73nd International Astronautical Congress (IAC), Paris, France, 18-22 September 2022.

IAC-22,C1,2,9,x69270

Application of deep reinforcement learning for attitude control of a satellite in the presence of uncertainties

Jan Luca Loettgen [a], **Matteo Ceriotti** [b], **Gerardo Aragon-Camarasa** [c], **Kevin Worrall** [d]

[a] James Watt School of Engineering, University of Glasgow, j.loettgen.1@research.gla.ac.uk
[b] James Watt School of Engineering, University of Glasgow, matteo.ceriotti@glasgow.ac.uk
[c] School of Computing Science, University of Glasgow, gerardo.aragoncamarasa@glasgow.ac.uk
[d] James Watt School of Engineering, University of Glasgow, kevin.worrall@glasgow.ac.uk

Abstract

This paper investigates the performance of satellite attitude controllers based on deep reinforcement learning, trained in idealised simulation environments and deployed into uncertain and noisy simulation environments. Additionally, it is investigated whether training directly in the uncertain environment improves performance when deployed to that environment. Uncertainties considered are Gaussian white noise superimposed onto sensor measurements of the satellite angular velocity and attitude quaternion and uncertainty in the satellite inertia tensor. The platform selected is a 6U CubeSat. The results indicate that the deep-reinforcement-learning-based attitude controller is able to maintain pointing accuracy on satellites of different inertia tensor to the training satellite. The pointing accuracy decreases when the sensor measurements are subject to Gaussian white noise. The results also suggest that training directly in the environment subject to these uncertainties does not improve pointing accuracy relative to the controller trained in the ideal environment if these uncertainties cause the state vector to break the Markov property. Furthermore, training in the uncertain environment can add instability to the learning process and prevent the controller from converging to a high performance behavioural policy.

1. Introduction

An autonomous attitude control system is one of the principal systems on-board satellites that require high pointing accuracy. Traditionally, autonomous onboard attitude control has been achieved through quaternion feedback [1, 2], Proportional Integral Derivative (PID), and sliding mode controllers [3–6]. While these methods are autonomous, robust, and can achieve high pointing accuracy, they require careful parameter tuning for every specific satellite and manoeuvre.

Deep Reinforcement Learning (DRL) may offer an attitude controller that is not sensitive to parameter tuning and is capable of adapting to uncertainties such as inertia tensor uncertainty, disturbances, or actuator degradation. The applications of DRL to satellite attitude control has been investigated for spacecraft detumbling after capturing an asteroid of unknown inertia tensor in Ref. [7]. In Ref. [8] a DRL agent was trained, using the Proximal Policy Optimisation (PPO) and curriculum learning, as a controller for the large angle slew manoeuvre of a spacecraft with an inertia tensor varying from the mass of a 1U-CubeSat to that of the International Space Station. The applications of the Deep Deterministic Policy Gradient DRL algorithm to the large angle slew manoeuvre is investigated in Ref. [9], where industry standard pointing accuracy is achieved and the resulting controller is shown to be robust against disturbances unseen during training and varied principal moments of inertia relative to the training satellite. In Ref. [10], the PPO algorithm is used with a discrete action space to achieve industry standard pointing accuracy during the large angle slew manoeuvre.

Even though satellite attitude control using DRL is promising, results in the approaches above were achieved in idealised simulations. Before DRL attitude controllers can be used on real satellite, it will be necessary to investigate the tolerance of DRL attitude controllers to a range of uncertainties including sensor measurement noise and inertia tensor uncertainty. Furthermore, DRL is currently not suitable to being trained online on a real satellite in space. This is due to risks to the satellite safety in the early training phases and due to the limited onboard available computational power. Therefore, there is a need to train DRL attitude controllers in simulation before they can be deployed onto real satellites. There will be inevitable differences between the simulated satellite, used for training, and the final satellite onto which the DRL attitude controller is deployed. In this paper, we investigate how these differences affect the performance of the DRL attitude controller during the large angle slew manoeuvre, by simulating them, on a 6U-CubeSat satellite. Specifically, we look at how the addition of Gaussian white noise in sensor measurements of the satellite angular velocity and attitude quaternion affect the performance of the DRL attitude controller. We also investigate how discrepancies in the products of in-

ertia between the training and testing satellites affect the performance of the DRL attitude controller. Finally, we investigate whether adding Gaussian white noise to sensor measurements during training and if training the DRL attitude controller on a satellite of varied products of inertia makes the resulting DRL attitude controller more robust to these uncertainties.

The rest of this paper is organised as follows. Section 2 covers the mathematical model of the 6U-CubeSat. Section 3 is a summary of reinforcement learning and Markov Decision Processes (MDPs). Section 4 is comprised of three major subsections, the first covering the definition of the state vector, the action space, and the reward function used to train the DRL agent. The second subsection details how Gaussian white noise in sensor measurements and inertia tensor uncertainty were implemented. The third subsection details the experiments conducted. The results of the simulations are covered in Section 5. Finally a summary of the findings and future work is covered in Section 6.

