Adaptive Attitude Control of Spacecraft via Deep Reinforcement Learning with Lyapunov-Based Reward Design

Kazutoshi Ito and Tomohiro Yanao (Waseda University)

Abstract

Recent and future space missions include various purposes, and autonomous spacecraft attitude controllers are getting important. In this study, we numerically implement adaptive attitude control of a spacecraft including large angle maneuvers via deep reinforcement learning. Deep reinforcement learning is a useful way to deal with autonomous adaptive attitude control problems. However, it takes vast time to learn the global policy. In this study, we use Lyapunov functions to design rewards for the adaptive attitude control, which improve the learning efficiency and achieve more stable learning. Finally, it is shown that the learned policy controls the attitude of a spacecraft robustly under perturbations.

Lyapunov 関数を用いた深層強化学習による宇宙機の適応的姿勢制御

伊藤司聖, 柳尾朋洋 (早稲田大学)

摘要

近年の宇宙ミッションでは様々な目的で宇宙機が設計されており，宇宙機の姿勢制御を自動化することがますます重要になってきている．本研究では，深層強化学習を利用して大きな姿勢マヌーバを含む適応的な姿勢制御を計算機上で実現する．深層強化学習は，宇宙機の適応的な姿勢制御に有効な手法であるが，目的の姿勢制御を実現するための大域的な方策を学習するのに大きな時間コストを要し，従来の報酬設計では方策の最適解の安定性や収束性も十分に保証されない．そこで本研究では，リアプノフ関数を利用して報酬設計を行うことにより，学習効率を向上させるとともに，安定した学習を可能とする．そして，学習で得られた方策が摂動に対してロバストな制御を実現することを示す．

I. Introduction

As space missions become more versatile, not only conventional satellites such as observation satellites and communication satellites, but also novel types of spacecraft such as the one to recover space debris, solar sails for deep space exploration [1, 2], and transformable spacecrafts [3] are proposed and designed. For future space missions, it is necessary to design and control spacecraft adequately depending on the purpose of each mission. Therefore, it is ideal for future space missions to automate all processes from spacecraft design to operation.

Reinforcement learning is actively studied as a means to solve continuous control tasks automatically. In particular, reinforcement learning is applied to high dimensional tasks that conventional control methods are difficult to solve, such as industrial robots performing pick-and-place tasks [4] and quadrupedal robots performing walking motions in environments without any prior knowledge [5].

In addition to robot control, reinforcement learning is applied to explain the motions of living things. One example is the swimming motion of fish. Fish achieve their swimming motions by periodically deforming their own bodies. However, it is still unclear how to make the swimming motion more efficient. Ref. [6] attempted to explain the swimming motion of fish by periodic deformation motions obtained by reinforcement learning.

Moreover, reinforcement learning have plenty of potential for space industry for the purpose of automating attitude and trajectory control of spacecraft [7, 8]. While reinforcement learning is much anticipated in many fields, its stability in control and robustness against perturbations are not sufficiently studied [9]. Therefore, it is necessary to consider a control method that guarantees the stability and robustness in order to apply it to more realistic attitude control problems of spacecraft, where safety must be sufficiently ensured.

In this study, we apply Lyapunov functions to design rewards in deep reinforcement learning for the robust attitude control of spacecraft. Sec. II introduces the model and methods. Sec. III defines the states, actions and rewards for the deep reinforcement learning. Sec. IV presents the results, and Sec. V concludes this paper.

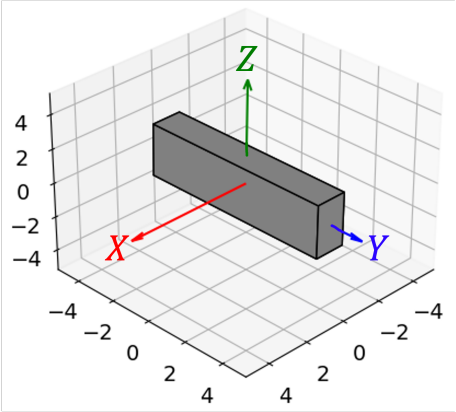

Fig. 1 Spacecraft Model

II. Methods

A. Spacecraft Model

In this study, we consider attitude control of the spacecraft shown in Fig. 1. Moment of inertia of the model is determined with reference to [7, 8] and set to be $J = \mathrm{diag}(0.872, 0.115, 0.797)\ \mathrm{kg}\cdot\mathrm{m}^2$. Attitude of the spacecraft is controlled by torque $\boldsymbol{g}$ generated by thrusters in the direction of each principal axis of inertia.

