Data-driven Economic NMPC using Reinforcement Learning

Mario Zanon, Alberto Bemporad

Reinforcement Learning

- model-free
- **optimal for the actual system**
- $\approx$ sample-based stochastic optimal control
- learning can be slow, expensive, **unsafe**
- **no stability guarantee** (commonly based on DNN)

Reinforcement Learning

- model-free
- **optimal for the actual system**
- $\approx$ sample-based stochastic optimal control
- learning can be slow, expensive, **unsafe**
- **no stability guarantee** (commonly based on DNN)

(Economic) Model Predictive Control

- **optimal** for the **nominal model**
- **constraint satisfaction**
- can represent complex control policies
- **stability and recursive feasibility guarantees**

Reinforcement Learning

- model-free
- **optimal for the actual system**
- $\approx$ sample-based stochastic optimal control
- learning can be slow, expensive, **unsafe**
- **no stability guarantee** (commonly based on DNN)

(Economic) Model Predictive Control

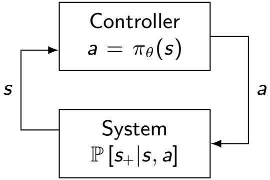
- **optimal** for the **nominal model**
- **constraint satisfaction**
- can represent complex control policies
- **stability and recursive feasibility guarantees**

Combine MPC and RL

- simple MPC formulations as proxy for complicated ones
- recover optimality, safety and stability for the true system

The Basics

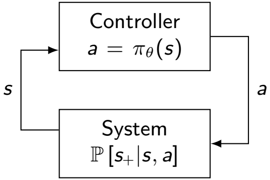
- state, action — $s,\ a$
- stochastic transition dynamics — $\mathbb{P}[s_+|s,a] \quad \Leftrightarrow \quad s_+ = f(s,a,w)$



The Basics

Assumption: the system is a Markov Decision Process (MDP)

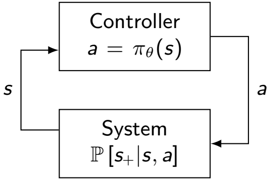
- state, action $s,\ a$
- stochastic transition dynamics $\mathbb{P}[s_+|s,a] \quad \Leftrightarrow \quad s_+ = f(s,a,w)$



The Basics

Assumption: the system is a Markov Decision Process (MDP)

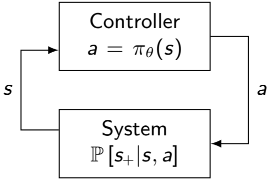
- state, action s, a
- stochastic transition dynamics $\mathbb{P}[s_+|s, a] \quad \Leftrightarrow \quad s_+ = f(s, a, w)$
- scalar reward / stage cost $L(s, a)$



The Basics

Assumption: the system is a Markov Decision Process (MDP)

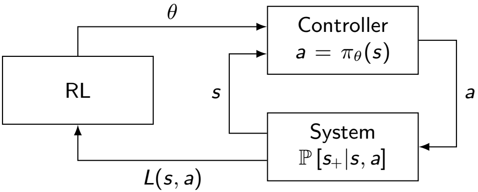
- state, action s, a
- stochastic transition dynamics $\mathbb{P}[s_+|s,a] \quad \Leftrightarrow \quad s_+ = f(s,a,w)$
- scalar reward / stage cost $L(s,a)$
- discount factor $0 < \gamma \leq 1$



The Basics

Assumption: the system is a Markov Decision Process (MDP)

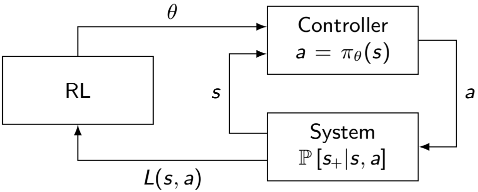
- state, action s, a
- stochastic transition dynamics $\mathbb{P}[s_+|s,a] \quad \Leftrightarrow \quad s_+ = f(s,a,w)$
- scalar reward / stage cost $L(s,a)$
- discount factor $0 < \gamma \leq 1$



The Basics

Assumption: the system is a Markov Decision Process (MDP)

- state, action — s, a
- stochastic transition dynamics — $\mathbb{P}[s_+|s,a] \quad \Leftrightarrow \quad s_+ = f(s,a,w)$
- scalar reward / stage cost — $L(s,a)$
- discount factor — $0 < \gamma \leq 1$



Goal:

- learn the optimal policy
- using no prior knowledge, observe
  - reward only

Main Concepts

Optimal policy:

Optimal value function:

Main Concepts

Optimal policy:

$$\pi_\star(s) = \arg\min_a L(s,a) + \gamma \mathbb{E}[V_\star(s_+) \mid s, a]$$

Optimal value function:

$$V_\star(s) = L(s, \pi_\star(s)) + \gamma \mathbb{E}[V_\star(s_+) \mid s, \pi_\star(s)]$$

Main Concepts

Optimal policy:

$$\pi_\star(s) = \arg\min_a L(s,a) + \gamma \mathbb{E}[V_\star(s_+) \,|\, s, a]$$

Optimal value function:

$$V_\star(s) = L(s, \pi_\star(s)) + \gamma \mathbb{E}[V_\star(s_+) \,|\, s, \pi_\star(s)]$$

If we know $V_\star$, we can compute $\pi_\star$

Main Concepts

Optimal policy:

$$\pi_\star(s) = \arg\min_a L(s,a) + \gamma \mathbb{E}[V_\star(s_+) \mid s, a]$$

Optimal value function:

$$V_\star(s) = L(s, \pi_\star(s)) + \gamma \mathbb{E}[V_\star(s_+) \mid s, \pi_\star(s)]$$

If we know $V_\star$, we can compute $\pi_\star$
but only if we know the model

Main Concepts

Optimal policy:

$$\pi_\star(s) = \arg\min_a L(s,a) + \gamma \mathbb{E}[V_\star(s_+) \,|\, s,a]$$

Optimal value function:

$$V_\star(s) = L(s,\pi_\star(s)) + \gamma \mathbb{E}[V_\star(s_+) \,|\, s,\pi_\star(s)]$$

If we know $V_\star$, we can compute $\pi_\star$
but only if we know the model

Optimal action-value function:

$$\begin{aligned} Q_\star(s,a) &= L(s,a) + \gamma \mathbb{E}[V_\star(s_+) \,|\, s,a] \\ &= L(s,a) + \gamma \mathbb{E}\left[\min_{a_+} Q_\star(s_+,a_+) \,\middle|\, s,a\right] \end{aligned}$$

Main Concepts

Optimal policy:

$$\pi_\star(s) = \arg\min_a L(s,a) + \gamma \mathbb{E}[V_\star(s_+) \,|\, s, a]$$

Optimal value function:

$$V_\star(s) = L(s, \pi_\star(s)) + \gamma \mathbb{E}[V_\star(s_+) \,|\, s, \pi_\star(s)]$$

If we know $V_\star$, we can compute $\pi_\star$
but only if we know the model

Optimal action-value function:

$$\begin{aligned} Q_\star(s,a) &= L(s,a) + \gamma \mathbb{E}[V_\star(s_+) \,|\, s, a] \\ &= L(s,a) + \gamma \mathbb{E}\left[\min_{a_+} Q_\star(s_+, a_+) \,\middle|\, s, a \right] \end{aligned}$$

Optimal policy:

$$\pi_\star(s) = \min_a Q_\star(s,a)$$

Main Concepts

Optimal policy:

$$\pi_\star(s) = \arg\min_a L(s,a) + \gamma \mathbb{E}[V_\star(s_+) \,|\, s, a]$$

Optimal value function:

$$V_\star(s) = L(s, \pi_\star(s)) + \gamma \mathbb{E}[V_\star(s_+) \,|\, s, \pi_\star(s)]$$

If we know $V_\star$, we can compute $\pi_\star$
but only if we know the model

Optimal action-value function:

$$\begin{aligned} Q_\star(s,a) &= L(s,a) + \gamma \mathbb{E}[V_\star(s_+) \,|\, s, a] \\ &= L(s,a) + \gamma \mathbb{E}\left[\min_{a_+} Q_\star(s_+, a_+) \,\middle|\, s, a\right] \end{aligned}$$

Optimal policy:

$$\pi_\star(s) = \min_a Q_\star(s,a)$$

If we know $Q_\star$, we know $\pi_\star$
(if we know how to minimise $Q_\star$)

Main Concepts

Optimal policy:

$$\pi_\star(s) = \arg\min_a L(s,a) + \gamma \mathbb{E}[V_\star(s_+) \,|\, s, a]$$

Optimal value function:

$$V_\star(s) = L(s, \pi_\star(s)) + \gamma \mathbb{E}[V_\star(s_+) \,|\, s, \pi_\star(s)]$$

If we know $V_\star$, we can compute $\pi_\star$
but only if we know the model

Optimal action-value function:

$$Q_\star(s,a) = L(s,a) + \gamma \mathbb{E}[V_\star(s_+) \,|\, s, a]$$
$$= L(s,a) + \gamma \mathbb{E}\left[\min_{a_+} Q_\star(s_+, a_+) \,\middle|\, s, a\right]$$

Optimal policy:

$$\pi_\star(s) = \min_a Q_\star(s,a)$$

If we know $Q_\star$, we know $\pi_\star$
(if we know how to minimise $Q_\star$)

LQR example:

P solves the Riccati equation

$$K_\star = (R + B^\top P B)^{-1}(S^\top + B^\top P A)$$
$$\pi_\star(s) = -K_\star s$$
$$V_\star(s) = s^\top P s + V_0$$

Main Concepts

Optimal policy:

$$\pi_\star(s) = \arg\min_a L(s,a) + \gamma \mathbb{E}[V_\star(s_+) \,|\, s, a]$$

Optimal value function:

$$V_\star(s) = L(s, \pi_\star(s)) + \gamma \mathbb{E}[V_\star(s_+) \,|\, s, \pi_\star(s)]$$

If we know $V_\star$, we can compute $\pi_\star$
but only if we know the model

Optimal action-value function:

$$Q_\star(s,a) = L(s,a) + \gamma \mathbb{E}[V_\star(s_+) \,|\, s, a]$$
$$= L(s,a) + \gamma \mathbb{E}\left[\min_{a_+} Q_\star(s_+, a_+) \,\middle|\, s, a \right]$$

Optimal policy:

$$\pi_\star(s) = \min_a Q_\star(s,a)$$

If we know $Q_\star$, we know $\pi_\star$
(if we know how to minimise $Q_\star$)

LQR example:

