11º Workshop em Engenharia e Tecnologia Espaciais
18, 19 e 20 de agosto de 2020

Reinforcement learning applied to the control of the pitch-axis of a satellite

Wilson de Sá Marques [1], **Ronan Arraes Chagas** [2]

[1] Instituto Nacional de Pesquisas Espaciais, São José dos Campos, SP, Brasil
Aluno de Mestrado do Curso de Mecânica Espacial e Controle - CMC.

[2]Instituto Nacional de Pesquisas Espaciais, São José dos Campos, SP, Brasil
Pesquisador da Divisão de Sistemas Espaciais - DSE

wilson.marques@inpe.br

Abstract. *In this paper, we assess the control of the pitch-axis of a satellite with the use of reinforcement learning techniques. The main goal of this work is to show the feasibility of this approach and to compare its performance with a traditional method of control design from the satellite literature. The state-of-the-art Twin Delayed Deep Deterministic Policy Gradient (TD3) reinforcement learning algorithm was used. Results show that the attained optimal policy can have similar performance to a classical PD control law, and it is able to adapt its output accordingly in order to control a different inertia that the one it has been trained with. In fact, the same policy trained to control a nanosatellite inertia was capable of controlling the inertia of a medium-sized satellite, assuming different initial conditions. While the PD law was also capable of controlling a different inertia from the one it has been originally tuned for, its performance dropped considerably, and it presented a high-frequency control signal, very difficult to be implemented by a typical satellite actuator.*

Keywords: Satellite; Attitude control; Reinforcement learning;

1. Introduction

Most modern satellites are equipped some sort of active attitude control system for pointing requirements. For instance, the solar panels must be oriented towards the sun for power generation, the antennas usually have to be directed to a specific point in the surface of the Earth. A popular choice for a satellite controller is a type of proportional-derivative law based on quaternion feedback [Markley and Crassidis 2014]. Even though it is widely used, this formulation is not suitable for handling uncertainty in the inertia parameters.

In situations where it is expected to occur changes in the spacecraft attitude and mass parameters, adaptive control schemes are required [Scarritt 2008]. Traditional methods of nonlinear adaptive control design are usually hand-crafted to solve a particular task and require significant engineering effort. For example, in applications involving on-orbit servicing, assembly, debris capture [Flores-Abad et al. 2014]. Such complex scenarios are very difficult to model and predict, which makes the control design more challenging. As shown in [Ma et al. 2018], reinforcement learning can be applied in these scenarios, enabling a control strategy to be learned rather than being specific designed.

[Koch et al. 2019] shows an interesting implementation of reinforcement learning for the attitude control of an Unmanned Aerial Vehicle (UAV), where the author compares different state-of-the-art reinforcement learning algorithms. Nonetheless, reinforcement leaning is having a far-reaching impact in robotics, with an abundance of success application cases. In [Hwangbo et al. 2019], a legged robot is controlled entirely with a reinforcement learning trained policy, and in [Andrychowicz et al. 2020] an agent was able to learn hand manipulation, which is a very daunting task, from the ground up. In both of these works, the policy was trained in simulation and then later deployed to hardware, showing a similar performance, despite the known difficulties related to the simulation-to-reality-gap [Kober et al. 2013]. Applications of reinforcement learning for spacececrafts, however, are still limited. Although, some very recent publications show excellent results.

For example, in [Hovell and Ulrich 2020], the authors choose to use a hybrid approach for a mission of redezvouz and docking, with a reinforcement learning trained neural network issuing the higher level commands in the guidance module and a conventional PD law as the lower level attitude controller. The work contains simulation and real experiments. The results are very promising and the authors claim that this approach would facilitate the certification procedure and the adoption of such novel control law by the scientific community.

Also, it is import to consider the great amount of development that has been occurring in nanosatellite technology, as it arises as a viable and cheap alternative to test novel controller concepts [Carrara et al. 2017]. As a matter of fact, in case very good results are obtained in simulation, a cubesat mission could possibly be requested to validate such control law.