B. Spacecraft Attitude Dynamics

In this study, we consider a spacecraft as a rigid body. The spacecraft follows Euler's rotational equation [10]:

$$J\dot{\boldsymbol{\omega}} = -\boldsymbol{\omega}\times J\boldsymbol{\omega} + \boldsymbol{g} + \boldsymbol{g}^{\mathrm{ext}}, \tag{1}$$

where J is moment of inertia, $\boldsymbol{\omega}$ is angular velocity, $\boldsymbol{g}$ is control torque and $\boldsymbol{g}^{\mathrm{ext}}$ is other disturbance torque.

Spacecraft attitude is represented by quaternion $\boldsymbol{q} = (q_s, \boldsymbol{q}_v) = (q_1, q_2, q_3, q_4)$, and its kinematic equation is expressed as

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega})\boldsymbol{q}, \quad \boldsymbol{\Omega}(\boldsymbol{\omega}) = \begin{pmatrix} 0 & -\boldsymbol{\omega}^T \\ \boldsymbol{\omega} & -\boldsymbol{\omega}^\times \end{pmatrix}. \tag{2}$$

Note that $\boldsymbol{\omega}^\times$ represents a skew-symmetric matrix [10] corresponding to angular velocity $\boldsymbol{\omega}$ defined by

$$\boldsymbol{\omega}^\times = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix}. \tag{3}$$

C. Reinforcement Learning

Reinforcement learning (RL) is a learning framework, in which, as shown in Fig. 2 [11], an agent interacts with external environment by transferring states, actions, and rewards. When an agent observes state S_t from the environment at discrete time t, it decides the agent's actions A_t using the policy $\pi(A_t|S_t)$ and acts on the environment. The environment changes to a new state based on the agent's action and sends the next state S_{t+1} and the reward R_{t+1} generated by the change to the agent. This cycle is repeated until the terminal condition is satisfied, and finally the agent learns the policy that maximizes the sum of rewards G_t expressed as

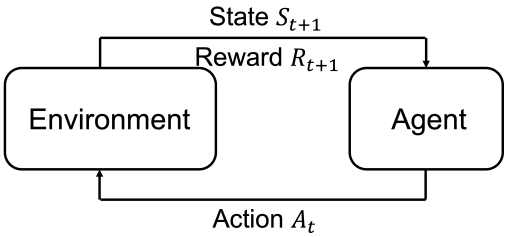

Fig. 2 Reinforcement Learning Framework

$$G_t = \sum_{k=0}^{T-t} \gamma^k R_{t+k}, \tag{4}$$

where γ is the discount factor, and T is the time horizon.

In a RL training, the value function and the action-value function are introduced to evaluate a policy. Value function$V^\pi(s)$ is a function of states that estimates total rewards for a given state $S_t = s$:

$$V^\pi(s) = \mathbb{E}_\pi\left[\sum_{k=0}^{T-t} \gamma^k R_{t+k} \middle| S_t = s\right]. \tag{5}$$

Action-value function is a function of state-action pairs that estimates total rewards for a given state-action pair $(S_t, A_t) = (s, a)$:

$$Q^\pi(s,a) = \mathbb{E}_\pi\left[\sum_{k=0}^{T-t} \gamma^k R_{t+k} \middle| S_t = s, A_t = a\right]. \tag{6}$$

D. Deep reinforcement learning

Deep reinforcement learning approximates a policy or value function and/or action-value function with neural networks as shown in Fig 3. In traditional RL, the policy is approximated using polynomial parameterized by some variables, or the value and/or action-value function is approximated using tabular functions [11]. While traditional RL algorithms without neural networks can solve tasks where state and action spaces can be discretized, it becomes challenging to solve tasks as state and spaces become higher dimensional. Neural networks can overcome this problem. In recent years, neural networks are used in most modern RL algorithms to solve many challenging high dimensional tasks, such as continuous control tasks [12–14] and Atari tasks [13, 15].

