COB-2023-0126 DEEP REINFORCEMENT LEARNING-BASED ALGORITHM TO REPLACE THE PID FOR CONTROLLING SATELLITE AXIS POINTING

Gabriel Goes Aragão Santana
Ronan Arraes Jardim Chagas
National Institute for Space Research
gabriel.santana@inpe.br, ronan.arraes@inpe.br

Abstract. *Satellites must be able to control their attitude throughout their lifetimes to carry out their missions. Reaction wheels driven by electric motors are frequently employed to store and exchange momentum with the satellite body, providing a way to control the satellite attitude. The applied torque is governed by some control law, usually a proportional-derivative (PD) control. However, fine-tuning these parameters is often a laborious and time-consuming task. This work presents an alternative controller development by employing artificial intelligence techniques. In Reinforcement Learning (RL), an agent learns how to behave in an environment employing a scalar reward signal. Control coupled with RL promises some unique and powerful capabilities: improvements in performance in nominal cases, adaptability for varying conditions, fault tolerance in unusual scenarios, attitude control for future capture missions, and a general and powerful framework for controller design. Recent developments in the RL field led to novel and efficient algorithms able to tackle continuous control problems. Two of these algorithms, namely, Deep Deterministic Policy Gradient (DDPG) and Twin-Delayed DDPG (TD3) are presented and applied to the current problem. Simple heuristics that inject previous knowledge into the agent, which enhances learning, are discussed here. All implementations and training were run on a simulated environment using the parameters of the Amazonia-1 satellite, developed by INPE and successfully launched in 2021. We compared the new controller with the PD employed in Amazonia-1. The RL controller outperformed their PD counterparts in the tested scenarios, suggesting their viability and hinting at an extensive array of possible uses.*

Keywords: *attitude control, satellite, reinforcement learning, intelligent control.*

1. INTRODUCTION

The term attitude describes the orientation of a solid body with reference to another chosen reference frame. The attitude of space vehicles must be properly controlled, as it directly impacts the tasks that needed to be carried out. These include pointing instruments and antennae towards positions in Earth and space, accurate orbital maneuvers and proper thermal control. This control is usually realized by a combination of reaction wheels, thrusters or magnetic coils and is very often governed by some type of proportional-integral-derivative (PID) control law. Nevertheless, fine tuning the controller parameters is a very demanding task. Furthermore, this control law may not be optimum under conditions different from a nominal scenario, e.g. actuator failures, if parameters change.

Recent advances in Reinforcement Learning (RL), a field with strong ties with optimum control, may provide an alternative, fruitful approach to continuous control problems and thus to the attitude control in even more challenging scenarios. These may be too complex to be properly solved by means of traditional control techniques and include actuator failures, varying or unknown vehicle parameters and debris capture missions. The RL framework, with its emphasis on learning from experience and improving a given performance metric, seems suited to tackle such unique problems.

In spite of the potential, research on this topic remains limited. Allison *et al.* (2019) used the Proximal Policy Optimization (PPO) algorithm to control satellites with a mass ranging from 0.1 kg to 100 tons, outperforming traditonal methods. Hovell and Ulrich (2020) adapted deep RL techniques for satellite guidance, using a conventional control to solve the problem of tracking and docking. Marques (2021) employed 3 modern RL algorithms in attitude control based on reaction wheels in the Amazonia-1 satellite. Nominal results were not better than the onboard PD control, but were able to deal with reaction wheels failures. Tipaldi *et al.* (2022) provided an overview of RL applications for spacecraft, including guidance, navigation and control for spacecraft landing on celestial bodies, constellation orbital control and maneuver planing. Current limitations and challenges were also discussed. The RL methodology has also been successfully applied to the attitude control of Unmanned Aerial Vehicles (UAVs) (Bohn *et al.*, 2021) (Koch *et al.*, 2018). The results are remarkable given the complexity and nonlinearities present in the attitude dynamics of UAVs.

In this paper, using real data from the Amazonia-1 satellite, two modern RL algorithms are applied to the single axis attitude problem. Compared with the real PD controller aboard the satellite, the resulting control laws, defined as neural networks, are significantly faster: in some cases, the settling time is almost 400 seconds shorter. The results underscore the feasibility of this approach and the possible uses of RL to the broader attitude control problem.

This work is structured as follows: Section 2 presents the RL foundation and its modern developments, Section 3 applies the RL framework to the satellite attitude problem, Section 4 presents the results and finally Section 5 discusses these results and other considerations to future research.

2. REINFORCEMENT LEARNING

In the RL framework, an agent interacts with a given environment and must find an optimum way to behave so as to maximize its expected reward. More specifically, this interaction happens in a sequence of time steps. In each one of them, the agent takes some allowed action a, changing the state vector of the environment from s to s' and receiving a reward r. This process goes on until some terminal state is reached. The ultimate goal of RL is to maximize the sum of rewards for any given initial state.

RL problems typically involve two main challenges. The first one is the credit assignment problem, first identified by Minsky (1961). The agent must assign some credit to its past decisions, but since rewards are often delayed it is not immediately obvious how to so. The second one is the classical dimensionality curse, an ever-present problem in the field of machine learning. If the number of possible actions that the agent can take is too large, then the computational time required to find the optimal solution tends to grow quickly, making the problem intractable.

A more formal definition of the RL problem is given hereafter. It is assumed the state vector contains all the necessary information to completely and unambiguously describe the system. Furthermore, it is assumed the state is at all times fully observable to the agent, i.e. agent always knows the current state. At the discrete time step t, the state S_t is a certain state s. The agent then takes an action $A_t = a$, where a is an allowed action, moving the environment to the new state $S_{t+1} = s'$. The agent receives a scalar reward r_{t+1} where $r_{t+1} = r_{t+1}(S_t = s, A_t = a)$. It is worth noting that according to this definition the reward is only available at the next time step. All states belong to a finite set $\mathcal{S}$. Likewise, all allowable actions belong to the finite set $\mathcal{A}$.

