
Reinforcement Learning with Adaptive Control Regularization for Safe Control of Critical Systems

Haozhe Tian Homayoun Hamedmoghadam Robert Shorten Pietro Ferraro
 Dyson School of Design Engineering, Imperial College London, London, UK
 {haozhe.tian21,h.hamed,r.shorten,p.ferraro}@imperial.ac.uk

Abstract

Reinforcement Learning (RL) is a powerful method for controlling dynamic systems, but its learning mechanism can lead to unpredictable actions that undermine the safety of critical systems. Here, we propose RL with Adaptive Control Regularization (RL-ACR) that ensures RL safety by combining the RL policy with a control regularizer that hard-codes safety constraints over forecasted system behaviors. The adaptability is achieved by using a learnable “focus” weight trained to maximize the cumulative reward of the policy combination. As the RL policy improves through off-policy learning, the focus weight improves the initial sub-optimum strategy by gradually relying more on the RL policy. We demonstrate the effectiveness of RL-ACR in a critical medical control application and further investigate its performance in four classic control environments.

1 Introduction

A wide array of control applications, ranging from medical to engineering, fundamentally deals with *critical systems*, i.e., systems of vital importance where control actions have to guarantee no harm to the system functionality. Examples include managing nuclear fusion [Degraive et al., 2022], performing robotic surgeries [Diana and Marescaux, 2015], and devising patient treatment strategy [Komorowski et al., 2018]. Due to the critical nature of these systems, ensuring control safety and reliability is paramount.

Reinforcement Learning (RL), which identifies the optimum policy through interactions with the environment, has seen substantial success in controlling complex systems [Silver et al., 2016, Ouyang et al., 2022]. However, RL’s search for optimal policy involves trial-and-error which can violate safety constraints in critical system applications [Henderson et al., 2018, Recht, 2019, Cheng et al., 2019b]. Despite RL’s potential in critical domains [Degraive et al., 2022], it remains a challenge to develop reliable RL-based algorithms for real-world “single-life” applications, where the control must succeed from the first trial [Chen et al., 2022]. The existing safe RL algorithms either fail to ensure safety during the training phase [Achiam et al., 2017, Yu et al., 2022] or require significant computational overhead for action verification [Cheng et al., 2019a, Anderson et al., 2020]. As a result, traditional control methods are often favored for critical applications, even though their performance is heavily dependent on the availability of accurate environment models.

Here, we address RL’s safety issue when an estimated model of the environment can be built to derive a control prior. This scenario is representative of many critical real-life applications [Hovorka et al., 2002, Liepe et al., 2014, Hippisley-Cox et al., 2017]. Consider an application where drug doses are prescribed to regulate a patient’s health status, which is a single-life setting where no harm to the patient is tolerated. An estimated model can be built from other patients’ records to predict patient responses and guarantee adherence to safety bounds (set based on clinical knowledge). However, individual variations in new patients’ responses, which deviates from the estimated model, pose a significant challenge in control adaptability and patient-treatment performance.

We propose a method, *RL with Adaptive Control Regularization* (RL-ACR), that simultaneously shows “safety” and “adaptability” properties required for critical applications in single-life settings. The method interacts with the actual environment using two parallel agents. The first (safe) agent is initialized with the estimated model and solves a constrained optimization that hard-codes safety. The second (adaptive) agent is a model-free RL agent which enables adaptability by learning from the actual environment interactions. Our proposed method introduces a focus mechanism that dynamically learns to combine the two agents’ policies, allowing the RL agent to explore the environment’s dynamics while correcting its effect with the safe agent’s policy until the optimum policy is found.

The idea of combining the RL policy with another policy is implemented in Cheng et al. [2019b] as well, but with a fixed weight which prevents convergence to the optimum policy. To the best of our knowledge, RL-ACR is the first work that automatically learns to combine a safe policy with the RL policy, allowing for unbiased convergence to the optimum policy. Furthermore, RL-ACR is readily applicable to the environment and ensures safe control even during training, which is another significant benefit that promotes applications of RL in critical systems.

The proposed algorithm is tested under extensive numerical experiments including comparisons with different baseline methods. We demonstrate the performance of the RL-ACR in real-world critical control with an important medical application. Estimated models are widely available for this problem, but the model parameters can differ considerably between patients making individualized adaptation crucial for control performance.

2 Preliminaries

RL is a data-driven approach that learns from interactions with the environment. Instead of relying on an explicit environment model, RL acts in each time step and observes the corresponding state transition and reward. The objective of an RL agent is to learn a policy that maximizes the “cumulative reward”, estimated from past observations. Formally, the environment is modeled as a Markov Decision Process (MDP) defined by $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}$ is the observation space, $\mathcal{A}$ is the action space, $P(s, a) : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is the state transition probability, $r(s, a) : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function, and $\gamma \in (0, 1)$ is the discount factor that exponentially reduces future rewards to prioritize immediate rewards. The expected future cumulative reward for state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$ is characterized by the Q-function $Q^\pi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, which is defined as follows:

$$Q^\pi(s, a) = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid \pi, s_0 = s, a_0 = a \right], \quad (1)$$

where $\pi : \mathcal{S} \rightarrow \mathcal{A}$ is a policy. At any state $s \in \mathcal{S}$, the optimum policy is determined by $\pi^* = \operatorname{argmax}_{\pi} Q^\pi(s, \pi(s))$ to maximize the expected cumulative reward. Therefore, the optimality of π depends on an accurate Q-function estimation. A fundamental aspect of this estimation is the application of the Bellman equation [Bellman, 1966]:

$$Q^\pi(s, a) = r(s, a) + \gamma \mathbb{E}_{s' \sim P(s, a)} [Q^\pi(s', \pi(s'))], \quad (2)$$

In modern RL, Q^π and π are respectively approximated with neural networks $Q_\phi(\cdot)$ and π_θ , where ϕ and θ are the learnable parameters. An approach to stabilize learning is using a Replay Buffer $\mathcal{D}$, where the experience in the environment is recorded as transitions $e_t = (s_t, a_t, s_{t+1}, r_t)$ at each time step t and a batch $\mathcal{B}$ of transitions from $\mathcal{D}$ are sampled to update ϕ and θ [Silver et al., 2016].

In this work, we are interested in taking safe control actions that can differ from the learned RL policy. RL approaches that allow different acting policy and learned policy are referred to as “Off-policy RL”. A state-of-the-art off-policy RL algorithm is Soft Actor-Critic (SAC), which uses a stochastic policy to capture environmental uncertainties and prevent sticking to sub-optimal policies. Similar to [Fujimoto et al., 2018], SAC uses the clipped double Q-learning and maintains two Q-networks $Q_{\phi_i}(\cdot)$, $i = 1, 2$. To update the Q-networks, gradient descent is performed using:

$$\nabla_{\phi_i} \frac{1}{|\mathcal{B}|} \sum_{(s, a, s', r, d) \in \mathcal{B}} (Q_{\phi_i}(s, a) - y)^2, \quad i = 1, 2 \quad (3)$$

where $d \in \{0, 1\}$ equals 1 if s' is a terminal state of the environment, and equals 0 otherwise. The objective y in Eq. (3) is given by:

$$y = r + \gamma(1 - d) \left(\min_{i=1,2} Q_{\phi_{\text{target}, i}}(s', a') - \alpha \log \pi_\theta(a' | s') \right), \quad a' \sim \pi_\theta(s'), \quad (4)$$

where $\log \pi_\theta(a'|s')$ is the entropy regularization term introduced by Haarnoja et al. [2018] to encourage random actions and facilitate exploration. Target Q -networks $\phi_{\text{target},i}(\cdot)$ proposed by Lillicrap et al. [2015] are used to reduce drastic changes in value estimates and stabilize training. The target Q -network parameters are initialized with $\phi_{\text{target},i} = \phi_i$, $i = 1, 2$. Each time ϕ_1 and ϕ_2 are updated, $\phi_{\text{target},1}, \phi_{\text{target},2}$ slowly track the update using $\tau \in (0, 1)$:

$$\phi_{\text{target},i} = \tau \phi_{\text{target},i} + (1 - \tau) \phi_i, \quad i = 1, 2. \quad (5)$$

To update the policy network $\pi_\theta(\cdot)$, gradient ascent is performed using:

$$\nabla_\theta \frac{1}{|\mathcal{B}|} \sum_{s \in \mathcal{B}} \left(\min_{i=1,2} Q_{\phi_i}(s, \pi_\theta(s)) - \alpha \log \pi_\theta(a|s) \right). \quad (6)$$

Note that the policy network relies on Q values estimated by $Q_{\phi_i}(\cdot)$ instead of $\phi_{\text{target},i}(\cdot)$ to allow quick policy updates in response to new information.

