

Underactuated Robots
Optimization methods
for planning and control:
Part 2
Giulio Turrisi

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI



SAPIENZA
UNIVERSITÀ DI ROMA

part 1


- some methods for finding a control law $u^*(x)$ that is optimal w.r.t to a cost function
- DP is only applicable on **small, finite** and **discrete** state and input spaces
- the HJB equation is the extension of the DP equation for state - input space and time
- the Riccati equation is a special solution of the HJB equation for linear system; the resulting LQR can be used as a control law for **nonlinear** systems, but it only works in the vicinity of the linearization point

outline of part 2

- discrete linear time-invariant LQR
- discrete linear time-varying LQR
- trajectory optimization
- linear constrained optimization
- nonlinear constrained optimization
- case study

continuous LQR - infinite horizon

- we derived and applied LQR for **linear time-invariant** (LTI) systems

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)$$


they do not depend on time

with the cost function

$$J(\mathbf{x}, \mathbf{u}) = \int_0^{\infty} \mathbf{x}(t)^T \mathbf{Q}\mathbf{x}(t) + \mathbf{u}(t)^T \mathbf{R}\mathbf{u}(t) dt$$

and we end up obtaining an optimal control law of the form

$$\mathbf{u}^*(t) = -\mathbf{R}^{-1} \mathbf{B}^T \mathbf{P}\mathbf{x}(t)$$

discrete LQR

- we can easily derive a discrete version of LQR using **dynamic programming**
- define a change of coordinates for the nonlinear system $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$

$$\hat{\boldsymbol{x}}(t) = \boldsymbol{x}(t) - \boldsymbol{x}^d \quad \hat{\boldsymbol{u}}(t) = \boldsymbol{u}(t) - \boldsymbol{u}^d$$

- and linearize the system using a first-order Taylor expansion

$$\dot{\hat{\boldsymbol{x}}}(t) = \left. \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \right|_{\boldsymbol{x}^d, \boldsymbol{u}^d} (\boldsymbol{x}(t) - \boldsymbol{x}^d) + \left. \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}} \right|_{\boldsymbol{x}^d, \boldsymbol{u}^d} (\boldsymbol{u}(t) - \boldsymbol{u}^d)$$



$$\dot{\hat{\boldsymbol{x}}} = \boldsymbol{A}\hat{\boldsymbol{x}}(t) + \boldsymbol{B}\hat{\boldsymbol{u}}(t)$$

- in general, we have $(\boldsymbol{x}^d, \boldsymbol{u}^d) = (\mathbf{0}, \mathbf{0})$ and therefore

$$\hat{\boldsymbol{x}}(t) = \boldsymbol{x}(t) \quad \hat{\boldsymbol{u}}(t) = \boldsymbol{u}(t)$$

discrete LQR

- discretize the resulting LTI system

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k$$

- the cost function is defined up to an horizon N

$$J(\mathbf{x}, \mathbf{u}) = \mathbf{x}_N^T \mathbf{Q}_N \mathbf{x}_N + \sum_{k=0}^{N-1} \mathbf{x}_k^T \mathbf{Q} \mathbf{x}_k + \mathbf{u}_k^T \mathbf{R} \mathbf{u}_k$$

- remember that in DP we want to retrieve the optimal value function with a **backward** computation

$$J^*(\mathbf{x}_k) = \min_{\mathbf{u}_k} [\mathbf{x}_k^T \mathbf{Q} \mathbf{x}_k + \mathbf{u}_k^T \mathbf{R} \mathbf{u}_k + J^*(\mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k)]$$

starting from a known **final cost**

discrete LQR

- defining $P_N = Q_N$ we can proceed in **backward**

$$J^*(\mathbf{x}_N) = \mathbf{x}_N^T \mathbf{P}_N \mathbf{x}_N$$

$$J^*(\mathbf{x}_{N-1}) = \min_{\mathbf{u}_{N-1}} [\mathbf{x}'_{N-1} \mathbf{Q} \mathbf{x}_{N-1} + \mathbf{u}'_{N-1} \mathbf{R} \mathbf{u}_{N-1} + \mathbf{x}_N^T \mathbf{P}_N \mathbf{x}_N]$$

$$\mathbf{x}_N = \mathbf{A} \mathbf{x}_{N-1} + \mathbf{B} \mathbf{u}_{N-1}$$

- the minimum \mathbf{u}_{N-1} is

$$\mathbf{u}_{N-1} = -(\mathbf{R}^{-1} \mathbf{B}^T \mathbf{P}_N \mathbf{B})^{-1} \mathbf{B}' \mathbf{P}_N \mathbf{A} \mathbf{x}_{N-1}$$

that we can plug back in $J^*(\mathbf{x}_{N-1})$

discrete LQR - recursion

$$P_N = Q_N$$

$$J^*(\mathbf{x}_N) = \mathbf{x}_N^T P_N \mathbf{x}_N$$



$$\mathbf{u}_{N-1} = -(\mathbf{R}^{-1} \mathbf{B}^T P_N \mathbf{B})^{-1} \mathbf{B}' P_N \mathbf{A} \mathbf{x}_{N-1}$$

$$P_{N-1} = \mathbf{Q} + \mathbf{A}^T (P_N - P_N \mathbf{B} (\mathbf{R} + \mathbf{B}^T P_{N+1} \mathbf{B})^{-1} \mathbf{B}^T P_{N+1}) \mathbf{A}$$

