Robot Reinforcement Learning on the Constraint Manifold

Anonymous Author(s)
Affiliation
Address
email

Abstract: Reinforcement Learning in robotics is extremely challenging, as these tasks raise many practical issues, which are normally not considered in the Machine Learning literature. One of the most important problems to consider is the necessity of satisfying physical and safety constraints throughout the learning process. While many Safe Exploration and Constrained Reinforcement Learning techniques exist in the machine learning literature, these methods are not yet applicable to real robotics tasks. However, different from generic Reinforcement Learning environments, it's often possible to consider as known both the model of the robotic agent and the mathematical definition of the constraints. Exploiting this knowledge, we are able to derive a method to learn robotics tasks in simulation efficiently while satisfying the constraints during the learning process.

Keywords: Robot Learning, Constrained Reinforcement Learning, Safe Exploration

1 Introduction

Despite the notable success of Deep Reinforcement Learning (RL) in solving complex tasks in the discrete world (e.g., Go [1]), video games (e.g., StarCraft [2]), as well as continuous control problems in simulation [3, 4], applying reinforcement learning in the real world remains a challenging task. One important factor that cannot be neglected in real-world applications is the necessity of satisfying constraints. A variety of different practical considerations can be formulated in the form of constraints, such as safety and mechanical viability. As an example, in the robot manipulation task, the robot should not take actions that damage the environment and can not take actions that exceed its feasible range. However, typical reinforcement learning algorithms, which maximize the cumulative reward by continuous trial and error, do not take into account the satisfaction of constraints during the exploration process. Exploring the environment while meeting the constraints is a very challenging problem.

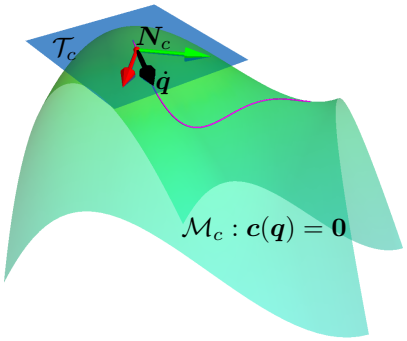


Figure 1: Learning on the Tangent Space of the Constraint Manifold. The constraint set $\boldsymbol{c}(\boldsymbol{q}) = \boldsymbol{0}$ is a differentiable manifold $\mathcal{M}_c$ embedded in original state space. We use a set of basis vectors $\boldsymbol{N}_c$ to represent the span of tangent space $\mathcal{T}_c$. The tangent space velocity/acceleration can be determined by a coordinate based on the bases, and the control action is determined based on the known model of velocity/acceleration, the resulting trajectory is maintained on the constraint manifold.

Safe exploration is an important field of RL which requires to comply with the constraints during the whole learning process [5]. There are several safe exploration frameworks in the literature: a possible direction is proposed in [6, 7] that relies on prior knowledge (policies, value functions) to initialize the system in a safe region and gradually increase the area of exploration using new information obtained from the environment. Other approaches rely on the definition of a safe policy [8, 9],

Submitted to the 5th Conference on Robot Learning (CoRL 2021). Do not distribute.

which tries to pull the agent back to a safe state. However, these choices require excessive work in defining such policy, and safe policies could conflict with each other when multiple constraints are violated. Finally, other methods incorporate model information of constraints with model-free RL algorithms and do not require the definition of a manual policy [10, 11, 12]. In these approaches, at each time step, the agent tries to find the feasible action using constrained optimization techniques.

In this paper, we propose a novel method, Acting on the TAngent Space of the Constraint Manifold (ATACOM), that explores in the tangent space of the constraint manifold, as shown in Figure 1. The constrained RL problem is converted to a typical unconstrained RL problem. This method allows us to utilize *any* model-free RL algorithms while maintaining the constraints below the tolerance. Furthermore, ATACOM can handle both equality and inequality constraints. For example, in the task of a robot wiping a table, the end-effector should always move on the surface of the table while the joint positions and velocities are within its joint limits. In addition, for tasks with equality constraints, our method explores the lower-dimensional manifold embedded in the original action space. To test our method, we demonstrate three different tasks, *CircleMoving*, *PlanarAirHockey* and *IiwaAirHockey*, with different combinations of equality and inequality constraints. We test five state-of-the-art model-free RL algorithms (PPO, TRPO, DDPG, TD3, SAC) in each environment. The result shows that all of the algorithms can learn the policy efficiently while the constraints are maintained below the tolerance.

The advantage of ATACOM can be summarized as follows: (i) can deal both with **equality and inequality constraints**. All of the constraints at each time step are maintained below the tolerance during the whole learning process. (ii) does not require an initial feasible policy, the agent can **learn from scratch**. (iii) requires **no safe backup policy** to move the system back into the safe region. (iv) can be applied to any type of model-free RL algorithm, using both **deterministic and stochastic policies**. (v) can focus the exploration on the **lower dimensional manifold** instead of exploring in the original action space for equality constrained problem. (vi) have **better learning performance** as the inequality constraints restrict to a smaller feasible state-action space. As a downside, our method assumes that the dynamic model is perfectly known: this assumption often does not hold in the real world. However, we believe our method is a step forward in bridging the gap between machine learning and robotics.

Related Work. In the last decades, Constrained Markov Decision Processes (CMDP) [13] has attracted a lot of interest from RL researches, particularly to solve constrained control problems. Under this framework, several different forms of constraints have been studied. One important form of constraint is the expected cost below a threshold. Many works focus on the objective that maximizing the expected return while maintaining the expected cost below a threshold [11, 14, 15, 16, 17, 18, 19]. Different types of constrained optimization techniques are applied in the policy update process. Achiam, et al., proposed a trust-region method Constrained Policy Optimization (CPO) inspired from Trust Region Policy Optimization (TRPO) [18]. Liu, et al., proposed the interior point method for the policy optimization [15]. Another type of approach is to adapt the Lagrangian relaxation method for the constrained RL setting [13, 16, 14, 17]. Lastly, Chow et al., proposed a method to generate the Lyapunov function which guarantees that the resulting policy satisfies the constraints [11, 20]. These approaches focus on the constraint of the cumulative cost and require an initial policy satisfying the constraints. However, this cumulative cost criterion cannot model properly tasks where avoid catastrophic failures is crucial, e.g., car crashing.

