2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Reinforcement Learning-based Optimal Control and Software Rejuvenation for Safe and Efficient UAV Navigation

Angela Chen, Konstantinos Mitsopoulos, and Raffaele Romagnoli

Abstract— **Unmanned autonomous vehicles (UAVs) rely on effective path planning and tracking control to accomplish complex tasks in various domains. Reinforcement Learning (RL) methods are becoming increasingly popular in control applications, as they can learn from data and deal with unmodelled dynamics. Cyber-physical systems (CPSs), such as UAVs, integrate sensing, network communication, control, and computation to solve challenging problems. In this context, Software Rejuvenation (SR) is a protection mechanism that refreshes the control software to mitigate cyber-attacks, but it can affect the tracking controller's performance due to discrepancies between the control software and the physical system state. Traditional approaches to mitigate this effect are conservative, hindering the overall system performance. In this paper, we propose a novel approach that incorporates Deep Reinforcement Learning (Deep RL) into SR to design a safe and high-performing tracking controller. Our approach optimizes safety and performance, and we demonstrate its effectiveness during UAV simulations. We compare our approach with traditional methods and show that it improves the system's performance while maintaining safety constraints. Our approach takes 10 seconds less to reach the goal and we interpret this enhancement through a p-norm analysis.**

I. Introduction

Path planning and tracking control are key elements for unmanned autonomous vehicles (UAVs). Reinforcement Learning [1] is gaining more attention in control applications [2] since it is able to deal with unmodelled dynamics by learning them from data. UAVs are applications of cyber-physical systems (CPSs) that integrate sensing, network communication, control, and computational methods to solve complex applications in applications such as transportation, healthcare, power supply, etc [3]. In general, the controller design considers only the physical dynamics of a CPS because it is assumed that the inertia of the physical system is slower than any operation performed in the cyber part. Due to the complexity of a CPS, that assumption is getting more and more unrealistic, particularly when solutions to protect the CPS from cyber-attacks are implemented [4], [5], [6]. This is the case of software rejuvenation (SR) [7], [8] which is a mechanism of protection that refreshes the run-time control software in order to mitigate the possible negative effects of a cyber-attack on it. This mechanism of protection imposes constraints to be satisfied, for example in [9], the trajectory setpoints must be updated only under specific conditions that involve time and system dynamics. Despite its effectiveness, in terms of safety and mission liveness [10], the overall control performance can be very poor in terms of trajectory tracking. One of the main issues is that there is a discrepancy between the state of the control software and the actual state of the system. In fact, at each software refresh a previous uncorrupted image of the control software is loaded. This discrepancy becomes more evident in the case of the controller making use of the state estimation. Modeling this aspect into the physical system dynamics in order to develop a tracking controller that mitigates this effect can be very challenging. In the SR framework, the trajectory tracking controller generates a sequence of setpoints that takes into account safety accordingly to Lyapunov's theory [11]. In real applications, the effects of the software rejuvenation on the state estimation error make it difficult to be modeled and find the optimal trajectory tracking algorithm that can improve the overall system performance. In fact, the proposed solutions are quite conservative which is good from the safety viewpoint, but this can be very limiting in terms of the application viewpoint.

In this paper, we involve Deep RL in the SR problem for the design of a safe tracking controller that also considers the system's performance during the mission. Our objective is to show the applicability and effectiveness of Deep RL in this context tracing the way of future research directions that combines control theory and Deep RL for safe-critical applications.

Deep RL algorithms learn optimal control policies by iteratively optimizing a reward function that measures the success of the control policy. While this approach have been successfully applied to many control problems, in this work we do not intend to replace traditional control methods. Rather, the objective is to integrate it into existing control frameworks to improve their performance and safety. In this paper, we apply it to the SR problem in control theory and demonstrate its potential to enhance the safety and performance of UAVs. Specifcially, we show that our apporach mitigates the effect of SR on the tracking controller's performance compared to traditional approaches. Our work contributes to the growing body of research that combines control theory and Deep RL to address critical safety issues in cyber-physical systems.