3 Methodology

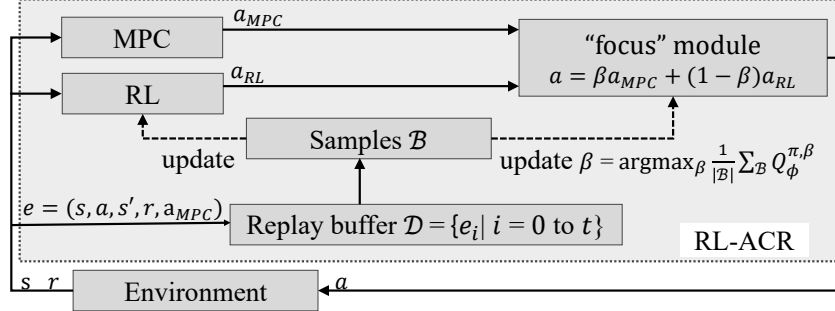


Figure 1: RL-ACR schematic. The actual action is the weighted sum of MPC and RL actions. The RL module and the focus weight β are updated using historical transitions in the replay buffer.

Here, we propose RL-ACR, an algorithm designed for the safe operation of critical dynamical systems. A schematic view of the procedure in RL-ACR is shown in Fig. 1. RL-ACR is composed of three main modules: The *MPC module* takes the form of a classic model predictive control (MPC) implementation. The *RL module* follows an off-policy paradigm, enabling learning from the combined MPC and RL policy, which is different from the RL strategy. The *“focus” module* learns a focus weight β that determines how to mix the MPC action a_{MPC} and RL action a_{RL} . The three modules function together when interacting with the environment, among them the RL module and the focus module are dynamically updated from historical observations.

The workflow of RL-ACR is outlined as follows: i) the RL module allows stochastic exploration and adaptation to the actual environment, ii) the Function Regularizer (the MPC module) generates a policy with guaranteed safety within a forecasted period by considering hard-coded safety constraints in an optimization problem, and iii) the adaptive action combination adjusts the focus weight β based on observations in the actual environment, allowing gradual switch to the RL policy as the RL module is trained. Algorithm 1 shows the pseudocode for updating RL-ACR. The detailed workflow will be explained in the following sections. The details of the RL module follow the SAC [Haarnoja et al., 2018] paradigm introduced in Section 2 and is omitted in this section.

3.1 Function regularizer

RL-ACR regularizes the RL action using an MPC module, which uses an in-built estimated system model to forecast the system behavior under different policies. At any state s_t , the MPC module finds an action a_t that optimizes the N -step system forecast (the horizon), while ensuring safety by formalizing safety measures as constraints. During each step, the following constrained optimization

Algorithm 1 RL-ACR

1: **Initialization:** empty replay buffer $\mathcal{D}$; MPC with estimated system model $f(\cdot)$; policy network $\pi_\theta(\cdot)$ with parameter θ ; Q -networks $Q_{\phi_i}(\cdot)$ with parameters ϕ_i , $i = 1, 2$; target Q -networks $Q_{\phi_{\text{targ},i}}(\cdot)$ with parameters $\phi_{\text{targ},i}$, $i = 1, 2$; focus weight $\beta = 1$; step $t = 0$.
2: Set $\phi_{\text{targ},1} \leftarrow \phi_1$, $\phi_{\text{targ},2} \leftarrow \phi_2$
3: **repeat**
4: Observe state s .
5: Take action $a = \beta a_{MPC} + (1 - \beta)\pi_\theta(s)$.
6: Store trajectory $e = (s, a, s', r, d, a_{MPC})$ in $\mathcal{D}$.
7: **if** $t > \text{time to update}$ **then**
8: Randomly sample a batch $\mathcal{B}$ of transitions from $\mathcal{D}$.
9: Update $Q_{\phi_i}(\cdot)$, $i = 1, 2$ by gradient descent with Eq. (3).
10: Update $\pi_\theta(\cdot)$ by gradient ascent with Eq. (6).
11: Update β by gradient ascent with:
$$\nabla_\beta \frac{1}{|\mathcal{B}|} \sum_{s \in \mathcal{B}} \min_{i=1,2} Q_{\phi_i}(s, \beta a_{MPC} + (1 - \beta)\pi_\theta(s))$$

12: Update $Q_{\phi_{\text{targ},i}}(\cdot)$, $i = 1, 2$, using Eq. (5).
13: **end if**
14: $t = t + 1$
15: **until** convergence is true

problem is solved in the MPC module:

$$\begin{aligned} & \min_{a_{t:t+N-1}} \sum_{k=t}^{t+N-1} J_k(s_k, a_k) + J_N(s_{t+N}) \\ & \text{s.t.:} \\ & \quad s_{k+1} = f(s_k, a_k) \\ & \quad s_{\min} \leq s_{k+1} \leq s_{\max}, \forall k \in \{t, \dots, t+N\} \\ & \quad a_{\min} \leq a_k \leq a_{\max}, \forall k \in \{t, \dots, t+N-1\}, \end{aligned} \tag{7}$$

where $J_k(s_k, a_k)$ is the stage cost function at time step k , and $J_N(\cdot)$ is the terminal cost function. The estimated model $f(\cdot)$ characterizes the dynamics of the environment. Solving the optimization problem in Eq. (7) yields a sequence of N actions $a_{t:t+N-1}$, with only the first term a_t adopted at time step t , i.e. the MPC policy $\pi_{MPC}(s_t) = a_t$. As the system transits from s_t to s_{t+1} as a result of a_t , the optimization (Eq. (7)) is performed again over the new horizon $\{t+1 : t+1+N\}$ to obtain a_{t+1} . Thereby, MPC iteratively solves for the N -step optimum action in each time step and steers the environment to the desired state. The optimization in Eq. (7) is efficiently solved using the interior-point filter line-search algorithm [Wächter and Biegler, 2006].

3.2 Adaptive action combination

The combined policy in the focus module is as follows:

$$\begin{aligned} a(s) &= \beta a_{MPC}(s) + (1 - \beta)a_{RL}(s) \\ a_{MPC} &= \pi_{MPC}, a_{RL}(s) = \pi_\theta, \beta \in [0, 1]. \end{aligned} \tag{8}$$

The weight β is initialized to be 1, prioritizing the MPC's policy when learning begins. As RL gradually improves through interactions with the environment, the agent increases its confidence in the RL policy by updating β to lower values. As proved in [Cheng et al., 2019b], the combined policy in Eq. (8) is equivalent to regularizing the RL policy a_{RL} with the function regularizer a_{MPC} .

Lemma 1. Assuming the convergence of the RL policy π_θ , the mean of the combined action $\bar{a}(s)$ is the solution to the following regularized optimization with regularization parameter $\lambda = \frac{\beta}{1-\beta}$:

$$\bar{a}(s) = \underset{a}{\operatorname{argmin}} \|a(s) - \underset{\pi_\theta}{\operatorname{argmax}} Q(s, \pi_\theta(s))\|_\Sigma + \frac{\beta}{1-\beta} \|a(s) - a_{MPC}(s)\|_\Sigma. \tag{9}$$

The proof of Lemma 1 is provided in the appendix. During the early stage of training, β is close to one, resulting in strong regularization from the MPC policy. The MPC module ensures safety using two features: 1) hard-coded safe constraints and 2) predictive capability. When a breach of constraint is forecasted, MPC sacrifices high rewards to ensure safety.

