AI-based attitude control for restricted reaction wheels

Kirill Djebko[1, 2], Tom Baumann[2], Erik Dilger[2], Frank Puppe[1], Sergio Montenegro[2]

[1]Julius-Maximilians-Universität Würzburg, Center for Artificial Intelligence and Data Science
Emil-Fischer-Straße 50, 97074 Würzburg, Germany
Phone: +49 931 31 86405, Mail: kirill.djebko@uni-wuerzburg.de

[2]Julius-Maximilians-Universität Würzburg, Chair of Computer Science VIII, Aerospace Information Technology
Emil-Fischer-Straße 70, 97074 Würzburg, Germany

Abstract: Traditional attitude control algorithms often require manual tuning and may underperform if operating conditions deviate from expectations, necessitating corrections in orbit. Hardware limitations can further compound these challenges. During the design of an AI-based attitude controller for the InnoCube 3U CubeSat, jointly developed by the Julius-Maximilians-Universität Würzburg and the Technische Universität Berlin for a technology demonstration mission, we encountered such a limitation: a wheel-rate restriction in the range of approximately -350 to 350 RPM. Prolonged operation in this range is discouraged due to the potential risk of inaccurate wheel rate estimation and control. In response, we incorporated magnetorquers for momentum management. Approaches using Deep Reinforcement Learning (DRL) offer adaptability, but the differing time horizons of the two actuators, spanning about two orders of magnitude, increase training complexity. To address this, we first train an AI agent for the dynamic reaction wheels in isolation, before adding momentum management capabilities by freezing the reaction wheel agent and training a magnetorquer agent, synergizing with the behavior of the reaction wheel agent. We evaluate our approach by simulating attitude control for the InnoCube CubeSat under realistic conditions.

1. INTRODUCTION

Based on our work in [1, 2], we are investigating not only the possibility of making an AI-based attitude controller robust to variations in the moments of inertia, but also its transfer beyond the boundaries of the training configuration. We use Deep Reinforcement Learning (DRL), a variant of classical Reinforcement Learning (RL) [3] using neural networks to process inputs, estimate the quality of the current policy, and to generate control signals. A disadvantage of DRL in general results from neural networks' inherent tendency to be good at interpolation but poor at extrapolation [4]. Therefore, during training, the operating conditions need to be varied in order to avoid out-of-distribution errors. Such conditions include moments of inertia, which we have successfully addressed in [1]. Building on this, we adapt the AI-based controller to the InnoCube 3U Nanosatellite, jointly developed by the Julius-Maximilians-Universität Würzburg and the Technische Universität Berlin [5, 6], which launched successfully on January 14, 2025. The goal is to train an AI-based attitude controller on the ground and then upload it to the satellite after it has finished its main mission, expected to take place in Q3 2025. While it was originally intended to perform inertial-pointing exclusively with the reaction wheels (RWs), we faced an unforeseen restriction regarding the wheel speeds. Prolonged operation of the wheels in the range of [-350, 350] RPM is discouraged, as it may cause inaccurate wheel rate estimation and control. As a precautionary measure, we therefore decided to include this restriction in the AI-based controller by incorporating momentum management using the magnetorquers (MTs). This, however, greatly increased training

complexity, as both actuators have vastly different requirements for training, which will be laid out in the following sections.

2. METHODOLOGY

For training, the Stable-Baselines3 [7] implementation of the Proximal Policy Optimization (PPO) DRL algorithm [8] within the Gymnasium framework [9] was employed and subsequently modified. The simulator used was the Basilisk Astrodynamics Simulation Framework [10].

2.1 Simulation Model

A model of the InnoCube satellite was created and parameterized based on its components' datasheets and CAD data for the moments of inertia. Table 1 lists the obtained key features relevant to the actuators.

Feature	Value
Satellite moments of inertia	[0.0427, 0.0427, 0.0068] $kg \cdot m^2$
RW max omega	[16384.0, 16384.0, 16384.0] RPM
RW inertia	[56.8, 56.8, 56.8] · 1e-6 $kg \cdot m^2$
RW min torque	[0.00001, 0.00001, 0.00001] Nm
RW max torque	[0.002, 0.002, 0.002] Nm
MT max dipole moment	[0.2, 0.2, 0.2] Am^2

Table 1: Key model parameters relevant to the actuator dynamics.

