Executive Summary of the Thesis

Satellite attitude control through reinforcement learning in the case of co-orbital ASAT attacks

Laurea Magistrale in Space Engineering - Ingegneria Spaziale

Author: Edoardo Biasini

Advisor: Prof. Pierluigi Di Lizia

Co-advisor: Cecilia Pisano, Enrico Congiu

Academic year: 2022-2023

1. Introduction

The thesis outlined in this summary originates from a collaboration with Nurjana Technologies and its research and development team. The research objective was to develop an attitude controller capable of operating on-off thrusters in the context of a co-orbital ASAT (Anti-Satellite) attack. The chosen methodology to pursue this goal is reinforcement learning, a branch of machine learning, which has been employed to simultaneously control all three rotation axes. The research in this regard is progressing towards an increasing interest in countermeasures for potential ASAT attacks [1]. During the research, several variations of existing algorithms were introduced, such as the partitioned replay memory (see section 4.3 and 5.1) and the pre-training of a network with synthetic data from a simpler controller (section 5.1).

2. ASAT weapons

Various ASAT weapons have been tested since the early days of the space age, and some have even been employed for military purposes. They can operate through physical contact or electromagnetic waves. In this context, the focus is on co-orbital physical ASAT weapons, namely space weapons used from satellite to satellite in orbit. These may include metal fragments generated by an explosion in orbit or, for example, technologies designed for debris removal that can also be adapted for use as weapons. Noteworthy examples in this regard are the tests conducted as part of the RemoveDEBRIS mission, where the functionality of a satellite-capturing net and a harpoon capable of attaching to target satellite surfaces by perforating them was evaluated [2]. Attacks with weapons of this kind will be those simulated during the testing of the controller obtained through reinforcement learning.

3. Theoretical Background

In order to better understand the topics covered, the initial chapters will introduce the theoretical context of the thesis, starting with the physics of rigid body rotation and concluding with an introduction to machine learning.

3.1. Physics of Satellite Rotation

Regarding the physics of rotation, the satellite has been treated as a rigid body. The physical model governing a rigid body, when the body frame is taken coincident with the principal axes of inertia, is represented by equation 1.

$$\begin{cases} I_x\dot{\omega}_x + (I_z - I_y)\omega_z\omega_y = M_x \\ I_y\dot{\omega}_y + (I_x - I_z)\omega_x\omega_z = M_y \\ I_z\dot{\omega}_z + (I_y - I_x)\omega_y\omega_x = M_z \end{cases} \tag{1}$$

The mentioned equation governs the temporal evolution of the satellite's angular velocity, given its inertia and applied moments. When considering orientation as well, it is necessary to choose a representation method. The most common method, also employed in this thesis, is quaternions. Quaternions are a four-element vector capable of describing the satellite's orientation. The evolution of quaternions, depending on the angular velocities of the rigid body, is described by the equation 2.

$$\frac{d\underline{q}}{dt} = \frac{1}{2}\mathbf{\Omega}(\underline{\omega})\underline{q} \tag{2}$$

Where $\mathbf{\Omega}(\underline{\omega})$ is a four by four matrix function of the angular velocity vector.

3.2. Machine Learning

In the chapter on machine learning, the two forms of this technology utilized in the thesis, namely supervised learning and reinforcement learning, have been extensively described. The foundational model of machine learning is the artificial neural network. It consists of nodes connected by weighted links, each node characterized by a bias and an activation function through which the input must pass. Neural networks aim to store and generalize what is learned during their training.

3.2.1 Supervised Learning

Supervised learning is a process in which the neural network is adjusted and modified to be capable of providing desired outputs for the given inputs. To achieve this, a training process is carried out on a dataset containing a large number of input data paired with corresponding desired outputs, known as labels. Through a process called backpropagation, the weights and biases of the network are adjusted to make accurate predictions based on inputs, even when those inputs were not seen during training. In this thesis, supervised learning is employed for pre-training on a synthetic dataset to prepare the network for effective training using reinforcement learning.

