Robot Perception and Learning

Introduction of Optimal Control

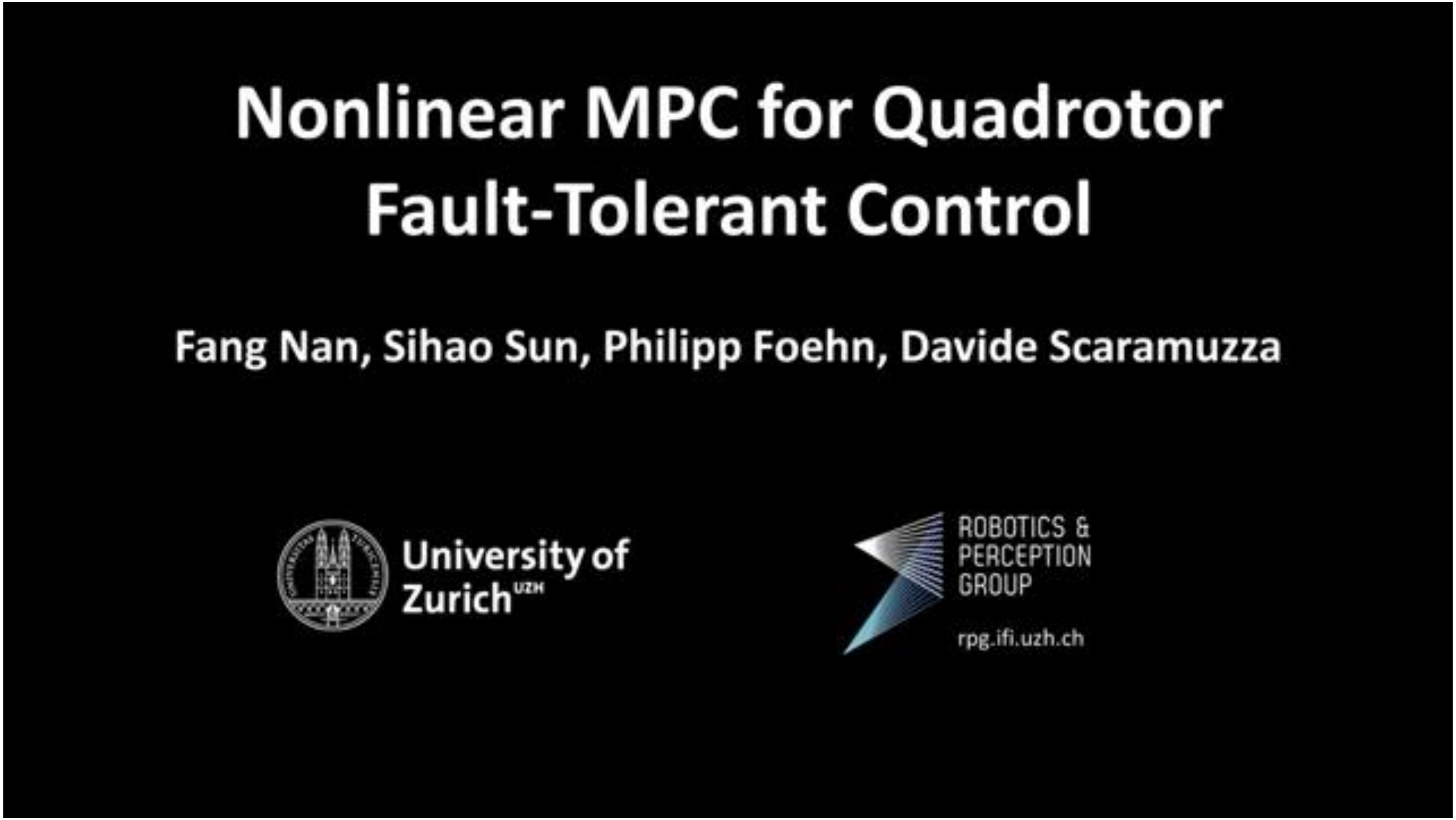
Tsung-Wei Ke

Fall 2025

Disclaimer

- This lecture heavily borrows materials from Zachary Manchester's 16-745 Optimal Control at CMU (https://optimalcontrol.ri.cmu.edu/)
- Optimization involves rich theories. "Convex Optimization" by Stephen Boyd and Lieven Vandenberghe is a good start.

Videos by P. Reist and R. Tedrake



https://youtu.be/Cn_836XGEnU?si=HO3bPw3PNWfnZA_U

Optimal Control: Optimization of Dynamic Systems

- Feedback control: measuring and minimizing tracking error at each time step
- Alternative: defining the control problem in a more general formulation.

- Define:
  1. Cost function $c(x_k, u_k)$, measuring the cost of action u_k at state x_k at time step k
     - The tracking error
  2. Terminal cost function $c_F(x_N)$, measuring the cost at state x_N at the last time step N
     - The distance to the target location
  3. Constraints $\mathbf{h}(x_k, u_k) \leq 0$ and $\mathbf{r}(x_k) \leq 0$ for action u_k and state x_k at time step k
     - Velocity / acceleration / … constraints

Optimal Control: Optimization of Dynamic Systems

- Deterministic Optimal Control Problem:

$$\min_{x,u} \sum_{k=1}^{N-1} c(x_k, u_k) + c_F(x_N)$$

such that

$$x_1 = \hat{x}_1 \qquad \text{initial state}$$

$$x_{k+1} = f(x_k, u_k), \quad k = 1, \dots, N-1 \qquad \text{dynamics}$$

$$0 \geq h(x_k, u_k), \quad k = 1, \dots, N-1 \qquad \text{constraints}$$

$$0 \geq r(x_N) \qquad \text{constraints}$$

Optimal Control: Optimization of Dynamic Systems

- Deterministic Optimal Control Problem:

$$\min_{x,u} \sum_{k=1}^{N-1} c(x_k, u_k) + c_F(x_N)$$

such that

$$x_1 = \hat{x}_1 \qquad \text{initial state}$$

$$x_{k+1} = f(x_k, u_k), \quad k = 1, \dots, N-1 \qquad \text{dynamics}$$

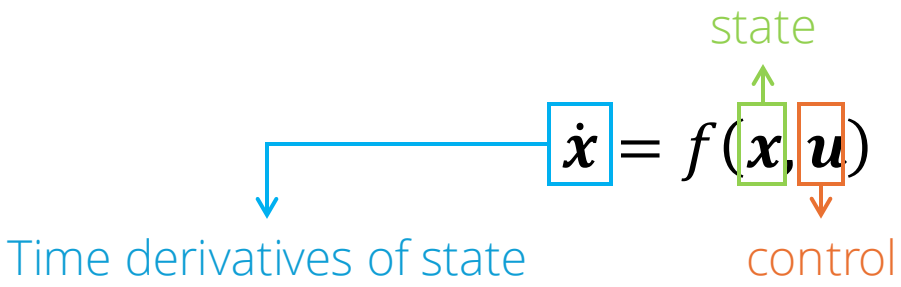
$$0 \geq h(x_k, u_k), \quad k = 1, \dots, N-1 \qquad \text{constraints}$$

$$0 \geq r(x_N) \qquad \text{constraints}$$

If the dynamics is given, can we plan a trajectory of control $u_{1:N}$ by solving the optimization problem?

Dynamics

- Continuous-time dynamics:

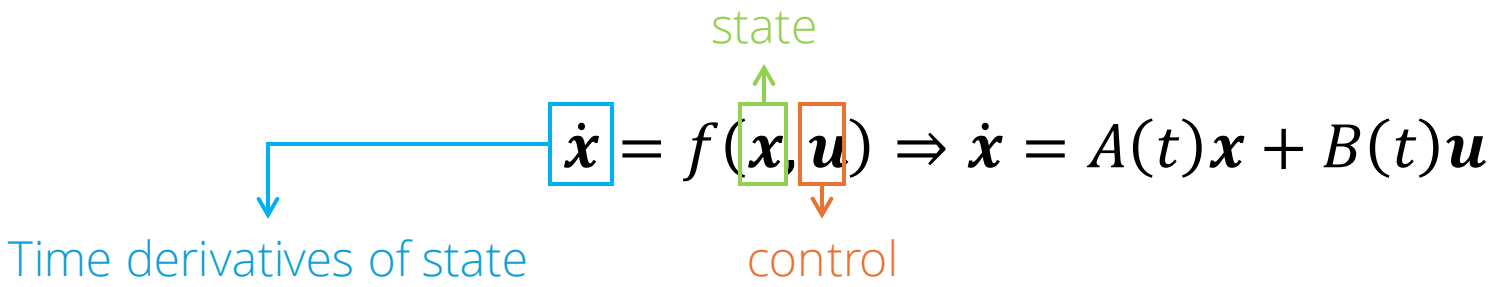


- Shouldn't dynamics consider acceleration / forces $\boldsymbol{F} = m\ddot{\boldsymbol{q}}$?
  - ➢ A state representations that merge multi-order equations into multiple 1st order equations

$$\boldsymbol{x} = \begin{bmatrix} q \\ \dot{q} \end{bmatrix} \Rightarrow \dot{\boldsymbol{x}} = \begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = f\left(\begin{bmatrix} q \\ \dot{q} \end{bmatrix}, \begin{bmatrix} 0 \\ u \end{bmatrix}\right)$$

Linearization of Dynamics

- Approximate non-linear dynamics as linear systems:



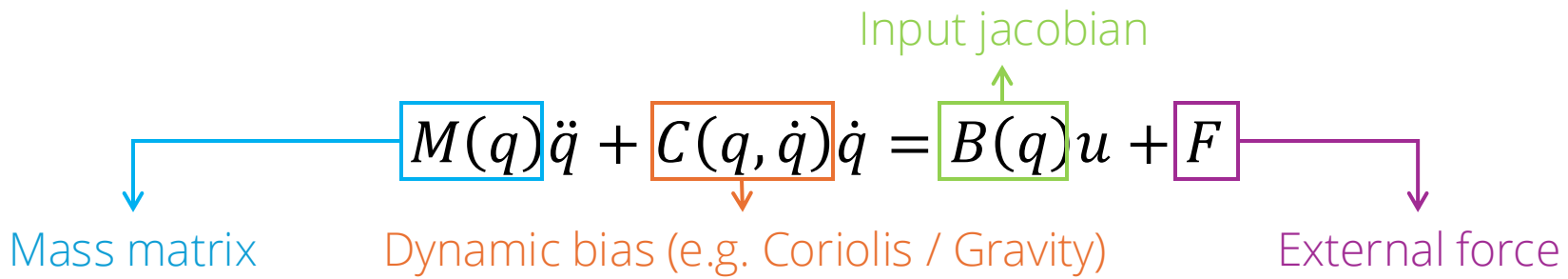
The system is "time invariant" if $A(t) = A$ and $B(t) = B$ for $\forall t$; otherwise, the system is "time varying"

- Linear approximation with Taylor expansion:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) \approx f(0,0) + \underbrace{\frac{\partial f}{\partial x}}_{A(t)} x + \underbrace{\frac{\partial f}{\partial u}}_{B(t)} u$$

Manipulator Dynamics

- Most dynamics in robotics can be written as:



- Continuous-time dynamics of manipulator dynamics:

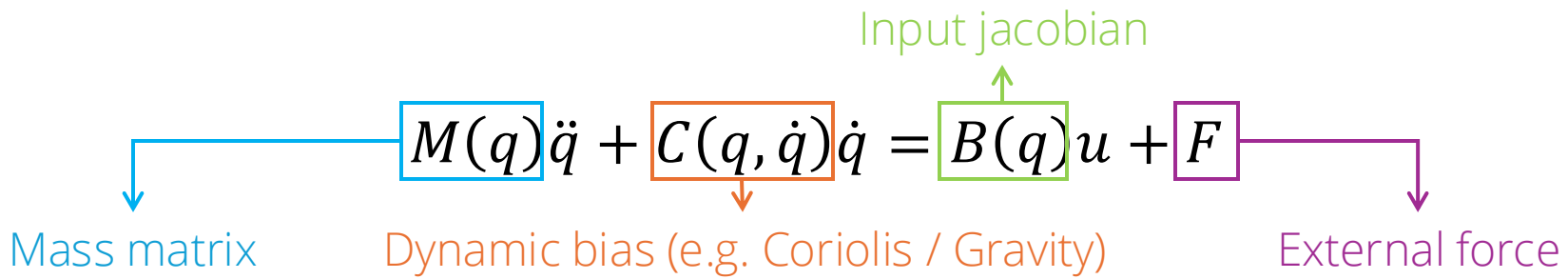
$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} \dot{q} \\ M^{-1}(q)[B(q)u + F - C(q,\dot{q})\dot{q}] \end{bmatrix}$$

which can be linearized as

$$\dot{\boldsymbol{x}} \approx \begin{bmatrix} 0 & I \\ M^{-1}\dfrac{\partial F}{\partial q} + \Sigma_j M^{-1}\dfrac{\partial B_j}{\partial q}u_j & 0 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \\ M^{-1}B \end{bmatrix} u$$

Manipulator Dynamics

- Most dynamics in robotics can be written as:



- Continuous-time dynamics of manipulator dynamics:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} \dot{q} \\ M^{-1}(q)[B(q)u + F - C(q,\dot{q})\dot{q}] \end{bmatrix}$$

which can be linearized as

$$\begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} \approx \begin{bmatrix} 0 & I \\ M^{-1}\dfrac{\partial F}{\partial q} + \Sigma_j M^{-1}\dfrac{\partial B_j}{\partial q}u_j & 0 \end{bmatrix} \begin{bmatrix} q \\ \dot{q} \end{bmatrix} + \begin{bmatrix} 0 \\ M^{-1}B \end{bmatrix} u$$

Continuous to Discrete dynamics

- Continuous-time dynamics: $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) = \frac{d\boldsymbol{x}}{dt}$
- But we can only simulate dynamics discretely: $\frac{\boldsymbol{x}_{\mathrm{t}+\Delta\mathrm{t}} - \boldsymbol{x}_{\mathrm{t}}}{\Delta\mathrm{t}}$
- Time-integration algorithm:
  1. Explicit time integration methods: Next time step can be computed entirely using values from the current time step or before
  2. Implicit time integration methods: Next time step is computed using values from the future
- How to choose the time integration algorithms?
  - Performance
  - Stability
  - Accuracy

Time-integration Algorithm

- Explicit time integration methods (Forward Euler Time Integration):

$$\dot{\boldsymbol{x}} \approx \frac{1}{\Delta t}(\boldsymbol{x}_{t+\Delta \mathrm{t}} - \boldsymbol{x}_t) \Rightarrow \boldsymbol{x}_{t+\Delta \mathrm{t}} = \boldsymbol{x}_t + f(\boldsymbol{x}_t, \boldsymbol{u}_t)\Delta \mathrm{t}$$

Conceptually simple, but simulation often explodes

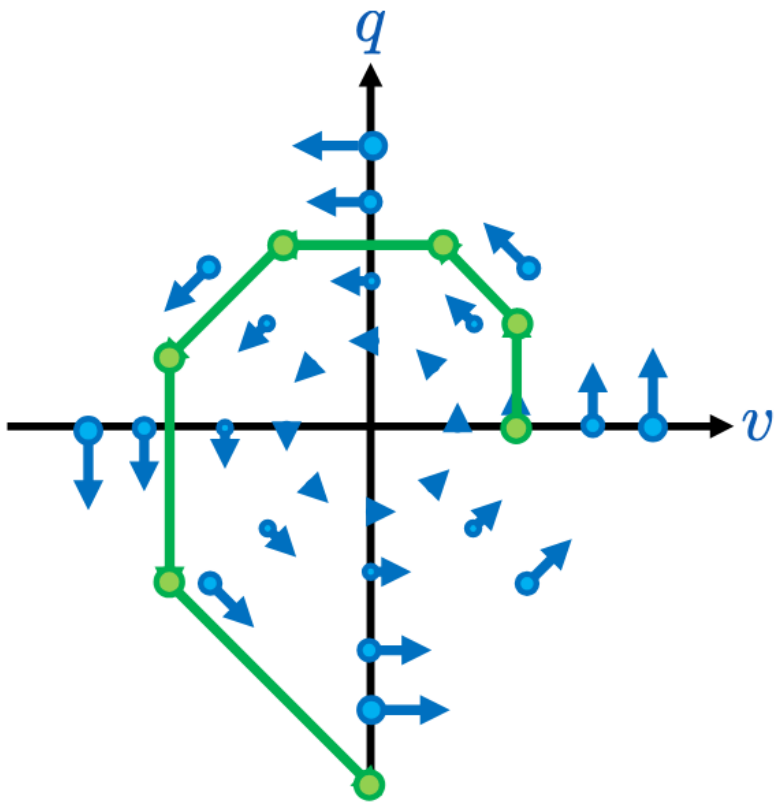
- Implicit time integration methods (Backward Euler Time Integration):

$$\dot{\boldsymbol{x}} \approx \frac{1}{\Delta t}(\boldsymbol{x}_{t+\Delta \mathrm{t}} - \boldsymbol{x}_t) \Rightarrow \boldsymbol{x}_{t+\Delta \mathrm{t}} = \boldsymbol{x}_t + f(\boldsymbol{x}_{t+\Delta \mathrm{t}}, \boldsymbol{u}_{t+\Delta \mathrm{t}})\Delta \mathrm{t}$$