II. Preliminaries

Let us consider a positive definite matrix $M > 0$ with $M \in \mathbb{R}^n$, and a vector $v \in \mathbb{R}^n$, the norm of v w.r.t P,

A. Chen and R. Romagnoli are with the Department of Electrical and Computer Engineering, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213, USA: xinyuc2@andrew.cmu.edu rromagno@andrew.cmu.edu

Konstantinos Mitsopoulos is with the Institute for Human and Machine Cognition, 40 South Alcaniz St, Pensacola, FL 32502, USA: kmitsopoulos@ihmc.org

979-8-3503-0123-6/23/$31.00 ©2023 IEEE

$P-$norm is

$$\|v\|_P = \sqrt{v^T P v}. \tag{1}$$

The ellipsoid of size ρ centered in $c \in \mathbb{R}^n$

$$\mathcal{E}(\rho, c) = \left\{ v \in \mathbb{R}^n \mid \|v - c\|_P^2 \leq \rho \right\}.$$

A linear time-invariant (LTI) continuous-time system is described by

$$\dot{x} = Ax + Bu, \tag{2}$$

where $x \in \mathbb{R}^n$ represents the state of the system, $u \in \mathbb{R}^p$ is the input vector, the matrix $A \in \mathbb{R}^{n \times n}$, and the matrix $B \in \mathbb{R}^{n \times p}$. The output vector is $y = Cx$ with $y \in \mathbb{R}^q$ and $C \in \mathbb{R}^{q \times n}$. Let us consider a state feedback controller $u = -Kx$ that defines the closed-loop system

$$\dot{x} = (A - BK)x, \tag{3}$$

where matrix $A - BK$ is Hurwitz. Since the controlled system is asymptotically stable there exists a positive definite matrix $P > 0$ with $P \in \mathbb{R}^{n \times n}$ that satisfies the Lyapunov equation

$$(A - BK)^T P + P(A - BK) = -Q, \tag{4}$$

where $Q \in \mathbb{R}^{n \times n}$ and $Q > 0$. The ellipsoid centered at the origin $\mathcal{E}(\rho, 0)$ is a Lyapunov level set which is positively invariant. Moving the system to another equilibrium point (or setpoint) x_{sp} the new control law is $u = -K(x - x_{sp})$, then the closed-loop system can be rewritten as

$$\dot{x} = (A - BK)(x - x_{sp}). \tag{5}$$

The Lyapunov analysis remains the same except for the origin of the system which is translated as the ellipsoid $\mathcal{E}(\rho, x_{sp})$. In case the state of the system is not measurable, and only the measurements y are available, a state estimation $\hat{x} \in \mathbb{R}^n$ can be used to close the loop. If the system is observable, thanks to the separation principle it is possible to design a deterministic observer

$$\dot{\hat{x}} = A\hat{x} + Bu + L\left(C\hat{x} - y\right). \tag{6}$$

By defining the estimation error $e \triangleq x - \hat{x}$, and substituting y with Cx, the dynamics of the estimation error is

$$\dot{e} = (A - LC)e. \tag{7}$$

Thanks to the observability property of the system it is possible to design L that makes the matrix $(A - LC)$ Hurwitz. The new control input is now

$$u = -K(\hat{x}(t) - x_{sp}). \tag{8}$$

III. Software Rejuvenation

Fig. 1 describes the SR approach over time. At the beginning of the mission the drone is in secure control (SC) mode which means that the control software is not vulnerable to attacks (e.g. not connected to the communication network). Before switching to mission control (MC) mode an image of the run-time software can be saved in a protected memory location, checkpoint (CP) since the system is assumed to be clean. During MC the drone can communicate through the

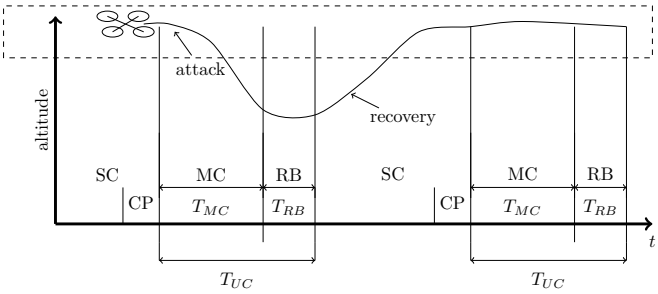

Fig. 1: Software Rejuvenation timeline.

