Safe Reinforcement Learning for Discrete-Time, Control Affine Systems with Input and State Constraints

SAGIP-2025

Authors: Satya Marthi, Mayank Shekhar Jha, Didier Theilliol, Jean-Christophe Ponsart

Mulhouse- May-2025

Contents

Contents

System considerations and problem statement

Let's consider a discrete-time dynamical system

- State: $x \in \Omega \subset \mathbb{R}^n$
- Control Input: $\mathcal{U} = \{u \in \mathbb{R}^m : -\lambda \leq u \leq \lambda\}$
- Dynamics: $x_{k+1} = f(x_k) + g(x_k)u$
- The system is fully observable and controllable
- $f(x_k) \in \mathbb{R}^n, g(x_k) \in \mathbb{R}^{n \times m}$ are Lipschitz continuous
- The sets Ω ,$\mathcal{U}$ are compact
- The final state $x_N = 0$ is known

Problem: Optimal Control

$$J(x_k, u_k) = \sum_{k=i}^{N-1} \frac{1}{2}(x_k^\top \boldsymbol{Q} x_k + u_k^\top \boldsymbol{R} u_k)$$

$$u^\star = \arg\min_u J(x_k, u_k)$$

where $Q = Q^\top \geq 0$; $R = R^\top > 0$

Objective: Solve the *regulation problem* $(x_k \to x_N = 0 \text{ as } k \to N)$ under input saturation and state constraints $(x_{\min} \leq x_k \leq x_{\max})$

Motivation

- **Motivation**: The traditional optimal control algorithm is offline and is not adaptive in nature. The objective is to develop an online adaptive optimal control algorithm under both state constraints (safety) and input saturation using Reinforcement Learning algorithms.

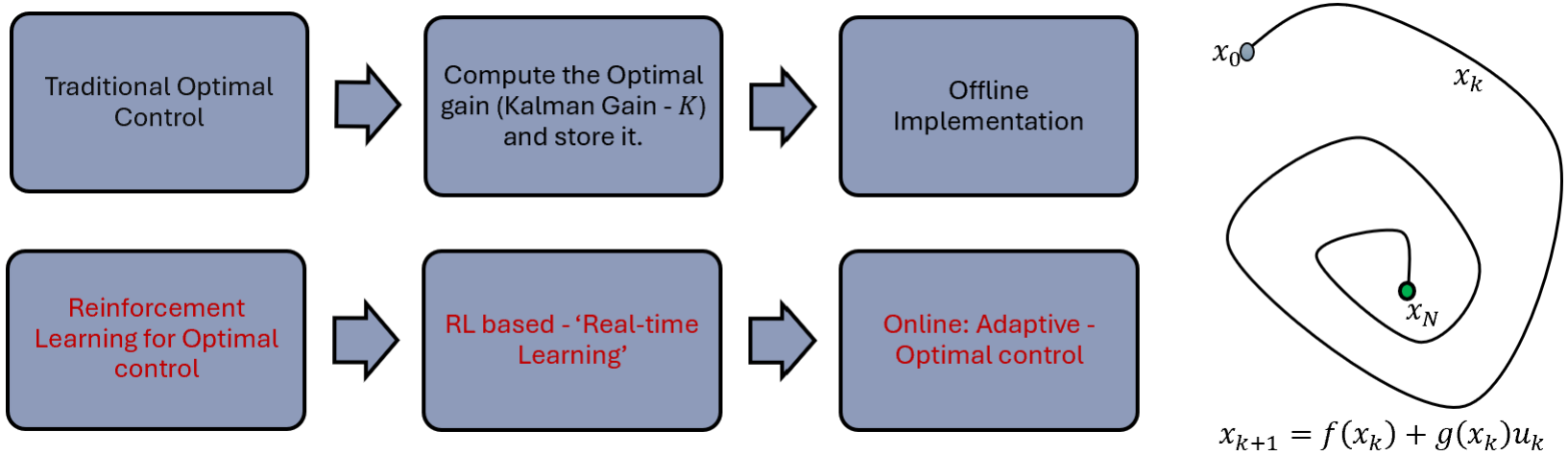


Figure 1: Traditional Optimal Control vs. Proposed RL based optimal control

The optimal control problem can be solved using Reinforcement Learning

Reinforcement Learning

- Reinforcement Learning is a family of learning algorithms where the objective is to learn the optimal policies to fulfil the user optimization requirements. The goal here is to find the optimal control policy/control law for a dynamical system (discrete or continuous).