3.2.2 Reinforcement Learning

Reinforcement learning, unlike supervised learning, does not rely on labeled data. Instead, it involves an agent making decisions based on a state within an environment. The decisions made lead to a subsequent state, where another decision must be made. At each transition from state to state, a reward is provided, and the ultimate goal of the algorithm is to find a map between states and actions to be taken, that maximize the total sum of rewards. The mathematical framework that underlies reinforcement learning, addressing the problem it seeks to solve, is the Markov Decision Process. The declinations of reinforcement learning used in this thesis are:

- **Q-Learning**: in this case, each possible state-action pair is associated with a number called the Q-Value within a table known as the Q-Table. This table is modified during the process to ensure that, for each state, the optimal choice corresponds to the highest associated Q-Value.
- **Deep Q-Learning**: it is conceptually similar to Q-Learning, with the only difference being that instead of the Q-Table, a neural network is used to perform the same function. Its training can be quite time-consuming, especially for long sequences of choices. It functions with the use of replay memory, which involves storing all the steps taken to utilize them for learning [3].
- **Proximal Policy Optimization**: looking at past research, this type of reinforcement learning appears to be the most widely used for solving the satellite control problem. It relies on two networks: one that makes decisions and explores the state-action space, and another that evaluates the quality of a sequence of choices. This approach allows for a better assessment of sequences of choices that may initially seem unfavorable but ultimately lead to very high rewards [4].

4. 1D Controller with Reinforcement Learning (RL)

To proceed step by step, the problem was initially addressed by starting with the one-dimensional case, involving only one axis of rotation. Initially, only angular velocity is considered, and later both angular velocity and angle are taken into account.

4.1. 1D angular velocity control with Q-Learning

In this initial attempt, the environment model was created based on the rotation equation around an axis, and the necessary Q-Table for the algorithm's operation was generated. The agent was designed so that at each step, it can decide whether to apply a zero moment, a positive moment of 1 Nm, or a negative moment of -1 Nm. The chosen reward function is positional, meaning it depends solely on the resulting state after taking the action. A really negative reward is given if the boundaries of the considered angular velocity field il reached.

$$\begin{cases} Rw(S_{t+1}) = -10 \cdot (S_{t+1})^2 \\ Rw(S_{t+1}) = -20000 \qquad \text{if } |S_{t+1}| = 2 \end{cases}$$

After a very short training period, the obtained result is a functional controller capable of reducing the angular velocity to zero (figure 1).

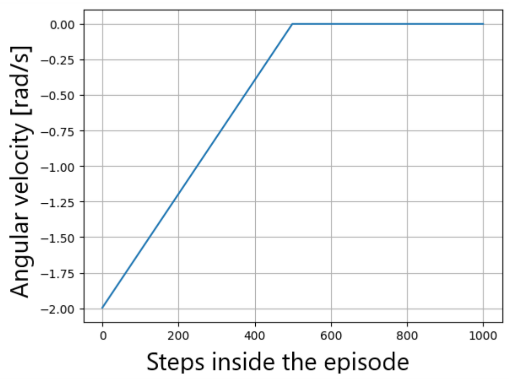

Figure 1: Q-Learning controller

4.2. 1D angular velocity control with Deep Q-Learning

Proceeding step by step, the same problem was solved using Deep Q-Learning. The environment remains the same as in the previous case, as do the possible actions that the agent can take. Even in this early stage of problem resolution, one can observe the increased complexity required for Deep Q-Learning to converge compared to simple Q-Learning. Indeed, besides a longer training time, convergence required the use of a relative reward function instead of a positional one, meaning it is a function of both the state before the action and the subsequent state.

$$Rw = \begin{cases} +1 & \text{if } |S_{t+1}| < |S_t| \\ -1 & \text{if } |S_{t+1}| > |S_t| \\ +2 & \text{if } |S_t| < 0.01 rad/s \\ +100 & \text{if } |S_t| < 0.01 rad/s \\ & \text{ and } |S_{t+1}| < 0.01 rad/s \end{cases}$$

The advantage gained through this complication of the problem is that the obtained network can control the satellite over a much broader range of velocities than those encountered during training (up to 2 rad/s compared to a maximum angular velocity observed in training of 0.1 rad/s). After just a few minutes of training, the obtained controller behaves exactly as expected (figure 2).