communication network and then be vulnerable to cyber-attacks. To avoid possible catastrophic consequences of a worst-case attack, a protected timer triggers the software refresh before it is too late for preventing any irreversible damage to the system. The amount of time the system is in MC mode is indicated by T_{MC}. During software refresh, the saved clean image of the run-time software is rolled back (RB), and the time needed for this operation is indicated with T_{RB}. During RB, the control input is kept constant and equal to the last provided. The total time the system is under unknown control is $T_{UC} = T_{MC} + T_{RB}$. Fig. (2) shows the mode-switching graph and it offers more details in particular for the recovery and setpoint update. For the

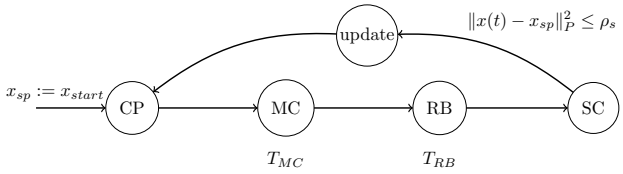

Fig. 2: SR mode-transition graph. Unlabeled transitions occur immediately after the operations for the preceding mode are completed, or when the time indicated for the mode has elapsed.

moment we consider that $x(t)$ is available and it is used to compute the control law, hence the time spent by the system in SC mode depends on the following condition

$$\|x(t) - x_{sp}\|_P^2 \leq \rho_s \tag{9}$$

that determines that the system has been fully recovered, and $0 < \rho_s < 1$. Since after RB, the software is clean and the system is in SC mode, and x_{sp} is the same saved during the previous CP, all the information used in (9) is not corrupted. If there is no attack, the time spent in SC mode can last only the period to check (9).

A. Safety and Setpoint Update

For a given setpoint x_{sp}, the safety set is provided by the ellipsoid $\mathcal{E}(1, x_{sp})$ which is an invariant set for the controlled system (5). Considering a ρ_m such that $0 < \rho_s < \rho_m < 1$, we compute T_{MC} as the time that $\forall x(t) \in \mathcal{E}(\rho_m, x_{sp})$, $x(t)$ is always recoverable into $\mathcal{E}(\rho_s, x_{sp})$ [11]. For the trajectory tracking, we assume that the setpoints x_{sp} are generated

along the line that joins two waypoints w_i, and w_{i+1}. The safety condition for the setpoint transition is

$$\|x(t) - x_{sp}\|_P^2 \leq \rho_m \Rightarrow \|x(t) - x'_{sp}\|_P^2 \leq \rho_m, \quad (10)$$

where x'_{sp} is the new setpoint [10]. Fig.3 shows the Assuming that the state is available, the above condition is verified if x_{sp} is updated as

$$x'_{sp} = x_{sp} + (\sqrt{\rho_m} - \sqrt{\rho_s})\mathbf{v}, \quad (11)$$

where $\mathbf{v}$ is the unitary vector along the trajectory.

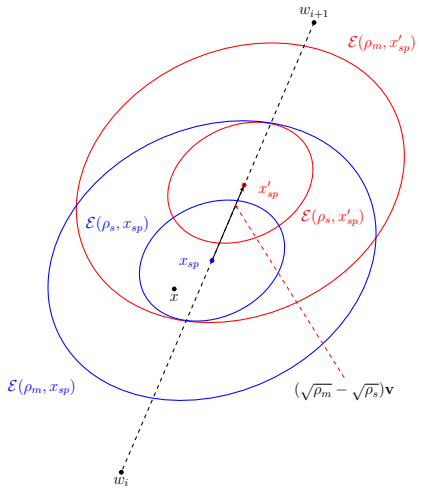

Fig. 3: Safe setpoint transition scheme.

B. State Estimation

In a real application, only the state estimation $\hat{x}$ is available. This information is stored in the run-time software and during RB the state estimation is computed starting from the initial conditions saved during CP. Since $\hat{x}$ is used to compute u (8), after each RB there is the effect of the estimation error e that may increase after each SR cycle making the system unstable. Moreover, $\hat{x}$ is now used for evaluating (9), and large estimation can make the SR scheme switch when the system cannot be exposed to possible attacks. In this situation, the safety conditions for T_{MC} and setpoint generation, can be the same by just replacing $x(t)$ with $\hat{x}(t)$ if T_{est} is introduced [12]. T_{est} is the minimum time for the system to be in SC mode after software refresh in order to reduce the estimation error to keep the system stable and safe against attacks.

