}) &= m l^2 \dot{\theta}\ddot{\theta} + mgl(\sin\theta)\dot{\theta} \\ &= \dot{\theta}(u - mgl\sin\theta) + mgl(\sin\theta)\dot{\theta} \\ &= \dot{\theta}u\end{aligned}$$

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. $V(0) = 0$

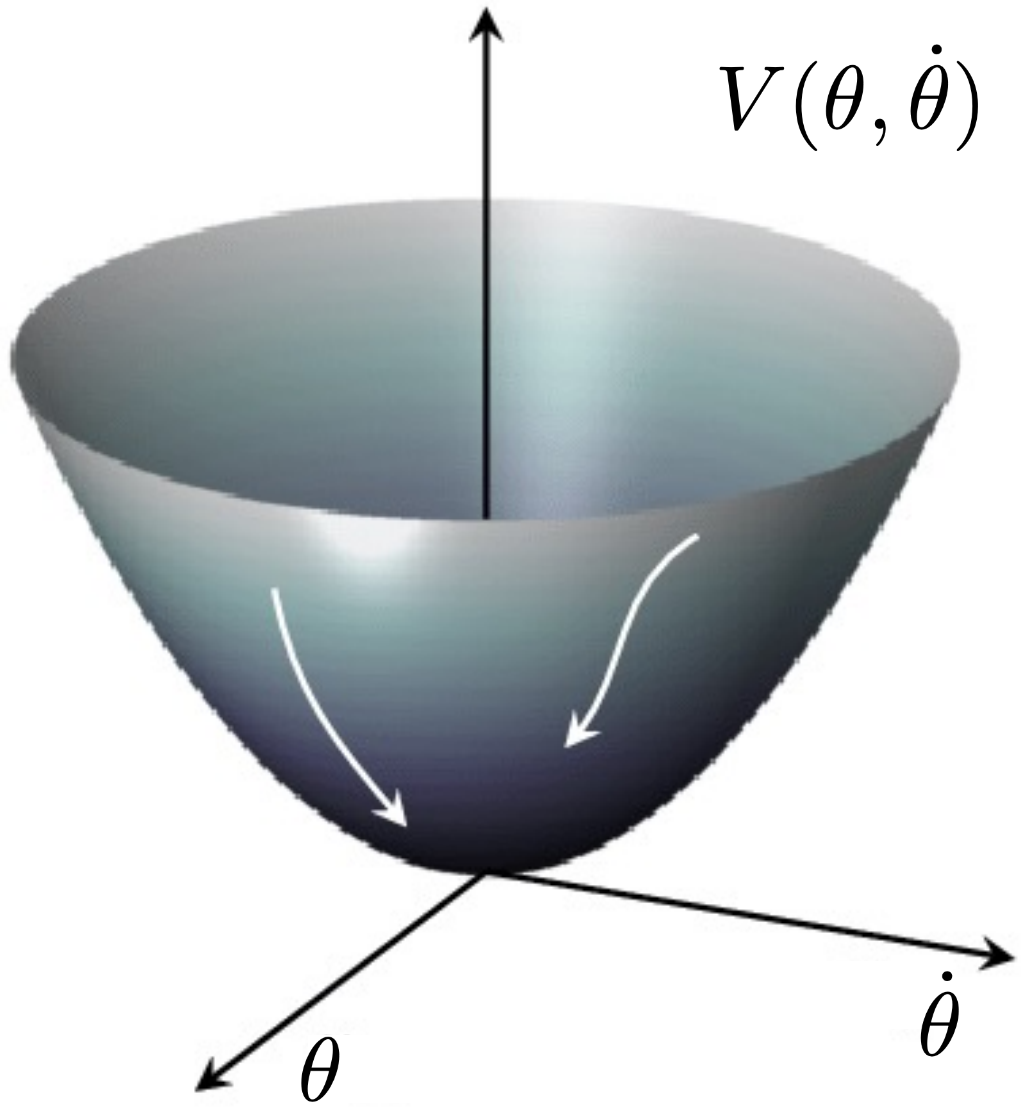
2. Choose a control law such that this energy always decreases

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

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!

