SSC21-WKIII-04

MATA-RL: Continuous Reaction Wheel Attitude Control using the MATA Simulation Software and Reinforcement Learning

Vanessa Tan, John Leur Labrador, Marc Caesar Talampas
Electrical and Electronics Engineering Institute, University of the Philippines - Diliman
Velasquez St., Diliman, Quezon City, Philippines; +63 8981-8500 loc 3305
vanessa.tan@eee.upd.edu.ph

ABSTRACT

As earth observation satellites, Diwata microsatellites need to have a high degree of target pointing accuracy. Additionally, being in low orbit, they could experience strong external disturbances. Current methods for attitude control have proven to be effective. However, they are prone to changes in control and mass parameters. In this paper, we explore using Deep Reinforcement Learning (RL) for attitude control. This paper also leverages on Diwata's simulator, MATA: Mission, Attitude, and Telemetry Analysis (MATA) software, in training the RL agent. We implemented two RL algorithms: Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC). We then simulated different scenarios and compared the performance of these algorithms to that of Diwata's current attitude controller, the Proportional-Integral-Derivative (PID) control. Our results show that reinforcement learning can outperform traditional controllers in terms of settling time, overshoot, and stability. The results of this research will help solve problems in conventional attitude controllers and enable satellite engineers to design a better Attitude Determination and Control System (ADCS).

INTRODUCTION

Diwata satellites are the Philippine's earth observation microsatellites that undertake scientific missions related to weather observation, environmental monitoring, and disaster mitigation. As earth observation satellites, there is a need to have a high degree of target pointing accuracy. However, Diwata satellites have low orbits and would experience relatively strong external disturbances from the sun and the earth's magnetic field and atmosphere compared to satellites in higher orbits. Satellite operators would need to determine the optimal parameters and commands for accurate target pointing.

The Diwata microsatellites currently use the Proportional-Integral-Derivative (PID) controller for attitude control which has proven to be effective in stable environments. However, PID controllers for satellites are highly dependent on its mass parameters.[1,2] Other traditional controllers like the backstepping control and fuzzy control have similar problems.[1–4] A sudden change in the control parameters caused by the deployment of a solar panel or booms and other unpredictable disturbances in the satellite would result in inaccurate attitude. With the rise in popularity of using machine learning in autonomous control, we explore a new attitude controller using Deep Reinforcement Learning (RL).

Reinforcement learning has proven to be effective in complex decision making problems in which an agent learns from its environment then predicts the next action that it needs to perform. Most satellite attitude control algorithms using RL use discrete control action spaces.[1,2,5] For this research, we explore the continuous action space to mimic real-valued telemetry data of the satellite. Two reinforcement learning algorithms are explored for attitude control: Proximal Policy Optimization (PPO)[6] and Soft Actor-Critic (SAC).[7] These algorithms are known to perform well in continuous control problems. The reinforcement algorithms are then compared to Diwata's current PID control.

The Diwata satellites use reaction wheels for fine attitude control. Hence, the speeds of the four reaction wheels of the satellite are used as the action space of the reinforcement learning algorithms. In order to train and evaluate these algorithms, we use the Mission, Attitude, and Telemetry Analysis (MATA) software. The MATA simulation software is designed from the mathematical model and telemetry data of the Diwata satellites.[8]

In summary, the main contributions of this paper are the following:

- implementation of two deep reinforcement learning algorithms for continuous attitude control using the reaction wheel speed as action space

- development and utilization of MATA simulator for reinforcement learning environment
- a comparison and analysis of attitude control performance between the RL algorithms and Diwata's PID control in different scenarios

BACKGROUND

This section discusses the satellite's dynamics and kinematics, the MATA simulation software, and the reinforcement learning algorithms.

Satellite Dynamics, Kinematics, and Control

The satellite dynamics is simulated using the time derivative of the rotation rate vector $\dot{\vec{\omega}}$, based on the rigid body equation:[8]

$$\dot{\vec{\omega}} = \boldsymbol{I}^{-1}\left(\vec{T_c} + \vec{T_d} - (\vec{\omega} \times \boldsymbol{I}\vec{\omega}) - \left(\vec{\omega} \times \vec{h}_{rw}\right)\right) \quad (1)$$

where $\boldsymbol{I}$ is the moment of inertia matrix, $\vec{T_c}$ the control torque imparted by the attitude controller, $\vec{T_d}$ the environmental disturbance torques (the gravity gradient and atmospheric drag are considered), $\vec{\omega}$ the rotation rate vector, and $\vec{h}_{rw}$ the angular momentum vector of the reaction wheels. The rotation rate, $\vec{\omega}$, is continuously calculated using the Local-Vertical Local-Horizontal (LVLH) frame. The satellite attitude, $\vec{q_e}$, is then updated until it reaches the target quaternion, $\vec{q_t}$.

Satellite kinematics describes the rotation of the satellite without considering external forces. For this research, the kinematics represented by the attitude quaternion is expressed in the following equation:[9]

$$q(t) = \exp\left(\frac{1}{2}\Omega t\right) q(0) \quad (2)$$

The kinematics equation can be expanded to:[9,10]

$$\begin{bmatrix} q_x \\ q_y \\ q_z \\ q_s \end{bmatrix} = \begin{bmatrix} c & n_3 s & -n_2 s & n_1 s \\ -n_3 s & c & n_1 s & n_2 s \\ n_2 s & -n_1 s & c & n_3 s \\ -n_1 s & -n_2 s & -n_3 s & c \end{bmatrix} \begin{bmatrix} q_{x,0} \\ q_{y,0} \\ q_{z,0} \\ q_{s,0} \end{bmatrix} \quad (3)$$

where:

$$c = \cos\left(\frac{1}{2}\|\vec{\omega}\|t\right) \quad (4)$$

$$s = \sin\left(\frac{1}{2}\|\vec{\omega}\|t\right) \quad (5)$$

$$n_i = \frac{w_i}{\|\vec{\omega}\|}, i = x, y, z \quad (6)$$

Meanwhile, fine attitude control of small satellites is usually done using reaction wheels (RW). Reaction wheels are momentum exchange devices commonly used for fine attitude control in small satellites requiring a high degree of pointing accuracy. Because the RWs operate mainly on smaller torque values (typically around 0.01 to 1 Nm), they are desirable actuators for spacecraft systems requiring small control steps.

A reaction wheel is basically a symmetric component consisting of an electrical motor with a flywheel attached to the rotor. The spin of this flywheel produces an angular momentum with respect to the RW's rotation axis. However, since the RW is part of the satellite's body, the angular momentum of the entire spacecraft is conserved. This conservation of momentum is ultimately manifested through a change in the spacecraft's orientation.

As an actuator, a RW can be characterized through the rotation rate (ω_{rw}) of its flywheel and the moment of inertia (I_{rw}) about the spin axis. For the simulation environment used in this paper, the following typical values are used:

- rotation rate: $\omega_{rw} \in [-1500, 1500] rev/min$
- moment of inertia: $I_{rw} = 4.67 \times 10^{-4} kg \cdot m/s^2$

In theory, a minimum of three RWs is required to achieve three-axis stabilization of the satellite. However, in practice, a fourth RW is usually included to provide additional torque, but mainly to introduce a redundancy should one of the units fail.

In the satellite system considered in the simulation environment, the RWs are arranged such that each affects two axes. Mathematically, the effect of each reaction wheel on the body frame can be expressed through the RW alignment matrix, $\mathbf{A}_{rw}$. For this satellite system, a 3x4 matrix is used as the $\mathbf{A}_{rw}$ to represent each of the four RWs.

Ultimately, attitude control aims to minimize the perceived error towards achieving a desired orientation (e.g., pointing towards a certain point on the earth's surface). In terms of using RWs for attitude control, this refers to using an algorithm that utilizes the error variable, $\mathbf{A}_{rw}$, and ω_{rw} to come up with the right control torque vector, $\vec{T}_{c,rw}$.