Other approaches focus on the state dependant constraints, which should be fulfilled at every time step. To meet this requirement, safe exploration methods can be employed. Garcia, et al., proposed a method based on a risk function and a baseline agent, where the control action is sampled based on the evaluation of the risk [6]. The shielding [8] and backup policy [9] frameworks interfere with the control action to pull the system back to the safe states. These approaches require a manual defined safe policy. Berkenkamp, et al. [7], Wachi, et al. [21], Koller, et al. [22], and Hewing, et al.[23] proposed model-based approaches based on Lyapunov stability or model predictive control. These approaches start from an initial feasible policy and progressively increase the safe region based on the learned dynamics model. Recent methods also try to incorporate the model and the constraint information with the model-free RL algorithms. Dalal, et al., added a safe layer which analytically finds the closest action w.r.t the policy derived one [10]. Cheng, et al., proposed a barrier function method to guarantee safety during the exploration [12]. Finally, Urain et al., imposed constraints into the soft-Q function, samples coming from the soft-Q function are guaranteed to be safe [24].

Our approach considers the second group of constraints. However, different from other comparable methods, ATACOM does not require an initial policy, it can learn from scratch. In addition, our method does not require a backup policy either, as the constraint violations are forecasted and corrected at every time step. Furthermore, our method is not specifically restricted to any learning algorithms.

2 Learning on the Constraint Manifold

In this section, we discuss ATACOM in detail. We first introduce the mathematical notation used in this paper. Then, to demonstrate the main concept, we start with a simple scenario that the constraint on the subset of the state variable $\boldsymbol{q}$ and the action can be formulated as a function of the state velocity $\boldsymbol{a} = \boldsymbol{\mathcal{A}}(\dot{\boldsymbol{q}})$. Next, considering the continuity of velocity (sampling over the velocity does not ensure the continuity of the velocity), we convert the original state constraint to a viability constraint that incorporates the velocity, and the action is chosen as a function of the acceleration $\boldsymbol{a} = \boldsymbol{\mathcal{A}}(\ddot{\boldsymbol{q}})$. From a robotics point of view, this $\boldsymbol{a}$ can be the torque applied to each joint, and $\boldsymbol{\mathcal{A}}$ is the inverse dynamics model. Then, to cope with the velocity limit, we add the viability condition to the acceleration. Lastly, we discuss some practical issues such as the error correction, the tangent space convention that determines the null space bases.

Definitions We consider the CMDP with continuous state-action space. A CMDP is a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma, \mathcal{C})$, where $\mathcal{S}$ is a state space, $\mathcal{A}$ is an action space, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$is a transition kernel, γ is a discount factor, and $\mathcal{C} : \{c_i : \mathcal{S} \to \mathbb{R} | i \in 1, ..., k\}$ is a set of *immediate state-constraint* functions.

In this paper, we assume that the state variable $\boldsymbol{s} \in \mathcal{S}$ can be decomposed into the directly controllable state $\boldsymbol{q} \in \mathcal{Q}$ and uncontrollable state $\boldsymbol{x} \in \mathcal{X}$, i.e., $\boldsymbol{s} = [\boldsymbol{q}\ \boldsymbol{x}]^\mathsf{T}$. Also, the constraints $\boldsymbol{c}(\boldsymbol{q}) \leq \boldsymbol{0}$ are known and depend purely on the controllable state. We choose the action $\boldsymbol{a}$ to be a function of i-th order time derivative of the controllable state, i.e., $\boldsymbol{a} = \mathcal{A}(\boldsymbol{q}^{(i)}), i \in \{1, 2, ...\}$. The general form of the constrained reinforcement learning problem can be formulated as

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{s}_t, \boldsymbol{a}_t} \left[\sum_{t=0}^{T} \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t) \right], \qquad \text{s.t.} \quad \boldsymbol{c}(\boldsymbol{q}_t) \leq \boldsymbol{0}.$$

As an example, in the robot grasping task, the controllable state is the position of each joint, the uncontrollable state is the pose of the object, the action can be joint positions/velocities/accelerations/torques, and the constraints are the feasible working area of the robot end-effector.

2.1 State Constraints

The state constraints are defined as

$$\boldsymbol{f}(\boldsymbol{q}) = \boldsymbol{0}, \quad \boldsymbol{g}(\boldsymbol{q}) \leq \boldsymbol{0}, \tag{1}$$

where $\boldsymbol{f} : \mathbb{R}^Q \to \mathbb{R}^F, \boldsymbol{g} : \mathbb{R}^Q \to \mathbb{R}^G$ are two C^2 mappings for F equality and G inequality constraints, and $F < Q$. We add the slack variables $\boldsymbol{\mu} \in \mathbb{R}^G$ in inequality constraints, to convert the original constraints (1) into equality constraints

$$\boldsymbol{c}(\boldsymbol{q}, \boldsymbol{\mu}) = \begin{bmatrix} \boldsymbol{f}(\boldsymbol{q}) & \boldsymbol{g}(\boldsymbol{q}) + \frac{1}{2}\boldsymbol{\mu}^2 \end{bmatrix}^T = \boldsymbol{0}. \tag{2}$$