The environment E defines how the state vector moves from one state to another. In the general case, E is stochastic and a probability should be assigned to a given transition. To elaborate further, defining the current sequence of states and actions τ as $\tau = (S_0 = s_0, A_0 = a_0, S_1 = s_1, A_1 = a_1, \ldots, S_t = s_t, A_t = a_t)$, then the probability that $S_{t+1} = s'$ should depend on all elements of τ. Nevertheless, the *Markov Property* often used in RL assumes that this value only depends on the last state-action pair:

$$P(S_{t+1} = s'|\tau) = P(S_{t+1} = s'|S_t = s, A_t = a) \tag{1}$$

Thus, E defines the probability such that $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$. In a similar manner, the reward function r can be defined as $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The agent-environment interaction starts in a certain initial state $S_0 = s_0$, distributed according to an initial distribution $\rho_0 : \mathcal{S} \to [0, 1]$, and continues until a terminal state s_T is reached at time T. Dropping the $S_t = s$ notation, the entire sequence $(s_0, a_0, s_1, a_1, \ldots, s_T)$ defines an episode.

The agent's behavior, that is, its action selection, is determined by its policy π, which defines the probability of taking a certain action a given that the environment's state is s. That probability is expressed by $\pi(a|s)$, so $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$. It is obvious that for any s, π must obey $\sum_{a' \in \mathcal{A}} \pi(a'|s) = 1$. If the policy μ is deterministic, only a single action is chosen with a nonzero probability and therefore the mapping can be written simply as $\mu : \mathcal{S} \to \mathcal{A}$, a mapping from states to actions. Unless otherwise stated, π will describe both deterministic and non-deterministic policies.

Now the goal of RL can be explicitly presented. The return G_t is defined as the sum of the rewards following the time-step t, that is:

$$G_t = r_{t+1}(s_t, a_t) + r_{t+2}(s_{t+1}, a_{t+1}) + \cdots + r_T(s_{T-1}, a_{T-1}) = \sum_{i=t}^{T-1} r_{i+1}(s_i, a_i) \tag{2}$$

Some environments, however, do not ever reach a terminal episode and their episodes go on indefinitely. A more general approach involves redefining G_t by using a discount rate γ, a number between 0 and 1, so that the return becomes:

$$G_t = r_{t+1}(s_t, a_t) + \gamma r_{t+2}(s_{t+1}, a_{t+1}) + \gamma^2 r_{t+3}(s_{t+2}, a_{t+2}), \cdots = \sum_{i=t}^{\infty} \gamma^{i-t} r_{i+1}(s_i, a_i) \tag{3}$$

The return G_0 now represents the sum of all discounted rewards in the episode. Since both the environment E and the policy π are in general stochastic, one expects G_0 to be a random variable. Thus it is the expected value of this term that we seek to maximize.

We can now offer a more rigorous definition of the RL problem as a Markov Decision Process (MDP), that is, an environment following Eq. (1), defined by the tuple $(\mathcal{S}, \mathcal{A}, P, r, \rho_0, \gamma)$, whose goal is finding a policy π in order to maximize $E[G_0]$.

2.1 Value-Functions

The delayed reward aspect present in RL makes it a challenging problem, as the agent may need to trade off between its short-term and long-term gains. A secondary signal is needed to guide the agent and bring all future rewards to the present moment. That is precisely the role played by the so-called value-functions. The state value-function $V_\pi(s)$ is defined as (Sutton and Barto, 2017):

$$V_\pi(s) = \mathbb{E}_{a\sim\pi}[G_0|S_0 = s] = \mathbb{E}_{a\sim\pi}\left[\sum_{i=0}^{\infty}\gamma^i r_{i+1}(s_i, a_i)|S_0 = s\right], \tag{4}$$

$V_\pi(s)$ is the total expected return in an episode, starting at the state s and following the policy π. A similar but slightly different concept is the action-value function $Q_\pi(s, a)$, given as:

$$Q_\pi(s, a) = \mathbb{E}_{a_t\sim\pi,t>0}[G_0|S_0 = s, A_0 = a] = \mathbb{E}_{a_t\sim\pi,t>0}\left[\sum_{i=0}^{\infty}\gamma^i r_{i+1}(s_i, a_i)|S_0 = s, A_0 = a\right] \tag{5}$$

Equation (5) defines the action-value as the total expected return, starting off in s, taking the initial action a, regardless what the current policy is, and then following the policy π thereafter. It is clear that we must have:

$$V_\pi(s) = \sum_{a\in\mathcal{A}}\pi(a|s)Q_\pi(s, a) \tag{6}$$

If $Q_\pi(s, a)$ is known for a given policy π, it is straightforward to obtain a new policy π' that is at least as good as π. That can be done by changing the action selection in a given state s to $\pi'(s) = \arg_a \max Q_\pi(s, a)$, i.e. being greedy in $Q_\pi(s, a)$. This forms the basis of the generalized policy iteration: first, the value-function of π is estimated (policy estimation) and then the policy is changed so as to increase $V_\pi(s)$, improving the return (policy improvement). This process is carried out until convergence to an optimum policy π^* (Sutton and Barto, 2017).