2. Mathematical model

This section covers the mathematical model of a 6U-Cubesat providing the required frames of reference, dynamics, and kinematics. The standard 6U rigid-body design is used: 6 kg, 0.3 m × 0.2 m × 0.1 m. No depolyables are modelled.

2.1. Reference frames

Three reference frames are considered in this paper: the inertial reference frame, the Principal Axes of Inertia (PAI) reference frame, and the Satellite Body (SB) reference frame. The inertial reference frame is a celestial reference frame that is inertial with respect to distant stars. The inertial reference frame considered is the Earth-Centered Inertial (ECI) reference frame. The ECI reference frame has its origin at the Earth's center of mass and has its x-axis aligned with the First Point of Aries, the z-axis is aligned with the Earth's celestial north pole. The y-axis of the ECI reference frame points to the equator such that the y-axis is perpendicular to the x and z axes. The ECI reference frame is the reference frame with respect to which the satellite tracks its own orientation and is also the reference frame with respect to which the target attitude for the large angle slew manoeuvre is defined.

The PAI reference frame is fixed with the satellite body, with its origin at the satellite center of mass. The PAI reference frame is the reference frame in which the products of inertia of the satellite are all zero. The orientation of the PAI reference frame relative to the satellite body is determined by the mass distribution of the satellite.

The SB frame is a body frame utilised by engineers when designing the satellite. The SB reference frame is fixed with the satellite body, and has its origin located at the satellite centre of mass. The SB frame considered has its x-axis perpendicular to one of the 2U×1U faces of the 6U-CubeSat. The z-axis is perpendicular to one of the 3U×2U faces of the CubeSat, and the y-axis completes the right handed pair such that it is perpendicular to the corresponding 3U×1U face of the CubeSat. This is the reference frame with respect to where the satellite actuators are placed, such that each actuator produces a torque about its corresponding satellite body axis. Parameters in the ECI, PAI, and SB reference frames are indicated with the subscript I, P and B respectively.

2.2. Dynamics

The satellite is assumed to be a rigid body, as such its rotational dynamics are governed by Euler's equation of rotational rigid-body dynamics:

$$\dot{\boldsymbol{\omega}}^{BI}(t) = \boldsymbol{J}_B^{-1}(\boldsymbol{L}_B(t) - \boldsymbol{\omega}^{BI}(t) \times (\boldsymbol{J}_B \boldsymbol{\omega}^{BI}(t))) \quad (1)$$

Euler's equation is expressed in the body reference frame. $\boldsymbol{\omega}^{BI}$ is the angular velocity of the SB reference frame relative to the the ECI reference frame, $\boldsymbol{J}_B$ is the inertia tensor of the satellite expressed in the SB reference frame, and $\boldsymbol{L}_B$ is the sum of external torques.

2.3. Kinematics

The attitude control system utilises a quaternion-based attitude parameterisation. The rate of change of the attitude quaternion (describing the orientation of the body reference frame relative to the ECI reference frame) is:

$$\dot{\boldsymbol{q}}^{BI}(t) = \frac{1}{2}\boldsymbol{\omega}^{BI}(t) \otimes \boldsymbol{q}^{BI}(t) = \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega}^{BI})\boldsymbol{q}^{BI}(t) \quad (2)$$

$$\boldsymbol{\Omega}(\boldsymbol{\omega}^{BI}) = \begin{bmatrix} 0 & -\omega_3 & \omega_2 & \omega_1 \\ \omega_3 & 0 & -\omega_1 & \omega_2 \\ -\omega_2 & \omega_1 & 0 & \omega_3 \\ -\omega_1 & -\omega_2 & -\omega_3 & 0 \end{bmatrix} \quad (3)$$

Eq. 2 contains 4 by 1 quaternion vectors, where the scalar element of the quaternion takes the last place in the column matrix.

Together, Eqs. 1 and 2 are sufficient to fully define the attitude of the spacecraft at any point in time, given its initial conditions, through numerical integration. The satellite was simulated by integrating Eqs. 1 and 2 using the Runge-Kutta 4^{th} order method with an integration time-step of 0.05 s. The integration time-step was chosen to minimise computational expenditure, as thousands of simulations are needed to train the controller, while maintaining simulation accuracy.

2.4. Inertia tensor

The 6U-CubeSat is modelled as a uniform density cuboid of mass $m = 6$ kg, length $l = 0.3$ m, width $w = 0.2$ m, and height $h = 0.1$ m. The three principal moments of inertia are then found using:

$$I_x = \frac{m}{12}(w^2 + h^2) = 0.025 \text{ kgm}^2 \tag{4}$$

$$I_y = \frac{m}{12}(l^2 + h^2) = 0.05 \text{ kgm}^2 \tag{5}$$

$$I_z = \frac{m}{12}(l^2 + w^2) = 0.065 \text{ kgm}^2 \tag{6}$$

For the case that these are the only components to the satellite inertia tensor then the PAI reference frame is parallel to the SB reference frame.