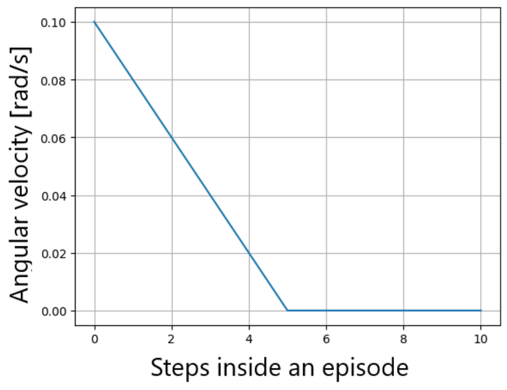

Figure 2: Deep Q-Learning controller

4.3. 1D angular velocity and angle control with Deep Q-Learning

In this section, a problem similar to the previous one was addressed, with the addition of the angle as a state to be controlled. The goal for the agent is now to bring both the angle and the angular velocity to zero. The environment is similar to the previous one, with the addition of the equation for angle propagation, and the actions the agent can take are the same as in the previous case. The chosen reward function is positional.

$$Rw = \frac{1}{2\pi} e^{(-||\underline{S}_{t+1}||)^{0.5}}$$

To aid the convergence of the algorithm in this case, a variation of the original Deep Q-Learning algorithm have been introduced, particularly concerning the replay memory. To ensure balanced learning among various possible actions, it was decided to divide the replay memory into compartments equal to the number of possible actions. An experience will then be stored in a specific compartment depending on the action taken during its occurrence. Later, during training, an equal number of experiences will be drawn from each compartment to ensure balance. This specific form of replay memory in this thesis will be referred to as partitioned replay memory. The training duration was relatively short (20 mins), but the adjustment of hyperparameters was not straightforward. The resulting controller operates as expected and, once again, in a broader range of states than those encountered during training (figure 3).

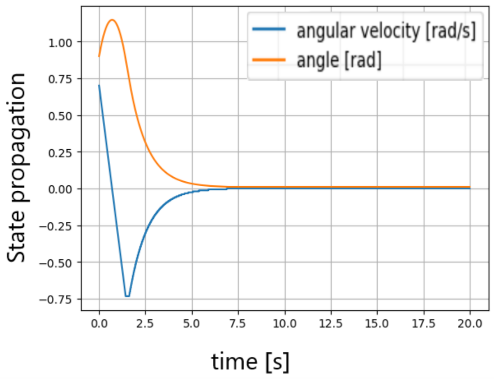

Figure 3: Deep Q-Learning controller

5. 3D Controller with Reinforcement Learning

In this section, the expansion of the problem to the three-dimensional case is described, which introduces numerous complications, starting from the significantly larger action space (27 elements) to a more than tripled state dimensionality.

5.1. 3D attitude control with Deep Q-Learning (DQN)

To address this problem, the environment was adapted to the three-dimensional case with the complete equations of motion. Initially, many combinations of reward functions and hyperparameters were tested, but the problem's complexity, coupled with the slow exploration due to Deep Q-Learning and its difficulty with long sequences of choices, made convergence challenging. To aid convergence, it was chosen to pretrain the network using informations obtained from the one-dimensional controller applied to each of the three axes. To transform this raw controller into a three-dimensional network, supervised learning was performed on synthetic data collected from the behavior of the one-dimensional controller. This provided a good initial guess for fine-tuning precision with Deep Q-Learning. The reward function used was:

$$\begin{cases} Rw = 1 - 0.99 \cdot [2 \cdot cos^{-1}(q_4)]^{0.2} & \text{if } Rw > 0 \\ Rw = 0 & else \end{cases}$$

The process was successful, and the use of partitioned replay memory proved crucial in the final precision refining stage; without it, the problem seemed to lack convergence. Overall, the training lasted almost ten hours and resulted in a controller that, while lacking in final pointing precision, exhibits a constant bias from zero, as observed in initial tests. Fortunately, this bias appears consistent, allowing for calibration. The obtained behavior aligns with expectations (figures 4 and 5).