As operating conditions, the initial wheel speeds were randomly set to -500 or 500 RPM for each wheel independently, with the satellite being in a detumbled state. Further, during training, the satellite's moments of inertia were varied by $\pm$10% per axis. The wheel-rate target was chosen to be $\pm$500 RPM to add a buffer to the required $\pm$350 RPM. The target wheel rate was set dynamically depending on which value was closer to the current wheel rate. The satellite's orbit was parameterized and simulated based on orbital elements derived from InnoCube TLE data obtained from CelesTrak [11]. Additionally, the orbit parameters were varied by an appropriate magnitude, based on the obtained data. Table 2 lists the orbital parameters of the InnoCube satellite and the applied variations.

Feature	Value	Variation (offset)
Attitude	Perigee: 508 km Apogee: 519 km	$U\sim$ (-5, 5) km
Eccentricity	7.630e-4	$U\sim$ (-1.0e-4, 3.0e-4)
Inclination	97.43°	$U\sim$ (-0.03, 0.03)°
RAAN, Periapsis, True anomaly	$U\sim$ (0, 360)°	

Table 2: Orbital parameters of the InnoCube satellite and the applied variations.

The time resolution of the simulation was one second. All operating conditions and variations were applied during training and evaluation.

2.2 Observation and Action Space

The observation space was defined as a 39-element vector consisting of the features listed in Table 3.

Feature (length)	Description	Feature (length)	Description
$err\_quat_{t_i}$(4)	Current attitude error quaternion	$mag\_b_{t_i}$ (3)	Current B-field (via magnetometers)
$err\_rr_{t_i}$ (3)	Current satellite rotation rate error	$mag\_b_{t_{i-1}}$ (3)	Previous B-field (via magnetometers)
$a\_rw_{t_{i-1}}$ (3)	Previous commanded reaction wheel torques	$err\_rwr_{t_i}$ (3)	Current RW rate error
$err\_quat_{t_{i-1}}$ (4)	Previous attitude error quaternion	$err\_rwr_{t_{i-1}}$ (3)	Previous RW rate error
$err\_rr_{t_{i-1}}$ (3)	Previous satellite rotation rate error	crm_t_i (3)	Cross product of current wheel rates and measured B-field divided by 2
$rw\_r_{t_i}$ (3)	Current RW rates	bn_{t_i}(1)	Norm of current measured B-field
$rw\_r_{t_{i-1}}$ (3)	Previous RW rates		

Table 3: Features of the observation space with length and description.

The RW rate error, $err\_rwr_{t_i}$, is the absolute difference between the current and target wheel speeds. All observation values were normalized to be within the interval [-1, 1].

2.3 Reward Function and Hyperparameters

For the training of the reaction wheels, we used a modified version of our reward function from [1], shown in Equations (1)-(6).

$$\text{err\_att}_{t_i} = 1 - \left|\text{err\_quat}_{t_i}[0]\right| \tag{1}$$

$$\Delta\text{angle}_{t_i} = \text{err\_att}_{t_i} - \text{err\_att}_{t_{i-1}} \tag{2}$$

$$\text{p\_norm}_{t_i} = \left\|\text{err\_rr}_{t_i}\right\| \tag{3}$$

$$\text{p\_att}_{t_i} = \frac{\text{err\_att}_{t_i}}{10} \tag{4}$$

$$r_{t_i} = \begin{cases} 1 + \dfrac{1}{\text{p\_norm}_{t_i} + 0.1} & \text{, if } \text{err\_att}_{t_i} < 3.8e^{-5} \\ e^{-\frac{\text{err\_att}_{t_i}}{0.14}} & \text{, else if } \text{err\_att}_{t_i} < \text{err\_att}_{t_{i-1}} \\ e^{-\frac{\Delta\text{angle}_{t_i}}{0.14}} \cdot 0.1 & \text{, else if } \Delta\text{angle}_{t_i} < \Delta\text{angle}_{t_{i-1}} \\ e^{-\frac{\text{err\_att}_{t_i}}{0.14}} - 1 & \text{, else} \end{cases} \tag{5}$$

$$\text{reward\_rw}_{t_i} = r_{t_i} - \text{p\_norm}_{t_i} - \text{p\_att}_{t_i} \tag{6}$$