P solves the Riccati equation

$$K_\star = (R + B^\top PB)^{-1}(S^\top + B^\top PA)$$
$$\pi_\star(s) = -K_\star s$$
$$V_\star(s) = s^\top Ps + V_0$$

$$Q_\star(s,a) = \begin{bmatrix} s \\ a \end{bmatrix}^\top M \begin{bmatrix} s \\ a \end{bmatrix} + V_0$$

$$M = \begin{bmatrix} Q + A^\top PA & S + A^\top PB \\ S^\top + B^\top PA & R + B^\top PB \end{bmatrix}$$

$$K_\star = M_{aa}^{-1} M_{as}$$

Main Concepts

Optimal policy:

$$\pi_\star(s) = \arg\min_a L(s,a) + \gamma \mathbb{E}[V_\star(s_+) \,|\, s, a]$$

Optimal value function:

$$V_\star(s) = L(s, \pi_\star(s)) + \gamma \mathbb{E}[V_\star(s_+) \,|\, s, \pi_\star(s)]$$

If we know $V_\star$, we can compute $\pi_\star$
but only if we know the model

Optimal action-value function:

$$Q_\star(s,a) = L(s,a) + \gamma \mathbb{E}[V_\star(s_+) \,|\, s, a]$$
$$= L(s,a) + \gamma \mathbb{E}\left[\min_{a_+} Q_\star(s_+, a_+) \,\middle|\, s, a\right]$$

Optimal policy:

$$\pi_\star(s) = \min_a Q_\star(s,a)$$

If we know $Q_\star$, we know $\pi_\star$
(if we know how to minimise $Q_\star$)

LQR example:

P solves the Riccati equation

$$K_\star = (R + B^\top PB)^{-1}(S^\top + B^\top PA)$$
$$\pi_\star(s) = -K_\star s$$
$$V_\star(s) = s^\top P s + V_0$$

$$Q_\star(s,a) = \begin{bmatrix} s \\ a \end{bmatrix}^\top M \begin{bmatrix} s \\ a \end{bmatrix} + V_0$$

$$M = \begin{bmatrix} Q + A^\top PA & S + A^\top PB \\ S^\top + B^\top PA & R + B^\top PB \end{bmatrix}$$

$$K_\star = M_{aa}^{-1} M_{as}$$

If we learn M directly, we do not need a model!

Main Concepts

Optimal policy:

$$\pi_\star(s) = \arg\min_a L(s,a) + \gamma \mathbb{E}[V_\star(s_+) \,|\, s, a]$$

Optimal value function:

$$V_\star(s) = L(s, \pi_\star(s)) + \gamma \mathbb{E}[V_\star(s_+) \,|\, s, \pi_\star(s)]$$

If we know $V_\star$, we can compute $\pi_\star$
but only if we know the model

Optimal action-value function:

$$Q_\star(s,a) = L(s,a) + \gamma \mathbb{E}[V_\star(s_+) \,|\, s, a]$$

$$= L(s,a) + \gamma \mathbb{E}\left[\min_{a_+} Q_\star(s_+, a_+) \,\middle|\, s, a\right]$$

Optimal policy:

$$\pi_\star(s) = \min_a Q_\star(s,a)$$

If we know $Q_\star$, we know $\pi_\star$
(if we know how to minimise $Q_\star$)

LQR example:

P solves the Riccati equation

$$K_\star = (R + B^\top PB)^{-1}(S^\top + B^\top PA)$$

$$\pi_\star(s) = -K_\star s$$

$$V_\star(s) = s^\top Ps + V_0$$

$$Q_\star(s,a) = \begin{bmatrix} s \\ a \end{bmatrix}^\top M \begin{bmatrix} s \\ a \end{bmatrix} + V_0$$

$$M = \begin{bmatrix} Q + A^\top PA & S + A^\top PB \\ S^\top + B^\top PA & R + B^\top PB \end{bmatrix}$$

$$K_\star = M_{aa}^{-1} M_{as}$$

If we learn M directly, we do not need a model!

How can we evaluate $V_\star$ and $Q_\star$?

How does it work?

How does it work?

- Policy Evaluation
  - Monte Carlo
  - Temporal Difference

How does it work?

- Policy Evaluation
  - Monte Carlo
  - Temporal Difference
- Policy Optimization
  - Greedy policy updates
  - ϵ-greedy
  - Exploration vs Exploitation

How does it work?

- Policy Evaluation
  - Monte Carlo
  - Temporal Difference
- Policy Optimization
  - Greedy policy updates
  - ϵ-greedy
  - Exploration vs Exploitation
- Abstract / generalize
  - Curse of dimensionality
  - Function approximation

How does it work?

- Policy Evaluation
  - Monte Carlo
  - Temporal Difference
- Policy Optimization
  - Greedy policy updates
  - ϵ-greedy
  - Exploration vs Exploitation
- Abstract / generalize
  - Curse of dimensionality
  - Function approximation
- Q-learning

How does it work?

- Policy Evaluation
  - Monte Carlo
  - Temporal Difference
- Policy Optimization
  - Greedy policy updates
  - ϵ-greedy
  - Exploration vs Exploitation
- Abstract / generalize
  - Curse of dimensionality
  - Function approximation
- Q-learning
- Policy search

Policy Evaluation - Monte Carlo

Consider the discrete case first: only finitely many states s and actions a

Policy Evaluation - Monte Carlo

Consider the discrete case first: only finitely many states s and actions a
Given policy π, what are V_π, Q_π?

Policy Evaluation - Monte Carlo

Consider the discrete case first: only finitely many states s and actions a
Given policy π, what are V_π, Q_π?

$V_\pi(s)$:

Policy Evaluation - Monte Carlo

Consider the discrete case first: only finitely many states s and actions a
Given policy π, what are V_π, Q_π?

$V_\pi(s)$:

- pick random s, increase counter $N(s)$

Policy Evaluation - Monte Carlo

Consider the discrete case first: only finitely many states s and actions a
Given policy π, what are V_π, Q_π?

$V_\pi(s)$:

- pick random s, increase counter $N(s)$
- compute cost-to-go

$$C_i(s) = \sum_{k=0}^{\infty} \gamma^k L(s, \pi(s))$$

Policy Evaluation - Monte Carlo

Consider the discrete case first: only finitely many states s and actions a
Given policy π, what are V_π, Q_π?

$V_\pi(s)$:

- pick random s, increase counter $N(s)$
- compute cost-to-go
$$C_i(s) = \sum_{k=0}^{\infty} \gamma^k L(s, \pi(s))$$
- empirical expectation:
$$V(s) \approx \sum_{i=1}^{N(s)} \frac{C_i(s, a)}{N(s)}$$

Policy Evaluation - Monte Carlo

Consider the discrete case first: only finitely many states s and actions a
Given policy π, what are V_π, Q_π?

$V_\pi(s)$:

- pick random s, increase counter $N(s)$
- compute cost-to-go
  $$C_i(s) = \sum_{k=0}^{\infty} \gamma^k L(s, \pi(s))$$
- empirical expectation:
  $$V(s) \approx \sum_{i=1}^{N(s)} \frac{C_i(s,a)}{N(s)}$$

$Q(s,a)$:

- $N(s,a)$
- $C_i(s,a) =$
  $$L(s,a) + \sum_{k=1}^{\infty} \gamma^k L(s, \pi(s))$$
- empirical expectation:
  $$Q(s,a) \approx \sum_{i=1}^{N(s,a)} \frac{C_i(s,a)}{N(s,a)}$$

Policy Evaluation - Monte Carlo

Consider the discrete case first: only finitely many states s and actions a
Given policy π, what are V_π, Q_π?

$V_\pi(s)$:

- pick random s, increase counter $N(s)$
- compute cost-to-go
$$C_i(s) = \sum_{k=0}^{\infty} \gamma^k L(s, \pi(s))$$
- empirical expectation:
$$V(s) \approx \sum_{i=1}^{N(s)} \frac{C_i(s,a)}{N(s)}$$

$Q(s, a)$:

- $N(s, a)$
- $C_i(s, a) =$
$$L(s,a) + \sum_{k=1}^{\infty} \gamma^k L(s, \pi(s))$$
- empirical expectation:
$$Q(s,a) \approx \sum_{i=1}^{N(s,a)} \frac{C_i(s,a)}{N(s,a)}$$

Recursive formulation:

$$V(s) \leftarrow V(s) + \frac{1}{N(s)} \left(\sum C_i - V(s) \right)$$

Policy Evaluation - Monte Carlo

Consider the discrete case first: only finitely many states s and actions a
Given policy π, what are V_π, Q_π?

$V_\pi(s)$:

- pick random s, increase counter $N(s)$
- compute cost-to-go
$$C_i(s) = \sum_{k=0}^{\infty} \gamma^k L(s, \pi(s))$$
- empirical expectation:
$$V(s) \approx \sum_{i=1}^{N(s)} \frac{C_i(s,a)}{N(s)}$$

$Q(s,a)$:

- $N(s,a)$
- $C_i(s,a) =$
$$L(s,a) + \sum_{k=1}^{\infty} \gamma^k L(s, \pi(s))$$
- empirical expectation:
$$Q(s,a) \approx \sum_{i=1}^{N(s,a)} \frac{C_i(s,a)}{N(s,a)}$$

Recursive formulation:

$$V(s) \leftarrow V(s) + \frac{1}{N(s)} \left(\sum C_i - V(s) \right)$$

Alternative:

$$V(s) \leftarrow V(s) + \alpha \left(\sum C_i - V(s) \right)$$

Policy Evaluation - Temporal Difference

Remember: $V_\pi(s) = L(s, \pi(s)) + \gamma \mathbb{E}[V_\pi(s_+) \,|\, s, \pi(s)]$

Policy Evaluation - Temporal Difference

Remember: $V_\pi(s) = L(s,\pi(s)) + \gamma\mathbb{E}[V_\pi(s_+) \,|\, s,\pi(s)]$

Idea of TD(0):

$$V(s) \leftarrow V(s) + \alpha\Big(L(s,\pi(s)) + \gamma V(s_+) - V(s)\Big)$$

Policy Evaluation - Temporal Difference

Remember: $V_\pi(s) = L(s, \pi(s)) + \gamma \mathbb{E}[V_\pi(s_+) \mid s, \pi(s)]$

Idea of TD(0):

$$V(s) \leftarrow V(s) + \alpha \Big(L(s, \pi(s)) + \gamma V(s_+) - V(s) \Big)$$

- TD-target: $L(s, \pi(s)) + \gamma V(s_+)$ is a proxy for infinite-horizon cost c

Policy Evaluation - Temporal Difference

Remember: $V_\pi(s) = L(s, \pi(s)) + \gamma \mathbb{E}[V_\pi(s_+) \,|\, s, \pi(s)]$