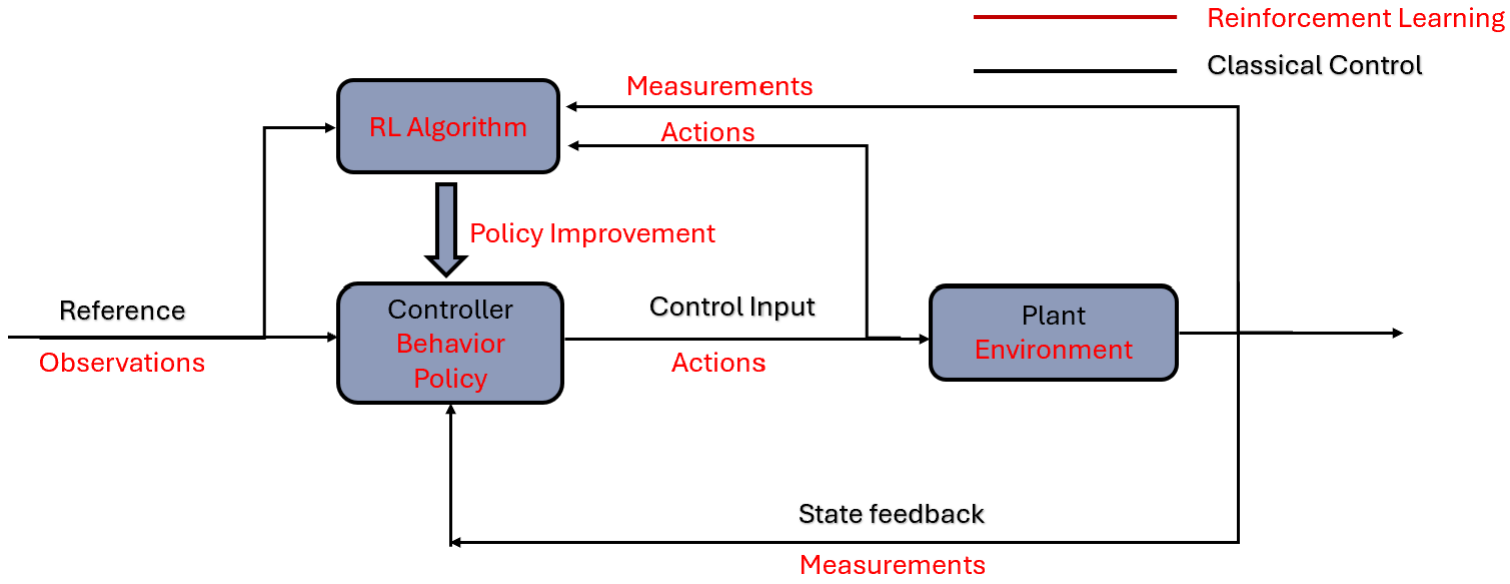


Figure 2: Learning in reinforcement learning compared to classical control theory

Reinforcement Learning Terminology

- **Reward function:** Is a quadratic function same as the instantaneous cost function in optimal control

$$r(x_k, u_k) = \frac{1}{2}(x_k^\top \boldsymbol{Q} x_k + u_k^\top \boldsymbol{R} u_k)$$

- **Value function**: It is the same *cost-to-go function*, also called as the *cumulative reward function*

$$V_h(x_k) = \sum_{k=i}^{N-1} \gamma^{k-i} \frac{1}{2}(x_k^\top \boldsymbol{Q} x_k + u_k^\top \boldsymbol{R} u_k)$$

where h is the control policy such as $u_k = h(x_k)$ and $0 \leq \gamma \leq 1$ is called the discount factor and $\gamma = 1$ for optimal control problems.

- The value function can be rewritten as,

$$V_h(x_k) = r\big(x_k, h(x_k)\big) + \gamma V_h(x_{k+1}), \qquad V(0) = 0$$

- Using Bellman's optimality principle, the optimal control policy can be obtained by,

$$V_h^\star(x_k) = r\big(x_k, h(x_k)\big) + \gamma V_h^\star(x_{k+1})$$

$$h^\star(x_k) = \arg\min_{h(\cdot)} \Big(r\big(x_k, h(x_k)\big) + \gamma V_h^\star(x_{k+1})\Big)$$

Contents

RL algorithm to solve the optimal control

- The RL framework provides an important algorithm to solve the optimal control problems, called *Policy Iteration* [1].
- The Policy Iteration (PI) algorithm can be implemented along the system trajectory to minimize the cost function

- Typically, the PI algorithm involves two steps:
  1. Policy evaluation (value function update)

$$\boxed{V_{h,j+1}(x_k) = r\big(x_k, h(x_k)\big) + \gamma V_{h,j+1}(x_{k+1})}$$

  2. Policy improvement

$$\boxed{h_{j+1}(x_k) = \arg\min_{h(\cdot)} r\big(x_k, h(x_k)\big) + \gamma V_{h,j+1}(x_{k+1})}$$

Here, k is the discrete time-step and j is the iteration step

[1] S Sutton and Andrew G Barto. Reinforcement learning: An introduction. MIT press, 2018

Limitations of policy iteration to solve optimal control problem

1. Policy Evaluation (Value function update)

$$V_{h,j+1}(x_k) = r\big(x_k, h(x_k)\big) + \gamma V_{h,j+1}(x_{k+1})$$

2. Policy Improvement

$$h_{j+1}(x_k) = \arg\min_{h(\cdot)} r\big(x_k, h(x_k)\big) + \gamma V_{h,j+1}(x_{k+1})$$

These equations are offline, nonlinear and backwards-in-time

Solution: 1. Reformulating the Policy evaluation step
2. Using Neural Networks (NN's) to approximate the unknown nonlinear functions

Online, Policy Iteration algorithm

1. The value function is reformulated

$$V_h(x_k) = r\big(x_k, h(x_k)\big) + \gamma V_h(x_{k+1})$$

$$e(x_k) = r\big(x_k, h(x_k)\big) + \gamma V_h(x_{k+1}) - V_h(x_k)$$

2. The functions $V(x_k)$ and $h(x_k)$ are unknown and nonlinear and neural networks are used to approximate the functions

$$V(x_k) = W^{\mathsf{T}}\psi(x) \quad \text{(also called as critic network)}$$

$$h(x_k) = U^{\mathsf{T}}\sigma(x) \text{ (also called as actor network)}$$

Thus, we get online, forward-in-time, RL algorithm. The new algorithm is [1]