The constraint set (2) is a $(F + G)$ dimensional manifold embedded in $(Q + G)$ dimensional space. We calculate the time derivative of $\boldsymbol{c}(\boldsymbol{q}, \boldsymbol{\mu})$

$$\dot{\boldsymbol{c}}(\boldsymbol{q}, \boldsymbol{\mu}, \dot{\boldsymbol{q}}, \dot{\boldsymbol{\mu}}) = \begin{bmatrix} \boldsymbol{J}_f(\boldsymbol{q}) & \boldsymbol{0} \\ \boldsymbol{J}_g(\boldsymbol{q}) & \operatorname{diag}(\boldsymbol{\mu}) \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{q}} \\ \dot{\boldsymbol{\mu}} \end{bmatrix} = \boldsymbol{J}_c(\boldsymbol{q}, \boldsymbol{\mu}) \begin{bmatrix} \dot{\boldsymbol{q}} \\ \dot{\boldsymbol{\mu}} \end{bmatrix}, \tag{3}$$

with the Jacobians $\boldsymbol{J}_f \in \mathbb{R}^{F \times Q}$ and $\boldsymbol{J}_g \in \mathbb{R}^{G \times Q}$ of $\boldsymbol{f}(\boldsymbol{q})$ and $\boldsymbol{g}(\boldsymbol{q})$, respectively. Both Jacobians are combined into the Jacobian Matrix $\boldsymbol{J}_c(\boldsymbol{q}, \boldsymbol{\mu})$ of the complete constraint set.

We can find the null space matrix $\boldsymbol{N}_c(\boldsymbol{q}, \boldsymbol{\mu}) = \operatorname{Null}[\boldsymbol{J}_c(\boldsymbol{q}, \boldsymbol{\mu})] \in \mathbb{R}^{(F+G) \times (Q-F)}$ via SVD [25] or QR [26] decomposition. Each column of the orthogonal matrix $\boldsymbol{N}_c(\boldsymbol{q}, \boldsymbol{\mu})$ represents a basis vector

of the null space of $\boldsymbol{J}_c$. This null space bases can also be viewed as the tangent space bases of the constraint manifold. We can construct a tangent space velocity of the constraint manifold by a coordinate $\boldsymbol{\alpha}$ as

$$\begin{bmatrix} \dot{\boldsymbol{q}}_{\mathcal{T}} \\ \dot{\boldsymbol{\mu}}_{\mathcal{T}} \end{bmatrix} = \boldsymbol{N}_c(\boldsymbol{q}, \boldsymbol{\mu})\boldsymbol{\alpha}, \tag{4}$$

Substituting $[\dot{\boldsymbol{q}}\ \dot{\boldsymbol{\mu}}]^\mathsf{T}$ of (3) by $[\dot{\boldsymbol{q}}_{\mathcal{T}}\ \dot{\boldsymbol{\mu}}_{\mathcal{T}}]^\mathsf{T}$ in (4) , we have the constraint velocity

$$\dot{\boldsymbol{c}}(\boldsymbol{q}, \boldsymbol{\mu}, \dot{\boldsymbol{q}}, \dot{\boldsymbol{\mu}}) = \boldsymbol{J}_c(\boldsymbol{q}, \boldsymbol{\mu})\boldsymbol{N}_c(\boldsymbol{q}, \boldsymbol{\mu})\boldsymbol{\alpha} = \boldsymbol{0}. \tag{5}$$

Equation (5) implies that the constraints does not change regardless the choice of $\boldsymbol{\alpha}$. Based on this concept, the ATACOM method can be summarized as followings: Starting from a feasible point $(\boldsymbol{q}(0), \boldsymbol{\mu}(0)) \in \{(\boldsymbol{q}, \boldsymbol{\mu}) | \boldsymbol{c}(\boldsymbol{q}, \boldsymbol{\mu}) = \boldsymbol{0}\}$, we choose the tangent space velocity $[\dot{\boldsymbol{q}}_{\mathcal{T}}(t), \dot{\boldsymbol{\mu}}_{\mathcal{T}}(t)]^\mathsf{T} = \boldsymbol{N}_c(\boldsymbol{q}(t), \boldsymbol{\mu}(t))\boldsymbol{\alpha}(t)$, and the corresponding action as $\boldsymbol{a}(t) = \boldsymbol{\mathcal{A}}(\dot{\boldsymbol{q}}_{\mathcal{T}}(t))$. The constrained RL problem can be convert to an unconstrained RL problem. The the result trajectory $\boldsymbol{q}(t)$ satisfies the constraints $\boldsymbol{c}(\boldsymbol{q}(t), \boldsymbol{\mu}(t)) = \boldsymbol{0}$.

2.2 Viability Constraints

For a physical system, it is often required a continuous velocity command. However, directly sampling velocities $\dot{\boldsymbol{q}}$ does not ensure this continuity. A simple solution is to sample accelerations, apply force to the system or determine the velocity via integration. Furthermore, when considering inequality constraints, it is also desirable that $\dot{\boldsymbol{g}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \leq \boldsymbol{0}$ when $\boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{0}$ to avoid overshooting. We convert the original state constraints (1) to *viability constraints* inspired by the linear viability condition [27]

$$\begin{aligned} \boldsymbol{f}(\boldsymbol{q}) + \boldsymbol{K}_f \dot{\boldsymbol{f}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) =& \boldsymbol{f}(\boldsymbol{q}) + \boldsymbol{K}_f \boldsymbol{J}_f(\boldsymbol{q})\dot{\boldsymbol{q}} = \boldsymbol{0}, \\ \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{K}_g \dot{\boldsymbol{g}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) =& \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{K}_g \boldsymbol{J}_g(\boldsymbol{q})\dot{\boldsymbol{q}} \leq \boldsymbol{0}, \end{aligned} \tag{6}$$

with diagonal matrices $\boldsymbol{K}_f \in \mathbb{R}^{F \times F}, \boldsymbol{K}_g \in \mathbb{R}^{G \times G}$ having all positive entries. The matrices $\boldsymbol{K}_f$ and $\boldsymbol{K}_g$ determine the maximum velocities of the constraints $\dot{\boldsymbol{f}}$ and $\dot{\boldsymbol{g}}$ w.r.t to the value of the constraints. The viability constraint of the inequality constraint is illustrated in Figure 2. When $g(q) < 0$, the upper bound of the constraint velocity is $\dot{g}_{\max} > 0$ which means that it's still possible to get close to the constraint boundary. However, if $g(q) > 0$, the upper bound of constraint velocity $\dot{g}_{\max}$ should be smaller than zero to pull the violations back.