In traditional small satellites, including the one referred to in this paper, attitude is controlled using the well-known Proportional-Integral-Differential (PID) control algorithm, which can be expressed in vector form as

$$\vec{T_c} = \vec{K_p}(er) + \vec{K_d}\frac{d}{dt}(er) + \vec{K_i}\int (er)dt \quad (7)$$

If we take the error (er) variable from equation 7 as an error quaternion $\vec{q}_{error}$ which is the difference between the current attitude, $\vec{q}_e$, and the target attitude, $\vec{q}_t$, expressed as

$$\vec{q}_{error} = \begin{bmatrix} \vec{v}_{err} \\ r_{err} \end{bmatrix} = \vec{q}_e^{-1} \vec{q}_t \tag{8}$$

the PD control equation can then be derived and rewritten as:[11]

$$\vec{T}_c = 2\ K_p\ \vec{v}_{error}\ r_{error} + K_d\ \vec{\omega} \tag{9}$$

Due to constraints in the resources of the satellite's onboard computing, the control implemented in the satellite is reduced to PD. This is also taken to be true for the simulation system. This limitation is addressed by the addition of a separate system for managing the steady-state error that should have been controlled by the integral constant.

The job of the satellite's onboard computer responsible for attitude control is then simplified to calculate the reaction wheel's future value to achieve the desired target orientation. The next value of the reaction wheel rotation speed, $\vec{\omega}_{rw_i,next}$, is derived from the control torque $\vec{T}_c$.

$\vec{T}_c$ is then decomposed to the torque contribution of each RW using the reaction wheel alignment matrix $\mathbf{A}_{rw}$ mentioned earlier through $\boldsymbol{A}_{T_c \to T_{rw}}$, derived from minimizing an optimizing criterion and the alignment matrix of the satellite (see Sec. 7.3.4 of[11]). The optimizing criterion can be selected to be the norm of the individual RW torques, expressed as a Hamiltonian function:[8]

$$H = \sum_{i=1}^{4} T_{rw_i}^2 \tag{10}$$

$\boldsymbol{A}_{T_c \to T_{rw}}$ can be expressed as a 3×4 matrix translating the satellite's control torque and individual RW control torques as:[8]

$$\begin{bmatrix} T_{rw_1} \\ T_{rw_2} \\ T_{rw_3} \\ T_{rw_4} \end{bmatrix} = \boldsymbol{A}_{T_c \to T_{rw}} \begin{bmatrix} T_{c,x} \\ T_{c,y} \\ T_{c,z} \\ 0 \end{bmatrix} \tag{11}$$

From the derived RW torque values, the corresponding required RW rotation rate can be obtained from the I_{rw}. This process presents the problem of having to optimize the K_p and K_d gains to achieve an optimal response. This approach is also susceptible to changes in assumed values (i.e., usually derived from CAD model) for I_{rw}, $\mathbf{A}_{rw}$, and even I_{sat}, which might change during the course of the spacecraft's life cycle.

MATA Simulation Software

The Mission, Attitude, and Telemetry Analysis (MATA) software is a simulation and visualization tool which can be used by satellite operators and engineers to read or generate telemetry data.[8] It can also be used in a Hardware-In-the-Loop (HIL) mode to test hardware components of an engineering model. Fig. 1 shows the interface of the MATA simulation software.

Figure 1: MATA Simulator[8]

The MATA simulator serves as the training environment of the RL agent since it can simulate satellite dynamics and generate telemetry data of Diwata microsatellites. The simulator was developed in the Unity game engine. With this engine, the researchers leveraged on using Unity's Machine Learning Agents (ML-Agents) Toolkit[12] for training reinforcement learning algorithms.

Reinforcement Learning

Reinforcement learning is an area of machine learning where an agent learns by interacting with its environment.[13] It is usually formulated as a Markov Decision Process (MDP) which consists of an environmental state S, action space A, state-action transition dynamics $T(s_{t+1}|s_t, a_t)$, and reward function $R(s_t, a_t)$.[5,13] At every time step, the agent gathers the observation/environmental states. It then selects the optimal action depending on the calculated reward. The goal of reinforcement learning is to find the optimal policy which maximizes the reward.[4,13]

For the attitude control system, the agent is the satellite and the training environment for our research is the MATA simulator. Given a set of observations from the MATA simulator, the satellite predicts the reaction wheels speed to accomplish the target attitude and stability as shown in Fig. 2. The attitude control system is trained on the Proximal Policy Optimization (PPO)[6] and the Soft Actor Critic (SAC) algorithms.[7]

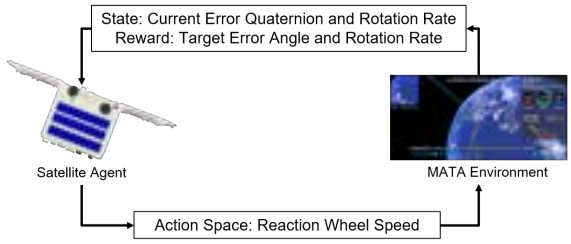

Figure 2: Overview of Attitude Control System using Reinforcement Learning

Both RL algorithms are policy-gradient, actor-critic methods. A policy-gradient method explicitly updates the policy π_θ in every time step. The equations below show a general objective function for the policy-gradient method and its gradient computation.[14,15]

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_t R(s_t, a_t)\right] \tag{12}$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\left(\sum_{t=1}^{T} \nabla_\theta log \pi_\theta(a_t|s_t)\right) R(\tau)\right] \tag{13}$$

In an actor-critic method, the critic updates the action-value $Q^\pi(s,a)$ or state-value $V^\pi(s)$ functions. The equations for these functions are shown below. Equation 16 is the advantage function which measures how good the action is compared to the average actions of the given state.[15] The actor then updates the policy in the direction calculated by the critic.[14]

$$Q^\pi(s,a) = \sum_t \mathbb{E}_{\pi_\theta}\Big[R(s_t, a_t)|s, a\Big] \tag{14}$$

$$V^\pi(s) = \sum_t \mathbb{E}_{\pi_\theta}\Big[R(s_t, a_t)|s\Big] \tag{15}$$

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s) \tag{16}$$

Proximal Policy Optimization (PPO)

PPO is one of the most popular RL algorithms for UAV attitude control because of its great performance and computational simplicity.[15,16] Recently, it has also gained attraction in spacecraft attitude control.[5,17] PPO is an on-policy algorithm which updates the policy in "real-time". This algorithm maximizes the objective function:[6]

$$J^{CLIP}(\theta) = \hat{\mathbb{E}}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \tag{17}$$

where θ is the policy parameter, r_t the ratio of the probability between the new and old policies, $\hat{A}_t$ the estimated advantage function, and ϵ a hyperparameter.[6,15]

Soft-Actor Critic (SAC)

On the other hand, SAC is an off-policy algorithm which is able to reuse previously collected data.[14] Because of this capability, SAC is sample efficient and can maximize the entropy of the policy.[7,14] With high entropy, the agent is encouraged to do more explorations. The entropy objective function of the algorithm is given by:[7,14]

$$J(\theta) = \sum_{t=1}^{T} \mathbb{E}_{\pi_\theta}[R(s_t, a_t) + \alpha H(\pi_\theta(\cdot|s_t))] \tag{18}$$

where H is the entropy function and α the temperature parameter which controls the stochasticity of the policy.[18] To the best of our knowledge, this research is the first to experiment on using the SAC algorithm to satellite attitude control.

METHODOLOGY

The formalization of the reinforcement learning problem for satellite attitude control is presented in this section.

State and Action Spaces

The goal of the satellite agent is to stabilize at its target orientation. For this research, the state vector is defined as $s_t = \{\vec{q}_{error}, \vec{wbi}, \vec{wbr}\}$ where $\vec{q}_{error}$ is the error quaternion, $\vec{wbi}$ the rotation rate of the satellite with respect to the inertial frame, and $\vec{wbr}$ the rotation rate of the satellite with respect to the reference frame.

The action space, $a_t = \{RW1, RW2, RW3, RW4\}$, is the four reaction wheel speeds of the Diwata satellites. The reaction wheel's maximum speeds are set to 1500 rpm. Additionally, the maximum ΔRW speed is set to 10 rpm.

A single pass for Diwata satellites is around 6 - 10 minutes.[19] For this research, the maximum 10 minute pass is simulated per episode. Initial parameters per episode are then randomized in order to simulate different scenarios of a satellite pass. Table 1 shows the limits of the parameters set per episode. The target rotation is randomized per episode while the target rotation rates are set to zero.