To enable updates of networks and the focus weight β , RL-ACR keeps track of the transition $e = (s, a, s', r, d, a_{MPC})$ using a replay buffer $\mathcal{D}$ after each interaction with the environment. Compared to the original replay buffer framework proposed by Lin [1992] and Mnih et al. [2013], we store an augmented e with an additional term a_{MPC} to enable the update of β .

The update of the focus weight β utilizes a batch $\mathcal{B}$ of randomly sampled past transitions from $\mathcal{D}$. The objective is to find the β that generates the combined action a that maximizes the expected return:

$$\operatorname{argmax}_{\beta \in [0,1]} \mathbb{E}_{s \sim p_\pi} [Q^{\pi_\theta, \beta}(s, \beta a_{MPC} + (1 - \beta)\pi_\theta(s))], \quad (10)$$

where p_π denotes the state distribution induced by following the policy π . Assume RL converges to the optimum policy π^* that satisfies $Q_\phi(s, \pi^*(s)) > Q_\phi(s, a), \forall a \in \mathcal{A} \setminus \pi^*(s)$, then the optimality in Eq. (10) is achieved by $\beta = 0$. Because if $\beta \neq 0$ then $\beta\pi_{MPC}(s) + (1 - \beta)\pi_\theta(s) = \beta\pi_{MPC}(s) + (1 - \beta)\pi^*(s) \neq \pi^*(s)$. The inequality follows because $\pi_{MPC} \neq \pi^*$ as the MPC is based on a sub-optimum model.

Theorem 1. *Let $D_{TV}(\cdot, \cdot)$ denote the total variance distance. The combined policy π has the following bias:*

$$D_{TV}(\pi, \pi^*) \geq D_{TV}(\pi_{MPC}, \pi^*) - (1 - \beta)D_{TV}(\pi_{MPC}, \pi_\theta). \quad (11)$$

However, assume RL converges to the optimum policy π^ that satisfies $Q^{\pi^*}(s, \pi^*(s)) > Q^\pi(s, a), \forall a \in \mathcal{A} \setminus \pi^*(s)$, the combined policy π is unbiased with $D_{TV}(\pi, \pi^*) = 0$.*

The proof of Theorem 1 is given in the appendix, which shows that by adaptively updating β , the convergence of the combined policy can be achieved with common assumptions.

Practically, learning an optimal β can be seen as a separate update step after updating the Q -networks and the policy network. To estimate $Q^{\pi_\theta, \beta}(\cdot)$ in Eq. (10), we utilize the updated Q -networks of the RL module. To stabilize training, the clipped double-Q learning [Fujimoto et al., 2018] was also used in the β update. Suppose $Q_{\phi_i}(\cdot), i = 1, 2$, and $\pi_\theta(\cdot)$ are updated by a batch $\mathcal{B}$ of past experiences. Then, β is updated by gradient ascent using:

$$\nabla_\beta \frac{1}{|\mathcal{B}|} \sum_{(s, a_{MPC}) \in \mathcal{B}} \min_{i=1,2} Q_{\phi_i}(s, \beta a_{MPC} + (1 - \beta)\pi_\theta(s)), \quad (12)$$

Note that the updated Q -networks and policy network are used in Eq. (12) to allow quick response to new information.

Although β converges to 0 theoretically, to avoid violation of the action bounds, β is clipped between 0 and 1 in practice. To allow adequate learning of the Q -networks and the policy network, the gradient ∇_β is clipped to the range $[-c, c], c > 0$. With a fixed learning rate lr , the maximum reduction of ∇_β in each step is bounded by $c \cdot lr$.

4 Numerical Experiments

In this section, we validate RL-ACR in the practical setting mentioned in Section 1, where only the estimated models are available to RL-ACR and baselines, but the actual environment models are unknown. To simulate this, we set different model parameters for the estimated models and the actual environment models (see Appendix D for details).

The baseline methods used in the experiments are: 1) Soft Actor-Critic (SAC), a state-of-the-art RL algorithm, 2) Constrained Policy Optimization (CPO), a commonly used risk-aware safe RL benchmark based on the trust region method [Achiam et al., 2017], and 3) Model Predictive Control (MPC), a widely adopted safe control method for critical systems. Since RL-ACR contains an estimated model through the MPC module, to make the comparison fair, we also include 4) pretrained SAC (SAC-PT) and 5) pretrained CPO (CPO-PT) in the comparison.

Table 1: The number of failed episodes in the first 100 training episodes in the actual environment. The definitions for failure for each environment are detailed in Section 4.1 and Section 4.2. PT stands for pre-training using the estimated model.

	SAC	SAC-PT	CPO	CPO-PT	MPC	RL-ACR
GLUCOSE	100	100	100	100	0	0
ACROBOT	69	1	73	1	0	0
MOUNTCAR	2	1	2	24	0	0
PENDULUM	22	1	26	0	0	0
CARTPOLE	100	21	100	2	0	0

SAC’s implementation follows Stable-Baselines3 [Raffin et al., 2021] and Huang et al. [2022]. The MPC implementation is based on do-mpc [Fiedler et al., 2023], in which the optimization in Eq. (7) relies on Andersson et al. [2019] and Wächter and Biegler [2006]. CPO’s implementation follows Sikchi [2021]. The set of hyperparameters used by RL-ACR is detailed in Table 3.

4.1 Bi-hormonal Glucose Regulation

Regulating the blood glucose level is a critical medical control problem. The objective is to inject hormones for maintaining the blood glucose level, denoted by G , especially in the face of meal-induced carbohydrate disturbances. Crossing certain safe boundaries of G can lead to catastrophic health consequences (hyperglycemia or hypoglycemia). Here, a recent blood glucose model proposed by Herrero et al. [2013] and Kalisvaart et al. [2023] is used, which extends the conventional single-action (insulin injection) models by a second action (glucagon injection). The extended action space leads to potentially better regulation performance but also increases the regulation complexity. The system has 12 state variables modeled by a set of Ordinary Differential Equations (ODEs):

$$\begin{aligned}\dot{\mathbf{x}} &= [\dot{Q}_1, \dot{Q}_2, \dot{x}_1, \dot{x}_2, \dot{x}_3, \dot{S}_1, \dot{S}_2, \dot{I}, \dot{Z}_1, \dot{Z}_2, \dot{N}, \dot{Y}]^T \\ &= f(\mathbf{x}, \mathbf{a}, U_G),\end{aligned}\tag{13}$$

where $\mathbf{a} = [a_I, a_N]$ are the actions, with a_I and a_N being the amount of injected insulin and glucagon, respectively. U_G denotes the disturbance of carbohydrates in a meal. Among the 12 states, only Q_1 is observable through the measurable blood glucose level G since $Q_1 = GV_G$ and V_G is a fixed parameter for glucose distribution volume. State feedback is applied to Q_1 to maintain close-loop control. See Appendix D.1 for the detailed system model in Eq. (13).

In this environment, a time-varying disturbance $U_G \propto t \cdot e^{-c \cdot t}$ derived by Hovorka et al. [2004] is introduced over time to simulate the ingestion of 80 g of carbohydrates. The objective is to regulate G to the target range between 3.9 and 7.8 mmol/L (shaded green in Fig. 2). Failure is defined as G reaching $G > 25$ mmol/L (hyperglycemia) or $G < 3$ mmol/L (hypoglycemia); these unsafe ranges are shaded red in Fig. 2. The parameters of the estimated model (available to the MPC module) and the actual environment are listed in Appendix Table 4.

The RL reward for this environment is a piece-wise linear function given in Eq. (24) in the Appendix. The MPC setting follows Kalisvaart et al. [2023], where the stage and terminal costs are given by $J_k(\mathbf{s}, \mathbf{a}) = J_N(\mathbf{s}, \mathbf{a}) = 3e^3 \times (G - 6)^2 + 8e^4 a_I + 2e^4 a_I a_N + 1e^6 a_N + 5e^7 \max(0, 6 - G)$. For CPO, the constraint function is $c = 10 \max(0, 6 - G)^2$.

Safety test. We first compare the safety of the methods in regulating blood glucose. The first row of Table 1 shows the number of failed episodes (G exceeding $[3, 25]$ bounds) out of 100 consecutive runs while the methods (except for MPC) learn the environment through interactions. It is seen that CPO does not offer any safety guarantee during the initial stage of training. SAC consistently fails during its policy exploration, which is unacceptable in critical applications. This issue persists even if the SAC is trained by the estimated model. Only MPC and RL-ACR guarantee safety in all episodes (SAC, SAC-PT, CPO, and CPO-PT failed in all 100 episodes).