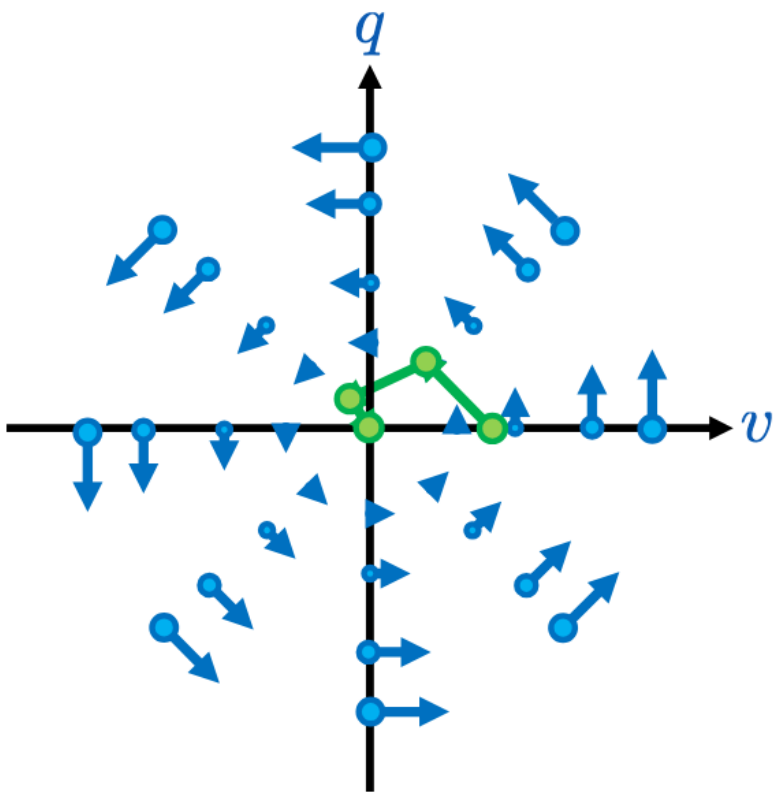
Conceptually complex, while simulation has strong damping

Forward vs. Backward Euler Time Integration

- Can we cancel out the exploding and damping effect? Yes! Let's try to mix these two integration methods.



Explicit (exploding!!!)



Implicit (damping)

Slide credit D. Levin

How to Solve Optimal Control Problems?

- An example of DOC problem:

$$\min_{x,u} \sum_{k=1}^{N-1} c(x_k, u_k) + c_F(x_N)$$

such that

$$x_1 = \hat{x}_1$$

$$x_{k+1} = f(x_k, u_k), \qquad k = 1, \dots, N-1$$

$$u_{min} \geq u_k \geq u_{max}, \qquad k = 1, \dots, N-1$$ --- Torque limits

$$x_{min} \geq x_k \geq x_{max}, \qquad k = 1, \dots, N-1$$ --- Workspace bounds

$$\left| x_{target} - x_N \right| - \varepsilon \leq 0$$ ---------- Target pose

How to Solve Optimal Control Problems?

- Deterministic Optimal Control Problem:

$$\min_{x,u} \sum_{k=1}^{N-1} c(x_k, u_k) + c_F(x_N)$$

such that

$$x_1 = \hat{x}_1$$

$$x_{k+1} = f(x_k, u_k), \qquad k = 1, \dots, N-1$$

$$0 \geq h(x_k, u_k), \qquad k = 1, \dots, N-1$$

$$0 \geq r(x_N)$$

- Options:
  1. Root finding
  2. Constrained minimization
  3. Dynamic programming
  4. Model predictive control

Root-Finding Problems and Newton's Method

- Minimization problems: $\min_x g(x)$, if g is smooth, $\frac{\partial g}{\partial x} = 0$ at local minimum x^*
- Root-finding problems: Given $f(x)$, find x^* such that $f(x^*) = 0$
- Newtons' method:
  - Taylor's expansion: $f(x + \Delta x) \approx f(x) + \frac{\partial f}{\partial x}\Delta x$
  - Set approximation to zero and solve for Δx:

$$f(x) + \frac{\partial f}{\partial x}\Delta x = 0 \Rightarrow \Delta x = -\frac{\partial f}{\partial x}^{-1} \mathrm{g}(x)$$

  - Update $x \leftarrow x + \Delta x$ and repeat
- Newton's method is a local root-finding method, that converges to the closest fixed point (min, max, saddle) to the initial guess

Root-Finding Problems and Newton's Method

- Minimization problems: $\min_x g(x)$, if g is smooth, $\frac{\partial g}{\partial x} = \mathbf{0}$ at local minimum x^*
- Solve $\nabla g = \mathbf{0}$ with Newton's method:

$$\nabla g(x + \Delta x) \approx \nabla g(x) + \frac{\partial}{\partial x}(\nabla g)\Delta x = 0$$

$$\Rightarrow \Delta x = -\frac{\partial}{\partial x}(\nabla g)^{-1}\nabla g(x) \Rightarrow \Delta x = -(\nabla^2 g)^{-1}\nabla g(x)$$

- Issues of Newton's method:
  - Convergence to local optima
  - How to deal with equality / inequality constraints?
  - Need to compute 2nd order derivatives

How to Solve Optimal Control Problems?

- Deterministic Optimal Control Problem:

$$\min_{x,u} \sum_{k=1}^{N-1} c(x_k, u_k) + c_F(x_N)$$

such that

$$x_1 = \hat{x}_1$$

$$x_{k+1} = f(x_k, u_k), \qquad k = 1, \dots, N-1$$

$$0 \geq h(x_k, u_k), \qquad k = 1, \dots, N-1$$

$$0 \geq r(x_N)$$

- Options:
  1. Root finding
  2. Constrained minimization
  3. Dynamic programming
  4. Model predictive control

Inequality-Constrained Minimization Problems

- Minimization problems with inequality constraints: $\min_x g(x)\,, s.t. r(x) \geq 0$
- We can define Lagrangian function:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = g(\boldsymbol{x}) + \boldsymbol{\lambda}^{\mathsf{T}} \boldsymbol{r}(\boldsymbol{x})$$

- K.K.T conditions for optimality:
  - Primal feasibility: $\boldsymbol{r}(\boldsymbol{x}) \geq \boldsymbol{0}$
  - Stationarity: $\nabla g(\boldsymbol{x}) - \boldsymbol{\lambda}^{\mathsf{T}} \nabla \boldsymbol{r}(\boldsymbol{x}) = 0$
  - Dual feasibility: $\boldsymbol{\lambda} \geq \boldsymbol{0}$
  - Complementarity: $\boldsymbol{\lambda}^{\mathsf{T}} \boldsymbol{r}(\boldsymbol{x}) = 0$
- Very complicated…

K.K.T systems:

$$\begin{bmatrix} \frac{\partial^2 \mathcal{L}}{\partial \boldsymbol{x}^2} & \frac{\partial \boldsymbol{r}^{\mathsf{T}}}{\partial \boldsymbol{x}} \\ \frac{\partial \boldsymbol{r}^{\mathsf{T}}}{\partial \boldsymbol{x}} & 0 \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{x} \\ \Delta \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) \\ -\boldsymbol{r}(\boldsymbol{x}) \end{bmatrix}$$

How to Solve Optimal Control Problems?

- Deterministic Optimal Control Problem:

$$\min_{x,u} \sum_{k=1}^{N-1} c(x_k, u_k) + c_F(x_N)$$

such that

$$x_1 = \hat{x}_1$$

$$x_{k+1} = f(x_k, u_k), \qquad k = 1, \dots, N-1$$

$$0 \geq h(x_k, u_k), \qquad k = 1, \dots, N-1$$

$$0 \geq r(x_N)$$

- Options:
  1. Root finding
  2. Constrained minimization
  3. Dynamic programming
  4. Model predictive control

Dynamic Programming

- Principle of Optimality:

"An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."

Bellman, 1957

- In short, Bellman observed that:
  1. Optimal control problems have an inherently sequential structure
  2. Past control inputs can only affect future states; Future control can't affect past states
  3. Sub-trajectories of optimal trajectories have to be optimal

This suggests a recursive structure!

Formulation of Dynamic Programming

- Optimal trajectories for the control problem:

$$J^*(x_k) = \min_{x,u} \sum_{i=k}^{N-1} c(x_i, u_i) + c_F(x_N) \qquad s.t\ x_0 = \hat{x}_0 \ldots$$

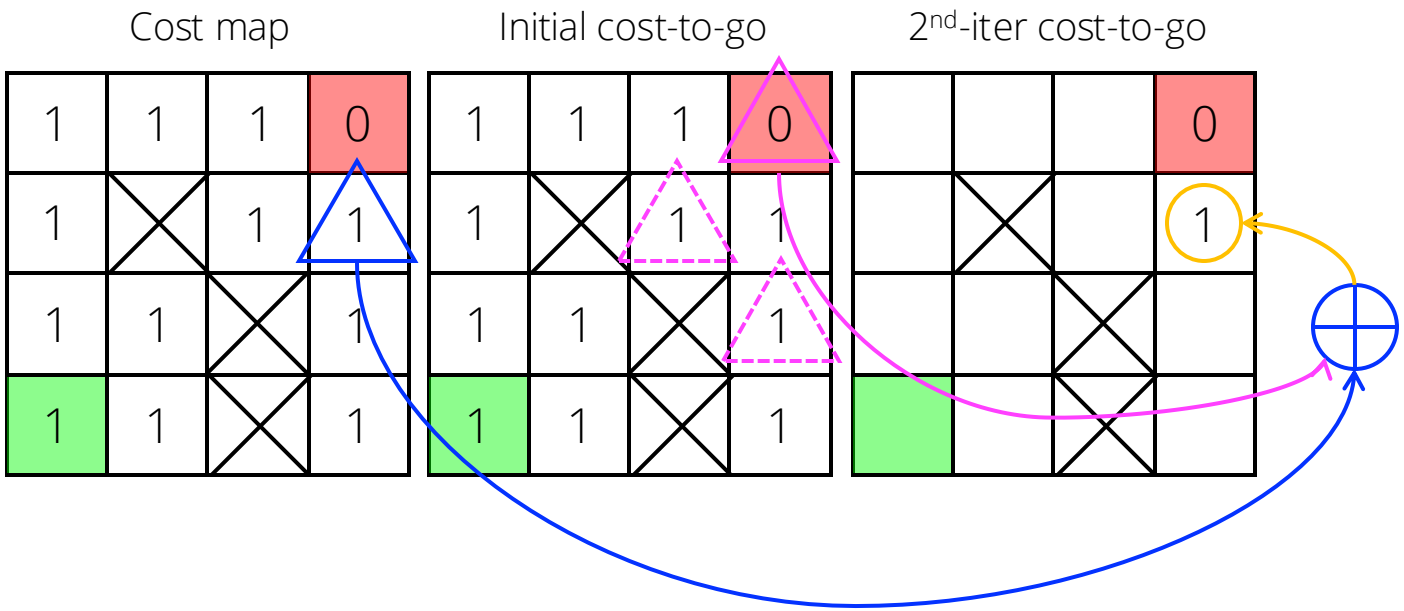
$$\Rightarrow J^*(x_k) = \min_{u_k} c(x_k, u_k) + \min_{\ldots} \sum_{i=k+1}^{N-1} c(x_i, u_i) + c_F(x_N)$$

$$\Rightarrow J^*(x_k) = \min_{u_k} c(x_k, u_k) + J^*(x_{k+1})$$

- J is called cost-to-go function, or "value function" in RL literature
- Optimal cost-to-go function can be obtained by dynamic programming

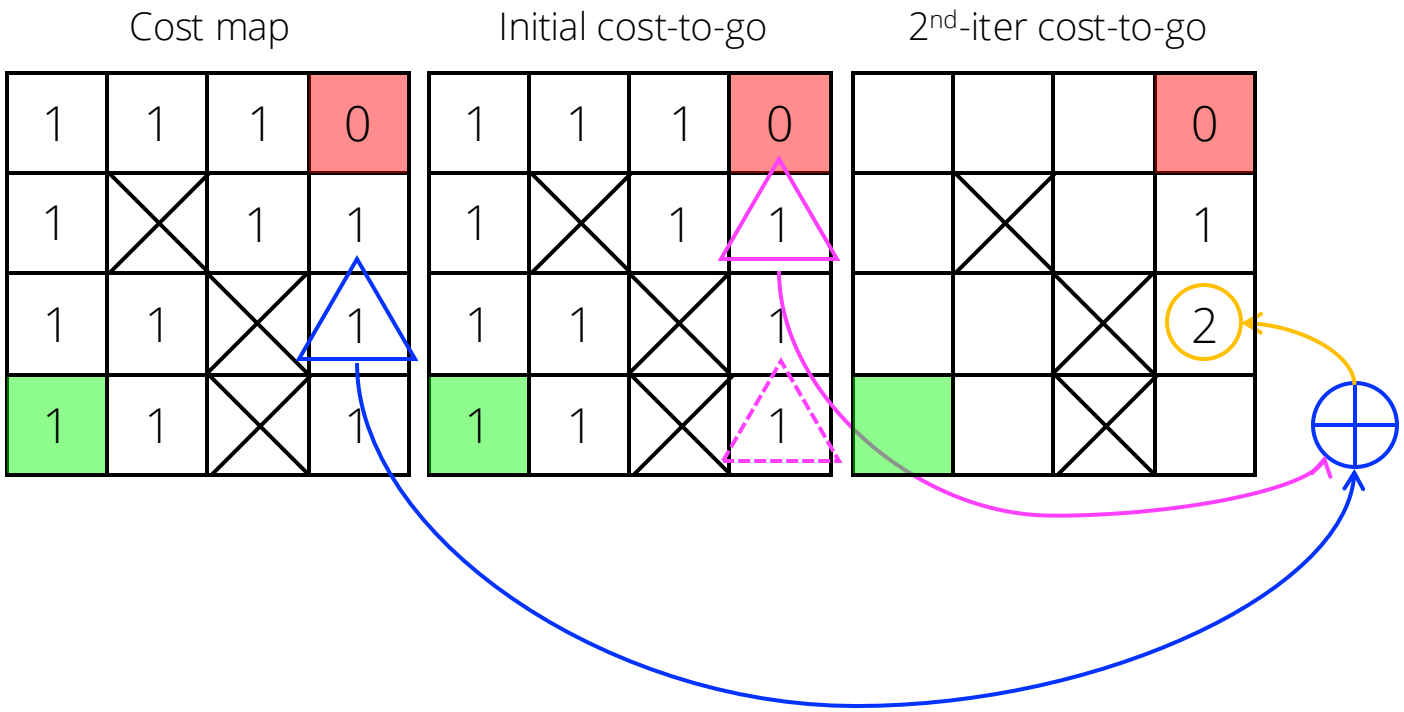
An Example of Estimating Optimal Cost-to-Go Function with Dynamic Programming

