MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Dynamics-Enabled Safe Deep Reinforcement Learning: Case Study on Active Suspension Control

Li, Z.; Chu, T.; Kalabic, U.

TR2019-082 August 24, 2019

Abstract

Reinforcement learning (RL) is in essence a trialand-error process which involves exploratory actions. These explorations can lead to system constraint violations and physical system damages, impeding RL's use in many realworld engineered systems. In this paper, we develop a safe RL framework that integrates model-free learning with modelbased safety supervision to bridge the gap. We exploit the underlying system dynamics and safety-related constraints to construct a safety set using recursive feasibility techniques. We then integrate the safety set in RL's exploration to guarantee safety while simultaneously preserving exploration efficiency by using the hit-and-run sampling. We design a novel efforts-toremain-safe penalty to effectively guide RL to learn system constraints. We apply the proposed safe RL framework to the active suspension system in which actuation and state constraints are present due to ride comfort, road handling, and actuation limits. We show that the developed safe RL is able to learn a safe control policy safely while outperforming a nominal controller.

IEEE Conference on Control Technology and Applications (CCTA)

Dynamics-Enabled Safe Deep Reinforcement Learning: Case Study on Active Suspension Control

Zhaojian Li, Tianshu Chu, Uroš Kalabić

***Abstract*— Reinforcement learning (RL) is in essence a trial-and-error process which involves exploratory actions. These explorations can lead to system constraint violations and physical system damages, impeding RL's use in many real-world engineered systems. In this paper, we develop a safe RL framework that integrates model-free learning with model-based safety supervision to bridge the gap. We exploit the underlying system dynamics and safety-related constraints to construct a safety set using recursive feasibility techniques. We then integrate the safety set in RL's exploration to guarantee safety while simultaneously preserving exploration efficiency by using the hit-and-run sampling. We design a novel efforts-to-remain-safe penalty to effectively guide RL to learn system constraints. We apply the proposed safe RL framework to the active suspension system in which actuation and state constraints are present due to ride comfort, road handling, and actuation limits. We show that the developed safe RL is able to learn a safe control policy safely while outperforming a nominal controller.**

I. Introduction

The past decades have witnessed the great success of reinforcement learning (RL) in a broad spectrum of applications such as board games [1], [2], natural language processing [3], [4], economics [5], [6], among many others. However, its applications to real-world engineered systems are still rare. The primary reason is that RL is essentially a trial-and-error learning process that may lead to *system constraint violations* during training, which could cause disastrous consequences such as battery overheat, robot breakdown, and car crashes. Therefore, there is a critical need to advance RL with safety guarantees to fill the gap of RL's applications towards real-world engineered systems.

Several methodologies have been proposed to treat system constraints during RL training and deployment. Worst-case scenario based approaches have been developed to maximize worst-case return, mitigating the effects of variability and uncertainty [7], [8]. Risk management approaches have also been developed to control risks such as variance of the return [9] and the probability of entering into an error state [10]. A comprehensive literature review on the topic of model-free safe RL can be found in [11]. These methodologies can manage risks in RL to some extent. However, they provide *no guarantees in avoiding constraint violations*. Recently, a human intervention based RL has been proposed in [12], where a human operator supervises the training to ensure the safety during the training process. This may not be practical since RL generally involves extensive training episodes which require a large amount of supervision effort, in which humans are typically error-prone. A promising direction has emerged which exploits system dynamics to formulate a safety certification for learning supervision [13]–[17]. However, these methods are computationally very expensive for online implementation [13], [14]. Also, these methods offer limited exploration efficiency as certifications are integrated using preliminary strategies.

In this paper, we propose a dynamics-enabled safe RL framework that combines model-free RL with a model-based safety regulator for learning supervision. The safe RL we propose is a significant extension of the conventional RL algorithm (see Figure 1). Specifically, we propose to supervise conventional learning process by incorporating a state-dependent constraint-admissible safety set. We leverage recursive feasibility techniques and exploit the underlying physical dynamics and safety-related system constraints to obtain the safety set. We integrate the safety set in RL's exploration to guarantee safety while preserving the exploration efficiency. We develop a safety enforcement scheme using hit-and-run sampling that projects unsafe exploratory actions to the safety set with good exploration efficiency. We design a penalty function to characterize unsafe explorations and use it to accelerate learning by augmenting actual experiences with unsafe exploration penalties. The proposed safe RL framework is independent of and complimentary to existing RL algorithms. As compared to our preliminary work [18], the constructed safety set is less conservative. In contrast to studies in [13]–[17], the proposed framework requires less computational power and has better exploration efficiency.

We apply the safe RL approach to active suspension system which involves system constraints due to ride comfort, road handling, and actuation limits [19]. Those constraints are sometimes considered implicitly in the cost function such as in the linear quadratic regulator (LQR) design [20] and H_∞ design [21]. These control designs thus cannot avoid constraint violations. Model predictive control (MPC) approaches have also been developed to explicitly deal with the constraints [19], [22], [23]. However, MPC controllers need to solve online optimization problems at each step, requiring great computation for online implementation. In this paper, we exploit the proposed safe RL framework and show that no constraint violations happen during either

This work was supported by Mitsubishi Electric Research Laboratories.