Policy evaluation: $W_j^{\mathsf{T}}\big(\psi(x_k) - \psi(x_{k+1})\big) = r\big(x_k, h_j(x_k)\big)$

Regression vector

Policy improvement: $h_{j+1} = \frac{\gamma}{2} R^{-1} g^{\mathsf{T}}(x_k) \nabla\psi^{\mathsf{T}}(x_{k+1}) W_{j+1}$

[1] S Sutton and Andrew G Barto. Reinforcement learning: An introduction. MIT press, 2018

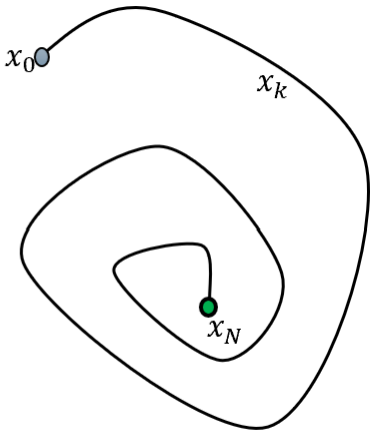
Online policy Iteration

Policy Evaluation: $W_j^{\mathsf{T}}\left(\psi(x_k) - \psi(x_{k+1})\right) = r\left(x_k, h_j(x_k)\right)$

Policy improvement: $h_{j+1} = \frac{\gamma}{2} R^{-1} g^{\mathsf{T}}(x_k) \nabla\psi^{\mathsf{T}}(x_{k+1}) W_{j+1}$

x_0 - Initial state
u_0 - Initial Policy

Time-step x_k	$k = 1:10$	$k = 11:20$	$k = 21:30$	---------- x_N
Closed loop control policy $u_k = h(x_k)$	$h_0(x) = \{u_0, \dots u_{10}\}$	$h_1(x) = \{u_{11}, \dots u_{20}\}$	$h_3(x) = \{u_{21}, \dots u_{30}\}$	$h_{\text{opt}}(x) = \{u_{opt}, \dots u_{opt}\}$
Iteration step	$j = 1$	$j = 2$	$j = 3$	----------



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

- The online data collected is used to solve the $\psi(x_k) - \psi(x_{k+1})$ regression vector for the neural network weights W^{T}.
- For the reward function $r(x_k, u_k)$, the algorithm can be used to obtain adaptive-optimal control policy.
- However, to initialize the algorithm the initial policy $u_0 = h_0(x_0)$ must be admissible.

Contents

Input Saturation

- To introduce input saturation, the reward function is modified from,

$$r(x_k, u_k) = \frac{1}{2} x_k^\top \boldsymbol{Q} x_k + u_k^\top \boldsymbol{R} u_k$$

- To a nonquadratic functional [2,3] given by:

$$r(x_k, u_k) = x_k^\top \boldsymbol{Q} x_k + U(u_k)$$

Where,

$$U(u_k) = \int_0^{u_k} \lambda \, \Gamma^{-\top}\left(\frac{v}{\lambda}\right) \boldsymbol{R} \, dv$$

One candidate for the $\boldsymbol{\Gamma(\cdot)}$ is $\boldsymbol{\tanh(\cdot)}$ function. Now, the policy improvement becomes,

$$h_{j+1} = \lambda \tanh\left(\frac{\gamma}{2} (R\lambda)^{-1} g^\top(x_k) \nabla \psi^\top(x_{k+1})\right) W_{j+1}$$

The control input is learnt under saturation bounds $|\boldsymbol{u_k}| \leq \boldsymbol{\lambda}$

[2] Shihan Liu, Lijun Liu, and Zhen Yu. "Safe Reinforcement Learning for Affine Nonlinear Systems with State Constraints and Input Saturation using Control Barrier Functions". In: Neurocomputing 518 (2023), pp. 562–576.
[3] Huaguang Zhang, Derong Liu. "Neural-Network-Based Near-Optimal Control for a Class of Discrete-Time Affine Nonlinear Systems with Control Constraints". In: IEEE Transactions on Neural Networks 20.9 (2009), pp. 1490–1503

Safety

- The concept of safety is to ensure that the system states are always within a user defined safe set $\mathcal{C}$.
- In other words, safety here is to provide formal guarantees on constraint enforcement.

Exploration noise: Typically, an external noise is added to the RL algorithm that improves the ability to learn optimal policies that could be devised such that it also satisfies Persistence of Excitation (PoE) condition.

- Advantages: Searches the state-space to find the optimal policy quickly and to avoid any local minima.
- Disadvantages: The system may reach undesirable control policies that my 'force' the system states out of the safe set (could lead to infeasibility of constraints).

Several ways to ensure safety, in this study we consider **Barrier function** based approach

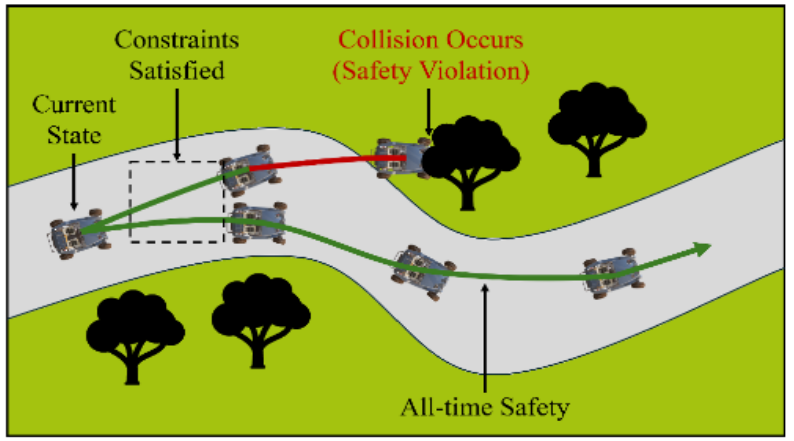


Figure 3: All time safety [4]

[4] Hwang, Sunwoo, et al. "Safe Motion Planning and Control for Mobile Robots: A Survey." International Journal of Control, Automation and Systems 22.10 (2024): 2955-2969.