$$J^*(x_k) = \min_{u_k} c(x_k, u_k) + J^*(x_{k+1})$$



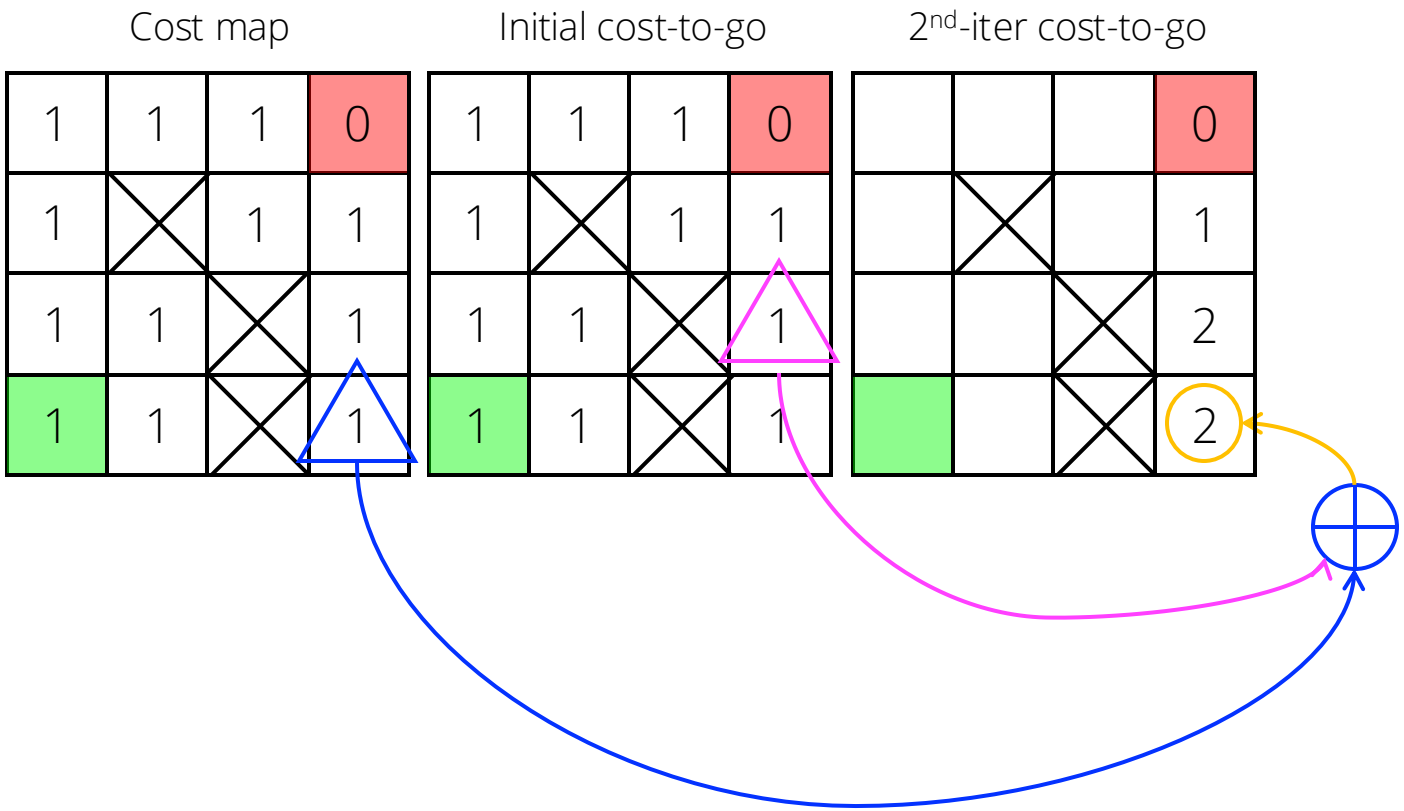
An Example of Estimating Optimal Cost-to-Go Function with Dynamic Programming

$$J^*(x_k) = \min_{u_k} c(x_k, u_k) + J^*(x_{k+1})$$



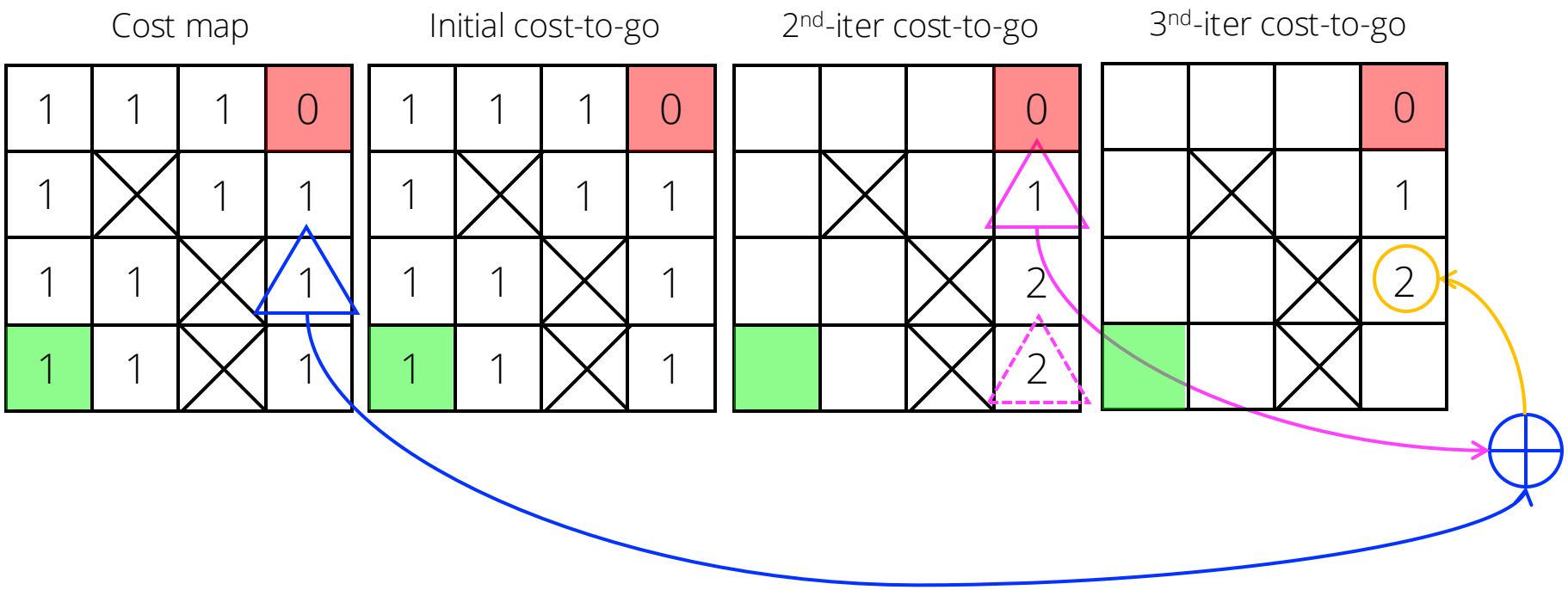
An Example of Estimating Optimal Cost-to-Go Function with Dynamic Programming

$$J^*(x_k) = \min_{u_k} c(x_k, u_k) + J^*(x_{k+1})$$



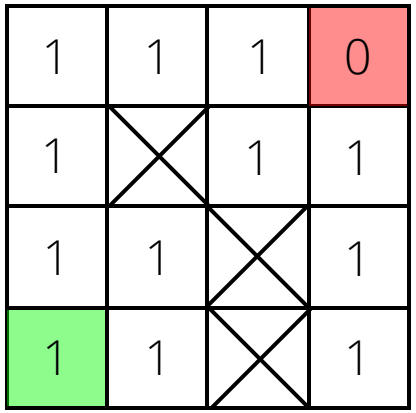
An Example of Estimating Optimal Cost-to-Go Function with Dynamic Programming

$$J^*(x_k) = \min_{u_k} c(x_k, u_k) + J^*(x_{k+1})$$

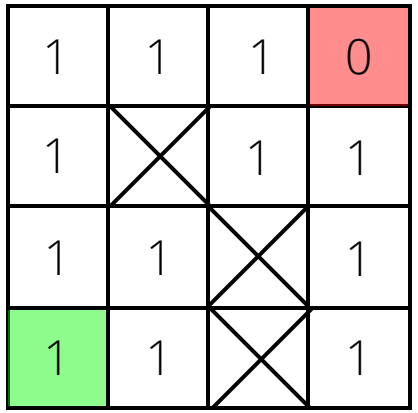


An Example of Estimating Optimal Cost-to-Go Function with Dynamic Programming

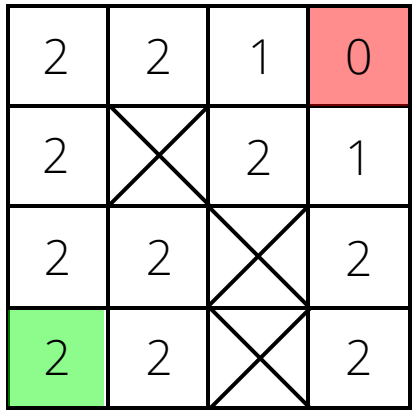
Cost map



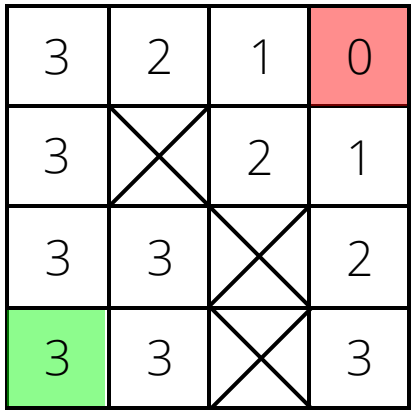
Initial cost-to-go



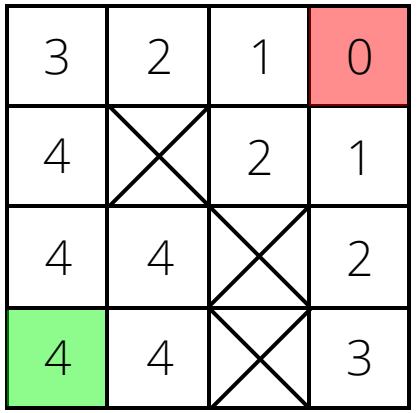
2nd-iter cost-to-go



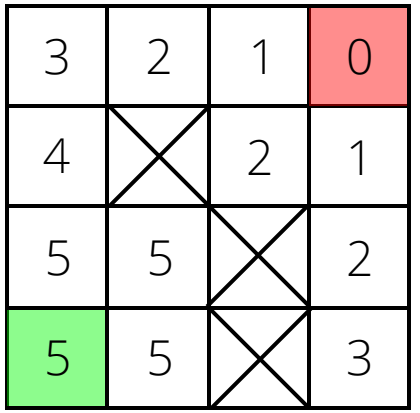
3rd-iter cost-to-go



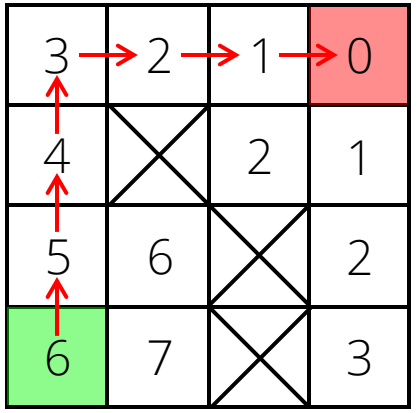
4th-iter cost-to-go



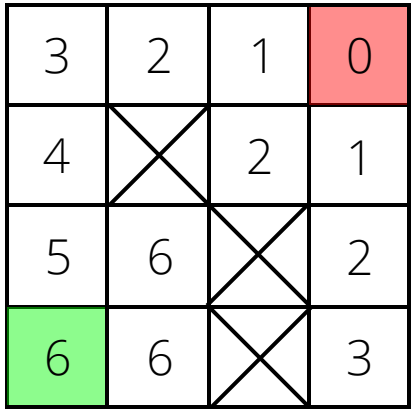
5th-iter cost-to-go



7th-iter cost-to-go



6th-iter cost-to-go



How to Obtain Optimal Policy?

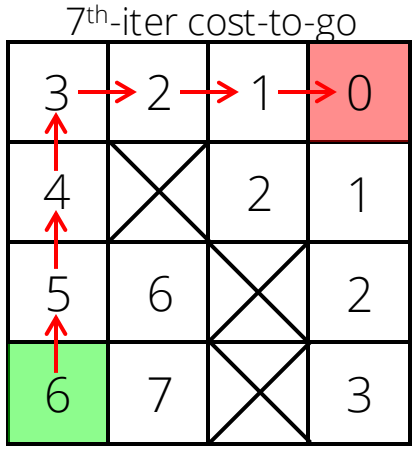
- Q-value function estimates how good a state-action pair is:

$$J^*(x_k) = \min_{u_k} c(x_k, u_k) + J^*(x_{k+1})$$
$$= \min_{u_k} Q^*(x_k, u_k)$$

- Optimal feedback control law:

$$\pi^*(x_k) = \arg\min_{u_k} Q^*(x_k, u_k)$$

- Value iteration:
  1. Compute optimal cost-to-go function $J^*(x)$
  2. Compute optimal policy $\pi^*(x) = \arg\min_{u} Q^*(x, u)$



We will talk more in RL sections!

How to solve optimal control with DP?

Linearized Optimal Control Problems

- Linear Quadratic Regulator (LQR): A very common formulation of control problems, which considers a linear dynamics $f(x,u) = Ax + Bu$ and quadratic cost functions $c(x,u) = x^\intercal Q x + u^\intercal R u$

$$\min_{x,u} \sum_{k=1}^{N-1} \underbrace{x_k{}^\intercal Q x_k + u_k{}^\intercal R u_k}_{\text{cost function } c(x_k,u_k)} + \underbrace{x_N{}^\intercal P x_N}_{\text{terminal cost function } c_f(x_N)}, \quad \underbrace{s.t.\, x_1 = \hat{x}_1, x_{k+1} = Ax_k + Bu_k}_{\text{Linearized dynamic functions}}$$

- Both Q and R are positive definite matrices: $z^\intercal Q z \geq 0$ and $z^\intercal R z \geq 0$ for any z
- We can obtain analytical solution!

Solve LQR via Dynamic Programming

- Let's ignore constraints in LQR:

$$\min_u \sum_{k=1}^{N-1} \frac{1}{2} {x_k}^\intercal Q x_k + \frac{1}{2} {u_k}^T R u_k + \frac{1}{2} {x_N}^\intercal P x_N$$

- Optimal cost-to-go functions:
  - Let $J_N(x) = \frac{1}{2} {x_N}^T P x_N$
  - Back up one step and compute

$$J_{N-1}(x) = \min_u \frac{1}{2} {x_{N-1}}^\intercal Q x_{N-1} + \frac{1}{2} {u_{N-1}}^\intercal R u_{N-1} + J_N(A x_{N-1} + B u_{N-1})$$

Solve LQR via Dynamic Programming

- Optimal cost-to-go functions at step $N-1$:

$$J_{N-1}(x) = \min_u \frac{1}{2} x_{N-1}{}^\intercal Q x_{N-1} + \frac{1}{2} u_{N-1}{}^\intercal R u_{N-1} + J_N(Ax_{N-1} + Bu_{N-1})$$

$$\Rightarrow J_{N-1}(x) = \min_u \frac{1}{2} x_{N-1}{}^\intercal Q x_{N-1} + \frac{1}{2} u_{N-1}{}^\intercal R u_{N-1} + \frac{1}{2}(Ax_{N-1} + Bu_{N-1})^\intercal P(Ax_{N-1} + Bu_{N-1})$$

$$\Rightarrow \nabla J_{N-1}(x) = 0 \Rightarrow Ru_{N-1} + B^\intercal P(Ax_{N-1} + Bu_{N-1}) = 0$$

$$\Rightarrow u_{N-1} = -(R + B^\intercal PB)^{-1} BPA x_{N-1} = -K_{N-1} x_{N-1}$$

- Plug $u_{N-1} = -K_{N-1} x_{N-1}$ back to $J_{N-1}(x)$:

$$J_{N-1}(x) = \frac{1}{2} x^\intercal \left[Q + K_{N-1}{}^\intercal R K_{N-1} + (A - BK_{N-1})^\intercal P (A - BK_{N-1}) \right] x = \frac{1}{2} x^\intercal P_{N-1} x$$