Idea of TD(0):

$$V(s) \leftarrow V(s) + \alpha\Big(L(s, \pi(s)) + \gamma V(s_+) - V(s)\Big)$$

- TD-target: $L(s, \pi(s)) + \gamma V(s_+)$ is a proxy for infinite-horizon cost c
- TD-error $\delta = L(s, \pi(s)) + \gamma V(s_+) - V(s)$

Policy Evaluation - Temporal Difference

Remember: $V_\pi(s) = L(s, \pi(s)) + \gamma \mathbb{E}[V_\pi(s_+) \mid s, \pi(s)]$

Idea of TD(0):

$$V(s) \leftarrow V(s) + \alpha\Big(L(s, \pi(s)) + \gamma V(s_+) - V(s)\Big)$$

- TD-target: $L(s, \pi(s)) + \gamma V(s_+)$ is a proxy for infinite-horizon cost c
- TD-error $\delta = L(s, \pi(s)) + \gamma V(s_+) - V(s)$
- Sample-based dynamic programming

Policy Evaluation - Temporal Difference

Remember: $V_\pi(s) = L(s,\pi(s)) + \gamma\mathbb{E}[V_\pi(s_+)\,|\,s,\pi(s)]$

Idea of TD(0):

$$V(s) \leftarrow V(s) + \alpha\Big(L(s,\pi(s)) + \gamma V(s_+) - V(s)\Big)$$

- TD-target: $L(s,\pi(s)) + \gamma V(s_+)$ is a proxy for infinite-horizon cost c
- TD-error $\delta = L(s,\pi(s)) + \gamma V(s_+) - V(s)$
- Sample-based dynamic programming
- learn before the episode ends

Policy Evaluation - Temporal Difference

Remember: $V_\pi(s) = L(s, \pi(s)) + \gamma \mathbb{E}[V_\pi(s_+) \mid s, \pi(s)]$

Idea of TD(0):

$$V(s) \leftarrow V(s) + \alpha\Big(L(s, \pi(s)) + \gamma V(s_+) - V(s)\Big)$$

- TD-target: $L(s, \pi(s)) + \gamma V(s_+)$ is a proxy for infinite-horizon cost c
- TD-error $\delta = L(s, \pi(s)) + \gamma V(s_+) - V(s)$
- Sample-based dynamic programming
- learn before the episode ends
- very efficient in Markov environments

Policy Evaluation - Temporal Difference

Remember: $V_\pi(s) = L(s, \pi(s)) + \gamma \mathbb{E}[V_\pi(s_+) \mid s, \pi(s)]$

Idea of TD(0):

$$V(s) \leftarrow V(s) + \alpha\Big(L(s, \pi(s)) + \gamma V(s_+) - V(s)\Big)$$

- TD-target: $L(s, \pi(s)) + \gamma V(s_+)$ is a proxy for infinite-horizon cost c
- TD-error $\delta = L(s, \pi(s)) + \gamma V(s_+) - V(s)$
- Sample-based dynamic programming
- learn before the episode ends
- very efficient in Markov environments

We can do the same for the action-value function:

$$Q(s, a) \leftarrow Q(s, a) + \alpha\Big(L(s, a) + \gamma Q(s_+, \pi(s_+)) - Q(s, a)\Big)$$

Policy Evaluation - Temporal Difference

Remember: $V_\pi(s) = L(s, \pi(s)) + \gamma \mathbb{E}[V_\pi(s_+) \,|\, s, \pi(s)]$

Idea of TD(0):

$$V(s) \leftarrow V(s) + \alpha\Big(L(s, \pi(s)) + \gamma V(s_+) - V(s)\Big)$$

- TD-target: $L(s, \pi(s)) + \gamma V(s_+)$ is a proxy for infinite-horizon cost c
- TD-error $\delta = L(s, \pi(s)) + \gamma V(s_+) - V(s)$
- Sample-based dynamic programming
- learn before the episode ends
- very efficient in Markov environments

We can do the same for the action-value function:

$$Q(s, a) \leftarrow Q(s, a) + \alpha\Big(L(s, a) + \gamma Q(s_+, \pi(s_+)) - Q(s, a)\Big)$$

We can evaluate V_π and Q_π, but how can we optimize them?

Learning

Learning

Greedy policy improvement

- model-based: $\pi'(s) = \arg\min_a \ L(s,a) + \gamma \mathbb{E}\left[V(s_+) | s, a\right]$
- model-free: $\pi'(s) = \arg\min_a \ Q(s,a)$

Learning

Greedy policy improvement

- model-based: $\pi'(s) = \arg\min_a \; L(s,a) + \gamma \mathbb{E}\left[V(s_+) | s, a\right]$
- model-free: $\pi'(s) = \arg\min_a \; Q(s,a)$

Problem:

- keep acting on-policy, i.e., $a = \pi(s)$
- how to ensure enough exploration?

Learning

Greedy policy improvement

- model-based: $\pi'(s) = \arg\min_a \; L(s,a) + \gamma \mathbb{E}\left[V(s_+) | s, a\right]$
- model-free: $\pi'(s) = \arg\min_a \; Q(s,a)$

Problem:

- keep acting on-policy, i.e., $a = \pi(s)$
- how to ensure enough exploration?

Simplest idea: ϵ-greedy:

$$\pi(s) = \begin{cases} \arg\max_a \; Q(s,a) & \text{with } p = 1-\epsilon \\ a_{\mathcal{U}\{1,n_a\}} & \text{with } p = \epsilon \end{cases}$$

Theorem

For any ϵ-greedy policy π, the ϵ-greedy policy π' is an improvement, i.e., $V_{\pi'}(s) \leq V_\pi(s)$

Learning

Greedy policy improvement

- model-based: $\pi'(s) = \arg\min_a \; L(s,a) + \gamma \mathbb{E}\left[V(s_+)|s,a\right]$
- model-free: $\pi'(s) = \arg\min_a \; Q(s,a)$

Problem:

- keep acting on-policy, i.e., $a = \pi(s)$
- how to ensure enough exploration?

Simplest idea: ϵ-greedy:

$$\pi(s) = \begin{cases} \arg\max_a \; Q(s,a) & \text{with } p = 1-\epsilon \\ a_{\mathcal{U}\{1,n_a\}} & \text{with } p = \epsilon \end{cases}$$

Theorem

For any ϵ-greedy policy π, the ϵ-greedy policy π' is an improvement, i.e., $V_{\pi'}(s) \leq V_{\pi}(s)$

In order to get optimality we need to be GLIE

Greedy in the limit with infinite exploration, e.g., ϵ-greedy with $\epsilon \to 0$

Q-Learning (basic version)

Q-Learning (basic version)

Update the action-value function as follows:

$$\delta \leftarrow L(s,a) + \gamma \min_{a_+} Q(s_+, a_+) - Q(s,a)$$

$$Q(s,a) \leftarrow Q(s,a) + \alpha\delta$$

Q-Learning (basic version)

Update the action-value function as follows:

$$\delta \leftarrow L(s,a) + \gamma \min_{a_+} Q(s_+, a_+) - Q(s,a)$$
$$Q(s,a) \leftarrow Q(s,a) + \alpha\delta$$

Curse of Dimensionality

- in general: too many state-action pairs
- need to generalize / extrapolate

Q-Learning (basic version)

Update the action-value function as follows:

$$\delta \leftarrow L(s,a) + \gamma \min_{a_+} Q(s_+, a_+) - Q(s,a)$$
$$Q(s,a) \leftarrow Q(s,a) + \alpha\delta$$

Curse of Dimensionality

- in general: too many state-action pairs
- need to generalize / extrapolate

Function Approximation

Features $\phi(s,a)$ and weights θ yield $Q_\theta(s,a) = \theta^\top \phi(s,a)$. Weights update

$$\delta \leftarrow L(s,a) + \gamma \min_{a_+} Q_\theta(s_+, a_+) - Q_\theta(s,a)$$
$$\theta \leftarrow \theta + \alpha\delta\nabla_\theta Q_\theta(s,a)$$

- this is a linear function approximation
- deep neural networks are nonlinear

Policy Search

Q-learning: fits $\mathbb{E}\left(Q_\theta - Q_\star\right)^2$

- no guarantee that π_θ is close to $\pi_\star$
- what we want is $\min_\theta J(\pi_\theta) := \mathbb{E}\sum_{k=0}^{\infty}\gamma^k L(s_k, \pi_\theta(s_k))$

Policy Search parametrizes π_θ and directly minimizes $J(\pi_\theta)$:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J$$

- model-based: construct a model f and simulate forward in time:

$$J \approx \frac{1}{B}\sum_{i=0}^{B}\sum_{k=0}^{N}\gamma^k L\left(s^{(i)}, \pi_\theta\left(s^{(i)}\right)\right), \qquad s_+^{(i)} = f\left(s^{(i)}, \pi_\theta\left(s^{(i)}\right), w\right)$$

- actor-critic (model-free): $A(s,a) = Q(s,a) - V(s)$

$$\begin{aligned} \text{Deterministic policy :} \qquad & \nabla_\theta J = \nabla_\theta \pi_\theta \nabla_a A_{\pi_\theta}, \\ \text{Stochastic policy :} \qquad & \nabla_\theta J = \nabla_\theta \log \pi_\theta A_{\pi_\theta}, \end{aligned}$$

  - use, e.g., Q-learning for A, or V
  - if you are careful: convergence to local min of $J(\pi_\theta)$

Main issues

- Can we guarantee anything?
  - Safety
  - Stability
  - Optimality

Main issues

- Can we guarantee anything?
  - Safety
  - Stability
  - Optimality
- Learning is potentially
  - Dangerous
  - Expensive
  - Slow

Main issues

- Can we guarantee anything?
  - Safety
  - Stability
  - Optimality
- Learning is potentially
  - Dangerous
  - Expensive
  - Slow

Prior knowledge is valuable

- Why learning from scratch?
- Can we use RL to improve existing controllers?
  - Retain stability and safety
  - Improve performance

Optimal Control - Model Predictive Control (MPC)

- use a model to predict the future
- constraint enforcement
- performance and stability guarantees

$$
\begin{aligned}
\min_{x,u} \quad & V^{\mathrm{f}}(x_N) + \sum_{k=0}^{N-1} L(x,u) \\
\text{s.t.} \quad & x_0 = s, \\
& x_{k+1} = f(x_k, u_k), \\
& h(x_k, u_k) \leq 0, \\
& x_N \in \mathbb{X}^{\mathrm{f}}.
\end{aligned}
$$

Optimal Control - Model Predictive Control (MPC)