Barrier function

- The idea is to add the barrier function within the reward function such that reward function value tends to infity when the system states approach the boundary of the safe set [5,6].

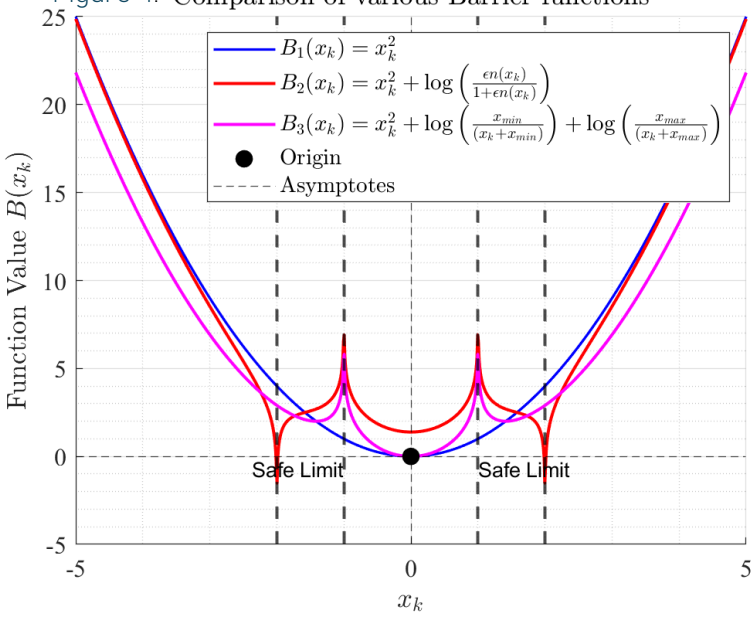
For example, the following function $B_\gamma(x_k)$ is added to the reward function along with the input saturation

$$B_\gamma(x_k) = \log\left(\frac{x_{\min}}{x_k + x_{\min}}\right) + \log\left(\frac{x_{\max}}{x_k + x_{\max}}\right)$$

$$r(x_k, u_k) = x_k^\top \boldsymbol{Q} x_k + \int_0^{u_k} \lambda\, \Gamma^{-\top}\left(\frac{v}{\lambda}\right) \boldsymbol{R}\, dv + B_\gamma(x_k)$$

Input saturation — Barrier function

Figure 4: Comparison of various Barrier functions



[5] Aaron D Ames et al. "Control Barrier Function Based Quadratic Programs for Safety Critical Systems". In: IEEE Transactions on Automatic Control 62.8 (2016), pp. 3861–3876.
[6] Richard Cheng et al. "End-to-end Safe Reinforcement Learning through Barrier Functions for Safety-Critical Continuous Control Tasks". In: Proceedings of the AAAI conference on artificial intelligence. Vol. 33. 01. 2019, pp. 3387–3395

Online policy iteration algorithm

- Due to the nature of policy iteration algorithm, we need an initial admissible policy to initiate the algorithm. To this end, we also require that the policy is safe. Thus, we utilize Control Lyapunov functions (CLF) and Control Barrier functions (CBF) [4].

- The CLF and CBF conditions are combined with Quadratic programming (QP) optimization to give an initial policy that is both admissible and safe [4]. (Also called as QP-CLF-CBF optimization).

$$
\begin{aligned}
& h_0(x_0) = \min \frac{1}{2} u_k^\top u_k \\
s.t. \quad & \Delta L(x_k, u_k) \leq -\alpha_1\big(L(x_k)\big) \quad \text{(CLF condition)} \\
& \Delta B(x_k, u_k) \geq -\alpha_2\big(B(x_k)\big) \quad \text{(CBF condition)}
\end{aligned}
$$

- Where, $L(x_k)$ is the Lyapunov function candidate and $B(x_k)$ is the barrier function candidate. $\alpha_1(\cdot), \alpha_2(\cdot)$ are class $\mathcal{K}_\infty$ functions. $\Delta L(x_k, u_k) = L(x_{k+1}) - L(x_k)$ and $\Delta B(x_k, u_k) = B(x_{k+1}) - B(x_k)$.

[5] Aaron D Ames et al. "Control Barrier Function Based Quadratic Programs for Safety Critical Systems". In: IEEE Transactions on Automatic Control 62.8 (2016), pp. 3861–3876.

Contents

Example – 1: Linear DC Motor system

- The equations of motion for the DC motor are in continuous time and the system is discretized using the 'zero-order hold' method. The equations of motion are:

$$\begin{bmatrix}\dot{\omega}\\ i\end{bmatrix} = \begin{bmatrix}-\frac{b}{J} & -\frac{k}{J}\\ -\frac{k}{L} & -\frac{R}{L}\end{bmatrix}\begin{bmatrix}\omega\\ i\end{bmatrix} + \begin{bmatrix}0\\ \frac{1}{L}\end{bmatrix}V$$

- The DT- system is: $x_{k+1} = A\,x_k + B\,u_k$. The system is fully observable and controllable.