Performance test. As indicated in Table 1, only MPC and RL-ACR can be safely applied to the actual environment in a single-life setting. CPO, SAC, and SAC-PT violate the safety bounds every run. SAC-PT, being trained on the estimated model, is overall safer compared to SAC and CPO (see other rows of Table 1), so we keep it in the rest of our performance analysis.

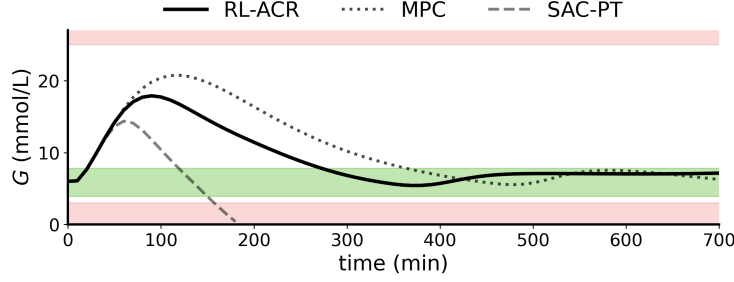


Figure 2: Performance comparison in the bi-hormonal glucose regulation environment. A disturbance $U_G \propto t \cdot e^{-c \cdot t}$ is introduced to simulate meal ingestion. The objective is to regulate the blood glucose level to the target range (the green shade) without violating safety bounds (the red shade).

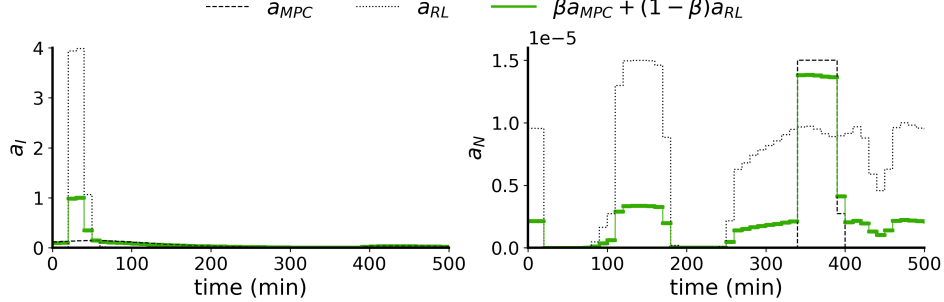


Figure 3: The policy of RL-ACR depicted over time. We plot the actions proposed by the MPC module, RL module, and their combination by the attention weight β . The figure shows the actions of insulin injection a_I (top) and glucagon injection a_N (bottom).

Fig. 2 shows the performance comparison in the glucose regulation environment. SAC-PT causes severe hypoglycemia when deployed to the actual model, but MPC and RL-ACR keep glucose levels within the safe range. RL-ACR achieves the best performance, clearly improving the MPC policy based on an estimated model. Fig. 3 shows the MPC action a_{MPC} and the RL action a_{RL} . The combined action $\beta a_{MPC} + (1 - \beta) a_{RL}$ (the green line in Fig. 3) is the actual policy of RL-ACR. It can be seen in Fig. 3 that MPC only injects small doses of insulin to avoid the danger of hypoglycemia, which results in poor management of high G level under disturbance. Meanwhile, model-free RL finds a better policy, resulting in better overall performance. The performance metrics are summarised later in Table 2

4.2 Classic Control Environments

In this section, we perform experiments on the continuous versions of the four classic control environments: Acrobot, Mountain Car, Pendulum, and Cart Pole [Towers et al., 2023]. Depending on the specific environment, we define “failure” as either reaching states that violate the safety bounds (for Cart Pole) or failing to reach the control objective within the allocated time frame (for Acrobot, Mountain Car, and Pendulum). For more details, such as the MPC cost function and CPO constraint functions, see Appendix E. Notice that, for environments where failure is defined as not achieving the objective in a given time (Acrobot, Mountain Car, and Pendulum), CPO triggers the recovery policy proposal and minimizes the constraint cost first [Achiam et al., 2017].

The *Acrobot* environment simulates two links connected by a joint, with one end of the connected links fixed. The links start facing downward. The objective is to swing the free end (the tip) above a given target height as quickly as possible by applying torque to the joint. Failure is defined as the tip

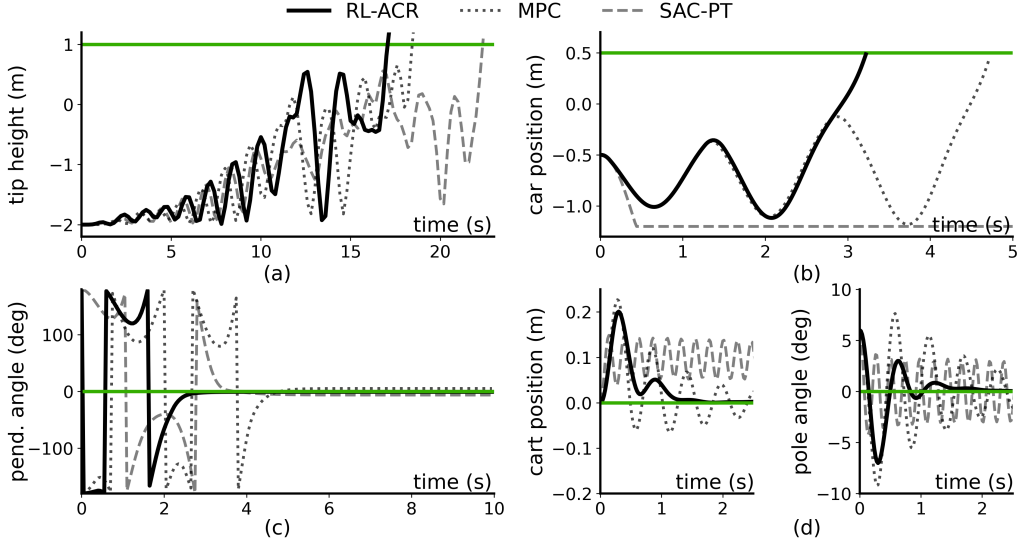


Figure 4: Performance comparison on classic control environments. The green line indicates the target state. (a) The Acrobot environment. (b) The Mountain Car environment. (c) The Pendulum environment. (d) The Cart Pole environment.

not reaching the target height in 200 time steps. The estimated model and the actual environment are detailed in Appendix D.2.

The *Mountain Car* environment simulates a car placed in the valley of a sinusoidal. The objective is to drive the car to the target location on the sinusoidal by applying force either left or right. Failure is defined as the car not reaching the target location in 1000 time steps. The estimated model and the actual environment are detailed in Appendix D.3.

The *Pendulum* environment simulates a hanging pole fixed at one end. The pendulum starts facing downward. The objective is to swing the pole upright and keep it in that position. Failure occurs if the pole does not attain and maintain an upward position (angular velocity not exceeding ± 0.1) within 300 time steps. The estimated model and the actual environment are detailed in Appendix D.4.

The *Cart Pole* environment simulates a pole attached to a cart that moves along a track. The pole starts with a 6-degree tilt. The objective is to maintain an upright position of the pole by applying force to the cart. Failure is defined as the tilt exceeding the ± 12 degrees angle. The estimated model and the actual environment are detailed in Appendix D.5.

Safety test. We first compare the safety of the methods using the concept of failure defined for each environment. We run the methods for 100 consecutive episodes on each environment. Table 1 shows the number of episodes that terminate with failure. CPO and SAC cause failures during training in all environments. SAC-PT, even with a strategy trained by the estimated model, still fails due to its stochastic policy exploration in the actual environment. Only MPC and RL-ACR can guarantee safety in all episodes, as suggested by 0 failed episodes in all environments reported in Table 1.

Performance test. As indicated in Table 1, SAC-PT, being trained on the estimated model, is overall safer compared to SAC and CPO, so we keep it in the performance analysis.