Table 1: Initial Parameters for Each Episode

Parameter	Description	Bounds of Initial Conditions
$\vec{qe}$	Rotation from the Reference Frame to the Body Frame	$\pm 360°$
$\vec{qt}$	Target Rotation	$\pm 360°$
$\vec{wbi}$	Rot. Rate of the Sat. Body wrt the Inertial Frame	$\pm 10°/s$
$\vec{wbr}$	Rot. Rate of the Sat. Body wrt the Reference Frame	$\pm 10°/s$

Reward Function

The reward function for this research is influenced by Deep Mimic's velocity reward.[20] It is an exponential function which gives the agent more rewards when it is near the target. The satellite agent's reward function consists of two components: the quaternion reward and the rotation rate reward as shown in the equations below. Both components use Euclidean distance to measure the error quaternion and rotation rates. Instead of using the sum of the two rewards, the total reward is their product. This would help the agent to maximize the reward[12] and make sure that the target quaternion and the target rotation rate are achieved at the same time. The agent is given an additional reward when $q_{error} < 0.1°$, which makes sure the satellite stays on target during the pass.

$$Q_{reward} = \exp[-0.1(\|\vec{q}_{target} - \vec{q}\|)] \tag{19}$$

$$W_{reward} = \exp[-0.1(\|\vec{w}_{target} - \vec{w}\|)] \tag{20}$$

$$Reward_{total} = Q_{reward} * W_{reward} \tag{21}$$

Implementation Details

For fair comparison of the two RL algorithms, the following hyperparameters were set during the agent's training: batch size = 512, initial learning rate = 0.0001, 2 hidden layers with 128 neurons, and a discount factor of 0.995. The agent's training ends after 30M timesteps. The RL algorithms are implemented in Unity ML-Agents with Tensorflow as backend using a computer with i9 CPU, 32GB RAM, and NVIDIA RTX 2080 GPU.

EVALUATION

This section discusses the results for the RL experiments. The comparison of the two RL algorithms is first presented. An evaluation of different case studies for the attitude controllers is then demonstrated.

Training Results

As shown in Fig. 3, the SAC algorithm achieved a higher cumulative reward (≈ 450) than the PPO (≈ 410). SAC's ability to reuse previously collected data enabled it to become more sample efficient, hence, faster convergence. SAC reached convergence around 15M steps while the PPO needed 30M steps to achieve convergence. Based on the training curves below, SAC is the best RL algorithm for attitude control in terms of the reward function. A possible future work in training RL algorithms for attitude control is the exploration of other observation states. A reduction of observation states could improve the cumulative reward and achieve faster convergence.[15] Another possible future work is to train the satellite agent without reward engineering.

Figure 3: Cumulative Reward of the RL Algorithms during Training for 30M Steps

Case Studies

To test the control performance of the satellite agent, different scenarios were simulated using the MATA simulator. The performance plots are presented in each scenario and the following quantitative metrics are utilized: settling time[15] (time it takes the agent to settle within 2% the target values), overshoot[15,16] (the peak value represented as a percentage relative to the steady state error), and stability[16] (the root mean square error which shows how stable the response is after the settling time).

For consistency, the following initial parameters are set for the comparison of the RL algorithms and Diwata's PID controller: $\vec{qe} = \langle 30°, 60°, 90° \rangle$, $\vec{qt} = \langle 0°, 0°, 0° \rangle$, $\vec{wbi} = \langle 0.1°/s, 1°/s, 2°/s \rangle$, and $\vec{wbr} = \langle 0.1°/s, 1°/s, 2°/s \rangle$.

Scenario 1: Diwata 2

The satellite agent is trained using Diwata 2's stowed inertia matrix, $\mathbf{I} = diag(1.977, 2.032, 2.232)\ kg \cdot m^2$. Hence, the stowed configuration of the Diwata 2 satellite (as shown in Fig. 4) would serve as baseline scenario for the attitude controllers. Fig. 5 and Fig. 6 show the control performance plots for Diwata 2's attitude angles and rotation rates, respectively.

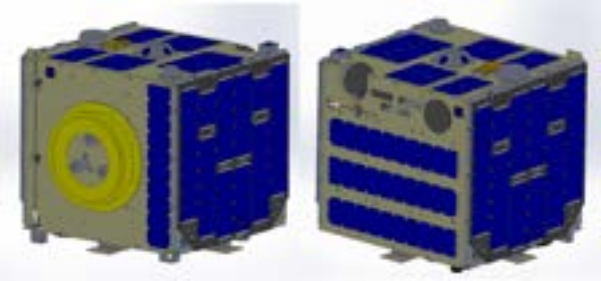

Figure 4: Diwata 2 Stowed Configuration[21]

As shown in Table 2, the algorithm which achieved the fastest settling time is the SAC algorithm with an average of 87 s. The PID controller achieved an average of 158 s while the worst performing algorithm, PPO, has an average of 224 s settling time. Meanwhile, in terms of stability, SAC has the lowest stability metric for attitude angles. On the other hand, the PID control has the lowest stability metric for rotation rate. Additionally, the SAC controller has the smallest overshoot for attitude angles while the PID has the smallest overshoot for rotation rate.

Overall, the PPO is the worst performing algorithm for the Diwata 2 stowed configuration. Moreover, the SAC algorithm is comparable to the PID controller in terms of stability. For this scenario, SAC is the fastest attitude controller.

Table 2: Evaluation Metrics for Diwata 2

Metrics	PPO	**SAC**	PID
Set. Time (s): Attitude Angles	221	**85**	169
Set. Time (s): Rot. Rate	228	**89**	147
Stability (rms): Attitude Angles	0.49885	**0.23026**	0.42607
Stability (rms): Rot. Rate	0.00409	0.00193	**0.00153**
Overshoot (%): Attitude Angles	155.024	**8.71910**	18.9224
Overshoot (%): Rot. Rate	26.8821	20.2030	**11.2179**

Scenario 2: Diwata 2 Configuration with Deployed Solar Panels and Antenna

The second scenario is a simulation of the deployment of the solar panels and antenna of Diwata 2 as shown in Fig. 7. This simulation would test the performance of the attitude controllers to disturbances and changes in inertia matrix. The inertia matrix of Diwata 2 would change to $\mathbf{I} = diag(0.950, 0.920, 0.970)\ kg \cdot m^2$ after deployment. The deployment starts at 300 s as shown in the plots of Fig. 8 and Fig. 9.

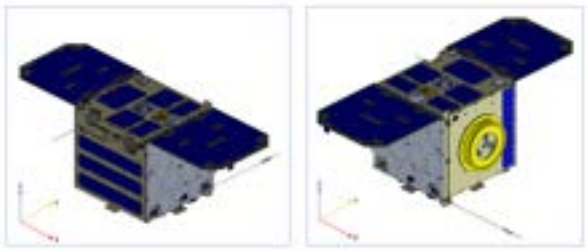

Figure 7: Diwata 2 Deployed Configuration[21]

Table 3 shows the evaluation metrics for the deployed configuration. The overshoot metric is measured after the 300 s deployment to test the robustness of the attitude controllers to sudden disturbances. For this scenario, the PPO is the fastest and has the smallest overshoot. Meanwhile, the SAC and the PID is the most stable in terms of the attitude angles and rotation rate, respectively. The PPO algorithm is more robust to sudden disturbance because of its "realtime" policy update.