Analogous to the derivations from equation (2) and (3), we have

$$\boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\mu}) = \begin{bmatrix} \boldsymbol{f}(\boldsymbol{q}) + \boldsymbol{K}_f \boldsymbol{J}_f(\boldsymbol{q})\dot{\boldsymbol{q}} \\ \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{K}_g \boldsymbol{J}_g(\boldsymbol{q})\dot{\boldsymbol{q}} + \frac{1}{2}\boldsymbol{\mu}^2 \end{bmatrix} = \boldsymbol{0}, \tag{7}$$

and

$$\dot{\boldsymbol{c}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{\mu}, \dot{\boldsymbol{\mu}}) = \underbrace{\begin{bmatrix} \boldsymbol{K}_f \boldsymbol{J}_f(\boldsymbol{q}) & \boldsymbol{0} \\ \boldsymbol{K}_g \boldsymbol{J}_g(\boldsymbol{q}) & \operatorname{diag}(\boldsymbol{\mu}) \end{bmatrix}}_{\boldsymbol{J}_c(\boldsymbol{q}, \boldsymbol{\mu})} \begin{bmatrix} \ddot{\boldsymbol{q}} \\ \dot{\boldsymbol{\mu}} \end{bmatrix} + \underbrace{\begin{bmatrix} \boldsymbol{J}_f(\boldsymbol{q})\dot{\boldsymbol{q}} + \boldsymbol{K}_f \boldsymbol{b}_f(\boldsymbol{q}, \dot{\boldsymbol{q}}) \\ \boldsymbol{J}_g(\boldsymbol{q})\dot{\boldsymbol{q}} + \boldsymbol{K}_g \boldsymbol{b}_g(\boldsymbol{q}, \dot{\boldsymbol{q}}) \end{bmatrix}}_{\boldsymbol{\psi}(\boldsymbol{q}, \dot{\boldsymbol{q}})} = \boldsymbol{0}, \tag{8}$$

where $\boldsymbol{b}_f(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \dot{\boldsymbol{q}}^\mathsf{T} \boldsymbol{H}_f(\boldsymbol{q})\dot{\boldsymbol{q}}$, $\boldsymbol{b}_g(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \dot{\boldsymbol{q}}^\mathsf{T} \boldsymbol{H}_g(\boldsymbol{q})\dot{\boldsymbol{q}}$ and $\boldsymbol{H}_f \in \mathbb{R}^{F \times Q \times Q}, \boldsymbol{H}_g(\boldsymbol{q}) \in \mathbb{R}^{G \times Q \times Q}$ are Hessians of $\boldsymbol{f}(\boldsymbol{q}), \boldsymbol{g}(\boldsymbol{q})$, respectively. We can construct the joint acceleration as

$$\begin{bmatrix} \ddot{\boldsymbol{q}} \\ \dot{\boldsymbol{\mu}} \end{bmatrix} = -\boldsymbol{J}_c^\dagger(\boldsymbol{q}, \boldsymbol{\mu})\boldsymbol{\psi}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{N}_c(\boldsymbol{q}, \boldsymbol{\mu})\boldsymbol{\alpha}, \tag{9}$$

with the pseudo-inverse $\boldsymbol{J}_c^\dagger(\boldsymbol{q}, \boldsymbol{\mu})$ and the null space matrix $\boldsymbol{N}_c(\boldsymbol{q}, \boldsymbol{\mu})$ of the Jacobian $\boldsymbol{J}_c(\boldsymbol{q}, \boldsymbol{\mu})$, respectively. The first term in equation (9) is the necessary acceleration that maintain the curvature of the constraints manifold (7) and the second term is the tangent space acceleration of the constraints. When the starting from the point $[\boldsymbol{q}(0), \dot{\boldsymbol{q}}(0), \boldsymbol{\mu}(0)] \in \{(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\mu}) | \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\mu}) = \boldsymbol{0}\}$, and sampling over the $\boldsymbol{\alpha}$, we have the joint acceleration $\ddot{\boldsymbol{q}}$ and the corresponding action $\boldsymbol{a}$ satisfying the constraints.

2.3 Viability Acceleration Bound

In robotic, as well as other mechanical systems, it's important to consider the velocity constraints of the actuator. Also, the acceleration should be bounded properly to avoid overshooting. We again use the concept of viability to determine the upper and lower bound of the acceleration

$$\begin{aligned} \boldsymbol{a}_u &= \max\left(\min\left(\boldsymbol{a}_{\max}, -\boldsymbol{K}_a(\boldsymbol{q} - \boldsymbol{v}_{\max})\right), \boldsymbol{a}_{\min}\right), \\ \boldsymbol{a}_l &= \min\left(\max\left(\boldsymbol{a}_{\min}, -\boldsymbol{K}_a(\boldsymbol{q} - \boldsymbol{v}_{\min})\right), \boldsymbol{a}_{\max}\right), \end{aligned}$$