Fig. 4 shows the behavior of SAC-PT, MPC, and our RL-ACR in the classic control environments. Fig. 4 (b) shows an instance where the SAC-PT strategy, although fully trained on an estimated model, still has a biased understanding of the car position and fails in the actual environment. Despite only having an inaccurate model, MPC achieves the control objectives without failures. Our results suggest that RL-ACR learns similar strategies to MPC (note the similar trajectories in Fig. 4). This shows that the off-policy RL module of RL-ACR effectively learns from the sub-optimum MPC policy. However, the adaptation to the actual environment enabled by model-free RL allows RL-ACR

Table 2: Summarized performance metrics (See Sections 4.2 and 4.1 for detail). The metrics are i) the sum of absolute deviations ($\|\mathbf{y} - \mathbf{r}\|_1$), ii) Response Time (RT) or Time Within Range (TWR) depending on the nature of the problem, and iii) Normalized Control Effort (NCE). “N/A” indicates failing to achieve the objective, or violating the environment’s safety bounds, in which case, the performance metric cannot be calculated over the whole duration of the episodes.

ENVIRONMENT	CRITERION	SAC-PT	MPC	RL-ACR
GLUCOSE	$\ \mathbf{y} - \mathbf{r}\ _1 (\times 10^2)$	N/A	9.52	8.62
	TWR	N/A	0.67	0.76
	NCE	N/A	0.10	0.06
ACROBOT	$\ \mathbf{y} - \mathbf{r}\ _1 (\times 10^2)$	1.22	1.14	1.09
	RT	113	94	87
	NCE	0.64	0.94	0.87
MOUNT CAR	$\ \mathbf{y} - \mathbf{r}\ _1 (\times 10^2)$	N/A	2.71	1.80
	RT	N/A	237	162
	NCE	N/A	0.99	0.95
PENDULUM	$\ \mathbf{y} - \mathbf{r}\ _1 (\times 10^3)$	7.73	11.54	6.47
	TWR	0.67	0.55	0.75
	NCE	0.26	0.44	0.17
CART POLE	$\ \mathbf{y} - \mathbf{r}\ _1 (\times 10^1)$	4.52	1.60	0.93
	TWR	0.15	0.75	0.89
	NCE ($\times 10^{-1}$)	3.65	0.12	0.03

to achieve the control objective faster (see Fig. 4 (a), (b), and (c), and note the significantly lower number of oscillations in the RL-ACR trajectory in Fig. 4 (d)).

Table 2 summarizes the performance metrics of SAC-PT, MPC, and RL-ACR on all environments. As Table 2 shows, RL-ACR outperforms MPC in all environments. This demonstrates RL-ACR’s ability to adapt to the actual environment and improve the original sub-optimum MPC policy. RL-ACR also achieves lower NCEs in four out of five environments. This further testifies a better policy is found by RL-ACR, as the stronger performance on other metrics does not come at the cost of taking more drastic actions.

5 Conclusions and Discussion

Controlling critical systems, where unsafe control actions have catastrophic consequences, has huge applications from engineering to medicine. We show that appropriate embedding of a control regularization can ensure safety as RL exploration searches for the optimum policy. The proposed method, RL-ACR, dynamically tunes the combination of an RL agent and a control regularizer via a learnable focus weight, which prevents early incorporation of poorly learned RL policy. As revealed in extensive numerical experiments in a single-life setting, the safety of RL-ACR outperforms the baselines. Meanwhile, RL-ACR was able to learn from explorations in the actual environment and find policies with stronger performance.

One limitation of our setting is the assumption that the estimated model has a reasonable accuracy to ensure safety in the actual environment. Although this assumption is common in the safe RL literature, one possible direction for future works is to design algorithms with more flexibility against deviations of the estimated model from the actual environment. A potential approach is to update the estimated model parameters using observations in the actual environment. However, one practical concern is that all parameters are assumed to be unknown and need updating, but only a small number of transitions are observed in the actual environment. In this case, managing controllability, convergence, and safety requires careful design and tuning.

6 Related Works

Similar to the characterization by García and Fernández [2015] and Krasowski et al. [2022], we divide existing safe RL works into roughly two categories. The first category does not require knowledge of the system dynamics. These methods often rely on probabilistic approaches [Geibel and Wysotzki, 2005, Liu et al., 2022] or constrained MDP [Achiam et al., 2017, Yang et al., 2020]. In more recent works, learned models are used to filter out unsafe actions [Bharadhwaj et al., 2020]. However, these methods need to observe failures to estimate the safety cost, thus do not ensure safety during training. Since this work requires safety while training in the actual environment, these methods do not apply.

The second category assumes known system dynamics. Some works use the control barrier function to enforce safety [Cheng et al., 2019b]. However, these works require a given valid safe set, which is unobtainable when the estimated model and the actual environment are different. Some methods compute a model-based projection to verify the safety of actions [Bastani, 2021, Kochdumper et al., 2023, Fulton and Platzer, 2018]. However, the scalability of these verification-based methods is a huge problem. Although Anderson et al. [2020] proposed to use neurosymbolic representations to alleviate the computational cost of verification, the computational cost is still high.

Gros and Zanon [2019] and Zanon et al. [2020] proposed to use MPC as the policy generator, and use RL to dynamically tune the MPC parameters in the cost functions and the system model. Assuming the discrepancy between the estimated model and the actual environment exists in the form of model parameters, the tuning increases MPC performance. However, this is a strong assumption since there are other discrepancies, such as neglected dynamics, discretization errors, etc. Instead, RL-ACR in this work can theoretically converge to the optimum policy.

It is important to note that, although the MPC policy accelerates the learning of the RL agent, RL-ACR is not a special case of transferring a learned policy. The main role of MPC in RL-ACR is to keep the actions safe in the actual environment, instead of transferring knowledge. Transfer learning in RL studies the effective reuse of knowledge, especially across different tasks [Taylor and Stone, 2009, Glatt et al., 2020]. By reusing prior knowledge, transferred RL agents skip the initial random trial-and-error and drastically increase sampling efficiency [Karimpanal et al., 2020, Da Silva and Costa, 2019]. However, transferred RL agents are not inherently risk-aware, and thus can still steer the actual environment into unsafely. For this reason, transferred RL policies are rarely considered a valid method for ensuring safety. On the other hand, the control regularizer in this work forecasts system behaviors and hard-codes safety into constrained optimizations.

References

- Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International conference on machine learning*, pages 22–31. PMLR, 2017.
- Greg Anderson, Abhinav Verma, Isil Dillig, and Swarat Chaudhuri. Neurosymbolic reinforcement learning with formally verified exploration. *Advances in neural information processing systems*, 33:6172–6183, 2020.
- Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019. doi: 10.1007/s12532-018-0139-4.
- Osbert Bastani. Safe reinforcement learning with nonlinear dynamics via model predictive shielding. In *2021 American control conference (ACC)*, pages 3488–3494. IEEE, 2021.
- Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.
- Homanga Bharadhwaj, Aviral Kumar, Nicholas Rhinehart, Sergey Levine, Florian Shkurti, and Animesh Garg. Conservative safety critics for exploration. *arXiv preprint arXiv:2010.14497*, 2020.
- Annie Chen, Archit Sharma, Sergey Levine, and Chelsea Finn. You only live once: Single-life reinforcement learning. *Advances in Neural Information Processing Systems*, 35:14784–14797, 2022.