$$J^*(\mathbf{x}_{N-1}) = \mathbf{x}'_{N-1} P_{N-1} \mathbf{x}_{N-1}$$

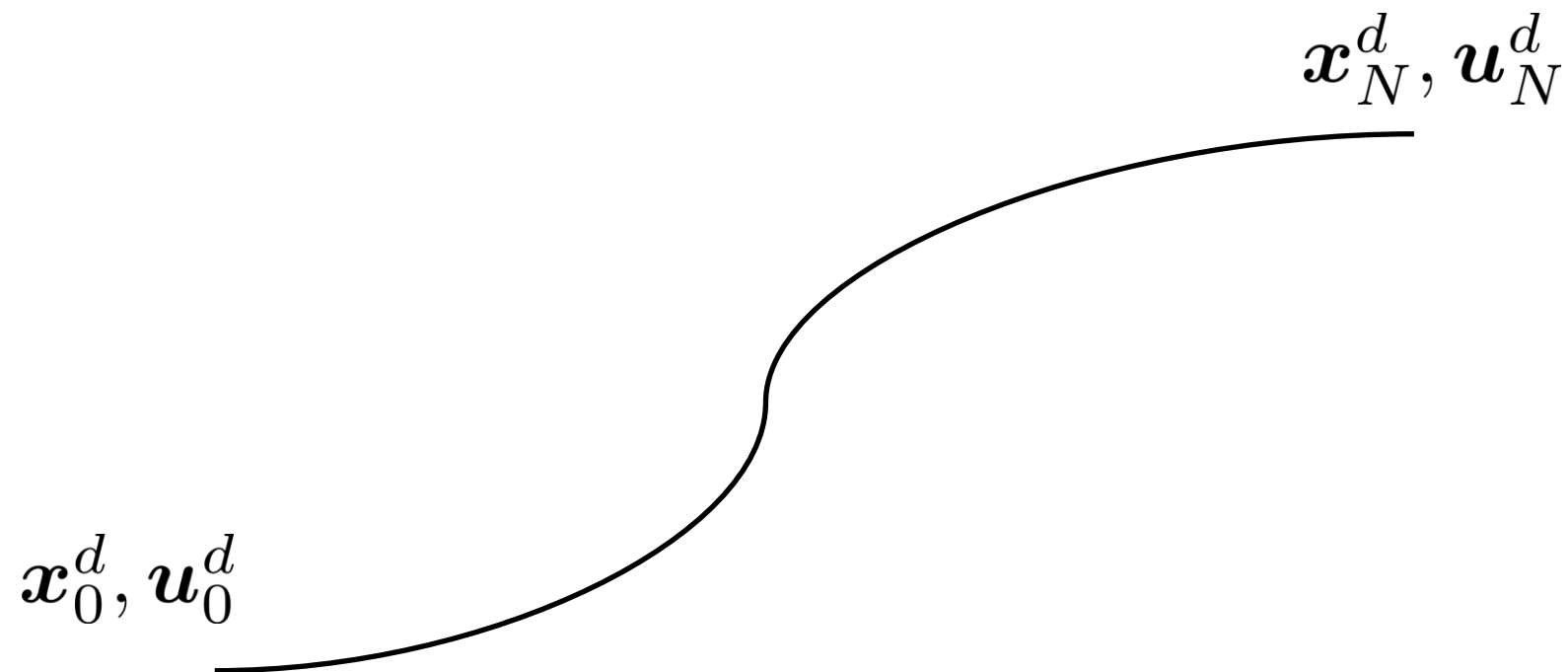


until we reach the first time instant

$$\mathbf{u}_0 = -(\mathbf{R} + \mathbf{B}^T P_1 \mathbf{B})^{-1} \mathbf{B}' P_1 \mathbf{A} \mathbf{x}_0$$

linear time-varying LQR

- LQR can be used as a **linear and local** control law for nonlinear systems, linearizing the model around a fixed point
- what happens if we linearize the system around a **trajectory**?



- **same computation** as before!

linear time-varying LQR

- suppose to have a discrete desired trajectory that we want to track, defined as a pair of state and input

$$[\mathbf{x}_i^d, \mathbf{u}_i^d] \text{ for } i = 0, \dots, N$$

- we can still apply LQR, but we must deal now with a **linear time-variant** (LTV) system

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t)$$



they depend on time!

since at each time step the linearization point changes **along the trajectory**

linear time-varying LQR

- we will consider the discrete version of LTV LQR
- define a change of coordinates for the nonlinear system $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$

$$\hat{\boldsymbol{x}}(t) = \boldsymbol{x}(t) - \boldsymbol{x}_k^d \quad \hat{\boldsymbol{u}}(t) = \boldsymbol{u}(t) - \boldsymbol{u}_k^d$$

and linearize the system using a first-order Taylor expansion

$$\dot{\hat{\boldsymbol{x}}}(t) = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}(\boldsymbol{x}(t) - \boldsymbol{x}_k^d) \Big|_{\boldsymbol{x}_k^d, \boldsymbol{u}_k^d} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}} \Big|_{\boldsymbol{x}_k^d, \boldsymbol{u}_k^d} (\boldsymbol{u}(t) - \boldsymbol{u}_k^d)$$



$$\dot{\hat{\boldsymbol{x}}} = \boldsymbol{A}(t)\hat{\boldsymbol{x}}(t) + \boldsymbol{B}(t)\hat{\boldsymbol{u}}(t)$$

linear time-varying LQR

- discretize the resulting LTV system

$$\hat{\mathbf{x}}_{k+1} = \mathbf{A}_k \hat{\mathbf{x}}_k + \mathbf{B}_k \hat{\mathbf{u}}_k$$

- penalize the deviation from the desired trajectory in the cost function

$$J_k(\mathbf{x}, \mathbf{u}) = \hat{\mathbf{x}}_N^T \mathbf{Q}_N \hat{\mathbf{x}}_N + \sum_{k=0}^{N-1} \hat{\mathbf{x}}_k^T \mathbf{Q} \hat{\mathbf{x}}_k + \hat{\mathbf{u}}_k^T \mathbf{R} \hat{\mathbf{u}}_k$$

- the optimal cost-to-go at the end of the trajectory is

$$J_k^*(\mathbf{x}_N) = \hat{\mathbf{x}}_N^T \mathbf{P}_N \hat{\mathbf{x}}_N$$

where

$$\mathbf{P}_N = \mathbf{Q}_N$$

$$\mathbf{P}_k = \mathbf{Q} + \mathbf{A}_k^T (\mathbf{P}_{k+1} - \mathbf{P}_{k+1} \mathbf{B}_k (\mathbf{R} + \mathbf{B}_k^T \mathbf{P}_{k+1} \mathbf{B}_k)^{-1} \mathbf{B}_k^T \mathbf{P}_{k+1}) \mathbf{A}_k$$

linear time-varying LQR

- iterate from the end of the trajectory to its beginning

$$P_N = Q_N$$

$$P_{N-1} = Q + A_{N-1}^T (P_N - P_N B_{N-1} (R + B_{N-1}^T P_N B_{N-1})^{-1} B_{N-1}^T P_N) A_{N-1}$$

...

...

$$P_0 = Q + A_0^T (P_1 - P_1 B_0 (R + B_0^T P_1 B_0)^{-1} B_0^T P_1) A_0$$

- obtaining for each discrete time step k the optimal control feedback for tracking the desired trajectory

$$u_k^* = u_k^d - (R + B_k^T P_{k+1} B_k)^{-1} B_k^T P_{k+1} A_k (x_k - x_k^d)$$



can use **time-varying weights**

trajectory optimization

- LTV LQR can be used to stabilize a nonlinear system around the desired trajectory
- it requires that the system starts **close** to the trajectory, otherwise the linearization is no longer accurate (same as in the LTI LQR case)
- the presented recursive algorithm used for solving the LTV LQR can be used (with some modifications) even to **generate** an unconstrained desired trajectory
- this problem is called **trajectory optimization**, and it can be solved using **iterative LQR** (ILQR)

iterative LQR - algorithm

- define the cost

$$J(\mathbf{x}, \mathbf{u}) = \mathbf{x}_N^T \mathbf{Q}_N \mathbf{x}_N + \underbrace{\sum_{k=0}^{N-1} \mathbf{x}_k^T \mathbf{Q} \mathbf{x}_k + \mathbf{u}_k^T \mathbf{R} \mathbf{u}_k}_{l(\mathbf{x}_k, \mathbf{u}_k)}$$

- steps to repeat from $h = 0$ to h_{max}

1. linearize the system over an **initial guess**

$$\begin{aligned} & \mathbf{x}_0^h, \mathbf{x}_1^h, \dots, \mathbf{x}_N^h \\ & \mathbf{u}_0^h, \mathbf{u}_1^h, \dots, \mathbf{u}_{N-1}^h \end{aligned}$$