Zhaojian Li is with the Department of Mechanical Engineering, Michigan State University, East Lansing, MI 48824, USA. Email: lizhaoj1@egr.msu.edu

Tianshu Chu is with the Department of Civil and Environmental Engineering, Stanford University, Stanford, CA, USA. Email: cts1988@stanford.edu

Uroš Kalabić is with Mitsubishi Electric Research Laboratories, Cambridge, MA, USA. Email: kalabic@merl.com

the training or validation process. We also show improved performance over a nominal controller.

The rest of the paper is organized as follows. In Section II, we present an overview of the proposed safe RL in comparison with conventional RL. In Section III, the constraint-admissible set is constructed for safety regulation. The complete safe RL algorithm is presented in Section IV with an application to active suspension system. Finally, conclusions and future work are discussed in Section V.

II. Safe RL with Safety Regulation

Conventional RL is a trial-and-error learning process that aims at optimizing the agent's policy to maximize accumulated rewards (or minimize accumulated costs) through its continuous interaction with the environment. Specifically, at each discrete time step t, the agent takes an observation $x(t) \in \mathbb{R}^n$, executes an action/control $u(t) \in \mathbb{R}^m$ and receives a scalar cost (or reward) $r(t) \in \mathbb{R}$. In general, the environment/system is typically impacted by some unknown disturbance $w(t)$. The entire history of observation and action is an information state, $s(t) = \big(x(0), u(0), \ldots, u(t-1), x(t)\big)$. We assume a Markovian environment and thus redefine information state as $s(t) = x(t)$. A policy, $\pi : \mathcal{X} \to \Pr(\mathcal{U})$, is a mapping from states to a probability distribution over the action space and it describes the agent's behavior. A RL agent learns a control policy from the experience $\{x(0), u(0), r(0), x(1), u(1), r(1), \cdots\}$ to maximize the overall future rewards defined by $R = \sum_{t=0}^{\infty} \gamma^t r(t)$, where γ is the discount factor. Many times the agent's policy is parameterized by some parameters θ, i.e., $\pi_\theta : \mathcal{X} \to \mathcal{U}$. The goal is to learn the values of parameters θ that maximize the reward function. Note that RL is a trial-and-error method, and *exploratory* actions have to be performed to explore the optimal policy. Therefore, disastrous consequences can happen due to violations of internal system constraints in real-world engineered systems during this exploration. We therefore propose a safe RL framework that can train the agent to *learn a safe policy safely*.

The proposed safe RL is a significant extension of the conventional RL algorithm. Specifically, as shown in Figure 1, we introduce a safety supervisory element between a standard RL agent and the environment to ensure safe and reliable system operations. If an exploratory action is not admissible according to the supervisor, it will be projected onto the safe set. At the same time, a bad-exploration penalty signal is passed to the agent, providing feedback to the RL agent to learn the system constraints. The proposed safety supervisor consists of four modules: a *safety set* that defines the state-dependent admissible actions, a *safety enforcement scheme* that projects unsafe exploratory actions to a safe one, and a *penalty function* that characterizes the unsafe exploration margin. We will describe these modules in the subsequent sections.

III. Safety Sets

In this section, we introduce the set of safe, admissible actions. The set is based on the stochastic analogue to the maximal output-admissible set [24], a preliminary version of which has been developed in [25]. To see how we construct the set, consider the closed-loop constrained linear system,

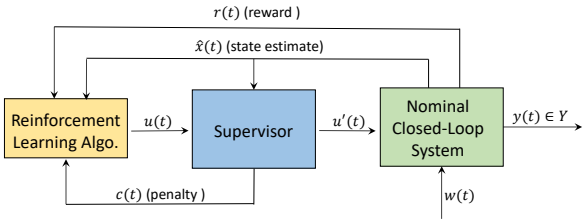

Fig. 1. Safe RL with model-enabled supervision.

$$x(t+1) = Ax(t) + Bu(t) + B_w w(t), \tag{1a}$$
$$y(t) = Cx(t) + Du(t) + D_w w(t) \in Y, \tag{1b}$$

where $x(t) \in \mathbb{R}^n$ is the state, $u(t) \in \mathbb{R}^m$ is the modifying control input, $w(t) \in \mathbb{R}^\eta$ is the disturbance input, and $y(t)$ is the output. The output-constraint set $Y \subset \mathbb{R}^p$ is a polytope containing 0. Since the system is closed-loop, we assume that A is asymptotically stable.