3. Reinforcement learning

3.1. Markov Decision Processes

Reinforcement learning is a framework for teaching a decision making agent to solve discrete-time MDPs. MDPs formalise the sequential decision making of an agent in an environment. The environment comprises everything but the decision-making agent; in this paper, the environment is the governing equations that simulate the CubeSat, as well as the sensors, actuators, and uncertainties. In a MDP at every time-step, the agent receives interpretation of the current state of the environment, based upon the current state of the environment the agent selects an action. The action causes the environment to transition into a new state, after which the agent receives a reward and a new interpretation of the current state of the environment as shown in Fig 1. This process of receiving a state interpretation, selecting an action, and then receiving a reward and a new state interpretation continues until the environment transitions into some terminal state, which ends the current episode of interactions with the MDP. The goal is for the agent to learn a behavioural policy that maximises the sum of rewards the agent expects to receive in any episode.

The behavioural policy $\pi(a|s)$ is a mapping from a state s to probabilities of selecting each of the available actions a in that state. Traditionally, the agent learns the behavioural policy by repeatedly interacting with the environment over many episodes. It should be noted that the agent is only the decision maker, i.e. the controller from traditionally control theory, and everything else including sensors, actuators, and the dynamics of the problem are part of the environment. The state measurement the agent receives are sensor measurements, and the action the agent selects are actuator control signals.

Deep reinforcement learning utilises artificial neural networks to parameterise the behavioural policy. The neural network takes as input the state vector and outputs values that can be used to derive actions, the exact nature of the output depends on the specific DRL algorithm. When the agent is learning the behavioural policy in DRL it is learning neural network parameters. The DRL algorithm used was the Proximal Policy Optimisation [11] algorithm, which was chosen for its continuous action space, fast policy convergence rate, and stability properties that limit destruction policy updates.

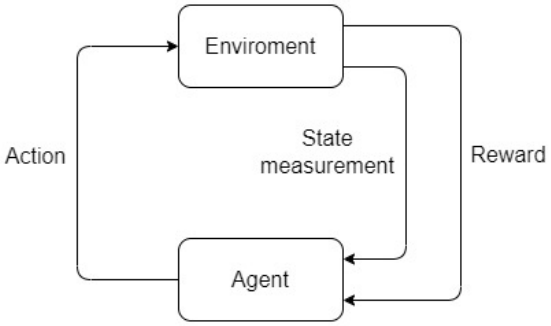


Fig. 1 Markov Decision Process

3.2. Markov Property

The state interpretation that the decision making agent receives at every time-step is referred to as the state vector. The state vector possess the Markov property if the state vector contains all the important information of all previous states and actions [12]. Equivalently, when the agent is provided the current state vector it should have the same relevant information, for the purpose of solving the problem, available as when provided with the entire history of measurements of the system and actuator control signals. Traditional reinforcement learning algorithms assume that the state vector will have the Markov property.

4. Methodology

4.1. State vector

The state vector replaces the error control signal, from traditional control methods, as the input to the attitude controller. The state vector:

$$s(t) = \begin{bmatrix} \boldsymbol{q}_m^{BT}(t) \\ \dot{\boldsymbol{q}}_m^{BT}(t) \\ \boldsymbol{\omega}_m^{BI}(t) \\ \hat{t} \end{bmatrix}^T \tag{7}$$

contains the four components of the error quaternion $\boldsymbol{q}^{BT}(t)$, describing the rotation between the current orientation and the target orientation, the four components of

the rate of change of the error quaternion $\dot{\boldsymbol{q}}^{BT}(t)$, the three components of the spacecraft angular velocity $\boldsymbol{\omega}^{BI}(t)$, and the current normalised simulation time $\hat{t}$. The subscript m indicates that these are the measured quantities which are not necessarily the same as the true quantities (without subscript m). Note, the state vector is a vector that summarises the state of the environment and provides the controller a measure of properties that may be useful for deriving actuator control signals.

While both the rate of change of the error quaternion and the angular velocity describe the same information it may be useful for the DRL attitude controller to be able to access both parameters without having to learn the transformation between them, which is why both are included in the state vector. The normalised simulation time is included such that the state vector satisfies the Markov property. Specifically, without access to the simulation time, the DRL attitude controller would have no measure of the remaining simulation time, and hence have to attempt to complete the manoeuvre as quickly as possible instead of completing the manoeuvre efficiently and safely.

4.2. Action space

The DRL attitude controller was trained using the Proximal Policy Optimisation (PPO) [11] algorithm with a continuous action space. As stated the PPO algorithm was selected for its continuous action space, fast convergence rate and its clip fraction in the loss function that aims to limit destructive policy updates. It is assumed that the CubeSat has a control system capable of producing the commanded control torques about the satellite body axes, saturating at a maximum torque of 2 mNm about each axis. The actuators are considered as external torque sources and represented by the $\boldsymbol{L}_B(t)$ term in Eq. 1.