2. run the LTV LQR (**backward pass**) to minimize $J(\mathbf{x}, \mathbf{u})$

3. apply the new inputs to the nominal model (**forward pass**)

4. set $h = h + 1$ and as **initial guess** the new states obtained running the forward pass and the applied inputs

iterative LQR - backward pass

- to minimize

$$J^*(\mathbf{x}_k, h) = \min_{\mathbf{u}_k} [l(\mathbf{x}_k, \mathbf{u}_k) + J^*(\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k))]$$

\uparrow
 $C(\mathbf{x}_k, \mathbf{u}_k)$

expand $C(\hat{\mathbf{x}}_k, \hat{\mathbf{u}}_k)$ at the second order around the last initial guess

$$\approx \frac{1}{2} \begin{bmatrix} 1 \\ \hat{\mathbf{x}}_k \\ \hat{\mathbf{u}}_k \end{bmatrix}^\top \begin{bmatrix} 0 & C_x^\top & C_u^\top \\ C_x & C_{xx} & C_{xu} \\ C_u & C_{ux} & C_{uu} \end{bmatrix} \begin{bmatrix} 1 \\ \hat{\mathbf{x}}_k \\ \hat{\mathbf{u}}_k \end{bmatrix}$$

with $\hat{\mathbf{x}}_k = \mathbf{x}_k - \mathbf{x}_k^h$ and $\hat{\mathbf{u}}_k = \mathbf{u}_k - \mathbf{u}_k^h$

iterative LQR - backward pass

$$\approx \frac{1}{2} \begin{bmatrix} 1 \\ \hat{\mathbf{x}}_k \\ \hat{\mathbf{u}}_k \end{bmatrix}^\top \begin{bmatrix} 0 & C_x^\top & C_u^\top \\ C_x & C_{xx} & C_{xu} \\ C_u & C_{ux} & C_{uu} \end{bmatrix} \begin{bmatrix} 1 \\ \hat{\mathbf{x}}_k \\ \hat{\mathbf{u}}_k \end{bmatrix}$$

with

$$C_x = l_x + \mathbf{f}_x^\top J'_x$$

$$C_u = l_u + \mathbf{f}_u^\top J'_x$$

discarded inside ILQR

$$C_{xx} = l_{xx} + \mathbf{f}_x^\top J'_{xx} \mathbf{f}_x + \cancel{J'_x \cdot \mathbf{f}_{xx}}$$

$$C_{uu} = l_{uu} + \mathbf{f}_u^\top J'_{xx} \mathbf{f}_u + \cancel{J'_x \cdot \mathbf{f}_{uu}}$$

$$C_{ux} = l_{ux} + \mathbf{f}_u^\top J'_{xx} \mathbf{f}_x + \cancel{J'_x \cdot \mathbf{f}_{ux}}$$

iterative LQR - backward pass

- compute the minimum

$$\hat{\mathbf{u}}_k = \underset{\hat{\mathbf{u}}_k}{\operatorname{argmin}} C(\hat{\mathbf{x}}_k, \hat{\mathbf{u}}_k) = -C_{uu}^{-1} (C_u + C_{ux} \hat{\mathbf{x}}_k)$$

- the value function at time k is given by

$$J^*(\mathbf{x}_k, h) = \Delta J + J_x + J_{xx}$$

where

$$\Delta J = -\frac{1}{2} C_u C_{uu}^{-1} C_u$$

$$J_x = C_x - C_u C_{uu}^{-1} C_{ux}$$

$$J_{xx} = C_{xx} - C_{xu} C_{uu}^{-1} C_{ux}$$

iterative LQR - forward pass

1. from step $k = 0$ to N , apply to the nominal model the control inputs

$$\begin{aligned} \mathbf{u}_k^* &= \mathbf{u}_k^h + \hat{\mathbf{u}}_k \\ &\downarrow \\ \mathbf{x}_{k+1}^* &= \mathbf{f}_k(\mathbf{x}_k^*, \mathbf{u}_k^*) \end{aligned}$$

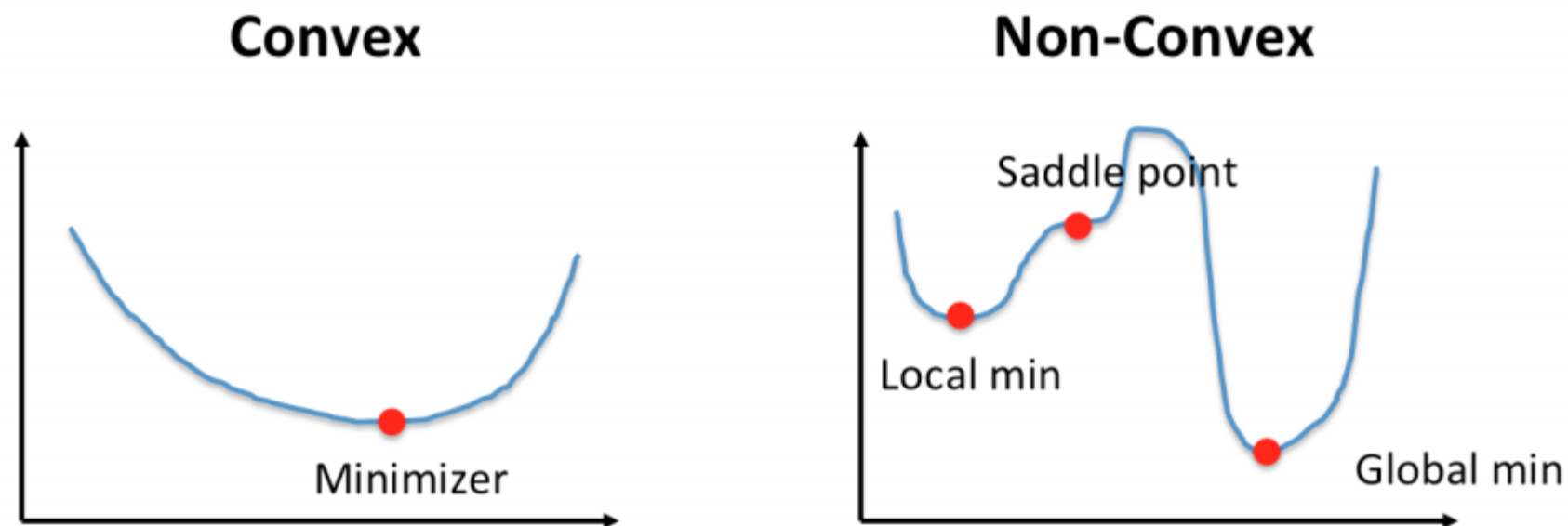
2. collect the new state evolution and set the new initial guess for the next iteration

$$\mathbf{x}^* \rightarrow \mathbf{x}_k^{h+1}, \mathbf{x}_{k+1}^{h+1}, \dots, \mathbf{x}_N^{h+1}$$

$$\mathbf{u}^* \rightarrow \mathbf{u}_k^{h+1}, \mathbf{u}_{k+1}^{h+1}, \dots, \mathbf{u}_{N-1}^{h+1}$$