In the application considered in this work, the predicted disturbance for time k performed at time t, $w(k|t)$, can be decomposed according to,

$$w(k|t) = w_d(k|t) + w_s(k|t), \tag{2}$$

where $w_d(t)$ and $w_s(t)$ are referred to as the worst-case and stochastic components of the disturbance input, respectively. Henceforth, for a vector signal v, the notation $v(k|t)$ denotes the prediction of $v(t+k)$ performed at time t. The prediction of the worst-case component $w_d(k|t)$ is assumed to be contained in a set $\mathcal{W}_k$, which is compact and contains 0. The stochastic component is assumed to be i.i.d. Gaussian, zero-mean, with covariance matrix W. Mathematically, these properties can be expressed as,

$$w_d(k|t) \in \mathcal{W}_k,\ w_s(t) \sim \mathcal{N}(0, W),\ \forall t, k \in \mathbb{Z}_+, \tag{3}$$

satisfying $\mathbb{E}(w_s(t) w_s(k)^{\mathrm{T}}) = 0$ if $t \neq k$.

We are interested in the stochastic, maximal output-admissible set for the system (1). This is the set of all initial conditions $x(0)$ that, when the control is held constant at 0, *i.e.*, $u(t) = 0$, constraints will be satisfied with probability β, when subject to any possible predicted disturbance $\{w_d(k|t)\}$, and for all present and future time. Its definition is given by,

$$O_\infty^\beta = \{x : x(0|t) = x,\ u(k|t) = 0,\ \{y(k|t)\} \oplus \mathcal{P}_k^\beta \subset Y, \\ \forall w_d(k|t) \in \mathcal{W}_k,\ k \in \mathbb{Z}_+\}, \tag{4}$$

where the predicted terms are given by,

$$x(k+1|t) = Ax(k|t) + Bu(k|t) + B_w w_d(k|t), \tag{5a}$$
$$y(k|t) = Cx(k|t) + Du(k|t) + D_w w_d(k|t), \tag{5b}$$

and $\mathcal{P}_k^\beta$ is the uncertainty ellipsoid containing $y(t+k) - y(k|t)$ with probability β, *i.e.*, it satisfies,

$$\mathbb{P}(y(t+k) - y(k|t) \in \mathcal{P}_k^\beta) = \beta.$$

Note that the predicted values in (5) exclude the stochastic disturbance term.

The computation of O_∞^β is described in [24], [25]. Note that oftentimes O_∞^β cannot be determined in a finite number of steps; however, we can always compute an arbitrarily close approximation of O_∞^β in a finite number of steps, with the number of steps of the computation algorithm proportional to the desired precision. In previous work on maximal output admissible sets, worst-case and stochastic disturbances have been considered separately. In the case of linear systems, the combination of the two is straightforward because, according to the principle of linear superposition, the state can be expressed as a linear combination of the contributions from each component.

Note that the deterministic disturbance set $\mathcal{W}_k$ depends on k. Since the set O_∞^β is used in a receding-horizon fashion, this implies that the worst-case component of the disturbance sequence is reconsidered at every run of the prediction algorithm. From a theoretical perspective, this does not make sense because repeated application of the worst-case disturbance can only be handled robustly by holding the disturbance bound to be constant; however, from a practical perspective, this formulation is useful when considering jump disturbances. Importantly, in the application to suspension systems, we set the disturbance set to equal $\mathcal{W}_k = [-j_{\max}, j_{\max}]$ for $k = 0, 1, \ldots, N_j$ and $\mathcal{W}_k = \{0\}$ for $k > N_j$, where N_j is a design parameter. In practice, this allows us to be somewhat robust to jumps of magnitude $j_{\max}$ and width N_j as long as the jumps happen infrequently enough to allow the effect of previous jumps to dissipate.

In this work, we wish to compute the control effort required to return the state to the maximal set where the nominal closed-loop controller is likely feasible, *i.e.*, the maximal output-admissible set O_∞^β. To do this we optimize,

$$\sum_{k=0}^{N-1} |u(k|t)| \to \min, \tag{6}$$

subject to the prediction dynamics (5), the constraint $\{y(k|t)\} \oplus \mathcal{P}_k^\beta \subset Y$, $k = 0, 1, \ldots, N-1$, and the requirement that $x(N|t) \in O_\infty^\beta$, for some time $N \in \mathbb{Z}_+$, which is a design parameter. This amounts to solving a constrained optimization problem.

We set up the constrained optimization by determining a maximal output-admissible set, with dynamics that have been modified to treat the control inputs as a state. The system with extended dynamics is therefore given by,

$$X(k+1|t) = \tilde{A}X(k|t) + \tilde{B}_w w_d(k|t), \tag{7a}$$
$$y(k|t) = \tilde{C}X(k|t) + \tilde{D}_w w_d(k|t), \tag{7b}$$

where,

$$X(k|t)^{\mathrm{T}} = \Big[x(k|t)^{\mathrm{T}} \quad u(k|t)^{\mathrm{T}} \quad u(k+1|t)^{\mathrm{T}} \quad \cdots \quad u(k+N-1|t)^{\mathrm{T}}\Big],$$

and the system matrices are determined according to definitions given previously.

