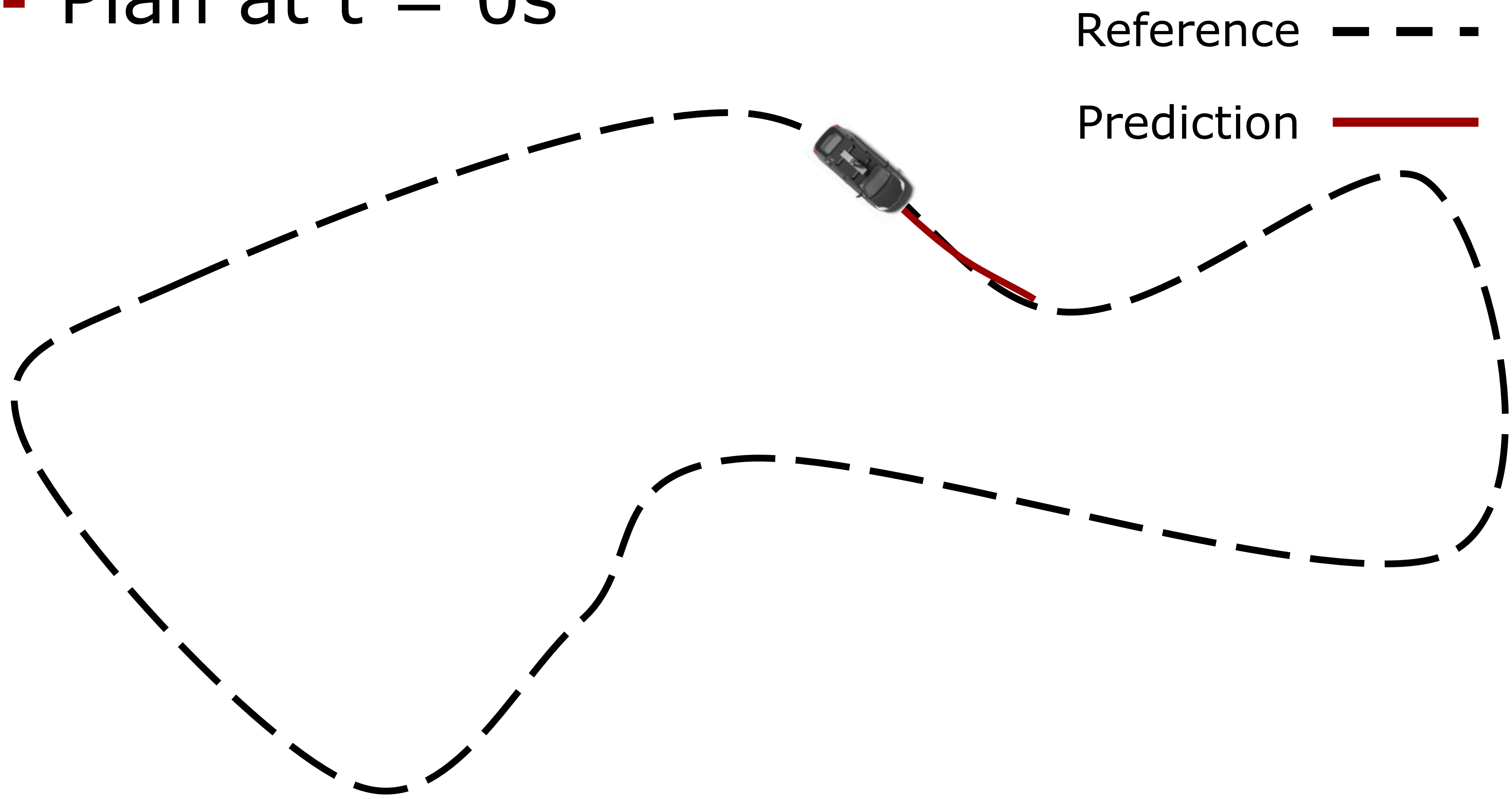
Photogrammetry & Robotics Lab

Numerical Methods for Model Predictive Control

Lasse Peters

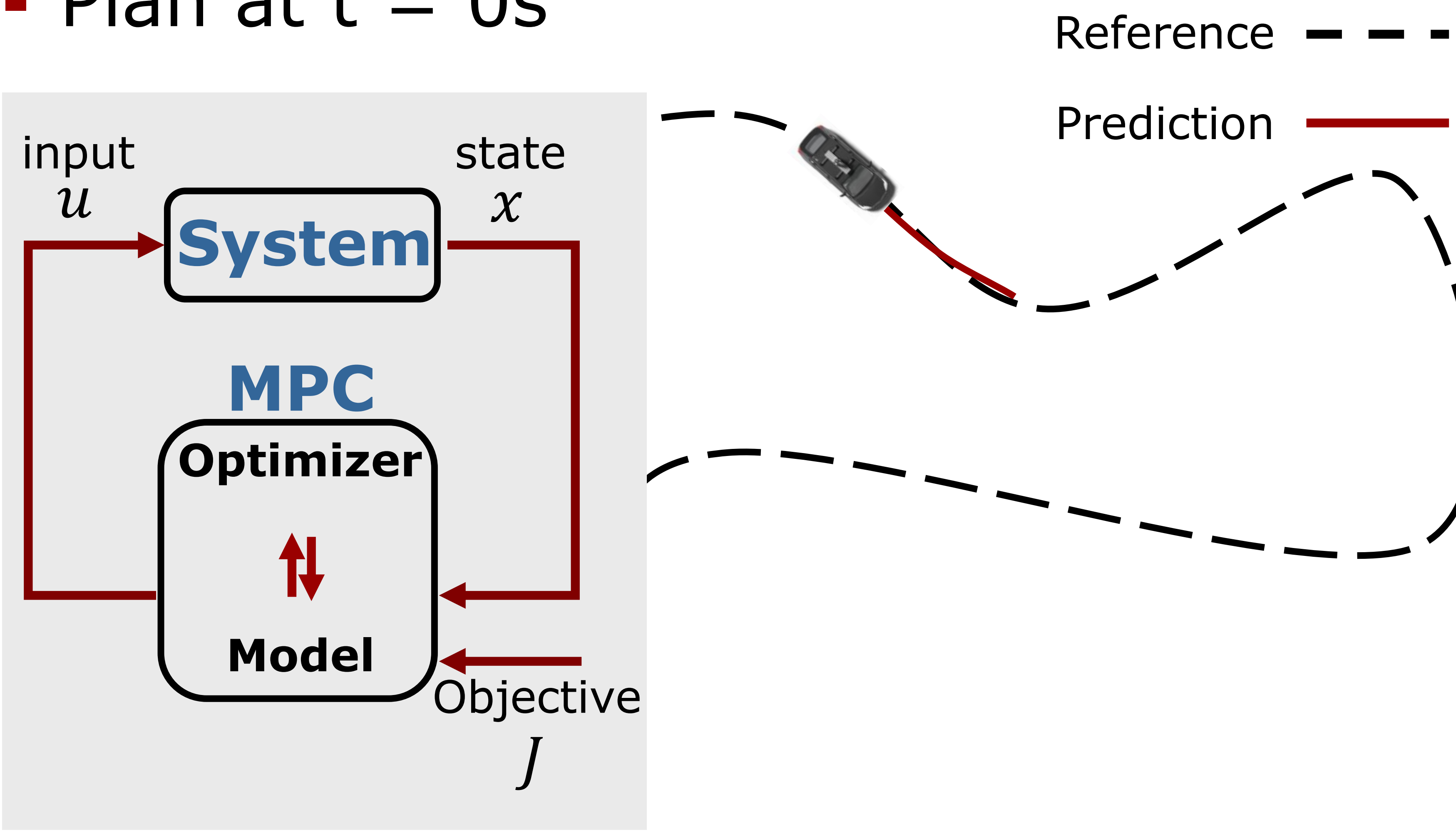
Recap: Model Predictive Control

- Plan at t = 0s



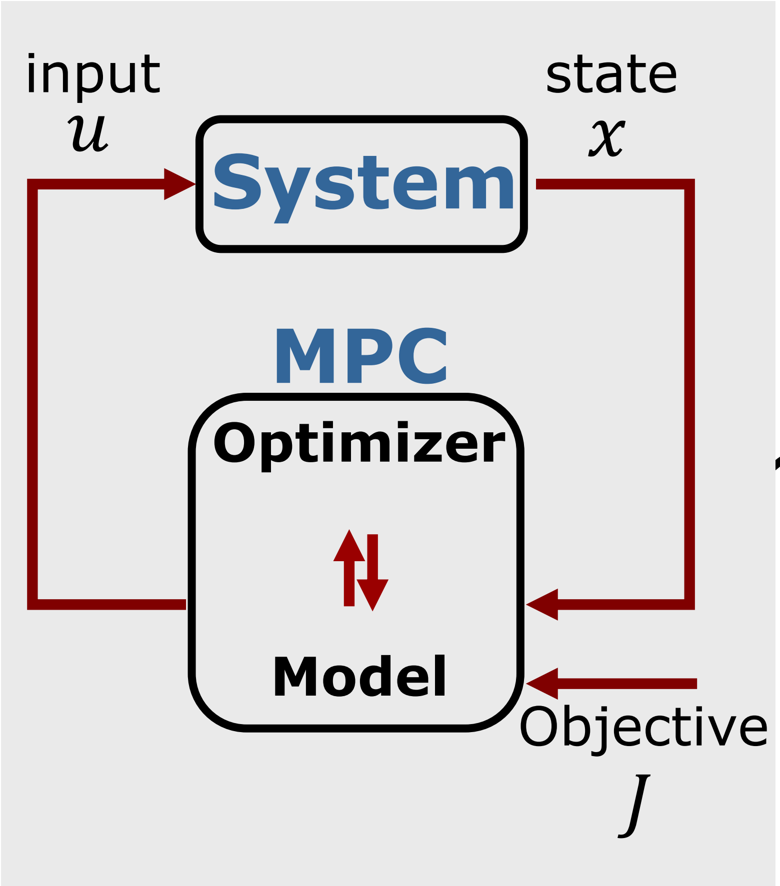
Recap: Model Predictive Control

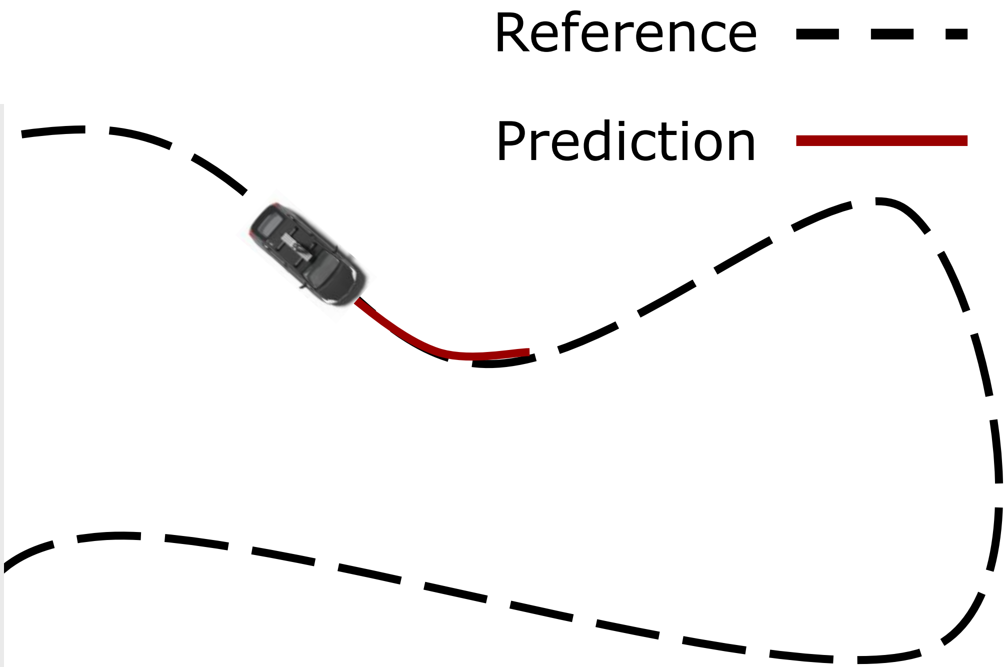
- Plan at t = 0s



Recap: Model Predictive Control

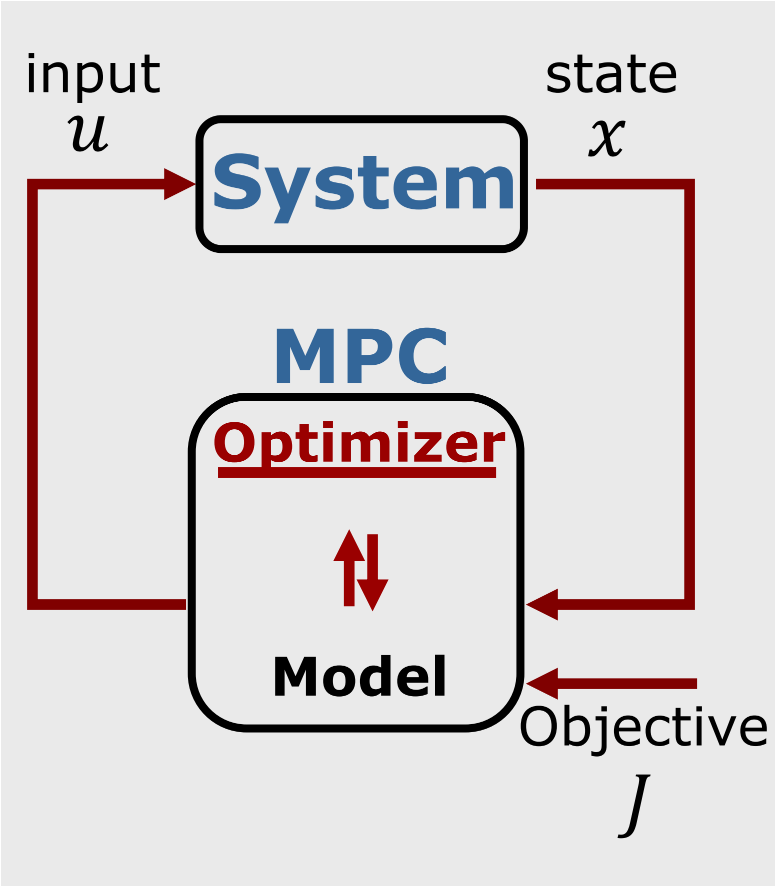
- Plan at t = 1s

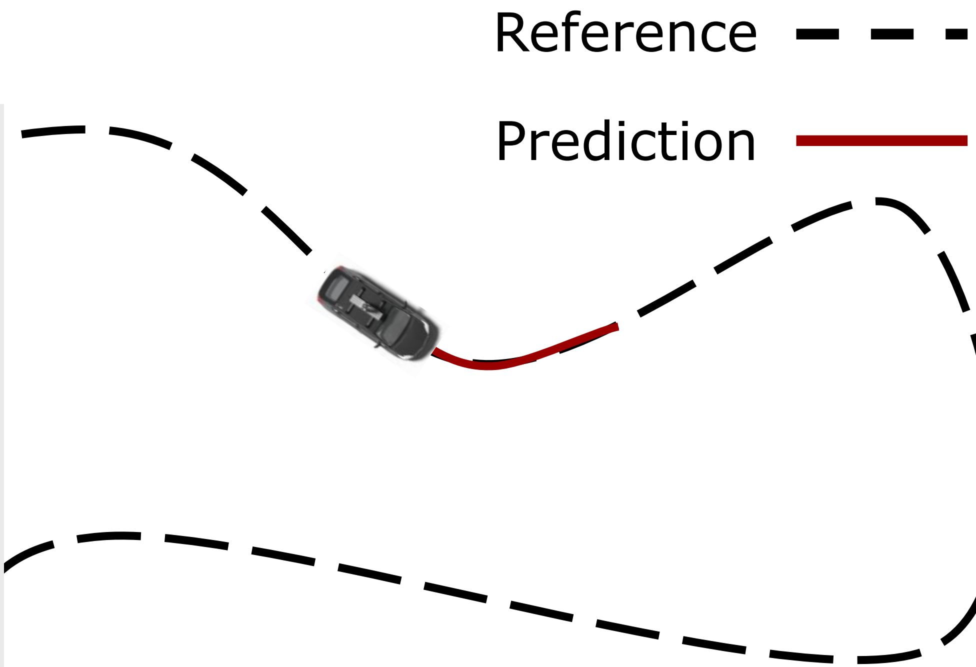




Recap: Model Predictive Control

- Plan at t = 2s





Recap: Control as Optimization

- **minimize the cost** of a …
- **dynamically feasible** trajectory that …
- does **not violate the constraints**.