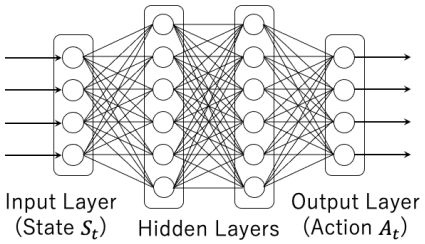

Fig. 3 Neural Network Model

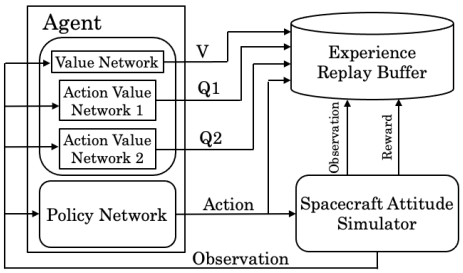

Fig. 4 SAC Framework

E. Soft Actor Critic (SAC)

Soft Actor Critic (SAC) is a deep RL algorithm proposed in Ref. [14]. Differing from the normal RL, SAC maximizes the following objective function J which includes total discounted rewards and an entropy:

$$J = \sum_{k=0}^{T-t} \gamma^k R_{t+k} + \alpha \mathcal{H}(\pi(\cdot|S_t)), \tag{7}$$

where α is a time-dependent value and $\mathcal{H}$ is an entropy.

As shown in Fig. 4, SAC uses an Actor-Critic model [11] that introduces a value function to evaluate the action in addition to a measure to determine the action. In particular, SAC learns by introducing two additional action value functions in addition to the value function. In SAC, each of the policy, the value function, and the action value function is approximated by a neural network. In addition, off-policy learning such as SAC is performed using transition data obtained from past interactions with the environment stored in replay buffer.

III. Reinforcement Learning for Spacecraft Attitude Control

As stated in the previous section, reinforcement learning is an agent-environment framework which transfers states, actions and rewards alternately. This section defines states, actions, and rewards of this study.

A. State and Action Settings

Attitude dynamics of a spacecraft following Eq. (1) is determined by quaternion $\boldsymbol{q}$, angular velocity $\boldsymbol{\omega}$ and control torque $\boldsymbol{g}$. In this setting, an agent controls torque $\boldsymbol{g}$ based on quaternion $\boldsymbol{q}$ and angular velocity $\boldsymbol{\omega}$ at time t.

Thus, state S_t at a discrete time t is given as

$$S_t = (\boldsymbol{q}^e(t), \boldsymbol{\omega}(t) - \boldsymbol{\omega}_d), \tag{8}$$

where $\boldsymbol{q}^e = (q_s^e, \boldsymbol{q}_v^e) = (q_1^e, q_2^e, q_3^e, q_4^e)$ is error quaternion between attitude at time t and target attitude, and $\boldsymbol{\omega}_d$ is target angular velocity. Then, action A_t at a discrete time t is given by

$$A_t = \boldsymbol{g}(t). \tag{9}$$

B. Lyapunov Stability Theory

We consider stability about an equilibrium point of the following nonlinear system:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}), \tag{10}$$

where $\boldsymbol{x}$ represents a displacement from an equilibrium point. Lyapunov stability theory categorizes stabilities of equilibrium points into three types: stable, asymptotically stable and exponentially stable [16].

An equilibrium point is stable if there exists a C^1 function V_L, functions $\alpha_i (i = 1, 2)$, positive constant r that satisfy the following equations for any $\boldsymbol{x}$ [16]:

$$\alpha_1(\|\boldsymbol{x}\|) \leq V_L(\boldsymbol{x}) \leq \alpha_2(\|\boldsymbol{x}\|), \quad \dot{V}_L = \frac{\partial V_L}{\partial \boldsymbol{x}} f(\boldsymbol{x}) \leq 0. \tag{11}$$

An equilibrium point is asymptotically stable if there exists a C^1 function V_L, functions $\alpha_i (i = 1, 2)$, positive constant r that satisfy the following equations for any $\boldsymbol{x}$ [16]:

$$\begin{aligned} &\alpha_1(\|\boldsymbol{x}\|) \leq V_L(\boldsymbol{x}) \leq \alpha_2(\|\boldsymbol{x}\|), \\ &\dot{V}_L = \tfrac{\partial V_L}{\partial \boldsymbol{x}} f(\boldsymbol{x}) \leq -\alpha_3(\|\boldsymbol{x}\|). \end{aligned} \tag{12}$$

An equilibrium point is exponentially stable if the equilibrium point is asymptotically stable and $\alpha_i(\|\boldsymbol{x}\|) = a_i \|\boldsymbol{x}\|^p (i = 1, 2, 3)$, where a_i is a positive constant and p is a positive integer [16].