Since the system (7) is asymptotically stable, we are able to determine its stochastic maximal output-admissible set. It is given by,

$$P^\beta = \{X : X(0|t) = X,\ \{y(k|t)\} \oplus \mathcal{P}_k^\beta \subset Y, \quad \forall w_d(k|t) \in \mathcal{W}_k,\ k \in \mathbb{Z}_+\}. \tag{8}$$

As mentioned previously, details on computing the set P^β are given in [24], [25].

IV. Safe Deep RL

A. Augmented Rewards and Safe Sampling

In this section, we integrate the safety set obtained from Section III to regulate unsafe explorations. Specifically, at time step t under state $x(t)$, RL takes an exploratory action $a(t)$. The reward $r(t)$ corresponding to the action $a(t)$ takes the form,

$$r(t) = \begin{cases} -c(t) - \alpha \cdot E(t), & \text{if } a(t) \text{ is admissible,} \\ -G, & \text{otherwise,} \end{cases} \tag{9}$$

where $c(t) = c(x(t))$ is the objective cost in standard RL, $E(t) = E(x(t), a(t))$ is an additional cost measuring the amount of effort needed to keep the system within constraints, and α is a weight on the latter cost. The cost $G \gg c \geq 0$ is a large penalty assigned to an unsafe exploratory action.

To determine if $a(t)$ is admissible and to obtain $E(t)$, we solve the following optimization problem,

$$E(x(t), a(t)) = \min \|U\|_1 \tag{10a}$$
$$\text{subj. to } X(t) \in P^\beta, \tag{10b}$$

where $X^{\mathrm{T}}(t) = \begin{bmatrix} x^{\mathrm{T}}(t) & a^{\mathrm{T}}(t) & U^{\mathrm{T}} \end{bmatrix}$ and P^β is defined in (8). The optimization variable $U \in \mathbb{R}^{(N-1)\cdot m}$ is the vector of admissible future actions in $N-1$ steps. In our application, the norm in the optimization (10) corresponds to the vector 1-norm, but could be any norm, depending on the type of distance that we wish to minimize. Note that since $x(t)$ and $a(t)$ are known, the problem (10) can be posed as a linear programming (LP) problem.

Since P^β is the set of all admissible control sequences, when a solution to (10) does not exist, it indicates that $a(t)$ is not admissible. Whenever this occurs, we perform the following procedure. Firstly, as shown in (9), we assign the RL agent a large penalty $-G$ to inform it that the exploratory action is unsafe. Secondly, in order to continue the learning process, we randomly sample an admissible action vector $\tilde{a}(t)$ that satisfies $\begin{bmatrix} x(t)^{\mathrm{T}} & \tilde{a}(t)^{\mathrm{T}} & U'^{\mathrm{T}} \end{bmatrix} \in P^\beta$ for some U'. Towards that end, we use a hit-and-run sampling scheme, which works efficiently to sample points inside a polyhedron [26]. It works as follows: for P^β, we begin with an initial point $p_0 \in P^\beta$. We then generate a random unit vector v uniformly sampled on $S^{N\cdot m-1} := \{x \in \mathbb{R}^{N\cdot m} : \|x\|_2 = 1\}$ and compute the smallest λ satisfying $p_0 + \lambda v \in P^\beta$, *i.e.*,

$$\lambda_0 = \min\{\lambda' : p_0 + \lambda' v \in P^\beta\}. \tag{11}$$

We then set,

$$p_1 = p_0 + \mu\lambda v, \tag{12}$$

where μ is a random variable uniformly sampled from the interval $[0, 1]$. We repeat this procedure to generate the sequence $\{p_k\}$. As k approaches ∞, it is guaranteed that a point in P^β will be sampled uniformly [26]. As a practical matter, we stop the procedure at some $k = k^*$, and set $\tilde{a}(t) = p_{k^*}[1 : m]$, the first m elements of the vector p_{k^*}.

Remark 4.1: The augmented reward defined in (9) has three terms, the regular cost $c(x(t))$, an efforts-to-retain-safe cost $E(t)$, and a big penalty G for unsafe explorations. These augmented reward signals present dynamic-enabled information that can guide the RL agent learn the system constraints efficiently.

B. Application to Active Suspension System

In this section, we apply the proposed safe RL to the active suspension system illustrated in Figure 2, in which M_s and M_{us}, respectively, represent the sprung mass (car body) and unsprung mass (wheel) of the quarter car; z_s and z_{us} represent the displacement of M_s and M_{us} from the equilibrium; the suspension is modeled as a spring with stiffness k_s and a damper with damping coefficient c_s. As opposed to semi-active suspension control, active suspension system can generate actuation force F from an actuator; the tire is modeled as a spring-damper system with spring stiffness k_t and damping coefficient c_t; and z_r is the vertical road disturbance.