- use a model to predict the future
- constraint enforcement
- performance and stability guarantees

$$
\begin{aligned}
\min_{x,u} \quad & V^{\mathrm{f}}(x_N) + \sum_{k=0}^{N-1} L(x,u) \\
\text{s.t.} \quad & x_0 = s, \\
& x_{k+1} = f(x_k, u_k), \\
& h(x_k, u_k) \leq 0, \\
& x_N \in \mathbb{X}^{\mathrm{f}}.
\end{aligned}
$$

Optimality hinges on

- quality of the model (how descriptive)
- system identification (estimate the correct model parameters)

Optimal Control - Model Predictive Control (MPC)

- use a model to predict the future
- constraint enforcement
- performance and stability guarantees

$$\begin{aligned} \min_{x,u} \quad & V^{\mathrm{f}}(x_N) + \sum_{k=0}^{N-1} L(x,u) \\ \mathrm{s.t.} \quad & x_0 = s, \\ & x_{k+1} = f(x_k, u_k), \\ & h(x_k, u_k) \leq 0, \\ & x_N \in \mathbb{X}^{\mathrm{f}}. \end{aligned}$$

Optimality hinges on

- quality of the model (how descriptive)
- system identification (estimate the correct model parameters)

but then, if the model is not “perfect”

Optimal Control - Model Predictive Control (MPC)

- use a model to predict the future
- constraint enforcement
- performance and stability guarantees

$$\begin{aligned} \min_{x,u} \quad & V^{\mathrm{f}}(x_N) + \sum_{k=0}^{N-1} L(x,u) \\ \text{s.t.} \quad & x_0 = s, \\ & x_{k+1} = f(x_k, u_k), \\ & h(x_k, u_k) \leq 0, \\ & x_N \in \mathbb{X}^{\mathrm{f}}. \end{aligned}$$

Optimality hinges on

- quality of the model (how descriptive)
- system identification (estimate the correct model parameters)

but then, if the model is not "perfect"

- can we recover optimality through learning?

Optimal Control - Model Predictive Control (MPC)

- use a model to predict the future
- constraint enforcement
- performance and stability guarantees

$$\begin{aligned}\min_{x,u}\quad & V^{\mathrm{f}}(x_N) + \sum_{k=0}^{N-1} L(x,u)\\ \text{s.t.}\quad & x_0 = s,\\ & x_{k+1} = f(x_k,u_k),\\ & h(x_k,u_k) \leq 0,\\ & x_N \in \mathbb{X}^{\mathrm{f}}.\end{aligned}$$

Optimality hinges on

- quality of the model (how descriptive)
- system identification (estimate the correct model parameters)

but then, if the model is not "perfect"

- can we recover optimality through learning?
- use MPC as function approximator

Optimal Control - Model Predictive Control (MPC)

- use a model to predict the future
- constraint enforcement
- performance and stability guarantees

$$\begin{aligned} V_\theta(s) := \min_{x,u} \quad & V_\theta^{\mathrm{f}}(x_N) + \sum_{k=0}^{N-1} L_\theta(x,u) \\ \text{s.t.} \quad & x_0 = s, \\ & x_{k+1} = f_\theta(x_k,u_k), \\ & h_\theta(x_k,u_k) \le 0, \\ & x_N \in \mathbb{X}_\theta^{\mathrm{f}}. \end{aligned}$$

Optimality hinges on

- quality of the model (how descriptive)
- system identification (estimate the correct model parameters)

but then, if the model is not "perfect"

- can we recover optimality through learning?
- use MPC as function approximator

Optimal Control - Model Predictive Control (MPC)

- use a model to predict the future
- constraint enforcement
- performance and stability guarantees

$$
\begin{aligned}
\pi_\theta(s) := \arg\min_{x,u} \quad & V_\theta^{\mathrm{f}}(x_N) + \sum_{k=0}^{N-1} L_\theta(x,u) \\
\text{s.t.} \quad & x_0 = s, \\
& x_{k+1} = f_\theta(x_k, u_k), \\
& h_\theta(x_k, u_k) \leq 0, \\
& x_N \in \mathbb{X}_\theta^{\mathrm{f}}.
\end{aligned}
$$

Optimality hinges on

- quality of the model (how descriptive)
- system identification (estimate the correct model parameters)

but then, if the model is not “perfect”

- can we recover optimality through learning?
- use MPC as function approximator

Optimal Control - Model Predictive Control (MPC)

- use a model to predict the future
- constraint enforcement
- performance and stability guarantees

$$
\begin{aligned}
Q_\theta(s,a) := \min_{x,u} \quad & V_\theta^{\mathrm{f}}(x_N) + \sum_{k=0}^{N-1} L_\theta(x,u) \\
\text{s.t.} \quad & x_0 = s, \quad u_0 = a, \\
& x_{k+1} = f_\theta(x_k, u_k), \\
& h_\theta(x_k, u_k) \leq 0, \\
& x_N \in \mathbb{X}_\theta^{\mathrm{f}}.
\end{aligned}
$$

Optimality hinges on

- quality of the model (how descriptive)
- system identification (estimate the correct model parameters)

but then, if the model is not "perfect"

- can we recover optimality through learning?
- use MPC as function approximator

Learn the true action-value function with MPC

- inaccurate MPC model $\mathbb{P}[\hat{s}_+|s,a] \neq \mathbb{P}[s_+|s,a]$

Learn the true action-value function with MPC

- inaccurate MPC model $\mathbb{P}[\hat{s}_+|s,a] \neq \mathbb{P}[s_+|s,a]$

Theorem [Gros, Zanon TAC2020]

Assume that the MPC parametrization (through θ) is rich enough.
Then, the exact $V_\star$, $Q_\star$, $\pi_\star$ are recovered.

Learn the true action-value function with MPC

- inaccurate MPC model $\mathbb{P}[\hat{s}_+|s, a] \neq \mathbb{P}[s_+|s, a]$

Theorem [Gros, Zanon TAC2020]

Assume that the MPC parametrization (through θ) is rich enough.
Then, the exact $V_\star$, $Q_\star$, $\pi_\star$ are recovered.

- SysId and RL are "orthogonal"
- RL cannot learn the true model

Learn the true action-value function with MPC

- inaccurate MPC model $\mathbb{P}[\hat{s}_+|s,a] \neq \mathbb{P}[s_+|s,a]$

Theorem [Gros, Zanon TAC2020]

Assume that the MPC parametrization (through θ) is rich enough.
Then, the exact $V_\star$, $Q_\star$, $\pi_\star$ are recovered.

- SysId and RL are "orthogonal"
- RL cannot learn the true model

$$\begin{aligned} \min_{x,u} \quad & V_\theta^{\mathrm{f}}(x_N) + \sum_{k=0}^{N-1} L_\theta(x,u) \\ \text{s.t.} \quad & x_0 = s, \\ & x_{k+1} = f_\theta(x_k,u_k), \\ & h_\theta(x_k,u_k) \leq 0, \\ & x_N \in \mathbb{X}_\theta^{\mathrm{f}}. \end{aligned}$$

Algorithmic framework:

[Zanon, Gros, Bemporad ECC2019]

- Enforce $L_\theta, V_\theta^{\mathrm{f}} \succ 0$
- Can use condensed MPC formulation
- Can use globalization techniques

Economic MPC and RL

- If $L_\theta(s,a) \succ 0$, then $V_\theta(s) \succ 0$ is a Lyapunov function
- In RL we can have $L(s,a) \not\succ 0 \Rightarrow V_\star(s) \not\succ 0$

Economic MPC and RL

- If $L_\theta(s,a) \succ 0$, then $V_\theta(s) \succ 0$ is a Lyapunov function
- In RL we can have $L(s,a) \not\succ 0 \Rightarrow V_\star(s) \not\succ 0$

ENMPC theory:

- if we rotate the cost we can have $V_\star(s) = \lambda(s) + V_\theta(s)$
- e.g., if $0 \prec L_\theta(s,a) = L(s,a) - \lambda(s) + \lambda(f(s,a))$

Economic MPC and RL

- If $L_\theta(s,a) \succ 0$, then $V_\theta(s) \succ 0$ is a Lyapunov function
- In RL we can have $L(s,a) \not\succ 0 \Rightarrow V_\star(s) \not\succ 0$

ENMPC theory:

- if we rotate the cost we can have $V_\star(s) = \lambda(s) + V_\theta(s)$
- e.g., if $0 \prec L_\theta(s,a) = L(s,a) - \lambda(s) + \lambda(f(s,a))$

Therefore, use

$$
\begin{aligned}
V_\theta(s) = \min_{x,u} \quad & \lambda_\theta(s) + V_\theta^{\mathrm{f}}(x_N) + \sum_{k=0}^{N-1} L_\theta(x,u) \\
\mathrm{s.t.} \quad & x_0 = s, \\
& x_{k+1} = f_\theta(x_k, u_k), \\
& h_\theta(x_k, u_k) \leq 0, \\
& x_N \in \mathbb{X}_\theta^{\mathrm{f}}.
\end{aligned}
$$

Economic MPC and RL

- If $L_\theta(s,a) \succ 0$, then $V_\theta(s) \succ 0$ is a Lyapunov function
- In RL we can have $L(s,a) \not\succ 0 \Rightarrow V_\star(s) \not\succ 0$

ENMPC theory:

- if we rotate the cost we can have $V_\star(s) = \lambda(s) + V_\theta(s)$
- e.g., if $0 \prec L_\theta(s,a) = L(s,a) - \lambda(s) + \lambda(f(s,a))$

Therefore, use

$$
\begin{aligned}
V_\theta(s) = \min_{x,u} \quad & \lambda_\theta(s) + V^{\mathrm{f}}_\theta(x_N) + \sum_{k=0}^{N-1} L_\theta(x,u) \\
\text{s.t.} \quad & x_0 = s, \\
& x_{k+1} = f_\theta(x_k, u_k), \\
& h_\theta(x_k, u_k) \le 0, \\
& x_N \in \mathbb{X}^{\mathrm{f}}_\theta.
\end{aligned}
$$

Enforce stability with $L_\theta(s,a) \succ 0$ and learn also $\lambda_\theta(s)$

Evaporation Process

Evaporation Process

Model and cost

$$\begin{bmatrix} \dot{X}_2 \\ \dot{P}_2 \end{bmatrix} = f\left(\begin{bmatrix} X_2 \\ P_2 \end{bmatrix}, \begin{bmatrix} P_{100} \\ F_{200} \end{bmatrix} \right), \qquad \ell(x, u) = \text{something complicated.}$$