Solve LQR via Dynamic Programming

- We can solve the problem recursively:

$$J_N(x) \leftarrow c_F(x_N)$$

$$k \leftarrow N$$

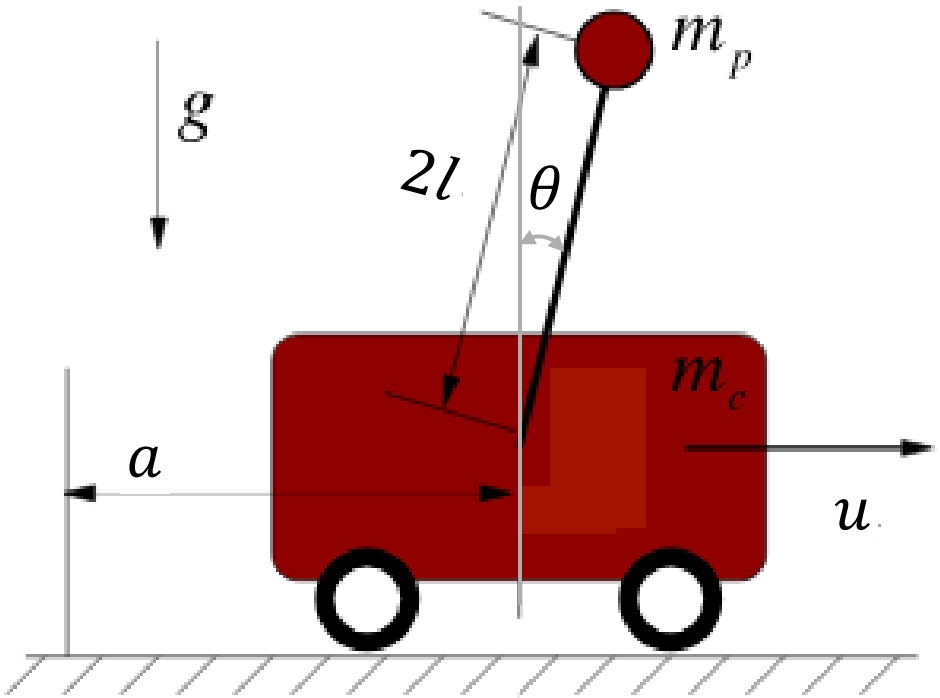
`while` $k > 1$

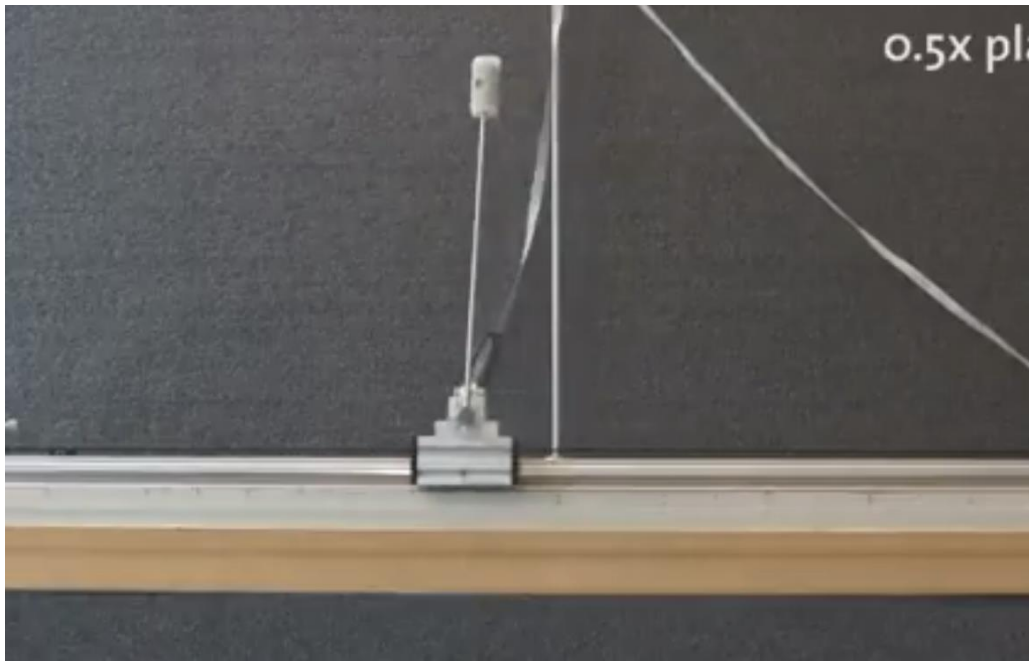
$$J_{k-1}(x) = \min_u \left[c(x_{k-1}) + J_k\left(f(x_{k-1}, u_{k-1})\right) \right]$$

$$k \leftarrow k - 1$$

`end`

An Example of Cart Pole





Slide adapted from Jia-Wei's note

Non-linear Dynamics of Cart Pole

- Let the pivot point locate at $(a, 0)$
- The coordinate of the pole's center of mass are:

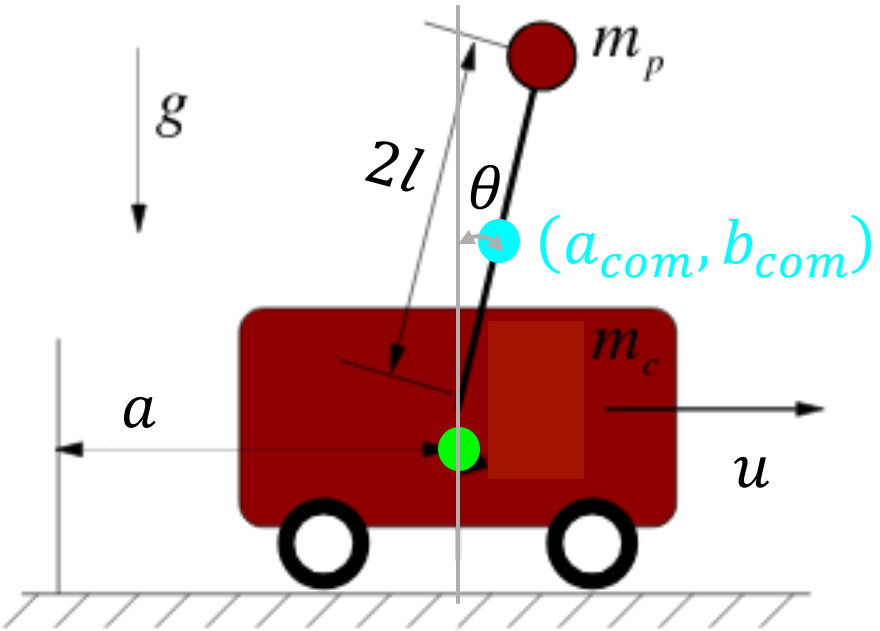
$$a_{com} = a + l \sin\theta$$
$$b_{com} = b + l \cos\theta$$

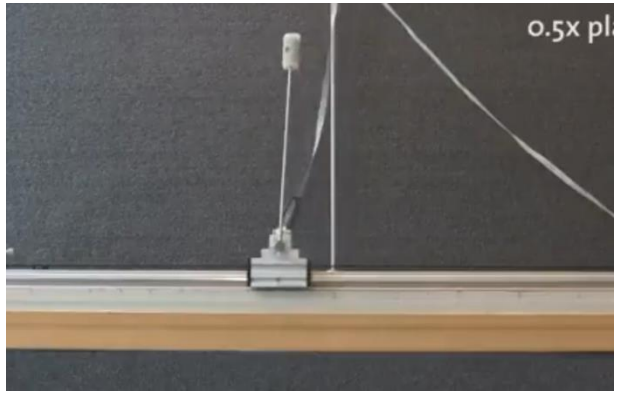
- The velocities are:

$$\dot{a}_{com} = \dot{a} + l \cos\theta \, \dot{\theta}$$
$$\dot{b}_{com} = -l \sin\theta \, \dot{\theta}$$

- The accelerations are:

$$\ddot{a}_{com} = \ddot{a} - l \sin\theta \, \dot{\theta}^2 + l \cos\theta \, \ddot{\theta}$$
$$\ddot{b}_{com} = -l \cos\theta \, \dot{\theta}^2 - l \sin\theta \, \ddot{\theta}$$





Non-linear Dynamics of Cart Pole

- Let the (H, V) denote the horizontal and vertical reaction forces at the pivot:

$$m_p \ddot{a}_{com} = H$$

$$m_p \ddot{b}_{com} = V - m_p g$$

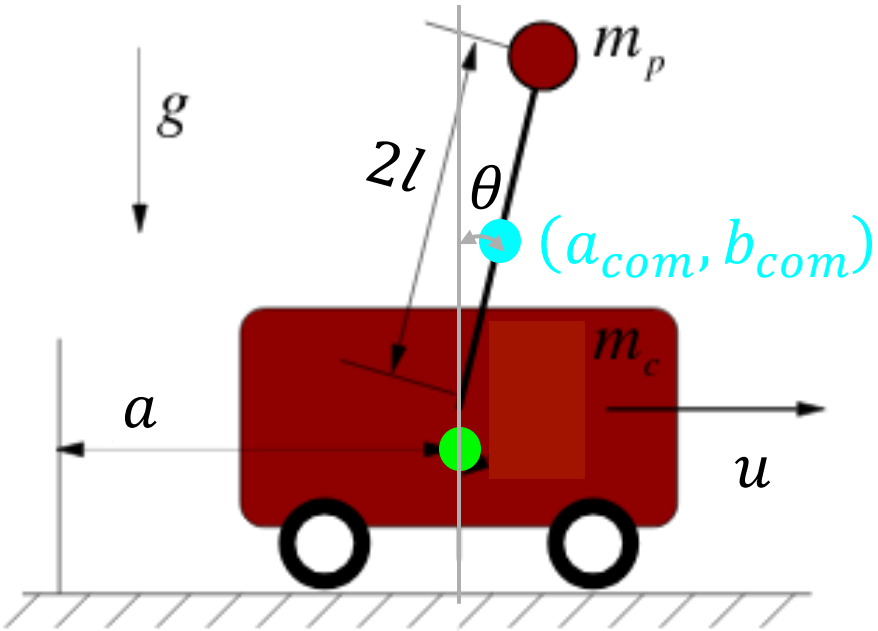
- The rotational dynamics about the center of mass:

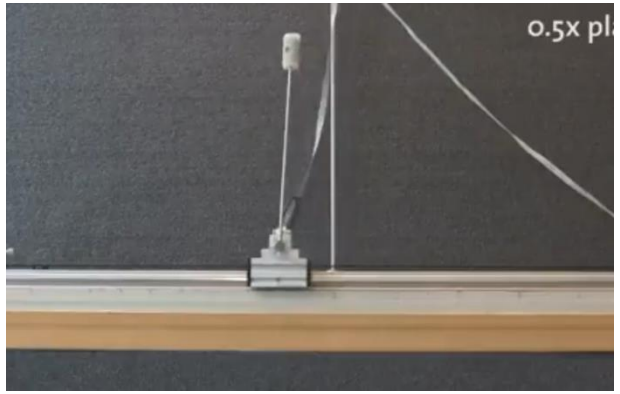
$$I_{com} \ddot{\theta} = l(H \cos\theta - V \sin\theta)$$

where

$$I_{com} = \frac{1}{12} m_p (2l)^2$$

- Force input: $m_c \ddot{a} = u - H$





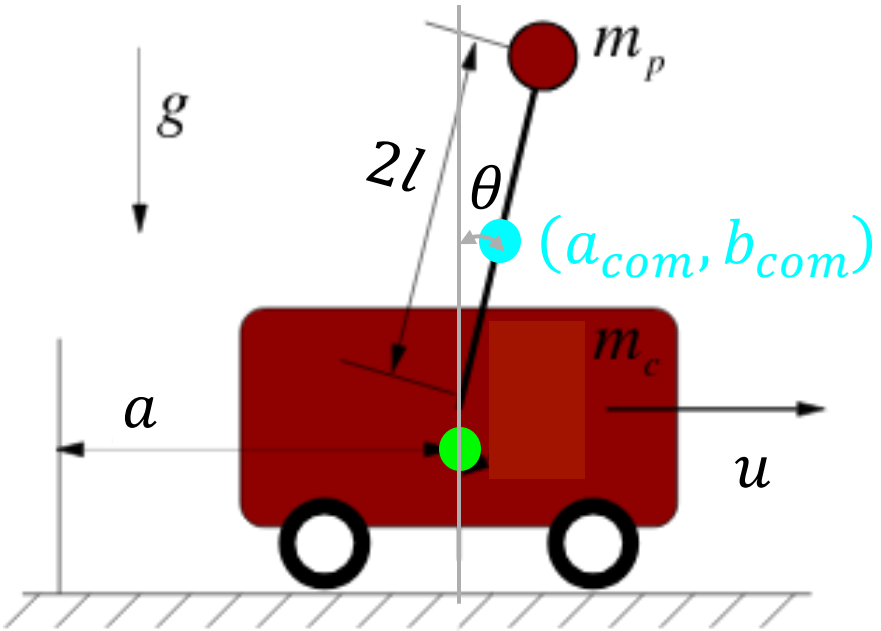
Non-linear Dynamics of Cart Pole

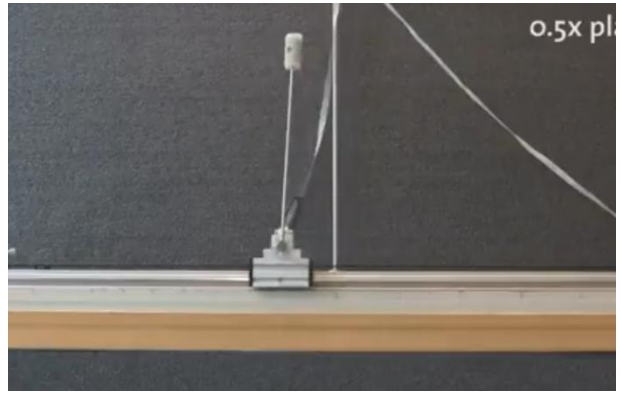
- The accelerations are:

$$\ddot{a}_{com} = \ddot{a} - l\sin\theta\,\dot{\theta}^2 + l\cos\theta\,\ddot{\theta}$$
$$\ddot{b}_{com} = -l\cos\theta\,\dot{\theta}^2 - l\,sin\,\theta\,\ddot{\theta}$$

- Rewrite (H, V) as:

$$H = m_p\ddot{a}_{com} = m_p\left(\ddot{a} - l\sin\theta\,\dot{\theta}^2 + l\cos\theta\,\ddot{\theta}\right)$$
$$V = m_p\ddot{b}_{com} + m_p g = m_p\left(-l\cos\theta\,\dot{\theta}^2 - l\,sin\,\theta\,\ddot{\theta} + g\right)$$





Slide adapted from Jia-Wei's note

Non-linear Dynamics of Cart Pole

- Rewrite (H, V) as:

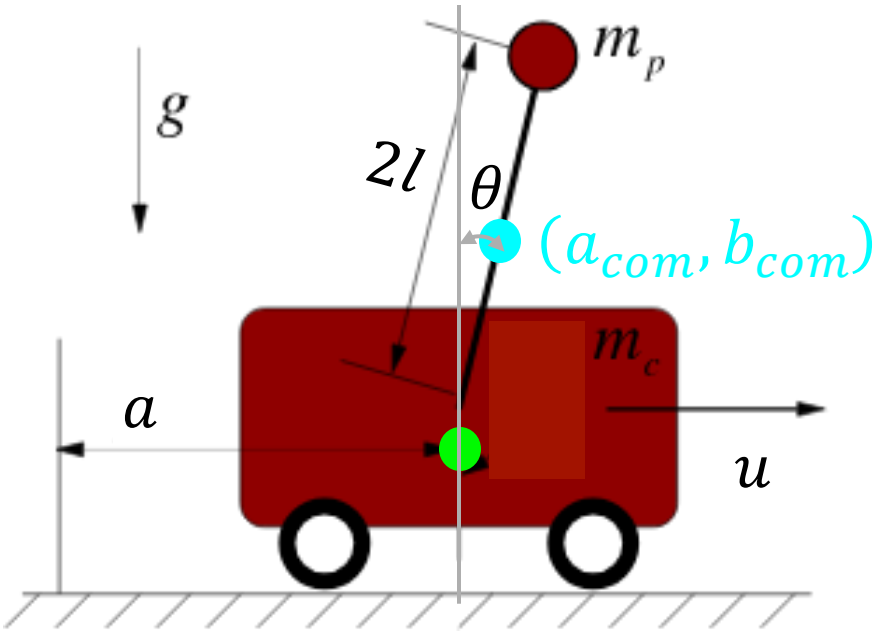
$$H = m_p \ddot{a}_{com} = m_p\left(\ddot{a} - l\sin\theta\,\dot{\theta}^2 + l\cos\theta\,\ddot{\theta}\right)$$

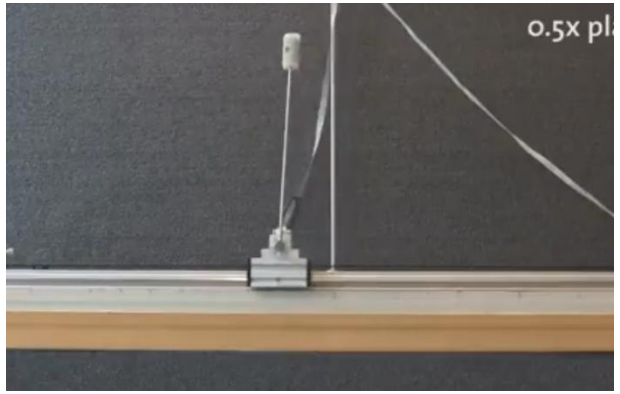
$$V = m_p \ddot{b}_{com} + m_p g = m_p\left(-l\cos\theta\,\dot{\theta}^2 - l\,sin\,\theta\,\ddot{\theta} + g\right)$$