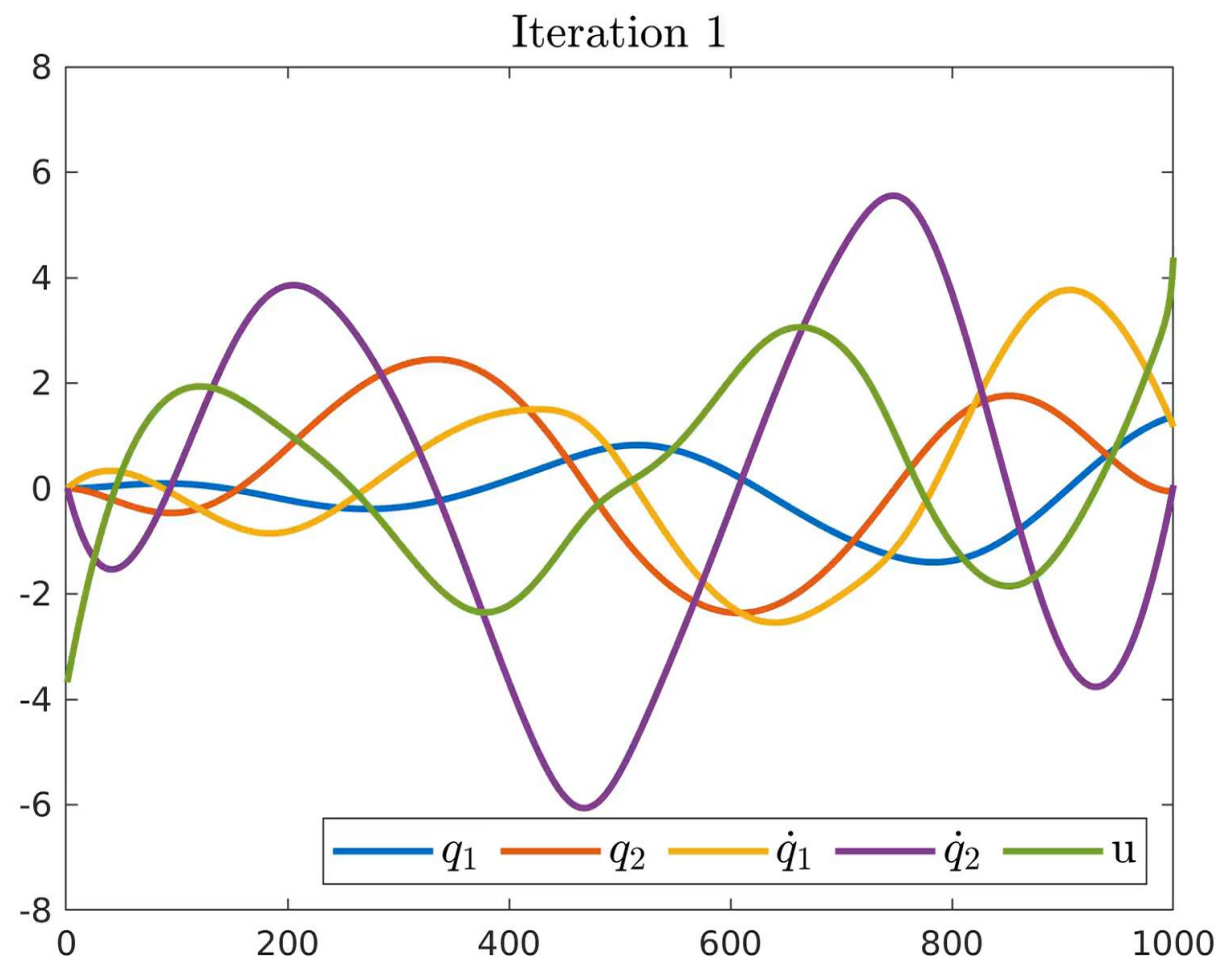
iterative LQR - algorithm

- after each iteration, the cost will be **lower** than the previous one
- the iterations can be stopped when x_N is close to x^d or there is no improvement in the cost J
- the system is nonlinear, hence this is a non-convex optimization problem and the algorithm can get stuck in a local minimum



iterative LQR - Acrobot

- consider the Acrobot, a 2R robot with the **first joint passive** and the **second joint actuated**
- the video below shows the evolution of the trajectory planned by the ILQR during each iteration
- state $\mathbf{x} = (q_1, q_2, \dot{q}_1, \dot{q}_2)$, goal $\mathbf{x}_e = (\pi, 0, 0, 0)$



iterative LQR - receding horizon

- ILQR generates an **unconstrained** open-loop trajectory for nonlinear systems
- we can **close the loop** and use ILQR directly for control, but only if we can obtain a solution fast enough to be **real-time**
- at every control time step k :
 1. measure the current state of the robot \mathbf{x}_k
 2. use ILQR to plan an open-loop trajectory $\mathbf{u}_k, \mathbf{u}_{k+1}, \dots, \mathbf{u}_{k+N-1}$ to minimize the cost function
 3. execute \mathbf{u}_k and **discard** $\mathbf{u}_{k+1}, \dots, \mathbf{u}_{k+N-1}$
- this strategy is called **receding horizon**

Synthesis of Complex Behaviors
with
Online Trajectory Optimization

Yuval Tassa, Tom Erez & Emo Todorov

IEEE International Conference
on Intelligent Robots and Systems
2012

iterative LQR - limitations

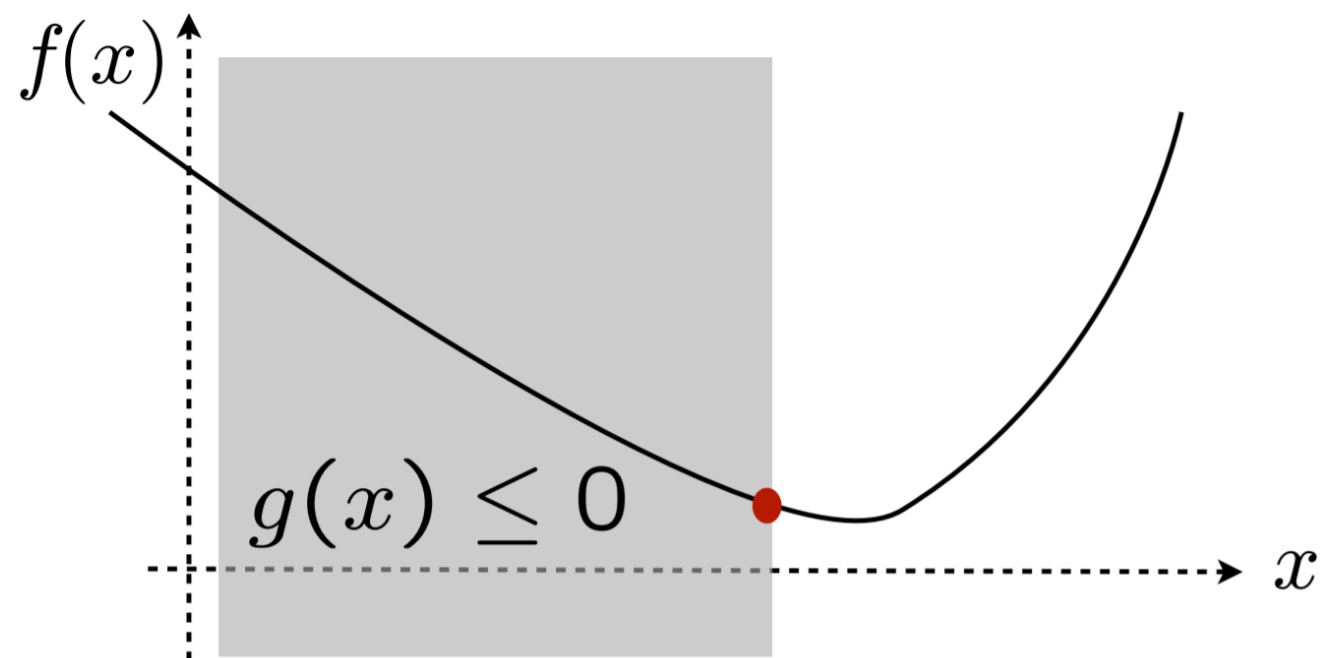
- ILQR can be used both for planning and control (for the last using a **receding horizon** strategy and only if the resulting controller is **real-time**)
- its performance depends largely on the **cost function** and on the chosen **horizon length** N (how far we look into the future)
- it does not scale well to high-dimensional state spaces, hindering its applicability for closed-loop control (but not for planning)
- it does not handle **state and input constraints** that can be essential on a real system