The Q-learning algorithm presented by Watkins (1989) offers a simple way to estimate Q_{π^*} in the tabular setting, when all $Q(s, a)$ values can be written in a finite table. The estimated $\hat{Q}(s, a)$ is updated every time a is selected in state s by:

$$\hat{Q}(s, a) \leftarrow \hat{Q}(s, a) + \alpha\left(r(s, a) + \gamma\max_{a'}\hat{Q}(s', a') - \hat{Q}(s, a)\right) \tag{7}$$

,where α is a small positive number, called the learning rate. Surprisingly, Watkins and Dayan (1992) proved that $\hat{Q}(s, a)$ converges to $Q_{\pi^*}(s, a)$ as long as all state-action pairs are selected an infinite number of times and a few other weaker conditions are satisfied. The optimum policy π^* itself can be easily obtained from $Q_{\pi^*}(s, a)$, since $\pi^*(s) = \arg_{a'}\max Q_{\pi^*}(s, a')$. If more than a single action maximizes the action-value for a given state, then a non-zero probability should be assigned only to these actions. The distribution itself is not important, as all obtained policies are equally optimum.´

In the RL parlance, Q-learning is said to be an *off-policy* algorithm, since the agent interacts with the environment by following a policy β that is different from the policy π that we want to learn something about. β is said to be the exploration policy while π is the target policy. In Q-learning, the exploration policy needs to be non-deterministic to ensure that all (s, a) pairs continue to be selected. This randomness allows the agent to explore different behaviors and is thus called *exploration*. The policy π, on the other hand, is often deterministic.

The counterpart of off-policy methods are *on-policy* methods, where $\beta = \pi$. The Sarsa algorithm is the on-policy variant of Q-learning and it updates $\hat{Q}(s, a)$ in a similar way to Eq. (7); the difference is that the parenthesis term becomes $r(s, a) + \gamma\hat{Q}(s', a') - \hat{Q}(s, a)$

2.2 Continuous Problems

The aforementioned development is valid for the tabular setting, in other words, when the sets $\mathcal{S}$ and $\mathcal{A}$ are finite. However, for most problems in the real world, especially in control, these sets are continuous. Even if they are finite, the sets might be too large to apply the techniques listed above. A new approach seems to be necessary. The problem is that we cannot explicitly write down the policy for all states and the action-value for all state-action pairs. Therefore, we need to introduce the idea of generalization by means of a parametric representation.

This idea gives rise to the *actor-critic* architecture, first introduced by Barto *et al.* (1983). The actor represents the policy $\pi(a|s)$ and is written as $\pi_\theta(a|s)$, where θ stands for its parameters. Likewise, the critic represents the action-value $Q(s, a)$, now as $Q^w(s, a)$, with w once again as the parameters. Many types of representations are conceivable, e.g. polynomial expression, linear combination, artificial neural networks (ANNs) and so on. The only requirement is that the chosen representation must be continuously differentiable in its parameters.

In recent years, modern, general-use approaches in RL have emerged with the capacity to tackle these types of problems in a large array of applications. The first breakthrough came with the *Deep Q-Network* (Mnih *et al.*, 2015) or simply DQN. Two key ideas underlie its success: the use of replay buffer, to speed up learning as suggested by Lin (1992), and the use of a target network for $Q(s, a)$. The first idea means storing transitions in the format (s, a, r, s') for later use in a batch, instead of using them once and then discarding, as it is done in Q-learning. The second one means having two ANNs to estimate $Q(s, a)$: one of them is constantly updated to reduce the loss function in the action-value while the other one is kept fixed to serve as the target and updated less frequently. Together they lead to the convergence of the action-value estimates. Despite its impressive performance in many Atari 2600 games, the original DQN formulation is only valid in environments with a finite number of actions.

An important theoretical result was presented by Silver *et al.* (2014) in the form of the deterministic policy gradient theorem, that is, the $\nabla_\theta G_0$ value. This result is noteworthy as it does not require any knowledge of the environment's dynamics. Though the gradient was known for stochastic policies (Sutton *et al.*, 1999), the deterministic case was regarded as more challenging. Building upon this theorem, the authors present their Deterministic Policy Gradient (DPG) algorithm. More specifically, for a deterministic policy μ:

$$\nabla_\theta G_0 = \mathbb{E}_{s\sim\rho^\mu}\left[\nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s,a)|_{a=\mu_\theta(s)}\right] \tag{8}$$

The core ideas in DQN and the deterministic policy gradient theorem are combined in the *Deep Deterministic Policy Gradient* (DDPG), as described by Lillicrap *et al.* (2015). DDPG represents the first general algorithm to continuous problems in the RL framework. As in DQN, a replay buffer and a target critic network are employed to update the action-value estimates. The actor is updated as in DPG, but now a target actor network is also present. Exploration is encouraged by adding noise - usually white noise - to the actor's output.

Fujimoto *et al.* (2018) implement a few changes to the DDPG and the resulting algorithm is named *Twin-Delayed DDPG* (TD3). The authors showed empirically that the $Q(s, a)$ estimates tend to be higher than their actual values, i.e. DDPG overestimates the action-value (positive bias). This may lead to suboptimal policies and thus degrade the performance. To address this issue, some modifications are carried out. The most important one, inspired by Double Q-learning (Hasselt, 2010) is the use of two critic networks - along their targets - for selection of the minimum value of $Q(s, a)$. Furthermore, the authors recommend the policy to be updated at a lower frequency than the critics.