C. Problem Statement

In this paper, we consider a UAV whose nonlinear dynamics can be reduced into the form of (2) and stabilized around a setpoint x_{sp} by a linear controller (3). We also assume that the state x is not directly accessible and a state observer (6) is needed to compute u as in (8) and for evaluating the SR condition (9). We assume that $\mathcal{E}(\rho_s, x_{sp})$, $\mathcal{E}(\rho_m, x_{sp})$, T_{MC}, and T_{est} given and they ensure safety against cyber-attacks [9].

In this paper, we are interested in the improvement of the performance of the system protected via SR when the system is not under attack. Specifically, we redesign (11) as

$$x'_{sp} = x_{sp} + \alpha\mathbf{v} \quad (12)$$

with

$$\alpha = f(\hat{x}, x_{sp}; \theta), \quad (13)$$

where f is a neural network with parameters θ optimized with RL. This formulation aligns with reference governor (RG) [13], [14] and explicit reference governor (ERG) [15] frameworks. The safe-trajectory controller for SR can be regarded as an ERG, albeit with distinct operating conditions. With our method we aim to show that the effects of the SR scheme can be captured by a learning technique which it would be difficult to model with the traditional control tools as shown in [9]. The goal is to demonstrate that the efficacy of RL approach improves performance in terms of reducing the time of the mission, while the safety conditions are satisfied. Finally, we also consider the presence of noise in the measurements.

IV. Learning Setpoint Generation

We consider a task where a UAV is required to navigate from a starting location A to a goal location B within a bounded 3D space free of obstacles. The UAV is controlled by (8), and we assume that an RL agent must effectively learn to modulate the displacement of a setpoint at discrete timesteps, based on input information related to the UAV's state. To accomplish this, the agent must select an appropriate value for the parameter α, which determines the magnitude of the displacement modulation. Between two consecutive decision-making points in simulation, disturbances SR affect the UAV, including state estimation errors that depend on the agent's choice of α. Generally, a higher value of α leads to a greater degree of disturbance experienced by the UAV. The main objective of the learning agent is to identify an optimal value of α that can modulate the UAV's displacement in a way that respects the safety constraints - discussed in Section III-A - while simultaneously improving the speed of the UAV.

In contrast to (11), which employs a more conservative approach that prioritizes safety but overlooks performance, our proposed approach seeks to optimize both safety and performance. Specifically, by using a learning agent that can dynamically adjust the value of α in response to the UAV's state, we can achieve a better balance between safety and performance, leading to improved task outcomes.

A. Reinforcement Learning

We formulate this task as a Markov Decision Process [16] (MDP) which is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, T, \mathcal{R}, \gamma \rangle$ where:

- $\mathcal{S}$ denotes the continuous state space; in our case the state at decision timestep k is $s_k = \hat{x} - x_{sp}$,
- $\mathcal{A}$ the continuous action space; in our case the action at timestep k is $a_k = \alpha \in [0, 1]$,
- $\mathcal{T}$ is the transition probability for arriving into state s_{k+1} when executing action a_k from state s_k,

- $\mathcal{R}$ is the reward function that defines the reward received by the agent for transitioning from state s_k to state s_{k+1} when taking action a_k,
- and γ is the discount factor that determines the importance of future rewards relative to immediate rewards. In this case, it can be used to model the trade-off between short-term and long-term objectives.

The objective of the agent is to maximize the expected return $G_k = \sum_{l=0}^{\infty} \gamma^l r_{k+l+1}$ from each state s_k, where r_k denotes a specific instance of the reward function, obtained at evaluating it at a specific state-action pair. The reward function in our case is defined in section V-A.

A solution to an MDP is obtained by finding an optimal policy $\pi(. \mid s_k)$, that maps a state s_k to a distribution over possible actions that lead the agent to higher sums of rewards. The probability of performing action a_k in state s_k is denoted by $a_k \sim \pi(a \mid s_k)$.