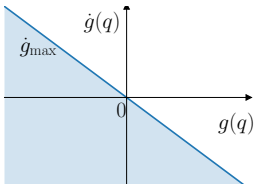


Figure 2: Viability Constraints

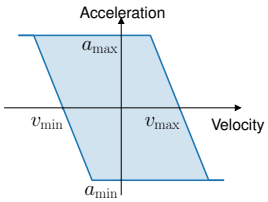


Figure 3: Feasible Acceleration Region

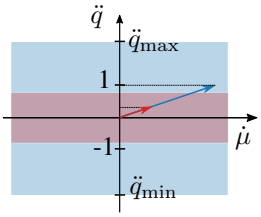


Figure 4: Tangent space bases

with the minimum and the maximum joint velocity limits $\boldsymbol{v}_{\min,\max}$ and the acceleration limits $\boldsymbol{a}_{\min,\max}$, $K_a > 0$ is a constant. The feasible acceleration region is illustrated in Figure 3. Analogous to the viability constraints, the feasible region of the acceleration is modified depending on the state of joint velocities. This technique effectively prevents overshooting.

2.4 Error Correction

For time-continuous systems, the state is obtained at a certain sampling rate and the action is applied for a certain period. This time discretization results in constraint violations at each time step. Therefore, we add an error correction term. We construct a P-controller with a diagonal matrix $\boldsymbol{K}_c$ for the constraints

$$\begin{bmatrix} \ddot{\boldsymbol{q}}_E \\ \dot{\boldsymbol{\mu}}_E \end{bmatrix} = -\boldsymbol{J}_c^{\dagger} \boldsymbol{K}_c \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\mu}). \tag{10}$$

Combining (9) with (10), we get the joint acceleration applied to the system

$$\begin{bmatrix} \ddot{\boldsymbol{q}} \\ \dot{\boldsymbol{\mu}} \end{bmatrix} = -\boldsymbol{J}_c^{\dagger}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\mu}) \left[\boldsymbol{K}_c \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\mu}) + \boldsymbol{\psi}(\boldsymbol{q}, \dot{\boldsymbol{q}})\right] + \boldsymbol{N}_c(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\mu})\boldsymbol{\alpha}. \tag{11}$$

The first term on the RHS is the necessary acceleration to maintain the constraint and the second term on the RHS is the tangent acceleration that can be explored freely.

2.5 Null Space Convention

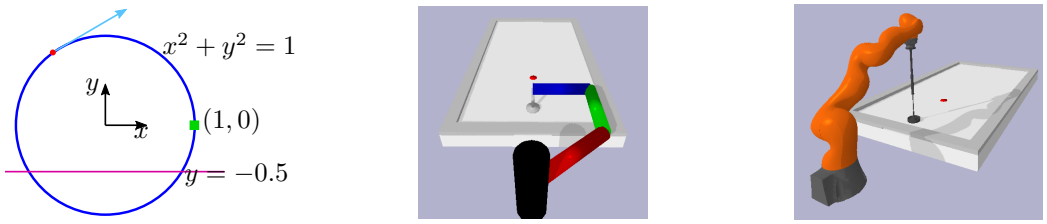
The orthogonal null space matrix $\boldsymbol{N}_c$ can be determined through SVD or QR decomposition. However, the representation of the null space bases is not unique. We can easily get a new set of bases by multiplying an orthogonal matrix with the original null space matrix. It is difficult to preserve the consistency of the null space bases computed by the numerical decomposition method. To solve this issue, we propose a convention to ensure the uniqueness of the null space bases.

Each column of the null space matrix $\boldsymbol{N}_c$ is a unit vector indicating a direction of $[\ddot{\boldsymbol{q}}\ \dot{\boldsymbol{\mu}}]^{\mathsf{T}}$. However, this unit vector could sometimes contribute majorly to the part of the slack variable and the entries for the joint accelerations could be very small. As a result, the joint acceleration obtained from

Algorithm 1: ATACOM

Input: Constraint: $\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{A}_f, \boldsymbol{A}_g, \boldsymbol{b}_f, \boldsymbol{b}_g$. Scale parameter: $\boldsymbol{K}_c, \boldsymbol{K}_f, \boldsymbol{K}_g$. Time step ΔT.

for *each episode* **do**
  Initial feasible state $\boldsymbol{s}_0$, slack variable $\boldsymbol{\mu}_0$.
  for *each time step* k **do**
    Sample policy action $\boldsymbol{\alpha}_k \sim \pi(\cdot|\boldsymbol{s}_k)$.
    Observe the $\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k$ from $\boldsymbol{s}_k$.
    Compute $\boldsymbol{J}_{c,k} = \boldsymbol{J}_c(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \boldsymbol{\mu}_k)$, $\boldsymbol{\psi}_k = \boldsymbol{\psi}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k)$, $\boldsymbol{c}_k = \boldsymbol{c}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \boldsymbol{\mu}_k)$.
    Compute the RCEF of tangent space basis of $\boldsymbol{N}_c^R$
    Compute the tangent space acceleration $[\ddot{\boldsymbol{q}}_k\ \dot{\boldsymbol{\mu}}_k]^{\mathsf{T}} = -\boldsymbol{J}_{c,k}^{\dagger}\left[\boldsymbol{K}_c \boldsymbol{c}_k + \boldsymbol{\psi}_k\right] + \boldsymbol{N}_c^R \boldsymbol{\alpha}_k$
    Integrate the slack variable $\boldsymbol{\mu}_{k+1} = \boldsymbol{\mu}_k + \dot{\boldsymbol{\mu}}_k \Delta T$
    Apply the control action $\boldsymbol{a}_k = \mathcal{A}(\ddot{\boldsymbol{q}}_k)$ to the environment.
    Observe the next state $\boldsymbol{s}_{k+1}$ and reward r_k from the environment.
    Provide the transition tuple $(\boldsymbol{s}_k, \boldsymbol{\alpha}_k, \boldsymbol{s}_{k+1}, r_k)$ to the RL algorithm



(a) *CircularMotion* (b) *PlanarAirHockey* (c) *IiwaAirHockey*

Figure 5: Experiment Environments

$\boldsymbol{\alpha} \in [\boldsymbol{\alpha}_{\min}, \boldsymbol{\alpha}_{max}]$ can only cover a small region of the acceleration. As illustrated in Figure 4, the red arrow is a unit basis vector of the tangent space, the reachable joint acceleration by a universal scaling factor could only cover a small range of the feasible joint acceleration, as the red area shown in Figure 4.