Table 3: Evaluation Metrics for Diwata 2 Deployed Configuration (at t = 300 s)

Metrics	**PPO**	SAC	PID
Set. Time (s): Attitude Angles	**221**	381	421
Set. Time (s): Rot. Rate	**250**	400	350
Stability (rms): Attitude Angles	0.81182	**0.58237**	0.96761
Stability (rms): Rot. Rate	0.00901	0.00536	**0.00333**
Overshoot (%): Attitude Angles	**5.37351**	75.6646	16.7054
Overshoot (%): Rot. Rate	**4.71183**	25.2574	16.2846

Scenario 3: Diwata 1 (Similar Flight Heritage and Mass with Diwata 2)

The third scenario demonstrates that RL algorithms still work well without re-tuning the param-

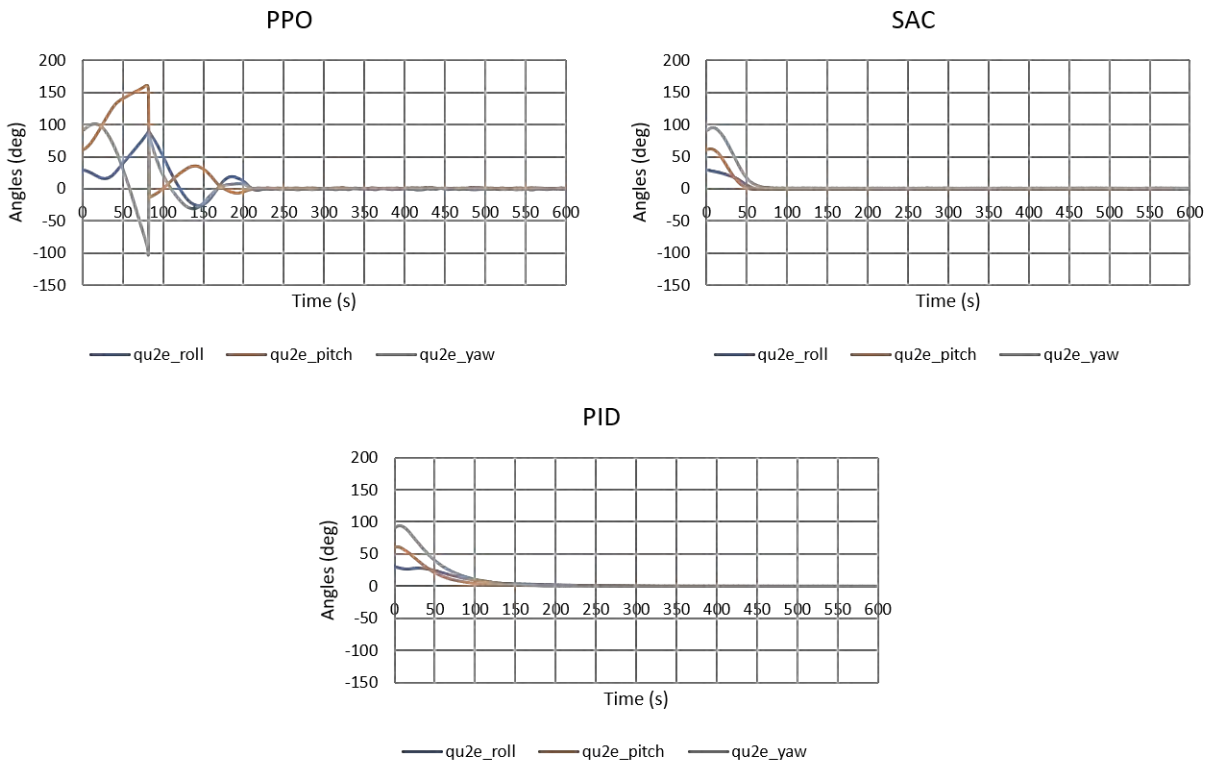

Figure 5: Control Performance for Diwata 2's Attitude Angle

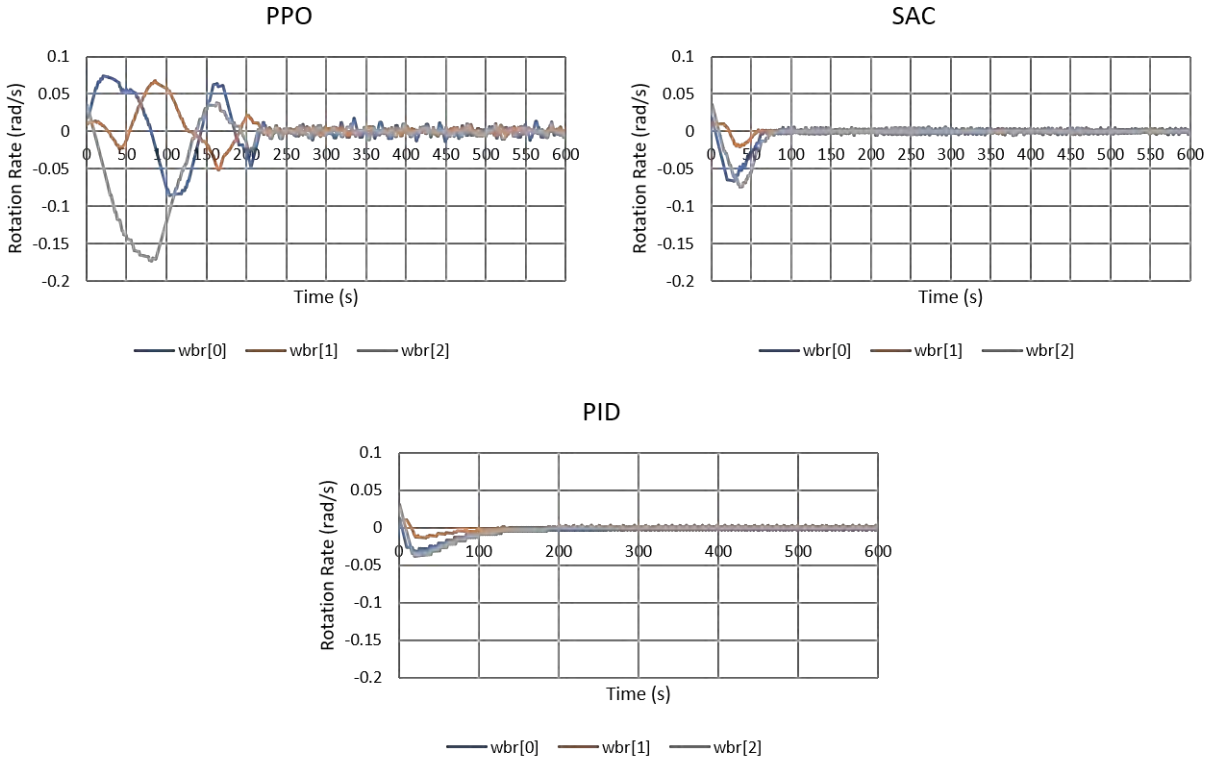

Figure 6: Control Performance for Diwata 2's Rotation Rate

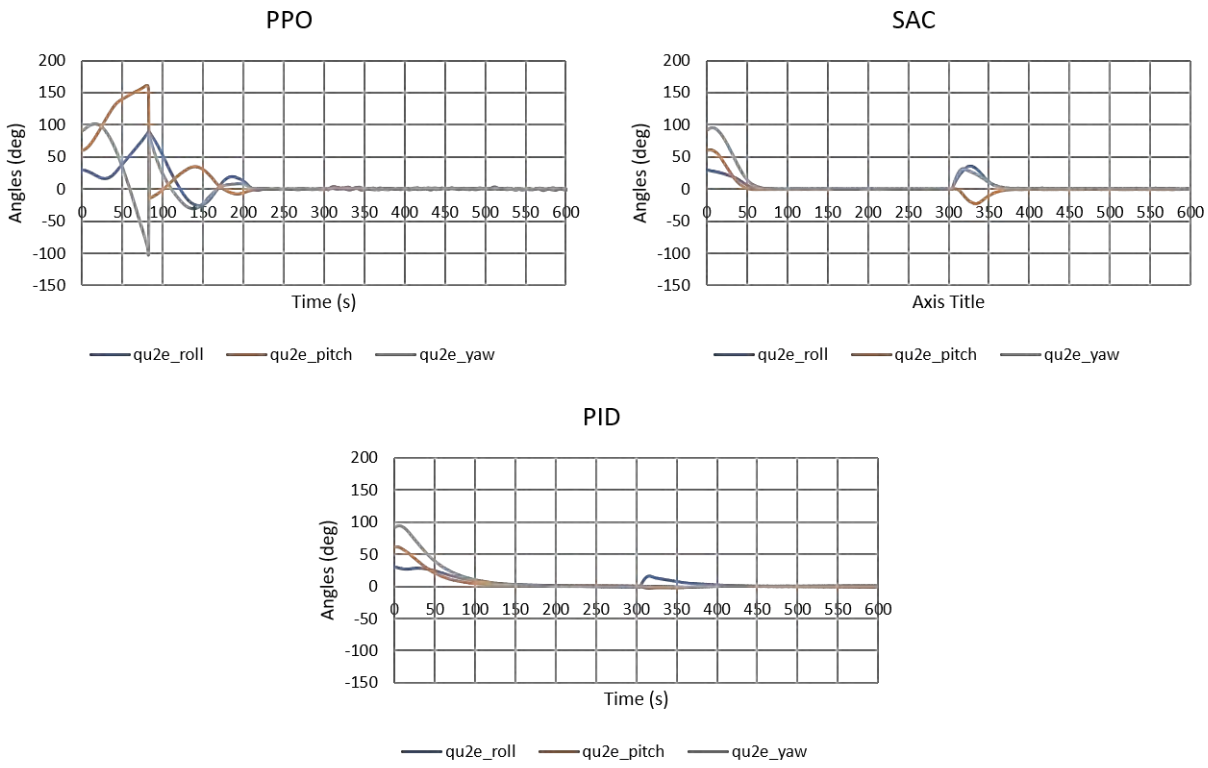

Figure 8: Control Performance for Diwata 2's Attitude Angle with Deployed Configuration