constrained optimization

- in general, **constrained** optimization problems are difficult to solve

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g(x) \leq 0 \end{aligned}$$

- the solution of the optimization problem cannot always be found in **closed form**

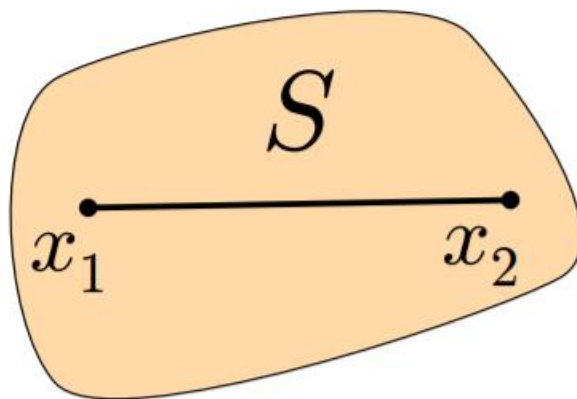


constrained optimization - convex set

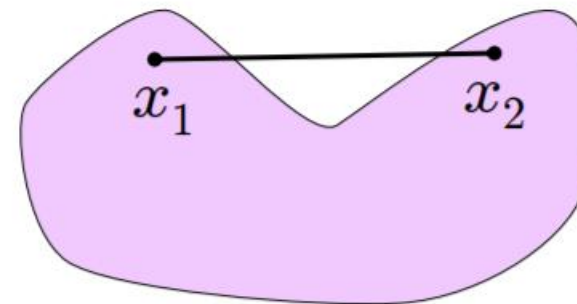
- optimization problems that are **convex** are easier to solve
- **definition**: a set S is convex if for each point $x_1, x_2 \in S$ we have

$$\lambda x_1 + (1 - \lambda)x_2 \in S, \quad \forall \lambda \in [0, 1]$$

convex set



nonconvex set

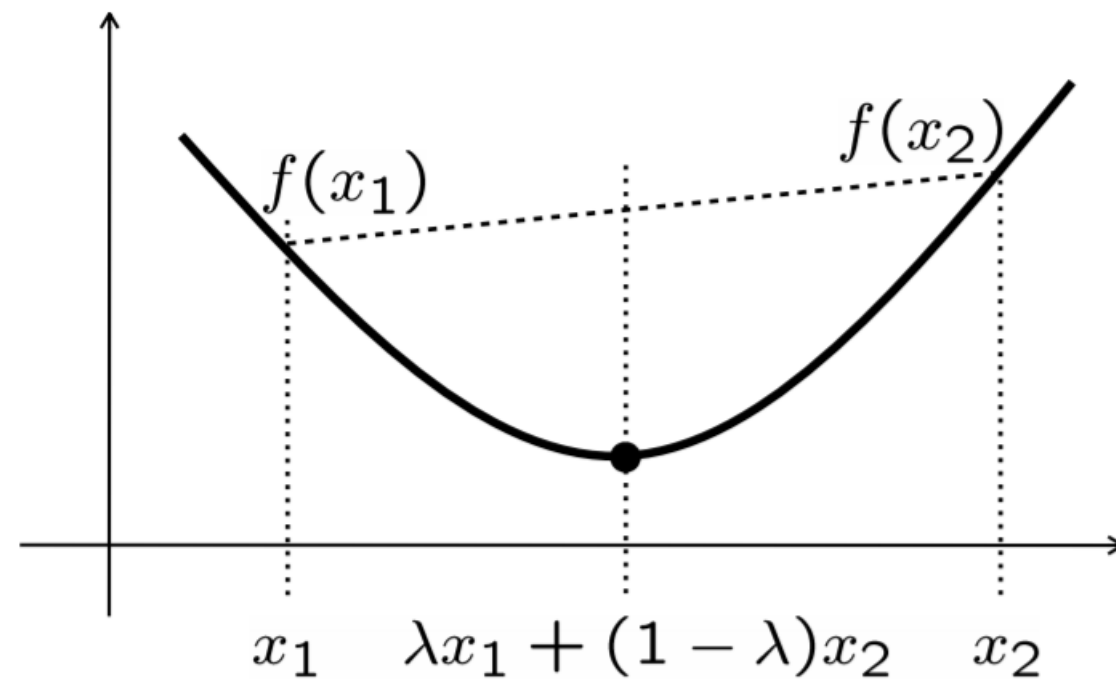


constrained optimization - convex function

- **definition:** a function $f : S \rightarrow R$ is convex if S is convex and

$$f(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda f(x_1) + (1 - \lambda)f(x_2),$$