4.3. Reward function

The reward function used to train the DRL attitude controller comprises of three reward elements. The first rewards the agent for pointing within 0.25° of the target orientation at the end of the episode:

$$r_1(t) = \begin{cases} 1, & \text{if } \phi(t) \leq 0.25^\circ \text{ at } t = 100 \text{ s} \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

The second reward element:

$$r_2(t) = \frac{\phi(t - \Delta t) - \phi(t)}{\pi} \tag{9}$$

is an auxiliary reward function that is used to supplement the first reward element. This second reward element is required due to the sparsity of the first reward element given that the random untrained agent is unlikely to ever obtain the first reward element. The second reward element is awarded after every time-step and is equal to the difference in the Euler angle ϕ, described by the error quaternion $\boldsymbol{q}^{BT}$, between the previous time-step and the current time-step. This linear function of the change in the Euler angle was used because the sum of reward it can provide is bounded and independent of the rate at which the large angle slew manoeuvre is completed. That is, the maximum possible sum of rewards provided by r_2 is fixed equal normalised start of episode Euler angle and does not encourage the agent to complete the manoeuvre in minimum time, allowing for safe and efficient large angle slew manoeuvres.

The third reward element:

$$r_3(t) = \begin{cases} -1, & \text{if } \mid \boldsymbol{\omega}^{BI}(t) \mid > 1 \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

penalises the agent if the magnitude of the satellite angular velocity becomes greater than 1 rad/s and ends the simulation early. This reward function is used to limit the search space of desirable behaviours, as excessively large angular velocities can be dangerous to the satellite and because the simulation accuracy can break down if the angular velocities become large (this is due to constant integration time-step used when simulating the satellite's dynamics).

The overall reward function provided to the agent at after every time-step is the sum of these three reward elements.

4.4. Measurement Noise

The investigated attitude control scenario utilises a rate gyro to provide measurements of the satellite angular velocity relative to the inertial reference frame. The rate gyro is modelled as a sensor whose measurements are subject to Gaussian white noise $\boldsymbol{N}_m$ directly:

$$\boldsymbol{\omega}_m^{BI}(t) = \boldsymbol{\omega}^{BI}(t) + \boldsymbol{N}_m(\sigma_m) \tag{11}$$

The noise is sampled from a Gaussian distribution, with a mean of 0.0 and standard deviation of $\sigma_m = 3.9738 \times 10^{-3}$ °, approximately in line with that of standard CubeSat gyroscopes. Note, the Gaussian distribution is sampled three times at every time-step to provide separate noise values for each of the three angular velocity vector components.

The attitude quaternion determination system considers an inertial strap-down measurement unit that tracks the attitude quaternion by integrating Eq. 2, but using the measured angular velocity ($\boldsymbol{\omega}_m^{BI}(t)$) in place of the true angular velocity ($\boldsymbol{\omega}^{BI}(t)$). The integration method used is the Runge-Kutta 4^{th} order integration method, with a time-step of 0.05 s as is used by the simulation, such that if there is no noise then the measured attitude quaternion is exactly the true attitude quaternion.

4.5. Inertia tensor uncertainty

The DRL attitude controller is trained in simulation and then tested in simulation. The testing simulation is meant to represent deployment into space on a true satellite. If there is uncertainty in the final inertia tensor of the deployed satellite then the DRL attitude controller cannot be trained, in simulation, for this satellite. Inertia tensor uncertainty can arise due to movement of components, such as solar panels, on the satellite or due to propellant sloshing or propellant consumption.

To simulate inertia tensor uncertainty in the testing satellite, three random normally distributed numbers are sampled for each of the products of inertia I_{xy}, I_{xz}, I_{yz}. The normal distributions for each of the three products of inertia moments of inertia have a mean of 0.0, and standard deviations of σ_{xy}, σ_{xz}, σ_{Iyz}. The addition of the products of inertia to the inertia tensor in body axes:

$$\boldsymbol{J}_B = \begin{bmatrix} I_x & -I_{xy}(\sigma_{xy}) & -I_{xz}(\sigma_{xz} \\ -I_{xy}(\sigma_{xy}) & I_y & -I_{yz}(\sigma_{yz}) \\ -I_{xz}(\sigma_{xy}) & -I_{yz}(\sigma_{yz}) & I_z \end{bmatrix} \quad (12)$$

has the effect of rotating the PAI reference frame relative to the SB reference frame. This causes further coupling in the dynamics as control torques are applied relative to the body reference frame. The standard deviations in the products of inertia were obtained by assuming that the uncertainty is represented by a 0.5 kg mass located 7.5cm, 5cm, and 2.5cm away from the CubeSat centre of mass along the x, y and z spacecraft body axes, respectively. The principal moments of inertia, and the standard deviations of the products of inertia are detailed in Table 1.

Table 1 Detailing the principal moments of inertia and standard deviation of products of inertia uncertainty.

Parameter	Value (kgm^2)
I_x	0.025
I_y	0.05
I_z	0.065
σ_{xy}	1.875×10^{-3}
σ_{xz}	0.9375×10^{-3}
σ_{yz}	0.625×10^{-3}

4.6. Investigated Scenarios

The DRL attitude controller was trained in four different scenarios of varying uncertainty. Scenario 1 is the state-of-the-art idealised scenario, where there is no uncertainty, and is used to obtain a baseline performance measure. Scenario 2 has angular velocity and attitude quaternion measurement noise as described in Section 4.4. Scenario 3 has inertia tensor uncertainty as described in Section 4.5. Scenario 4 has both measurement noise and inertia tensor uncertainty. Table 2 shows the uncertainties in every scenario. In total four agents were trained, one in each scenario.