Evaporation Process

Model and cost

$$\begin{bmatrix} \dot{X}_2 \\ \dot{P}_2 \end{bmatrix} = f\left(\begin{bmatrix} X_2 \\ P_2 \end{bmatrix}, \begin{bmatrix} P_{100} \\ F_{200} \end{bmatrix} \right), \qquad \ell(x, u) = \text{something complicated.}$$

Bounds

$$X_2 \geq 25\,\%, \qquad 40\,\mathrm{kPa} \leq P_2 \leq 80\,\mathrm{kPa},$$
$$P_{100} \leq 400\,\mathrm{kPa}, \qquad F_{200} \leq 400\mathrm{kg/min}.$$

Evaporation Process

Model and cost

$$\begin{bmatrix} \dot{X}_2 \\ \dot{P}_2 \end{bmatrix} = f\left(\begin{bmatrix} X_2 \\ P_2 \end{bmatrix}, \begin{bmatrix} P_{100} \\ F_{200} \end{bmatrix} \right), \qquad \ell(x, u) = \text{something complicated.}$$

Bounds

$$X_2 \geq 25\,\%, \qquad 40\,\mathrm{kPa} \leq P_2 \leq 80\,\mathrm{kPa},$$

$$P_{100} \leq 400\,\mathrm{kPa}, \qquad F_{200} \leq 400\mathrm{kg/min}.$$

Nominal optimal steady state

$$\begin{bmatrix} X_2 \\ P_2 \end{bmatrix} = \begin{bmatrix} 25\,\% \\ 49.743\,\mathrm{kPa} \end{bmatrix}, \qquad \begin{bmatrix} P_{100} \\ F_{200} \end{bmatrix} = \begin{bmatrix} 191.713\,\mathrm{kPa} \\ 215.888\,\mathrm{kg/min} \end{bmatrix}.$$

Evaporation Process

Model and cost

$$\begin{bmatrix} \dot{X}_2 \\ \dot{P}_2 \end{bmatrix} = f\left(\begin{bmatrix} X_2 \\ P_2 \end{bmatrix}, \begin{bmatrix} P_{100} \\ F_{200} \end{bmatrix}\right), \qquad \ell(x,u) = \text{something complicated.}$$

Bounds

$$X_2 \geq 25\,\%, \qquad 40\,\mathrm{kPa} \leq P_2 \leq 80\,\mathrm{kPa},$$
$$P_{100} \leq 400\,\mathrm{kPa}, \qquad F_{200} \leq 400 \mathrm{kg/min}.$$

Nominal optimal steady state

$$\begin{bmatrix} X_2 \\ P_2 \end{bmatrix} = \begin{bmatrix} 25\,\% \\ 49.743\,\mathrm{kPa} \end{bmatrix}, \qquad \begin{bmatrix} P_{100} \\ F_{200} \end{bmatrix} = \begin{bmatrix} 191.713\,\mathrm{kPa} \\ 215.888\,\mathrm{kg/min} \end{bmatrix}.$$

Nominal Economic MPC gain:

- large in the nominal case
- about 1.5 % in the stochastic case: cannot even guarantee to have any

Evaporation Process

Reinforcement learning based on

$$\min_{x,u,\sigma} \overbrace{x_0^\top H_\lambda x_0 + h_\lambda^\top x_0 + c_\lambda}^{\lambda(x_0)} + \gamma^N \Big(\overbrace{x_N^\top H_{V^\mathrm{f}} x_N + h_{V^\mathrm{f}}^\top x_N + c_{V^\mathrm{f}}}^{V^\mathrm{f}(x_N)} \Big)$$
$$+ \sum_{k=0}^{N-1} \gamma^k \Big(\underbrace{\begin{bmatrix} x_k \\ u_k \end{bmatrix}^\top H_\ell \begin{bmatrix} x_k \\ u_k \end{bmatrix} + h_\ell^\top \begin{bmatrix} x_k \\ u_k \end{bmatrix} + c_\ell + \sigma_k^\top H_\sigma^\top \sigma_k + h_\sigma^\top \sigma_k}_{\ell(x_k, u_k, \sigma_k)} \Big)$$

$$\begin{aligned} \text{s.t.} \quad & x_0 = s, \\ & x_{k+1} = f(x_k, u_k) + c_f, \\ & u_\mathrm{l} \le u_k \le u_\mathrm{u}, \\ & x_\mathrm{l} - \sigma_k^\mathrm{l} \le x_k \le x_\mathrm{u} + \sigma_k^\mathrm{u}. \end{aligned}$$

Evaporation Process

Reinforcement learning based on

$$
\begin{aligned}
\min_{x,u,\sigma} \quad & \overbrace{x_0^\top H_\lambda x_0 + h_\lambda^\top x_0 + c_\lambda}^{\lambda(x_0)} + \gamma^N \Big(\overbrace{x_N^\top H_{V^\mathrm{f}} x_N + h_{V^\mathrm{f}}^\top x_N + c_{V^\mathrm{f}}}^{V^\mathrm{f}(x_N)} \Big) \\
& + \sum_{k=0}^{N-1} \gamma^k \Big(\underbrace{\begin{bmatrix} x_k \\ u_k \end{bmatrix}^\top H_\ell \begin{bmatrix} x_k \\ u_k \end{bmatrix} + h_\ell^\top \begin{bmatrix} x_k \\ u_k \end{bmatrix} + c_\ell + \sigma_k^\top H_\sigma^\top \sigma_k + h_\sigma^\top \sigma_k}_{\ell(x_k, u_k, \sigma_k)} \Big) \\
\text{s.t.} \quad & x_0 = s, \\
& x_{k+1} = f(x_k, u_k) + c_f, \\
& u_\mathrm{l} \le u_k \le u_\mathrm{u}, \\
& x_\mathrm{l} - \sigma_k^\mathrm{l} \le x_k \le x_\mathrm{u} + \sigma_k^\mathrm{u}.
\end{aligned}
$$

Parameters to learn: $\theta = \{H_\lambda, h_\lambda, c_\lambda, H_{V^\mathrm{f}}, h_{V^\mathrm{f}}, c_{V^\mathrm{f}}, H_\ell, h_\ell, c_\ell, c_f, x_\mathrm{l}, x_\mathrm{u}\}$

Evaporation Process

Reinforcement learning based on

$$\min_{x,u,\sigma} \overbrace{x_0^\top H_\lambda x_0 + h_\lambda^\top x_0 + c_\lambda}^{\lambda(x_0)} + \gamma^N \Big(\overbrace{x_N^\top H_{V^\mathrm{f}} x_N + h_{V^\mathrm{f}}^\top x_N + c_{V^\mathrm{f}}}^{V^\mathrm{f}(x_N)} \Big)$$
$$+ \sum_{k=0}^{N-1} \gamma^k \Big(\underbrace{\begin{bmatrix} x_k \\ u_k \end{bmatrix}^\top H_\ell \begin{bmatrix} x_k \\ u_k \end{bmatrix} + h_\ell^\top \begin{bmatrix} x_k \\ u_k \end{bmatrix} + c_\ell + \sigma_k^\top H_\sigma^\top \sigma_k + h_\sigma^\top \sigma_k}_{\ell(x_k, u_k, \sigma_k)} \Big)$$

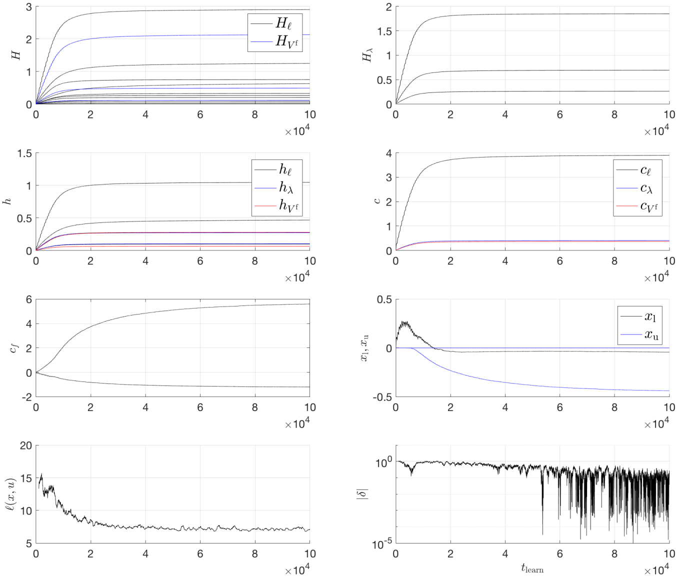
$$\begin{aligned} \text{s.t.} \quad & x_0 = s, \\ & x_{k+1} = f(x_k, u_k) + c_f, \\ & u_\mathrm{l} \le u_k \le u_\mathrm{u}, \\ & x_\mathrm{l} - \sigma_k^\mathrm{l} \le x_k \le x_\mathrm{u} + \sigma_k^\mathrm{u}. \end{aligned}$$

Parameters to learn: $\theta = \{H_\lambda, h_\lambda, c_\lambda, H_{V^\mathrm{f}}, h_{V^\mathrm{f}}, c_{V^\mathrm{f}}, H_\ell, h_\ell, c_\ell, c_f, x_\mathrm{l}, x_\mathrm{u}\}$

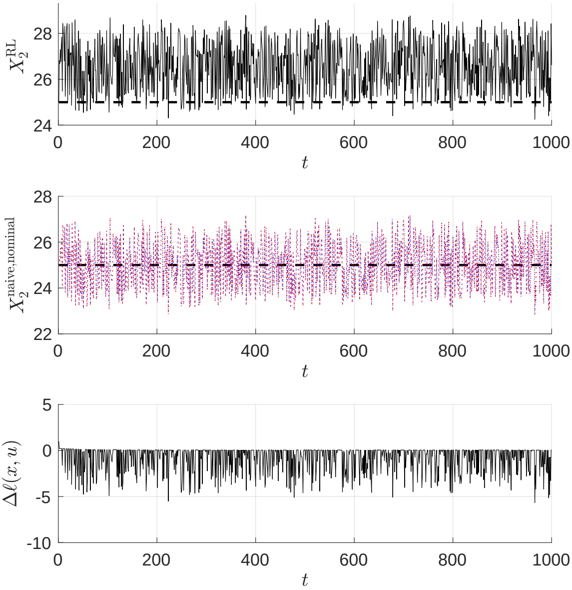
Initial guess: $\bar{\theta} = \{0, 0, 0, H_{V^\mathrm{f}}, 0, 0, H_\ell, 0, 0, 0, x_\mathrm{l}, x_\mathrm{u}\}$

$$H_{V^\mathrm{f}} = I, \quad H_\ell = I, \quad x_\mathrm{l} = \begin{bmatrix} 25 \\ 100 \end{bmatrix}, \quad x_\mathrm{u} = \begin{bmatrix} 100 \\ 80 \end{bmatrix}$$