One way to obtain an optimal policy is to use value-based RL methods. The action value $Q^{\pi}(s,a) = \mathbb{E}[G_k \mid s_k = s, a]$ is the expected return for selecting action a in state s and following policy π. The optimal value function $Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a)$ gives the maximum action value for state s and action a achievable by any policy. Similarly, the value of state s under policy π is defined as $V^{\pi}(k) = \mathbb{E}[G_{\mathrm{t}} \mid s_k = s]$ and is simply the expected return for following policy π from state s. The optimal state value function is given by $V^*(s) = \max_{a \in \mathcal{A}} Q^{\pi^*}(s,a)$. Value functions can be used to define a policy (e.g. ϵ-greedy). RL methods that estimate value functions are usually called *critic methods*.

In many real-world scenarios, the state and action spaces of an MDP are so large that it is impractical to enumerate all possible combinations. For an agent to learn a successful policy it is necessary to be able to estimate value functions of unseen states. The action value function could be represented using a function approximator, such as a neural network. Let $Q(s,a;\theta)$ be an approximate action-value function with parameters θ. The updates to θ can be derived from a variety of reinforcement learning algorithms which aim to directly approximate the optimal action value function: $Q^*(s,a) \approx Q(s,a;\theta)$.

In contrast to value-based methods, policy-based model-free methods directly parameterize the policy $\pi(a \mid s;\theta)$ and update the parameters θ by performing, typically approximate, gradient ascent on $\mathbb{E}[G_k]$. One example of such a method is the REINFORCE family of algorithms [17] which updates the policy parameters θ in the direction $\nabla_{\theta} \log \pi(a_k \mid s_k;\theta) G_k$. Such types of methods are called *actor methods*. As discussed, we can introduce an estimation of the return in the form of a critic which results in Actor-Critic methods.

In this work we employ the Soft Actor Critic [18] (SAC) algorithm to demonstrate the importance of learning methods in improving performance and ensuring safety in control applications. SAC is an entropy-regularized RL method that changes the RL problem (i.e obtain an optimal policy π^*) to:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{k=0}^{K} \gamma^k \left(R(s_k,a_k) + \beta H(\pi(\cdot \mid s_k))\right)\right], \tag{14}$$

where the temperature parameter β controls the stochasticity of the optimal policy as it determines the relative importance of the entropy H of the policy term against the reward. SAC incorporates a modified action and state value functions that offer the agent a bonus proportionate to the policy's entropy. This approach renders policies optimized for maximum entropy ([19], [20]) more robust, allowing for a greater ability to respond successfully to unexpected perturbations during testing. Additionally, optimizing for maximum entropy during training can improve both the algorithm's robustness to hyperparameters and its sample efficiency, making SAC a useful tool for control problems [21].

V. Experimental setup and Simulation Results

A. Simulation Environment

To simulate the interaction between the RL agent and the UAV system, we developed a customized OpenAI gym environment. The environment models the nonlinear dynamics of the UAV system [9], along with the effects of the software rejuvenation and recovery periods. The state estimation $\hat{x}(t)$, computed as (6), is evaluated after each cycle of the SR scheme of Fig. 2, with $T_{MC} = 200$ ms, $T_{RB} = 10$ ms, and $T_{est} = 1.7$ s. Those numbers have been computed accordingly to [9]. The total time needed for one cycle of SR is at least 1.910s. At approximately every 2s interval, the RL agent receives the current state s_k of the system and selects an action $a_k = \alpha$, indicating its displacement from the current location as depicted in 3. Based on (11), the α value is bounded as $0 \leq \alpha \leq \sqrt{\rho_m} - \sqrt{\rho_s}$ for safety considerations, and we set the size of the outer ellipsoid $\rho_m = 0.01$ and the inner ellipsoid $\rho_s = 0.0012$. In this experiment, the drone starts from (1,1,1) and stops when it reaches (5,5,5).

Reward Function: The reward function considers the effect of the action, generated by the agent, to the SR period and how far from the goal the UAV is:

$$R(s_k,a_k) = -r_{\text{mpn}} - \|x_k - x_{\text{goal}}\|_P^2, \tag{15}$$

where r_{mpn} is the maximum p-norm of all $\|x(t) - x_{sp}\|_P^2$ evaluated during the entire SR cycle. The groundtruth state when the SR cycle has been completed is indicated by x_k. At any point in time if the system was becoming unstable ($\|x(t) - x_{sp}\|_P^2 > 10$) we terminated the simulation with $r_{\text{mpn}} = 10$. Since the policy gradient method aims to maximize the total expected return and our reward function penalizes the agent if it stagnates at non-goal points, our proposed approach ensures a consistent trajectory towards the goal along the specified path.