Encouraged by the difficulties in the control design for complex tasks, and the recent promising results that reinforcement learning is presenting in various applications, this paper studies the use of reinforcement learning for the low level attitude controller of a satellite.

Results have shown that the neural network controller is capable of controlling an inertia more than 250 times larger than the one it has been trained with. However, it was incapable of completely removing the steady-state error. An alternative reward function will have to be design in future developments of this work to tackle this issue.

2. Methodology

2.1. Problem Statement

In order to obtain insight about the complete three-axis attitude control problem, it is useful to start with a simplified version of the problem. Hence, in this paper a simple single-axis attitude control system is designed.

2.1.1. Kinematics and Dynamics Models

The kinematics will be given by a quaternion parametrization. Quaternions are a popular redundant attitude coordinate set which is singularity free in its attitude representation. The quaternion $q = (q_1, q_2, q_3, q_4)$ is defined as

$q_1 = e_1 sin(\phi/2)$

$q_2 = e_2 sin(\phi/2)$

$q_3 = e_3 sin(\phi/2)$

$q_4 = cos(\phi/2)$

where $e = [e_1, e_2, e_3]$ and ϕ are the Euler vector and angle, respectively.

The differential kinematic equation in the quaternion parametrization is given by:

$$\dot{q} = \frac{1}{2} W q, \tag{1}$$

where

$W = \begin{bmatrix} -[\Omega_B^{B/I} \times] & \Omega_B^{B/I} \\ -(\Omega_B^{B/I})^T & 0 \end{bmatrix}$, $[\Omega_B^{B/I} \times]$ is a skew-symmetric matrix, and $\Omega_B^{B/I}$ is the angular velocity vector of the Body frame *B* with respect to the Inertial frame *I*, represented in the *B* frame.

For this simplified case, the dynamics is given by the Euler equation:

$$J\ddot{\theta} = u, \tag{2}$$

where J is the satellite inertia, θ is the pitch angle, and u is the applied torque.

Figure 1 illustrates the definition of the pitch angle, denoted by θ.

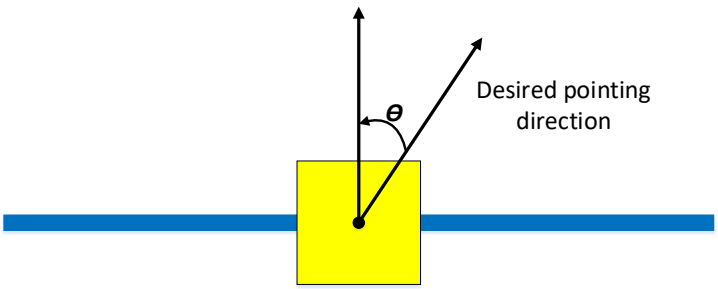

Figure 1: A rigid satellite pointing geometry.

Different types of actuators are used in satellites. Reaction wheels and control moment gyros work by the principle of conservation of angular momentum. They produce an internal torque by means of an rotating mass which exchange moment with the satellite structure. On the other hand, thruster and magnetorque coils provide an external torque. However, for simplification, in this paper the dynamics of the actuator are neglected, so it is assumed that the torque command u is applied directly to the satellite body, but it is saturated within the limits shown in Table 2.

2.2. Deep Reinforcement Learning

Reinforcement learning is a branch of artificial intelligence which is concerned with the dynamical interaction of an agent and its environment. The idea of reinforcement learning is to

have an agent that chooses actions with the objective of maximizing the rewards it accumulates over a period of time. At each timestep, a scalar reward is given to the agent in response to the action performed at the previous timestep, and it indicates the quality of that action [Sutton and Barto 2018].

Through trial-and-error, the agent attempts to learn a policy that maps inputs to the actions that maximize the accumulated expected reward. Therefore, by selecting an appropriate reward function, complex behaviours can be learned by the agent without being explicitly programmed. That is the main appeal of Reinforcement learning [Hovell and Ulrich 2020].