3. ATTITUDE DYNAMICS AND CONTROL WITH RL

The goal of the present section is to apply the RL theory to the attitude control of a satellite, showing its implementation details. Real data from the Amazonia-1 satellite - developed by INPE and launched in 2021 - will be used to compare it with traditional methods. The general rotation dynamics in three dimensions of a rigid body leads to the Euler equations, a system of coupled differential equations describing the angular speeds around the principal axes of the body. Writing ω_j, I_j and T_j as respectively the angular velocity, moment of inertia and applied torque around the j-axis, one of the three principal axes, we have (Carrara, 2012):

$$I_x\dot{\omega}_x = T_x + (I_y - I_z)\omega_y\omega_z \tag{9}$$
$$I_y\dot{\omega}_y = T_y + (I_z - I_x)\omega_z\omega_x \tag{10}$$
$$I_z\dot{\omega}_z = T_z + (I_x - I_y)\omega_x\omega_y \tag{11}$$

The axes of a satellite are mostly arranged so as to be as close as possible to the principal axes. Furthermore, the angular velocities encountered in practice are very small, around 0.01 rad/s. Therefore, the equations can decoupled and their control treated separately for each axis. Since $\omega_j = \dot{\theta}_j$, the equations can be discretized around a chosen axis as:

$$\dot{\theta}(k+1) = \dot{\theta}(k) + \ddot{\theta}(k)\Delta t \tag{12}$$
$$\theta(k+1) = \theta(k) + \dot{\theta}(k)\Delta t + \frac{1}{2}\ddot{\theta}(k)(\Delta t)^2 \tag{13}$$

Δt is the time-step and k describes the variable at the time $k\Delta t$. The angular acceleration can be found by $\ddot{\theta}(k) = T(k)/I$. The control problem will be treated around a single axis as in Eq. (12) and Eq. (13) from now on; the full problem can be solved combining the solutions for the three axis. The z-axis was chosen, as it displays the largest moment of inertia. Its value is $I_z = 530.7\,kgm^2$.

At first glance, a natural representation for the state would be $s_t = [\theta(t), \dot{\theta}(t)]$. However, real applications commonly employ the quaternion attitude representation. A quaternion is made up of 4 elements and can represent a rotation of an angle θ around the unitary axis $\vec{u}$ as $q = [cos(\theta/2), \vec{u}\,sin(\theta/2)]$. The last 3 elements are called the vectorial part and are often used for the full attitude control. Therefore, to compare with PID control, the state is going to be defined as $s_t = [sin(\theta(t)/2), \dot{\theta}(t)]$. In practice, this state must be determined by measurements from onboard sensors, which is the responsibility of the Attitude Determination Subsystem (ADS), that provides its estimates to the Attitude Control Subsystem (ACS). The environment can be assumed to be fully observable.

In the Amazonia-1 satellite, reaction wheels are used to store and exchange momentum with the vehicle. The wheels are driven by electrical motors, that are limited in terms of the maximum torque T_{max} that they can apply. This value is $0.075\,Nm$ for Amazonia-1, that is, the torque $T(t)$ must be in the $[-0.075, 0.075]\,Nm$ interval. Since the actor is represented by an ANN, it was found more practical to limit its output a_t to the interval $[-1, 1]$ and include T_{max} directly in the equations. This arrangement, together with the environment dynamics, is shown in Fig. 1.

A PD controller is used in Amazonia-1. The torque around the z-axis applied to the satellite is given by:

$$T(t) = -k_p sin(\theta(t)/2) - k_d\dot{\theta}(t) = -1.019\,sin(\theta(t)/2) - 42.21\,\dot{\theta}(t) \tag{14}$$

The form of the reward r can, at first, be chosen freely, as long as it rewards the expected behavior. In the current case, this means bringing the satellite to standstill at $\theta = 0$. A simple way to do so is to choose:

$$r_{t+1}(s_t, a_t) = r_{t+1}(s_t) = -1.0\,sin^2(\theta_t/2) \tag{15}$$

Equation (15) has a maximum at $\theta = 0$. Notice that the angular velocity term is not included, nor the applied action. The idea was to keep the reward as simple as possible, while conveying the desired outcome.

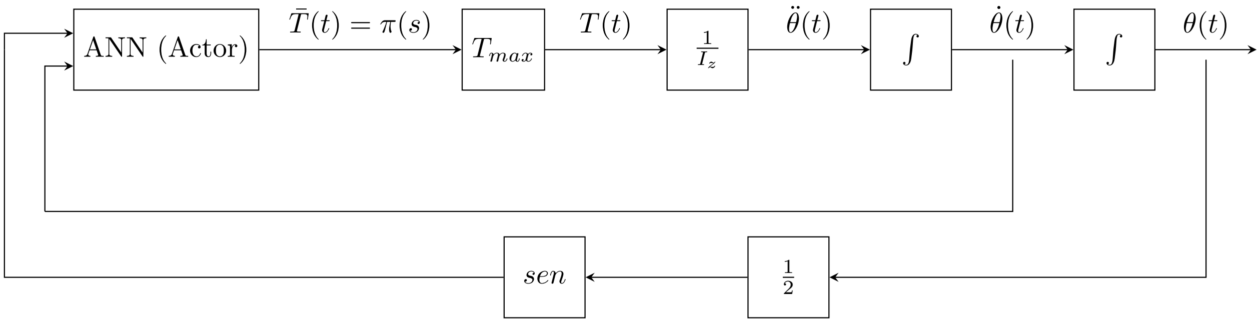


Figure 1: Block diagram for the proposed attitude control

The episodes start randomly in a state with an angle uniformly distributed in $[-\pi, \pi]\,rad$ and angular velocity uniformly distributed in $[-0.025, 0.025]\,rad/s$. The interaction with the environment goes until the norm of s_t is less than 0.0001 or t reaches 4000. The time-step Δt equals 1 second.