To alleviate the previously mentioned issue, we compute the Reduced Column Echlon Form (RCEF) of the null space matrix $\boldsymbol{N}_c^R = \text{RCEF}(\boldsymbol{N}_c)$. Given that the RCEF of a matrix is unique, we obtain the unique bases of the null space. In addition, for RCEF, each row containing a leading 1 has zeros in all its other entries. Generally speaking, there exist N independent joints whose acceleration can be solely determined by $\boldsymbol{\alpha}$, where N is the dimensions of the null space. Also, we can define the feasible range of $\boldsymbol{\alpha}$ as $\alpha_i \in [\ddot{q}_{i,\min}, \ddot{q}_{i,\max}]$. Through this convention and the feasible range of $\boldsymbol{\alpha}$, the joint acceleration is able to cover the full feasible range as illustrated in Figure 4, the blue vector is the basis of RCEF, and the blue area is the feasible range of the joint acceleration.

3 Experiments and Evaluation

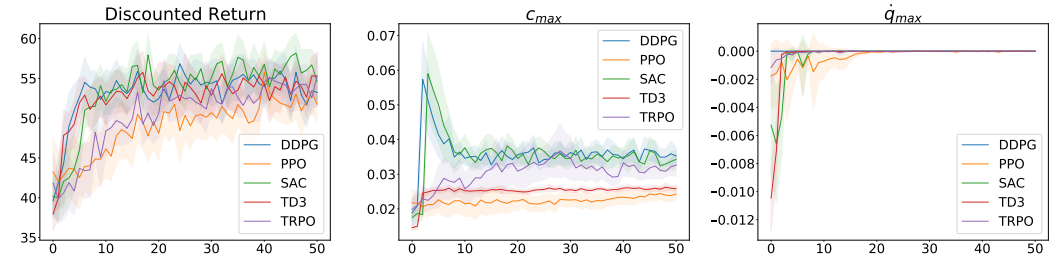
To illustrate the properties of our approach, we demonstrate three different experiments in this section. We first demonstrate a toy task, *CircularMotion*, shown in Figure 5a. In this task, we consider state equality, inequality, and velocity constraints. Secondly, we show a robotic environment with only inequality constraints, *PlanarAirHockey* shown in Figure 5b, a 3 DoF planar robot playing the hitting task in the air hockey scenario while keeping the end-effector inside the table boundary and robot's joint positions and velocities within its limits. Finally, we demonstrate, *IiwaAirHockey* in Figure 5c, a 7 DoF KUKA IIWA robot learning the hitting task in the simulator. In addition to the constraints of the 3-dimensional task, an equality constraint is added to ensure that the robot end-effector stays on the table surface. More details of the experimental setups can be found in the appendix.

CircularMotion. In this task, shown in Figure 5a, the red point tries to move along a unit circle in 2D space while keeping the velocity of each direction below the velocity limits and maintaining the position above a certain height. The objective is to reach the target point (green square) located in $(1, 0)$. The control action is the acceleration $\boldsymbol{a} = [\ddot{x}\ \ddot{y}]^\mathsf{T}$.

We compare ATACOM with two other approaches for the task. (i) *TerminatedCircularMotion* where the episode terminates when the maximum constraint violations up to a threshold. (ii) *ErrorCorrectionCircularMotion*, where the error correction term in (10) is added before the action is applied to the environment. We test five model-free RL algorithms (SAC, DDPG, TD3, TRPO, and PPO implemented in *Mushroom-RL* [28]) for each approach.

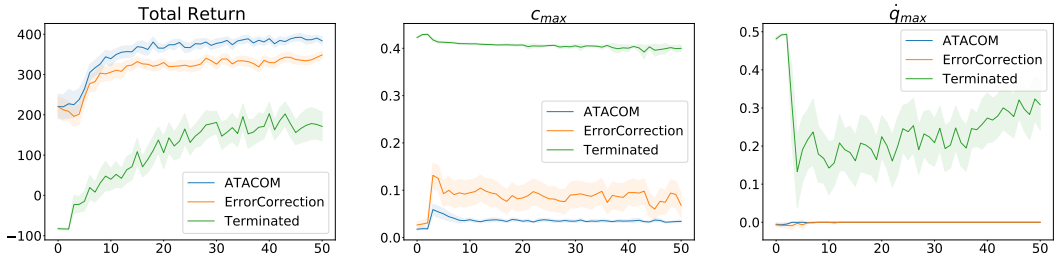
Figure 6 shows the learning curve and constraint violations of all test RL algorithms for ATACOM. Every algorithm is able to improve the learning performance and SAC outperforms the algorithm, which matches our expectation. Figure 6b and 6c shows the maximum constraint function and maximum joint velocity constraints at each time step. It can be shown that the maximum constraint violations during the whole learning process remain small. And the velocity limit violations are zero after 30 epochs which means the learned policies try to move on the margin of the constraints.

In Figure 7 we compare ATACOM, *TerminatedCircularMotion* and *ErrorCorrectionCircularMotion*. We select the best learning algorithms for each approach (which is SAC for all cases). Compared to the *ErrorCorrection* and *TerminatedCircularMotion*, our method focus on a lower dimensional ex-



(a) Discounted Reward (b) Max Constraint Violations (c) Max Velocity Limits Violations

Figure 6: ATACOM for the *CircularMotion*. 6a shows the discounted return at each epochs. 6b and 6c shows the maximum constraint violations and maximum joint velocity limits violations.