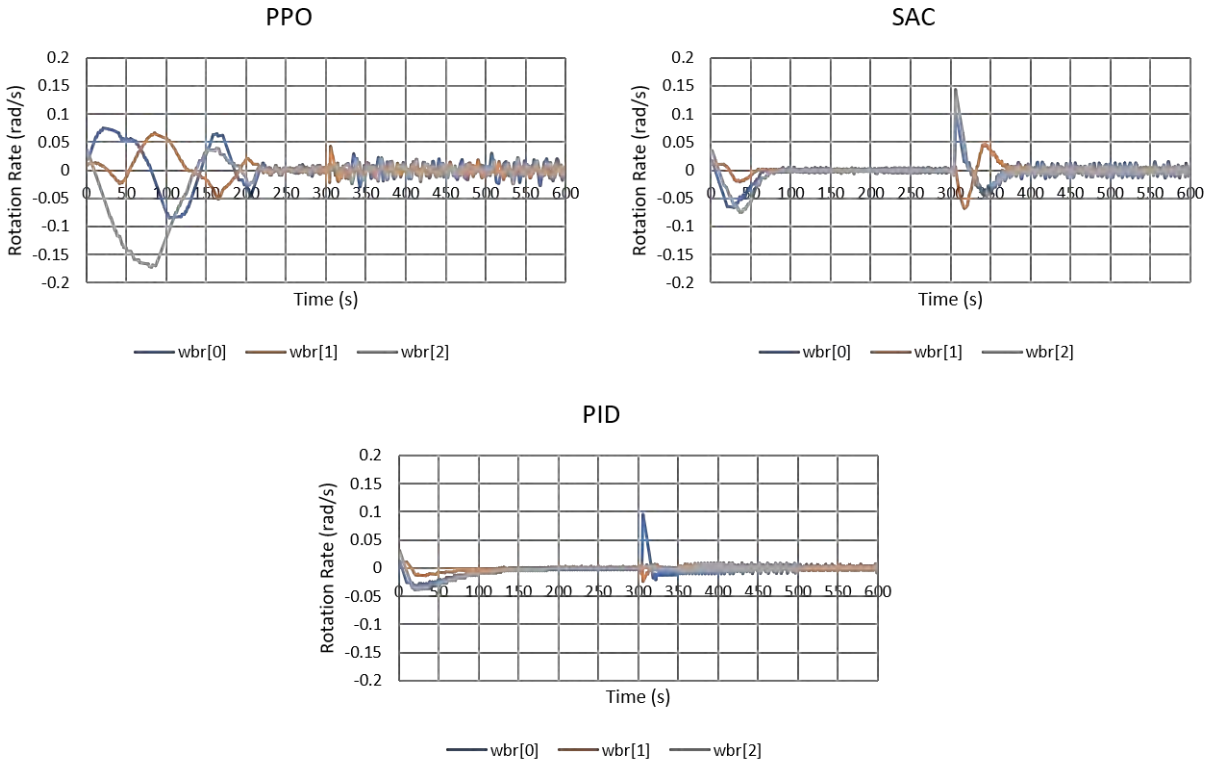

Figure 9: Control Performance for Diwata 2's Rotation Rate with Deployed Configuration

eters. The Diwata 1 satellite (as shown in Fig. 10), which has a similar flight heritage with Diwata 2, is utilized for this scenario. Both satellites have four reaction wheels in their attitude control systems.[22,23] Diwata 1 is 50 kg satellite[22] while Diwata 2 is a 56 kg satellite.[23] The inertia matrix of Diwata 1 is $\mathbf{I} = diag(1.098, 0.749, 0.814)\ kg \cdot m^2$.

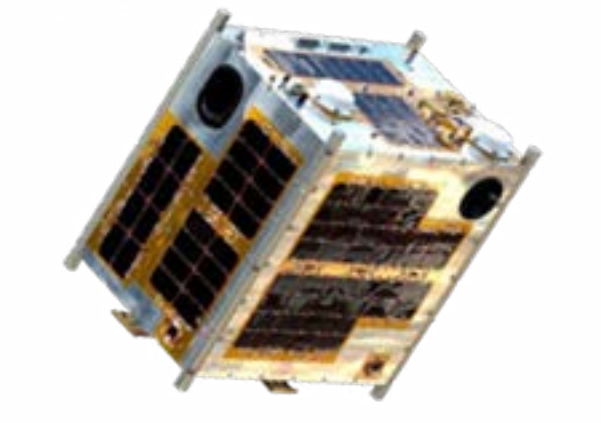

Figure 10: Diwata 1 Model

As shown in Fig. 11 and Fig. 12, the RL algorithms can still achieve the target attitude angles and rotation rate without retraining or re-tuning the parameters of the algorithms. The PID controller for this scenario was re-tuned to demonstrate how the untuned RL algorithms perform against Diwata 1's actual on-board attitude controller. Table 4 shows that the SAC algorithm is the fastest controller and has the smallest overshoot. The PID controller for this scenario is the most stable

Table 4: Evaluation Metrics for Diwata 1

Metrics	PPO	SAC	PID
Set. Time (s): Attitude Angles	128	**60**	126
Set. Time (s): Rot. Rate	138	**71**	134
Stability (rms): Attitude Angles	1.21973	0.31388	**0.19524**
Stability (rms): Rot. Rate	0.00781	0.00586	**0.00392**
Overshoot (%): Attitude Angles	54.9796	**2.08016**	17.3205
Overshoot (%): Rot. Rate	24.9663	**8.55225**	35.4689

Scenario 4: LM 50 (Different Flight Heritage and Mass with Diwata satellites)

LM 50 is a nano-satellite bus system weighing 10 - 100 kg.[24] As shown in Fig. 13, this satellite was used by Elkins et al.[5,25] for their RL experiments. The authors only provided the inertia matrix: $\mathbf{I} = diag(0.872, 0.115, 0.797)\ kg \cdot m^2$.[25] Hence, the simulation environment of the last scenario for this research does not represent the actual behaviour of the satellite. The last scenario is a demonstration how the attitude controllers can still perform even if they are trained with a different flight heritage and mass properties. The PID controller for this scenario was re-tuned so it can reach the target attitude angles and rotation rate. However, it does not reflect the actual on-board controller used by LM 50.

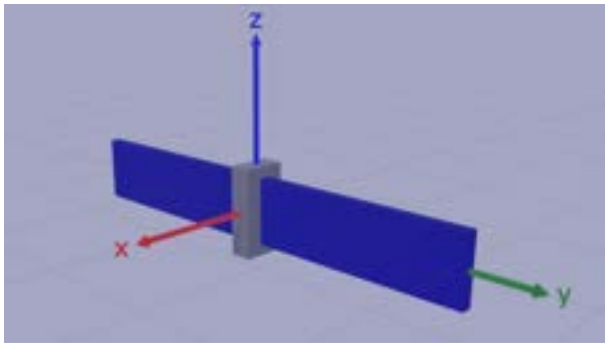

Figure 13: LM50 Model[5,25]

Fig. 14 and Fig. 15 show the control performance plots for LM 50's attitude angles and rotation rates, respectively. It can be seen that the controllers for the LM 50 has a longer settling time than the Diwata satellites. Table 5 shows that the SAC algorithm is still the fastest algorithm among the attitude controllers. Furthermore, it is the most stable and has the smallest overshoot.