With $\left|\text{err\_quat}_{t_i}[0]\right|$ being the scalar component of the error quaternion at time step t_i. The reward function for the magnetorquers is given by Equations (7)–(9).

$$\text{err\_deviation} = \sum \text{err\_rwr}_{t_i} \cdot 16384.0 \tag{7}$$

$$\text{p\_action} = \frac{\sum \text{a\_m}_{t_i}}{0.6 \cdot 50} \tag{8}$$

$$\text{reward\_mt}_{t_i} = \frac{1\text{-p\_action}}{\sqrt{\text{err\_deviation}+1}} \tag{9}$$

Here, a_mt_{t_i} are the MT actions at the time step t_i. The penalty term p_action is used as a regularization term to incentivize the agent to keep the dipole moments small after achieving its goal, similar to how p_norm and p_att serve as regularization terms for the RW subnetwork. To avoid erratic control signals, we employed generalized State Dependent Exploration (gSDE) [12, 13].

A major challenge arose from the conflicting training requirements of the RWs and MTs. While the RWs can achieve the goal attitude in about 20 seconds, the MTs require approximately two orders of magnitude more time. Consequently, the training of the RWs focuses on immediate rewards, while the training of the MTs emphasizes long-term rewards. Additionally, the MT training is further complicated by two key issues. First, a single action from the MTs has minimal effect on its own, and any action can easily be counteracted by the RWs. This creates a credit assignment problem. Second, the MTs face the limitation of underactuation. As a result, even if a correct control signal is generated for the MTs in a given state, it may have no effect if the satellite's orientation is unfavorable relative to the Earth's magnetic field, or even be penalized if it yields a negative reward.

To address these challenges, we employ a split policy network with dedicated subnetworks for each actuator type. Table 4 presents the hyperparameters and network architectures used, while Sections 2.4 and 2.5 describe the training procedure.

Hyperparameter	RW training [/fine-tuning]	MT training
total_timesteps	5e8 / 2e8	5e8
n_steps (rollout)	2,000 / 16,000	40,000
gamma	0.95	0.97
batch_size	64 / 512	64
clip_range	0.2	0.2
target_kl	0.2	0.2
learning_rate (init)	1e-4 / 1e-6	5e-5
sde_sample_freq	once per rollout	every 8 steps
log_std_init	-2.0	0.0
episode_length	50	5000 (effective length 500)
num_skip	0	9
Weight initialization	Orthogonal weight init with weight scalars of 0.01 for RW action network, 10.0 for MT action network, 1.0 for all other networks	
Networks	RW policy / value network:[64, 64, 64] / [64, 64] MT policy / value network: [64, 64, 64, 64] / [64, 64, 64, 64] Activation function: SiLU	

Table 4: Hyperparameters and network architectures used for the RW and MT subnetworks.

2.4 Twin Network Policy

In general, an AI agent trained with Proximal Policy Optimization (PPO) comprises three main components: a feature extractor (policy network) that encodes raw observations into latent features (a vector of floating-point numbers); an action network that maps these features to a (Gaussian) action distribution, producing the action means; and a value network that estimates the expected return for a given observation under the current policy. The training procedure follows the algorithm described in [8]. PPO optimizes the policy using a clipped surrogate objective, which stabilizes learning by limiting abrupt policy changes. The agent collects experience by interacting with the environment and stores transitions in a rollout buffer. Policy updates are then performed via mini-batch gradient ascent, balancing reward maximization and policy divergence constraints. In parallel, the value network is updated to improve so-called advantage estimates, which quantify how much better or worse a taken action was compared to the expected value of the state. These advantage values scale the policy gradient updates, reinforcing favorable actions and penalizing unfavorable ones. For full details of the training algorithm, we refer the reader to [8].