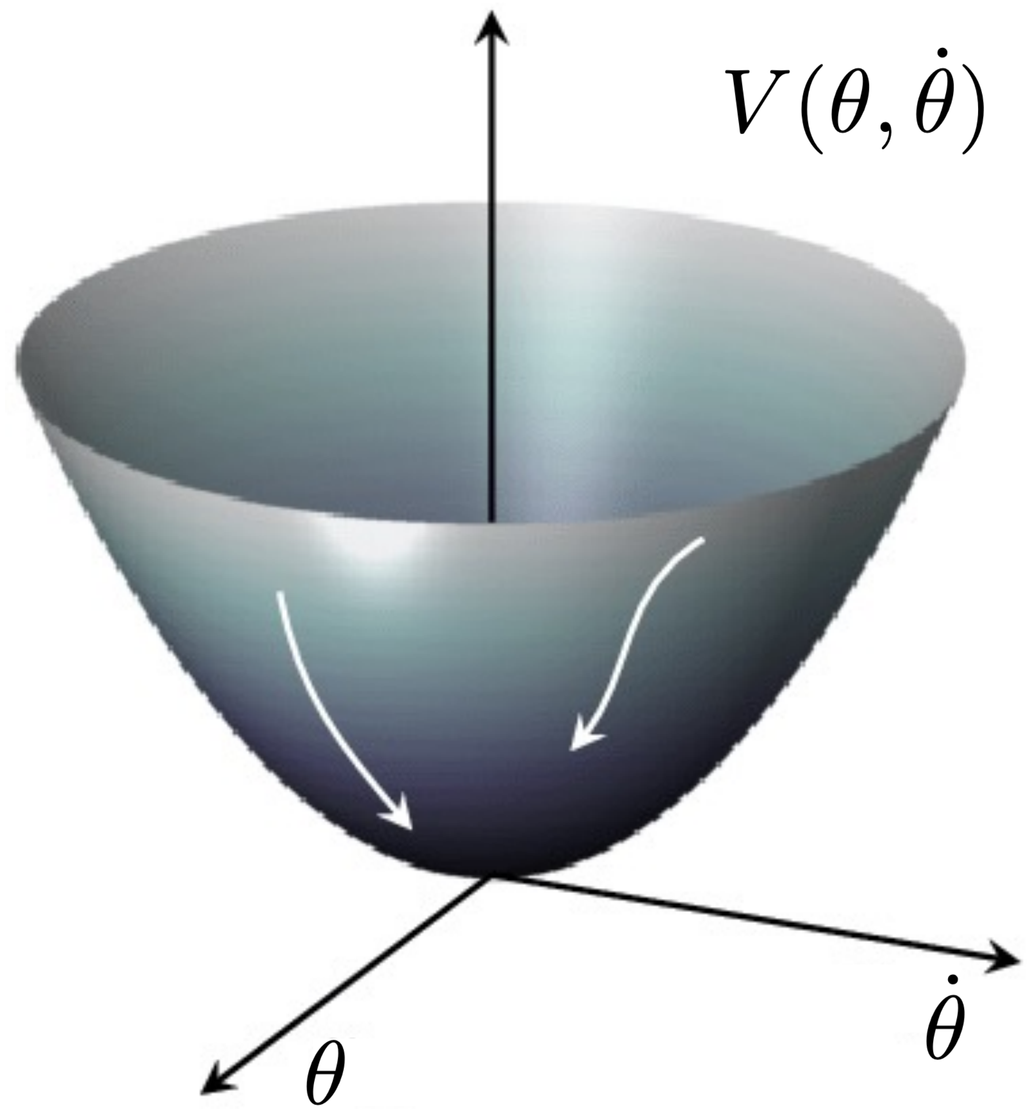
Intuition for Lyapunov Functions



1. V can only be 0 at 0, and is always positive
2. dV/dt is negative, meaning V is going towards 0

   → since it is positive and can only be 0 at 0, eventually this will converge to 0

Why are Lyapunov functions useful?



Provides a way to guarantee stability of a system/controller without exhaustive forward simulation

Finding Lyapunov Functions

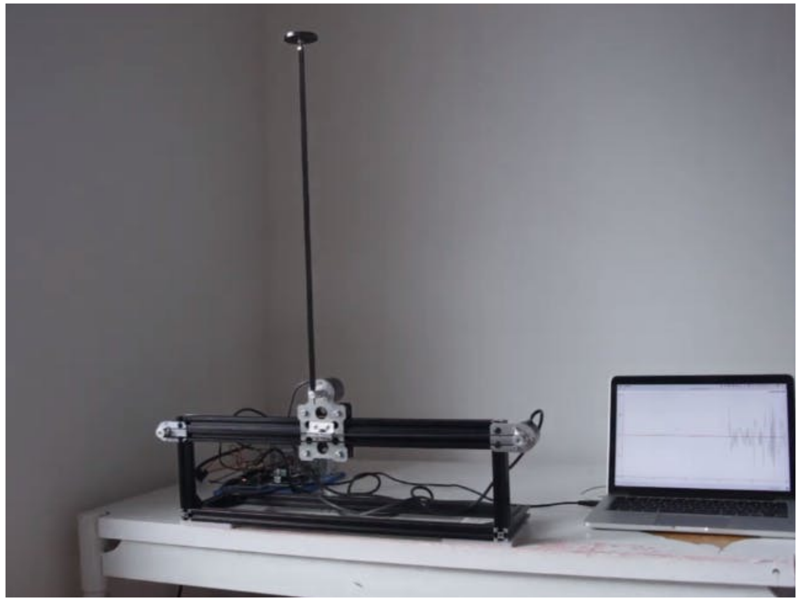
Strategy 1: Use domain knowledge and hand-design