Table 5: Evaluation Metrics for LM 50

Metrics	PPO	SAC	PID
Set. Time (s): Attitude Angles	250	**150**	200
Set. Time (s): Rot. Rate	250	**175**	200
Stability (rms): Attitude Angles	2.60921	**0.49835**	0.54162
Stability (rms): Rot. Rate	0.00654	**0.00313**	0.00355
Overshoot (%): Attitude Angles	36.0648	**2.65233**	64.1698
Overshoot (%): Rot. Rate	46.1414	**19.2666**	120.238

Overall Results

As shown in the previous section, SAC is the fastest attitude controller when no sudden disturbances occur. It is also comparable with the PID

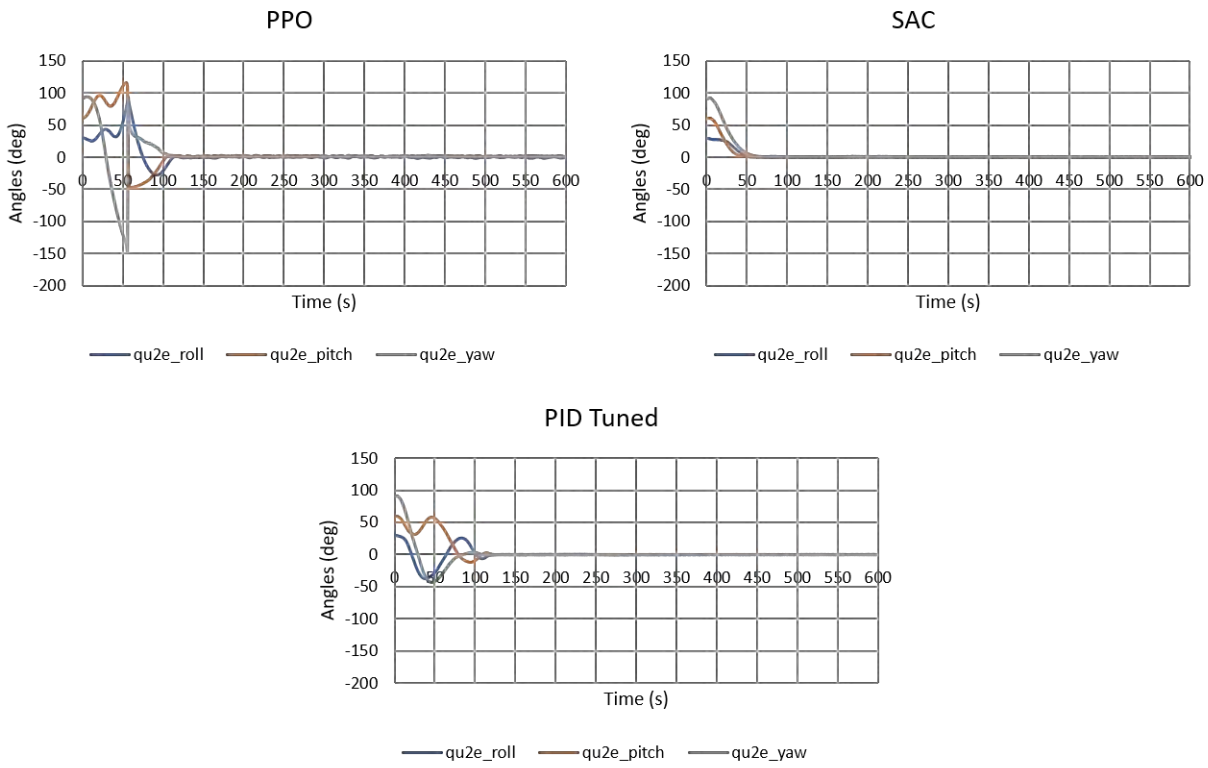

Figure 11: Control Performance for Diwata 1's Attitude Angle

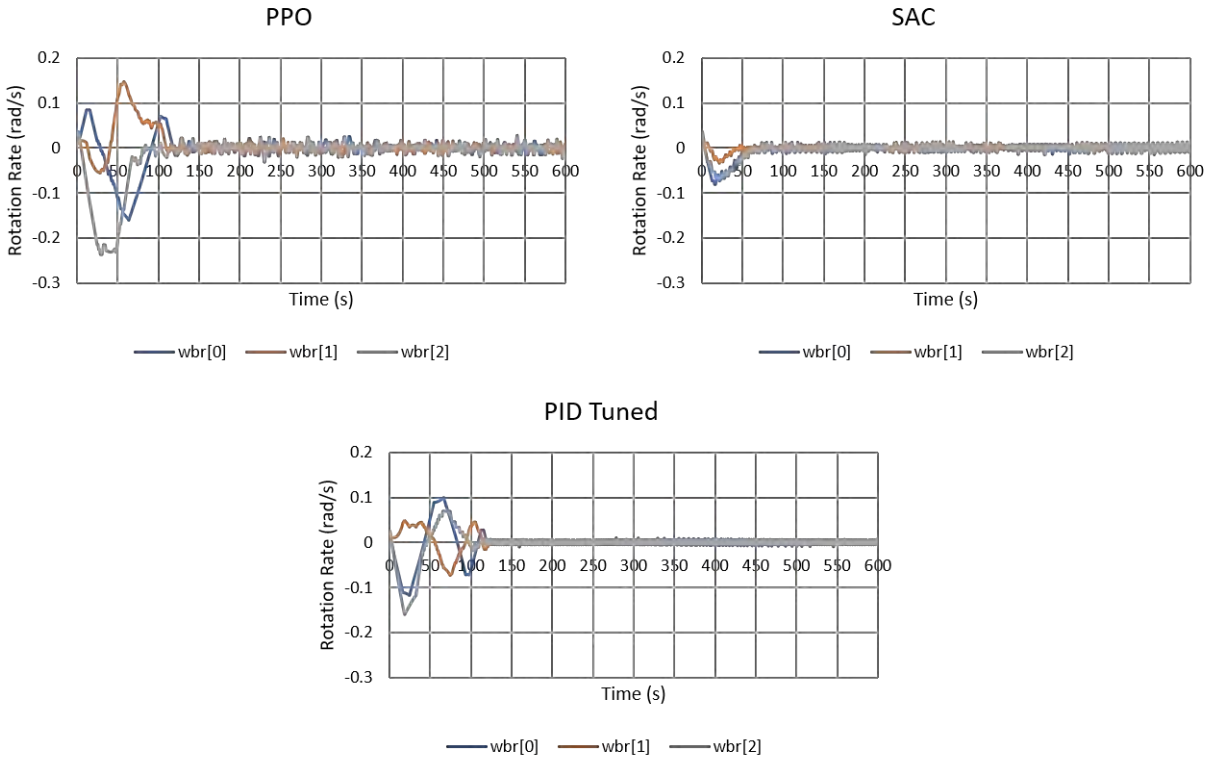

Figure 12: Control Performance for Diwata 1's Rotation Rate

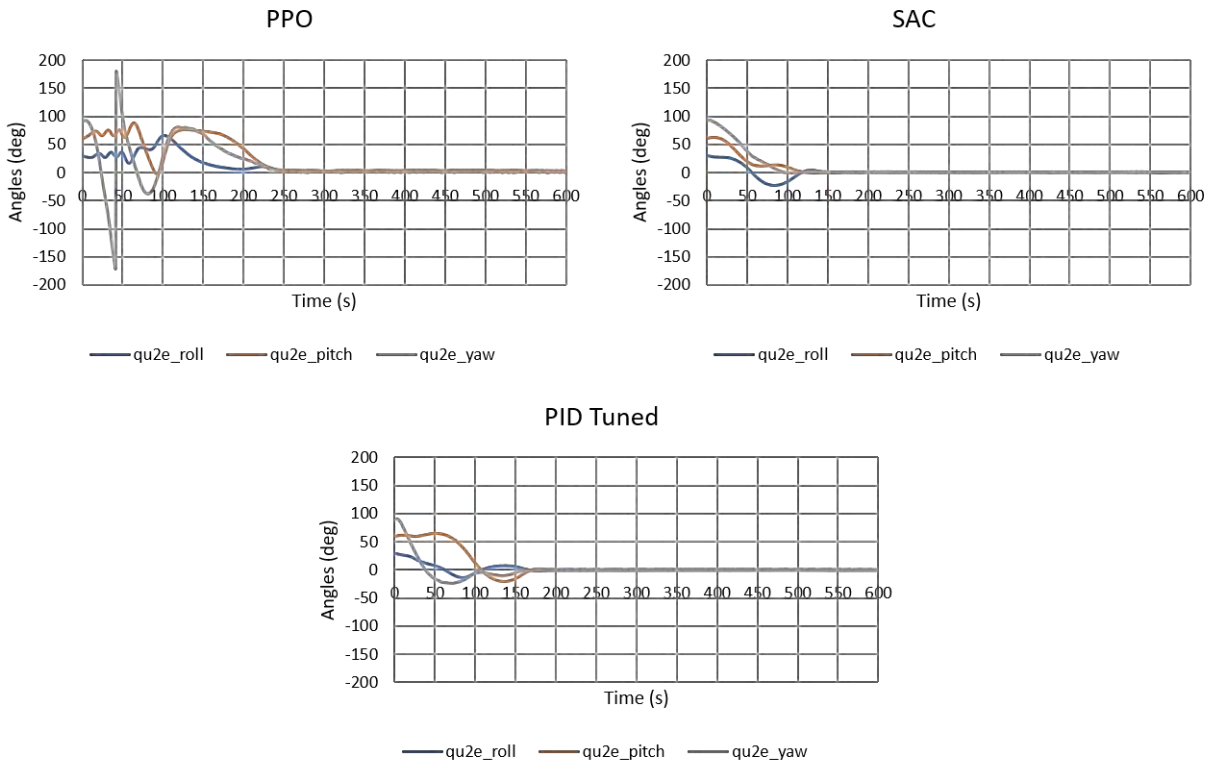

Figure 14: Control Performance for LM 50's Attitude Angle

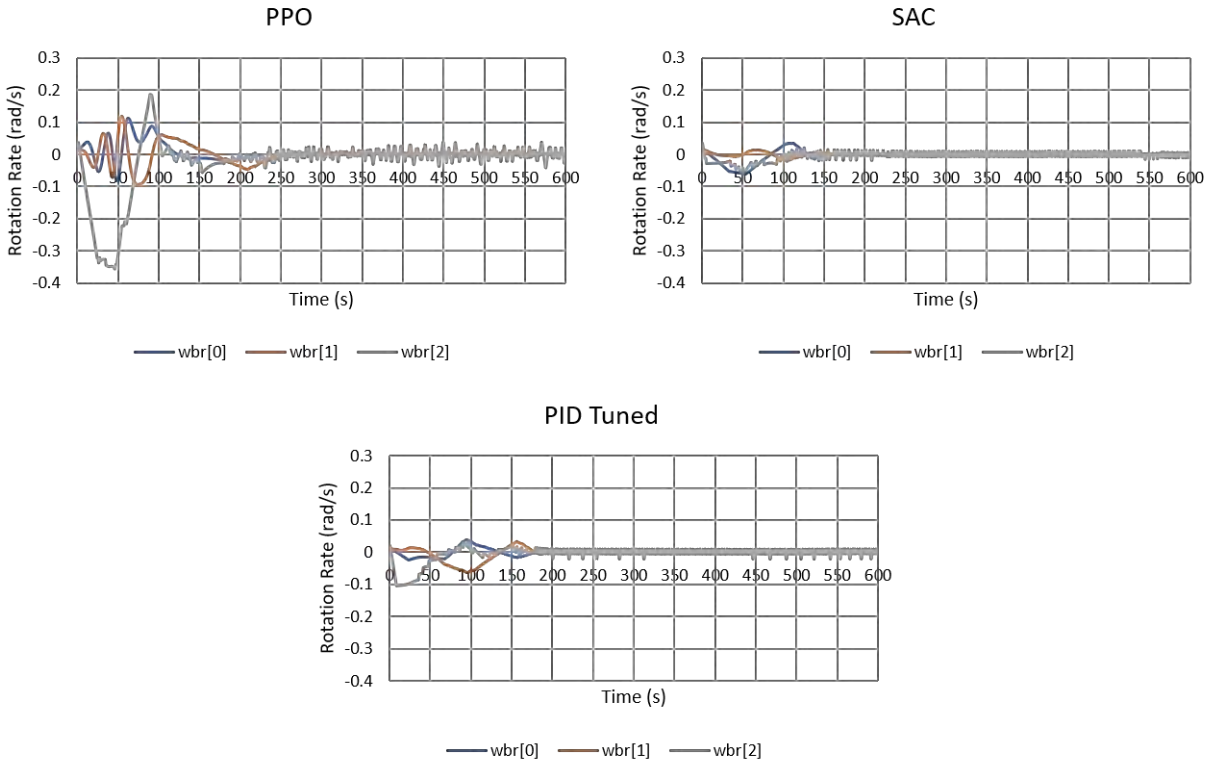

Figure 15: Control Performance for LM 50's Rotation Rate

controller in terms of the stability and overshoot metrics. The case studies also showed that PPO has the worst performing metrics, however, it is the most resilient to sudden disturbances.

The case studies only showed scenarios with fixed initial parameters. To get a general overview of the performance of the attitude controllers, additional evaluation was implemented. For the overall evaluation, the Diwata 2 satellite was utilized. The initial parameters, $\vec{qe}, \vec{qt}, \vec{wbi}$, and $\vec{wbr}$, were randomized for each episode. Similar with training the agent, the target rotation rates for evaluation were set to zero. The metrics used for the overall evaluation are the following: minimum error angle, minimum rotation rate, and the times when the minimum error angles and rotation rate occurred. Table 6 shows the average evaluation metrics for 5000 episodes. To asses which attitude controllers perform statistically better, a pair-wise post-hoc Conover-Inman test is implemented.[26] Fig. 16 and Fig. 17 show the box-plots for the statistical test. Same colored box-plots indicate that the metrics are not statistically significant. It can be seen that the SAC and PID controllers are comparable in terms of the minimum error angle. The results also show that the PID and PPO are comparable in terms of the time the rotation rate reaches its minimum. Overall, SAC is the best performing algorithm in terms of reaching the minimum error angles and rotation rates. It is also consistent with the case studies that the SAC is the fastest attitude controller.