To unify the training procedure, the policy and action network, in the following simply referred to as the policy network, responsible for generating the control signals for the actuators, is split. We use one subnetwork for the reaction wheels (comparable to an RW agent) and one subnetwork for the magnetorquers (comparable to an MT agent), each producing actions only for their respective actuators. The outputs from both subnetworks are then concatenated, resulting in a six-element action vector, with three components corresponding to the RW torques and three components corresponding to the MT dipole moments. Each subnetwork can be set to one of the three states: DISABLED, FROZEN, or ACTIVE. In the DISABLED state, the subnetwork produces no output (a zero vector), and its action probabilities are set to one, while its entropy is zero. Additionally, the weights of a subnetwork in this state are not trainable. In the FROZEN state, the subnetwork produces actions based on its policy network, but, like in the DISABLED state, it does not contribute to training, and its weights remain non-trainable. In the ACTIVE state, the subnetwork generates actions according to its policy network, fully participates in the training process, and its weights are trainable.

Employing this policy, first, the RW subnetwork is trained with its reward function and hyperparameters from Section 2.3, where the satellite model starts with a random attitude and must achieve a random goal attitude, without consideration for the rate restrictions. Subsequently, the RW network is frozen and the MT network is trained with the MT hyperparameters and reward function. Here, the agent has to achieve target wheel rates of ± 500 RPM while the trained RW subnetwork continues to attain random goal attitudes. This approach ensures that the trained RW controller influences the training of the MT controller, while mitigating potential conflicts between the actuators' requirements for hyperparameters and the reward function. For fine-tuning, the combined controller can be post-trained by activating the RW subnetwork, freezing the MT subnetwork, and performing an additional training run. This way, the RW subnetwork learns to handle the additional disturbances introduced by the MTs, which could not be accounted for during its initial training.

2.5 SkipPPO Trainer

Besides the differences in the actuators' requirements for the training hyperparameters and reward function, there still persists the challenge of the long time horizon of the MTs. To address this, we modified the PPO algorithm to include partial freezing of actions. During training, we initially generate control signals for the RWs and MTs, but freeze the MT actions for num_skip time steps, while generating new actions for the RWs at each time step. This reduces the effective episode length for the MT subnetwork to episode_length/(num_skips+1). Rewards obtained during the skipped steps are accumulated in the rollout buffer and assigned to the corresponding action, amplifying the reward signal and alleviating the credit assignment problem.

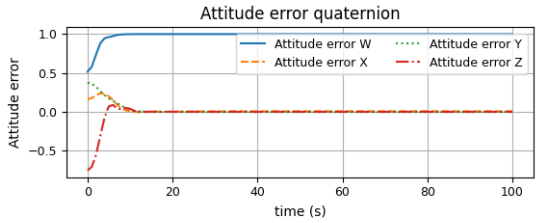
2.6 Hybrid Controller

Initially, a hybrid controller combining an AI-based RW controller with a classical MT controller was considered. The MT controller was implemented and calibrated using simulation and Bayesian Optimization [14]. However, this approach was discarded in favor of a full AI controller to gain the adaptability provided by DRL for both actuators.

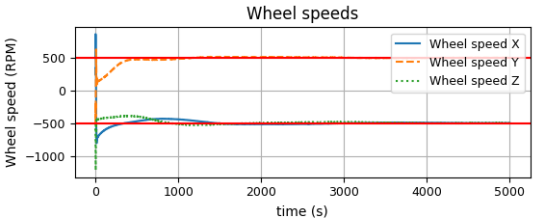
3. TRAINING AND RESULTS

The subnetworks were trained using the policy and trainer outlined in Sections 2.4 and 2.5. During training, the variations outlined in Section 2.1 were applied. After the initial training, the combined AI controller was fine-tuned by unfreezing and post-training the RW subnetwork to adjust for the disturbances introduced by the MT subnetwork. For effective training of the agent, the learning rate was annealed quadratically, due to the tendency of (g)SDE to initially converge on a good policy but subsequently degrade, presumably due to continued over-exploration [12].