Baseline Method: This method refers to the setpoint update (11). To make a fair comparison with the RL method, we used the following computation

$$\alpha(\hat{x}, x_{sp}) = \sqrt{\rho_m} - \|\hat{x}(t) - x_{sp}\|_p, \tag{16}$$

where $\hat{x}(t)$ is the current state estimation, x_{sp} denotes the current setpoint, and ρ_m is the size of the outer ellipsoid set. The new formulation (16) provides less conservative values than (11), because of (9) $\|\hat{x}(t) - x_{sp}\|_p \leq \sqrt{\rho_s}$.

Reinforcement Learning Method: The motivation for combining RL with SR is to reduce oscillations and total steps, thereby enabling the drone to maintain higher speeds without the need for stopping and restarting at each setpoint. While the conventional control baseline mandates a fixed step size due to SR's safety constraints, our hybrid data-driven approach permits variable step sizes, facilitating a smoother trajectory. The RL method operates under the Lyapunov theory which guarantees safety. We adopt the Soft Actor-Critic algorithm, with both Actor and Critic having two fully connected hidden layers 256 hidden units each. The model takes the difference between the state estimation and the current setpoint $\hat{x}(t) - x_{sp}$ as input and learns the optimal α value to generate the next setpoint under the safety constraints. We train the model with 20,000 steps on an NVIDIA GeForce RTX 3090, and the training takes approximately an hour to converge. During training, the policy is stochastic whereas during evaluation is deterministic.

B. Simulation Results

Under the baseline method, the drone completes the task in around 116 s, and the drone's 3D trajectory is shown in Fig. 4. The baseline method produces α based on equation (16) and generates setpoints shown as red dots in the figure. Our objective is to minimize the time required to reach the goal while ensuring that the system satisfies the safety conditions.

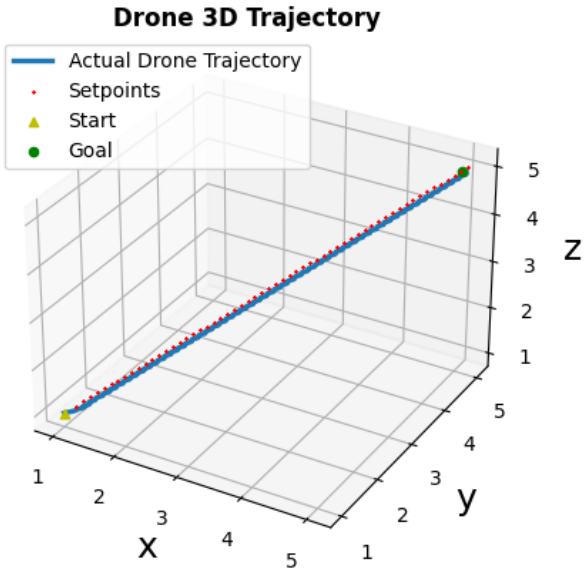

Fig. 4: Drone trajectory with the baseline method. The drone starts from (1,1,1) and ends at (5,5,5). The red dots are the waypoints produced by the baseline method. The blue line shows the actual trajectory of the drone.

Our RL method reduces the total time required to complete the task, achieving the goal within 106 s, as demonstrated in the last three plots of Fig. 5. In order to explain the benefit of the RL approach we consider the behavior of $\|x(t) - x_{sp}\|_P^2$ over time. This function shows how far the system is from the boundaries of the ellipsoids $\mathcal{E}(\rho_m, x_{sp})$ and $\mathcal{E}(\rho_s, x_{sp})$ that guarantee the safety of the system under SR. Fig. 6 shows the first 5 s of the baseline method along with the safety bounds.

The sequence (A, B, C, D, A) forms a complete SR cycle, as shown in Fig. 2. At local minima A, the setpoint is updated, and the system enters the MC mode from B to C. To ensure safety, B points should be less than ρ_m, so $x \in \mathcal{E}(\rho_m, x_{sp})$. At point C, the software refreshes to the same value as the previous cycle's point B. From C to A, the system is in SC mode, and x can be outside $\mathcal{E}(\rho_m, x_{sp})$ as in D, which should be kept small to avoid stability issues.