- Substitute back to:

$$I_{com}\ddot{\theta} = l(H\cos\theta - V\sin\theta)$$

$$\Rightarrow \frac{4}{3} m_p l^2 \ddot{\theta} + m_p l\cos\theta\,\ddot{a} + m_p g l\,sin\,\theta = 0$$





Slide adapted from Jia-Wei's note

Non-linear Dynamics of Cart Pole

- Rewrite (H, V) as:

$$H = m_p \ddot{a}_{com} = m_p\left(\ddot{a} - l\sin\theta\,\dot{\theta}^2 + l\cos\theta\,\ddot{\theta}\right)$$

$$V = m_p \ddot{b}_{com} + m_p g = m_p\left(-l\cos\theta\,\dot{\theta}^2 - l\,sin\,\theta\,\ddot{\theta} + g\right)$$

- Substitute back to:

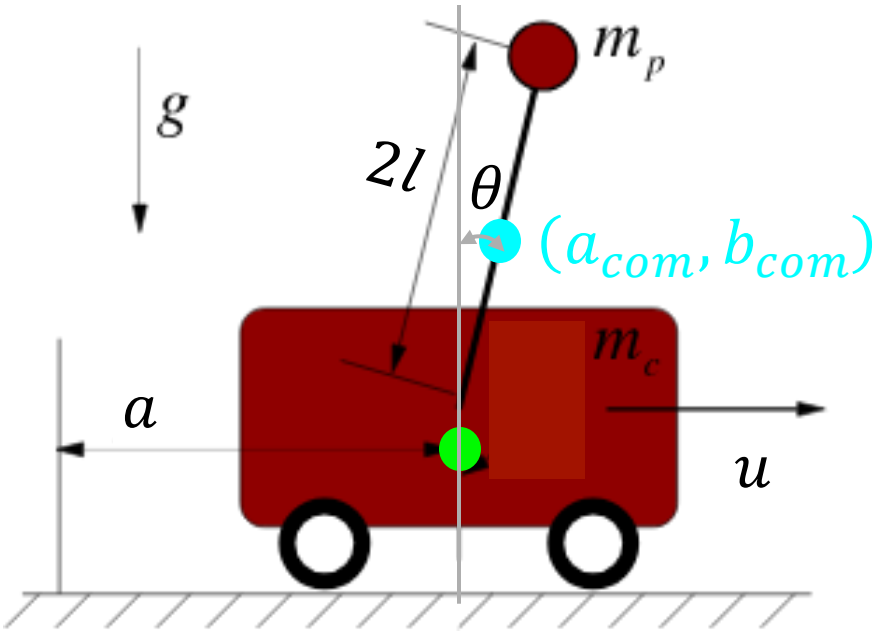
$$I_{com}\ddot{\theta} = l(H\cos\theta - V\sin\theta)$$

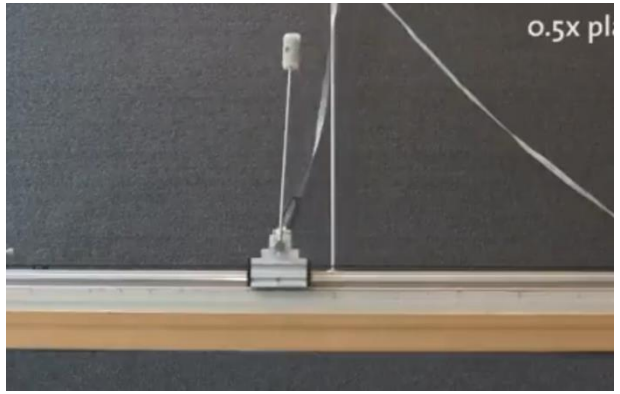
$$\Rightarrow \frac{4}{3} m_p l^2 \ddot{\theta} + m_p l\cos\theta\,\ddot{a} + m_p g l\,sin\,\theta = 0$$

- Substitute back to:

$$m_c \ddot{a} = u - H$$

$$\Rightarrow \left(m_c + m_p\right)\ddot{a} + m_p l\cos\theta\,\ddot{\theta} - m_p l\,sin\,\theta\,\dot{\theta}^2 = u$$



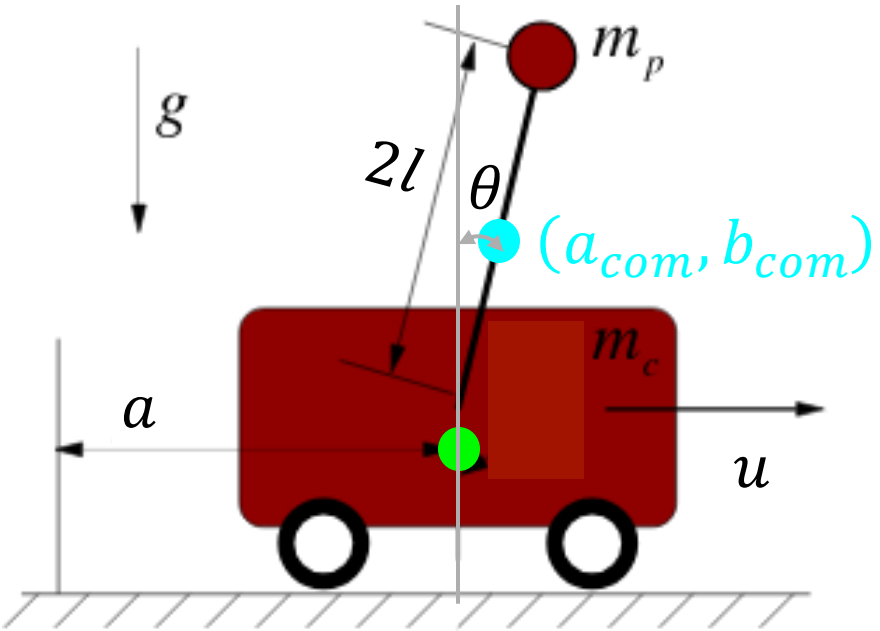


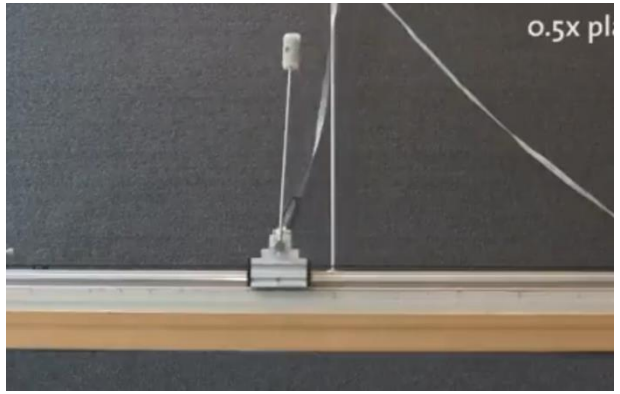
Non-linear Dynamics of Cart Pole

- The final dynamics:

$$\ddot{\theta} = \frac{g\,sin\,\theta - \cos\theta\,\frac{u+m_p l\dot{\theta}^2\,sin\,\theta}{(m_c+m_p)}}{l\left(\frac{4}{3} - \frac{m_p}{(m_c+m_p)}\cos^2\theta\right)}$$

$$\ddot{a} = \frac{u + m_p l\dot{\theta}^2\,sin\,\theta}{(m_c+m_p)} - \frac{m_p l\cos\theta}{(m_c+m_p)}\ddot{\theta}$$





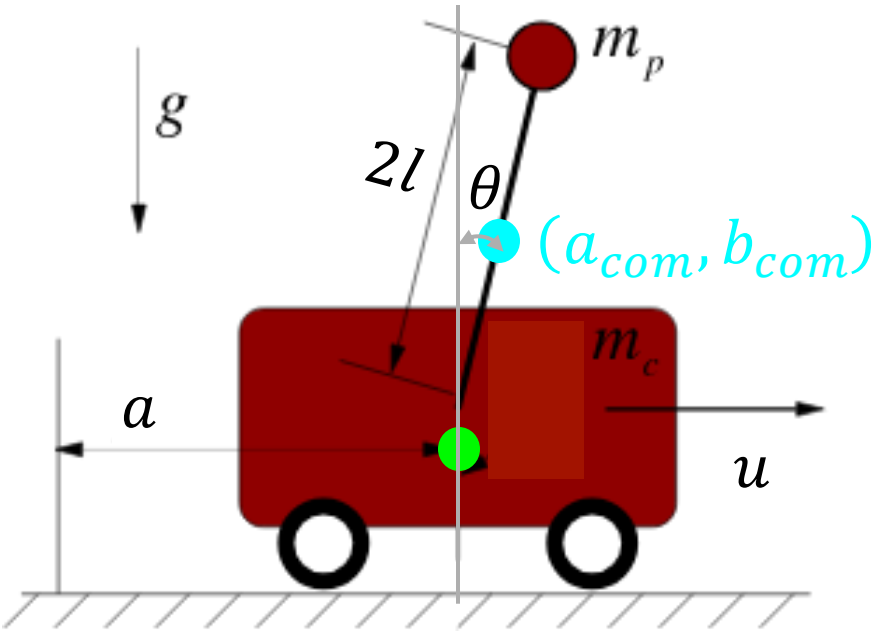
Slide adapted from Jia-Wei's note

Non-linear Dynamics of Cart Pole

- The final dynamics:

$$\ddot{\theta} = \frac{g\sin\theta - \cos\theta \frac{u + m_p l\dot{\theta}^2 \sin\theta}{(m_c + m_p)}}{l\left(\frac{4}{3} - \frac{m_p}{(m_c + m_p)}\cos^2\theta\right)}$$

$$\ddot{a} = \frac{u + m_p l\dot{\theta}^2 \sin\theta}{(m_c + m_p)} - \frac{m_p l\cos\theta}{(m_c + m_p)}\ddot{\theta}$$

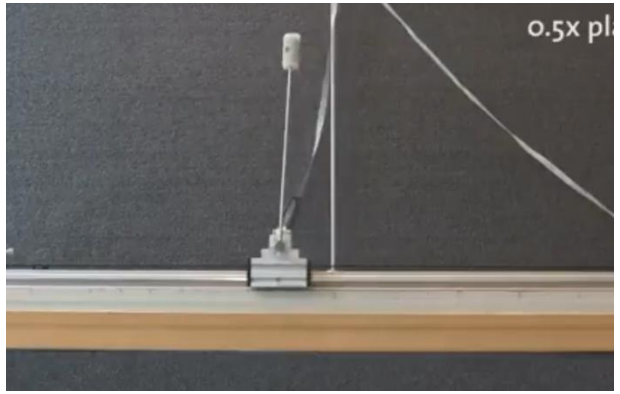


$$\min_{x,u} \sum_{k=1}^{N-1} c(x_k, u_k) + c_F(x_N)$$

such that

$$x_1 = \hat{x}_1$$
$$x_{k+1} = f(x_k, u_k), \quad k = 1, \ldots, N-1$$
$$0 \geq h(x_k, u_k), \quad k = 1, \ldots, N-1$$
$$0 \geq r(x_N)$$

Let's linearize dynamics!



Slide adapted from Jia-Wei's note

Linearized Dynamics of Cart Pole

- The equations of motions:

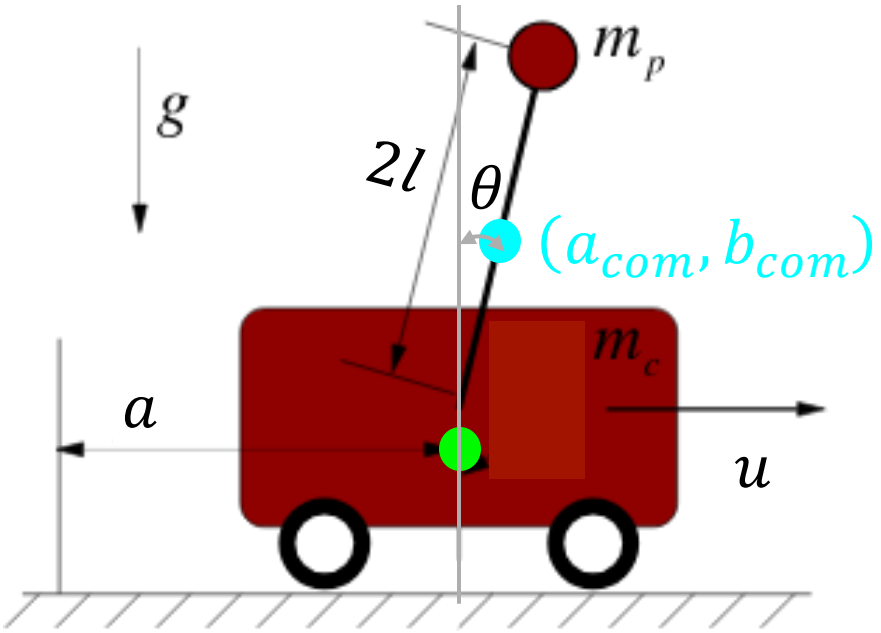
$$u = (m_c + m_p)\ddot{a} + m_p l\ddot{\theta}\cos\theta - m_p l\dot{\theta}^2 \sin\theta$$

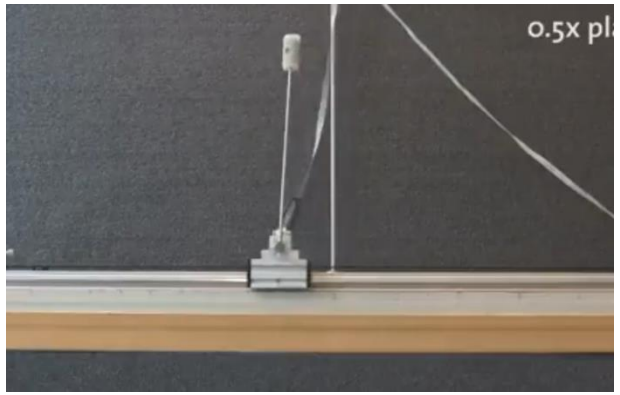
$$0 = \frac{4}{3} m_p l^2 \ddot{\theta} + m_p l \cos\theta\, \ddot{a} + m_p g l \sin\theta$$

which can be re-written as

$$\underset{B(q)}{\begin{bmatrix} 1 \\ 0 \end{bmatrix}} u + \underset{F}{\begin{bmatrix} 0 \\ -m_p g l \sin\theta \end{bmatrix}} =$$

$$\underset{M(q)}{\begin{bmatrix} m_c + m_p & m_p l\cos\theta \\ m_p l\cos\theta & \frac{4}{3} m_p l^2 \end{bmatrix}} \begin{bmatrix} \ddot{a} \\ \ddot{\theta} \end{bmatrix} + \underset{C(q,\dot{q})}{\begin{bmatrix} 0 & -m_p l\dot{\theta}\sin\theta \\ 0 & 0 \end{bmatrix}} \begin{bmatrix} \dot{a} \\ \dot{\theta} \end{bmatrix}$$

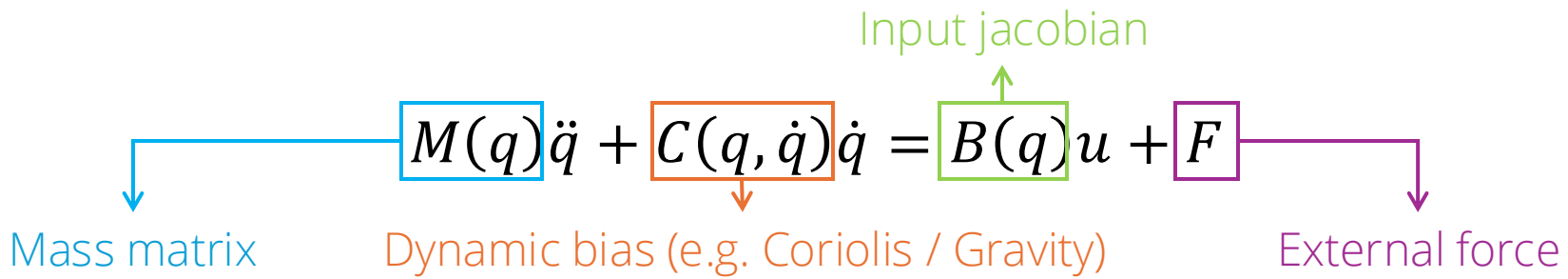




Slide adapted from Underactuated Robots by R. Tedrake

Recap: Manipulator Dynamics

- Most dynamics in robotics can be written as:



- Continuous-time dynamics of manipulator dynamics:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} \dot{q} \\ M^{-1}(q)[B(q)u + F - C(q,\dot{q})\dot{q}] \end{bmatrix}$$

which can be linearized as

$$\begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} \approx \begin{bmatrix} 0 & I \\ M^{-1}\dfrac{\partial F}{\partial q} + \Sigma_j M^{-1}\dfrac{\partial B_j}{\partial q}u_j & 0 \end{bmatrix} \begin{bmatrix} q \\ \dot{q} \end{bmatrix} + \begin{bmatrix} 0 \\ M^{-1}B \end{bmatrix} u$$

An Example of Cart Pole

- The equations of motions:

$$u = (m_c + m_p)\ddot{a} + m_p l\ddot{\theta}\cos\theta - m_p l\dot{\theta}^2 \sin\theta$$

$$0 = \frac{4}{3} m_p l^2 \ddot{\theta} + m_p l \cos\theta\, \ddot{a} + m_p g l \sin\theta$$

which can be re-written as

$$\underbrace{\begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{B(q)} u + \underbrace{\begin{bmatrix} 0 \\ -m_p g l \sin\theta \end{bmatrix}}_{F} =$$

$$\underbrace{\begin{bmatrix} m_c + m_p & m_p l\cos\theta \\ m_p l\cos\theta & \frac{4}{3} m_p l^2 \end{bmatrix}}_{M(q)} \begin{bmatrix} \ddot{a} \\ \ddot{\theta} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 & -m_p l\dot{\theta}\sin\theta \\ 0 & 0 \end{bmatrix}}_{C(q,\dot{q})} \begin{bmatrix} \dot{a} \\ \dot{\theta} \end{bmatrix}$$

- We have

$$\frac{\partial F}{\partial q} = \begin{bmatrix} 0 & 0 \\ 0 & -m_p g l \cos\theta\, \dot{\theta} \end{bmatrix}$$

$$\frac{\partial B_j}{\partial q} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

- We know linearized dynamics!