When neural networks, which are universal function approximators, are used to represent the policy, the word 'Deep' is added to the name, so we have 'Deep Reinforcement Learning'. In the context of this paper, the agent is the controller and the environment includes the satellite dynamics, the reward function and any external disturbance to the model. After training, the learned neural network receives the states as inputs and returns the torque commands to be applied to the satellite body.

The theory of deep reinforcement learning is rich, and there are a variety of algorithms and techniques available to choose from, each one being more suitable for a specific category of problems. The algorithm used in this work belongs to the family of actor-critic methods, since we are dealing with a problem with continuous state and action spaces. In this method, there is a policy neural network, $\pi_\theta(s)$, also referred to as the actor, that maps states to actions, and a value neural network, Q, also referred to as the critic, that maps a state-action pair into a probability distribution of the expected rewards to be received from taking action a in state s and continuing to follow the policy until the end of the episode. The referred reward function is defined as the expectation over the state and actions, with tunable parameters θ, is given by:

$$J(\theta) = \mathbf{E}\{Q_\phi(s, \pi_\theta(s))\}, \tag{3}$$

where $J(\theta)$ is the expected reward from the given state, as a function of the parameters θ, and E denotes the expectation. The goal of reinforcement learning is to find a policy π that maximizes $J(\theta)$.

Figure 2 illustrates the interconnection between the actor and critic neural networks with the environment.

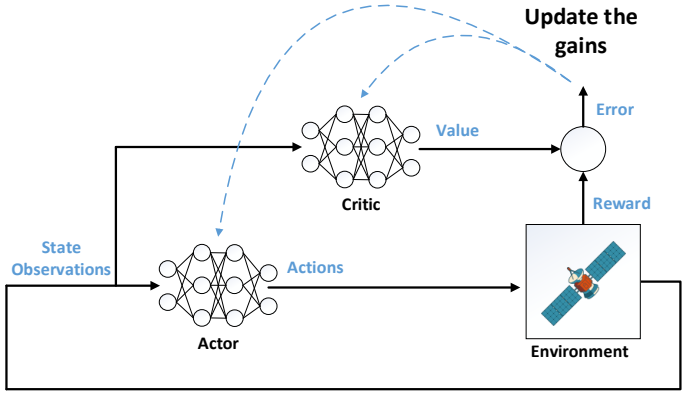

Figure 2: Actor and critic neural networks.

2.2.1. Twin Delayed DDPG (TD3)

The Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm [Fujimoto et al. 2018] belongs to the family of actor-critic methods. It was chosen for this problem because it works on environments with continuous state and actions spaces, and it has a deterministic output. The TD3 is an improvement of the original Deep Deterministic Policy Gradient (DDPG) algorithm [Lillicrap et al. 2015].

The TD3 algorithm is presented in pseudocode below in Algorithm 1 [Fujimoto et al. 2018].

Algorithm 1 TD3

Initialize critic networks Q_{θ_1}, Q_{θ_2}, and actor network π_ϕ
with random parameters θ_1, θ_2, ϕ
Initialize target networks $\theta'_1 \leftarrow \theta_1$, $\theta'_2 \leftarrow \theta_2$, $\phi' \leftarrow \phi$
Initialize replay buffer $\mathcal{B}$
for $t = 1$ **to** T **do**
  Select action with exploration noise $a \sim \pi_\phi(s) + \epsilon$,
  $\epsilon \sim \mathcal{N}(0, \sigma)$ and observe reward r and new state s'
  Store transition tuple (s, a, r, s') in $\mathcal{B}$

  Sample mini-batch of N transitions (s, a, r, s') from $\mathcal{B}$
  $\tilde{a} \leftarrow \pi_{\phi'}(s') + \epsilon, \quad \epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$
  $y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta'_i}(s', \tilde{a})$
  Update critics $\theta_i \leftarrow \text{argmin}_{\theta_i} N^{-1} \sum (y - Q_{\theta_i}(s, a))^2$
  if t mod d **then**
    Update ϕ by the deterministic policy gradient:
    $\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$
    Update target networks:
    $\theta'_i \leftarrow \tau\theta_i + (1 - \tau)\theta'_i$
    $\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$
  end if
end for

2.3. Reward Function

In this section, we discuss the reward function used to incentivize the desired behaviour. A properly engineered reward function is crucial for the success of reinforcement learning. Unfortunately, there is no recipe for designing a cost function, since it is problem-specific. Actually, it is through the reward function that the designer can inject domain-specific knowledge into the agent.