Table 2 Investigated Scenarios

Scenario	Measurement noise	Inertia uncertainty
1	×	×
2	✓	×
3	×	✓
4	✓	✓

4.7. Training

An episode begins by putting the satellite into a uniformly-random orientation relative to the ECI reference frame, the angular velocity of the satellite is set to zero. The target orientation for the large angle slew manoeuvre is always set as having the satellite body reference frame aligned with the ECI reference frame. The error quaternion is calculated as the quaternion difference between the current orientation and the desired orientation. The DRL attitude controller receives the first state measurement based upon which it computes control torques to apply to each of the satellite body axes. The controller then receives a new state measurement and a reward and again selects an action. This process continues until the environment transitions into the terminal state which is if either the simulation reaches the maximum simulation time of 100 s, or if the satellites exceeds the maximum allowed angular velocity of 1 rad/s. Each agent was trained for 5000 episodes. When inertia tensor uncertainty is included, then at the start of the episode a random inertia tensor is generated by sampling the products of inertia from a normal distribution as discussed in Section 4.5.

The dynamics of the of the satellite were implemented as an OpenAI gym environment [13] in Python. The implementation of the PPO algorithm used was from Stable-Baseline3 [14] with a discount factor of 0.99, a learning rate of 3×10^{-3}, a batch size of 64, and a clip range of 0.2, other hyper parameters were kept as the default values specified at [14]. Separate policy and value functions networks were used, both with a neural network architecture of two dense hidden layers of 128 neurons each.

At the start of the learning process, the neural network parameters are initialised randomly, this causes the initial decision making policy to be random. The data used to train the neural network is gathered during training episodes, while partially following the policy, is also somewhat random. This can cause differences in the final effectiveness of the policy between repeat runs of the same agent in the

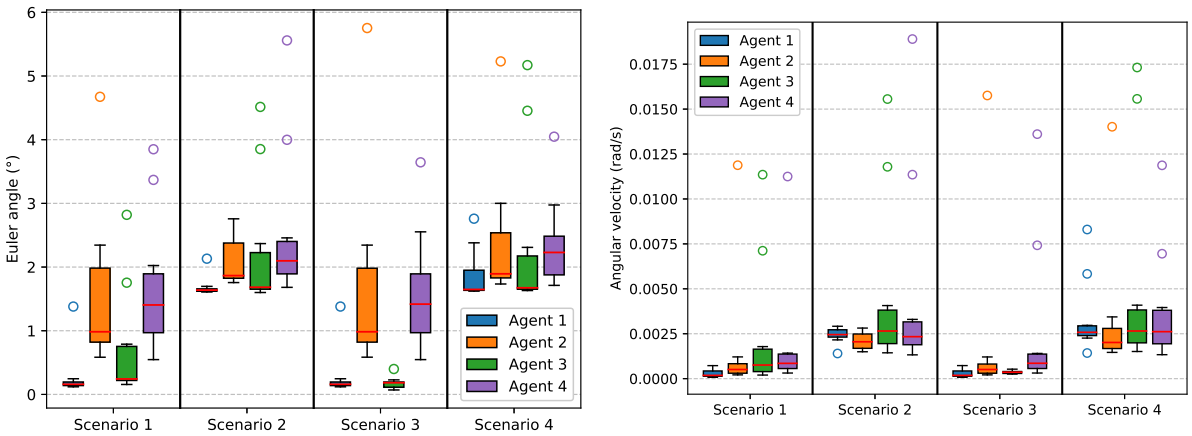
same environment under the same conditions. To ensure that results are repeatable and not dependent on random neural network initialisation, each agent is trained and tested in 10 independent repeat runs. After a DRL attitude controller has been trained for 5000 episodes in each scenario, it is then independently tested in every scenario for 1000 testing episodes.

5. Results

Figures 2a and 2b show the end-of-episode Euler angle and angular velocity magnitude respectively, for testing of each agent in every scenario over the ten repeat runs. Every run of every agent was tested in 1000 episodes in each scenario, the mean end of episode Euler angle and angular velocity of the 10 repeat runs are shown as a box plot. The red line indicates the median across the 10 runs, the boxes indicate the interquartile range, and the whiskers indicate the minimum and maximum run that fell within 1.5 times the interquartile range relative to the lower and upper quarterly respectively. All runs that did not fall within the whiskers are plotted as flier points. In Figs. 2a and 2b Agent 1-4 refers to the agents trained in Scenarios 1-4 respectively. If an agent was able to learn a high performance policy, then the attitude controller should reliably bring the satellite into orientation that minimises the end of episode Euler angle and angular velocity magnitude. Across all runs of every agent in every scenario the mean pointing error always remained below 10°. For all four agents, there were some runs which performed significantly worse than the median run. Figures 2a and 2b show that the runs with a lower median pointing error also had a lower end of episode angular velocity, showing that the policy these agents learnt had the agent stabilise the satellite orientation about the desired orientation.