$$
\begin{aligned}
\min_{x_{1:T}, u_{1:T}} \quad & J(x_{1:T}, u_{1:T}) \\
\text{subject to} \quad & x_{t+1} = f(x_t, u_t), \quad && \forall t \in [T-1] \\
& u_t \in \mathcal{U}_t, && \forall t \in [T] \\
& x_t \in \mathcal{X}_t, && \forall t \in [T] \\
& x_1 = x_{\text{init}}
\end{aligned}
$$

$$
\begin{aligned}
\text{where} \quad & x_{1:T} := (x_1, \ldots, x_T) \\
& u_{1:T} := (u_1, \ldots, u_T)
\end{aligned}
$$

Software Tools for Optimization

- CasADi (C++, Python, Matlab)
- JuMP.jl (Julia)

$$\begin{aligned} \underset{x,y}{\text{minimize}} \quad & (y-x^2)^2 \\ \text{subject to} \quad & x^2+y^2=1 \\ & x+y \geq 1, \end{aligned}$$

```
opti = casadi.Opti()
x, y = opti.variable(), opti.variable()

opti.minimize(  (y-x**2)**2   )
opti.subject_to( x**2+y**2==1 )
opti.subject_to(       x+y>=1 )

opti.solver('ipopt')
sol = opti.solve()
```

Today

How to solve optimal control problems numerically?

Optimal Control Problem (OCP)

- Here, no state and input constraints

$$
\begin{aligned}
\min_{x_{1:T}, u_{1:T}} \quad & J(x_{1:T}, u_{1:T}) \\
\text{subject to} \quad & x_{t+1} = f(x_t, u_t), \quad && \forall t \in [T-1] \\
& \cancel{u_t \in \mathcal{U}_t,} && \cancel{\forall t \in [T]} \\
& \cancel{x_t \in \mathcal{X}_t,} && \cancel{\forall t \in [T]} \\
& x_1 = x_{\text{init}}
\end{aligned}
$$

Optimal Control Problem (OCP)

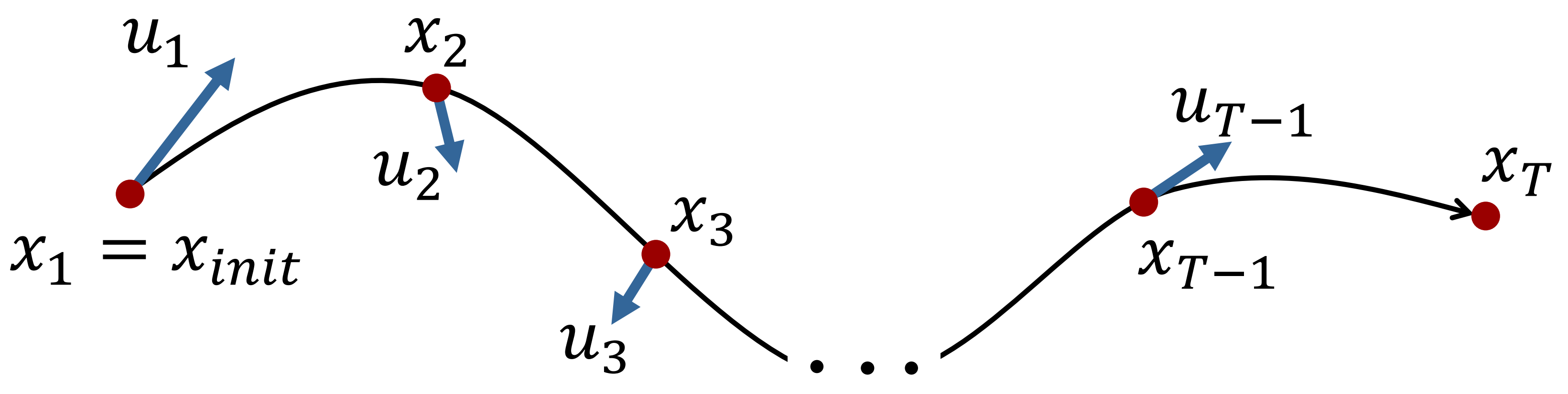
- Here, no state and input constraints

$$
\begin{aligned}
\min_{x_{1:T}, u_{1:T}} \quad & \overbrace{\sum_{t \in [T]} g_t(x_t, u_t)}^{J(x_{1:T}, u_{1:T}) :=} \\
\text{subject to} \quad & x_{t+1} = f(x_t, u_t), \quad \forall t \in [T-1] \\
& x_1 = x_{\text{init}}
\end{aligned}
$$

State Elimination

Recasting the problem as unconstrained optimization.

Forward Simulation



- Each state can be expressed in terms of the initial state and previous inputs

$$x_1 = x_{\text{init}}$$
$$x_2 = f(x_{\text{init}}, u_1)$$
$$x_3 = f(f(x_{\text{init}}, u_1), u_2)$$
$$\dots$$

Forward Simulation

- Let F_t be the function that obtains $x_{t:T}$ from $u_{t:T}$ and x_t via forward simulation

$$F_1(x_{\text{init}}, u_{1:T}) := \left\{ \begin{array}{l} x_1 = x_{\text{init}} \\ x_2 = f(x_{\text{init}}, u_1) \\ x_3 = f(f(x_{\text{init}}, u_1), u_2) \\ \quad \cdots \end{array} \right\} \triangleq x_{1:T}$$

Elimination of State Variables

- Using F_1 we can define an equivalent objective that only depends upon $u_{1:T}$

$$\tilde{J}(u_{1:T}; x_{\text{init}}) := J(F_1(x_{\text{init}}, u_{1:T}), u_{1:T})$$

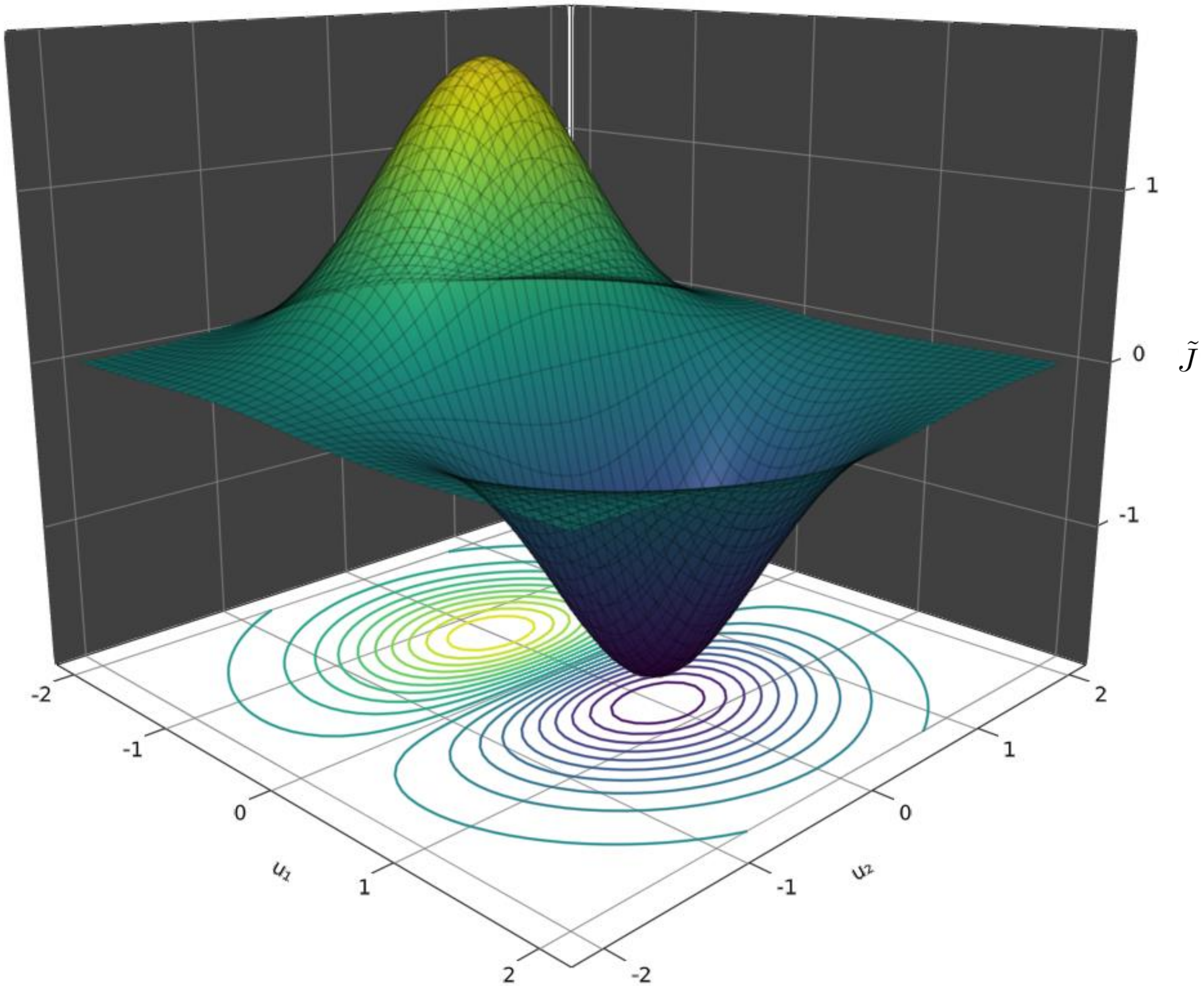
- Unconstrained formulation of the OCP

$$\min_{u_{1:T}} \tilde{J}(u_{1:T}; x_{\text{init}})$$

  - Fewer decision variables: **only controls**
  - Dynamics are implicitly satisfied

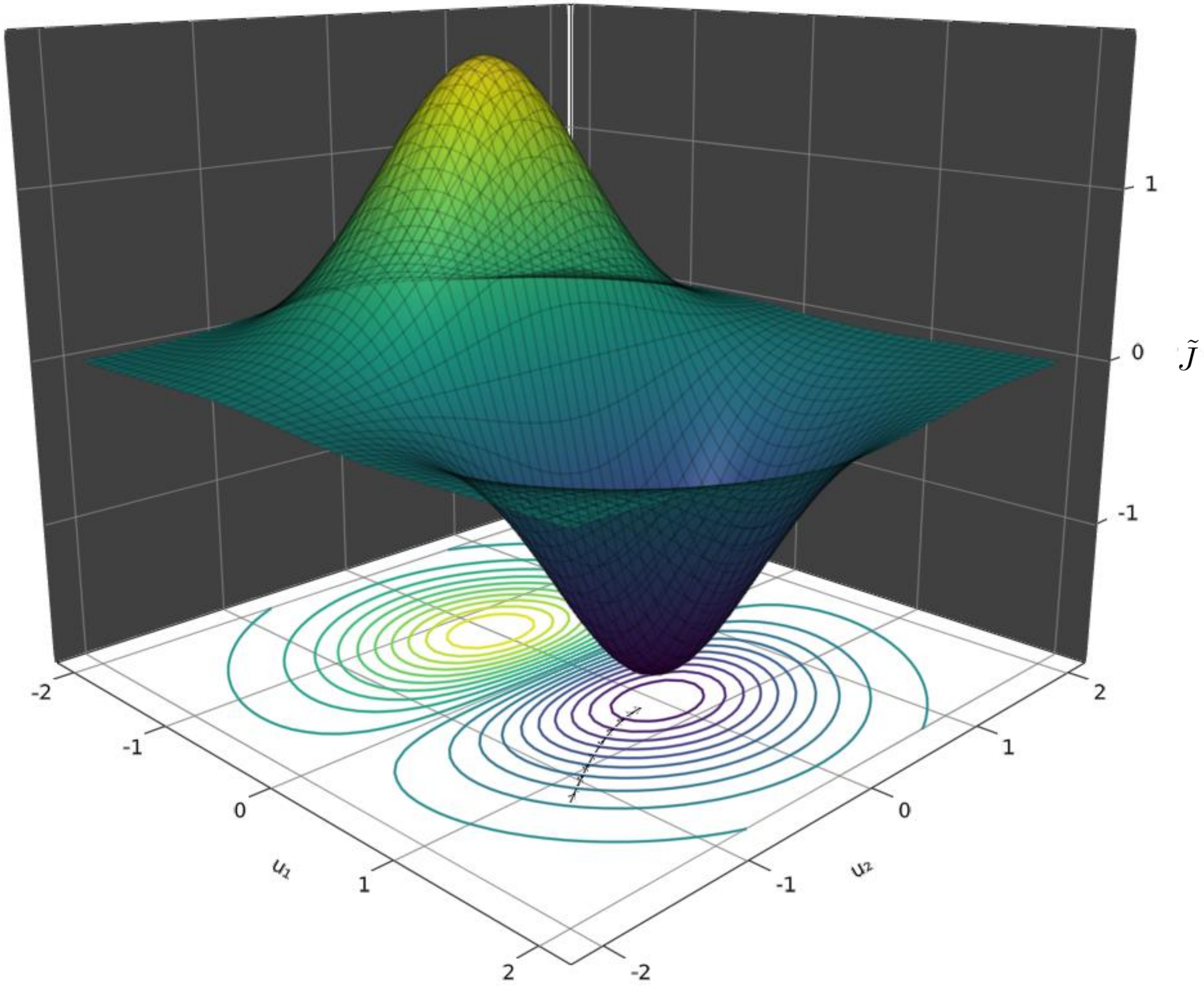
Unconstrained Optimization

- How to minimize the cost $\tilde{J}$?