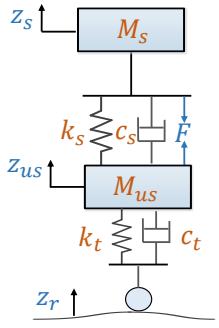

Fig. 2. A quarter-car active suspension system.

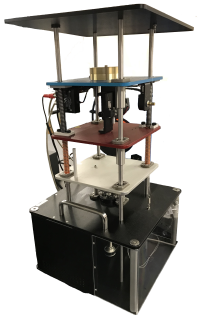

Fig. 3. Quanser Active Suspension Station

Define $x = [z_s - z_{us}, \dot{z}_s, z_{us} - z_r, \dot{z}_{us}]^{\mathrm{T}}$, $u = F$, $w = \dot{z}_r$ and $y = [x_1, \dot{x}_2, x_3, u]$, then the equation of motion of the suspension system can be written as:

$$\begin{aligned} \dot{x} &= A_o x + Bu + B_w w, \\ y &= C_o x + Du, \end{aligned} \tag{13}$$

where $A_o = \begin{bmatrix} 0 & 1 & 0 & -1 \\ -\frac{k_s}{M_s} & -\frac{c_s}{M_s} & 0 & \frac{c_s}{M_s} \\ 0 & 0 & 0 & 1 \\ \frac{k_s}{M_{us}} & \frac{c_s}{M_s} & -\frac{k_t}{M_{us}} & -\frac{c_s+c_t}{M_{us}} \end{bmatrix}$, $B_o = \begin{bmatrix} 0 \\ \frac{1}{M_s} \\ 0 \\ -\frac{1}{M_{us}} \end{bmatrix}$,

$B_w = \begin{bmatrix} 0 \\ 0 \\ -1 \\ \frac{c_t}{M_{us}} \end{bmatrix}$, $C_o = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -\frac{k_s}{M_s} & -\frac{c_s}{M_s} & 0 & \frac{c_s}{M_s} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$, and $D = \begin{bmatrix} 0 \\ \frac{1}{M_s} \\ 0 \\ 1 \end{bmatrix}$.

The system constraints are represented using the output vector. Specifically, the suspension displacement travel is limited due to mechanical design and in the Quanser suspension station (see Figure 3) we have $-1 \le y_1 \le 0.015$; the vertical body acceleration is related to ride comfort and is subject to $-2 \le y_2 \le 2$; the tire deflection represents the road handling and is also limited by the mechanical design in the Quanser suspension system: $y_3 \ge -0.015$; and the control force is constrained as $-10 \le u \le 10$ due to physical limitations.

We implement a nominal LQR controller with a gain K corresponding to $Q = diag\{10, 0, 1, 0\}$ and $R = 1e-3$. Then system (13) can be re-written as

$$\begin{aligned} \dot{x} &= Ax + Bv + B_w w, \\ y &= Cx + Du, \end{aligned} \tag{14}$$

where $A = A_o - BK$ and $C = C_o - DK$ are the closed-loop system matrices. v is the control adjustment and the goal is to learn a policy $v(x(t))$ such that the system constraints are satisfied while maximizing riding comfort, i.e., minimizing $|y_2|$. We apply the proposed safe RL algorithm and the results are shown in the next subsection.

C. Learning implementations

While the proposed safe RL framework can work with many off-policy RL algorithms, in this paper, we use Deep Deterministic Policy Gradient (DDPG) as the RL strategy due to its capability of dealing with continuous controls. DDPG learns both a *critic* network to estimate the long-term value for a given policy and a *actor* network to sample the optimal action according to the value estimation. More details of DDPG algorithm can be found in [27]. In this section we mainly address how to integrate the proposed augmented reward and safe sampling (see Section IV-A) with the conventional DDPG to ensure safety in both learned policies and exploration during training.

As many off-policy RL algorithms, DDPG applies action noise on the output of the *actor* network to generate exploratory actions and maintains a replay buffer to store the explored experience $(x(t), a(t), x(t+1), r(t))$ for training. In particular, it adopts the *Ornstein-Uhlenbeck* (OU) process to achieve temporally correlated exploration:

$$a(t) = \min(\max(\pi_\theta(x(t)) + \epsilon_{OU}(t), u_{\min}), u_{\max}), \tag{15}$$

where $u_{\min}$ and $u_{\max}$ are used to constrain the exploration to the valid control range. Obviously, there is no safety guarantee on $a(t)$, and unsafe exploration is indeed required for DDPG to learn to avoid it in the future. By virtue of the admissible set developed earlier, we can determine if a certain exploration is safe without actually applying it to the system. Specifically we solve Eq. (10) to find the safety effort $E(x(t), a(t))$ for a sampled OU exploration $a(t)$. If there is no valid solution, it indicates that the current exploration will lead to unsafe conditions as discussed in Section IV-A. Then we find its closest safe control $\tilde{a}(t)$ by projecting it to the admissible set and perform $\tilde{a}(t)$ instead to the system to collect the experience $(x(t), \tilde{a}(t), x(t+$

$1), r(t))$. In this way all exploratory actions are guaranteed to be safe during training. However, the agent will not be able to learn a safe policy at the end as it never has a chance to explore unsafe actions and update the value estimations accordingly. To address this issue, we create a virtual experience $(x(t), a(t), x(t+1), -G)$ whenever $a(t)$ is the original unsafe exploration function. The pseudo-code of this modified algorithm is illustrated in Algorithm 1, on top of traditional model-free DDPG.