Across all four scenarios Agent 1 had the lowest median pointing error and the smallest interquartile range, across the 10 runs, showing that it was the most stable agent and was consistently able to learn a high performance policy. Agent 3 has a similar median pointing accuracy as Agent 1 but larger interquartile range in all scenarios except Scenario 3 where Agent 3 had a similar interquartile range and median pointing accuracy. Scenario 3 is the only scenario where the worst and best performance run of Agent 3 performed better than the worst and best performance run of Agent 1 respectively.

Agents 2 and 4 performed worse than Agents 1 and 3 in all four scenarios. The relatively poor performance of most runs of these agents shows that the policy learnt were subpar and the large spread in performance of the runs across all scenarios shows poor convergence to a high performance policy. This is due to the fact that the addition of Gaussian noise to sensor measurements causes the state vector to break the Markov property and the state vector only contains measurements from the current time-step, preventing the agent from being able to filter out the noise. Furthermore, the reward function uses the true measures of angular velocity and orientation and not the noisy measurements, as a result of which a measured state that the agent believes to be rewarding may not be rewarding. Together these issues hindered the training process of Agents 2 and 4 preventing them from learning a policy that is capable of successfully controlling the satellite in even the baseline scenario. Agents 1 and 3 were also not able to maintain performance in the face of noisy measurements due to the issue with the state vector and the Markov property, but outperformed Agents 2 and 4 as



(a) Mean Euler angle at episode end

(b) Mean angular velocity at episode end

Fig. 2 The Euler angle of the attitude quaternion at the end of the episode (a) and the magnitude of the angular velocity at the end of the episode (b) obtained when testing each of the ten runs of each of the four agents in each of the four scenarios for 1000 episodes.

their training environments did not have the issue with the Markov property and reward function allowing the runs of these agents to converge to a high performance policy for the baseline scenario.

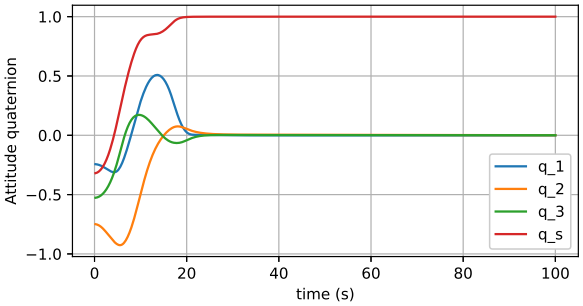
For every agent the mean pointing error in Scenario 4 is approximately equal to the sum of the mean pointing errors, of that agent, in Scenarios 2 and 3. The pointing error of all agents increased in Scenarios 2 and 4 relative to Scenario 1. None of the four agents achieve a median pointing accuracy of better than 0.1°, this is because the reward function provides diminishing rewards for smaller pointing errors. Specifically, the reward received for a pointing error of 0° relative to a pointing error of 0.1° is only 5.556×10^{-4} which is extremely small compared to the sum of rewards in any given episode.

Example episodes of each trained agent in its it corresponding scenario is shown in Figs 3-6 respectively. In all four, randomly selected, example episodes the reorientation into the target attitude is successful, as can be seen by the scalar component of the error quaternion reaching approximately 1. However, Agents 2 and 4 do not succeed in stabilising their orientation due to the noise in measurements of the spacecraft angular velocity. In all four example episodes actuator saturation, at 2 mNm, can also be seen.

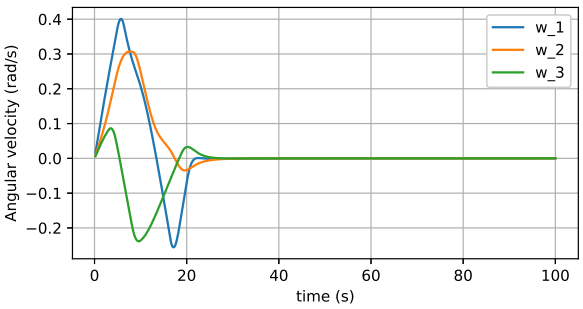
6. Conclusion

This paper sought to answer if deep reinforcement learning satellite attitude controllers, that were trained in idealised simulations, can maintain their performance when tested in environments where the satellite sensor measurements are subject to Gaussian white noise or where the mass distribution on the testing satellite is different to that of the training satellite. Furthermore, it was investigated whether training in the environment subject to these uncertainties would improve performance when testing under these conditions. The agent, trained in the idealised environment, had a median pointing accuracy of 0.1195° when tested in the idealised environment. This agent maintained performance when tested on satellites of different mass distribution despite this change not being observable to the agent. When tested with Gaussian white noise present in the sensor measurements, performance degraded to a median pointing accuracy of 1.6384°. Despite the differences between the training environment and the testing environment, the baseline agent never exceeded a mean terminal pointing error of 3° or a mean terminal angular velocity magnitude of 0.01 rad/s.