Considering the satellite pointing problem, we know that when the body and inertial frames are aligned, the angle θ becomes zero, as shown in 1, or equivalently, the rotation quaternion becomes $q = [1, 0, 0, 0]$. For that reason, we include the term $(1 - q_4)^2$ in the cost. In order to maintain the correct orientation, the angular velocity ω needs to go to zero, so we add it to the reward function. Finally, since we do not want an excessive control action, the control signal u

is also included.

$$J = k_1 \times (1 - q_4)^2 + k_2 \times \omega^2 + k_3 \times u^2 \tag{4}$$

where k_1, k_2, k_3, are scalar gains.

3. Results and Discussion

In order to train the neural network a modelling environment was set up in python using the OpenAI's Gym environment standard [Brockman et al. 2016], which is a typical interface for interacting with reinforcement learning algorithms. The developed environment contains the satellite kinematic and dynamical models. This environment also contains the reward function. As explained in Section 2.3, this reward function was created inspired in a classical PID control law with the goal of obtaining an appropriate performance, the final values of the gains used in the cost function were $k_1 = 100$, $k_2 = 0.1$, $k_3 = 0.01$.

The TD3 algorithm used in this work is the implementation from the Stable Baselines [Hill et al. 2018], which is an open source library for reinforcement learning algorithms. For these first experiments, the default Hyperparameters have been used. Table 1 shows the respective Hyperparameters values used for the results in this paper.

Table 1. Hyperparameters.

Hyperparameter	**Value**
Optimizer	Adam
Feature extraction	Multi-layer perceptron
Number of hidden layers	2
Number of neurons in each hidden layer	64
Batch size	128
Replay Buffer size	10^5
Action Noise	$N(0, \sigma = 0.1)$

Table 2 shows both satellite configurations used in the experiments performed for this paper. As the reader can see, both have very different parameters. One of them is a Nanosatellite with a inertia around the z-axis of just $2kg.m^2$ while the other has the inertia of the Amazonia-1 satellite, which is a medium-sized satellite developed at INPE (Brazilian National Institute for Space Research).

Table 2. Satellite Physical Parameters.

Parameter	**Symbol**	**Nanosatellite**	**Medium-sized satellite (Amazonia-1**
Satellite inertia moment (Kg.m^2)	J	2.0	530.7
Maximum torque (Nm)	T	0.0006	0.075

Figure 3 shows the ideal PD controller response as it was tuned for the inertia of the Amazonia-1 satellite, as presented in Table 2.

However, in Figure 4 we see that if the initial pointing error is increased to 180°, which would be the worst case scenario, and the gains remain unchanged, the performance deteriorates considerably. In effect, there is a large overshoot greater than 50 degrees, but note that there is no steady-state error.

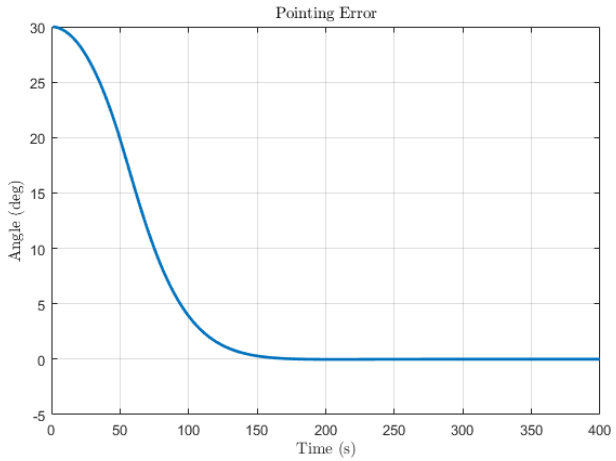

Figure 3: Pointing error for the PD controller tuned for the inertia of 530.7 $kg.m^2$ and $\theta = 30$.