Normalization is an important step in DDPG training as the scale of each input signal is maintained when it is passed through critic and actor networks. To obtain appropriate normalization factors, we run a few trajectories with uniformly sampled admissible actions to find the typical range for each state, and then we calculate the normalization offset and scale accordingly. Note all normalized states are clipped to $[-1.5, 1.5]$ to prevent outliers.

Algorithm 1: Safe DDPG

initialize DDPG model and replay buffer;
for *each training episode* **do**
  initialize a safe initial state $x(0)$;
  while $t < T$ **do**
    observe state $x(t)$;
    /* OU exploration */
    sample exploration $a(t)$ using (15);
    /* safety regulation */
    if (10) *returns a solution* **then**
      retain control $u(t) = a(t)$;
    else
      project $a(t)$ to the admissible polyhedral P^β to obtain an admissible control $u(t) = \tilde{a}(t)$ using (11) and (12);
    end
    /* experience collection */
    perform safe control $u(t)$ to system ;
    observe next state $x(t+1)$, reward $r(t)$ by (9);
    store $(x(t), u(t), r(t), x(t+1))$ to replay buffer;
    if $u(t) \neq a(t)$ **then**
      store $(x(t), a(t), -C, x(t+1))$ to replay buffer;
    end
    update traditional DDPG using sampled data from replay buffer;
  end
end

Now we describe the implementation regarding practical challenges in the active suspension control systems. The road disturbance is specified as a combination of Gaussian process with zero mean and variance $1e-5$ and random jumps with jump size 0.001 for 10 steps. Those jumps can represent road anomalies such as potholes and speed bumps. Due to these jumps, training a single safe and optimal DDPG agent is challenging since the agent only learns a statistical knowledge about jump distributions while the violation usually happens when there are frequent jumps within a particular time window. To address this challenge, we train two DDPG agents: DDPG_{opt} and $\text{DDPG}_{\text{safe}}$. DDPG_{opt} learns a safe and optimal action statistically, with $\alpha = 0.2$ and a wide exploration range $a(t) \in [-2, 2]$. $\text{DDPG}_{\text{safe}}$ learns a safe action that will not cause violation even for frequent jumps, with $\alpha = 0.5$ and a conservative exploration range $a(t) \in [-1, 1]$. The states for $\text{DDPG}_{\text{safe}}$ also include a state that indicates if there is a jump during the last step. We set episode length of each run as $T = 500$ steps and safety penalty as $G = 100$.

D. Results

For fair comparison, we train DDPG_{opt} and $\text{DDPG}_{\text{safe}}$ over 100 episodes with shared random seeds, and we evaluate their performance over 5 testing episodes after every 5 training episodes. Note both agents perform safe exploration during training episodes while they strictly follow the learned policies during testing episodes. First, we compare the learning capability of optimality between DDPG_{opt} and $\text{DDPG}_{\text{safe}}$, by plotting the evaluated returns $R = \sum_{t=0}^{T} r(t)$ in Fig. 4. To prevent exploited value after huge penalty after safety violation, only non-violated testing episodes are used for return evaluation. The curves in Fig. 4 show the average return while the shades show the standard deviations of returns across testing episodes. Obviously, DDPG_{opt} is able to learn a more optimal policy after a wider exploration, and the evaluated return is improved from -24 to -16 over training episodes. On the other hand, $\text{DDPG}_{\text{safe}}$ barely improves return due to a safety dominated reward definition and discouraged exploration.

Second, we compare the learning capability of safety between DDPG_{opt} and $\text{DDPG}_{\text{safe}}$, by plotting the evaluated number of violations in Fig. 4. As expected, both agents are able to reach zero violation over this small set of testing episodes near the end. More specifically, DDPG_{opt} is able to reach constant zero violations after 20 training episodes, while $\text{DDPG}_{\text{safe}}$ is able to achieve a safe policy even at the first evaluation. This may because of the narrow exploration range, the safety focused reward, and the additional state of jump indicator in $\text{DDPG}_{\text{safe}}$.

Third, Fig. 6 illustrates the convergence of both agents by plotting the average weight magnitude over training steps. In order to remove the dependency on randomly generated initial weights, we divide the absolute weight magnitude by that of initial weight to estimate the relative weight magnitude instead. Based on Fig. 6, $\text{DDPG}_{\text{safe}}$ shows nice monotonic convergence while DDPG_{opt} experiences some difficulties to find a stable optimal solution, which may because the optimal policy is sensitive to non-Markovian jump distribution.