$$\forall (x_1, x_2) \in S, \forall \lambda \in [0, 1] \in S$$



constrained optimization - convex problem

- the optimization problem

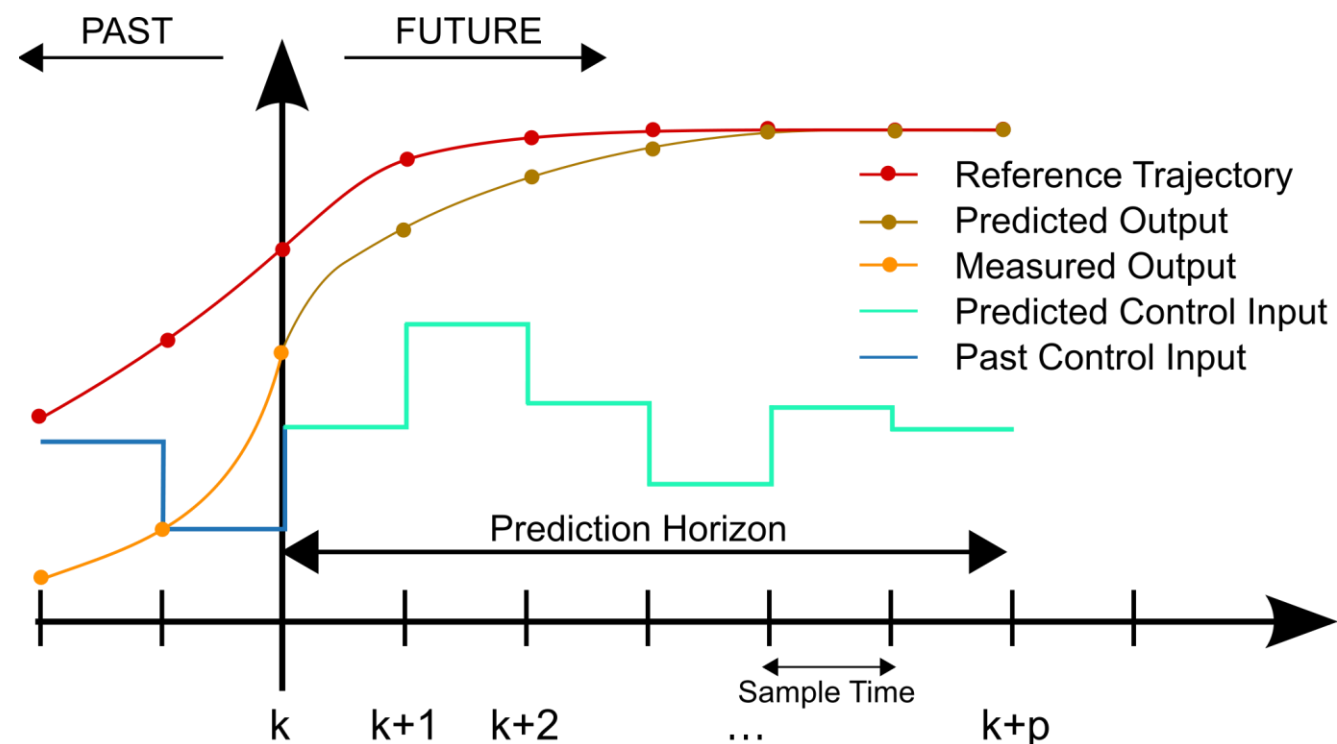
$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & x \in \mathcal{S} \end{aligned}$$

is a **convex** optimization problem if \mathcal{S} is a convex set and $f : \mathcal{S} \rightarrow \mathcal{R}$ is a convex function

- every convex optimization problem has a **unique** optimum
- for unconstrained optimization, the optimum can be found in **closed form**
- for constrained optimization, different solvers are available, e.g. **QPOASES** for quadratic problem (**QP**)

Model Predictive Control

- Model Predictive Control (MPC) is a control approach which can handle **state and input constraints**
- it solves at each time step an optimization problem over a fixed **prediction horizon**, computing an optimal sequence of control inputs
- only **the first input** is applied to the system, while the rest is discarded (receding horizon strategy)



linear MPC

- linear prediction model

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k \quad \Rightarrow \quad \mathbf{x}_k = \mathbf{A}^k \mathbf{x}_0 + \sum_{j=0}^{k-1} \mathbf{A}^j \mathbf{B}\mathbf{u}_{k-1-j}$$

- constraints

$$\mathbf{x}_{min} \leq \mathbf{x}_k \leq \mathbf{x}_{max}$$

$$\mathbf{u}_{min} \leq \mathbf{u}_k \leq \mathbf{u}_{max}$$

- optimization problem with a **quadratic** cost function

$$\min_{\mathbf{z} = [\mathbf{u}_0, \dots, \mathbf{u}_{N-1}]} J(\mathbf{x}, \mathbf{u}) = \mathbf{x}_N^T \mathbf{Q}_N \mathbf{x}_N + \sum_{k=0}^{N-1} \mathbf{x}_k^T \mathbf{Q} \mathbf{x}_k + \mathbf{u}_k^T \mathbf{R} \mathbf{u}_k$$

$$s.t. \quad \mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k \quad k = 0, \dots, N - 1$$

$$\mathbf{x}_{min} \leq \mathbf{x}_k \leq \mathbf{x}_{max} \quad k = 1, \dots, N$$

$$\mathbf{u}_{min} \leq \mathbf{u}_k \leq \mathbf{u}_{max} \quad k = 0, \dots, N - 1$$

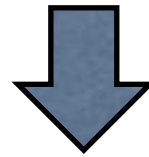
linear MPC - formulation

- condensed form

$$\begin{aligned}
 J(\mathbf{z}, \mathbf{x}_0) = & \mathbf{x}_0' \mathbf{Q} \mathbf{x}_0 + \underbrace{\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_{N-1} \\ \mathbf{x}_N \end{bmatrix}^T \begin{bmatrix} \mathbf{Q} & 0 & 0 & \dots & 0 \\ 0 & \mathbf{Q} & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & \mathbf{Q} & 0 \\ 0 & 0 & \dots & 0 & \mathbf{Q}_N \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_{N-1} \\ \mathbf{x}_N \end{bmatrix}}_{\bar{\mathbf{Q}}} \\
 & + \underbrace{\begin{bmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_{N-1} \end{bmatrix}^T \begin{bmatrix} \mathbf{R} & 0 & \dots & 0 \\ 0 & \mathbf{R} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \mathbf{R} \end{bmatrix} \begin{bmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_{N-1} \end{bmatrix}}_{\bar{\mathbf{R}}} \\
 \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_N \end{bmatrix} = & \underbrace{\begin{bmatrix} \mathbf{B} & 0 & \dots & 0 \\ \mathbf{AB} & \mathbf{B} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}^{N-1}\mathbf{B} & \mathbf{A}^{N-2}\mathbf{B} & \dots & \mathbf{B} \end{bmatrix}}_{\bar{\mathbf{S}}} \underbrace{\begin{bmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_{N-1} \end{bmatrix}}_{\mathbf{z}} + \underbrace{\begin{bmatrix} \mathbf{A} \\ \mathbf{A}^2 \\ \vdots \\ \mathbf{A}^N \end{bmatrix}}_{\bar{\mathbf{T}}} \mathbf{x}_0
 \end{aligned}$$

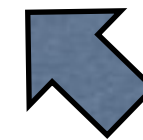
linear MPC - formulation

$$\begin{aligned}
 J(\mathbf{z}, \mathbf{x}_0) &= (\bar{\mathbf{S}}\mathbf{z} + \bar{\mathbf{T}}\mathbf{x}_0)^T \bar{\mathbf{Q}} (\bar{\mathbf{S}}\mathbf{z} + \bar{\mathbf{T}}\mathbf{x}_0) + \mathbf{z}^T \bar{\mathbf{R}}\mathbf{z} + \mathbf{x}_0^T \mathbf{Q}\mathbf{x}_0 \\
 &= \frac{1}{2}\mathbf{z}^T \mathbf{H} \mathbf{z} + \mathbf{x}_0^T \mathbf{F}^T \mathbf{z} + \frac{1}{2}\mathbf{x}_0^T \mathbf{Y} \mathbf{x}_0
 \end{aligned}$$