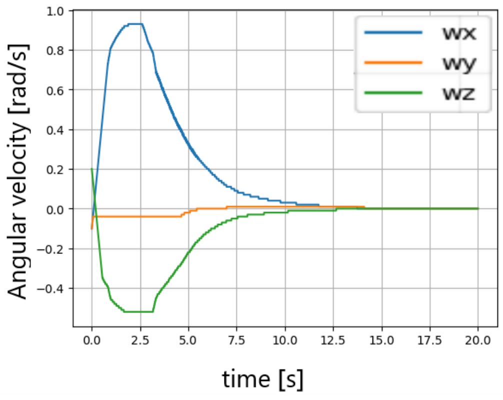

Figure 4: 3D_DQN calibrated controller angular velocity

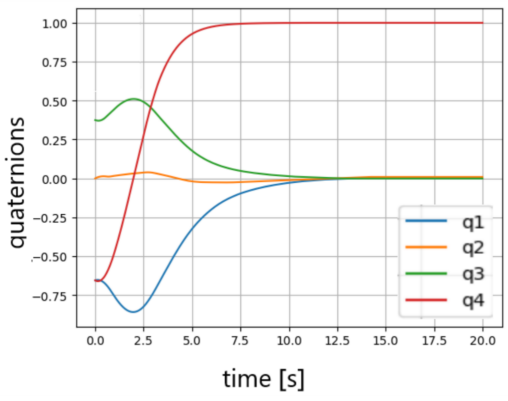

Figure 5: 3D_DQN calibrated controller quaternion

5.2. 3D control with Proximal Policy Optimization (PPO)

Given that the use of Deep Q-Learning for controller development proved to be complex and required the use of partitioned replay memory and pre-training, it was decided to also test PPO, an algorithm commonly used for such problems. The environment, as well as the action space consisting of 27 elements, remains the same as in the previous case, but the reward function has been modified.

$$\begin{cases} Rw = 1 - 0.99 \cdot (\|[q_1, q_2, q_3]\|)^{0.2} & \text{if } Rw > 0 \\ Rw = 0 & else \end{cases}$$

The reward is further multiplied by a factor of two if the state is very close to the desired state, and the maneuver performed is zero torque for all three axes. Convergence with PPO was much simpler and more linear compared to Deep Q-Learning, although tuning of hyperparameters was still required. The training lasted over eight hours and produced a controller in line with expectations (figures 6 and 7).

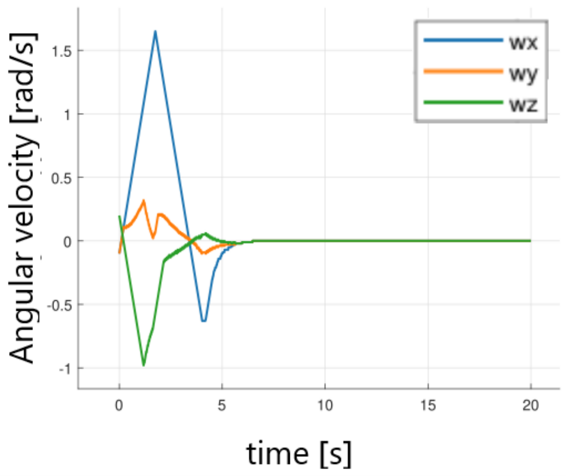

Figure 6: 3D_PPO controller angular velocity

6. Testing

The testing chapter aims to assess the three-dimensional controllers obtained: the one composed of three 1D controllers, referred to as 1Dx3, the Deep Q-Learning with pre-training controller referred to as 3D_DQN, and the one created with PPO, referred to as 3D_PPO, in the face of a co-orbital ASAT attack. The attack, simulating what could be a harpoon or a