The algorithms were ran in the Julia programming language, using the implementations available in the *ReinforcementLearning* package (Tian and other contributors, 2020). The hyper-parameters, common to all simulations, are shown in Tab. (1). As suggested by Marques (2021), the bias term was not used in the actor network, to guarantee that $a_t([0, 0]) = 0$. Furthermore, the selected activation function for the actor was the hyperbolic tangent as it ensures the odd property observed in Eq. (14), i.e. $tanh(x) = -tanh(-x)$. Altogether, the actor contains 1120 parameters in the form of its weights, while the critic has 17153. Although the choice of number of layers and neurons is somewhat arbitrary, they were chosen so that the critic would have more parameters available and thus be to able achieve a better approximation of $Q_\pi(s, a)$, since its estimates are key to updating the policy.

Table 1: Hyper-parameters used in the simulations

Hyper-parameter	**Value**
Discount rate	0.99
Number of hidden layers (actor)	2
Number of neurons per layer (actor)	32
Activation function (actor)	tanh
Number of hidden layers (critic)	2
Number of neurons per layer (critic)	128
Activation function (critic)	ReLU
Replay buffer size	10000
Batch size	32
Number of episodes	100000
Optimizer	ADAM

4. RESULTS

For the sake of reference, Fig. 2 shows the torque applied by Amazonia PD Controller, following Eq. (14) and considering actuator limits. The colors in the heatmap show the fraction of the nominal torque T_{max} applied as a function of angle and angular speed of the satellite.

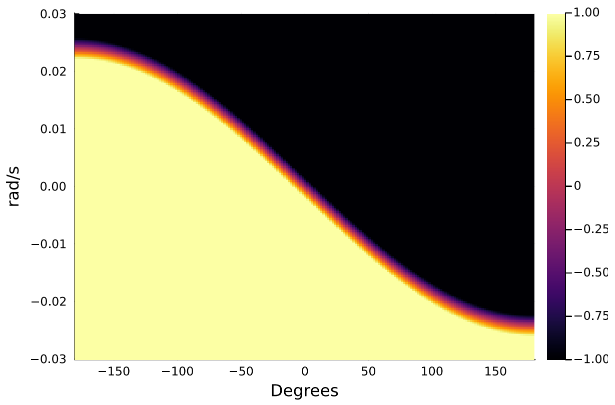


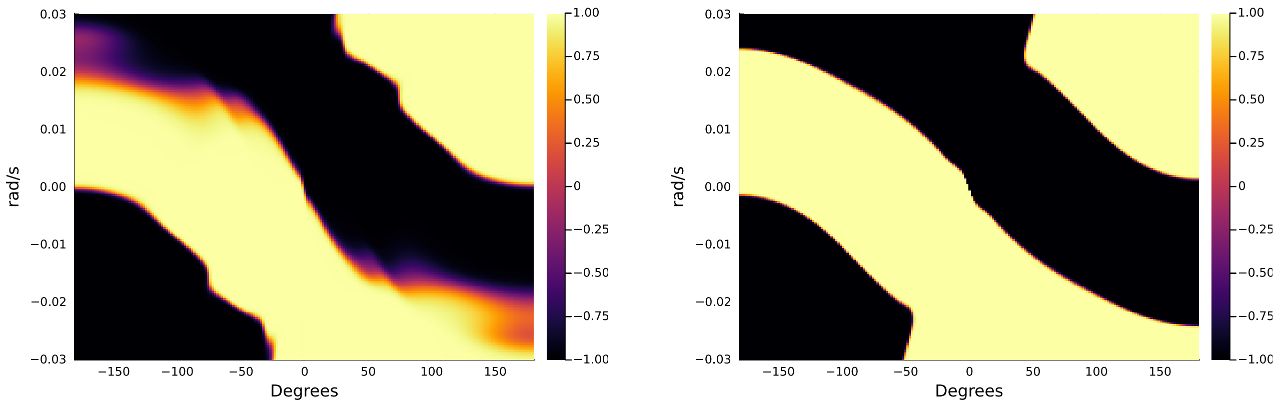
Figure 2: Fraction of maximum torque applied by Amazonia PD Controller

Figure 3 shows the deterministic policies obtained by DDPG and TD3 algorithms, that is, the mapping $\mu : [\theta, \omega] \to \bar{T}$. Note that the defined state is $s_t = [sin(\theta/2), \omega]$, but since $sin(\theta/2)$ is monotonic in $[-\pi, \pi]$ the angle is used for the plots. Despite some slight differences, two striking and common features can be observed. First, the obtained torque is an odd function, that is, $\bar{T}(\theta, \omega) = -\bar{T}(-\theta, -\omega)$. That is the result of using no bias term and adopting the $tanh$ activation function. Second, the two saturation regions are present as in the PD controller, but they are now accompanied by two other saturation regions with opposite signs in the upper right and lower left corners. Understanding this unconventional behavior requires some examples to compare their performance.

Figure 4 provides some insight into this question. The RL agents are able to learn that is helpful having a (relatively) high positive angular speed when the angle is close to but below π. Therefore, they speed up the satellite, that comes to standstill from a counterclockwise direction. On the other hand, the PD controller first tries to slow down the satellite and then brings it to standtill from a clockwise direction. The end result is that the RL controllers are faster. Defining the settling time as the time it takes the satellite to reach a state such that $|s_t| < 0.0001$ - roughly one hundredth of a degree -, then PD has a time of 738 s, DDPG 349 s and TD3 464 s.

Figure 5 and Figure 6 show other initial conditions. Again, the RL agents are faster; DDPG in particular seems to be the faster than TD3, but it displays a slight overshoot, which can be seen in Fig. (6). Figure 7 presents the settling time, starting from the state $s_0 = [sin(\theta_0/2), 0]$. Even though DDPG times are significantly shorter, it is difficult to argue that it outperforms TD3, as the torque applied by latter seems to be somewhat smoother than the former. Either way, this proves that the reward, as given by Eq. (15) was able to properly convoy the desired goal, as both RL controllers are significantly faster than the PD controller.