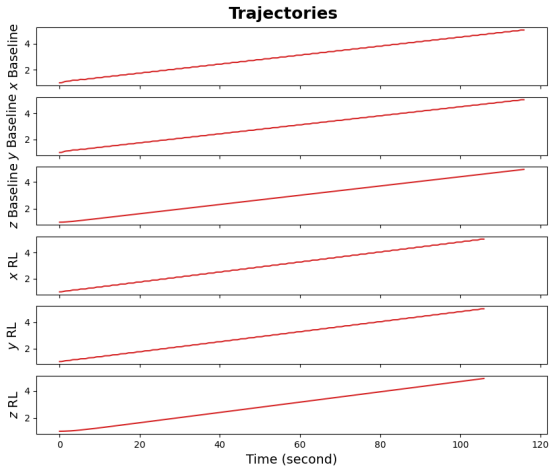

Fig. 5: Drone trajectories of baseline method (top three plots) and RL method (bottom three plots). The baseline method takes 116 seconds, and the RL method takes 106 seconds to complete the same task.

From Fig. 7, we compare $\|x(t) - x_{sp}\|_P^2$ value between the baseline method and the RL method. The baseline method has B points at around 0.0075 during MC, while the RL method pushes B points up to 0.009, reducing the gap with the upper safety bound.

VI. Conclusions

This work demonstrated the effectiveness of incorporating Reinforcement Learning and optimal control methods in the design of safe and efficient UAV navigation systems. Our approach optimizes a reward function that balances safety and performance and incorporates Software Rejuvenation (SR) protection mechanisms to mitigate cyber-attacks. Results from simulations of UAVs show that our approach improves the system's performance while respecting the safety bounds compared to traditional methods. This work contributes to the growing body of research that combines

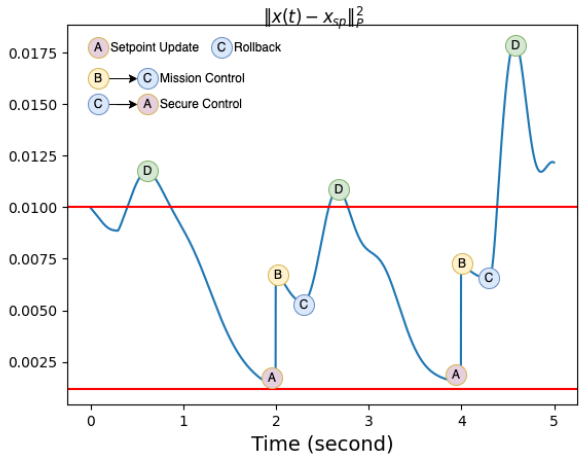

Fig. 6: Example of p-norm analysis of the actual state w.r.t the current x_{sp} selected from the first 5 s of our simulation with the baseline method. The red lines indicate the safety bounds $\rho_m = 0.01$ and $\rho_s = 0.0012$. (A, B, C, D, A) is a full checkpoint update cycle in Fig. 2.

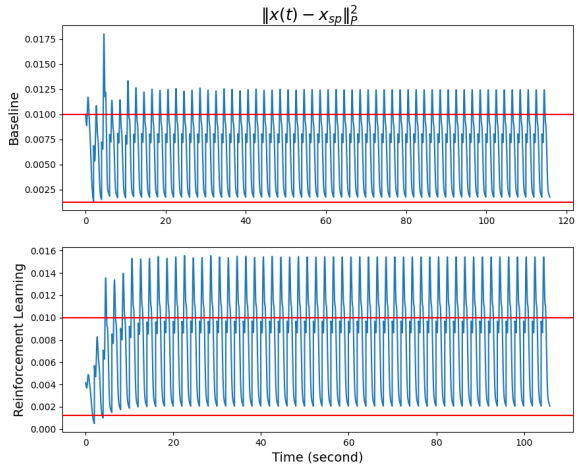

Fig. 7: p-norm analysis of the actual state w.r.t the current x_{sp}. The top figure is $\|x(t) - x_{sp}\|_P^2$ with the baseline method, and the bottom one is for the RL method. The RL method pushes the equilibrium points higher and closer to the upper safety bound than the baseline method.

control theory and Reinforcement Learning to address critical safety issues in cyber-physical systems. Future work can explore the application of our approach in other domains (e.g Bipedal walking) and investigate its impact on system performance and safety.