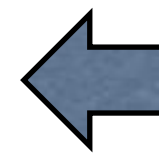
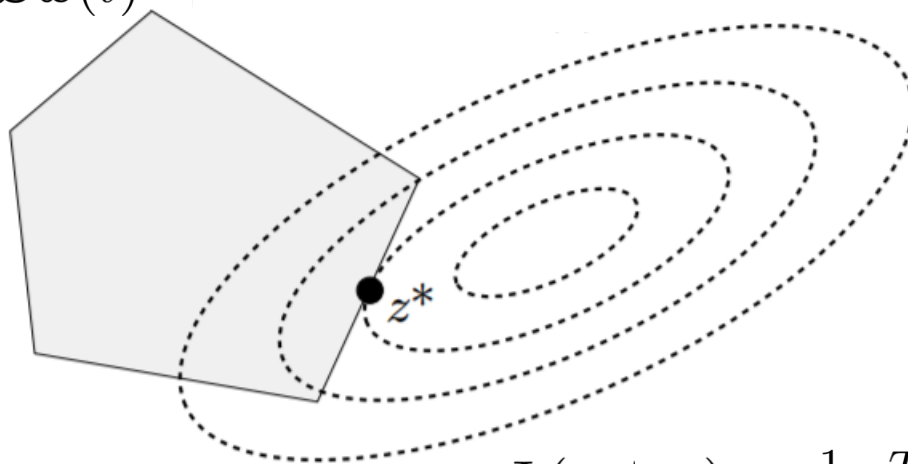
$$\begin{aligned}
 \min_{\mathbf{z} = [\mathbf{u}_0, \dots, \mathbf{u}_{N-1}]} \quad & J(\mathbf{z}, \mathbf{x}_0) = \frac{1}{2}\mathbf{z}^T \mathbf{H} \mathbf{z} + \mathbf{x}_0^T \mathbf{F}^T \mathbf{z} + \frac{1}{2}\mathbf{x}_0^T \mathbf{Y} \mathbf{x}_0
 \end{aligned}$$

$$\begin{aligned}
 \text{s.t.} \quad & \mathbf{x}_{min} \leq \mathbf{x}_k \leq \mathbf{x}_{max} \quad k = 1, \dots, N \\
 & \mathbf{u}_{min} \leq \mathbf{u}_k \leq \mathbf{u}_{max} \quad k = 0, \dots, N - 1
 \end{aligned}$$



$$\mathbf{G}\mathbf{z} \leq \mathbf{W} + \mathbf{S}\mathbf{x}(t)$$

$$\mathbf{G}\mathbf{z} \leq \mathbf{W} + \mathbf{S}\mathbf{x}(t)$$



constrained QP problem

$$J(\mathbf{z}, \mathbf{x}_0) = \frac{1}{2}\mathbf{z}^T \mathbf{H} \mathbf{z} + \mathbf{x}_0^T \mathbf{F}^T \mathbf{z} + \frac{1}{2}\mathbf{x}_0^T \mathbf{Y} \mathbf{x}_0$$

LQR and MPC: comparison

LQR

- infinite horizon
- explicit linear solution
- feedback matrix computed only once
- does not handle any constraints

Linear MPC

- limited horizon length
- in general no explicit solution
- needs to be recomputed at each control step
- handles constraints

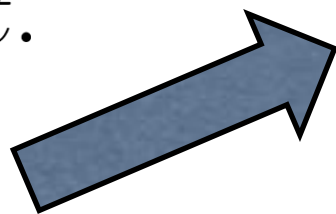
linear time varying MPC

- we can use MPC for **trajectory tracking**
- for linear systems, we can just formulate the cost function in order to penalize the deviation w.r.t the desired trajectory
- for nonlinear systems, we can replicate the procedure for LTV LQR, **linearizing** the model along the trajectory

$$\min_{z = [\hat{u}_0, \dots, \hat{u}_{N-1}]} J(\hat{x}, \hat{u}) = \hat{x}_N^T Q_N \hat{x}_N + \sum_{k=0}^{N-1} \hat{x}_k^T Q \hat{x}_k + \hat{u}_k^T R \hat{u}_k$$

s.t.

LTV system



$$\hat{x}_{k+1} = \hat{A}_k \hat{x}_k + B_k \hat{u}_k \quad k = 0, \dots, N - 1$$

$$\mathbf{x}_{min} \leq \mathbf{x}_k \leq \mathbf{x}_{max} \quad k = 1, \dots, N$$

$$\mathbf{u}_{min} \leq \mathbf{u}_k \leq \mathbf{u}_{max} \quad k = 0, \dots, N - 1$$

- the optimization problem remains a **QP**

from LTV MPC to Nonlinear MPC

- LTV MPC works only if we start close enough to the desired trajectory
- we can obtain better performance solving directly the nonlinear optimization problem, which will not be convex anymore
- multiple **local optima** may be present and convergence at the **global optimum** is not guaranteed
- Nonlinear MPC (NMPC) can be useful to deal with strong nonlinearities and/or nonlinear constraints/costs
- NMPC is more **computationally intensive** to solve compared to Linear and LTV MPC
- different optimization methods can be used, such as **Sequential Quadratic Programming** (SQP) and **Interior Point Method**