Solve Cart Pole with Linear Quadratic Regulator

- Deterministic Optimal Control Problem:

$$\min_{x,u} \sum_{k=1}^{N-1} \frac{1}{2} x_k{}^{\intercal} Q x_k + \frac{1}{2} u_k{}^{\intercal} R u_k + \frac{1}{2} x_N{}^{\intercal} P x_N$$

such that

$$x_1 = \hat{x}_1$$

$$x_{k+1} = A x_k + B u_k, \qquad k = 1, \ldots, N-1$$

- Both Q and R are positive definite matrices: $z^T Q z \geq 0$ and $z^T R z \geq 0$ for any z

Solve Cart Pole with Linear Quadratic Regulator

- Deterministic Optimal Control Problem:

$$\min_{x,u} \sum_{k=1}^{N-1} \frac{1}{2} x_k{}^{\intercal} Q x_k + \frac{1}{2} u_k{}^{\intercal} R u_k + \frac{1}{2} x_N{}^{\intercal} P x_N$$

What does these losses mean?

such that

$$x_1 = \hat{x}_1$$

$$x_{k+1} = A x_k + B u_k, \qquad k = 1, \dots, N-1$$

- LQR is a state-feedback controller.
  1. We can consider this formulation as minimizing the tracking error with a reference point of $x_{ref} = 0$.
  2. We can generalize the formulation to track any non-zero trajectory
- We can generalize the formulation to solve non-linear systems

Solution 1: Solve LQR with DP

- Let's consider $k = N - 1$

$$\min_{x,u} x_{N-1}^{\intercal} Q x_{N-1} + u_{N-1}^{\intercal} R u_{N-1} + x_N{}^T P x_N, \qquad s.t. x_0 = \hat{x}_0, x_{k+1} = Ax_k + Bu_k$$

cost function $c(x_k, u_k)$

terminal cost function $c_f(x_N)$

Solution 1: Solve LQR with DP

- Let's consider $k = N - 1$

$$\min_{x,u} x_{N-1}{}^{\intercal} Q x_{N-1} + u_{N-1}{}^{\intercal} R u_{N-1} + x_N{}^{T} P x_N, \qquad s.t. x_0 = \hat{x}_0, x_{k+1} = Ax_k + Bu_k$$

cost function $c(x_k, u_k)$

terminal cost function $c_f(x_N)$

Also known as cost-to-go function

$$J_{N-1}(x_{N-1}) = \min_u x_{N-1}{}^{\intercal} Q x_{N-1} + u_{N-1}{}^{\intercal} R u_{N-1} + J_N(x_N)$$

$$= \min_u x_{N-1}{}^{\intercal} Q x_{N-1} + u_{N-1}{}^{\intercal} R u_{N-1} + J_N(Ax_{N-1} + Bu_{N-1})$$

$$= \min_u x_{N-1}{}^{\intercal} Q x_{N-1} + u_{N-1}{}^{\intercal} R u_{N-1} + (Ax_{N-1} + Bu_{N-1})^{\intercal} P (Ax_{N-1} + Bu_{N-1})$$

Solution 1: Solve LQR with DP

1. Let's consider $k = N - 1$

$$J_{N-1}(x) = \min_u x^\intercal Q x + u^\intercal R u + J_N(Ax + Bu)$$
$$= \min_u x^\intercal Q x + u^\intercal R u + (Ax + Bu)^\intercal P (Ax + Bu)$$

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

$$\nabla_u J_{N-1}(x) = 0 \Rightarrow 2Ru + 2B^\intercal P(Ax + Bu) = 0$$
$$\Rightarrow u = -\underbrace{(R + B^T P B)^{-1} B^\intercal P A}_{K_{N-1}} x$$

Solution 1: Solve LQR with DP

1. Let's consider $k = N - 1$

$$J_{N-1}(x) = \min_u x^\intercal Q x + u^\intercal R u + J_N(Ax + Bu)$$
$$= \min_u x^\intercal Q x + u^\intercal R u + (Ax + Bu)^\intercal P (Ax + Bu)$$

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

$$\nabla_u J_{N-1}(x) = 0 \Rightarrow 2Ru + 2B^\intercal P(Ax + Bu) = 0$$
$$\Rightarrow u = -\underbrace{(R + B^T P B)^{-1} B^\intercal P A}_{K_{N-1}} x$$

3. Substitute (2) into (1), we obtain:

$$J_{N-1}(x) = x^\intercal \underbrace{[Q + {K_{N-1}}^\intercal R K_{N-1} + (A + BK_{N-1})^\intercal P (A + BK_{N-1})]}_{P_{N-1}} x \Rightarrow J_{N-1}(x) = x^\intercal P_{N-1} x$$

Solution 1: Solve LQR with DP

1. Let's consider $k = N - 2$

$$J_{N-2}(x) = \min_u x^\intercal Q x + u^\intercal R u + J_{N-1}(Ax + Bu)$$
$$= \min_u x^\intercal Q x + u^\intercal R u + (Ax + Bu)^\intercal P_{N-1}(Ax + Bu)$$

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

$$\nabla_u J_{N-2}(x) = 0 \Rightarrow 2Ru + 2B^\intercal P_{N-1}(Ax + Bu) = 0$$
$$\Rightarrow u = -(R + \underbrace{B^\intercal P_{N-1} B)^{-1} B^\intercal P_{N-1}}_{K_{N-1}} Ax$$

Slide adapted from P. Abbeel

Solution 1: Solve LQR with DP

1. Let's consider $k = N - 2$

$$J_{N-2}(x) = \min_u x^\intercal Q x + u^\intercal R u + J_{N-1}(Ax + Bu)$$
$$= \min_u x^\intercal Q x + u^\intercal R u + (Ax + Bu)^\intercal P_{N-1}(Ax + Bu)$$

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

$$\nabla_u J_{N-2}(x) = 0 \Rightarrow 2Ru + 2B^\intercal P_{N-1}(Ax + Bu) = 0$$
$$\Rightarrow u = -\underbrace{(R + B^\intercal P_{N-1} B)^{-1} B^\intercal P_{N-1}}_{K_{N-2}} Ax$$

3. Substitute (2) into (1), we obtain:

$$J_{N-2}(x) = x^\intercal [\underbrace{Q + K_{N-2}{}^\intercal R K_{N-2} + (A + BK_{N-2})^\intercal P_{N-1}(A + BK_{N-1})}_{P_{N-2}}]x \Rightarrow J_{N-2}(x) = x^\intercal P_{N-2} x$$

Slide adapted from P. Abbeel

Solution 1: Solve LQR with DP

Problem: $\min_{x,u} \sum_{k=0}^{N-1} x_k{}^\intercal Q x_k + u_k{}^\intercal R u_k + x_N{}^\intercal P x_N, \;\; s.t. \, x_0 = \hat{x}_0, x_{k+1} = A x_k + B u_k$

Set $P_N = P$

For $\mathrm{i} = N - 1 \ldots 0$

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

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

we have

$$u_i = K_i x$$

$$J_i(x) = x^\intercal P_i x$$

Slide adapted from P. Abbeel

Solution 1: Solve LQR with DP

Problem: $\min_{x,u} \sum_{k=0}^{N-1} x_k{}^\intercal Q x_k + u_k{}^\intercal R u_k + x_N{}^\intercal P x_N, \;\; s.t. \, x_0 = \hat{x}_0, x_{k+1} = A x_k + B u_k$

Requires tuning!

Set $P_N = P$

For $\mathrm{i} = N - 1 \ldots 0$

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

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

we have

$$u_i = K_i x$$

$$J_i(x) = x^\intercal P_i x$$

Look familiar? Remember we talked about Proportional Controller: $u(t) = K_p q_e(t)$.
LQR is a state-feedback controller, with a zero-valued reference trajectory!

We can Solve LQR was Quadratic Programming

- Minimization problems with inequality constraints: $\min_x g(x), s.t. r(x) \geq 0$
- We can define Lagrangian function:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = g(\boldsymbol{x}) + \boldsymbol{\lambda}^{\mathsf{T}} \boldsymbol{r}(\boldsymbol{x})$$

- K.K.T conditions for optimality:
  - Primal feasibility: $\boldsymbol{r}(\boldsymbol{x}) \geq \boldsymbol{0}$
  - Stationarity: $\nabla g(\boldsymbol{x}) - \boldsymbol{\lambda}^{\mathsf{T}} \nabla \boldsymbol{r}(\boldsymbol{x}) = 0$
  - Dual feasibility: $\boldsymbol{\lambda} \geq \boldsymbol{0}$
  - Complementarity: $\boldsymbol{\lambda}^{\mathsf{T}} \boldsymbol{r}(\boldsymbol{x}) = 0$
- Very complicated…

K.K.T systems:

$$\begin{bmatrix} \frac{\partial^2 \mathcal{L}}{\partial \boldsymbol{x}^2} & \frac{\partial \boldsymbol{r}^{\mathsf{T}}}{\partial \boldsymbol{x}} \\ \frac{\partial \boldsymbol{r}^{\mathsf{T}}}{\partial \boldsymbol{x}} & 0 \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{x} \\ \Delta \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) \\ -\boldsymbol{r}(\boldsymbol{x}) \end{bmatrix}$$

Solution 2: Solve LQR via Quadratic Programming

- Define $\boldsymbol{z} = \begin{bmatrix} u_1 \\ x_2 \\ u_2 \\ x_3 \\ \vdots \\ x_N \end{bmatrix}$
- Define $H = \begin{bmatrix} R_1 & & & & 0 \\ & Q_1 & & & \\ & & \ddots & & \\ & & & R_N & \\ 0 & & & & Q_N \end{bmatrix}$
- The cost functions can be re-written:

$$\frac{1}{2}\sum_{k=1}^{N-1} x_k{}^{\mathsf{T}} Q x_k + u_k{}^{\mathsf{T}} R u_k + x_N{}^{\mathsf{T}} P x_N = \frac{1}{2}\boldsymbol{z}^{\mathsf{T}} H \boldsymbol{z}$$

Solution 2: Solve LQR via Quadratic Programming

- Define $\boldsymbol{z} = \begin{bmatrix} u_1 \\ x_1 \\ u_2 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$
- Define $H = \begin{bmatrix} R_1 & & & & 0 \\ & Q_1 & & & \\ & & \ddots & & \\ & & & R_N & \\ 0 & & & & Q_N \end{bmatrix}$
- The cost functions can be re-written:

$$\frac{1}{2}\sum_{k=1}^{N-1} x_k{}^{\mathsf{T}} Q x_k + u_k{}^{\mathsf{T}} R u_k + x_N{}^{\mathsf{T}} P x_N = \frac{1}{2}\boldsymbol{z}^{\mathsf{T}} H \boldsymbol{z}$$

- The dynamic functions can be re-written:

$$\begin{bmatrix} B_1 & -I & 0 & 0 & \cdots & 0 \\ 0 & A_2 & B_2 & -I & \cdots & 0 \\ & \vdots & & \vdots & & \vdots \\ 0 & 0 & \cdots & A_{N-1} & B_{N-1} & -I \end{bmatrix} \begin{bmatrix} u_1 \\ x_2 \\ u_2 \\ x_3 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} -A_1 x_1 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$C \qquad\qquad z \qquad\qquad d$$

A Special Case: Linear Quadratic Regulator

- The deterministic optimal control problem is re-written as a standard QP:

$$\min_{x,u} \frac{1}{2} \boldsymbol{z}^{\mathsf{T}} H \boldsymbol{z} \quad s.t. C\boldsymbol{z} = \mathbf{d}$$

- The Lagrangian function of this QP is:

$$\mathcal{L}(\boldsymbol{z}, \boldsymbol{\lambda}) = \frac{1}{2} \boldsymbol{z}^{\mathsf{T}} H \boldsymbol{z} + \boldsymbol{\lambda}^{\mathsf{T}} [C\boldsymbol{z} - \mathbf{d}]$$

- K.K.T conditions:

$$\nabla_{\boldsymbol{z}} \mathcal{L}(\boldsymbol{z}, \boldsymbol{\lambda}) = H\boldsymbol{z} + \boldsymbol{\lambda}^{\mathsf{T}} C = 0$$
$$\nabla_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{z}, \boldsymbol{\lambda}) = C\boldsymbol{z} - \boldsymbol{d} = 0$$

$$\Rightarrow \quad \begin{bmatrix} H & C^{\mathsf{T}} \\ C & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{z} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{d} \end{bmatrix}$$

We have the exact solution! But the complexity is too high…

A Closer Look at the LQR as QP

- The K.K.T system for LQR is very sparse (lots of zeros) and has lots of structures

$$\begin{bmatrix} H & C^\top \\ C & 0 \end{bmatrix}\begin{bmatrix} \boldsymbol{z} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{d} \end{bmatrix} \Rightarrow \begin{bmatrix} R_1 & & & & & & B_1^\top & & \\ & Q_1 & & & & & -I & A_2^\top & \\ & & R_2 & & & & & B_2^\top & \\ & & & Q_2 & & & & -I & A_3^\top \\ & & & & R_3 & & & & B_3^\top \\ & & & & & Q_3 & & & -I \\ B_1 & -I & & & & & & & \\ & A_2 & B_2 & -I & & & & 0 & \\ & & & A_3 & B_3 & -I & & & \end{bmatrix}\begin{bmatrix} u_1 \\ x_2 \\ u_2 \\ x_3 \\ u_3 \\ x_4 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -A_1 x_1 \\ 0 \\ 0 \end{bmatrix}$$

$$
\begin{bmatrix}
R_1 & & & & & B_1^{\mathsf{T}} & \\
 & Q_1 & & & & -I & A_2^{\mathsf{T}} \\
 & & R_2 & & & & B_2^{\mathsf{T}} \\
 & & & Q_2 & & & -I \\
 & & & & R_3 & & \\
 & & & & & Q_3 & \\
B_1 & -I & & & & & \\
 & A_2 & B_2 & -I & & & 0 \\
 & & & A_3 & B_3 & -I &
\end{bmatrix}
\begin{bmatrix}
u_1 \\ x_2 \\ u_2 \\ x_3 \\ u_3 \\ x_4 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4
\end{bmatrix}
=
\begin{bmatrix}
0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -A_1 x_1 \\ 0 \\ 0
\end{bmatrix}
$$

$$
Q_3 x_4 - \lambda_4 = 0 \Rightarrow \lambda_4 = Q_3 x_4
$$

$$\begin{bmatrix} R_1 & & & & B_1^T & & \\ & Q_1 & & & -I & A_2^T & \\ & & R_2 & & & B_2^T & \\ & & & Q_2 & & -I & A_3^T \\ & & & & R_3 & & B_3^T \\ & & & & & Q_3 & -I \\ B_1 & -I & & & & & \\ & A_2 & B_2 & -I & & 0 & \\ & & & A_3 & B_3 & -I & \end{bmatrix} \begin{bmatrix} u_1 \\ x_2 \\ u_2 \\ x_3 \\ u_3 \\ x_4 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -A_1 x_1 \\ 0 \\ 0 \end{bmatrix}$$

$$Q_3 x_4 - \lambda_4 = 0 \Rightarrow \lambda_4 = Q_3 x_4$$

$$R_3 u_3 + B_3^T \lambda_4 = 0 \Rightarrow R_3 u_3 + B_3^T Q_3 x_4 = 0$$
$$\Rightarrow R_3 u_3 + B_3^T Q_3 (A_3 x_3 + B_3 u_3) = 0$$
$$\Rightarrow u_3 = -\left(R_3 + B_3^T Q_3 B_3\right)^{-1} B_3^T Q_3 A_3 x_3$$