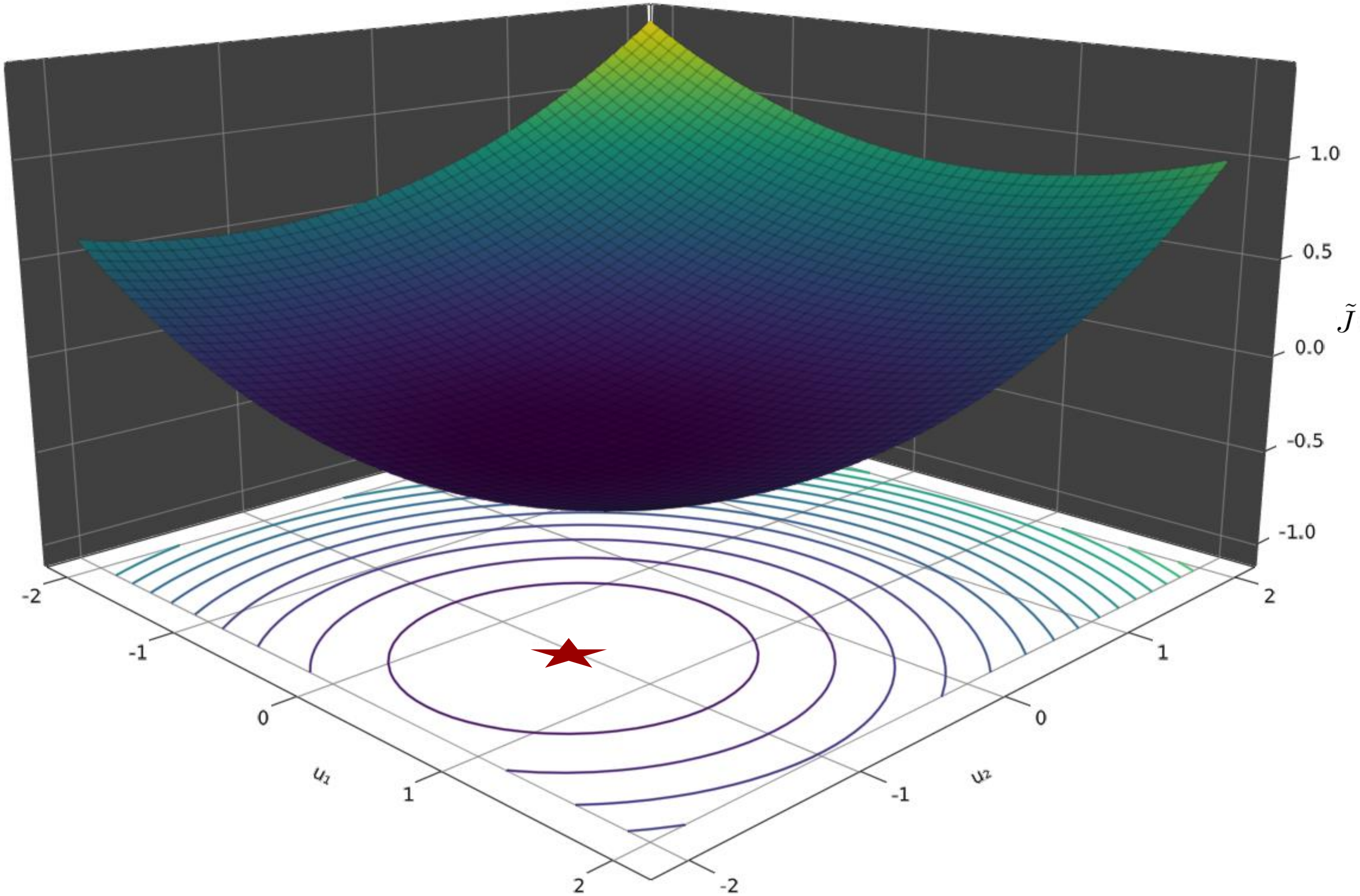
Unconstrained Optimization

- Typically, **iterative approach**



Unconstrained Optimization

- Special case: **convex quadratic cost**



Linear-Quadratic Regulator (LQR)

- LQR is a special OCP with
  - **Linear Dynamics**
  - **Quadratic Costs**

$$
\begin{aligned}
\min_{x_{1:T}, u_{1:T}} \quad & \sum_{t \in [T]} \left[x_t^\top Q x_t + u_t^\top R u_t \right] \\
\text{subject to} \quad & x_{t+1} = A x_t + B u_t, \quad \forall t \in [T-1] \\
& x_1 = x_{\text{init}}
\end{aligned}
$$

$$
\begin{aligned}
\text{where} \quad & Q = Q^\top, Q \succeq 0 \\
& R = R^\top, R \succ 0
\end{aligned}
$$

Batch Solution to LQR

Solving for all controls at once via state elimination and linear least-squares.

LQR: Forward Simulation

- Forward simulation of linear dynamics

$$F_1(x_{\text{init}}, u_{1:T}) = \left[\begin{array}{l} x_1 = x_{\text{init}} \\ x_2 = Ax_{\text{init}} + Bu_1 \\ x_3 = A(Ax_{\text{init}} + Bu_1) + Bu_2 \\ \quad \cdots \\ x_t = A^{t-1}x_{\text{init}} + \sum_{k=1}^{t-1} A^{t-k-1}Bu_k \\ \quad \cdots \end{array}\right.$$

LQR: Forward Simulation

- Forward simulation as matrix equation

$$\underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_T \end{bmatrix}}_{X:=} = \underbrace{\begin{bmatrix} I \\ A \\ \vdots \\ \vdots \\ A^{T-1} \end{bmatrix}}_{\bar{A}:=} x_{\text{init}} + \underbrace{\begin{bmatrix} 0 & \cdots & \cdots & 0 \\ B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \ddots & \ddots & 0 \\ A^{T-2}B & \cdots & AB & B \end{bmatrix}}_{\bar{B}:=} \underbrace{\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_{T-1} \end{bmatrix}}_{U:=}$$

$$X := \bar{A} x_{\text{init}} + \bar{B} U$$

LQR: Reformulation of the Cost

- Defining the stacked cost matrices …

$$\bar{Q} := \mathrm{blockdiag}(Q, \ldots, Q)$$
$$\bar{R} := \mathrm{blockdiag}(R, \ldots, R)$$

- We can compactly write the cost as in terms of stacked states X and inputs U

$$\bar{J}(X, U) := X^\top \bar{Q} X + U^\top \bar{R} U$$

LQR: Elimination of State Variables

- Finally, we can eliminate all state variables by substitution of dynamics:

$$\bar{J}(X,U) := X^\top \bar{Q} X + U^\top \bar{R} U$$
$$X := \bar{A} x_{\text{init}} + \bar{B} U$$

$$\tilde{\bar{J}}(U; x_{\text{init}}) := U^\top H U + 2 x_{\text{init}} F U + C$$

$$\text{where} \quad \begin{aligned} H &:= \bar{B}^\top \bar{Q} \bar{B} + \bar{R}, \\ F &:= \bar{A}^\top \bar{Q} \bar{B}, \\ C &:= x_{\text{init}}^\top \bar{A}^\top \bar{Q} \bar{A} x_{\text{init}} \end{aligned}$$

LQR: Batch Solution

- Batch formulation of LQR

$$\min_U \overbrace{U^\top H U + 2x_{\text{init}} F U + C}^{\tilde{\tilde{J}}(U;x_{\text{init}}) \triangleq}$$

- The cost is **convex quadratic** in U since $H \succ 0 \impliedby R \succ 0$, $Q \succeq 0$
- Minimum: solve for **zero gradient**

$$\partial_U \tilde{\tilde{J}}(U^*; x_{\text{init}}) \triangleq 2HU^* + 2F^\top x_{\text{init}} = 0$$
$$\implies U^* = -H^{-1} F^\top x_{\text{init}}$$

Summary: Batch LQR

- Approach
  - Eliminate all state variables to obtain an unconstrained problem
  - Solve for all controls in one batch

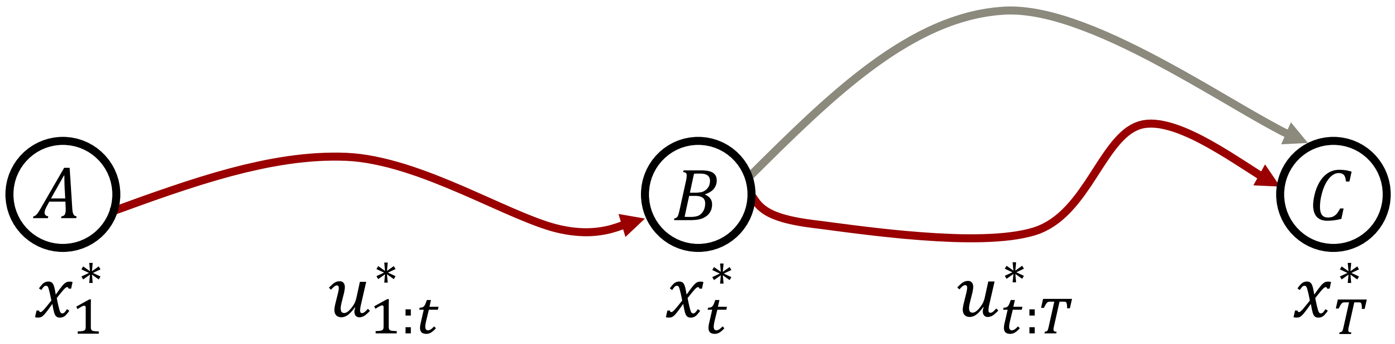
- Comments
  - Can be solved **analytically**
  - Still requires to invert one large $(mT \times mT)$ matrix $H^{-1} \triangleq (\bar{B}^\top \bar{Q} \bar{B} + \bar{R})^{-1}$

Dynamic Programming LQR

Exploiting the sequential structure of the problem.

Principle of Optimality: Example

- If the **optimal plan** from A to C goes through B, then the tail portion of the plan that starts in B and ends in C must also be optimal.



Principle of Optimality

- Let $u^*_{1:T}$ be the solution of an OCP with optimal state trajectory $x^*_{1:T}$. Then the tail sequence $u^*_{t:T}$ solves **tail problem**

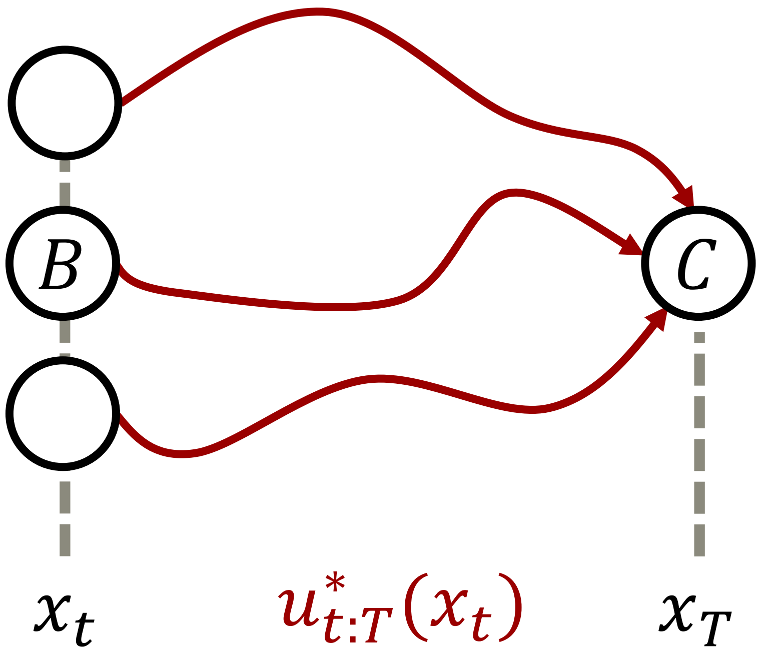
$$
\begin{aligned}
\min_{x_{t:T}, u_{t:T}} \quad & J_t(x_{t:T}, u_{t:T}) \\
\text{subject to} \quad & x_{k+1} = f(x_k, u_k), \forall k \in \{t, \ldots, T-1\} \\
& x_t = x^*_t
\end{aligned}
$$

- Where we define the **cost-to-go**

$$
J_t(x_{t:T}, u_{t:T}) := \sum_{k=t}^{T} g_k(x_k, u_k)
$$

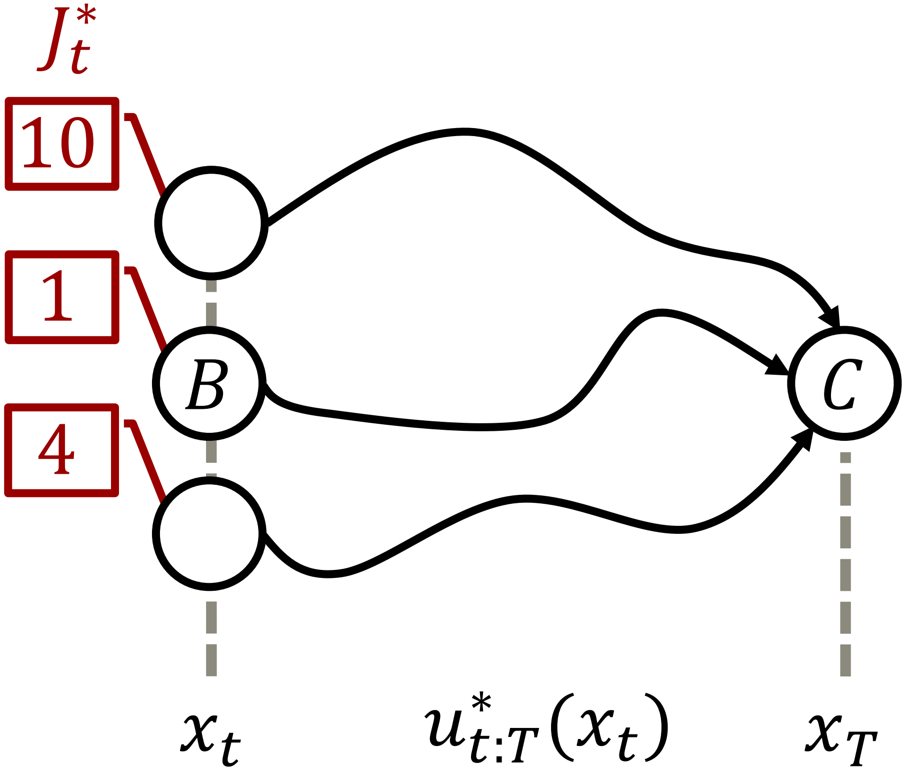
Dynamic Programming: Main Idea

- Break up the problem into smaller tail problems:
  - Optimal **feedback strategy** starting at time t as a **function of state** x_t:



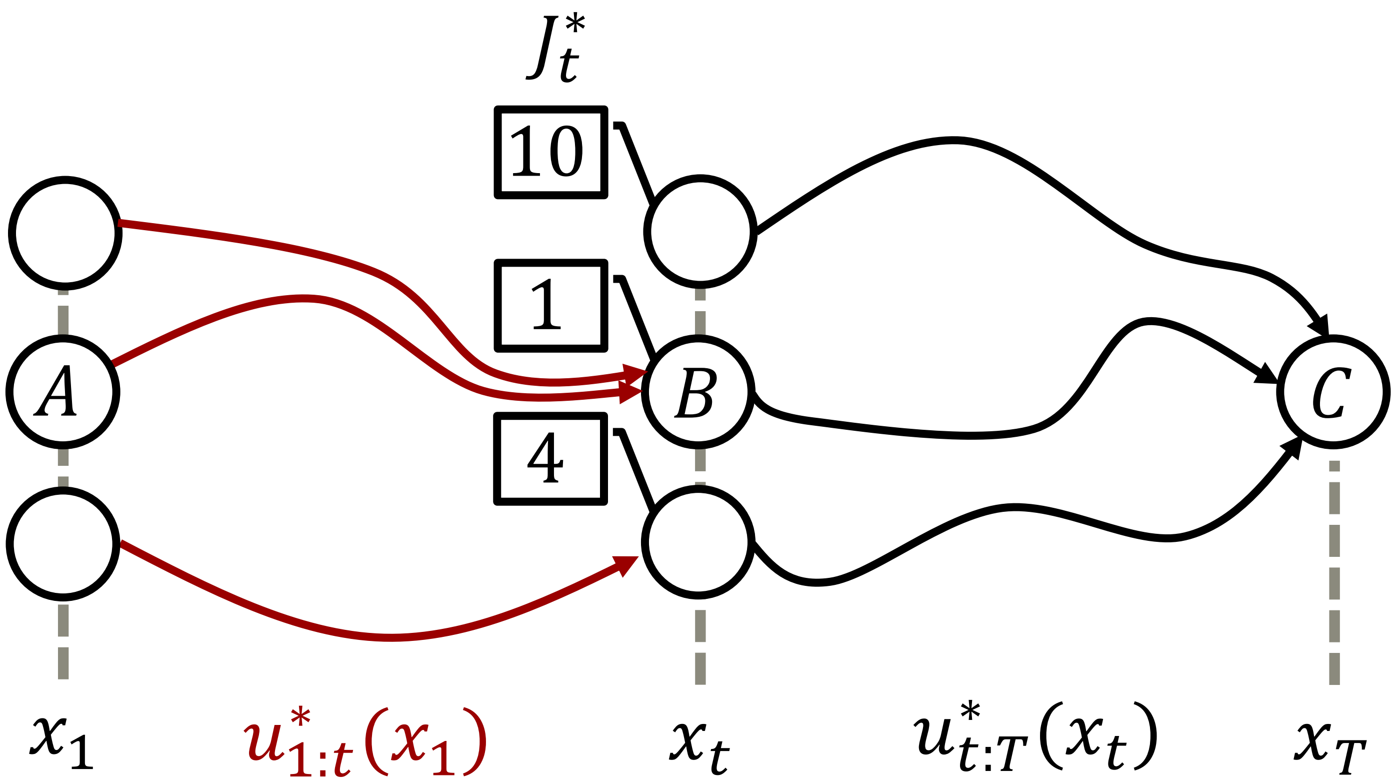
Dynamic Programming: Main Idea

- Also record **optimal cost-to-go**, as a **function of state** x_t



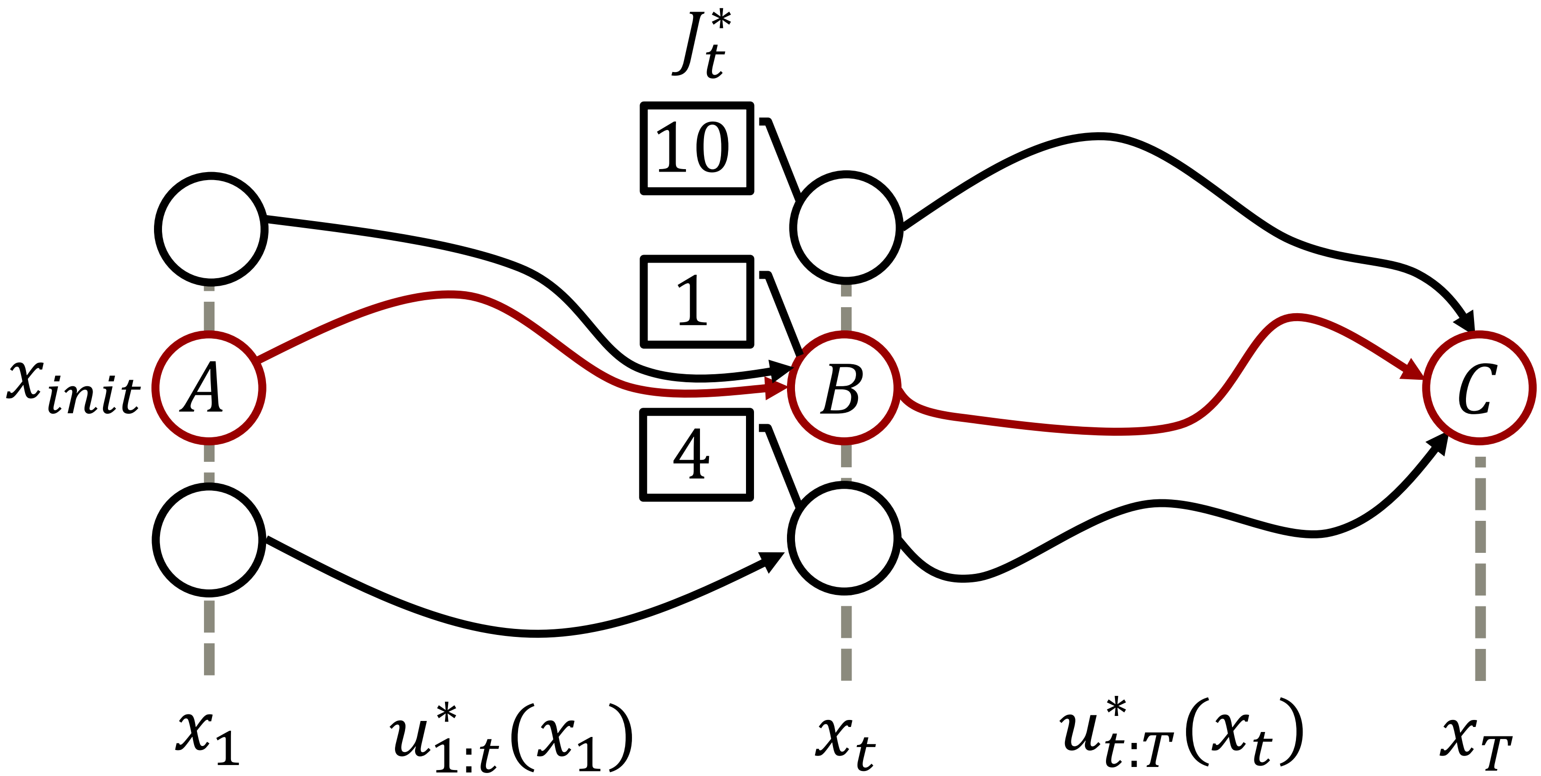
Dynamic Programming: Main Idea

- Repeat this process until $t = 1$
- Minimize: immediate cost + cost-to-go



Dynamic Programming: Main Idea

- Evaluate the **feedback strategy** at the initial state $x_1 = x_{init}$



Dynamic Programing: Comments

- We must be able to represent the **optimal cost-to-go**

$$J_t^*(x_t) := J_t(F_t(x_t, u_{t:T}^*), u_{t:T}^*(x_t))$$

- Naïve approach: discretize the state-space and store $J_t^*(\cdot)$ as a lookup table

- For LQR, we can represent it compactly in **closed form**

Recursive Setup of LQR

$$\min_{x_{1:T}, u_{1:T}} \quad \sum_{t \in [T]} [x_t^\top Q x_t + u_t^\top R u_t]$$

$$\text{subject to} \quad x_{t+1} = Ax_t + Bu_t, \quad \forall t \in [T-1]$$

$$x_1 = x_{\text{init}}$$

- Smallest tail-problem: only final stage

$$u_T^*(x_T) \triangleq \arg\min_{u_T} x_T^\top Q x_T + u_T^\top R u_T \equiv 0$$

$$J_T^*(x_T) \triangleq x_T^\top P_T x_T, \text{where } P_T := Q$$

LQR: One-Step Backup

- Expand the tail sequence by one step

$$u^*_{T-1}(x_{T-1}) \triangleq \arg \min_{x_T, u_{T-1}} \overbrace{g_{T-1}(x_{T-1}, u_{T-1})}^{\text{immediate cost}} + \overbrace{J^*_T(x_T)}^{\text{cost-to-go}}$$

$$\text{subject to} \quad x_T = f(x_{T-1}, u_{T-1})$$

LQR: One-Step Backup

- Eliminate future state and constraint

$$u^*_{T-1}(x_{T-1}) \triangleq \arg\min_{u_{T-1}} \quad \overbrace{x^\top_{T-1} Q x_{T-1} + u^\top_{T-1} R u_{T-1}}^{\text{immediate cost}}$$
$$+ \underbrace{J^*_T(Ax_{T-1} + Bu_{T-1})}_{\text{cost-to-go}}$$

- This one-stage problem has only a **single decision** variable, u_{T-1}, and **no constraints**

LQR: One-Step Backup

- Substitution of the optimal cost-to-go yields a **convex quadratic objective** in u_{T-1} :

$$
\begin{aligned}
J^*_T(x_T) &\triangleq x_T^\top P_T x_T \\
\Longrightarrow u^*_{T-1}(x_{T-1}) &\triangleq \arg\min_{u_{T-1}} \quad x_{T-1}^\top (A^\top P_T A + Q) x_{T-1} \\
&\qquad + u_{T-1}^\top (B^\top P_T B + R) u_{T-1} \\
&\qquad + 2 x_{T-1}^\top A^\top P_T B u_{T-1}
\end{aligned}
$$

LQR: One-Step Backup | Gains

- **Convex** objective in u_{T-1}:
  Find minimum by solving for **zero gradient**

$$2(B^\top P_T B + R)u^*_{T-1} + 2B^\top P_N A x_{T-1} = 0$$

$$\Rightarrow u^*_{T-1}(x_{T-1}) \triangleq F_{T-1} x_{T-1}$$

$$\text{where } F_{T-1} := -(B^\top P_T B + R)^{-1} B^\top P_T A$$

- The resulting **feedback strategy** $u^*_{T-1}(\cdot)$ is **linear** in the state x_{T-1}.

LQR: One-Step Backup | Costs

- Back substitution into J_{T-1} yields the backup of the **optimal cost-to-go**

$$J^*_{T-1}(x_{T-1}) = x_{T-1}^\top P_{T-1} x_{T-1}$$
$$\text{where } P_{T-1} := A^\top P_T A + Q + A^\top P_T B F_{T-1}$$

- The optimal cost-to go remains quadratic and positive semi-definite
$P_{T-1} \succeq 0 \Longleftarrow R \succ 0\,,\, Q \succeq 0$

Dynamic Programming LQR Solution

- Initialize the final stage cost-to-go

$$P_T := Q$$

- **Recursively** compute the optimal feedback strategy backward in time

$$F_{t-1} := -(B^\top P_t B + R)^{-1} B^\top P_t A$$

$$P_{t-1} := A^\top P_t A + Q + A^\top P_t B F_{t-1}$$

- The resulting feedback strategy can be **evaluated at arbitrary states** to recover the optimal control inputs

$$u_t^*(x_t) = F_t x_t$$

$$J_t^*(x_t) = x_t^\top P_t x_t$$

Summary: DP LQR

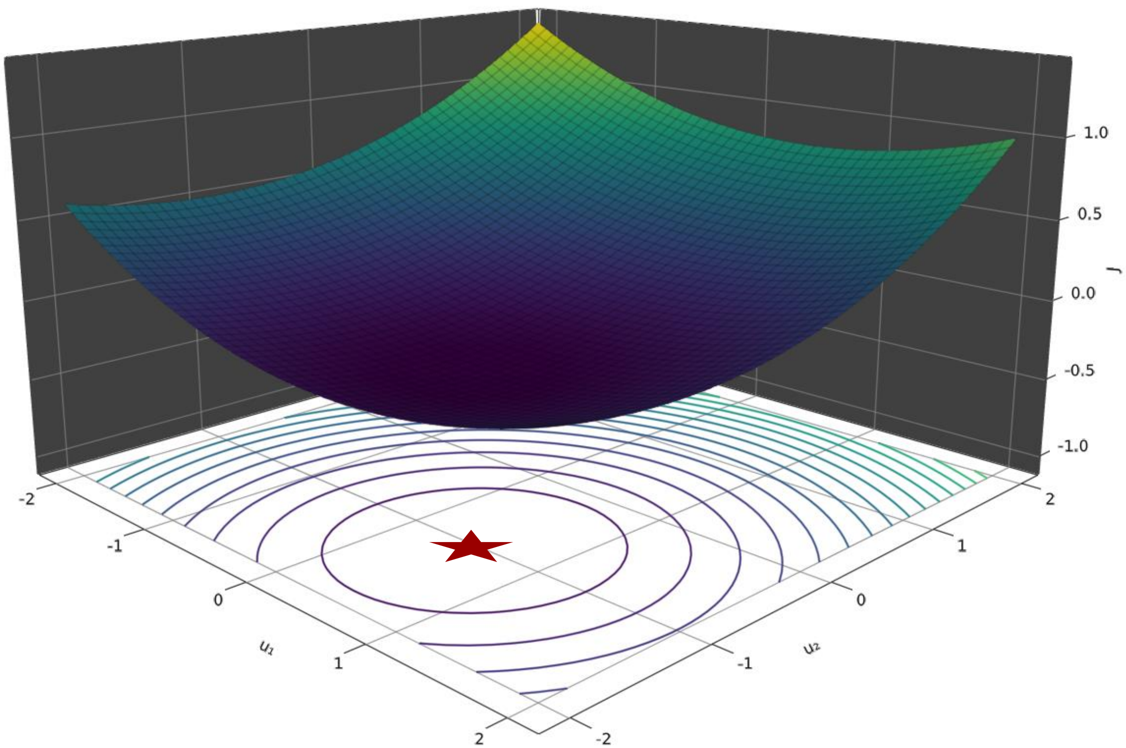
- Approach
  - Decompose LQR into T one-step optimizations
  - Solve for optimal feedback strategy recursively, **one stage at a time**
- Comments; in contrast to batch LQR
  - Instead of inverting one big $(mT \times mT)$ batch matrix, we only need to invert T smaller $(m \times m)$ matrices $\Longrightarrow$ **linear complexity** in horizon T
  - The result is a **feedback strategy** that can be evaluated at arbitrary states, not only at a prefixed initial state $\Longrightarrow$ can compensate errors

Iterative LQR (ILQR)

Solving nonlinear OCPs via iterative linear-quadratic approximations.