Evaporation Process



Evaporation Process



14% gain

Enforcing Safety [Gros, Zanon, Bemporad (IFAC2020,rev)]

Ensure that $\pi(s) \in \mathcal{S}$:

Enforcing Safety [Gros, Zanon, Bemporad (IFAC2020,rev)]

Ensure that $\pi(s) \in \mathcal{S}$:

1. **Penalize** constraint violation
   - violations rare but cannot be excluded
   - ok when safety not at stake

Enforcing Safety [Gros, Zanon, Bemporad (IFAC2020,rev)]

Ensure that $\pi(s) \in \mathcal{S}$:

1. **Penalize** constraint violation
   - violations rare but cannot be excluded
   - ok when safety not at stake

2. **Project** policy onto feasible set:

$$\pi_\theta^\perp(x) = \arg\min_u \ \|u - \pi_\theta\| \quad \text{s.t. } u \in \mathcal{S}$$

Enforcing Safety [Gros, Zanon, Bemporad (IFAC2020,rev)]

Ensure that $\pi(s) \in \mathcal{S}$:

1. **Penalize** constraint violation
   - violations rare but cannot be excluded
   - ok when safety not at stake

2. **Project** policy onto feasible set:

$$\pi_\theta^\perp(x) = \arg\min_u \; \|u - \pi_\theta\| \quad \text{s.t. } u \in \mathcal{S}$$

   - cannot explore outside $\mathcal{S}$ $\Rightarrow$ constrained RL problem

Enforcing Safety [Gros, Zanon, Bemporad (IFAC2020,rev)]

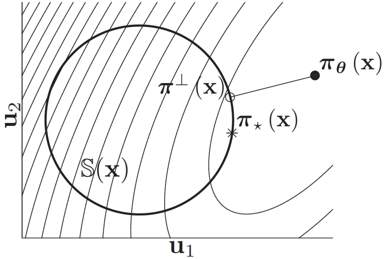
Ensure that $\pi(s) \in \mathcal{S}$:

1. **Penalize** constraint violation
   - violations rare but cannot be excluded
   - ok when safety not at stake

2. **Project** policy onto feasible set:

$$\pi_\theta^\perp(x) = \arg\min_u \; \|u - \pi_\theta\| \quad \text{s.t. } u \in \mathcal{S}$$

   - cannot explore outside $\mathcal{S}$ $\Rightarrow$ constrained RL problem
   - Q-learning
     - projection must be done carefully



Enforcing Safety [Gros, Zanon, Bemporad (IFAC2020,rev)]

Ensure that $\pi(s) \in \mathcal{S}$:

1. **Penalize** constraint violation
   - violations rare but cannot be excluded
   - ok when safety not at stake

2. **Project** policy onto feasible set:

$$\pi_\theta^\perp(x) = \arg\min_u \; \|u - \pi_\theta\| \quad \text{s.t. } u \in \mathcal{S}$$

   - cannot explore outside $\mathcal{S}$ $\Rightarrow$ constrained RL problem
   - Q-learning
     - projection must be done carefully
   - DPG: $\nabla_\theta \pi_\theta^\perp \nabla_u A_{\pi^\perp} = \nabla_\theta \pi_\theta M \nabla_u A_{\pi^\perp}$

Enforcing Safety [Gros, Zanon, Bemporad (IFAC2020,rev)]

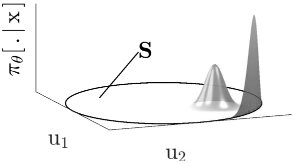
Ensure that $\pi(s) \in \mathcal{S}$:

1. **Penalize** constraint violation
   - violations rare but cannot be excluded
   - ok when safety not at stake

2. **Project** policy onto feasible set:

$$\pi_\theta^\perp(x) = \arg\min_u \ \|u - \pi_\theta\| \quad \text{s.t. } u \in \mathcal{S}$$

   - cannot explore outside $\mathcal{S}$ $\Rightarrow$ constrained RL problem
   - Q-learning
     - projection must be done carefully
   - DPG: $\nabla_\theta \pi_\theta^\perp \nabla_u A_{\pi^\perp} = \nabla_\theta \pi_\theta M \nabla_u A_{\pi^\perp}$
   - SPG: 1. draw a sample; 2. project
     - dirac-like structure on the boundary



Enforcing Safety [Gros, Zanon, Bemporad (IFAC2020,rev)]

Ensure that $\pi(s) \in \mathcal{S}$:

1. **Penalize** constraint violation
   - violations rare but cannot be excluded
   - ok when safety not at stake

2. **Project** policy onto feasible set:

$$\pi_\theta^\perp(x) = \arg\min_u \; \|u - \pi_\theta\| \quad \text{s.t. } u \in \mathcal{S}$$

   - cannot explore outside $\mathcal{S}$ $\Rightarrow$ constrained RL problem
   - Q-learning
     - projection must be done carefully
   - DPG: $\nabla_\theta \pi_\theta^\perp \nabla_u A_{\pi^\perp} = \nabla_\theta \pi_\theta M \nabla_u A_{\pi^\perp}$
   - SPG: 1. draw a sample; 2. project
     - dirac-like structure on the boundary
     - use IP method for projection: $\approx$ Dirac, but continuous

Enforcing Safety [Gros, Zanon, Bemporad (IFAC2020,rev)]

Ensure that $\pi(s) \in \mathcal{S}$:

1. **Penalize** constraint violation
   - violations rare but cannot be excluded
   - ok when safety not at stake

2. **Project** policy onto feasible set:

$$\pi_\theta^\perp(x) = \arg\min_u \; \|u - \pi_\theta\| \quad \text{s.t. } u \in \mathcal{S}$$

   - cannot explore outside $\mathcal{S} \Rightarrow$ constrained RL problem
   - Q-learning
     - projection must be done carefully
   - DPG: $\nabla_\theta \pi_\theta^\perp \nabla_u A_{\pi^\perp} = \nabla_\theta \pi_\theta M \nabla_u A_{\pi^\perp}$
   - SPG: 1. draw a sample; 2. project
     - dirac-like structure on the boundary
     - use IP method for projection: $\approx$ Dirac, but continuous
     - $\nabla_\theta J(\pi_\theta^\perp) = \nabla_\theta \log \pi_\theta \nabla_u A_{\pi^\perp}$ evaluate score function gradient on unprojected sample

Enforcing Safety [Gros, Zanon, Bemporad (IFAC2020,rev)]

Ensure that $\pi(s) \in \mathcal{S}$:

1. **Penalize** constraint violation
   - violations rare but cannot be excluded
   - ok when safety not at stake

2. **Project** policy onto feasible set:

$$\pi_\theta^\perp(x) = \arg\min_u \ \|u - \pi_\theta\| \quad \text{s.t. } u \in \mathcal{S}$$

   - cannot explore outside $\mathcal{S} \Rightarrow$ constrained RL problem
   - Q-learning
     - projection must be done carefully
   - DPG: $\nabla_\theta \pi_\theta^\perp \nabla_u A_{\pi^\perp} = \nabla_\theta \pi_\theta M \nabla_u A_{\pi^\perp}$
   - SPG: 1. draw a sample; 2. project
     - dirac-like structure on the boundary
     - use IP method for projection: $\approx$ Dirac, but continuous
     - $\nabla_\theta J(\pi_\theta^\perp) = \nabla_\theta \log \pi_\theta \nabla_u A_{\pi^\perp}$ evaluate score function gradient on unprojected sample

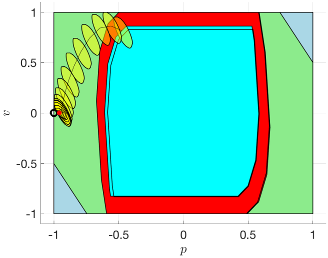
3. **Safety by construction**

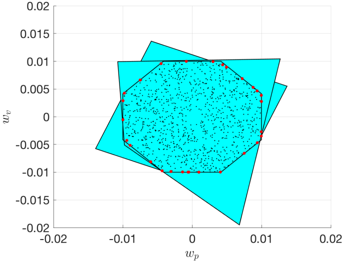
Safe RL [Zanon, Gros (TAC,rev.)], [Gros, Zanon (TAC,rev.)]

- Robust MPC as function approximator
- Model used for data compression

Safe RL [Zanon, Gros (TAC,rev.)], [Gros, Zanon (TAC,rev.)]

- Robust MPC as function approximator
- Model used for data compression

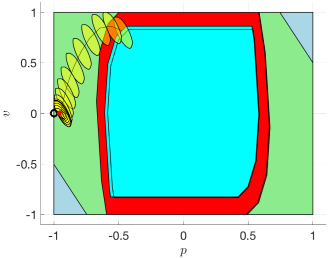


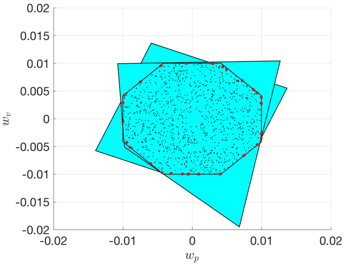


MPC-based RL

Safe RL [Zanon, Gros (TAC,rev.)], [Gros, Zanon (TAC,rev.)]

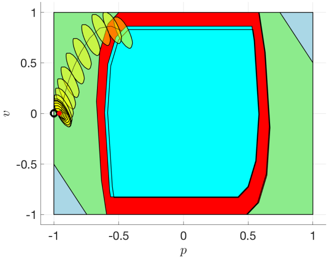
- Robust MPC as function approximator
- Model used for data compression
- Q-learning: no adaptation required

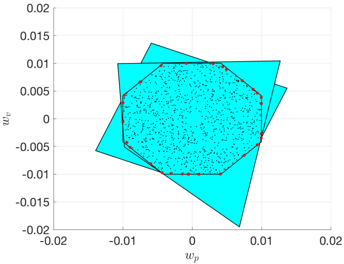




Safe RL [Zanon, Gros (TAC,rev.)], [Gros, Zanon (TAC,rev.)]

- Robust MPC as function approximator
- Model used for data compression
- *Q*-learning: no adaptation required
- Actor-critic:
  - best possible performance
  - constraints pose technical difficulties





MPC-based RL

Safe RL [Zanon, Gros (TAC,rev.)], [Gros, Zanon (TAC,rev.)]