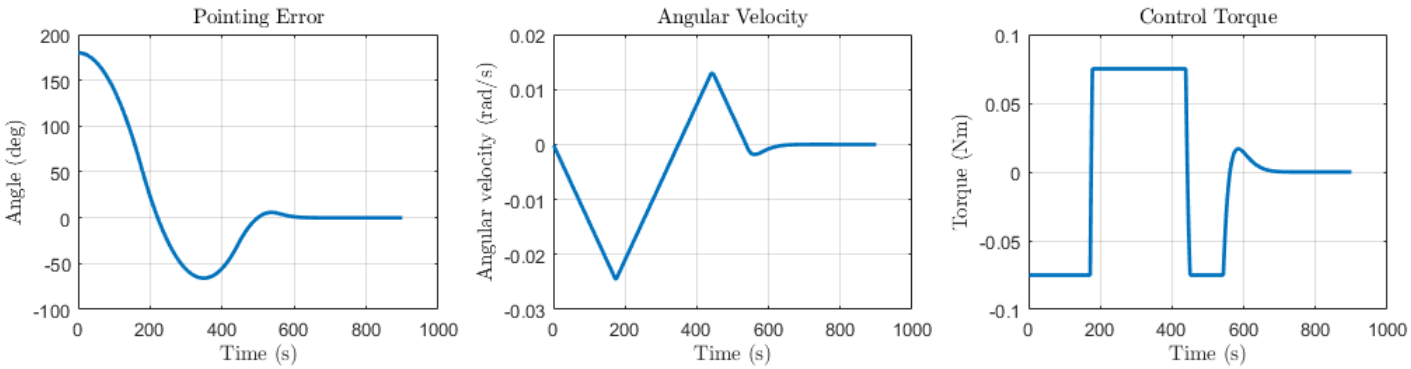

Figure 4: PD controller: Response for Amazonia-1 satellite inertia.

For the response presented in Figure 5, the inertia value and the torque limit have been changed to the Nanosatellite case, according to values in Table 2. We see that the PD controller, with the same gains, was able to stabilize the satellite and completely eliminate the steady-state error. Nonetheless, the control activity became too intense, which would be infeasible to implement in practice.

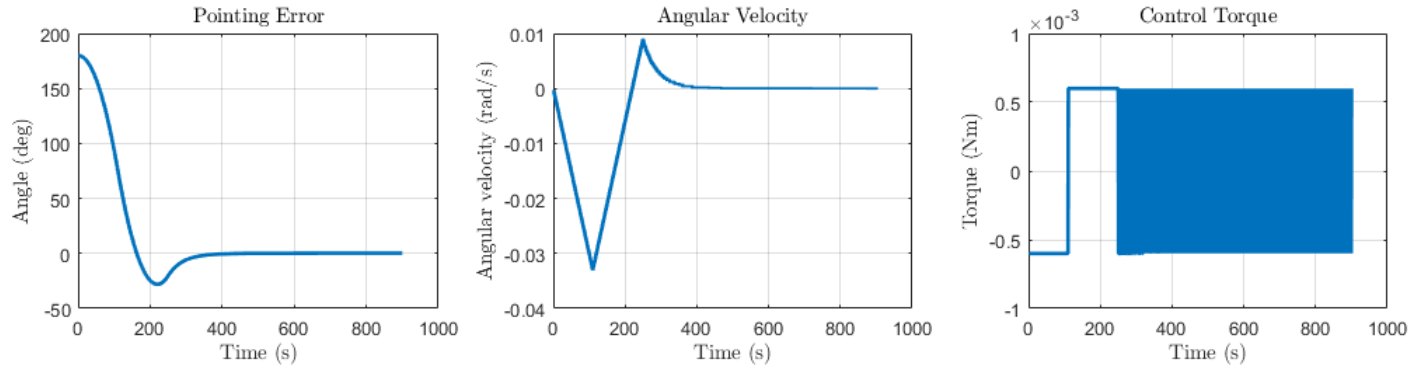

Figure 5: PD controller: Response for Nanosatellite inertia.