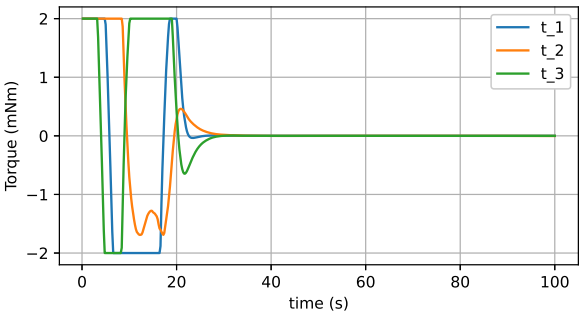
The agent trained in the environment with measurement noise performed worse than the baseline agent both when tested in the noiseless and noisy environments. The agent trained on satellites of slightly different mass distributions performed similarly to the baseline agent in all scenarios. As such there is no evidence that training in an uncertain environment helps to improve performance when tested in that environment if the uncertainty causes the state vector to violate the Markov property as is the case here. Future research will be conducted to investigate if providing previous state measurements and actions to the agent allows the agent to learn the spacecraft mass distribution thus improving performance in the face of inertia tensor uncertainty. Additionally, we will investigate the robustness of the DRL attitude controller against sudden angular velocity and constant torque disturbances.



(a) Attitude quaternion

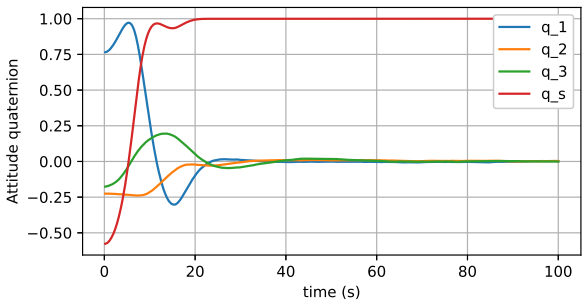


(b) Angular Velocity

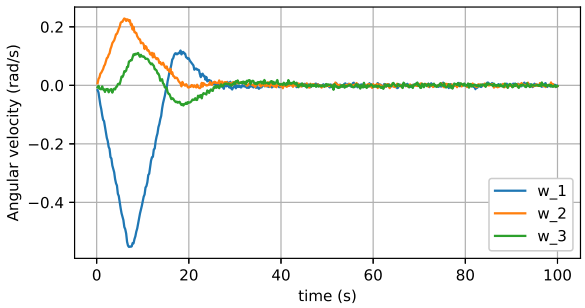


(c) Applied Torque

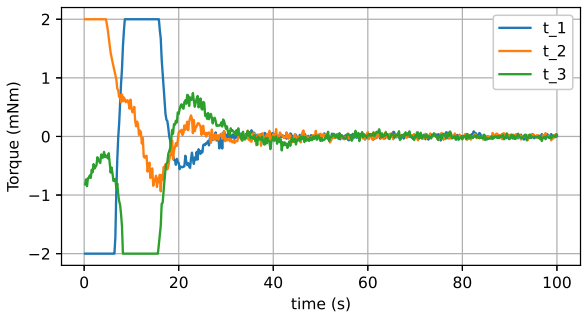
Fig. 3 Sample episode Agent 1 in Scenario 1



(a) Attitude quaternion

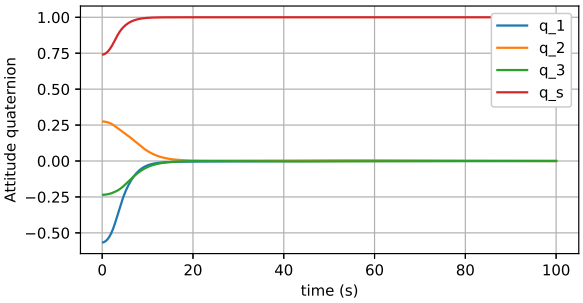


(b) Angular Velocity

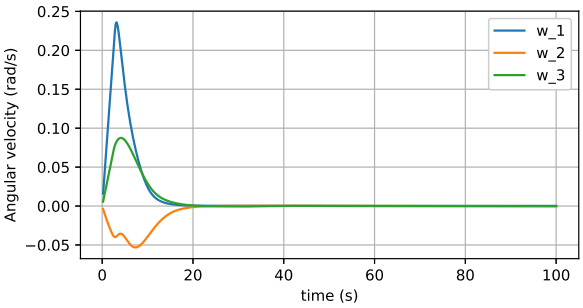


(c) Applied Torque

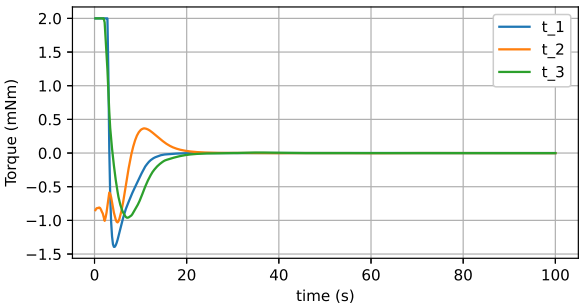
Fig. 4 Sample episode Agent 2 in Scenario 2