- Robust MPC as function approximator
- Model used for data compression
- Q-learning: no adaptation required
- Actor-critic:
  - best possible performance
  - constraints pose technical difficulties

How to explore safely?

$$\begin{aligned} \min_{x,u} \quad & d^\top u_0 + V_\theta^{\mathrm{f}}(x_N) + \sum_{k=0}^{N-1} L_\theta(x,u) \\ \text{s.t.} \quad & x_0 = s, \\ & x_{k+1} = f_\theta(x_k, u_k), \\ & h_\theta(x_k, u_k) \leq 0, \\ & x_N \in \mathbb{X}_\theta^{\mathrm{f}}. \end{aligned}$$

Gradient d perturbs the MPC solution.

MPC-based RL

Safe Q-learning [Zanon, Gros (TAC,rev.)]

Evaporation process

Nominal optimal steady state

$$\begin{bmatrix} X_2 \\ P_2 \end{bmatrix} = \begin{bmatrix} 25\,\% \\ 49.743\,\mathrm{kPa} \end{bmatrix}, \qquad \begin{bmatrix} P_{100} \\ F_{200} \end{bmatrix} = \begin{bmatrix} 191.713\,\mathrm{kPa} \\ 215.888\,\mathrm{kg/min} \end{bmatrix}.$$

Safe Q-learning [Zanon, Gros (TAC,rev.)]

Evaporation process

Nominal optimal steady state

$$\begin{bmatrix} X_2 \\ P_2 \end{bmatrix} = \begin{bmatrix} 25\,\% \\ 49.743\,\mathrm{kPa} \end{bmatrix}, \qquad \begin{bmatrix} P_{100} \\ F_{200} \end{bmatrix} = \begin{bmatrix} 191.713\,\mathrm{kPa} \\ 215.888\,\mathrm{kg/min} \end{bmatrix}.$$

- Satisfy $X_2 \geq 25$ robustly

Safe Q-learning [Zanon, Gros (TAC,rev.)]

Evaporation process

Nominal optimal steady state

$$\begin{bmatrix} X_2 \\ P_2 \end{bmatrix} = \begin{bmatrix} 25\,\% \\ 49.743\,\mathrm{kPa} \end{bmatrix}, \qquad \begin{bmatrix} P_{100} \\ F_{200} \end{bmatrix} = \begin{bmatrix} 191.713\,\mathrm{kPa} \\ 215.888\,\mathrm{kg/min} \end{bmatrix}.$$

- Satisfy $X_2 \geq 25$ robustly
- Adjust uncertainty set representation

Safe Q-learning [Zanon, Gros (TAC,rev.)]

Evaporation process

Nominal optimal steady state

$$\begin{bmatrix} X_2 \\ P_2 \end{bmatrix} = \begin{bmatrix} 25\,\% \\ 49.743\,\mathrm{kPa} \end{bmatrix}, \qquad \begin{bmatrix} P_{100} \\ F_{200} \end{bmatrix} = \begin{bmatrix} 191.713\,\mathrm{kPa} \\ 215.888\,\mathrm{kg/min} \end{bmatrix}.$$

- Satisfy $X_2 \geq 25$ robustly
- Adjust uncertainty set representation
- Adjust the RMPC cost

Safe Q-learning [Zanon, Gros (TAC,rev.)]

Evaporation process

Nominal optimal steady state

$$\begin{bmatrix} X_2 \\ P_2 \end{bmatrix} = \begin{bmatrix} 25\,\% \\ 49.743\,\mathrm{kPa} \end{bmatrix}, \qquad \begin{bmatrix} P_{100} \\ F_{200} \end{bmatrix} = \begin{bmatrix} 191.713\,\mathrm{kPa} \\ 215.888\,\mathrm{kg/min} \end{bmatrix}.$$

- Satisfy $X_2 \geq 25$ robustly
- Adjust uncertainty set representation
- Adjust the RMPC cost
- Adjust feedback K

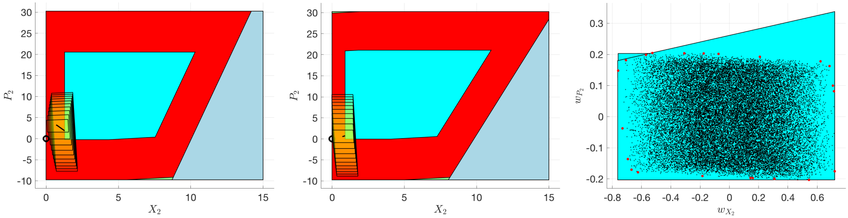
Safe Q-learning [Zanon, Gros (TAC,rev.)]

Evaporation process

Nominal optimal steady state

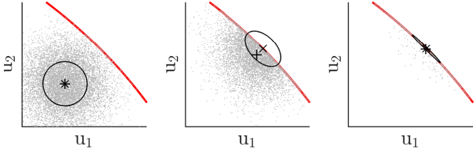
$$\begin{bmatrix} X_2 \\ P_2 \end{bmatrix} = \begin{bmatrix} 25\,\% \\ 49.743\,\mathrm{kPa} \end{bmatrix}, \qquad \begin{bmatrix} P_{100} \\ F_{200} \end{bmatrix} = \begin{bmatrix} 191.713\,\mathrm{kPa} \\ 215.888\,\mathrm{kg/min} \end{bmatrix}.$$

- Satisfy $X_2 \geq 25$ robustly
- Adjust uncertainty set representation
- Adjust the RMPC cost
- Adjust feedback K



Safe Actor-Critic RL [Gros, Zanon (TAC,rev.)]

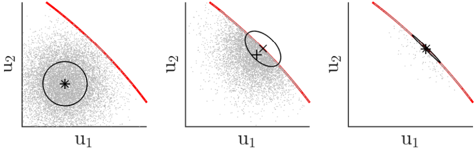
Safe exploration distorts the distribution



Distribution of d, a

Safe Actor-Critic RL [Gros, Zanon (TAC,rev.)]

Safe exploration distorts the distribution



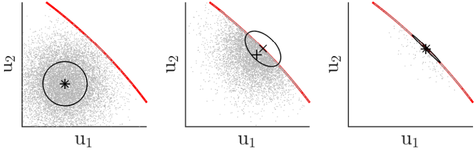
Distribution of d, a

Deterministic PG

$$\nabla_\theta J = \nabla_\theta \pi_\theta \nabla_a \underbrace{A_{\pi_\theta}}_{\text{Advantage Function}}$$

Safe Actor-Critic RL [Gros, Zanon (TAC,rev.)]

Safe exploration distorts the distribution



Distribution of d, a

Deterministic PG

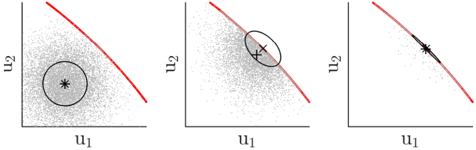
$$\nabla_\theta J = \nabla_\theta \pi_\theta \nabla_a \underbrace{A_{\pi_\theta}}_{\text{Advantage Function}}$$

Bias in $\nabla_a A_{\pi_\theta}$:

- $\mathbb{E}[a - \pi_\theta(s)] \neq 0$
- $\mathrm{Cov}[a - \pi_\theta(s)] \rightarrow$ rank deficient

Safe Actor-Critic RL [Gros, Zanon (TAC,rev.)]

Safe exploration distorts the distribution



Distribution of d, a

Deterministic PG

$$\nabla_\theta J = \nabla_\theta \pi_\theta \underbrace{\nabla_a A_{\pi_\theta}}_{\text{Advantage Function}}$$

Stochastic PG

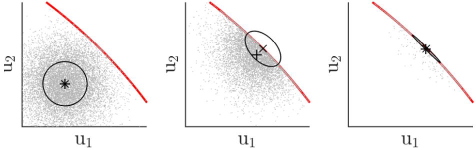
$$\nabla_\theta J = \nabla_\theta \log \pi_\theta \underbrace{\delta^V_{\pi_\theta}}_{\text{TD Error}}$$

Bias in $\nabla_a A_{\pi_\theta}$:

- $\mathbb{E}[a - \pi_\theta(s)] \neq 0$
- $\mathrm{Cov}[a - \pi_\theta(s)] \rightarrow$ rank deficient

Safe Actor-Critic RL [Gros, Zanon (TAC,rev.)]

Safe exploration distorts the distribution



Distribution of d, a

Deterministic PG

$$\nabla_\theta J = \nabla_\theta \pi_\theta \nabla_a \underbrace{A_{\pi_\theta}}_{\text{Advantage Function}}$$

Bias in $\nabla_a A_{\pi_\theta}$:

- $\mathbb{E}[a - \pi_\theta(s)] \neq 0$
- $\mathrm{Cov}[a - \pi_\theta(s)] \to$ rank deficient

Stochastic PG

$$\nabla_\theta J = \nabla_\theta \log \pi_\theta \underbrace{\delta^V_{\pi_\theta}}_{\text{TD Error}}$$

- sampling $\nabla_\theta \log \pi_\theta$ too expensive
  - if action infeasible: resample
- gradient perturbation is the solution
  - ∇J more expensive than DPG

MPC Sensitivities [Gros, Zanon (TAC2020)], [Zanon, Gros (TAC,rev.)], [Gros, Zanon (TAC,rev.)]

$$\theta \leftarrow \theta + \alpha \Big\{ \underbrace{\delta \nabla_\theta Q_\theta(s,a)}_{Q\text{-learning}}, \quad \underbrace{\nabla_\theta \pi_\theta \nabla_a A_{\pi_\theta}}_{\text{DPG}}, \quad \underbrace{\nabla_\theta \pi_\theta \delta_{\pi_\theta}}_{\text{SPG}} \Big\}$$

MPC Sensitivities [Gros, Zanon (TAC2020)], [Zanon, Gros (TAC,rev.)], [Gros, Zanon (TAC,rev.)]

$$\theta \leftarrow \theta + \alpha \Big\{ \underbrace{\delta \nabla_\theta Q_\theta(s,a)}_{Q\text{-learning}}, \quad \underbrace{\nabla_\theta \pi_\theta \nabla_a A_{\pi_\theta}}_{\text{DPG}}, \quad \underbrace{\nabla_\theta \pi_\theta \delta_{\pi_\theta}}_{\text{SPG}} \Big\}$$

Differentiate MPC [Gros, Zanon TAC2020], [Zanon, Gros, Bemporad ECC2019]

Need to compute $\nabla_\theta Q_\theta(s,a)$, $\nabla_\theta \pi_\theta(s)$, $\nabla_a A_{\pi_\theta}$

MPC-based RL

MPC Sensitivities [Gros, Zanon (TAC2020)], [Zanon, Gros (TAC,rev.)], [Gros, Zanon (TAC,rev.)]