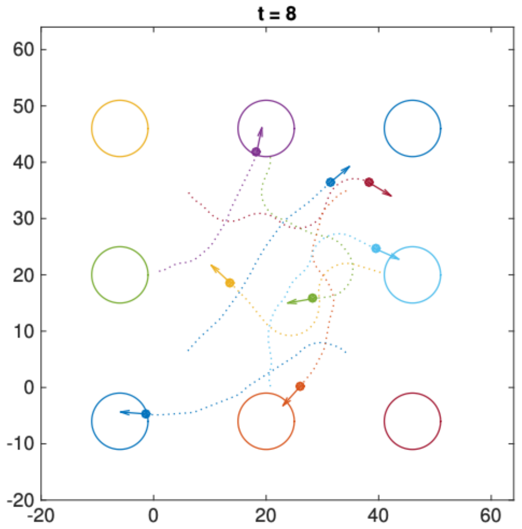
Strategy 2: For linear systems, can solve LMI to obtain a function

Strategy 3: For non-linear functions, can use ideas like sum of squares programming to find V

Strategy 4: Do bilevel optimization to find V or learn V

Uses of Lyapunov Functions

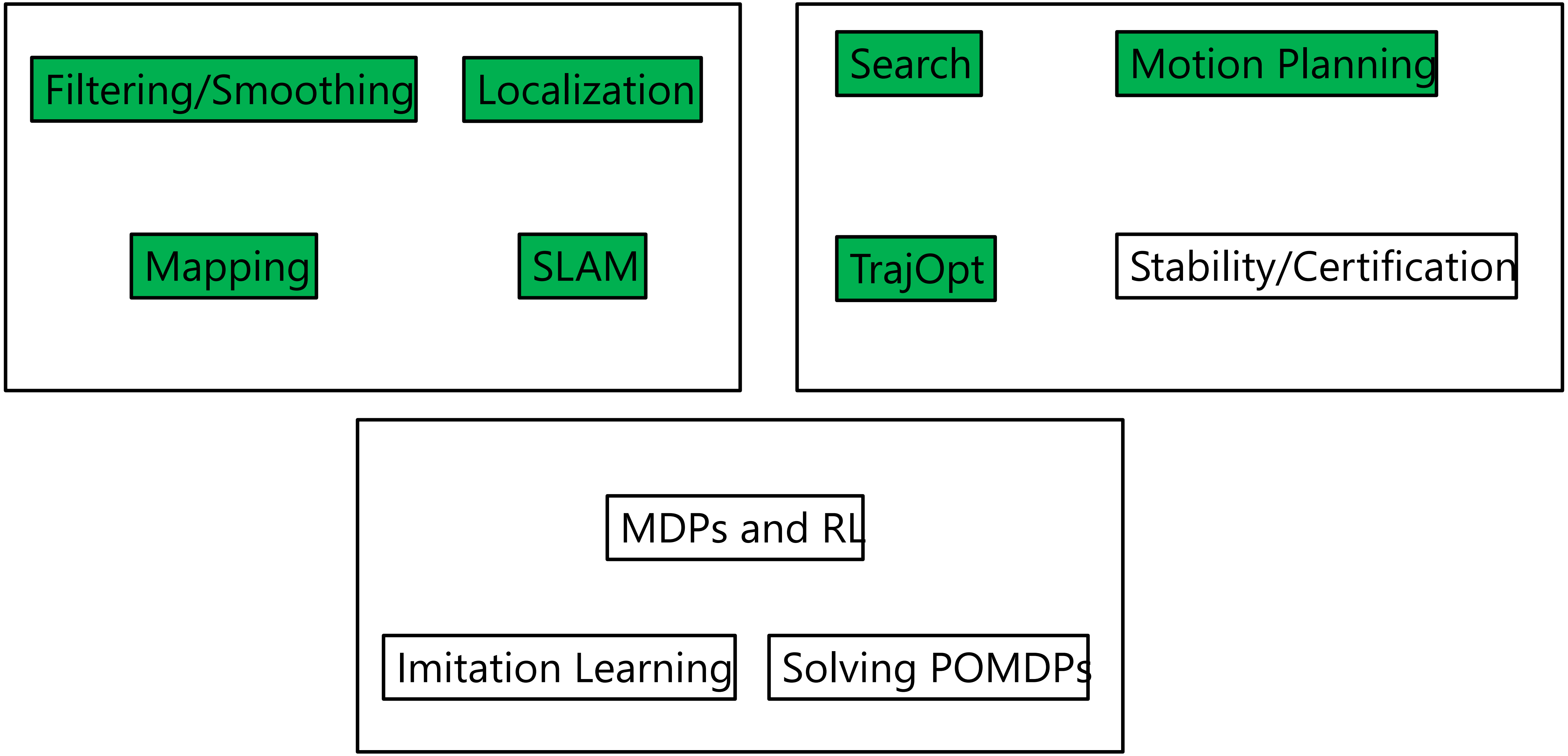




Lecture Outline

$$\begin{tikzcd}
\textbf{LQR} \arrow[d] \\
\textbf{Optimal Control: Collocation and Shooting} \arrow[d] \\
\textbf{Lyapunov Stability}
\end{tikzcd}$$

Recap: Course Overview


Filtering/Smoothing
Localization
Mapping
SLAM
Search
Motion Planning
TrajOpt
Stability/Certification
MDPs and RL
Imitation Learning
Solving POMDPs