Finally, we compare the performance of trained DDPG agents against the nominal controller (LQR) over a larger evaluation set of 500 episodes, using the same random seed across agents for each episode. Similarly, the violation rate and average return over non-violated runs are recorded to demonstrate the safety and optimality of the controller. We

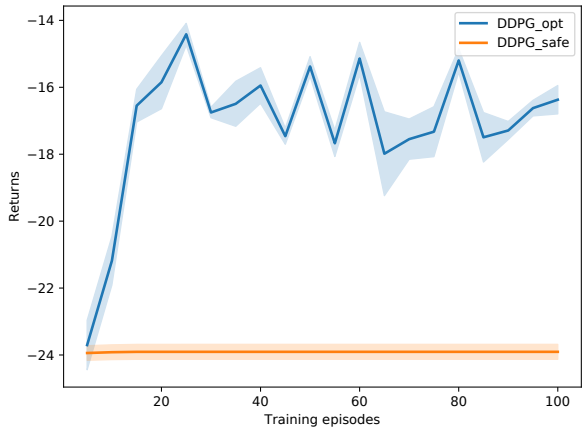

Fig. 4. Optimality comparison between DDPG_{opt} and $\text{DDPG}_{\text{safe}}$ during training. Curve and shade indicate the average and standard deviation of returns over non-violated testing episodes.

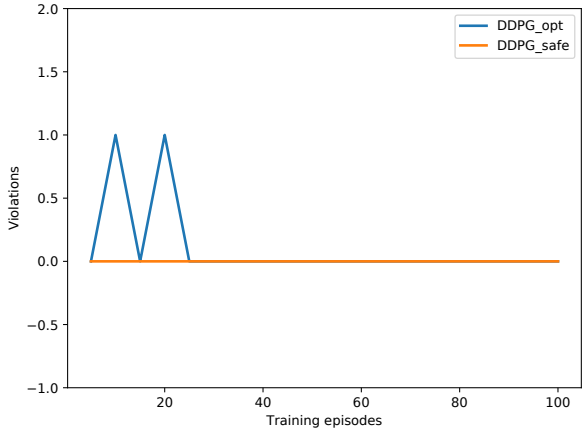

Fig. 5. Optimality comparison between DDPG_{opt} and $\text{DDPG}_{\text{safe}}$ during training. Curve indicates the number of testing episodes that are early terminated due to safety violation.

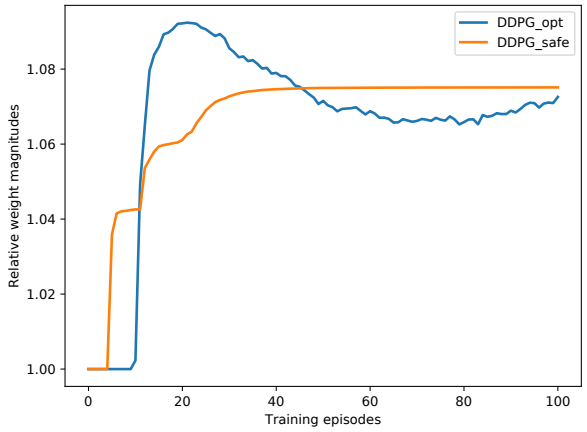

Fig. 6. Convergence comparison between DDPG_{opt} and $\text{DDPG}_{\text{safe}}$ during training. Curve shows average weight magnitude per training step w.r.t. the initial weight.

first evaluate DDPG_{opt} and $\text{DDPG}_{\text{safe}}$ separately, and then evaluate a combined agent $\text{DDPG}_{\text{comb}}$ that applies DDPG_{opt} if there is no jump detected in the last step and $\text{DDPG}_{\text{safe}}$ otherwise. The result is summarized in Table I. As expected, $\text{DDPG}_{\text{comb}}$ beats the nominal controller in both accumulated return (-23.14 vs. -23.61) and constraint preservations (0% vs. 1.8%).

TABLE I
EVALUATION RESULTS.

	DDPG_{opt}	$\text{DDPG}_{\text{safe}}$	$\text{DDPG}_{\text{comb}}$	Nominal
violation rate	1.9%	0.0%	0.0%	1.8%
average return	-16.04	-24.01	-23.14	-23.61

V. Conclusions and Future Work

In this paper, we developed a deep safe RL framework that integrates model-based safety supervision and model-free learning. We exploit the underlying dynamics and system constraints to construct a safety set for learning exploration regulation using recursive feasibility techniques. We design augmented reward signals to efficiently guide the RL agent to learn the system constraints. We use the hit-and-run sampling technique to sample safe actions to preserve exploration efficiency. We applied the safe RL framework on an active suspension station and showed that we can learn a safe policy that outperforms the nominal controller. Future work will include the implementation and testing in the physical suspension station plant. We will also extend the framework to general nonlinear systems.

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, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, Jan. 2016.

[2] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," *Nature*, vol. 550, pp. 354–, Oct. 2017.