K_3

$$
\begin{bmatrix}
R_1 & & & B_1^{\intercal} & & \\
& Q_1 & & -I & A_2^{\intercal} & \\
& & R_2 & & B_2^{\intercal} & \\
& & & Q_2 & -I & A_3^{\intercal} \\
& & & & R_3 & B_3^{\intercal} \\
& & & & & Q_3 & -I \\
B_1 & -I & & & & \\
& A_2 & B_2 & -I & & 0 \\
& & & A_3 & B_3 & -I
\end{bmatrix}
\begin{bmatrix} u_1 \\ x_2 \\ u_2 \\ x_3 \\ u_3 \\ x_4 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \end{bmatrix}
=
\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -A_1 x_1 \\ 0 \\ 0 \end{bmatrix}
$$

$$Q_3 x_4 - \lambda_4 = 0 \Rightarrow \lambda_4 = Q_3 x_4$$

$$
\begin{aligned}
&R_3 u_3 + B_3^{\intercal} \lambda_4 = 0 \Rightarrow R_3 u_3 + B_3^{\intercal} Q_3 x_4 = 0 \\
&\Rightarrow R_3 u_3 + B_3^{\intercal} Q_3 (A_3 x_3 + B_3 u_3) = 0 \\
&\Rightarrow u_3 = -\underbrace{\left(R_3 + B_3^{\intercal} Q_3 B_3\right)^{-1} B_3^{\intercal} Q_3 A_3}_{K_3} x_3
\end{aligned}
$$

$$
\begin{aligned}
&Q_2 x_3 - \lambda_3 + A_3^{\intercal} \lambda_4 = 0 \\
&\Rightarrow Q_2 x_3 - \lambda_3 + A_3^{\intercal} Q_3 x_4 = 0 \\
&\Rightarrow Q_2 x_3 - \lambda_3 + A_3^{\intercal} Q_3 (A_3 x_3 + B_3 u_3) = 0 \\
&\Rightarrow Q_2 x_3 - \lambda_3 + A_3^{\intercal} Q_3 (A_3 x_3 - B_3 K_3 x_3) = 0 \\
&\Rightarrow \lambda_3 = \underbrace{\left(Q_2 + A_3^{\intercal} Q_3 (A_3 - B_3 K_3)\right)}_{P_3} x_3
\end{aligned}
$$

Recursive Solutions

- Ricatti equation / recursion:

$$P_N = Q_N$$

$$K_n = -\left(R_n + B_n{}^{\intercal} P_{n+1} B_n\right)^{-1} B_n{}^{\intercal} P_{n+1} A_n$$

$$P_n = Q_{n-1} + A_n{}^{\intercal} P_{n+1} (A_n - B_n K_n)$$

- Optimal u and λ are:

$$u_n = -K_n x_n$$

$$\lambda_n = P_n x_n$$

- Insights: The QP of LQR can be solved by a backward Ricatti pass, followed by a forward rollout to computer $x_{1:N}$ and $u_{1:N}$

Controllability

- When will LQR work?
- Let's check the final state:

For time-invariant case: $A_1 = \cdots = A_N$ and $B_1 = \cdots = B_N$

$$x_N = A_{N-1}x_{N-1} + B_{N-1}u_{N-1} \Rightarrow x_N = Ax_{N-1} + Bu_{N-1}$$

$$\Rightarrow x_N = A(Ax_{N-2} + Bu_{N-2}) + Bu_{N-1}$$

$$\vdots$$

$$\Rightarrow x_N = A^{N-1}x_1 + A^{N-2}Bu_1 + A^{N-3}Bu_2 + \cdots + Bu_{N-1}$$

$$\Rightarrow x_N = \begin{bmatrix} B & AB & A^2B & \cdots & A^{N-2}B \end{bmatrix} \begin{bmatrix} u_{N-1} \\ u_{N-2} \\ u_{N-3} \\ \vdots \\ u_1 \end{bmatrix} + A^{N-1}x_1$$

Controllability

- When will LQR work?
- Let's check the final state:

$$x_N = \begin{bmatrix} B & AB & A^2B & \cdots & A^{N-2}B \end{bmatrix} \begin{bmatrix} u_{N-1} \\ u_{N-2} \\ u_{N-3} \\ \vdots \\ u_1 \end{bmatrix} + A^{N-1}x_1$$

$$\Rightarrow x_N = M\boldsymbol{u} + A^{N-1}x_1 \Rightarrow \boldsymbol{u} = M^\intercal(MM^\intercal)^{-1}(x_N - A^{N-1}x_1)$$

- The least-square solution exists when $MM^\intercal$ is invertible: $\mathbf{rank}(C) = \mathbf{dim}(x) = d$

Controllability

- When will LQR work?
- Let's check the final state:

$$x_N = [B \quad AB \quad A^2B \quad \cdots \quad A^{N-2}B] \begin{bmatrix} u_{N-1} \\ u_{N-2} \\ u_{N-3} \\ \vdots \\ u_1 \end{bmatrix} + A^{N-1}x_1$$

$$\Rightarrow x_N = M\boldsymbol{u} + A^{N-1}x_1 \Rightarrow \boldsymbol{u} = M^\intercal(MM^\intercal)^{-1}(x_N - A^{N-1}x_1)$$

- The least-square solution exists when $MM^\intercal$ is invertible: $\mathbf{rank}(C) = \mathbf{dim}(x) = d$
- Computation can be more efficient: from Cayley-Hamilton's theorem, A^N is a linear combination of lower power of A up to a power of d: $A^N = \sum_{k=0}^{d-1} \alpha_k A^k$
- We only need the controllability matrix: $C=[B \quad AB \quad A^2B \quad \cdots \quad A^{d-1}B]$

Flying Helicopters by Solving Linear Quadratic Problems

https://people.eecs.berkeley.edu/~pabbeel/autonomous_helicopter.html

LQR Extension 1: Affine Systems

- Affine system:

$$\min_{x,u} \sum_{k=0}^{N-1} x_k{}^{\mathsf{T}} Q x_k + u_k{}^{\mathsf{T}} R u_k + x_N{}^{\mathsf{T}} P x_N, \qquad s.t. \, x_0 = \hat{x}_0, x_{k+1} = A x_k + B u_k + c$$

- Solution: Re-define the state as $z_k = \begin{bmatrix} x_k \\ 1 \end{bmatrix}$

$$z_{k+1} = \begin{bmatrix} x_{k+1} \\ 1 \end{bmatrix} = \begin{bmatrix} A x_k + B u_k + c \\ 1 \end{bmatrix} = \begin{bmatrix} A & c \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ 1 \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u_k = A' z_k + B' u_k$$

Slide adapted from P. Abbeel

LQR Extension 2: Stochastic Systems

- Stochastic system: optimize the expectations of the cost-to-go function

$$\min_{x,u} E\left[\sum_{k=0}^{N-1} x_k{}^{\intercal} Q x_k + u_k{}^{\intercal} R u_k + x_N{}^{\intercal} P x_N\right],$$
$$s.t.\, x_0 = \hat{x}_0, x_{k+1} = A x_k + B u_k + \epsilon_k, \epsilon_k \sim N(0, \Sigma)$$

- Rewrite the cost-to-go function

$$J_{k-1}(x) = \min_u x^{\intercal} Q x + u^{\intercal} R u + E[J_k(Ax + Bu + \epsilon)]$$

- See the solution of stochastic optimal control by Laurent Lessard

Slide adapted from P. Abbeel

LQR Extension 3: Penalize for Change in Control Inputs

- Standard LQR:

$$\min_{x,u} \sum_{k=0}^{N-1} x_k{}^{\intercal} Q x_k + u_k{}^{\intercal} R u_k + x_N{}^{\intercal} P x_N \,, \qquad s.t. \, x_0 = \hat{x}_0, x_{k+1} = A x_k + B u_k$$

- On real systems, there's always noise (e.g. a noisy transition function). Without modeling the noise, we end up obtaining high-frequency control inputs.

- Solution: Include change in controls Δu

$$\begin{bmatrix} x_{k+1} \\ u_k \end{bmatrix} = \begin{bmatrix} A & B \\ 0 & I \end{bmatrix} \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix} + \begin{bmatrix} B \\ I \end{bmatrix} \Delta u \Rightarrow z_{k+1} = A' z_k + B' \Delta u$$

Slide adapted from P. Abbeel

LQR Extension 3: Penalize for Change in Control Inputs

- Solution: Include change in controls Δu

$$\begin{bmatrix} x_{k+1} \\ u_k \end{bmatrix} = \begin{bmatrix} A & B \\ 0 & I \end{bmatrix} \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix} + \begin{bmatrix} B \\ I \end{bmatrix} \Delta u \Rightarrow z_{k+1} = A' z_k + B' \Delta u$$

- Re-formulated LQR:

$$\min_{x,u} \sum_{k=0}^{N-1} z_k{}^{\intercal} Q' z_k + \Delta u^{\intercal} R' \Delta u + z_N{}^{\intercal} P' z_N \,, \qquad s.t. \, z_0 = \begin{bmatrix} \bar{x}_0 \\ \bar{u}_0 \end{bmatrix}, z_{k+1} = A' z_k + B' \Delta u$$

with $Q' = \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix}$ and R' = penalty for change in controls

Slide adapted from P. Abbeel

LQR Extension 4: Linear Time Varying (LTV) System

- LTV system:

$$\min_{x,u} \sum_{k=0}^{N-1} x_k{}^{\mathsf{T}} Q_k x_k + u_k{}^{\mathsf{T}} R_k u_k + x_N{}^{\mathsf{T}} P x_N \,, \qquad s.t.\, x_0 = \hat{x}_0, x_{k+1} = A_k x_k + B_k u_k$$

- Algorithm:

Set $P_N = P$
For $\mathrm{i} = N - 1 \ldots 0$

$$K_i = -(R_i + B_i{}^{\mathsf{T}} P_{i+1} B_i)^{-1} B_i{}^{\mathsf{T}} P_{i+1} A_i$$
$$P_i = Q_i + K_i{}^{\mathsf{T}} R_i K_i + (A_i + B_i K_i)^{\mathsf{T}} P_{i+1} (A_i + B_i K_i)$$

we have

$$u_i = K_i x$$
$$J_i(x) = x^{\mathsf{T}} P_i x$$

This is an uncommon case in real systems. But we can generalize the idea to solve non-linear systems!

Slide adapted from P. Abbeel

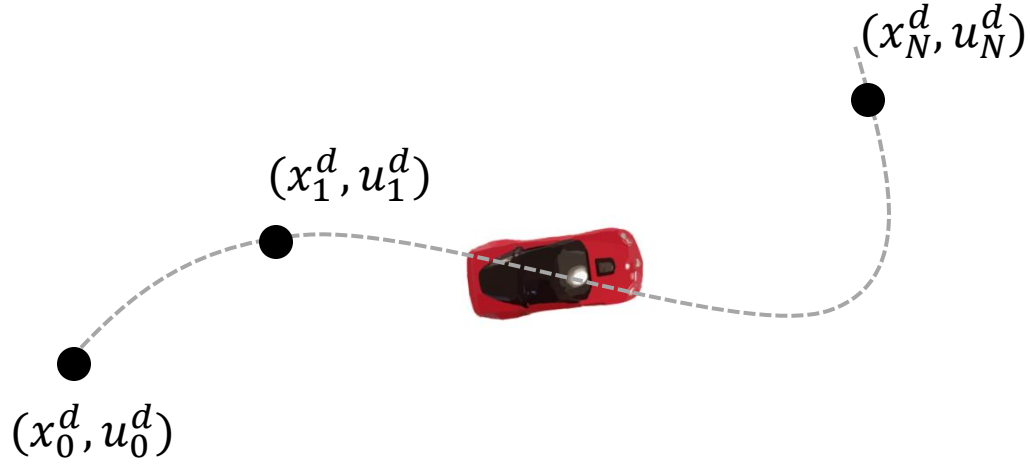
LQR Extension 5: Trajectory Following for Non-Linear System

- Suppose we have a non-linear system:

$$x_{k+1} = f(x_k, u_k)$$

- A state-control trajectory$(x_0^d, u_0^d, x_1^d, u_1^d, \ldots, x_N^d, u_N^d)$ is feasible if and only if

$$x_{k+1}^d = f(x_k^d, u_k^d)$$



LQR Extension 5: Trajectory Following for Non-Linear System

- Suppose we have a non-linear system:

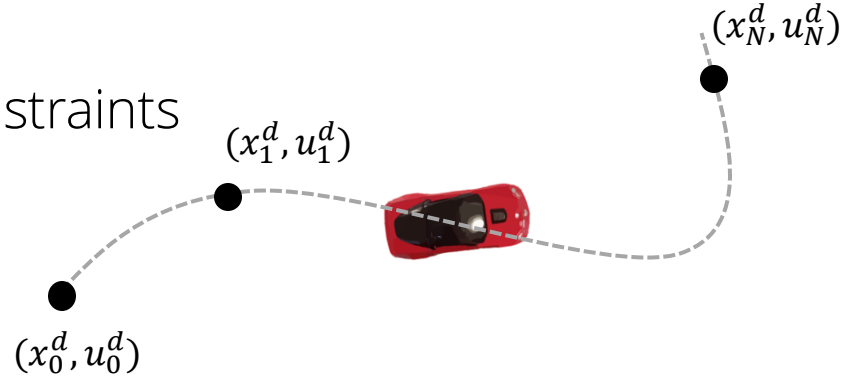
$$x_{k+1} = f(x_k, u_k)$$

- A state-control trajectory$(x_0^d, u_0^d, x_1^d, u_1^d, \dots, x_N^d, u_N^d)$ is feasible if and only if

$$x_{k+1}^d = f(x_k^d, u_k^d)$$

- In real systems, the given trajectory might be infeasible, because of modeling error/noise

- How can we regenerate the control inputs with which the executed trajectory best follows a feasible trajectory?
  - ➢ Regenerate more optimal control trajectories
  - ➢ Regenerate trajectories that follow additional constraints



LQR Extension 5: Trajectory Following for Non-Linear System

- Trajectory following: given a feasible trajectory$(x_0^d, u_0^d, x_1^d, u_1^d, \dots, x_N^d, u_N^d)$

$$\min_{x,u} \sum_{k=0}^{N-1} (x_k - x_k^d)^\intercal Q (x_k - x_k^d) + (u_k - u_k^d)^\intercal R (u_k - u_k^d) + (x_N - x_N^d)^\intercal P (x_N - x_N^d),$$
$$s.t., x_{k+1} = f(x_k, u_k)$$

LQR Extension 5: Trajectory Following for Non-Linear System

- Trajectory following: given a feasible trajectory $(x_0^d, u_0^d, x_1^d, u_1^d, \dots, x_N^d, u_N^d)$

$$\min_{x,u} \sum_{k=0}^{N-1} (x_k - x_k^d)^\intercal Q (x_k - x_k^d) + (u_k - u_k^d)^\intercal R (u_k - u_k^d) + (x_N - x_N^d)^\intercal P (x_N - x_N^d),$$
$$s.t., x_{k+1} = f(x_k, u_k)$$

- Idea: transform it into a Linear Time Varying problem

$$x_{k+1} = f(x_k, u_k) \approx f(x_k^d, u_k^d) + \frac{\partial f}{\partial x}(x_k^d, u_k^d)(x_k - x_k^d) + \frac{\partial f}{\partial u}(x_k^d, u_k^d)(u_k - u_k^d)$$

Taylor expansion

$$\approx x_{k+1}^d + \underbrace{\frac{\partial f}{\partial x}(x_k^d, u_k^d)}_{A_k}(x_k - x_k^d) + \underbrace{\frac{\partial f}{\partial u}(x_k^d, u_k^d)}_{B_k}(u_k - u_k^d)$$

$$\Rightarrow x_{k+1} - x_{k+1}^d \approx \frac{\partial f}{\partial x}(x_k^d, u_k^d)(x_k - x_k^d) + \frac{\partial f}{\partial u}(x_k^d, u_k^d)(u_k - u_k^d)$$