- Richard Cheng, Gábor Orosz, Richard M Murray, and Joel W Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3387–3395, 2019a.
- Richard Cheng, Abhinav Verma, Gabor Orosz, Swarat Chaudhuri, Yisong Yue, and Joel Burdick. Control regularization for reduced variance reinforcement learning. In *International Conference on Machine Learning*, pages 1141–1150. PMLR, 2019b.
- Felipe Leno Da Silva and Anna Helena Reali Costa. A survey on transfer learning for multiagent reinforcement learning systems. *Journal of Artificial Intelligence Research*, 64:645–703, 2019.
- Jonas Degraeve, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022.
- M Diana and JJBJoS Marescaux. Robotic surgery. *Journal of British Surgery*, 102(2):e15–e28, 2015.
- Felix Fiedler, Benjamin Karg, Lukas Lücken, Dean Brandner, Moritz Heinlein, Felix Brabender, and Sergio Lucia. do-mpc: Towards fair nonlinear and robust model predictive control. *Control Engineering Practice*, 140:105676, 2023.
- Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.
- Nathan Fulton and André Platzer. Safe reinforcement learning via formal methods: Toward safe control through proof and learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.
- Javier Garcia and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.
- Peter Geibel and Fritz Wysotzki. Risk-sensitive reinforcement learning applied to control under constraints. *Journal of Artificial Intelligence Research*, 24:81–108, 2005.
- Ruben Glatt, Felipe Leno Da Silva, Reinaldo Augusto da Costa Bianchi, and Anna Helena Reali Costa. Decaf: deep case-based policy inference for knowledge transfer in reinforcement learning. *Expert Systems with Applications*, 156:113420, 2020.
- Sébastien Gros and Mario Zanon. Data-driven economic nmpc using reinforcement learning. *IEEE Transactions on Automatic Control*, 65(2):636–648, 2019.
- Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.
- Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.
- Pau Herrero, Pantelis Georgiou, Nick Oliver, Monika Reddy, Desmond Johnston, and Christofer Toumazou. A composite model of glucagon-glucose dynamics for in silico testing of bihormonal glucose controllers. *Journal of diabetes science and technology*, 7(4):941–951, 2013.
- Julia Hippisley-Cox, Carol Coupland, and Peter Brindle. Development and validation of qrisk3 risk prediction algorithms to estimate future risk of cardiovascular disease: prospective cohort study. *bmj*, 357, 2017.
- Roman Hovorka, Fariba Shojaee-Moradie, Paul V Carroll, Ludovic J Chassin, Ian J Gowrie, Nicola C Jackson, Romulus S Tudor, A Margot Umpleby, and Richard H Jones. Partitioning glucose distribution/transport, disposal, and endogenous production during ivgtt. *American Journal of Physiology-Endocrinology and Metabolism*, 282(5):E992–E1007, 2002.

- Roman Hovorka, Valentina Canonico, Ludovic J Chassin, Ulrich Haueter, Massimo Massi-Benedetti, Marco Orsini Federici, Thomas R Pieber, Helga C Schaller, Lukas Schaupp, Thomas Vering, et al. Nonlinear model predictive control of glucose concentration in subjects with type 1 diabetes. *Physiological measurement*, 25(4):905, 2004.
- Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022. URL <http://jmlr.org/papers/v23/21-1342.html>.
- Dylan Kalisvaart, Jorge Bonekamp, and Sergio Grammatico. Bi-hormonal linear time-varying model predictive control for blood glucose regulation in type 1 diabetes patients. In *2023 IEEE Conference on Control Technology and Applications (CCTA)*, pages 552–558. IEEE, 2023.
- Thommen George Karimpanal, Santu Rana, Sunil Gupta, Truyen Tran, and Svetha Venkatesh. Learning transferable domain priors for safe exploration in reinforcement learning. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–10. IEEE, 2020.
- Niklas Kochdumper, Hanna Krasowski, Xiao Wang, Stanley Bak, and Matthias Althoff. Provably safe reinforcement learning via action projection using reachability analysis and polynomial zonotopes. *IEEE Open Journal of Control Systems*, 2:79–92, 2023.
- Matthieu Komorowski, Leo A Celi, Omar Badawi, Anthony C Gordon, and A Aldo Faisal. The artificial intelligence clinician learns optimal treatment strategies for sepsis in intensive care. *Nature medicine*, 24(11):1716–1720, 2018.
- Hanna Krasowski, Jakob Thumm, Marlon Müller, Lukas Schäfer, Xiao Wang, and Matthias Althoff. Provably safe reinforcement learning: A theoretical and experimental comparison. *arXiv preprint arXiv:2205.06750*, 2022.
- Juliane Liepe, Paul Kirk, Sarah Filippi, Tina Toni, Chris P Barnes, and Michael PH Stumpf. A framework for parameter estimation and model selection from experimental data in systems biology using approximate bayesian computation. *Nature protocols*, 9(2):439–456, 2014.
- Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.
- Long-Ji Lin. *Reinforcement learning for robots using neural networks*. Carnegie Mellon University, 1992.
- Zuxin Liu, Zhepeng Cen, Vladislav Isenbaev, Wei Liu, Steven Wu, Bo Li, and Ding Zhao. Constrained variational policy optimization for safe reinforcement learning. In *International Conference on Machine Learning*, pages 13644–13668. PMLR, 2022.
- Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744, 2022.
- Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL <http://jmlr.org/papers/v22/20-1364.html>.
- Benjamin Recht. A tour of reinforcement learning: The view from continuous control. *Annual Review of Control, Robotics, and Autonomous Systems*, 2:253–279, 2019.
- Harshit Sikchi. pytorch-CPO, 2021. URL https://github.com/hari-sikchi/pytorch_CPO.

- David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.
- Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7), 2009.
- Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. Gymnasium, March 2023. URL <https://zenodo.org/record/8127025>.
- Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106:25–57, 2006.
- Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J Ramadge. Projection-based constrained policy optimization. *arXiv preprint arXiv:2010.03152*, 2020.
- Dongjie Yu, Haitong Ma, Shengbo Li, and Jianyu Chen. Reachability constrained reinforcement learning. In *International Conference on Machine Learning*, pages 25636–25655. PMLR, 2022.
- Mario Zanon, Vyacheslav Kungurtsev, and Sébastien Gros. Reinforcement learning based on real-time iteration nmpc. *IFAC-PapersOnLine*, 53(2):5213–5218, 2020.

Appendix

A Proof of Lemma 1

Lemma 1. Assuming the convergence of the RL policy π_θ , the mean of the combined action $\bar{a}(s)$ is the solution to the following regularized optimization with regularization parameter $\lambda = \frac{\beta}{1-\beta}$:

$$\bar{a}(s) = \operatorname{argmin}_a \|a(s) - \operatorname{argmax}_{\pi_\theta} Q(s, \pi_\theta(s))\|_\Sigma + \frac{\beta}{1-\beta} \|a(s) - a_{MPC}(s)\|_\Sigma. \quad (14)$$

Proof. For a fixed β , the proof is similar to the proof by Cheng et al. [2019b]. Because the RL policy is a Gaussian distributed policy, i.e. $\pi_\theta(a|s) \sim \mathcal{N}(\bar{a}_{RL}(s), \Sigma)$, the combined policy is also a Gaussian distribution:

$$a(s) \sim \mathcal{N}(\beta a_{MPC} + (1-\beta)\bar{a}_{RL}(s), (1-\beta)^2 \Sigma). \quad (15)$$

The distribution is equivalent to the product of two multivariate Gaussian distributions:

$$\mathcal{N}(\beta a_{MPC} + (1-\beta)\bar{a}_{RL}(s), (1-\beta)^2 \Sigma) = \frac{1}{1-\beta} \mathcal{N}(a_{MPC}, \frac{1}{\beta} \Sigma) \cdot \mathcal{N}(\bar{a}_{RL}(s), \frac{1}{1-\beta} \Sigma) \quad (16)$$

Define $\|a_1 - a_2\|_\Sigma = (a_1 - a_2)^T \Sigma^{-1} (a_1 - a_2)$. The probability density function can be written as:

$$\begin{aligned} f(a(s)) &= (1-\beta)^{-1} (2\pi)^{-k/2} \beta^{-k/2} |\Sigma|^{1/2} \exp\left(-\frac{\beta}{2} \|a(s) - a_{MPC}(s)\|_\Sigma\right) \\ &\quad \times (2\pi)^{-k/2} (1-\beta)^{-k/2} |\Sigma|^{1/2} \exp\left(-\frac{1-\beta}{2} \|a(s) - \bar{a}_{RL}(s)\|_\Sigma\right) \\ &= c \exp\left(\frac{1-\beta}{2} \left(-\|a(s) - \bar{a}_{RL}(s)\|_\Sigma - \frac{\beta}{1-\beta} \|a(s) - a_{MPC}(s)\|_\Sigma\right)\right), \\ c &= \frac{1}{(2\pi)^k \beta^{k/2} (1-\beta)^{(k+2)/2} |\Sigma|}. \end{aligned} \quad (17)$$