Table 6: Average Results over 5000 Episodes

Method	Error Angle (deg)	Error Angle Time (s)	Rot. Rate (rad/s)	Rot. Rate Time (s)
PID	0.627830	422.741	0.001145	396.689
SAC	**0.561143**	**330.539**	**0.000730**	**363.754**
PPO	1.149576	388.152	0.000945	400.478

CONCLUSION AND FUTURE WORK

This paper demonstrates two Reinforcement Learning algorithms for satellite attitude control. The algorithms are then compared to Diwata's current PID controller. The attitude controllers were evaluated to different scenarios and satellite configurations. The results show that the RL algorithms can outperform the traditional controller.

There are different criteria in choosing which RL algorithm fits the intended attitude control. If the priority of the satellite is to be robust in sudden disturbances, the best algorithm is the PPO. For fast attitude target, the best RL algorithm is the SAC. It is also the most comparable algorithm with the PID controller in terms of stability.

A RL algorithm with the combined features of PPO and SAC can be explored for future work. The researchers would also explore RL algorithms without reward engineering. Reward functions vary in different works for attitude control. If the agent could predict without relying on rewards, a more generalized satellite agent can be developed. Furthermore, the researchers would investigate how to implement and test these RL algorithms in an engineering model.

Acknowledgments

This research is funded by the Department of Science and Technology (DOST) under the Philippine Council for Industry, Energy, Emerging Technology, Research and Development (PCIEERD). The authors would like to thank the PHL-50 team for their support and contributions.

References

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

[2] Wang Y., Ma Z., Yang Y., Wang Z., and Tang L. A new spacecraft attitude stabilization mechanism using deep reinforcement learning method. In *8TH European Conference for Aeronautics and Space Sciences (EUCASS)*, 2019.