(a) Total Return (b) Max Constraint Violations (c) Max Velocity Limits Violations

Figure 7: Comparison between ATACOM, *TerminatedCircle*, *ErrorCorrectionCircle*.

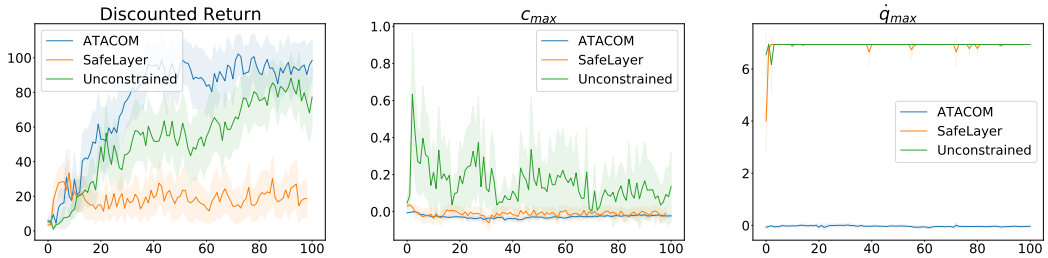
ploration space which avoid the constraint violations while the others do not. We can conclude from the plot that ATACOM has not only lower constraint violations but also better learning performance among the three approaches.

PlanarAirHockey. In this experiment, we apply a 3-joints planar robot for the air-hockey hitting task, as illustrated in Figure 5b. The end-effector of the robot is kept on the table surface and the objective is to hit the puck to the opponent's goal. In this environment, only inequality constraints are considered, i.e., the robot end-effector should stay inside the table's region, and the joint positions and velocities should not exceed its limits. The control action is the torque applied on each joint.

In this task, we compared ATACOM with the *SafeLayer* method proposed by Dalal et al.,[10] and the *Unconstrained* air-hockey environments. Since the *SafeLayer* method at the beginning requires a free exploration process to learn the constraint function, we only compare the learning performance and the constraint violations after this process. For the unconstrained environment, the robot is completely free to explore, and the episodes only terminate when the maximum episode step is reached. In this experiment, we only compare the best DDPG result after the parameter sweep, as the available implementation of *SafeLayer* only supports DDPG.

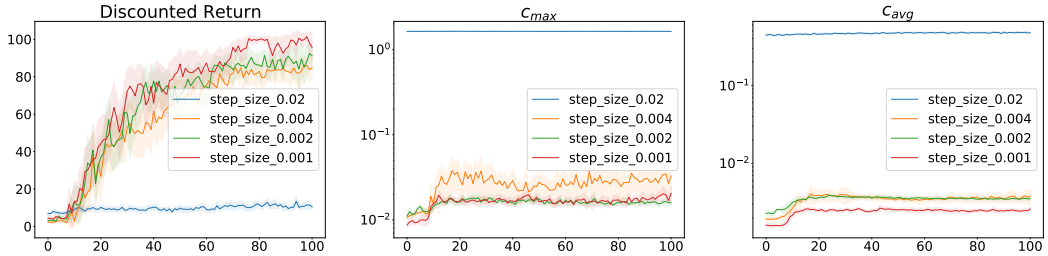
The result is shown in Figure 8. We can see that ATACOM have the best learning performance and the minimum constraint violations among the three methods. *SafeLayer* did not learn the constraint function of joint velocities properly. Furthermore, the learned constraints appear to be too restrictive to learn a good policy. Compared to the method of *Unconstrained* approach, although ATACOM has the same dimension as the *Unconstrained*, ATACOM explores only in the feasible region while the *Unconstrained* approach explores the whole state-action space. This consideration explains why ATACOM outperforms the baselines in terms of learning performances.

IiwaAirHockey. In the third experiment, we demonstrate the same air-hockey hitting task with a KUKA IIWA-LBR14 Robot in the Pybullet simulator. In this task, we add an equality constraint to ensure the end-effector stays on the table surface. We also add inequality constraints of the end-effector and joint positions as mention in the *PlanarAirHockey* task. As the real-world's KUKA controller enforces the joint velocity limits by default, we enforce the joint velocity constraints in the simulation. We compare the impact on the choice of the simulation step size. The step size refers



(a) Discounted Return (b) Max Constraint Violations (c) Max Velocity Limits Violations

Figure 8: Comparison between ATACOM, *SafeLayer* [10], *Unconstrained* AirHockey in DDPG.



(a) Discounted Return (b) Max Constraint Violations (c) Average Constraint Violations

Figure 9: Comparison of different step size of simulation in *IiwaAirHockey*.

to the sampling frequency in the real world. At each simulation step, the torque is computed by the previous agent's action until the new control action is received. The error correction term is added at each time step. We choose the simulation step size as 0.02s, 0.004s, 0.002s, and 0.001s and keep agent control frequency to be 50Hz.

From the result shown in Figure 9, for a sufficiently small step size, as 0.004s, 0.002s, 0.001s, the learning agent is able to improve the hitting policy. When the step size is too big, e.g., 0.02s, the error correction term dominates the control action and the agent is not able to learn a good policy. From the result in this experiment, we demonstrate that ATACOM is able to work for the high dimensional task. The simulation result also provides us the guidance for the real-world application: the higher frequency of sampling and error correction, the smaller constraint violations will occur. However, the high frequency and error correction also requires larger bandwidth and heavier computation load.