The expected returns for all the agents are shown in Tab. (2). These results were obtained after running each agent for a million episodes, each starting in a random state, as defined previously. Since the rewards are all negative, so is the



(a) DDPG (b) TD3

Figure 3: Fraction of maximum torque applied by RL controllers

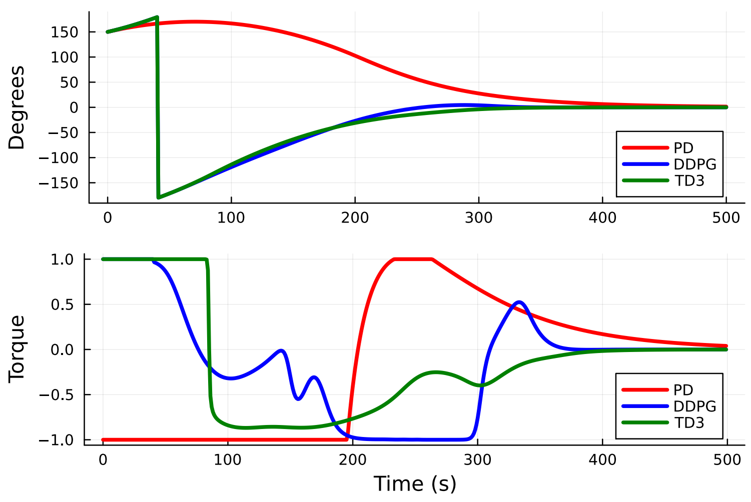


Figure 4: PD and RL controllers comparison. Initial conditions are $\theta_0 = 150°$ and $\omega_0 = 0.02\,rad/s$

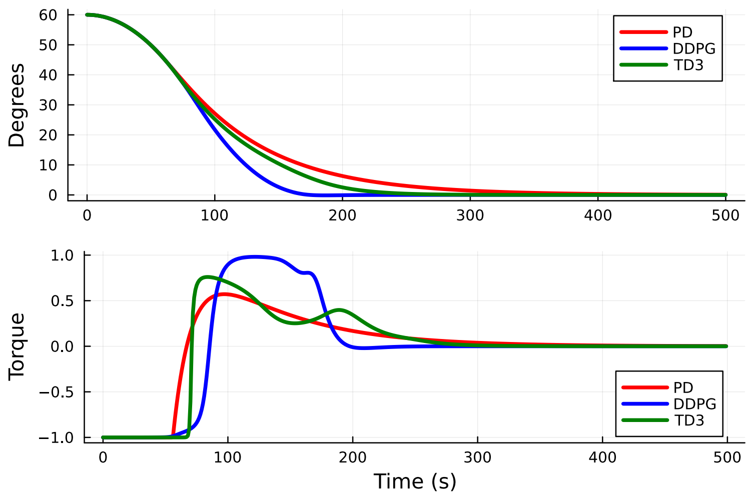


Figure 5: PD and RL controllers comparison. Initial conditions are $\theta_0 = 60°$ and $\omega_0 = 0.0\,rad/s$

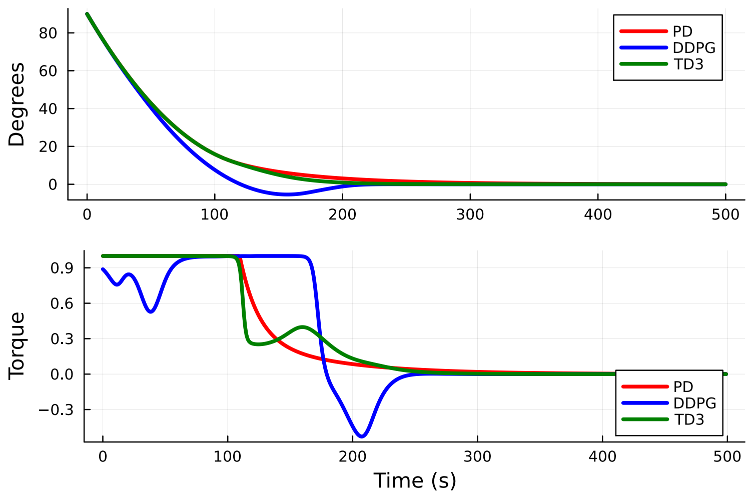


Figure 6: PD and RL controllers comparison. Initial conditions are $\theta_0 = 90°$ and $\omega_0 = -0.02\,rad/s$

return. The negative of the return can be seen as a cost that we want to minimize. In that case, DDPG's cost is about 5% smaller and TD3's 5.4%. Their similar performance can be seen in Fig. (8), which shows the $V_\pi(s)$ and thus the return starting in a given state s and following the agent's policy.