C. Lyapunov-Based Reward Design

There are various reward designs for control tasks. For example, sparse reward is defined as follows:

$$r_t = \begin{cases} +1 & \text{task success,} \\ -1 & \text{otherwise.} \end{cases} \tag{13}$$

Sparse reward can easily reach the global optimal solution, but the convergence speed is low.

Another example is a distance-based reward defined as distance from a current position $\boldsymbol{x}$ and a target position $\boldsymbol{x}_d$ expressed as

$$r_t = -\|\boldsymbol{x} - \boldsymbol{x}_d\|, \quad \text{or} \quad r_t = \exp(-\|\boldsymbol{x} - \boldsymbol{x}_d\|). \tag{14}$$

As opposed to sparse reward, distance-based reward is faster to learn, although it tends to fall into local optimal solutions. These reward design above are useful to solve many control problems including attitude control problems [7, 8].

However, these methods do not guarantee the stability around target points. In addition, optimization problems are difficult to get global optima as more parameters need to be optimized. Therefore, in this study, we design reward for the attitude control of a rigid spacecraft based on the Lyapunov function. In this Lyapunov-based reward design, agent receives a positive reward if the Lyapunov function V_L satisfies the exponentially stable conditions. Otherwise, agent receives a negative reward. We thus define Lyapunov-based reward as follows:

$$r_t = \begin{cases} +1 & a_1\|\boldsymbol{x}\|^p \le V_L(\boldsymbol{x}) \le a_2\|\boldsymbol{x}\|^p \\ & \text{and} \quad \dot{V}_L = \frac{\partial V_L}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}} \le -a_3\|\boldsymbol{x}\|^p, \\ -1 & \text{otherwise.} \end{cases} \tag{15}$$

For the attitude control of a rigid-body-like spacecraft, we define displacement $\boldsymbol{x}$ and Lyapunov function V_L as [17]:

$$\boldsymbol{x} = \Big(\boldsymbol{q}^e - \boldsymbol{q}_d, \boldsymbol{\omega}(t) - \boldsymbol{\omega}_d\Big), \tag{16}$$

$$V_L(\boldsymbol{x}) = \frac{k_d}{2}(\boldsymbol{\omega} - \boldsymbol{\omega}_d)^T J(\boldsymbol{\omega} - \boldsymbol{\omega}_d) + k_p\Big[(1 - q_s^e)^2 + \boldsymbol{q}_v^{eT}\boldsymbol{q}_v^e\Big], \tag{17}$$

where $\boldsymbol{q}_d^e = (1, 0, 0, 0)$ is the error quaternion for the target attitude.

IV. Results

Table 1 shows the settings of environment and agent.

Table 1 Settings of Environment and Agent

Parameters	Value
Moments of inertia	diag(0.872, 0.115, 0.797) kg·m^2
Stacked time	0.1 s
Frameskip time	0.0 s
Control frequency	10Hz (0.1 s)
Max control torque	0.5 N·m
# of steps per episode	1000 steps (100.0 s)
k_d	1.0
k_p	5.0
a_1	0.01
a_2	5.0
a_3	0.001
p	2

We consider two types of attitude control problems: the case with impulse disturbance torque and the case without any disturbance torque. In both cases, agent is trained with the initial conditions that error quaternion is randomly and uniformly distributed, and initial angular velocity is $\boldsymbol{\omega}_0 = (0.0, 0.0, 0.0)$ rad/s. In agent's training, we use 1M total timesteps to update agent's policy, and each episode is terminated if $\|\boldsymbol{q}^e - \boldsymbol{q}_d^e\| < 0.01$ or the number of steps is 1000 steps.