$$\theta \leftarrow \theta + \alpha \Big\{ \underbrace{\delta \nabla_\theta Q_\theta(s,a),}_{Q\text{-learning}} \qquad \underbrace{\nabla_\theta \pi_\theta \nabla_a A_{\pi_\theta},}_{\text{DPG}} \qquad \underbrace{\nabla_\theta \pi_\theta \delta_{\pi_\theta}}_{\text{SPG}} \Big\}$$

Differentiate MPC [Gros, Zanon TAC2020], [Zanon, Gros, Bemporad ECC2019]

Need to compute $\nabla_\theta Q_\theta(s,a)$, $\nabla_\theta \pi_\theta(s)$, $\nabla_a A_{\pi_\theta}$

Result from parametric optimization:

- $\nabla_\theta Q_\theta(s,a) = \nabla_\theta \mathcal{L}_\theta$, $\quad \mathcal{L}_\theta$ = Lagrangian of MPC
- $M \nabla_\theta \pi_\theta(s) = \frac{\partial r_\theta}{\partial \theta}$, $\quad M, r$ = KKT matrix and residual
- $\nabla_a A_{\pi_\theta} = \nu$, $\quad \nu$ = multiplier of $u_0 = a$

MPC Sensitivities [Gros, Zanon (TAC2020)], [Zanon, Gros (TAC,rev.)], [Gros, Zanon (TAC,rev.)]

$$\theta \leftarrow \theta + \alpha \Big\{ \underbrace{\delta \nabla_\theta Q_\theta(s,a)}_{Q\text{-learning}}, \quad \underbrace{\nabla_\theta \pi_\theta \nabla_a A_{\pi_\theta}}_{\text{DPG}}, \quad \underbrace{\nabla_\theta \pi_\theta \delta_{\pi_\theta}}_{\text{SPG}} \Big\}$$

Differentiate MPC [Gros, Zanon TAC2020], [Zanon, Gros, Bemporad ECC2019]

Need to compute $\nabla_\theta Q_\theta(s,a)$, $\nabla_\theta \pi_\theta(s)$, $\nabla_a A_{\pi_\theta}$

Result from parametric optimization:

- $\nabla_\theta Q_\theta(s,a) = \nabla_\theta \mathcal{L}_\theta$, $\quad \mathcal{L}_\theta =$ Lagrangian of MPC
- $M \nabla_\theta \pi_\theta(s) = \frac{\partial r_\theta}{\partial \theta}$, $\quad M, r =$ KKT matrix and residual
- $\nabla_a A_{\pi_\theta} = \nu$, $\quad \nu =$ multiplier of $u_0 = a$

Derivatives are cheap!

- $\nabla_\theta \mathcal{L}_\theta$ much cheaper than MPC
- M already factorized inside MPC
- ν is for free

MPC Sensitivities [Gros, Zanon (TAC2020)], [Zanon, Gros (TAC,rev.)], [Gros, Zanon (TAC,rev.)]

$$\theta \leftarrow \theta + \alpha \left\{ \delta \nabla_\theta Q_\theta(s,a), \quad \nabla_\theta \pi_\theta \nabla_a A_{\pi_\theta}, \quad \nabla_\theta \pi_\theta \delta_{\pi_\theta} \right\}$$

Q-learning DPG SPG

Differentiate MPC [Gros, Zanon TAC2020], [Zanon, Gros, Bemporad ECC2019]

Need to compute $\nabla_\theta Q_\theta(s,a)$, $\nabla_\theta \pi_\theta(s)$, $\nabla_a A_{\pi_\theta}$

Result from parametric optimization:

- $\nabla_\theta Q_\theta(s,a) = \nabla_\theta \mathcal{L}_\theta$, $\mathcal{L}_\theta$ = Lagrangian of MPC
- $M \nabla_\theta \pi_\theta(s) = \frac{\partial r_\theta}{\partial \theta}$, M, r = KKT matrix and residual
- $\nabla_a A_{\pi_\theta} = \nu$, ν = multiplier of $u_0 = a$

Derivatives are cheap!

- $\nabla_\theta \mathcal{L}_\theta$ much cheaper than MPC
- M already factorized inside MPC
- ν is for free

Safe RL:

- ∇ constraint tightening
- Actor-critic
  - Additional computations
  - DPG cheaper than SPG

Real-Time NMPC and RL [Zanon, Kungurtsev, Gros (IFAC2020,rev)]

Real-time feasibility:

Real-Time NMPC and RL [Zanon, Kungurtsev, Gros (IFAC2020,rev)]

Real-time feasibility:

- NMPC can be computationally heavy

Real-Time NMPC and RL [Zanon, Kungurtsev, Gros (IFAC2020,rev)]

Real-time feasibility:

- NMPC can be computationally heavy
- use RTI

Real-Time NMPC and RL [Zanon, Kungurtsev, Gros (IFAC2020,rev)]

Real-time feasibility:

- NMPC can be computationally heavy
- use RTI

Sensitivities:

- formulae only hold at convergence

Real-Time NMPC and RL [Zanon, Kungurtsev, Gros (IFAC2020,rev)]

Real-time feasibility:

- NMPC can be computationally heavy
- use RTI

Sensitivities:

- formulae only hold at convergence
- compute sensitivities of RTI QP
- justified in a patfollowing framework

Real-Time NMPC and RL [Zanon, Kungurtsev, Gros (IFAC2020,rev)]

Real-time feasibility:

- NMPC can be computationally heavy
- use RTI

Sensitivities:

- formulae only hold at convergence
- compute sensitivities of RTI QP
- justified in a patfollowing framework

Evaporation process:

Real-Time NMPC and RL [Zanon, Kungurtsev, Gros (IFAC2020,rev)]

Real-time feasibility:

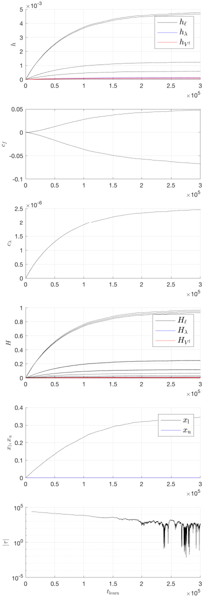
- NMPC can be computationally heavy
- use RTI

Sensitivities:

- formulae only hold at convergence
- compute sensitivities of RTI QP
- justified in a patfollowing framework

Evaporation process:

- Similar results as fully converged NMPC



Real-Time NMPC and RL [Zanon, Kungurtsev, Gros (IFAC2020,rev)]

Real-time feasibility:

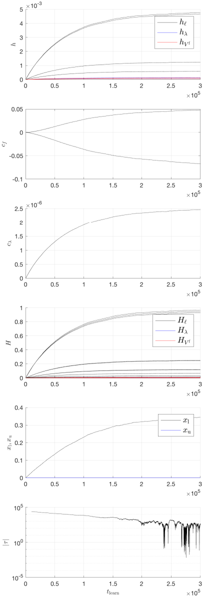
- NMPC can be computationally heavy
- use RTI

Sensitivities:

- formulae only hold at convergence
- compute sensitivities of RTI QP
- justified in a patfollowing framework

Evaporation process:

- Similar results as fully converged NMPC
- Gain $\approx 15 - 20\%$ over standard RTI



Mixed-Integer Problems [Gros, Zanon IFAC2020,rev.]

Can we handle mixed-integer problems?

- RL was born for integer problems
- Apply, e.g., SPG: expensive

Mixed-Integer Problems [Gros, Zanon IFAC2020,rev.]

Can we handle mixed-integer problems?

- RL was born for integer problems
- Apply, e.g., SPG: expensive

What about a combination of SPG and DPG?

- separate continuous and integer parts
- continuous: DPG
- integer: SPG

Mixed-Integer Problems [Gros, Zanon IFAC2020,rev.]

Can we handle mixed-integer problems?

- RL was born for integer problems
- Apply, e.g., SPG: expensive

What about a combination of SPG and DPG?

- separate continuous and integer parts
- continuous: DPG
- integer: SPG

Real system:

$$x_{k+1} = x_k + u_k i_k + w_k, \qquad w_k \sim \mathcal{U}[0, 0.05]$$

Mixed-Integer Problems [Gros, Zanon IFAC2020,rev.]

Can we handle mixed-integer problems?

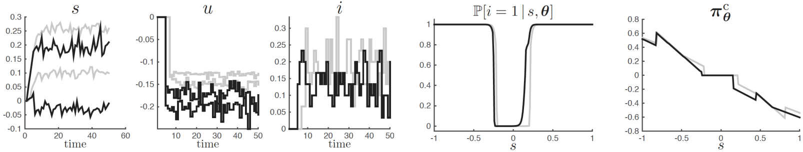
- RL was born for integer problems
- Apply, e.g., SPG: expensive

What about a combination of SPG and DPG?

- separate continuous and integer parts
- continuous: DPG
- integer: SPG

Real system:

$$x_{k+1} = x_k + u_k i_k + w_k, \qquad\qquad w_k \sim \mathcal{U}[0, 0.05]$$



Conclusions

The goal:

- simplify the computational aspects
- provide safety and stability guarantees
- achieve true optimality
- self-tuning optimal controllers

Conclusions

The goal:

- simplify the computational aspects
- provide safety and stability guarantees
- achieve true optimality
- self-tuning optimal controllers

We are not quite there yet!

Conclusions

The goal:

- simplify the computational aspects
- provide safety and stability guarantees
- achieve true optimality
- self-tuning optimal controllers

We are not quite there yet! Challenges:

- exploration vs exploitation (identifiability and persistent excitation)
- data noise and cost
- can we combine SYSID and RL effectively?

Our contribution

1. Gros, S. and Zanon, M. **Data-Driven Economic NMPC using Reinforcement Learning.** IEEE Transactions on Automatic Control, 2020, in press.
2. Zanon, M., Gros, S., and Bemporad, A. **Practical Reinforcement Learning of Stabilizing Economic MPC.** European Control Conference 2019
3. Zanon, M. and Gros, S. **Safe Reinforcement Learning Using Robust MPC.** Transaction on Automatic Control, (under review).
4. Gros, S. and Zanon, M. **Safe Reinforcement Learning Based on Robust MPC and Policy Gradient Methods** IEEE Transactions on Automatic Control, (under review).
5. Gros, S., Zanon, M, and Bemporad, A. **Safe Reinforcement Learning via Projection on a Safe Set: How to Achieve Optimality?** IFAC World Congress, 2020 (submitted)
6. Gros, S. and Zanon, M. **Reinforcement Learning for Mixed-Integer Problems Based on MPC.** IFAC World Congress, 2020 (submitted)

Thank you for your attention!