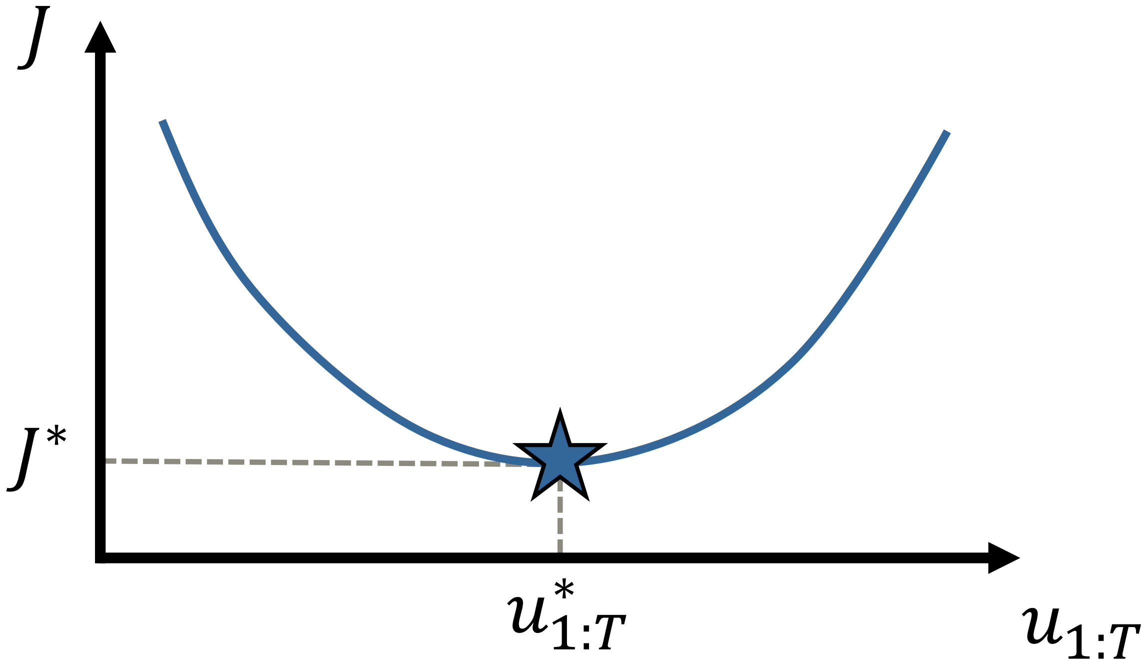
ILQR: Main Idea

- Recall: LQR problems can be solved **analytically**
  - 2D example



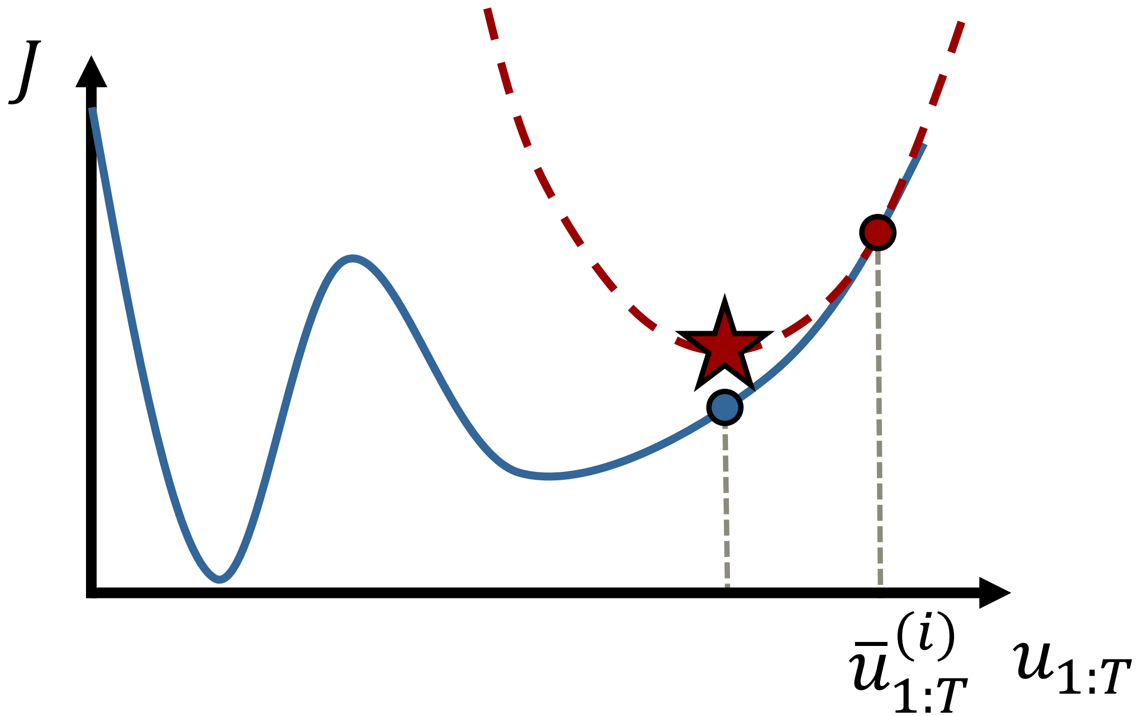
ILQR: Main Idea

- Recall: LQR problems can be solved **analytically**
  - 1D example



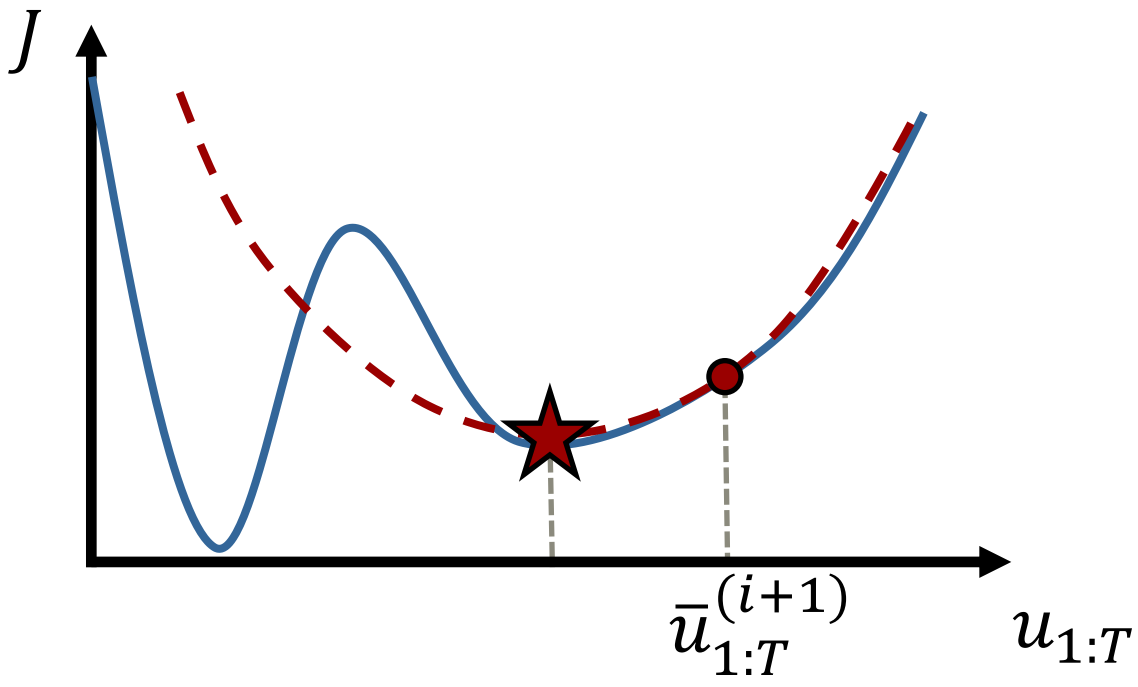
ILQR: Main Idea

- Approximate the solution to a nonlinear OCP via local LQR updates



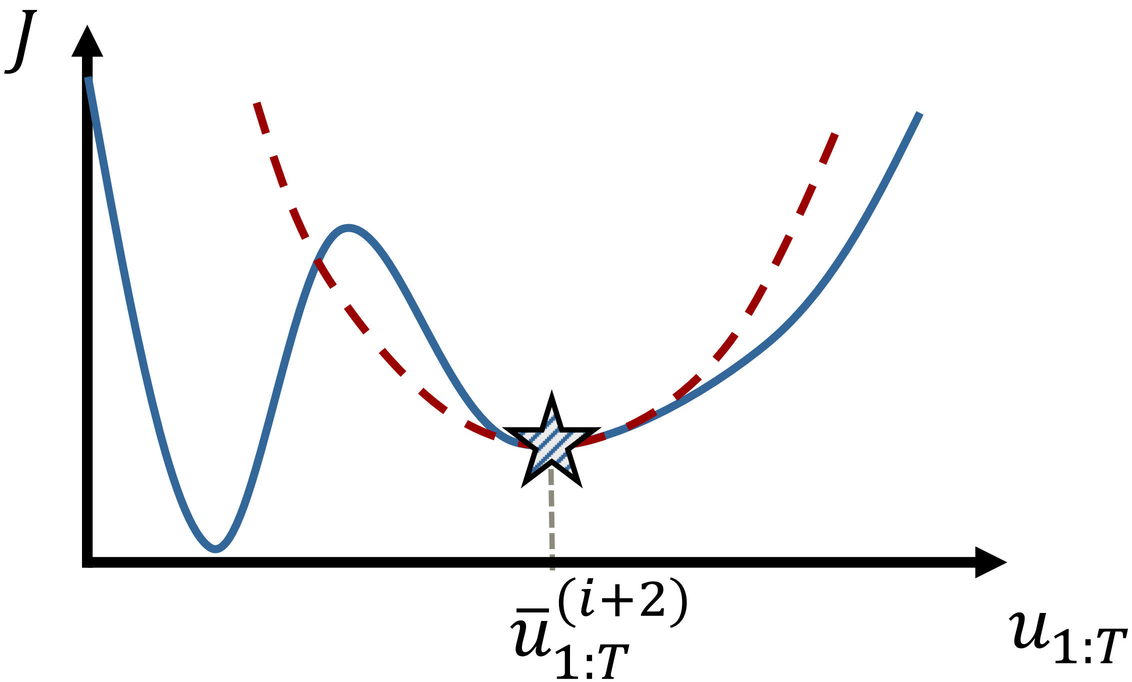
ILQR: Main Idea

- Approximate the solution to a nonlinear OCP via local LQR updates

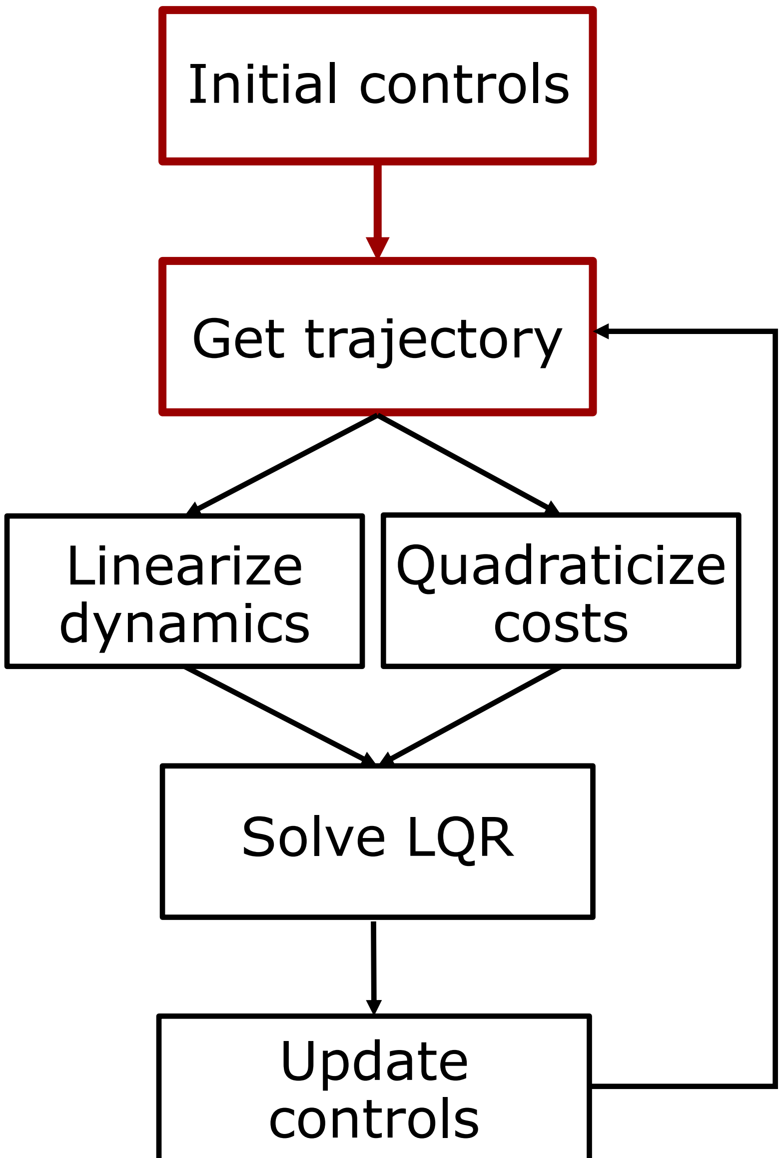


ILQR: Main Idea

- Approximate the solution to a nonlinear OCP via local LQR updates
  - Special case of **Newton's method**



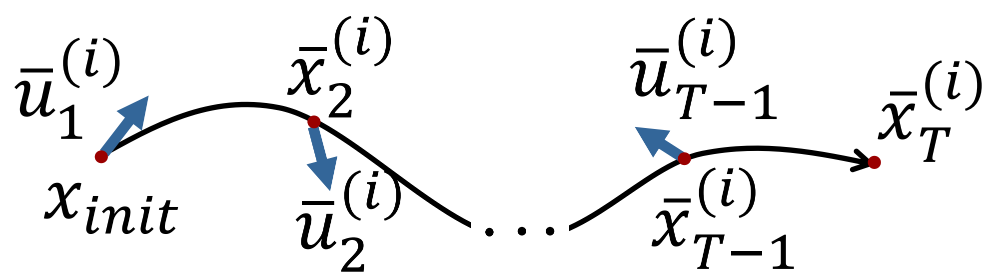
ILQR: Algorithm



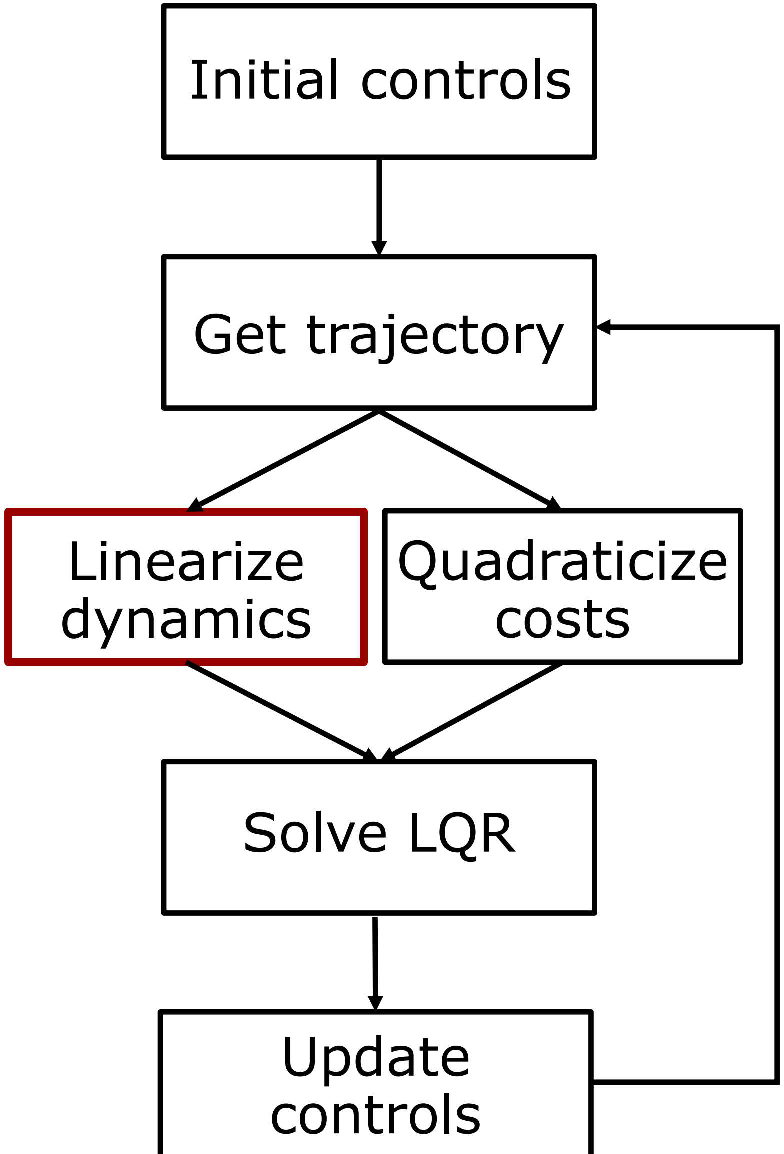
Rollout

- Forward simulation of **nonlinear dynamics**

$$\bar{x}_{1:T}^{(i)} = F_1(x_{\text{init}}, \bar{u}_{1:T}^{(i)})$$



ILQR: Algorithm



Linearize Dynamics

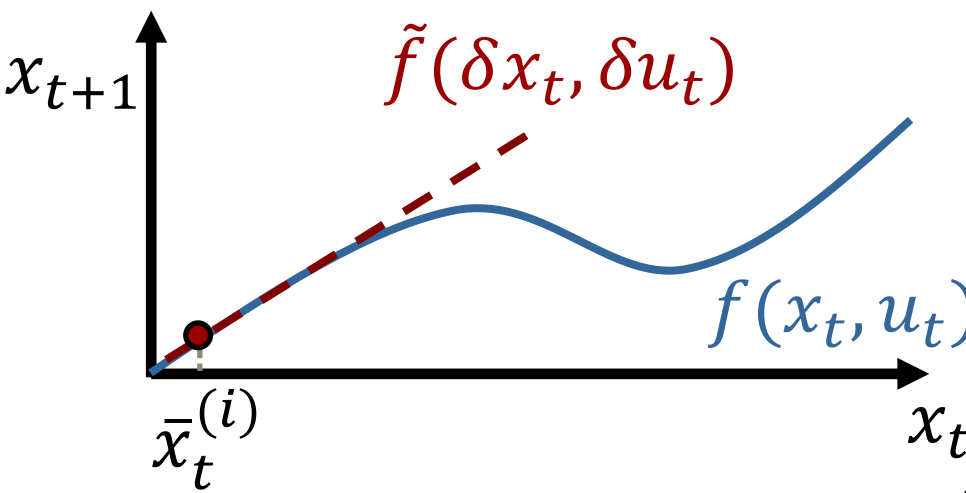
$$\delta x_{t+1} \approx \tilde{A}_t \delta x + \tilde{B}_t \delta u_t$$

$$\text{where} \quad \tilde{A}_t := \partial_{x_t} f(\bar{x}_t^{(i)}, \bar{u}_t^{(i)})$$