For $V(x_k) = x_k^{\mathsf{T}} P x_k$, the ARE becomes:

$$A^{\mathsf{T}}\boldsymbol{P}A - \boldsymbol{P} + Q - A^{\mathsf{T}}\boldsymbol{P}B(R + B^{\mathsf{T}}\boldsymbol{P}B)^{-1}B^{\mathsf{T}}\boldsymbol{P}A = 0$$

The optimal gain (Kalman gain):

$$\boldsymbol{K} = (R + B^{\mathsf{T}}\boldsymbol{P}B)^{-1}\boldsymbol{P}A$$

The critic network is expected to learn the solution to the ARE equation, whereas the actor network is expected to learn the optimal gain.

DC motor (contd.)

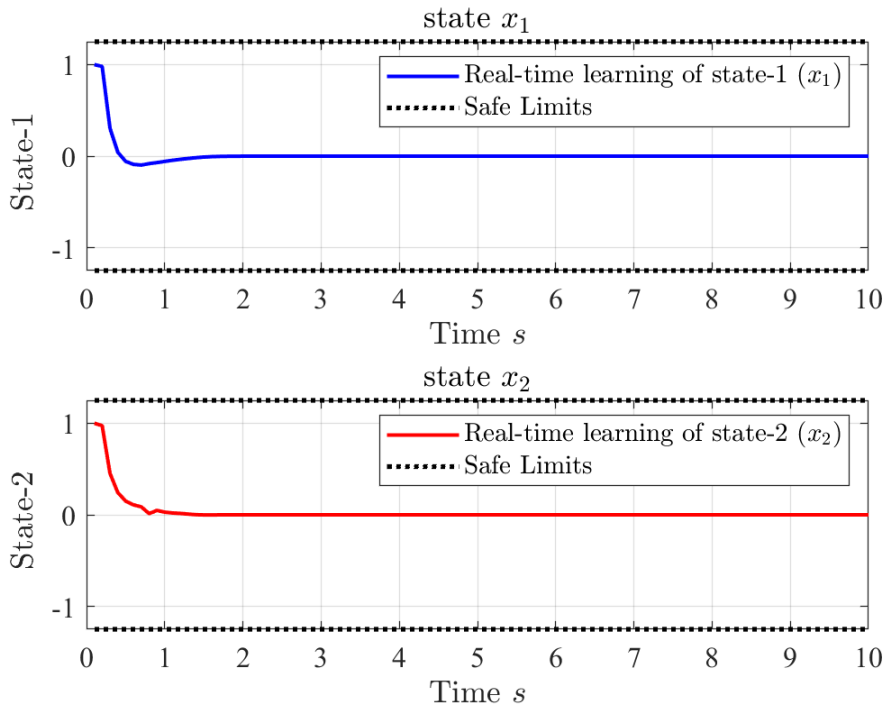


Figure 6: Real-time learning of system states

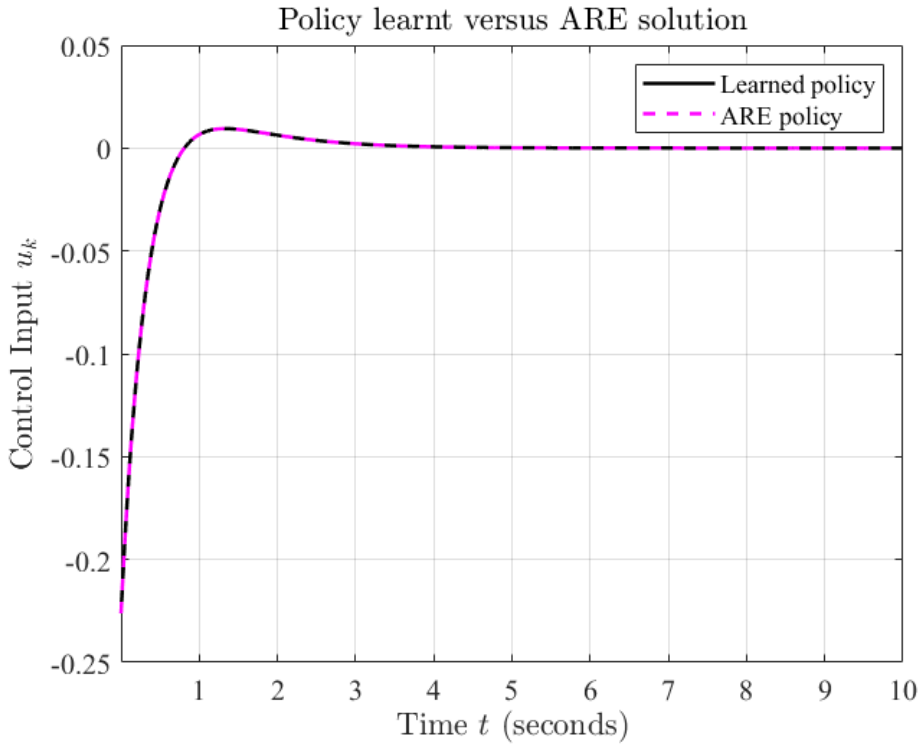


Figure 7: Control learnt vs ARE solution

DC motor (contd.)