Since β is in range $(0, 1)$, $f(a(s))$ monotonically decreases. Therefore, the probability of $a(s)$ is maximized when the exponential term is minimized, leading to the following optimization problem:

$$\bar{a}(s) = \operatorname{argmin}_a \|a(s) - \bar{a}_{RL}(s)\|_\Sigma + \frac{\beta}{1-\beta} \|a(s) - a_{MPC}(s)\|_\Sigma. \quad (18)$$

Assuming the RL policy π_θ converges to $\operatorname{argmax}_{\pi_\theta} Q(s, \pi_\theta(s))$, Eq. (18) can be written as:

$$\bar{a}(s) = \operatorname{argmin}_a \|a(s) - \operatorname{argmax}_{\pi_\theta} Q(s, \pi_\theta(s))\|_\Sigma + \frac{\beta}{1-\beta} \|a(s) - a_{MPC}(s)\|_\Sigma. \quad (19)$$

□

B Proof of Theorem 1

Theorem 1. Let $D_{TV}(\cdot, \cdot)$ denote the total variance distance. The combined policy π has the following bias:

$$D_{TV}(\pi, \pi^*) \geq D_{TV}(\pi_{MPC}, \pi^*) - (1-\beta) D_{TV}(\pi_{MPC}, \pi_\theta). \quad (20)$$

However, assume RL converges to the optimum policy π^* that satisfies $Q^{\pi^*}(s, \pi^*(s)) > Q^\pi(s, a), \forall a \in \mathcal{A} \setminus \pi^*(s)$, the combined policy π is unbiased with $D_{TV}(\pi, \pi^*) = 0$.

Proof. The proof for the lower bound is similar to the proof Cheng et al. [2019b]. First, we expand the total variance distance between the MPC policy and the combined policy as:

$$\begin{aligned} D_{TV}(\pi_{MPC}, \pi) &= \sup_{(s,a) \in \mathcal{S} \times \mathcal{A}} |\pi_{MPC} - \beta \pi_{MPC} - (1-\beta) \pi_\theta| \\ &= (1-\beta) D_{TV}(\pi_{MPC}, \pi_\theta) \end{aligned} \quad (21)$$

Following triangle inequality, the total variance distance between the combined policy and the optimal policy is as follows:

$$D_{TV}(\pi, \pi^*) \geq D_{TV}(\pi_{MPC}, \pi^*) - D_{TV}(\pi_{MPC}, \pi_\theta) \quad (22)$$

To prove the upper bound on $D_{TV}(\pi, \pi^*)$ after convergence, we first examine the convergence of β . According to the assumption, for state s , there exists one optimum policy $Q^{\pi^*}(s, \pi^*(s)) > Q^\pi(s, a), \forall a \in \mathcal{A} \setminus \pi^*(s)$. If RL learns the optimum action (i.e. $\pi_\theta(s) = \pi^*(s)$), then the optimum in Eq. (10) is achieved by $\beta = 0$. Because if $\beta \neq 0$ then $\beta\pi_{MPC}(s) + (1 - \beta)\pi_\theta(s) = \beta\pi_{MPC}(s) + (1 - \beta)\pi^*(s) \neq \pi^*(s)$. The inequality follows as $\pi_{MPC} \neq \pi^*$ because the MPC is based on a sub-optimum model. Because $\beta = 0$, the combined policy has variance equals to π_θ and mean equals to the following according to Lemma 1:

$$\begin{aligned} \bar{a}(s) &= \operatorname{argmin}_a \|a(s) - \operatorname{argmax}_{\pi_\theta} Q(s, \pi_\theta(s))\|_\Sigma + \frac{\beta}{1 - \beta} \|a(s) - a_{MPC}(s)\|_\Sigma \\ &= \operatorname{argmin}_a \|a(s) - \operatorname{argmax}_{\pi_\theta} Q(s, \pi_\theta(s))\|_\Sigma = \bar{a}_{RL}. \end{aligned} \quad (23)$$

Thus the the combined policy π is unbiased with $D_{TV}(\pi, \pi^*) = D_{TV}(\pi_\theta, \pi^*) = 0$ $\square$

C Hyperparameters

The same set of parameters was utilized in all environments to demonstrate the robustness of RL-ACR against the selection of hyperparameters. The hyperparameters in RL-ACR are given in Table 3.

Table 3: RL-ACR Hyperparameters

HYPERPARAMETER	VALUE
discount factor γ	0.99
target network coefficient τ	0.05
entropy regularization coefficient α	0.2
time to update	500
update frequency	1
policy frequency	2
batch size $ \mathcal{B} $	256
attention weight learning rate lr_β	1×10^{-5}
Q -network learning rate $lr_{Q_\phi(\cdot)}$	1×10^{-3}
policy network learning rate $lr_{\pi_\theta(\cdot)}$	3×10^{-4}
clip rate c for ∇_β	10
action minimum log std. $\log \sigma_\theta(\mathbf{s})_{\min}$	-5
action maximum log std. $\log \sigma_\theta(\mathbf{s})_{\max}$	2

D Estimated models and the actual environment models

D.1 Bi-hormonal Glucose Model

The blood glucose model used in this work contains 12 state variables regulated by the ODEs given below:

$$\begin{aligned} \dot{Q}_1 &= -F_{01}^c(G) - x_1 Q_1 + k_{12} Q_2 - F_R \\ &\quad + (1 - x_3) \text{EGP}_0 + c_{conv} U_G + Y Q_1 \\ \dot{Q}_2 &= x_1 Q_1 - (k_{12} + x_2) Q_2 \\ \dot{x}_1 &= -k_{a1} x_1 + k_{b1} I \\ \dot{x}_2 &= -k_{a2} x_2 + k_{b2} I \\ \dot{x}_3 &= -k_{a3} x_3 + k_{b3} I \\ \dot{S}_1 &= u_I - \frac{S_1}{t_{max, I}} \end{aligned}$$

$$\begin{aligned}
\dot{S}_2 &= \frac{S_1}{t_{max,I}} - \frac{S_2}{t_{max,I}} \\
\dot{I} &= \frac{S_2}{V_I t_{max,I}} - k_e I \\
\dot{Z}_1 &= u_N - \frac{Z_1}{t_{max,N}} \\
\dot{Z}_2 &= \frac{Z_1}{t_{max,N}} - \frac{Z_2}{t_{max,N}} \\
\dot{N} &= -k_N(N - N_b) + \frac{Z_2}{V_N t_{max,N}} \\
\dot{Y} &= -pY + pS_N(N - N_b),
\end{aligned}$$

The last four equations form the glucagon subsystem proposed by Herrero et al. [2013] and Kalisvaart et al. [2023]. These newly proposed dynamics allow an expanded action space. While conventional blood glucose models have one action, insulin injection [Hovorka et al., 2004], the glucose model in this work has two actions u_I and u_N , representing the injection of insulin and glucagon. The measurable blood glucose mass $G = Q_1/V_G$ is lowered by insulin injection and raised by glucagon injection. The intermediate variables depend on the value of G , as shown below:

$$\begin{aligned}
F_{01}^c &= \begin{cases} F_{01}, & G \geq 4.5 \text{ mmol L}^{-1} \\ F_{01}G/4.5, & \text{otherwise} \end{cases}, \\
F_R &= \begin{cases} 0.003(G - 9)V_G, & G \geq 9 \text{ mmol L}^{-1} \\ 0, & \text{otherwise} \end{cases}.
\end{aligned}$$

The term U_G represents the time-varying disturbance caused by the injection of D_G of carbohydrates:

$$U_G = \frac{D_G A_G}{t_{max,G}^2} \cdot t \cdot e^{-t/t_{max,G}},$$

the model parameters adopted by the estimated model and the actual environment are given by Table 4. The reward function for blood glucose regulation is a piece-wise linear function given below:

$$r = \begin{cases} -400 \times (6 - G), & G < 6 \\ 0, & 6 \leq G < 7.8 \\ -\frac{40}{7.2} \times (G - 7.8), & 7.8 \leq G < 20 \\ -300, & 7.8 \leq G < 25 \\ -2000, & G \geq 25 \end{cases}. \quad (24)$$