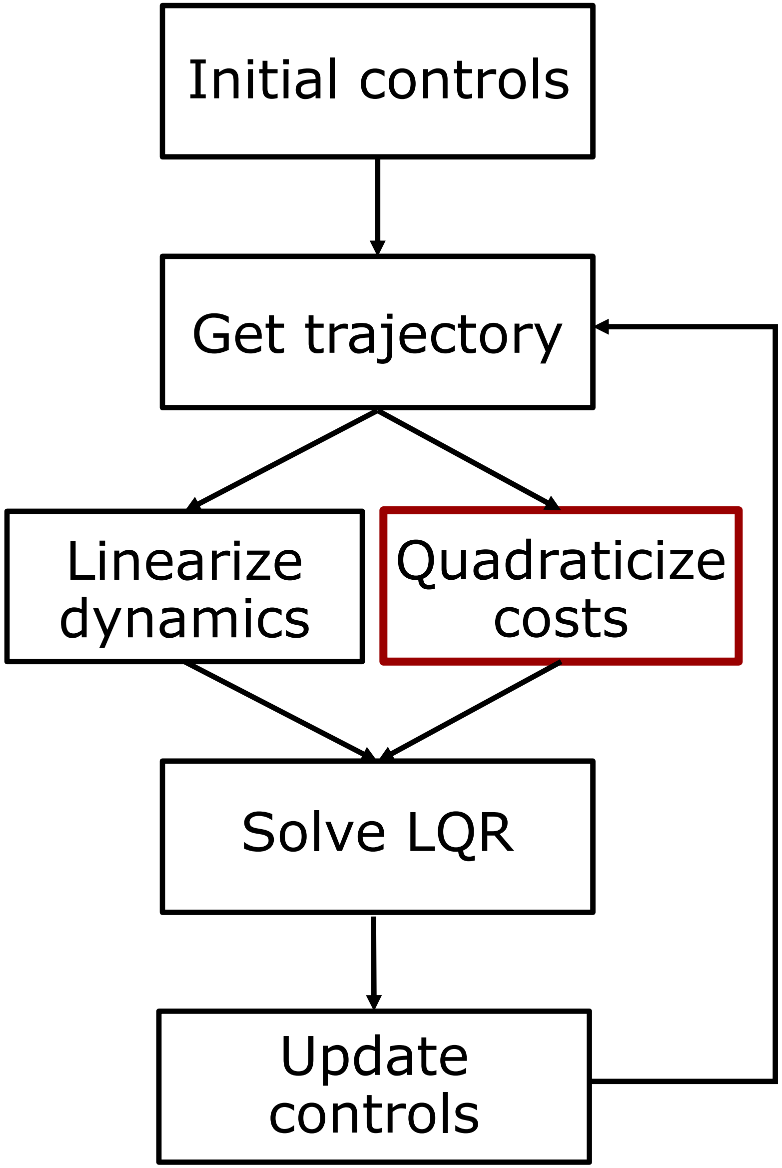
$$\tilde{B}_t := \partial_{u_t} f(\bar{x}_t^{(i)}, \bar{u}_t^{(i)})$$

$$\delta x_t := (x_t - \bar{x}_t^{(i)})$$

$$\delta u_t := (u_t - \bar{u}_t^{(i)})$$

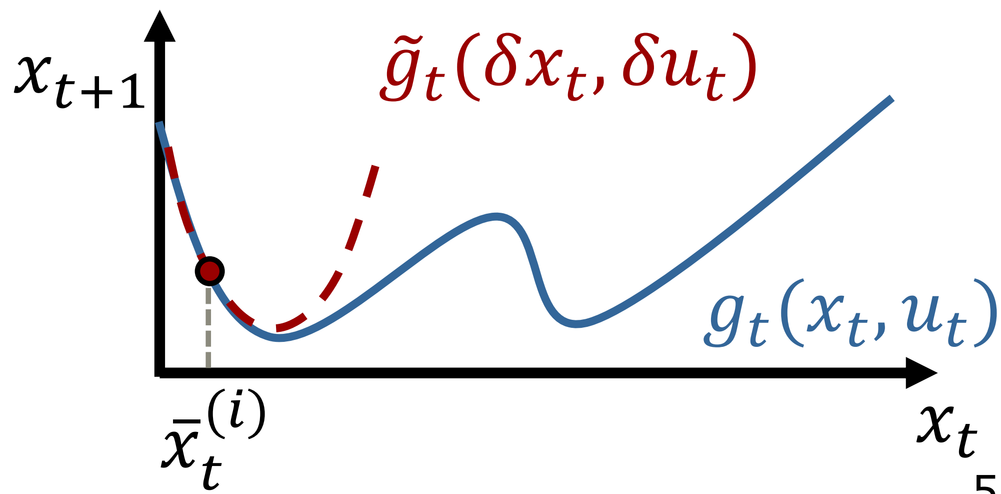


ILQR: Algorithm

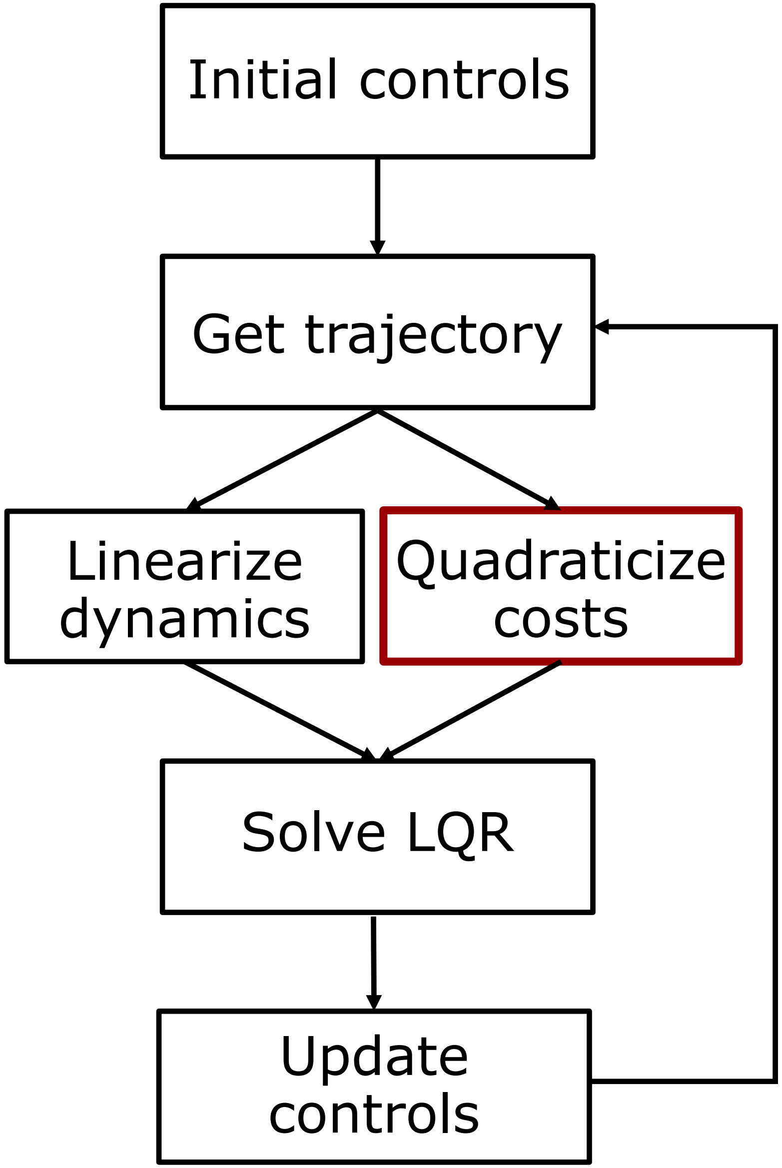


Quadraticize Costs

$$\tilde{g}_t(\delta x_t, \delta u_t) = \delta x_t^\top \tilde{Q}_t \delta x_t + \delta u^\top \tilde{R}_t \delta u + \delta x_t^\top \tilde{q}_t + \delta u_t^\top \tilde{r}_t + \tilde{c}_t$$



ILQR: Algorithm

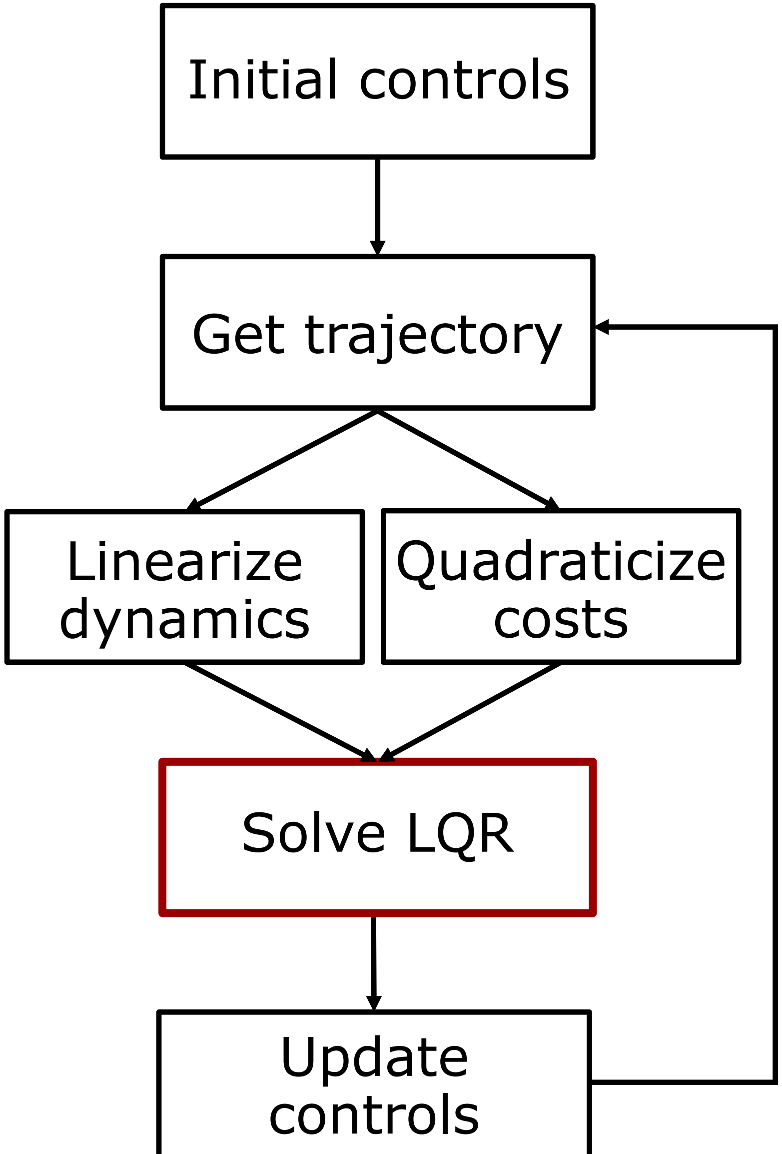


Quadraticize Costs

$$\tilde{g}_t(\delta x_t, \delta u_t) = \underline{\delta x_t^\top \tilde{Q}_t \delta x_t + \delta u^\top \tilde{R}_t \delta u}$$
$$+ \delta x_t^\top \tilde{q}_t + \delta u_t^\top \tilde{r}_t + \tilde{c}_t$$

where
$$\tilde{Q}_t := \frac{1}{2} \partial_{x_t x_t} g_t(\bar{x}_t^{(i)}, \bar{u}_t^{(i)})$$
$$\tilde{R}_t := \frac{1}{2} \partial_{u_t u_t} g_t(\bar{x}_t^{(i)}, \bar{u}_t^{(i)})$$
$$\tilde{q}_t := \partial_{x_t} g_t(\bar{x}_t^{(i)}, \bar{u}_t^{(i)})$$
$$\tilde{r}_t := \partial_{u_t} g_t(\bar{x}_t^{(i)}, \bar{u}_t^{(i)})$$
$$\tilde{c}_t := g_t(\bar{x}_t^{(i)}, \bar{u}_t^{(i)})$$

ILQR: Algorithm



Solve LQR

- Augment state and input

$$z_t := \begin{bmatrix} \delta x_t \\ 1 \end{bmatrix}, \qquad v_t := \begin{bmatrix} \delta u_t \\ 1 \end{bmatrix}$$

$$A_t := \begin{bmatrix} \tilde{A}_t & 0 \\ 0 & 1 \end{bmatrix}, \qquad B_t := \begin{bmatrix} \tilde{B}_t \\ 0 \end{bmatrix}$$