$\begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix}$	Critic weights W	$P_{11} = 5.4282, P_{12} = -2.1311,$ $P_{21} = -2.1311, P_{22} = 3.5967$
	ARE solution	$P_{11} = 5.4282\ , P_{12} = -2.1312,$ $P_{21} = -2.1310, P_{22} = 3.5968$
$[K_1 \quad K_2]$	Actor weights U	$K_1 = 0.0425, K_2 = 0.0704$
	ARE solution	$K_1 = 0.0425, K_2 = 0.0704$

For $Q = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}, R = 1$

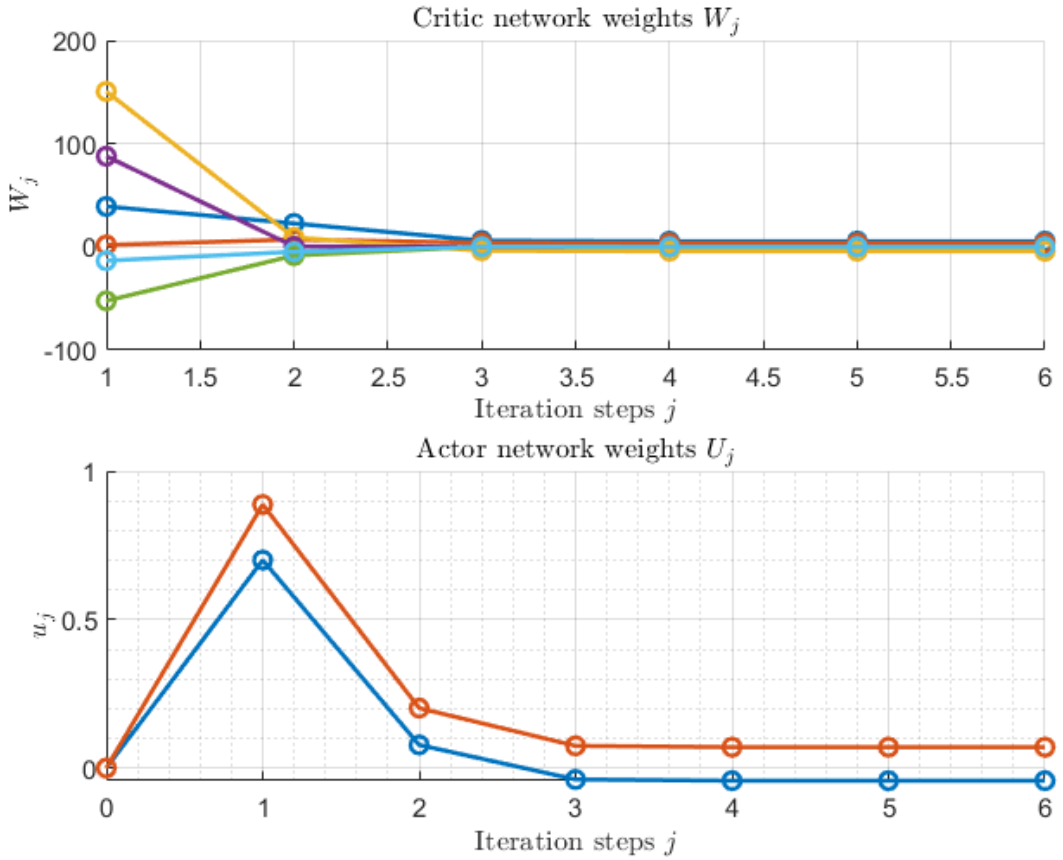


Figure 8: Convergence of actor and critic network weights

Example-2: Nonlinear system

- The nonlinear system considered is an inverted pendulum system with the following dynamics:

$$x_{1_{k+1}} = x_{1_t} + x_{2_t}\Delta t + \frac{3g}{2l}\sin x_{1_t}\,\Delta t^2 + \frac{3}{ml^2}u_t\Delta t^2$$

$$x_{2_{t+1}} = x_{2_t} + \frac{3g}{2l}\sin x_{1_t}\,\Delta t + \frac{3}{ml^2}u_t\Delta t$$

Here, Δt is the sampling time, $-2.2 \leq x_1 \leq 2.2$ is the angular position and $-2.2 \leq x_2 \leq 2.2$ is the angular velocity and u_t is the control input torque with $-5 \leq u_t \leq +5$.

The neural network considered are basis functions given by:

$$\text{Critic : } \phi(x) = [x_1^2, x_2^2, x_1x_2, x_1^2x_2^3, x_1^3x_2^2, x_1^2\,x_2^2]$$

$$\text{Actor: } \psi(x) = [x_1, x_2]$$

The exploration noise added is:

$$e_t = \sum a^j \omega \sin f_\eta$$

where $0 < a < 1$ is a decay constant, ω is a Gaussian random variable and $f_\eta = [1,3,5,7,9]$

Inverted pendulum (contd.)