NMPC - SQP algorithm

- at each control step k , start with an initial guess

$$z^0 = [\mathbf{u}_k^0, \mathbf{u}_{k+1}^0, \dots, \mathbf{u}_{k+N-1}^0]$$

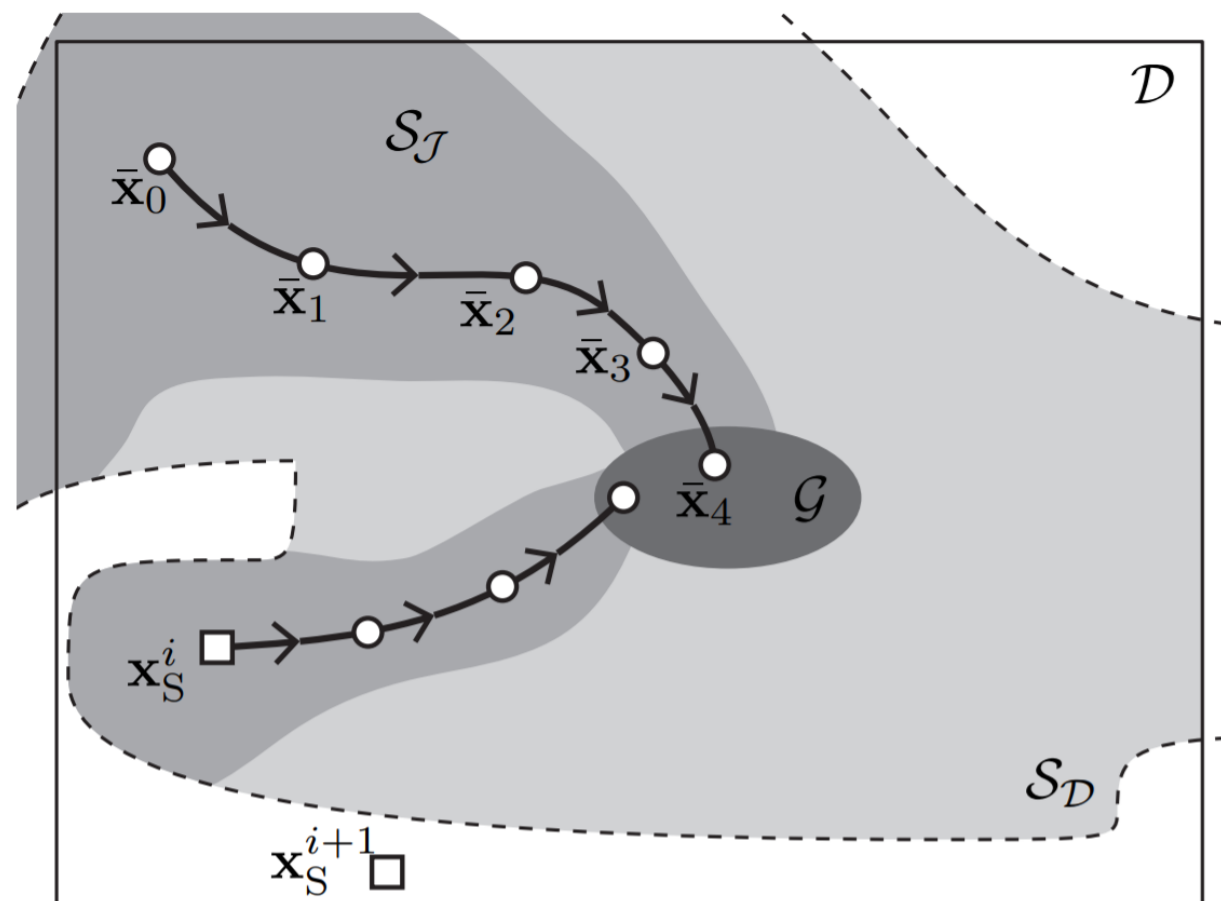
- for $h = 0$ to h_{max} repeat:
 1. simulate the nonlinear system from the current state under the inputs z^h , obtaining the state evolution $\mathbf{x}_{k+1}^h, \dots, \mathbf{x}_{k+N}^h$
 2. linearize the nonlinear system around the new state evolution and input z^h
 3. get z^* from the solution of the QP (as in LTV MPC)
 4. update the control input $z^{h+1} = z^h + z^*$ and set it as the new initial guess for the next iteration

case study - LQR-Tree

- **aim**: stabilize a nonlinear system from a **wide** range of different initial conditions
- **idea**: grow a randomized tree of stabilizing controllers that leads the system from any initial condition to the goal
- **steps**:
 - randomly explore a bounded set of initial states by tracing an open-loop trajectory leading to the goal (e.g. **nonlinear optimization with SQP**)
 - stabilize the trajectories by a feedback controller (**LTV LQR**)
 - approximate the set of states that can be stabilized to the goal tracking the planned trajectories (**funnel**)

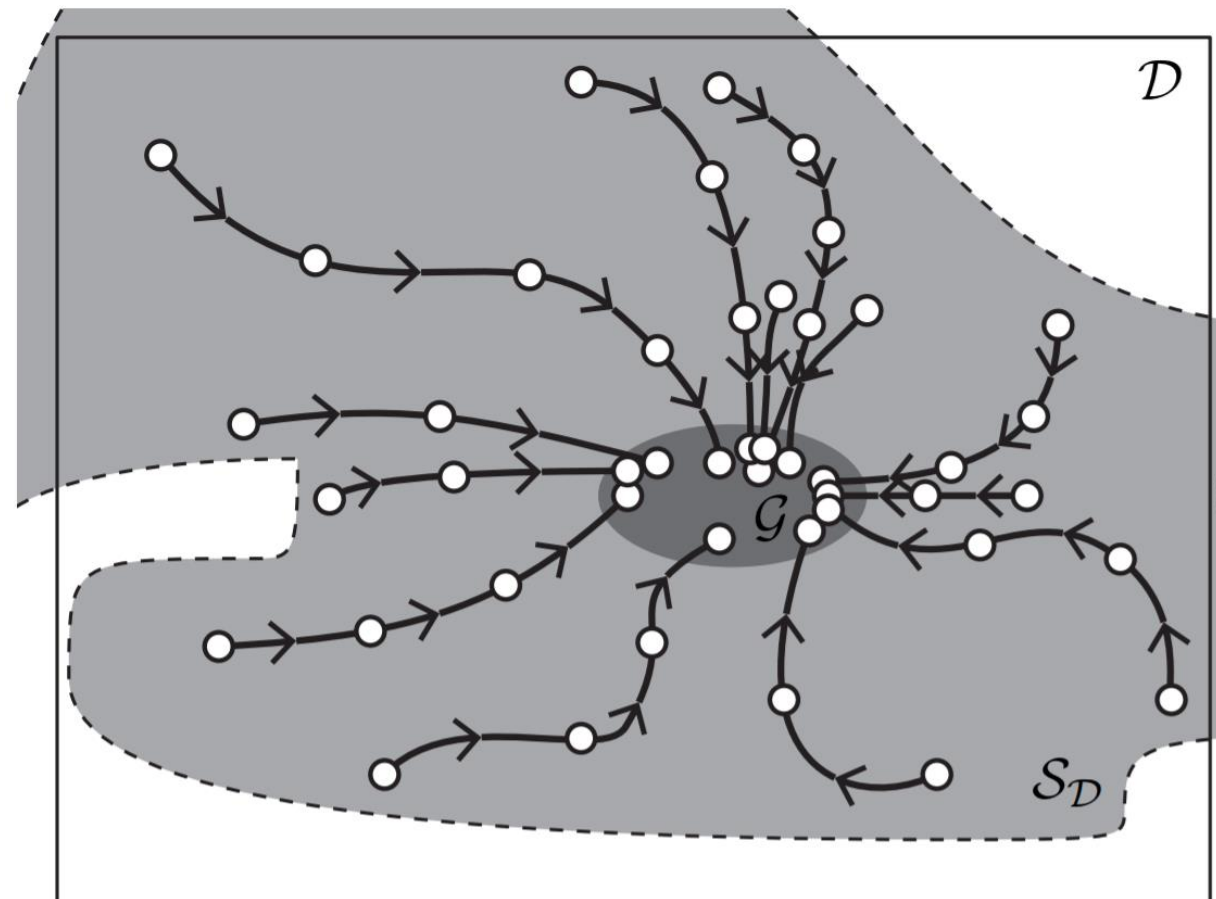
case study - LQR-Tree

- **funnel** - set of states around a trajectory that can be stabilized to a goal applying a feedback tracking controller (LTV LQR)



case study - LQR-Tree

- we want to add **offline** trajectories to the tree until the concatenation of their **funnel** covers all the possible initial states



- once the tree is created, we search **online** for a trajectory whose funnel contains the actual state of the system and apply the corresponding feedback tracking controller

Feedback-Motion-Planning with Simulation-Based LQR-Trees

Philipp Reist
Pascal Preiswerk
ETH Zurich

Russ Tedrake
MIT

"Selected Experiments with the Cart-Pole"
Multimedia Extension #2

The International Journal of Robotics Research