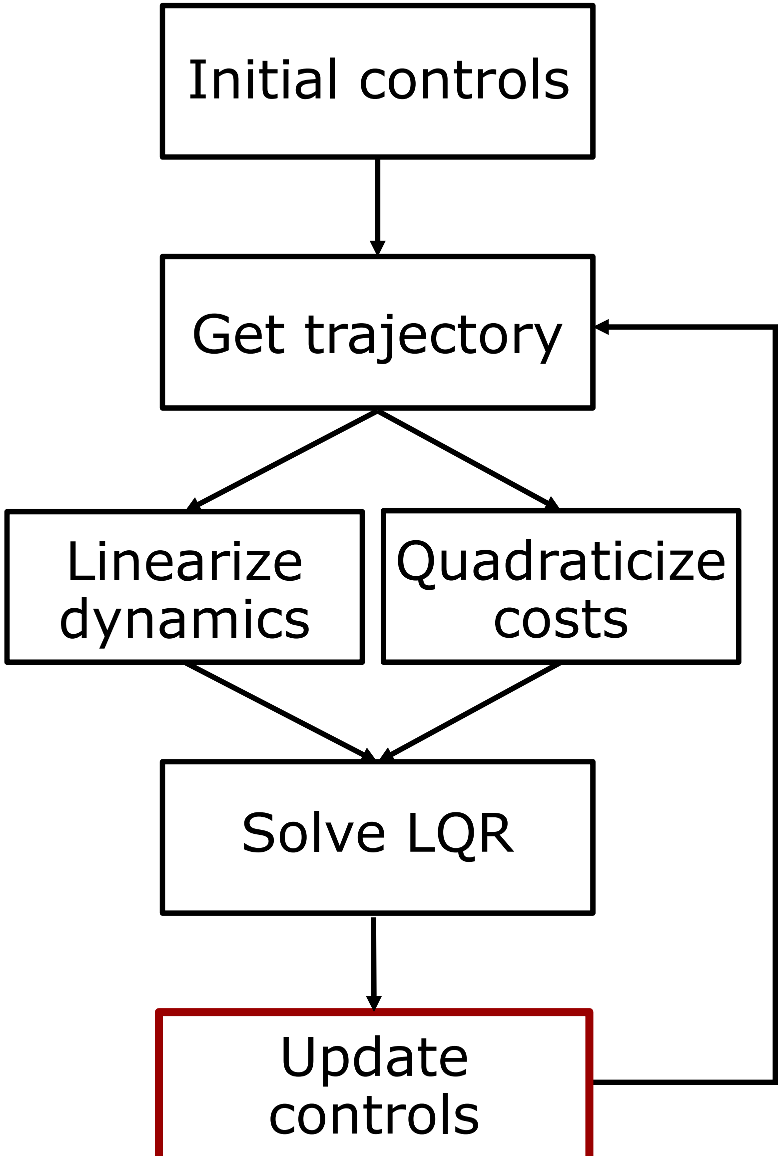
$$Q_t := \begin{bmatrix} \tilde{Q}_t & \frac{1}{2}\tilde{q}_t \\ \frac{1}{2}\tilde{q}_t^\top & \tilde{c}_t \end{bmatrix}, \quad R_t := \begin{bmatrix} \tilde{R}_t & \frac{1}{2}\tilde{r}_t \\ \frac{1}{2}\tilde{r}_t^\top & 0 \end{bmatrix}$$

- Solve standard LQR

$$\min_{z_{1:T}, v_{1:T}} \quad \sum_{t \in [T]} z_t^\top Q_t z_t + v_t^\top R_t v_t$$

$$\text{subject to} \quad z_{t+1} = A_t z_t + B_t v_t,$$

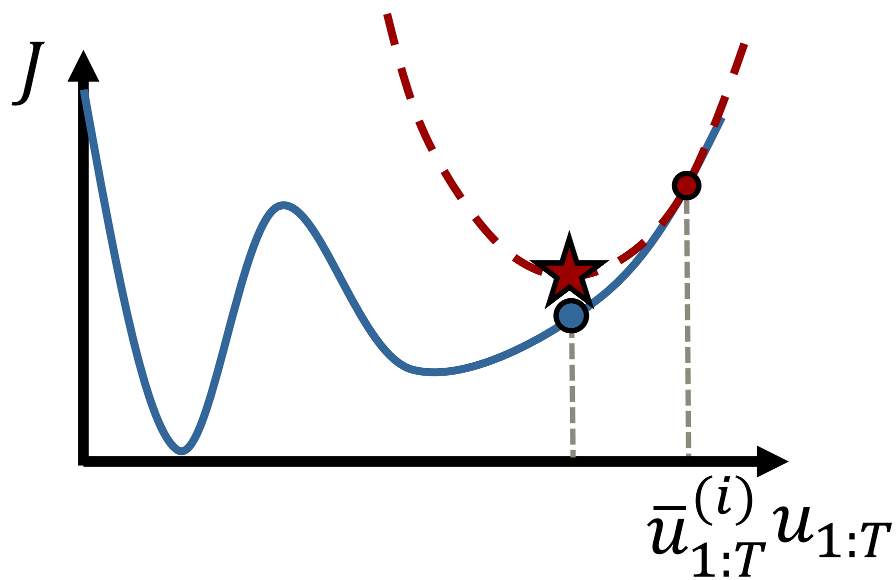
ILQR: Algorithm



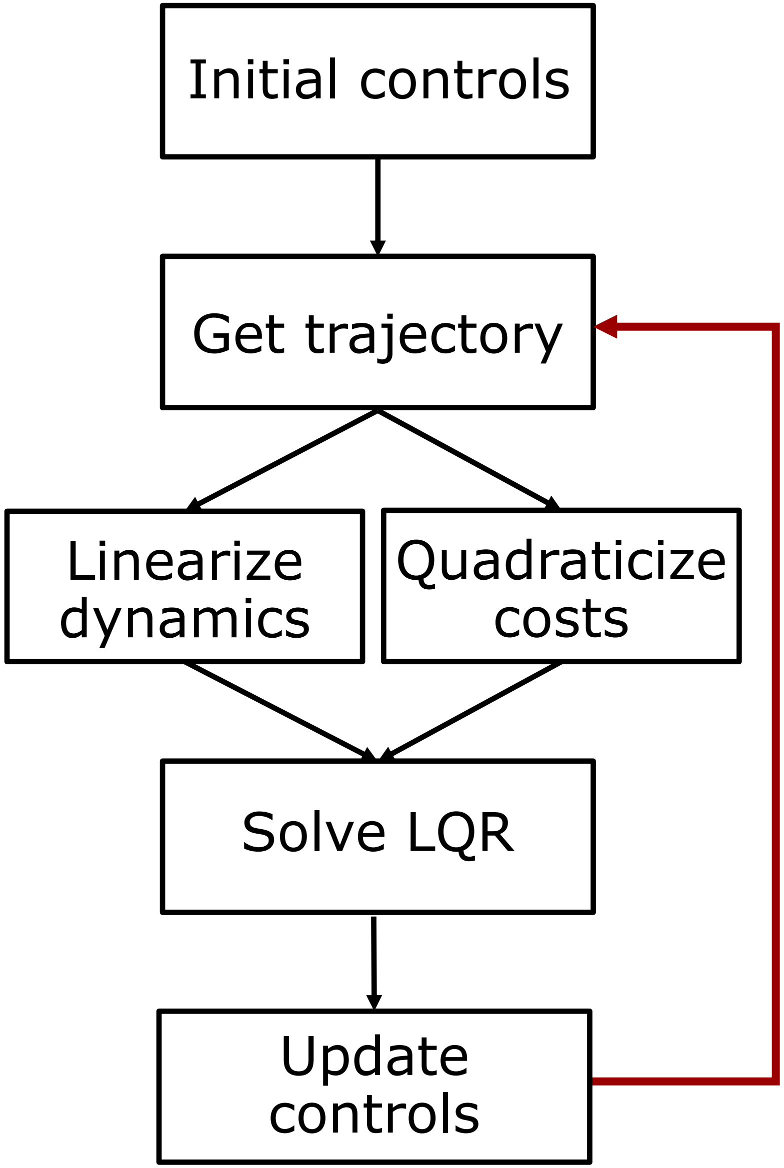
Update Controls

- Step towards the LQR solution with step-size $\alpha \in (0, 1]$

$$\bar{u}_t^{(i+1)} = \bar{u}_t^{(i)} + \alpha \delta u_t^*, \forall t \in [T]$$



ILQR: Algorithm



Repeat

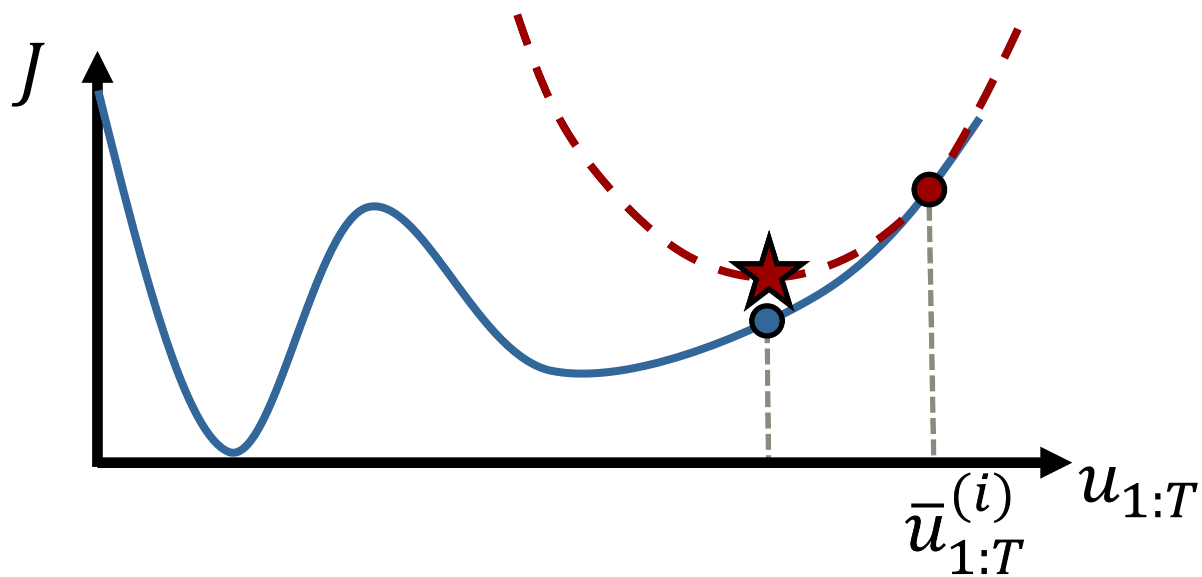
- Rollout the updated controls to get $\bar{x}_{1:T}^{(i+1)}$
- Solve a new LQR problem
- Stop once operating point has converged $\bar{x}_{1:T}^{(i+1)} \approx \bar{x}_{1:T}^{(i)}$

ILQR: Practical Considerations

- The quadratic approximation of the cost must remain **positive definite**

$$u_t^*(x_t) = F_t x_t$$

$$\text{where } F_t := -(B^\top P_{t+1} B + R)^{-1} B^\top P_{t+1} A$$

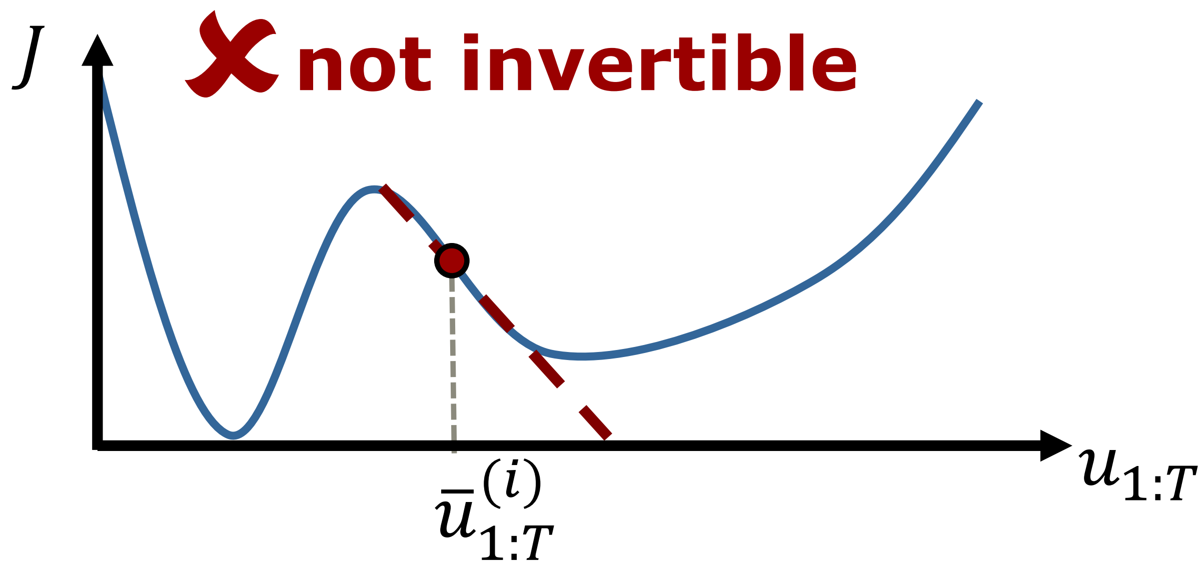


ILQR: Practical Considerations

- The quadratic approximation of the cost must remain **positive definite**

$$u_t^*(x_t) = F_t x_t$$

$$\text{where } F_t := -(B^\top P_{t+1} B + R)^{-1} B^\top P_{t+1} A$$

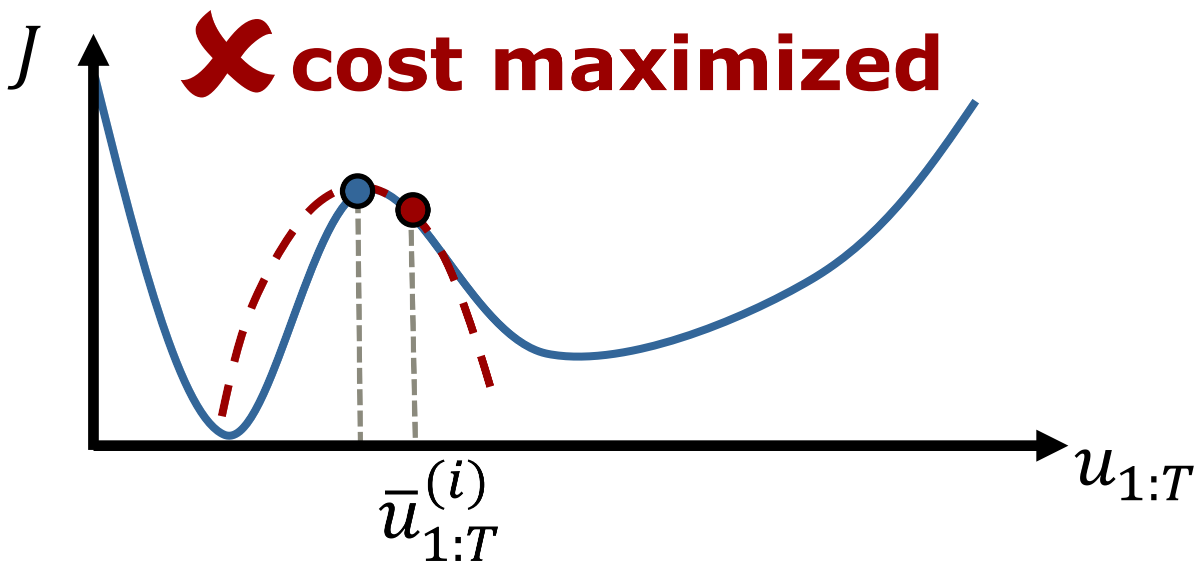


ILQR: Practical Considerations

- The quadratic approximation of the cost must remain **positive definite**

$$u_t^*(x_t) = F_t x_t$$

$$\text{where } F_t := -(B^\top P_{t+1} B + R)^{-1} B^\top P_{t+1} A$$

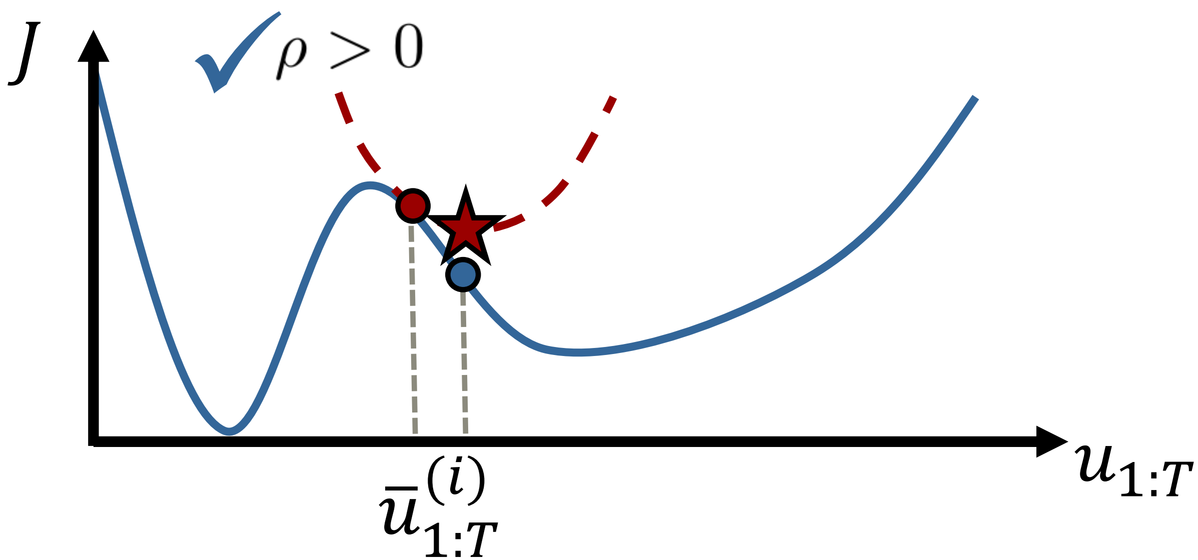


ILQR: Practical Considerations

- The quadratic approximation of the cost must remain **positive definite**
- Fix: **regularization**