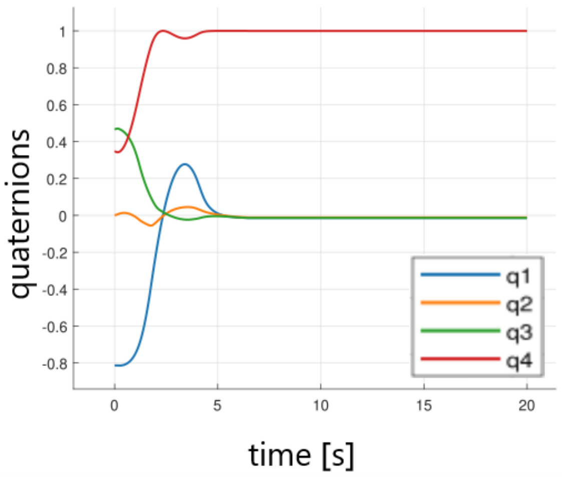

Figure 7: 3D_PPO controller quaternions

net thrown at the target satellite, is simulated through a sudden change in inertia and an applied impulsive external moment, deviating the satellite from its equilibrium. In a time span of 100 seconds, each controller's ability to restore the satellite to equilibrium is evaluated, considering precision and settling time, and comparing them to a PD controller. This test is conducted for three attack magnitudes and three increasing amounts of available control torque, in each of the combinations. The results demonstrate that the controllers can restore the system to equilibrium in cases where the PD controller proves too slow, as expected due to its dependence on the satellite's initial inertias. However, the precision and settling velocities achieved in the preliminary testing during training were not replicated in this final testing, indicating that precision is still dependent on training parameters, particularly the inertias and available control torques. Additionally, 3D_PPO exhibits uncertain behavior in states not encountered during training.

7. Conclusions

The work aimed at creating a controller through reinforcement learning capable of stabilizing a satellite under co-orbital ASAT attack with on-off thrusters, being able to simultaneously act on all three control axes. The research process yielded three 3D controllers with different characteristics, which were analyzed during testing. To achieve convergence for the 3D_DQN controller, the variation of the replay memory called partitioned replay memory was necessary, coupled with pre-training based on synthetic data obtained from a simpler controller. These adjustments were necessary to achieve effective operation of Deep Q-Learning with this problem. The obtained controllers demonstrated good behavior but were unable to generalize the precision achieved in training to various combinations of inertias and available control torques. The controllers showed higher precision as the ratio between available torque and inertia approached that of the training. Another final note can be made: it is clear that PPO is generally more suitable for developing such a controller compared to Deep Q-Learning, because of the linearity and simplicity that was demonstrated during the training. The same conclusion has been drawn in other previous works, which consistently avoid choosing Deep Q-Learning when solving attitude related problems. [5].

References

[1] Todd Harrison et al. Space Threat Assessment 2022. *Center for Strategies and International Studies.*

[2] Guglielmo S. Aglietti, Ben Taylor, Simon Fellowes, Thierry Salmon, Ingo Retat, Alexander Hall, Thomas Chabot, Aurélien Pisseloup, C. Cox, A. Zarkesh, A. Mafficini, N. Vinkoff, K. Bashford, Cesar Bernal, François Chaumette, Alexandre Pollini, and Willem H. Steyn. The active space debris removal mission RemoveDebris. Part 2: In orbit operations. *Acta Astronautica*, 168:310–322, March 2020.

[3] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning, December 2013. arXiv:1312.5602 [cs].

[4] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, August 2017. arXiv:1707.06347 [cs].

[5] James Allison, Matthew West, Alexander Ghosh, and Fnu Vedant. Reinforcement Learning for Spacecraft Attitude Control. October 2019.