4 Conclusion

In this article, we present ATACOM, a safe exploration method for Constrained RL based on the knowledge of the model and the mathematical formulations of constraints. ATACOM explores the tangent space of the constraint manifold. This exploration technique enables us to utilize any type of model-free RL method while maintaining the constraint violations below a small threshold. From the experiments, we have shown that ATACOM not only have small constraints violations but also better learning performance w.r.t. the other baselines. These performance gains occur because ATACOM only focus on the safe region (from inequality) and subspace (from equality constraint) of the whole state-action space.

However, our method still has some limitations. We assume the perfect knowledge of the system model and constraints. This assumption does not hold in most real-world applications, since model errors, disturbances, sensor noise can not be neglected. The model mismatch could potentially cause unexpected constraint violations. To deploy this method in real-world robots, future works will focus on the model mismatch problem.

References

[1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, 529(7587):484–489, 2016.

[2] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al. Grandmaster Level in StarCraft II Using Multi-Agent Reinforcement Learning. *Nature*, 575(7782):350–354, 2019.

[3] Y. Duan, X. Chen, C. X. B. Edu, J. Schulman, P. Abbeel, and P. B. Edu. Benchmarking Deep Reinforcement Learning for Continuous Control. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2016. URL https://github.com/.

[4] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous Control with Deep Reinforcement Learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016. URL https://goo.gl/J4PIAz.

[5] J. García and F. Fernández. A Comprehensive Survey on Safe Reinforcement Learning. *Journal of Machine Learning Research*, 16:1437–1480, 2015.

[6] J. Garcia and F. Fernandez. Safe Exploration of State and Action Spaces in Reinforcement Learning. *Journal of Artificial Intelligence Research*, 45:515–564, dec 2012. ISSN 10769757.

[7] F. Berkenkamp, M. Turchetta, A. P. Schoellig, and A. Krause. Safe Model-based Reinforcement Learning with Stability Guarantees. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*, 2017.

[8] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu. Safe Reinforcement Learning via Shielding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[9] A. Hans, D. Schneegaß, A. M. Schäfer, and S. Udluft. Safe Exploration for Reinforcement Learning. In *ESANN*, 2008. ISBN 2930307080.

[10] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, and Y. Tassa. Safe Exploration in Continuous Action Spaces. *arXiv preprint arXiv:1801.08757*, 2018.

[11] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh. A Lyapunov-based Approach to Safe Reinforcement Learning. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*, 2018.

[12] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick. End-to-End Safe Reinforcement Learning through Barrier Functions for Safety-Critical Continuous Control Tasks. In *33rd AAAI Conference on Artificial Intelligence*, volume 33, pages 3387–3395. AAAI Press, jul 2019. ISBN 9781577358091. doi:10.1609/aaai.v33i01.33013387.

[13] E. Altman. Constrained Markov Decision Processes with Total Cost Criteria: Lagrangian Approach and Dual Linear Program. *Mathematical methods of operations research*, 48(3): 387–417, 1998.

[14] C. Tessler, D. J. Mankowitz, and S. Mannor. Reward Constrained Policy Optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[15] Y. Liu, J. Ding, and X. Liu. IPO: Interior-point Policy Optimization under Constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[16] A. Stooke, J. Achiam, and P. Abbeel. Responsive Safety in Reinforcement Learning by PID Lagrangian Methods. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

[17] D. Ding, X. Wei, Z. Yang, Z. Wang, and M. R. Jovanovi´cjovanovi´c. Provably Ecient Safe Exploration via Primal-Dual Policy Optimization. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 130, 2021.

[18] J. Achiam, D. Held, A. Tamar, and P. Abbeel. Constrained Policy Optimization. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.

[19] H. B. Ammar, R. Tutunov, and E. Eaton. Safe policy search for lifelong reinforcement learning with sublinear regret. In *International Conference on Machine Learning*, pages 2361–2369. PMLR, 2015.

[20] Y. Chow, O. Nachum, A. Faust, E. Duenez-Guzman, and M. Ghavamzadeh. Lyapunov-based Safe Policy Optimization for Continuous Control. In *Reinforcement Learning for Real Life (RL4RealLife) Workshop in the 36 th International Conference on Machine Learning*, 2019.

[21] A. Wachi, Y. Sui, Y. Yue, and M. Ono. Safe Exploration and Optimization of Constrained MDPs Using Gaussian Processes. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2018. URL www.aaai.org.

[22] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause. Learning-Based Model Predictive Control for Safe Exploration. In *Proceedings of the IEEE Conference on Decision and Control*, 2018.

[23] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger. Learning-Based Model Predictive Control: Toward Safe Learning in Control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1):269–296, 2020. ISSN 2573-5144. doi:10.1146/annurev-control-090419-075625.

[24] J. Urain, A. Li, P. Liu, C. D'Eramo, and J. Peters. Composable energy policies for reactive motion generation and reinforcement learning. In *Robotics: Science and Systems*, 2021.

[25] R. P. Singh and P. W. Likins. Singular Value Decomposition for Constrained Dynamical Systems. *Journal of Applied Mechanics, Transactions ASME*, 52(4):943–948, 1985. doi:10.1115/1.3169173.

[26] S. S. Kim and M. J. Vanderploeg. QR Decomposition for State Space Representation of Constrained Mechanical Dynamic Systems. *Journal of Mechanisms, Transmissions, and Automation in Design*, 108:183–188, 1986.

[27] M. Faroni, M. Beschi, N. Pedrocchi, and A. Visioli. Viability and Feasibility of Constrained Kinematic Control of Manipulators. *Robotics*, 7(3), jul 2018. ISSN 22186581. doi:10.3390/robotics7030041.

[28] C. D'Eramo, D. Tateo, A. Bonarini, M. Restelli, and J. Peters. Mushroomrl: Simplifying reinforcement learning research. https://github.com/MushroomRL/mushroom-rl, 2020.