D.2 Acrobot Implementation

In this work, the system model utilized by MPC and RL-ACR for the Acrobot environment is defined as:

$$\begin{aligned}
\ddot{\theta}_1 &= \frac{-(d_2 \ddot{\theta}_1 + \phi_1)}{d_1} \\
\ddot{\theta}_2 &= \frac{a + d_2 \phi_1 / d_1 - m_2 l_1 l_2 c \dot{\theta}_1^2 \sin \theta_2 - \phi_2}{m_2 l_2^2 c^2 + I_2 - d_2^2 / d_1},
\end{aligned}$$

where a is the action, the default model parameters are given in Table 5, and the intermediate variables are:

$$\begin{aligned}
d_1 &= m_1 l_1^2 c^2 + m_2 (l_1^2 + l_2^2 c^2 + 2l_1 l_2 c \cos \theta_2) + I_1 + I_2 \\
d_2 &= m_2 (l_2^2 c^2 + l_1 l_2 c \cos \theta_2) + I_2 \\
\phi_2 &= m_2 l_2 c g \cos(\theta_1 + \theta_2 - \pi/2)
\end{aligned}$$

Table 4: Glucose parameters for the estimated model and the actual environment

PARAMETER	UNIT	ESTIMATED MODEL	ACTUAL ENVIRONMENT
D_G	kg	0.08	0.08
V_G	L/kg	0.18	0.14
k_{12}	min^{-1}	0.0343	0.0968
F_{01}	$\text{mmol}/(\text{kg min})$	0.0121	0.0199
EGP_0	$\text{mmol}/(\text{kg min})$	0.0148	0.0213
A_g	-	0.8	0.8
$t_{max,G}$	min	40	40
$t_{max,I}$	min	55	55
V_I	L kg^{-1}	0.12	0.12
k_e	min^{-1}	0.138	0.138
k_{a1}	min^{-1}	0.0031	0.0088
k_{a2}	min^{-1}	0.0752	0.0302
k_{a3}	min^{-1}	0.0472	0.0118
k_{b1}	$\text{L}/(\text{min}^2 \text{ mU})$	29.4×10^{-4}	86.1×10^{-4}
k_{b2}	$\text{L}/(\text{min}^2 \text{ mU})$	0.9×10^{-4}	4.7×10^{-4}
k_{b3}	$\text{L}/(\text{min mU})$	401×10^{-4}	720×10^{-4}
$t_{max,N}$	min	32.46	20.59
k_N	min^{-1}	0.620	0.735
V_N	mL kg^{-1}	16.06	23.46
p	min^{-1}	0.016	0.074
$S_N \cdot 10^{-4}$	mL/pg min^{-1}	1.96	1.98
M_g	g/mol	180.16	180.16
BW	kg	68.5	68.5

$$\phi_1 = -m_2 l_1 l_2 c \dot{\theta}_2^2 \sin \theta_2 - 2m_2 l_1 l_2 c \dot{\theta}_1 \dot{\theta}_2 \sin \theta_2 + (m_1 l_1 c + m_2 l_1) g \cos(\theta_1 - \pi/2) + \phi_2.$$

For this particular environment, we utilized a custom RL reward function. This allows SAC and CPO to achieve the control objective within 100 episodes, thus benefiting the comparison in Table 1. The reward function is:

$$r = \begin{cases} -2 - l_1 \cos \theta_1 - l_2 \cos \theta(\theta_1 + \theta_2), & -l_1 \cos \theta_1 - l_2 \cos \theta(\theta_1 + \theta_2) < l_2 \\ +100, & \text{otherwise} \end{cases}.$$

Table 5: Acrobot parameters for the estimated model and the actual environment

PARAMETER	ESTIMATED MODEL	ACTUAL ENVIRONMENT
m_1 (kg)	1.0	1.0
m_2 (kg)	1.0	1.0
l_1 (m)	1.0	1.1
l_2 (m)	1.0	1.0
c	0.5	0.5
I_1 ($\text{kg} \cdot \text{m}^2$)	1.0	1.0
I_2 ($\text{kg} \cdot \text{m}^2$)	1.0	1.0
g ($\text{m} \cdot \text{s}^{-2}$)	9.8	9.8

D.3 Mountain Car Implementation

In this work, the system model utilized by MPC and RL-ACR for the Mountain Car environment is defined as:

$$\begin{aligned} x' &= x + v \\ v' &= v + ac_f - 0.0025 \cos(3(x + x_0)), \end{aligned}$$

where a is the action, the default model parameters are given in Table 6.

Table 6: Mountain Car parameters for the estimated model and the actual environment

PARAMETER	ESTIMATED MODEL	ACTUAL ENVIRONMENT
c_f	0.0015	0.0015
x_0	0	0.4

D.4 Pendulum Implementation

In this work, the system model utilized by MPC and RL-ACR for the Pendulum environment is defined as:

$$\begin{aligned}\theta' &= \theta + \dot{\theta}\Delta t \\ \dot{\theta}' &= \dot{\theta} + \left(\frac{3g}{2l}\sin\theta + \frac{3}{ml^2}a\right)\Delta t,\end{aligned}$$

where a is the action, the default model parameters are given in Table 7.

Table 7: Pendulum parameters for the estimated model and the actual environment

PARAMETER	ESTIMATED MODEL	ACTUAL ENVIRONMENT
g ($\text{m} \cdot \text{s}^{-2}$)	9.8	9.8
m (kg)	1.0	1.0
l (m)	1.0	0.9
Δt (s)	0.05	0.05

D.5 Cart Pole Implementation

In this work, the system model utilized by MPC and RL-ACR for the Acrobot environment is defined as:

$$\begin{aligned}\ddot{\theta} &= \frac{g \sin \theta - d \cos \theta}{l(4/3 - m_p \cos^2 \theta / (m_p + m_c))} \\ \ddot{x} &= d - \frac{m_p l \ddot{\theta} \cos \theta}{m_p + m_c},\end{aligned}$$

where a is the action, the default model parameters are given in Table 8, and the intermediate variable d is:

$$d = \frac{ac_f + m_p l \dot{\theta}^2 \sin \theta}{m_p + m_c}$$

Table 8: Cart Pole parameters for the estimated model and the actual environment

PARAMETER	ESTIMATED MODEL	ACTUAL ENVIRONMENT
g ($\text{m} \cdot \text{s}^{-2}$)	9.8	9.8
m_c (kg)	1.0	0.8
m_p (kg)	0.1	0.3
l (m)	0.5	0.6
c_f	10.0	10.0

E MPC Costs and CPO constraints for the Classic Control Environments

Here we illustrate the CPO constraints and the MPC cost functions in the classic control environments in Section 4.

For the Acrobot environment, the CPO constraint function we utilized was $c = 1 + \cos \theta_1 + \cos(\theta_1 + \theta_2)$. The MPC costs we utilized are:

$$J_k(\mathbf{s}, \mathbf{a}) = J_N(\mathbf{s}, \mathbf{a}) = 100 \times (2 + \cos \theta_1 + \cos(\theta_1 + \theta_2))^2.$$

For the Mountain Car environment, the CPO constraint function we utilized was $c = \max(0, 0.45 - x)^2$. The MPC costs we utilized are:

$$J_k(\mathbf{s}, \mathbf{a}) = J_N(\mathbf{s}, \mathbf{a}) = \max(0, 0.45 - x)^2.$$

For the Pendulum environment, the CPO constraint function we utilized was $c = 2 \max(0, |\theta| - 4)^2$. The MPC costs we utilized are:

$$\begin{aligned} J_k(\mathbf{s}, \mathbf{a}) &= 10\theta^2 + 0.1\dot{\theta}^2 \\ J_N(\mathbf{s}, \mathbf{a}) &= 80\theta^2 + 0.1\dot{\theta}^2. \end{aligned}$$

For the Cart Pole environment, the CPO constraint function we utilized was $c = \max(0, |\theta| - 6)^2$. The MPC costs we utilized are:

$$J_k(\mathbf{s}, \mathbf{a}) = J_N(\mathbf{s}, \mathbf{a}) = 100x^2 + 100\theta^2.$$