Figure 6 presents the satellite response when controlled by the Neural Network trained with the reinforcement learning techniques as explained in this paper. This response corresponds to the Nanosatellite configuration since it was the one used during training. We see that the response was smooth, with no overshoot, but looking close it is possible to note a small bump in the

pointing error plot. It indicates that the satellite have reached zero pointing error and then went back for a short period of time until returning to approach zero in steady-state. Also, although it is not evident from Figure 6 due to the scale, actually there was a small steady-state error of less than 1°. Since the objective of this paper is only to assess the feasibility of the method, it was regarded as enough, but depending on the satellite mission, as for instance the Amazonia-1 satellite, whose mission is to take pictures for remote sensing, this would not be acceptable. A possible approach to solve this problem is to have a better design for the cost function, which was not sufficiently explored in this work and also a hyperparameter tuning procedure should be attempted.

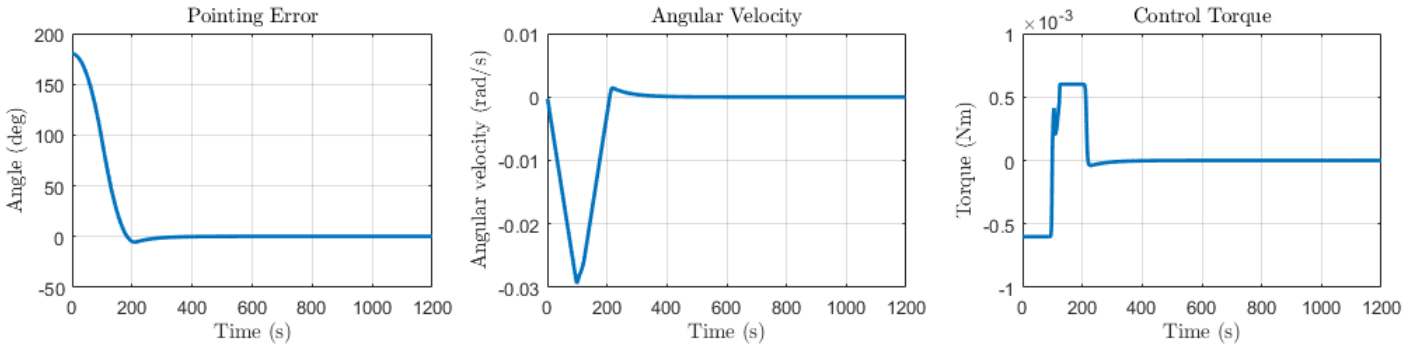

Figure 6: NN controller: Response for Nanosatellite inertia.

Finally, Figure 7 shows the result for the larger inertia case. For this test, the Neural Network gains were kept unchanged. Notably, the NN controller managed to successfully control the satellite even though its parameters have been so drastically changed. Still, the reader can see that the small overshoot barely noticeable in Figure 6 persisted in this case and actually have been considerably intensified, which might be due to some poor behavior that the neural network have learned during the training process. At this point, it is important to remember that the training process is essentially an optimization process, therefore the neural network may have converged to a local minimum, which is a known issue of policy gradient methods. Nevertheless, if this result is compared with the PD controller response in Figure 4, we see that both responses are very close. This shows that the neural network learned the correct behavior, despite the small steady-state error.