VII. Acknowledgements

The authors would like to express their gratitude to Tao Jin and Prof. Anthony Rowe from the Wireless Sensing and Embedded Systems Lab at Carnegie Mellon University for their invaluable support in providing computational resources for this research.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
[2] S. Levine, "Reinforcement learning and control as probabilistic inference: Tutorial and review," *arXiv preprint arXiv:1805.00909*, 2018.
[3] A. Platzer, *Logical Foundations of Cyber-Physical Systems*. Springer, 2018.
[4] Y. Ashibani and Q. H. Mahmoud, "Cyber physical systems security: Analysis, challenges and solutions," *Computers & Security*, vol. 68, pp. 81–97, 2017.
[5] S. M. Dibaji, M. Pirani, D. B. Flamholz, A. M. Annaswamy, K. H. Johansson, and A. Chakrabortty, "A systems and control perspective of CPS security," *Annual Reviews in Control*, vol. 47, pp. 394–411, 2019.
[6] A. Humayed, J. Lin, F. Li, and B. Luo, "Cyber-physical systems security—a survey," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1802–1831, 2017.
[7] M. A. Arroyo, M. T. I. Ziad, H. Kobayashi, J. Yang, and S. Sethumadhavan, "YOLO: Frequently resetting cyber-physical systems for security," in *Autonomous Systems: Sensors, Processing, and Security for Vehicles and Infrastructure 2019*, vol. 11009, International Society for Optics and Photonics, May 2019.
[8] F. Abdi, C.-Y. Chen, M. Hasan, S. Liu, S. Mohan, and M. Caccamo, "Preserving physical safety under cyber attacks," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6285–6300, 2018.
[9] R. Romagnoli, B. H. Krogh, D. de Niz, A. D. Hristozov, and B. Sinopoli, "Software rejuvenation for safe operation of cyber–physical systems in the presence of run-time cyberattacks," *IEEE Transactions on Control Systems Technology*, 2023.
[10] R. Romagnoli, B. H. Krogh, and B. Sinopoli, "Safety and liveness of software rejuvenation for secure tracking control," in *2019 18th European Control Conference (ECC)*, pp. 2215–2220, IEEE, 2019.
[11] R. Romagnoli, B. H. Krogh, and B. Sinopoli, "Design of software rejuvenation for cps security using invariant sets," in *2019 American Control Conference (ACC)*, pp. 3740–3745, IEEE, 2019.
[12] R. Romagnoli, B. H. Krogh, and B. Sinopoli, "Robust software rejuvenation for cps with state estimation and disturbances," in *2020 American Control Conference (ACC)*, pp. 1241–1246, IEEE, 2020.
[13] M. A. Taylor and L. F. Giraldo, "Data-driven design of a reference governor using deep reinforcement learning," in *2021 IEEE Conference on Control Technology and Applications (CCTA)*, pp. 956–961, IEEE, 2021.
[14] Y. Li, N. Li, H. E. Tseng, A. Girard, D. Filev, and I. Kolmanovsky, "Safe reinforcement learning using robust action governor," in *Learning for Dynamics and Control*, pp. 1093–1104, PMLR, 2021.
[15] K. Liu, N. Li, D. Rizzo, E. Garone, I. Kolmanovsky, and A. Girard, "Model-free learning to avoid constraint violations: An explicit reference governor approach," in *2019 American control conference (ACC)*, pp. 934–940, IEEE, 2019.
[16] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. USA: John Wiley & Sons, Inc., 1st ed., 1994.
[17] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, pp. 229–256, 1992.
[18] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, pp. 1861–1870, PMLR, 2018.
[19] E. T. Jaynes, "Information theory and statistical mechanics. ii," *Physical review*, vol. 108, no. 2, p. 171, 1957.
[20] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey, *et al.*, "Maximum entropy inverse reinforcement learning.," in *Aaai*, vol. 8, pp. 1433–1438, Chicago, IL, USA, 2008.
[21] T. Haarnoja, *Acquiring diverse robot skills via maximum entropy deep reinforcement learning*. University of California, Berkeley, 2018.