[3] Huo B., Xia Y., Yin L., and Fu M. Fuzzy adaptive fault-tolerant output feedback attitude-tracking control of rigid spacecraft. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(8):1898–1908, 2017.

[4] Su R., Wu F., and Zhao J. Deep reinforcement learning method based on ddpg with simulated annealing for satellite attitude control system. In *2019 Chinese Automation Congress (CAC)*, pages 390–395, 2019.

[5] Elkins J., Sood R., and Rumpf C. Autonomous spacecraft attitude control using deep reinforcement learning. In *71st International Astronautical Congress*, October 2020.

[6] Schulman J., Wolski F., Dhariwal P., Radford A., and Klimov O. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

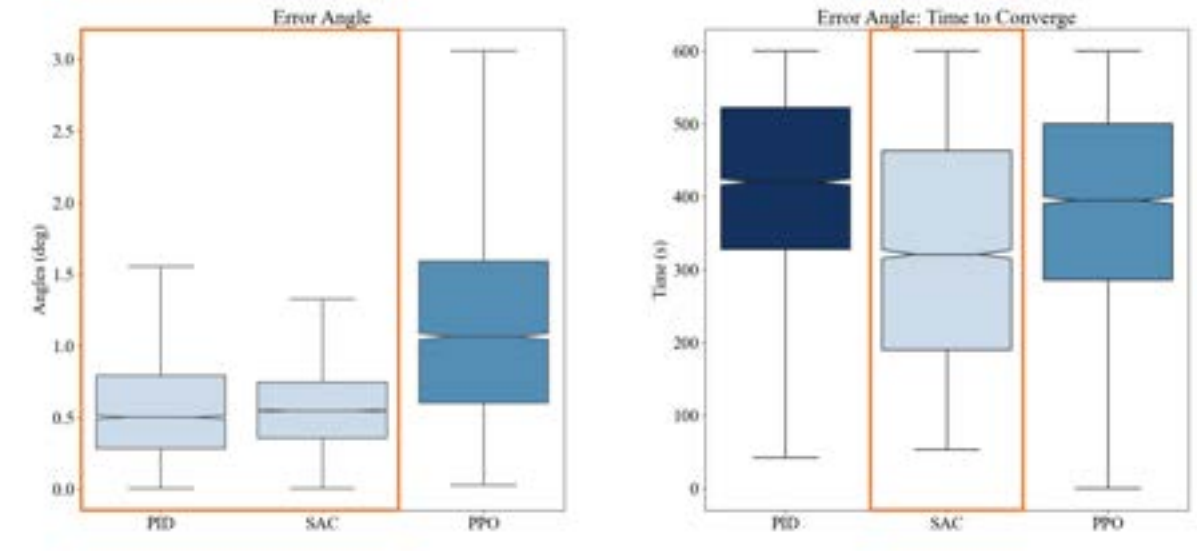

Figure 16: Overall Results for Minimum Error Angle

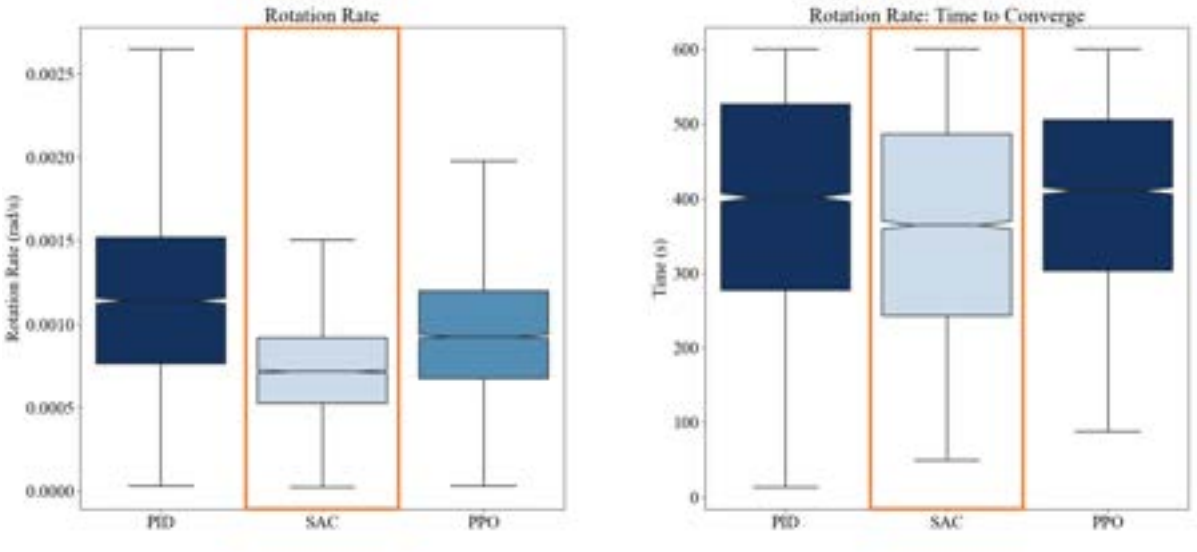

Figure 17: Overall Results for Minimum Rotation Rate

[7] Haarnoja T., Zhou A., Abbeel P., and Levine S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290, 2018.

[8] Tan V., Labrador J.L., and Talampas M.C. Mata: Mission, attitude, and telemetry analysis software for micro-satellites. In *2020 IEEE Region 10 Conference (TENCON)*, pages 614–619, 2020.

[9] Labrador J.L. Development of an improved micro-satellite simulator environment for attitude determination and control system verification through diwata-1 flight data analysis. submitted, January 2017.

[10] Wertz J.R. Spacecraft attitude determination and control. *Astrophysics and Space Science Library*, 73, January 1978.

[11] Sidi M.J. *Spacecraft Dynamics and Control: A Practical Engineering Approach*. Cambridge Aerospace Series. Cambridge University Press, 1997.

[12] Juliani A., Berges V.P., Vckay E., Gao Y., Henry H., Mattar M., and Lange D. Unity: A general platform for intelligent agents. *CoRR*, abs/1809.02627, 2018.

[13] Arulkumaran K., Deisenroth M.P., M. Brundage, and A.A. Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, November 2017.

[14] Weng L. Policy gradient algorithms. *lilianweng.github.io/lil-log*, 2018.

[15] Bøhn E., Coates E. M., Moe S, and Johansen T. Deep reinforcement learning attitude control of fixed-wing uavs using proximal policy optimization. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 523–533, 2019.

[16] Koch W., Mancuso R., West R., and Bestavros A. Reinforcement learning for uav attitude control. *ACM Trans. Cyber-Phys. Syst.*, 3(2), February 2019.

[17] Allison J., West M., Ghosh A., and Vedant F. Reinforcement learning for spacecraft attitude control. In *70th International Astronautical Congress*, October 2019.

[18] Barros G.M. and Colombini E.L. Using soft actor-critic for low-level UAV control. *CoRR*, abs/2010.02293, 2020.

[19] Retamar A. Establishment of a multi-mission ground receiving station for the philippines. In *SpaceOps 2016 Conference*, 2016.

[20] Peng X.B., Abbeel P., Levine S., and van de Panne M. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *CoRR*, abs/1804.02717, 2018.

[21] PHL-Microsat. Diwata-2. *https://phl-microsat.upd.edu.ph/diwata2*.

[22] Mitchao D., Totani T., Sakamoto Y., Wakita M., and Nagata H. Thermal design and on-orbit validation of the first philippine micro-satellite: Diwata-1. In *47th International Conference on Environmental Systems*, July 2017.

[23] Gonzalez A., Labrador J.L., Mitchao D., Talampas M.C., and Marciano J.J.J. Diwata-2: Earth observation microsatellite with a compact bus system, electronically tunable multispectral imager, and amateur radio communications capability. In *34th Annual Small Satellite Conference*, 2020.

[24] Fabey M. Lockheed martin unveils new satellite bus lineup. *https://spacenews.com/*, September 2017.

[25] Elkins J., Sood R., and Rumpf C. Adaptive continuous control of spacecraft attitude using deep reinforcement learning. In *2020 AAS/AIAA Astrodynamics Specialist Conference*, August 2020.

[26] Conover W.J. and Iman R.L. Multiple-comparisons procedures. informal report. February 1979.