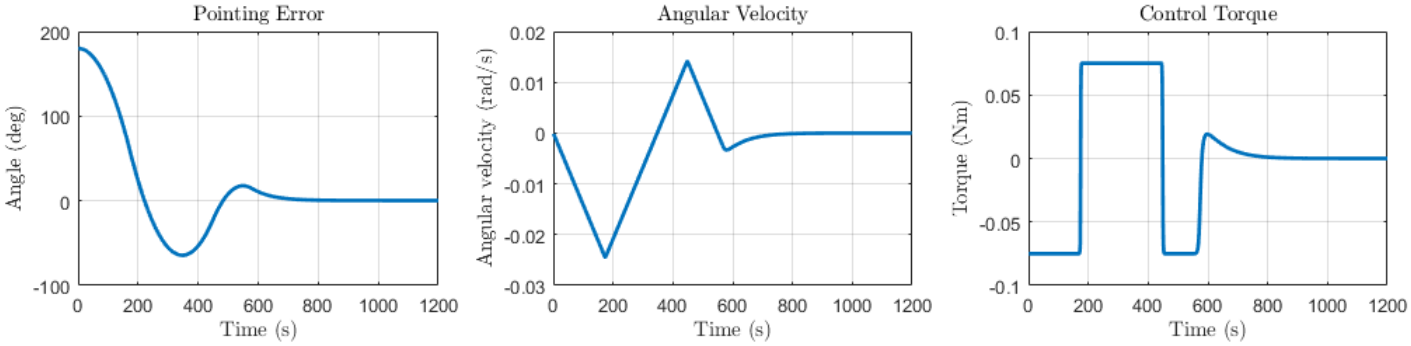

Figure 7: NN controller: Response for Amazonia-1 satellite inertia.

4. Conclusion

According to the results presented above, one can see there are benefits to further explore the application of reinforcement learning to satellite attitude control. However, it is clear the necessity

to correctly choose the hyperparameters as well as the design of a suitable reward function to the success of the method. This work fulfilled its goal of proving the feasibility of the method and the following steps will be to craft a better cost function and to find an appropriate set of hyperparameters in order to completely eliminate the steady-state error, and to extend this approach to the full tridimensional attitude control problem.

Acknowledgments: *The authors would like to thank CAPES for the Postgraduate Scolarship.*

References

Andrychowicz, O. M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., et al. (2020). Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym.

Carrara, V., Januzi, R. B., Makita, D. H., Santos, L. F. d. P., and Sato, L. S. (2017). The itasat cubesat development and design. *Journal of Aerospace Technology and Management*, 9(2):147–156.

Flores-Abad, A., Ma, O., Pham, K., and Ulrich, S. (2014). A review of space robotics technologies for on-orbit servicing. *Progress in Aerospace Sciences*, 68:1–26.

Fujimoto, S., Van Hoof, H., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*.

Hill, A., Raffin, A., Ernestus, M., Gleave, A., Kanervisto, A., Traore, R., Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., and Wu, Y. (2018). Stable baselines. `https://github.com/hill-a/stable-baselines`.

Hovell, K. and Ulrich, S. (2020). On deep reinforcement learning for spacecraft guidance. In *AIAA Scitech 2020 Forum*, page 1600.

Hwangbo, J., Lee, J., Dosovitskiy, A., Bellicoso, D., Tsounis, V., Koltun, V., and Hutter, M. (2019). Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26).

Kober, J., Bagnell, J. A., and Peters, J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274.

Koch, W., Mancuso, R., West, R., and Bestavros, A. (2019). Reinforcement learning for uav attitude control. *ACM Transactions on Cyber-Physical Systems*, 3(2):1–21.

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. *arXiv preprint arXiv:1509.02971*.

Ma, Z., Wang, Y., Yang, Y., Wang, Z., Tang, L., and Ackland, S. (2018). Reinforcement learning-based satellite attitude stabilization method for non-cooperative target capturing. *Sensors*, 18(12):4331.

Markley, F. L. and Crassidis, J. L. (2014). *Fundamentals of spacecraft attitude determination and control*, volume 33. Springer.

Scarritt, S. (2008). Nonlinear model reference adaptive control for satellite attitude tracking. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, page 7165.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.