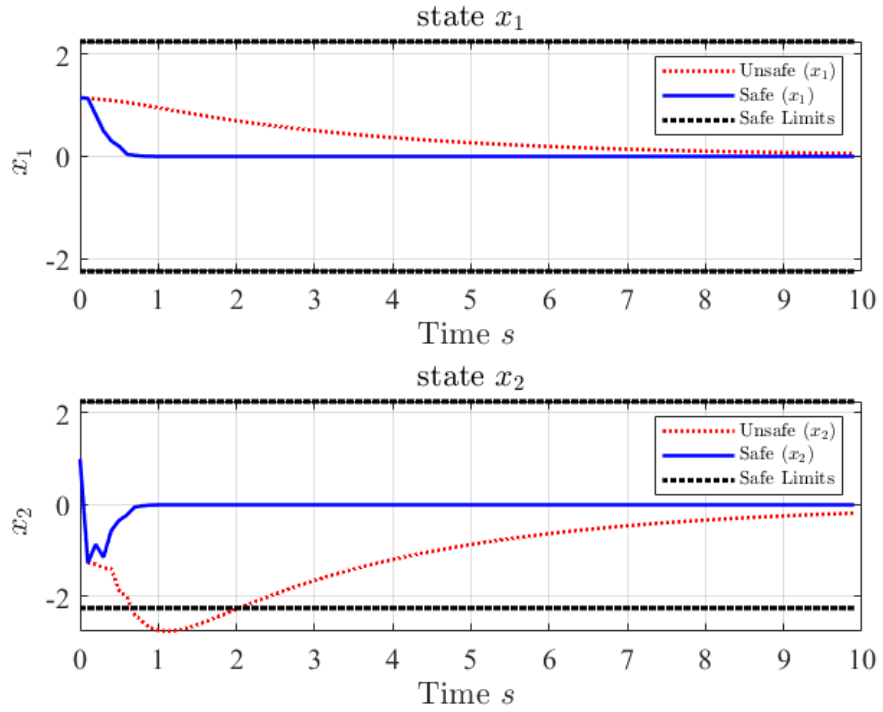


Figure 10: System states with the proposed method

Inverted Pendulum (contd.)

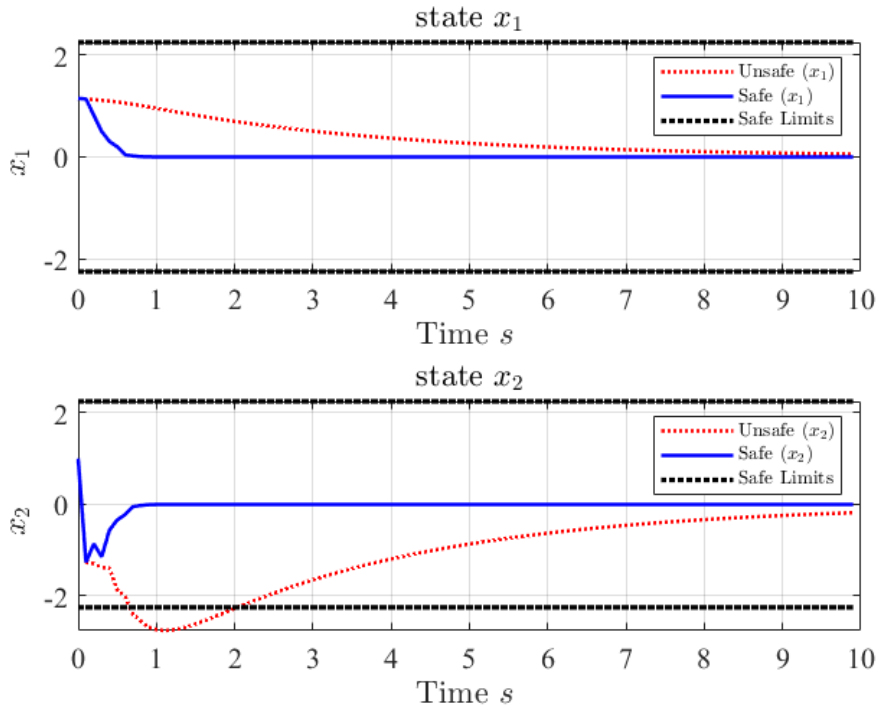


Figure 10: System states with the proposed method

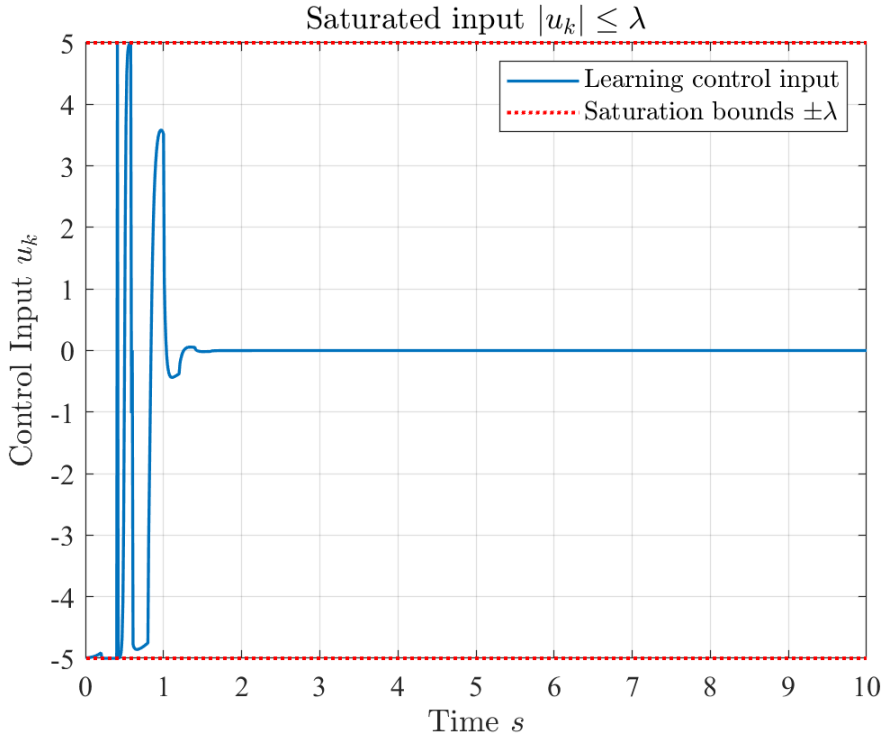


Figure 11: Control Input

Inverted Pendulum (contd.)