A. Results without Disturbance Torque

We first evaluate agent's policy under the condition that no disturbance torque is applied during control. Initial and target conditions are set as in Table 2.

Results are shown in Figs. 5 - 6. As we see from Fig. 5, agent was able to control spacecraft to target attitude successfully. We observe that the attitude maneuvers satisfy Lyapunov stability conditions through the entire evaluating episode as in Fig. 6.

Table 2 Initial and target conditions in attitude control without any disturbance torque

Parameters	Value
Initial error quaternion $\boldsymbol{q}_0^e$	(0.707, −0.308, −0.308, −0.308)
Initial angular velocity $\boldsymbol{\omega}_0$	(0.0, 0.0, 0.0) rad/s
Target error quaternion $\boldsymbol{q}_d^e$	(1.0, 0.0, 0.0, 0.0)
Target angular velocity $\boldsymbol{\omega}_d$	(0.0, 0.0, 0.0) rad/s

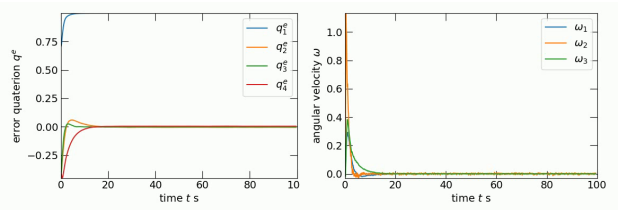

Fig. 5 Time evolution of quaternion (left) and angular velocity (right) without disturbance torque

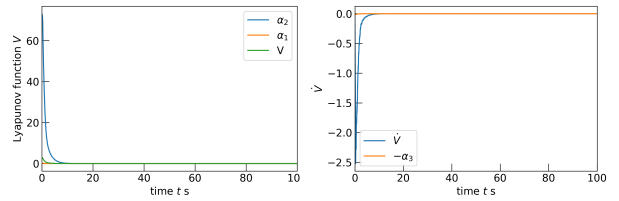

Fig. 6 Time evolution of V_L (left) and $\dot{V}_L$ (right) without disturbance torque

B. Results with Impulse Torque

We evaluate agent's policy under the condition that an impulse torque is applied at $t = 40.0$ s. Initial and target conditions and impulse torque are set as in Table 3.

Results are shown in Figs. 7-8. As we see from Fig. 7, agent was able to control spacecraft to target attitude suc-

cessfully. We observe that the attitude maneuvers mostly satisfy Lyapunov stability conditions even after an impulse torque was applied at $t = 40.0$ s as shown in Fig. 8.

Table 3 Initial and target attitudes and disturbance torque in the case of attitude control under an impulse disturbance torque

Parameters	Value
Initial error quaternion $\boldsymbol{q}_0^e$	$(0.707, -0.308, -0.308, -0.308)$
Initial angular velocity $\boldsymbol{\omega}_0$	$(0.0, 0.0, 0.0)$ rad/s
Target error quaternion $\boldsymbol{q}_d^e$	$(1.0, 0.0, 0.0, 0.0)$
Target angular velocity $\boldsymbol{\omega}_d$	$(0.0, 0.0, 0.0)$ rad/s
Impulse torque $\boldsymbol{g}^{\text{ext}}$	$(5.0, 5.0, 5.0)$ N·m
Time to apply impulse torque	40.0 s

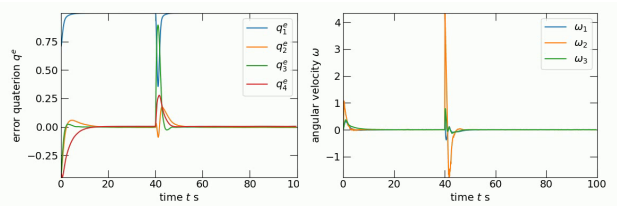

Fig. 7 Time evolution of error quaternion (left) and angular velocity (right) with disturbance torque

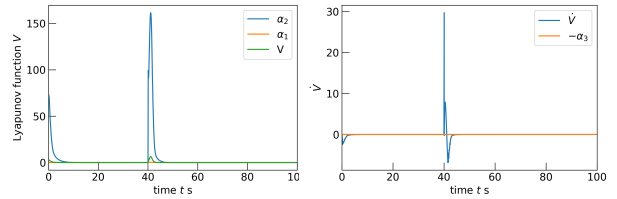

Fig. 8 Time evolution of V_L (left) and $\dot{V}_L$ (right) with disturbance torque

C. Robustness for Different Initial Angular Velocities

To quantify the robustness for different initial angular velocities, we measure the total number of steps where the difference between the error quaternion and the target error quaternion $\|\boldsymbol{q}^e - \boldsymbol{q}_d^e\|$ is less than 0.01 in one episode.