[3] B. Dhingra, L. Li, X. Li, J. Gao, Y. Chen, F. Ahmed, and L. Deng, "End-to-end reinforcement learning of dialogue agents for information access," *CoRR*, vol. abs/1609.00777, 2016.

[4] J. Li, W. Monroe, A. Ritter, M. Galley, J. Gao, and D. Jurafsky, "Deep reinforcement learning for dialogue generation," *CoRR*, vol. abs/1606.01541, 2016.

[5] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep direct reinforcement learning for financial signal representation and trading," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, pp. 653–664, March 2017.

[6] J. N. Tsitsiklis and B. V. Roy, "Regression methods for pricing complex american-style options," *IEEE Transactions on Neural Networks*, vol. 12, pp. 694–703, Jul 2001.

[7] M. Heger, "Consideration of risk in reinforcement learning," 1994.

[8] A. Nilim and L. El Ghaoui, "Robust control of Markov decision processes with uncertain transition matrices," *Operations Research*, vol. 53, no. 5, pp. 780–798, 2005.

[9] M. Sato, H. Kimura, and S. Kobayashi, "Td algorithm for the variance of return and mean-variance reinforcement learning," *Transactions of the Japanese Society for Artificial Intelligence*, vol. 16, no. 3, pp. 353–362, 2001.

[10] P. Geibel and F. Wysotzki, "Risk-sensitive reinforcement learning applied to control under constraints," *J. Artif. Int. Res.*, vol. 24, pp. 81–108, July 2005.
[11] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.
[12] W. Saunders, G. Sastry, A. Stuhlmüller, and O. Evans, "Trial without error: Towards safe reinforcement learning via human intervention," *CoRR*, vol. abs/1707.05173, 2017.
[13] A. K. Akametalu, J. F. Fisac, J. H. Gillula, S. Kaynama, M. N. Zeilinger, and C. J. Tomlin, "Reachability-based safe learning with Gaussian processes," in *53rd IEEE Conference on Decision and Control*, pp. 1424–1431, Dec 2014.
[14] K. P. Wabersich and M. N. Zeilinger, "Linear model predictive safety certification for learning-based control," *CoRR*, vol. abs/1803.08552, 2018.
[15] R. B. Larsen, A. Carron, and M. N. Zeilinger, "Safe learning for distributed systems with bounded uncertainties," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 2536 – 2542, 2017. 20th IFAC World Congress.
[16] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. H. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems," *CoRR*, vol. abs/1705.01292, 2017.
[17] K. P. Wabersich and M. N. Zeilinger, "Scalable synthesis of safety certificates from data with application to learning-based control," *CoRR*, vol. abs/1711.11417, 2017.
[18] Z. Li, U. Kalabić, and T. Chu, "Safe reinforcement learning: Learning with supervision using a constraint-admissible set," in *2018 Annual American Control Conference (ACC)*, pp. 6390–6395, June 2018.
[19] H. E. Tseng and D. Hrovat, "State of the art survey: active and semi-active suspension control," *Vehicle System Dynamics*, vol. 53, no. 7, pp. 1034–1062, 2015.
[20] Y. M. Sam, M. R. H. A. Ghani, and N. Ahmad, "Lqr controller for active car suspension," in *2000 TENCON Proceedings. Intelligent Systems and Technologies for the New Millennium (Cat. No.00CH37119)*, vol. 1, pp. 441–444 vol.1, Sept 2000.
[21] A. H. Shirdel, E. Gatavi, and Z. Hashemiyan, "Comparison of h-infinity and optimized-lqr controller in active suspension system," in *2010 Second International Conference on Computational Intelligence, Modelling and Simulation*, pp. 241–246, Sept 2010.
[22] C. Göhrle, A. Wagner, A. Schindler, and O. Sawodny, "Active suspension controller using mpc based on a full-car model with preview information," in *2012 American Control Conference (ACC)*, pp. 497–502, June 2012.
[23] B. E. Durmaz, B. Kaçmaz, . Mutlu, and M. T. Söylemez, "Implementation and comparison of lqr-mpc on active suspension system," in *2017 10th International Conference on Electrical and Electronics Engineering (ELECO)*, pp. 828–835, Nov 2017.
[24] I. Kolmanovsky and E. G. Gilbert, "Theory and computation of disturbance invariant sets for discrete-time linear systems," *Math. Problems Eng.*, vol. 4, pp. 317–367, 1998.
[25] U. Kalabić, C. Vermillion, and I. Kolmaonvsky, "Constraint enforcement for a lighter-than-air wind-energy system: An application of reference governors with chance constraints," in *Proc. IFAC World Congress*, (Toulouse, France), pp. 13258–13263, Jul. 2017.
[26] C. Bélisle, A. Boneh, and R. J. Caron, "Convergence properties of hit–and–run samplers," *Communications in Statistics. Stochastic Models*, vol. 14, no. 4, pp. 767–800, 1998.
[27] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.