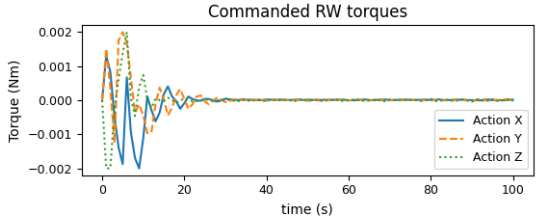
Figure 1 shows the trajectory of the attitude error quaternion (a), the wheel speeds (b), the commanded RW torques (c), and the commanded MT dipole moments (d) for a representative run of the combined AI agent.



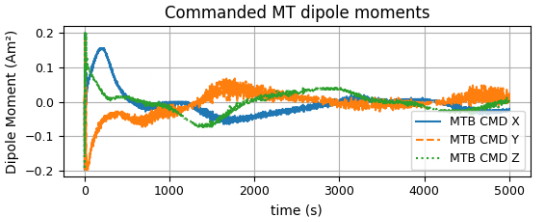
(a) Attitude error quaternion over 100 time steps



(b) Wheel speeds over 5000 time steps



(c) RW torques over 100 time steps



(d) MT dipole moments over 5000 time steps

Figure 1: Attitude error quaternion over 100 time steps (a), wheel speeds over 5000 time steps (b), commanded RW torques over 100 time steps (c), and commanded MT dipole moments over 5000 time steps (d) for a representative episode.

It can be seen that the AI agent reliably attains and maintains the goal attitude, and that the magnetorquer momentum management successfully achieves wheel speeds of ± 500 RPM. However, the MT control signals lack smoothness, indicating potential for improving energy efficiency. Table 5 shows the results obtained for the combined AI controller evaluated over 10,000 episodes with an episode length of 5,000 time steps each, num_skip set to 0, and randomized initial and goal attitudes, along with the variations described in Section 2.1. The metrics include rise time, settling time (attitude), steady-state error (attitude), settling time (wheel rates), and total control effort.

Metric (tolerance attitude: 1 °; tolerance RW rates: 25 RPM)	mean	std
Rise Time (s; 90% → 10% angular attitude error)	8.07	2.44
Settling Time Attitude (s)	17.77	5.10
Steady-State Error Attitude (°)	0.94	0.13
Settling Time Wheel Rates (min)	33.83	18.56
Total Control Effort (Am^2s)	379.62	158.64

Table 5: Mean and standard deviation for different metrics obtained for the combined AI controller, evaluated over 10,000 episodes with 5,000 time steps each, num_skip set to 0, and using random initial and goal attitudes, along with the variations described in Section 2.1.

The results demonstrate the controller's ability to achieve fast and stable attitude responses while maintaining reaction wheel speeds within acceptable limits. Notably, the low steady-state error indicates precise achievement of the goal attitude. In summary, the results confirm the viability of the proposed approach for AI-based attitude control in the presence of wheel-rate restrictions.

4. DISCUSSION AND OUTLOOK

In this work, we introduced an AI-based attitude controller for rate-restricted reaction wheels. We did so by incorporating magnetorquers for momentum management to keep the reaction wheels' rotation rates outside a specific interval. We used a split policy network architecture to resolve the conflict of the actuators regarding the hyperparameters and reward function. We modeled the InnoCube nanosatellite, and trained an AI controller, while incorporating changes to the operating conditions, namely the satellite's moments of inertia and its orbital parameters. The performance of the trained controller was evaluated using different metrics, demonstrating the general feasibility and effectiveness of our approach. However, more work is needed in regard to post-training the combined controller. We want to investigate methods to activate and fine-tune both subnetworks simultaneously. While the general conflict between the actuators still persists in this case, its severity should be reduced, since the trained agent at this point has already learned how to achieve its goal, requiring only minor adjustments to refine the performance. Additionally, the MT control signals of the AI agent could be improved to optimize energy efficiency, possibly by modifying the action penalty term. For successful real-world application, a sim-to-real transfer is essential. Varying the operating conditions is an important step of this process, which we will expand in the future by incorporating additional variations, such as sensor noise and external disturbances. Further, verification is necessary to ensure these variations adequately capture orbital perturbations and discrepancies between simulation and real-world conditions. Our future work will build on this basis, by refining the training procedure and network architecture, investing more effort into post-training, and tackling the sim-to-real transfer. Our goal is to upload our AI attitude