Slide adapted from P. Abbeel

LQR Extension 5: Trajectory Following for Non-Linear System

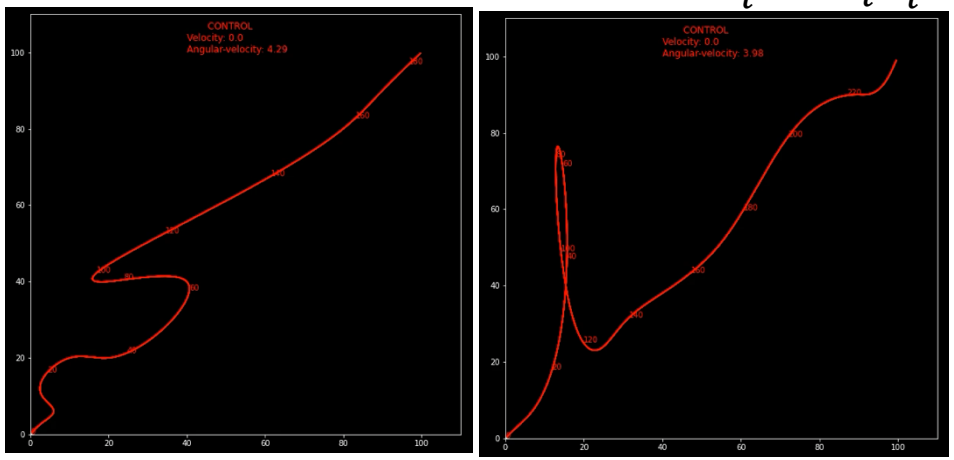
- Trajectory following: given a feasible trajectory$(x_0^d, u_0^d, x_1^d, u_1^d, \dots, x_N^d, u_N^d)$

$$\min_{x,u} \sum_{k=0}^{N-1} {x'_k}^{\mathsf{T}} Q x'_k + {u'_k}^{\mathsf{T}} R u'_k + {x'_N}^{\mathsf{T}} P x'_N, \qquad s.t., x'_{k+1} = A_k x'_k + B_k u'_k$$

$$\text{with } x'_k = x_k - x_k^d \text{ and } u'_k = u_k - u_k^d$$

- Solve it as a standard LQR problem

$$u'_i = K_i x'_i \Rightarrow u_k = u_k^d + K_i(x_i - x_i^d)$$



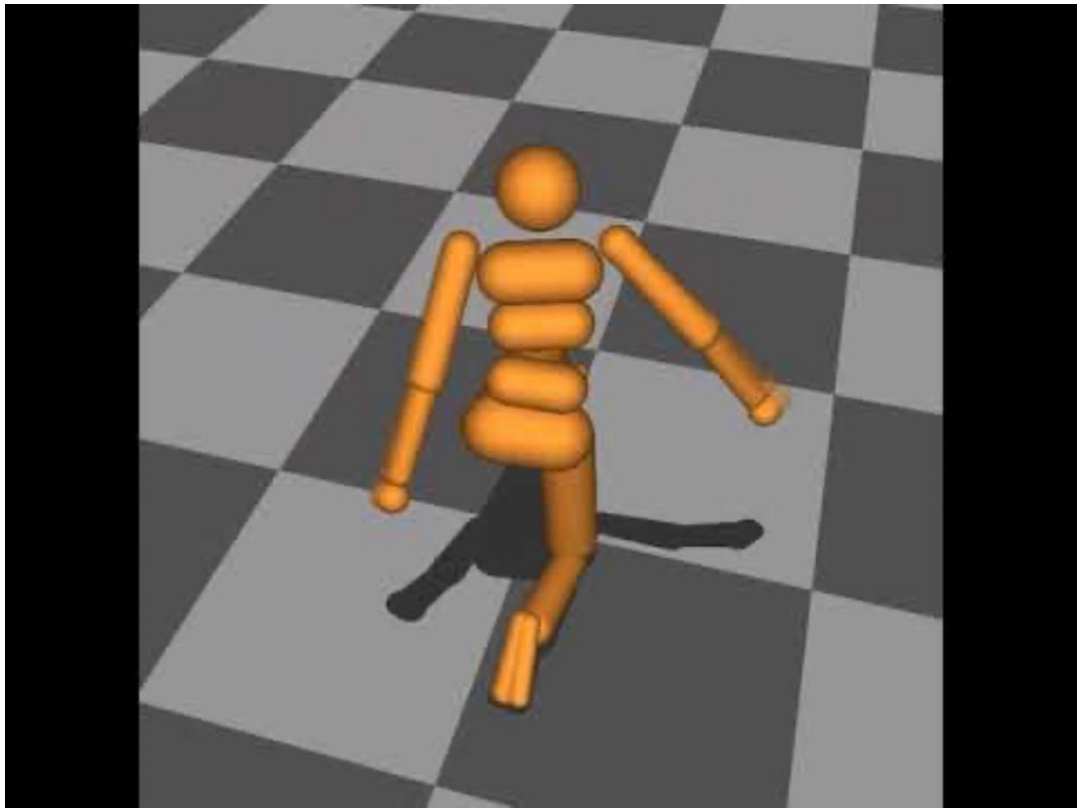
https://kowshikchilamkurthy.medium.com/rl-vs-optimal-control-lqr-for-trajectory-tracking-with-python-code-105ce44e1761

Slide adapted from P. Abbeel

Optimal Control for Non-Linear Costs and Non-Linear Dynamics

- What if we want to solve control problems with non-linear cost functions and non-linear dynamics?

$$\min_{x,u} \sum_{k=0}^{N-1} c(x_k, u_k), \qquad s.t. x_0 = \hat{x}_0, x_{k+1} = f(x_k, u_k)$$



https://www.youtube.com/watch?v=anIsw2-Lbco

Iterative Linear Quadratic Regulator (iLQR)

- What if we want to solve control problems with non-linear cost functions and non-linear dynamics?

$$\min_{x,u} \sum_{k=0}^{N-1} c(x_k, u_k), \qquad s.t. x_0 = \hat{x}_0, x_{k+1} = f(x_k, u_k)$$

- Idea: First guess and then refine
  1. Propose an initial control trajectory. Roll out to obtain a feasible state-control trajectory.
  2. Linearize the dynamics and make the cost function quadratic around the roll out.
  3. Obtain a more optimal control trajectory by solving LQR
  4. Repeat 2.

- A local-optimal feedback control method

Slide adapted from S. Choudhury

The iLQR Algorithm

1. Propose some initial (feasible) trajectory $\{x_t, u_t\}_{t=0}^{T-1}$
2. Linearize the dynamics, f about trajectory:

$$\left.\frac{\partial f}{\partial x}\right|_{x_t} = A_t, \quad \left.\frac{\partial f}{\partial u}\right|_{u_t} = B_t$$

   Linearization can be obtained by three methods:

   (a) Analytical: either manually or via *auto-diff*, compute the correct derivatives.

   (b) Numerical: use finite differencing.

   (c) Statistical: Collect samples by deviations around the trajectory and fit linear model.

3. Compute second order Taylor series expansion the cost function $c(x, u)$ around x_t and u_t and get a quadratic approximation $c_t(\tilde{x}_t, \tilde{u}_t) = \tilde{x}_t^\top \tilde{Q}_t \tilde{x}_t + \tilde{u}_t^\top \tilde{R}_t \tilde{u}_t$ where the $\tilde{x}_t, \tilde{u}_t$ variables represent *changes* in the proposed trajectory in homogenous coordinates. [12]
4. Given $\{A_t, B_t, \tilde{Q}_t, \tilde{R}_t\}_{t=0}^{T-1}$, solve an affine quadratic control problem and obtain the proposed feedback matrices (on the homogeneous represenation of x).
5. Forward simulate the full nonlinear model $f(x, u)$ using the computed controls $\{u_t\}_{t=0}^{T-1}$ that arise from feedback matrices applied to the sequence of states $\{x_t\}_{t=0}^{T-1}$ that arise from that forward simulation.
6. Using the newly obtained $\{x_t, u_t\}_{t=0}^{T-1}$ repeat steps from 2.

Check “Synthesis and Stabilization of Complex Behaviors through Online Trajectory Optimization” for more details!

Jia-Wei will give a lecture (in Mandarin) of iLQR on next Wednesday 10/8! Sorry if you can’t understand Mandarin! But we will upload the video and check the translated subtitles on NTU COOL.

Slide adapted from S. Choudhury

What Could Go Wrong?

- The new trajectory might be very different from the previous roll out. Since we **approximate** cost functions and dynamics around the previous roll out, the new trajectory might be in fact sub-optimal…

- Solution: Be conservative! Stay close to the region where linearized/quadricized approximation is still precise

$$c'(x_k, u_k) = (1-\alpha)c(x_k, u_k) + \alpha(\|(x_k - x_k^d\|_2^2 + \|(u_k - u_k^d\|_2^2)$$

with x_k^d, u_k^d as previous state-control roll out

Stay close to the previous roll out!

How to Solve Optimal Control Problems?

- Deterministic Optimal Control Problem:

$$\min_{x,u} \sum_{k=1}^{N-1} c(x_k, u_k) + c_F(x_N)$$

such that

$$x_1 = \hat{x}_1$$

$$x_{k+1} = f(x_k, u_k), \qquad k = 1, \dots, N-1$$

$$0 \geq h(x_k, u_k), \qquad k = 1, \dots, N-1$$

$$0 \geq r(x_N)$$

- Options:
  1. Root finding
  2. Constrained minimization
  3. Dynamic programming
  4. Model predictive control

Model Predictive Control

- Let's think the original formulation of LQR:

$$\min_{x,u} \sum_{k=1}^{N-1} x_k{}^\intercal Q x_k + u_k{}^\intercal R u_k + x_N{}^\intercal P x_N, \qquad s.t. x_1 = \hat{x}_1, x_{k+1} = Ax_k + Bu_k$$

We didn't consider any constraint…

Model Predictive Control

- Let's think the original formulation of LQR:

$$\min_{x,u} \sum_{k=1}^{N-1} x_k{}^{\mathsf{T}} Q x_k + u_k{}^{\mathsf{T}} R u_k + x_N{}^{\mathsf{T}} P x_N, \qquad s.t. x_1 = \hat{x}_1, x_{k+1} = A x_k + B u_k$$

We didn't consider any constraint…

- Solution 0: solve it with 1-step DP but add constraints per step

$$J^*(x_k) = \min_{u_k} c(x_k, u_k) + J^*(x_{k+1})$$

$$\Rightarrow u_k = \arg\min_{u_k} u_k{}^{\mathsf{T}} R u_k + \underbrace{(A x_k + B u_k)^{\mathsf{T}} \mathrm{P}_{\mathrm{k+1}} (A x_k + B u_k)}$$

The solution will be sub-optimal, as this cost-to-go function doesn't consider constraints…

Model Predictive Control

- Solution 1: solve it with multi-step DP and add constraints

$$\min_{u_{k:k+H}, x_{k:k+H}} \sum_{h=0}^{H-1} x_{k+h}^{\intercal} Q x_{k+h} + u_{k+h}^{\intercal} R u_{k+h} + x_{k+H}^{\intercal} P_{k+H} x_{k+H}$$

  where $H \ll N$ is called horizon

- Intuition: explicit constrained optimization over first H steps gets the state close enough to the reference that constraints are no longer active and LQR cost-to-go is valid further into the future

Model Predictive Control

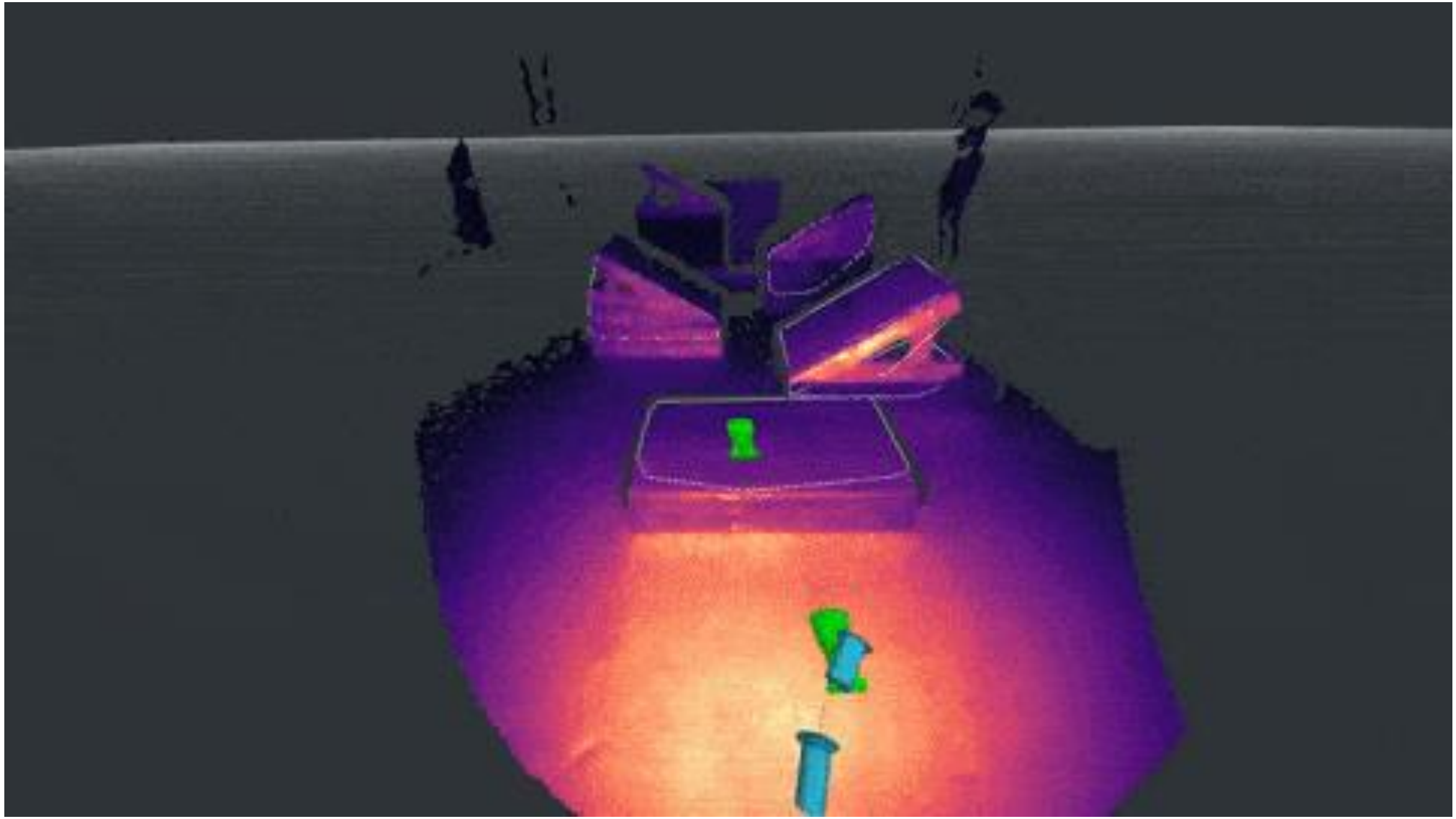
- Given initial state $x_0 = \hat{x}_0$
- For $\mathbf{t = 0,1,2, \dots, N}$

  Solve

$$\min_{x,u} \sum_{k=t}^{H+t-1} c(x_k, u_k) + \tilde{J}(x_{t+H}), \qquad s.t. x_t = \hat{x}_t, x_{k+1} = f(x_k, u_k)$$

  Execute u_t

  Observe resulting state $x_t = \hat{x}_t$



Video from Boston Dynamic
https://bostondynamics.com/blog/flipping-the-script-with-atlas/

More Materials on Optimal Control

- 16-745 Optimal Control at CMU by Zachary Manchester (https://optimalcontrol.ri.cmu.edu/)
- CS 287 Advanced Robotics at UC Berkeley by Pieter Abbeel (https://people.eecs.berkeley.edu/~pabbeel/cs287-fa19/)
- Model Predictive Control and Reinforcement Learning at University of Freiburg by Joschka Boedecker and Moritz Diehl (https://www.syscop.de/teaching/ss2021/model-predictive-control-and-reinforcement-learning)
- 6.8210 Underactuated Robotics at MIT by Russ Tedrake (https://underactuated.csail.mit.edu/Spring2024/)