It must be emphasized that the PD controller was specially tuned to the Amazonia satellite. Therefore, the fact that the

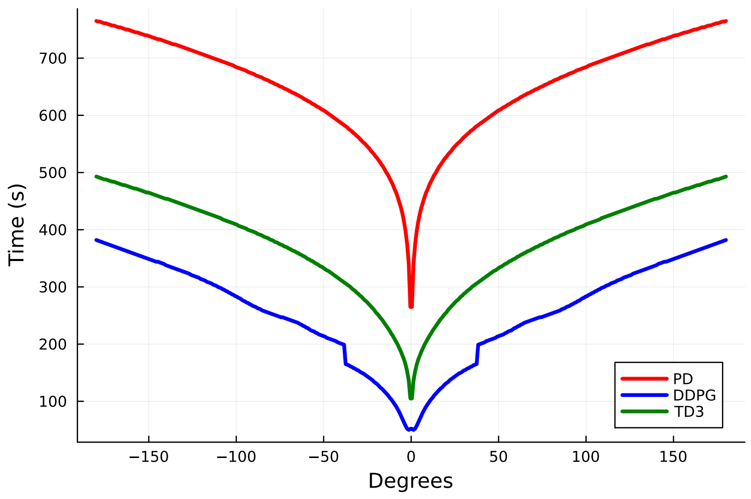
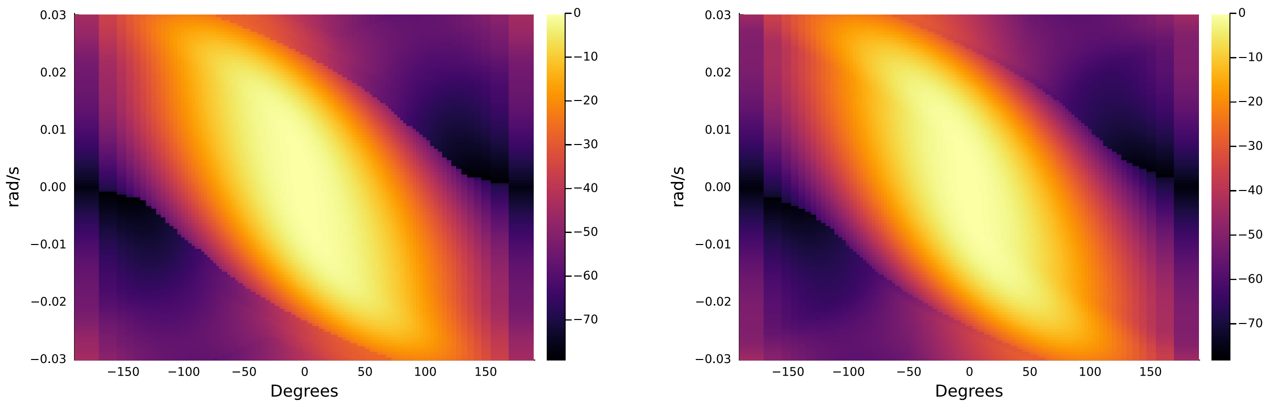


Figure 7: Settling time given zero initial angular velocity



(a) DDPG (b) TD3

Figure 8: Real $V_\pi(s)$ for the RL agent policies

RL controllers were able to outperform it, even by a relatively small margin, is a testament to the capabilities of modern RL algorithms. Although the specific choice for the reward $r(s, a)$ is arbitrary, the authors are conviced the DDPG and TD3 controllers are very close to the optimum control given the environment dynamics and constraints.

Table 2: Expected Return

Agent	$\mathbb{E}[G_0]$
PD Control	-36.56
DDPG	-34.74
TD3	-34.58

Lastly, it is important to consider the critic performance, as it greatly influences the policy update, as given by Eq. (8). Figure 9 and Figure 10 compare the predictions of $V_\pi(s)$ from the critics with the real values. Even though strictly speaking the networks estimate $Q_\pi(s, a)$, the state value can be quickly obtained, since $V_\pi(s) = Q(s, \pi(s))$. For TD3, in particular, the minimum value was chosen. As one could expect from the theory, DDPG overestimates the real values while TD3 underestimates them.

5. CONCLUSION

The use of state-of-the-art RL algorithms for attitude control was investigated. The settling time for the RL controllers was significantly lower than the conventional PD controller, even though the latter was specifically fine tuned for the Amazonia-1 satellite. In fact, they were able to find a non-trivial behavior that greatly improved the performance.

Still, some key decisions probably helped to speed up the learning and thus led to a better performance. The use of $tanh$ activation function and the absence of a bias in the actor effectively halve the problem: if in a given state s the agent should do a, then in $-s$ it does $-a$.