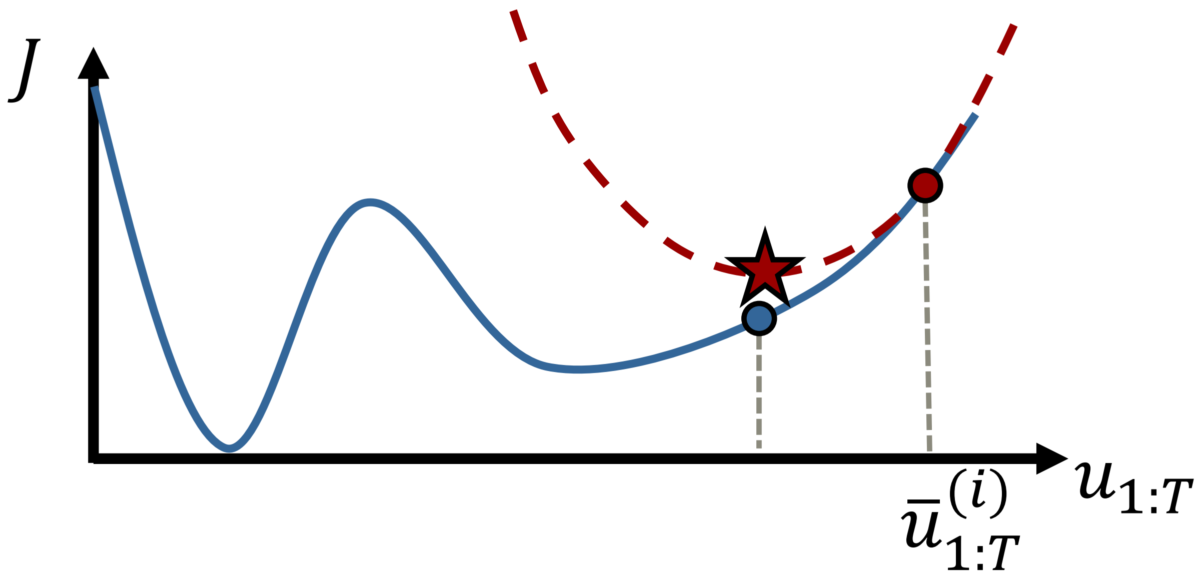
$$u_t^*(x_t) = F_t x_t$$

$$\text{where } F_t := -(B^\top P_{t+1} B + R + \underline{\rho I})^{-1} B^\top P_{t+1} A$$



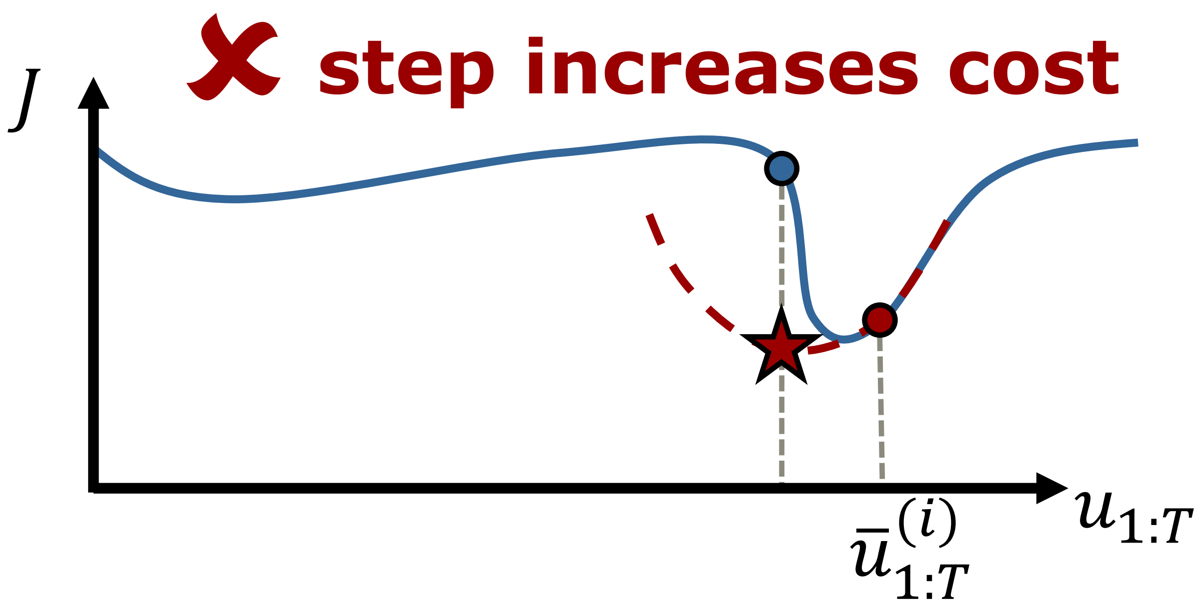
ILQR: Practical Considerations

- The step-size α must balance between
  - Large: Make **progress** towards goal
  - Small: **Reduce error** of the local model



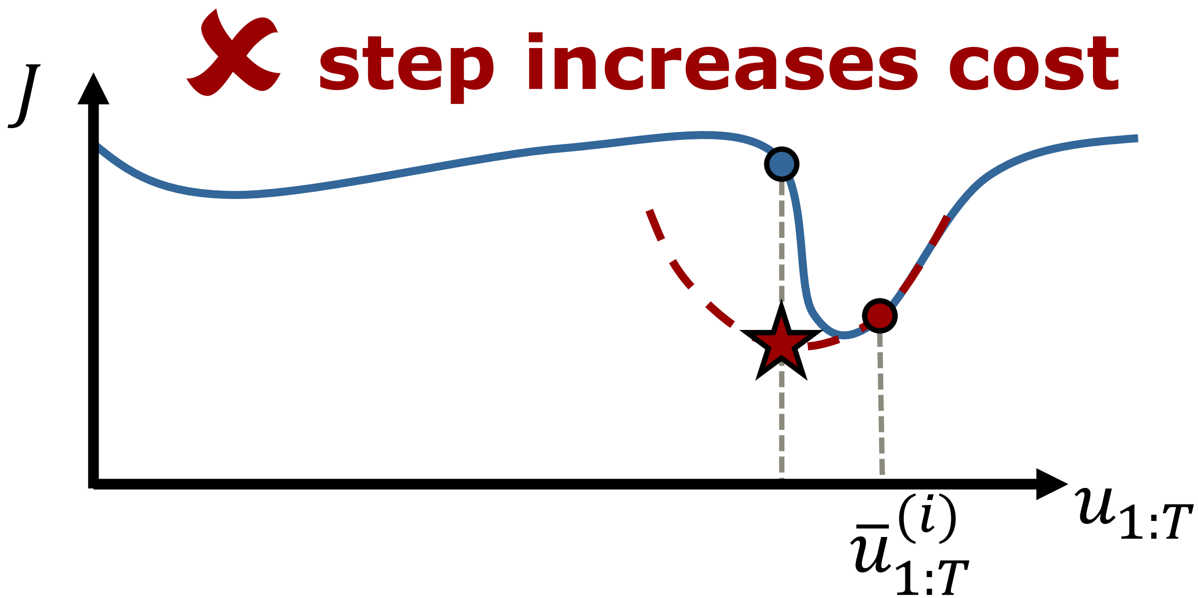
ILQR: Practical Considerations

- The step-size α must balance between
  - Large: Make **progress** towards goal
  - Small: **Reduce error** of the local model



ILQR: Practical Considerations

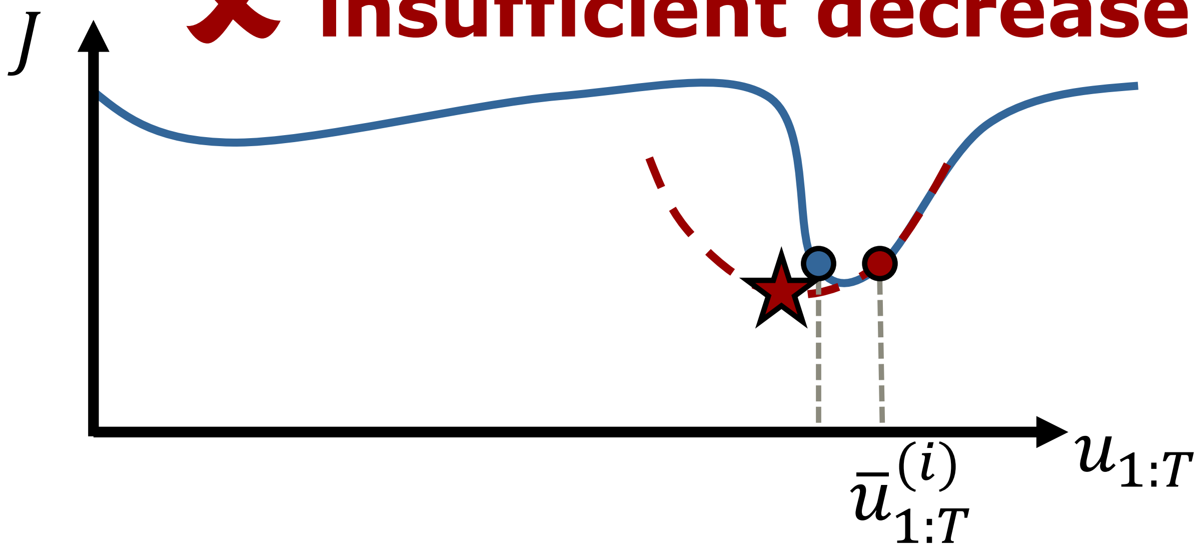
- The step-size α must balance between
  - Large: Make **progress** towards goal
  - Small: **Reduce error** of the local model
- Fix: **line search** over $\alpha \in (0, 1]$: $\alpha = 1$



ILQR: Practical Considerations

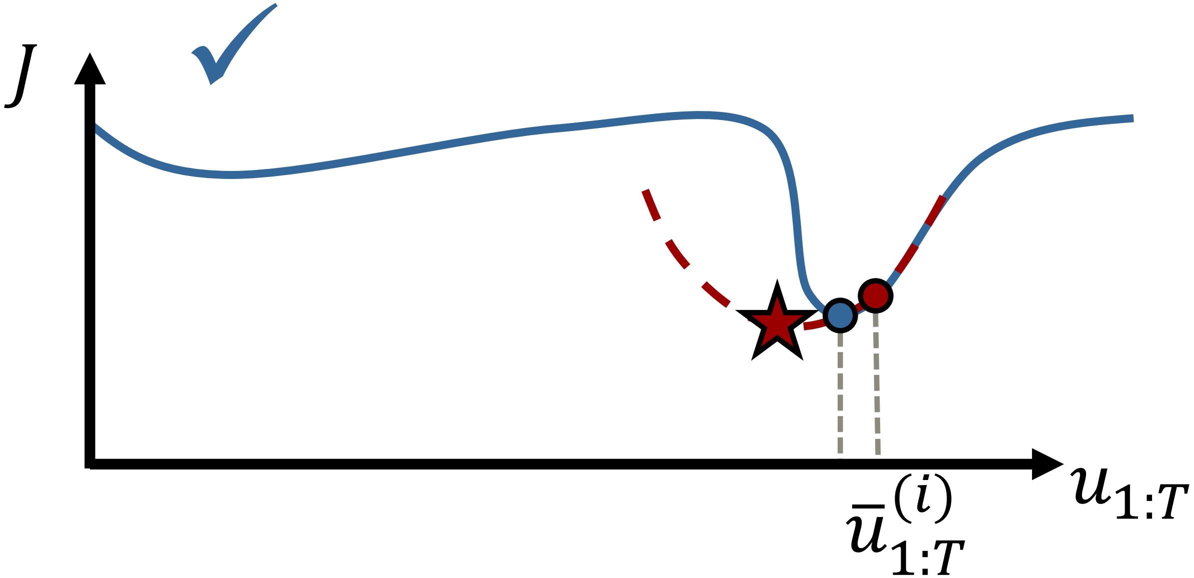
- The step-size α must balance between
  - Large: Make **progress** towards goal
  - Small: **Reduce error** of the local model
- Fix: **line search** over $\alpha \in (0, 1]$: $\alpha = 0.5$

✘ **insufficient decrease**



ILQR: Practical Considerations

- The step-size α must balance between
  - Large: Make **progress** towards goal
  - Small: **Reduce error** of the local model
- Fix: **line search** over $\alpha \in (0, 1]$: $\alpha = 0.25$



ILQR: Summary

- Approach
  - Start with an **initial guess** for controls
  - Iteratively update the solution via **local LQR updates**
- Comments
  - Very efficient and well established for mildly nonlinear OCPs
  - Requires regularization and line-search to work reliably
  - Can be extended to include inequality constraints

Summary

- Recap: Model Predictive Control
- Elimination of states and dynamics constraints via **forward simulation**
- Reformulation as unconstrained optimization
- **LQR** solutions
  - Batch solution via linear least-squares
  - **Dynamic Programming**
- **Iterative LQR** for nonlinear OCP

Resources

- "Predictive Control for Linear and Hybrid Systems" by Borrelli et al.
  Link: http://www.mpc.berkeley.edu/mpc-course-material
- "Numerical Optimization" by Nocedal & Wright
- Underactuated Robotics class of Russ Tedrake, MIT
  Link: http://underactuated.mit.edu

Thank you for your attention