As shown in Table 4, we examine 9 different initial angular velocities. For each initial angular velocity, we simulated for 10 episodes. The resulted number of steps satisfying $\|\boldsymbol{q}^e - \boldsymbol{q}_d^e\| < 0.01$ (=success steps) is shown in Fig. 9. When the initial angular velocity is increased, it takes longer to reach the target attitude, but the control to the target attitude is still accurately performed.

D. Robustness for Different Moments of Inertia

We investigate the robustness of the attitude control for different moments of inertia from that of the training. We thus measure the number of success steps for moments of inertia of 0.1, 0.3, 0.5, 2.0, 5.0 and 10.0 times the moment of inertia used in agent's training. The results are summarized in Fig. 10. When the moment of inertia is 0.1 or 0.3 times the moment of inertia used in agent's training, spacecraft attitude control was not successful with the learned policy, whereas in the other cases, the learned policy was able to control the attitude with the similar performance to the moment of inertia used in the training. In particular, the robustness tends to be maintained for the moments of inertia larger than the learned one. Therefore, it is suggested that learning with models with smaller moments of inertia is more advantageous to secure the robustness against the changes in moment of inertial of a spacecraft.

Table 4 Nine cases of initial angular velocities to examine the robustness of agent's learned policy

	Initial Angular Velocity $\boldsymbol{\omega}_0$
Case1	$(0.0, 0.0, 0.0)$ rad/s
Case2	$(-0.1, -0.1, -0.1)$ rad/s
Case3	$(0.1, 0.1, 0.1)$ rad/s
Case4	$(1.0, 1.0, 1.0)$ rad/s
Case5	$(-1.0, -1.0, -1.0)$ rad/s
Case6	$(2.5, 2.5, 2.5)$ rad/s
Case7	$(-2.5, -2.5, -2.5)$ rad/s
Case8	$(5.0, 5.0, 5.0)$ rad/s
Case9	$(-5.0, -5.0, -5.0)$ rad/s

V. Conclusion

We have used Lyapunov functions to design rewards in the deep reinforcement learning for the attitude control of a spacecraft. The resulted attitude control has been robust to a disturbance torque and the difference in initial angular velocities, satisfying the Lyapunov stability conditions for almost the entire episode. Moreover, the learned policy has successfully controlled the attitude of spacecraft whose moments of inertia are larger than the one used in the training.

References

[1] Mori, O., Sawada, H., Funase, R., Morimoto, M., Endo, T., Yamamoto, T., Tsuda, Y., Kawakatsu, Y., Kawaguchi, J., Miyazaki, Y., Shirasawa, Y., and Demonstration Team and Solar Sail W, I., "First Solar Power Sail Demonstration by IKAROS," *Transactions of the Japan Society for Aeronautical and Space Sciences, Aerospace Technology Japan*, Vol. 8, No. ists27, 2010, pp. To_4_25–To_4_31.

[2] Fu, B., Sperber, E., and Eke, F., "Solar sail technology—A state of the art review," *Progress in Aerospace Sciences*, Vol. 86, 2016, pp. 1–19.