(a) Attitude quaternion

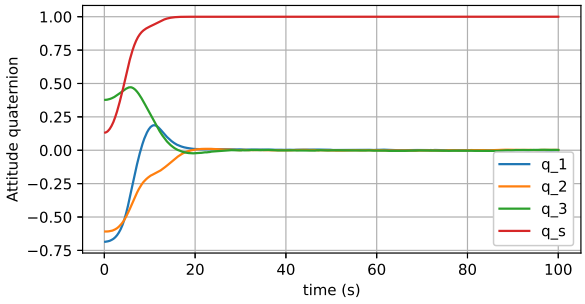


(b) Angular Velocity

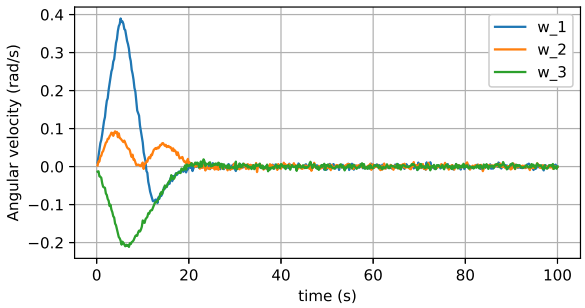


(c) Applied Torque

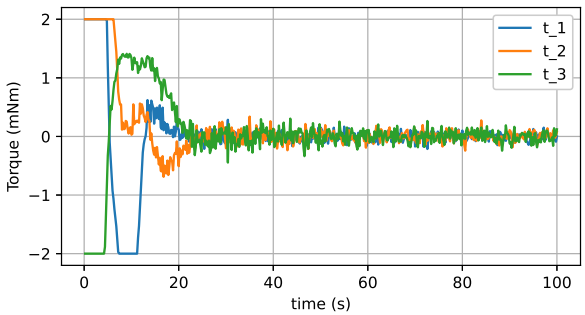
Fig. 5 Sample episode Agent 3 in Scenario 3



(a) Attitude quaternion



(b) Angular Velocity



(c) Applied Torque

Fig. 6 Sample episode Agent 4 in Scenario 4

References

[1] B. Wie and P. M. Barba, "Quaternion feedback for spacecraft large angle maneuvers," *Journal of Guidance, Control, and Dynamics*, vol. 8, no. 3, pp. 360–365, 1985. DOI: 10.2514/3.19988.

[2] B. Wie, H. Weiss, and A. Arapostathis, "Quaternion feedback regulator for spacecraft eigenaxis rotations," *Journal of Guidance, Control, and Dynamics*, vol. 12, no. 3, pp. 375–380, 1989. DOI: 10.2514/3.20418.

[3] S. R. Vadali, "Variable-structure control of spacecraft large-angle maneuvers," *Journal of Guidance, Control, and Dynamics*, vol. 9, no. 2, pp. 235–239, 1986. DOI: 10.2514/3.20095.

[4] T. A. W. Dwyer and H. Sira-Ramirez, "Variable-structure control of spacecraft attitude maneuvers," *Journal of Guidance, Control, and Dynamics*, vol. 11, no. 3, pp. 262–270, 1988. DOI: 10.2514/3.20303.

[5] Y.-P. Chen and S.-C. Lo, "Sliding-mode controller design for spacecraft attitude tracking maneuvers," in *1993 American Control Conference*, 1993, pp. 195–196. DOI: 10.23919/ACC.1993.4792835.

[6] Y. Jan and J. Chiou, "Minimum-time spacecraft maneuver using sliding-mode control," *Acta Astronautica*, vol. 54, no. 1, pp. 69–75, 2004, ISSN: 0094-5765. DOI: 10.1016/S0094-5765(03)00194-2.

[7] Y. Wang, Z. Ma, Y. Yang, Z. Wang, and L. Tang, "A new spacecraft attitude stabilization mechanism using deep reinforcement learning method," 2017. DOI: 10.13009/EUCASS2019-33.

[8] J. Allison, M. West, A. Ghosh, and F. Vedant, "Reinforcement learning for spacecraft attitude control," in *70th International Astronautical Congress (IAC) 2019*, Washington DC, United States of America", Oct. 2019.

[9] J. Elkins, R. Sood, and C. Rumpf, "Adaptive continuous control of spacecraft attitude using deep reinforcement learning," in *Proceedings of 2020 AAS/AIAA Astrodynamics Specialist Conference*, Aug. 2020.

[10] J. Elkins, R. Sood, and C. Rumpf, "Autonomous spacecraft attitude control using deep reinforcement learning," in *71st International Astronautical Congress (IAC) 2020*, CyberSpace Edition, Oct. 2020.

[11] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal policy optimization algorithms*, 2017. DOI: 10.48550/ARXIV.1707.06347.

[12] P. Montague, "Reinforcement learning: An introduction, by sutton, r.s. and barto, a.g.," *Trends in Cognitive Sciences*, vol. 3, no. 9, p. 360, 1999, ISSN: 1364-6613. DOI: 10.1016/S1364-6613(99)01331-5.

[13] G. Brockman *et al.*, "Openai gym," *CoRR*, vol. abs/1606.01540, 2016. DOI: 10.48550/arXiv.1606.01540.

[14] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.