controller to the InnoCube nanosatellite and evaluate our approach in orbit starting in Q3 2025. The InnoCube project is funded by the German Federal Ministry of Economic Affairs and Climate Action (BMWK) under grant number (FKZ) 50RU2000.

5. FUNDING

This work was conducted as part of the LeLaR project (FKZ: 50RA2403), funded by the German Federal Ministry of Economic Affairs and Climate Action (BMWK) on the basis of a decision by the German Bundestag.

6. REFERENCES

[1] K. Djebko, F. Puppe, S. Montenegro, T. Baumann and M. Faisal, Learning attitude control. Presented at *14th IAA Symposium on Small Satellites for Earth System Observation*, Berlin, Germany, May 7-12, 2023

[2] R. Gerlich, R. Gerlich, S. Montenegro, F. Puppe, K. Djebko, C. Plasberg and M. Bädorf, It's the data, stupid! Constructive and analytical quality-assurance for AI-based space systems. Presented at *DASIA 2023 (Data Systems in Aerospace)*, Sitges, Spain, June 6–8, 2023

[3] R. Sutton, F. Bach and A. Barto, Reinforcement learning: An introduction. *MIT Press Ltd, Massachusetts*, 2018

[4] S. Tsimenidis, Limitations of Deep Neural Networks: a discussion of G. Marcus' critical appraisal of deep learning. *arXiv preprint*, arXiv:2012.15754, 2020

[5] B. Grzesik, T. Baumann, T. Walter, F. Flederer, F. Sittner, E. Dilger, S. Gläsner, J.-L. Kirchler, M. Tedsen, S. Montenegro and E. Stoll, InnoCube—a wireless satellite platform to demonstrate innovative technologies. *Aerospace*, vol. 8, no. 5, p. 127, 2021

[6] S. Montenegro, T. Baumann, E. Dilger, F. Sittner, M. Strohmeier, T. Walter and S. Gläsner, InnoCube: Der erste drahtloser Satellit. *Deutsche Gesellschaft für Luft- und Raumfahrt - Lilienthal-Oberth e.V.*, 2022

[7] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus and N. Dormann, Stable-baselines3: Reliable reinforcement learning implementations. *J. Mach. Learn. Res.*, no. 22(268), pp. 1–8, 2021

[8] J. Schulman, F. Wolski, P. Dhariwal, A. Radford and O. Klimov, Proximal policy optimization algorithms. *arXiv preprint*, arXiv:1707.06347, 2017

[9] M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. De Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG, R. Perez-Vicente, A. Pierré, S. Schulhoff, J. J. Tai, H. Tan and O. G. Younis, Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint*, arXiv:2407.17032, 2024

[10] P. W. Kenneally, S. Piggott and H. Schaub, Basilisk: A flexible, scalable and modular astrodynamics simulation framework. *J. Aerosp. Inf. Syst.*, vol. 17, no. 9, pp. 496–507, 2020

[11] CelesTrak, SatCat Table: INNOCUBE. https://celestrak.org/satcat/table-satcat.php?NAME=INNOCUBE, last visited February 14, 2025

[12] T. Rückstieß, M. Felder and J. Schmidhuber, State-dependent exploration for policy gradient methods. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer Berlin Heidelberg, pp. 234–249, 2008

[13] A. Raffin, J. Kober and F. Stulp, Smooth exploration for robotic reinforcement learning. In *Conference on Robot Learning*, PMLR, pp. 1634–1644, 2022

[14] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams and N. De Freitas, Taking the human out of the loop: A review of Bayesian optimization. *Proc. IEEE*, vol. 104, no. 1, pp. 148–175, 2015