[3] Ohashi, K., Chujo, T., and Kawaguchi, J., "Optimal motion planning in attitude maneuver using non-holonomic turns

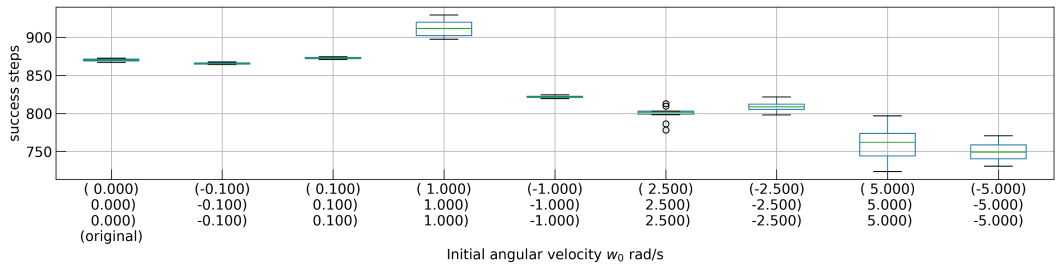

Fig. 9 Number of steps satisfying $\|q^e - q_d^e\| < 0.01$ in each episode for different initial angular velocities. The leftmost box-and-whisker diagram represents the result for the initial angular velocity used in agent's training, serving as a benchmark to compare with other initial angular velocities.

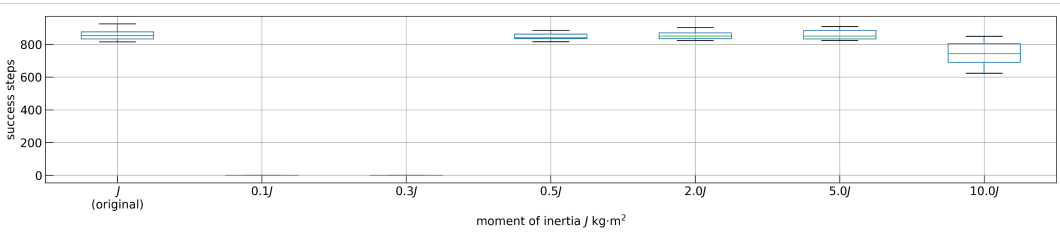

Fig. 10 Number of steps satisfying $\|q^e - q_d^e\| < 0.01$ in each episode for different moments of inertia. The leftmost box-and-whisker diagram represents the moments of inertia used in the agent's training, serving as a benchmark to compare with other moments of inertia.

for a transformable spacecraft," *Advances in the Astronautical Sciences*, Vol. 167, 2018, pp. 2735–2745.

[4] Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., and Levine, S., "QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation," , 2018.

[5] Tsounis, V., Alge, M., Lee, J., Farshidian, F., and Hutter, M., "DeepGait: Planning and Control of Quadrupedal Gaits using Deep Reinforcement Learning," , 2020.

[6] Jiao, Y., Ling, F., Heydari, S., Heess, N., Merel, J., and Kanso, E., "Learning to swim in potential flow," *Phys. Rev. Fluids*, Vol. 6, 2021, p. 050505.

[7] Elkins, J. G., Sood, R., and Rumpf, C. M., "Adaptive Continuous Control of Spacecraft Attitude Using Deep Reinforcement Learning," *AAS Astrodynamics Specialist Conference*, , No. August, 2020, pp. 1–18.

[8] Elkins, J. G., Sood, R., and Rumpf, C. M., "Autonomous Spacecraft Attitude Control Using Deep Reinforcement Learning," *AAS Astrodynamics Specialist Conference*, , No. October, 2020, pp. 12–14.

[9] Shirobokov, M., Trofimov, S., and Ovchinnikov, M., "Survey of machine learning techniques in spacecraft control design," *Acta Astronautica*, Vol. 186, 2021, pp. 87–97.

[10] Hughes, P. C., *Spacecraft attitude dynamics*, Courier Corporation, 2004.

[11] Richard S., S., and Barto, A. G., *Reinforcement Learing - An Introduction*, 2018.

[12] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D., "Continuous control with deep reinforcement learning," 2015.

[13] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O., "Proximal Policy Optimization Algorithms," 2017, pp. 1–12.

[14] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S., "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *International conference on machine learning*, PMLR, 2018, pp. 1861–1870.

[15] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D., "Human-level control through deep reinforcement learning," *Nature*, Vol. 518, No. 7540, 2015, pp. 529–533.

[16] 井村順一, "《弟 8 回》非線形システムの安定性理論の基礎," **計測と制御**, Vol. 43, No. 2, 2004, pp. 178–185.

[17] 山田克彦, "宇宙機の姿勢制御," **計測と制御**, Vol. 40, No. 6, 2001, pp. 433–440.