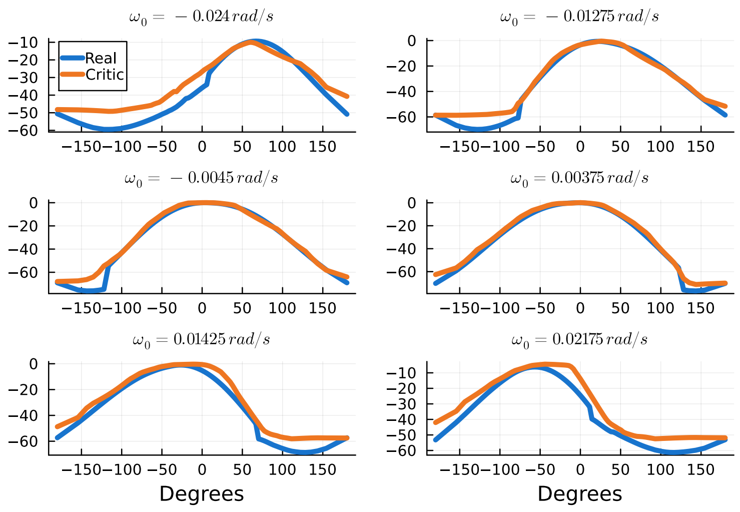


Figure 9: DDPG critic estimation of $V_\pi(s)$ compared to the real values

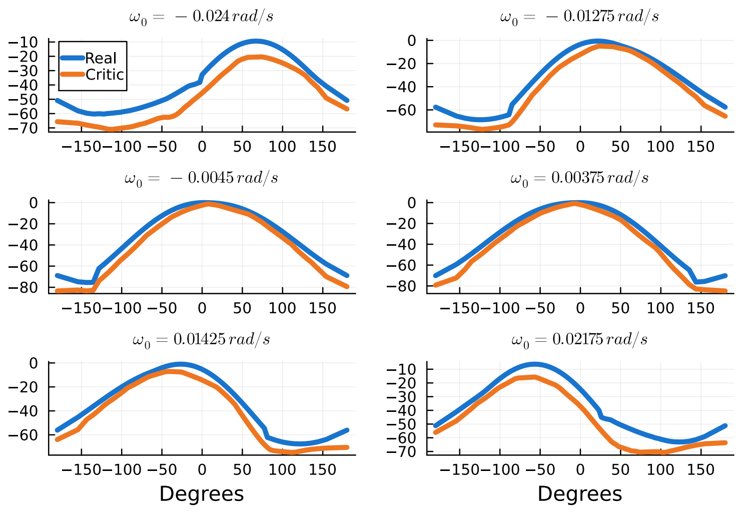


Figure 10: TD3 critic estimation of $V_\pi(s)$ compared to the real values

The precise form of the reward is by no means a trivial task. It should guide the agent to the goal, but should not dictate how it behaves in order to achieve it. The agent may find a way to maximize the return, but the obtained policy may not translate well into the real task. Many other choices, beyond the one used here, are conceivable.

The RL framework is powerful and general enough to be applied successfully to a large class of problems. As shown in this work, this also includes the attitude control. The extension to the full attitude problem is straightforward. The algorithms can be directly tested on the three dimensional case or the control for each individual axis can be simply combined. Many other challenging problems in satellite control which cannot be satisfactory addressed by conventional techniques might be suited for RL methods. These include problems with varying or unknown parameters, debris capture and actuator failures.

As the processing power available in satellites and space vehicles expands, control strategies as the one presented in this work become increasingly more feasible and thus might soon find real applications in future missions.

6. REFERENCES

Allison, J., West, M., Ghosh, A. and Vedant, F., 2019. "Reinforcement learning for spacecraft attitude control".

Barto, A.G., Sutton, R.S. and Anderson, C.W., 1983. "Neuronlike adaptive elements that can solve difficult learning control problems". *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-13, No. 5, pp. 834–846. doi: 10.1109/TSMC.1983.6313077.

Bohn, E., Coates, E.M., Reinhardt, D. and Johansen, T.A., 2021. "Data-efficient deep reinforcement learning for attitude control of fixed-wing uavs: Field experiments".

Carrara, V., 2012. *Cinemática e Dinâmica de Satélites Artificias*. INPE, São José dos Campos. URL `http://urlib.net/8JMKD3MGP7W/3B96GD8`.

Fujimoto, S., van Hoof, H. and Meger, D., 2018. "Addressing function approximation error in actor-critic methods". *CoRR*, Vol. abs/1802.09477. URL `http://arxiv.org/abs/1802.09477`.

Hasselt, H.v., 2010. "Double q-learning". In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 2*. Curran Associates Inc., Red Hook, NY, USA, p. 2613–2621.

Hovell, K. and Ulrich, S., 2020. *On Deep Reinforcement Learning for Spacecraft Guidance*. doi:10.2514/6.2020-1600. URL `https://arc.aiaa.org/doi/abs/10.2514/6.2020-1600`.

Koch, W., Mancuso, R., West, R. and Bestavros, A., 2018. "Reinforcement learning for UAV attitude control". *CoRR*, Vol. abs/1804.04154. URL `http://arxiv.org/abs/1804.04154`.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. and Wierstra, D., 2015. "Continuous control with deep reinforcement learning".

Lin, L.J., 1992. "Self-improving reactive agents based on reinforcement learning, planning and teaching". *Mach. Learn.*, Vol. 8, No. 3–4, p. 293–321. ISSN 0885-6125. doi:10.1007/BF00992699. URL `https://doi.org/10.1007/BF00992699`.

Marques, W.J.S., 2021. "Intelligent attitude control of satelittes via deep reinforcement learning".

Minsky, M., 1961. "Steps toward artificial intelligence". *Proceedings of the IRE*, Vol. 49, No. 1, pp. 8–30.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M.A., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S. and Hassabis, D., 2015. "Human-level control through deep reinforcement learning". *Nature*, Vol. 518, pp. 529–533.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D. and Riedmiller, M., 2014. "Deterministic policy gradient algorithms". In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*. JMLR.org, ICML'14, p. I–387–I–395.

Sutton, R.S. and Barto, A.G., 2017. *Reinforcement Learning: An Introduction*. The MIT Press, Massachusetts, 2nd edition.

Sutton, R.S., McAllester, D., Singh, S. and Mansour, Y., 1999. "Policy gradient methods for reinforcement learning with function approximation". In *Proceedings of the 12th International Conference on Neural Information Processing Systems*. MIT Press, Cambridge, MA, USA, NIPS'99, p. 1057–1063.

Tian, J. and other contributors, 2020. "Reinforcementlearning.jl: A reinforcement learning package for the julia programming language". URL `https://github.com/JuliaReinforcementLearning/ReinforcementLearning.jl`.

Tipaldi, M., Iervolino, R. and Massenio, P.R., 2022. "Reinforcement learning in spacecraft control applications: Advances, prospects, and challenges". *Annual Reviews in Control*, Vol. 54, pp. 1–23. ISSN 1367-5788. doi:https://doi.org/10.1016/j.arcontrol.2022.07.004. URL `https://www.sciencedirect.com/science/article/pii/S136757882200089X`.

Watkins, C., 1989. "Learning from delayed rewards".

Watkins, C. and Dayan, P., 1992. "Q-learning". *Machine Learning*, Vol. 8, pp. 279–292.

7. RESPONSIBILITY NOTICE