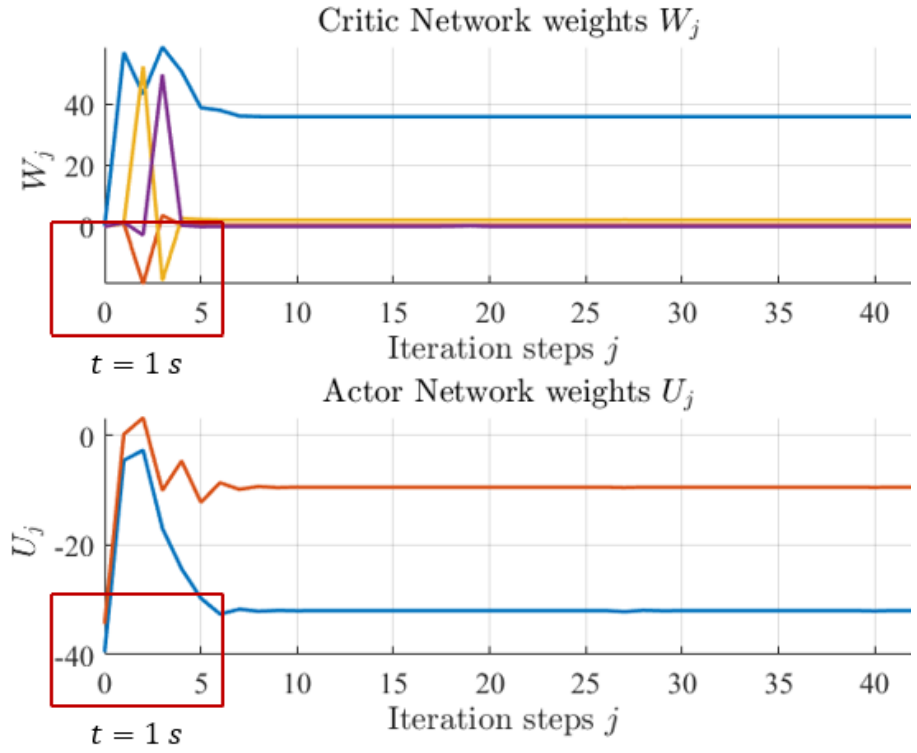


Figure 12: Actor and critic weights

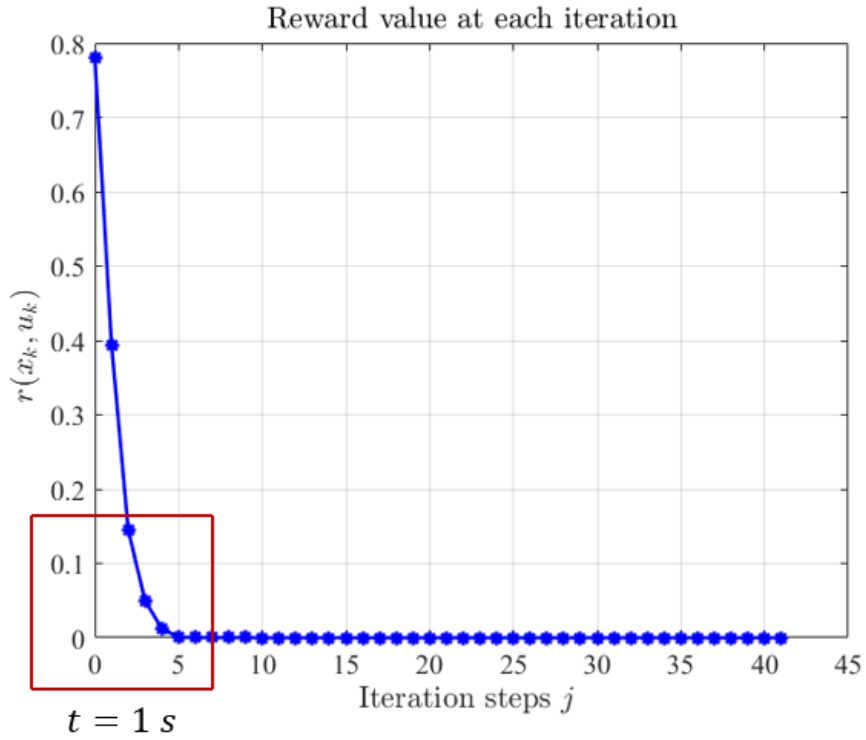


Figure 13: Reward function value at each iteration

Contents

Conclusions and future work

- A safe, adaptive optimal control algorithm, under input saturation and state constraints are studied and applied.
- The input saturation and state constraint enforcement is guaranteed.

- Use Neural Networks or Gaussian Processes to identify the system dynamics to remove any dependency on the apriori model knowledge.
- Finally, apply the algorithm on a real-time system.

- We have submitted the extension of this work for publication at the CDC - 2025.
- In the study, we have also compared our approach with the traditional approaches for formally guaranteeing constraint enforcement and input saturation.

Safe Reinforcement Learning for Discrete-Time, Control Affine Systems with Input and State Constraints
SAGIP-2025

Authors: Satya Marthi, Mayank Shekhar Jha, Didier Theilliol, Jean-Christophe Ponsart

Mulhouse - May-2025