UDC 629.5 https://doi.org/10.15407/itm2025.04.077

V. V. SOROCHINSKII[1], S. V. KHOROSHYLOV[1], I. L. LEVCHUK[2], T. M. DUBOVYK[2], H. M. HUZ[2], O. O. ROMANCHUK[2]

ON-OFF SPACECRAFT RELATIVE CONTROL IN SLIDING MODE VIA REINFORCEMENT LEARNING

[1]*Institute of Technical Mechanics of the National Academy of Science of Ukraine and the State Space Agency of Ukraine, 15 Leshko-Popel Str., Dnipro, 49005, Ukraine; e-mail: vovas99@ukr.net*

[2]*Ukrainian State University of Science and Technologies,2 Lazariana St., Dnipro, 49010, Ukraine*

Розглянуто задачу відносного імпульсного керування рухом космічного апарата у ковзному режимі для автономних орбітальних сервісних операцій за наявності обмежень на амплітуду керуючих впливів, дискретності дій та параметричних невизначеностей. Метою роботи є розробка й оцінювання підходу, що поєднує принципи ковзного керування з сучасними методами навчання з підкріпленням, орієнтованими на бортову реалізацію з обмеженими ресурсами. Динаміку відносного руху задано в орбітальній системі координат у нормалізованих змінних і дискредитовано. Імпульсний характер впливів виконавчих органів відображено через бінарні дії з широтно-імпульсною модуляцією та обмеженнями на рівень тяги, тривалість і період увімкнень. Запропоновано комбінований синтез, у якому параметри поверхні ковзання та правила перемикання налаштовуються методом проксимальної оптимізації політики з використанням архітектури актор-критик. Актор і критик реалізовані у вигляді нейронних мереж, які відповідно апроксимують політику та функцію цінності. Нейронна мережа актора приймає вектор стану як вхідну інформацію і видає середнє значення та стандартне відхилення параметрів закону керування у ковзному режимі. Функція цінності штрафує як за помилку стану, так і за витрати на керування, що дозволяє забезпечити компроміс між швидкістю реагування, точністю та витратою палива. Два незалежні агенти розроблені для керування відносним орбітальним рухом космічного апарата окремо в напрямку площини орбіти та у перпендикулярному напрямку. Гіперпараметри оптимізації проксимальної політики обрано для забезпечення компромісу між часом навчання, стабільністю та якістю керування. Агенти навчання з підкріпленням навчені та проаналізовані з урахуванням чотирьох випадків, що відрізняються рівнями тяги та ваговими матрицями. Функціонал якості об'єднує штрафи за відхилення стану та використання тяги, що дає змогу знаходити компроміс між швидкодією, точністю та витратами робочого тіла. Отримані результати підтверджують потенціал такого підходу для задач автономного керування космічних апаратів в умовах обмежень та невизначеності. У порівнянні з відомими результатами навчений агент продемонстрував кращу робастність по відношенню до невизначенності параметрів моделі об'єкта керування, що пояснюється сильними робастними властивостями керування в ковзному режимі. Отримані результати мають потенціал підвищити ефективність та автономність орбітальних сервісних операцій.

Ключові слова: *навчання з підкріпленням, проксимальна оптимізація політики, керування космічним апаратом, орбітальні сервісні операції, on-off керування, автономні системи керування.*

The paper addresses the problem of on-off spacecraft relative control in sliding mode for autonomous on-orbit servicing operations under actuator amplitude limits, action discreteness, and parametric uncertainties. The goal is to develop and assess an approach that combines sliding-mode control with modern reinforcement-learning methods tailored for resource-constrained onboard implementation. Relative motion dynamics is formulated in an orbital coordinate frame with normalized states and discretized in time. Binary actions with pulse-width modulation, subject to constraints on the thrust level, pulse duration, and duty cycle, represent the impulsive nature of actuation. We propose a combined synthesis in which the sliding-surface parameters and switching rules are tuned via proximal policy optimization within an actor-critic architecture. The actor and critic are implemented as neural networks that approximate the policy and the value function, respectively. The actor neural network takes the state vector as input information and outputs the mean and standard deviation of the parameters of the sliding mode control law. The value function penalizes both the state error and control effort, thus enabling a trade-off among the response speed, accuracy, and propellant consumption. Two uncoupled agents are designed to control spacecraft relative orbital motion in in-plane and out-of-plane directions independently. The proximal policy optimization hyperparameters are selected to ensure a trade-off among the learning time, stability, and control performance. The reinforcement-learning agents are trained and analyzed considering four cases that differ in the thrust levels and weighting matrices. The quality functional combines state deviation and thrust use penalties, thus enabling a trade-off among the response speed, accuracy, and propellant consumption. The results confirm the potential of this approach for autonomous spacecraft control under constraints and uncertainty. Compared with reported baselines, the trained agent shows superior robustness to plant-parameter uncertainty, which we attribute to the inherent robust properties of sliding-mode control. These findings have the potential to improve the efficiency and autonomy of on-orbit servicing operations.

Keywords: *reinforcement learning, proximal policy optimization, spacecraft control, on-orbit servicing, on–off control, autonomous control systems*.

1. Introduction

Modern space missions increasingly require complex maneuvers with minimum human involvement. Typical tasks include on-orbit servicing operations such as docking, towing, and active debris removal [1, 2].

To perform these operations, a servicing spacecraft (SSC) must maneuver close to the orbital object (OO), addressing the challenge of relative guidance and control. Control inputs are commonly generated by reaction thrusters (RTs). Unlike reaction wheels or control-moment gyros (CMGs), thruster outputs are typically binary (“on/off”). This switching behavior makes actuation highly nonlinear and complicates direct controller design. To implement continuous-time control laws, pulse-width modulation (PWM) and pulse-frequency modulation (PFM) are used [3, 4], transforming the continuous control signal into a sequence of pulses with specified duration or frequency. The accuracy of this approximation depends on the modulator resolution: coarse quantization significantly reduces control performance [5], so the regulator and modulator parameters must be co-designed, which is a complex task.

The impressive results achieved using deep learning (DL) methods [6–10] have recently intensified interest in AI approaches among researchers and practitioners worldwide. Recently, these methods have also been applied to space problems [11–16].

Publication [17] demonstrated the feasibility of directly synthesizing control laws via reinforcement learning (RL), mapping the state vector to binary on/off thruster commands. That approach outperformed a linear regulator in conjunction with a PWM in terms of accuracy, response speed, and the number of thruster firings. At the same time, it requires many training episodes and does not provide strict robustness guarantees when training and operational conditions differ. To mitigate these drawbacks, this study proposes combining sliding-mode control (SMC) with RL.

This study aims to assess the feasibility of on/off spacecraft relative control in sliding mode that is implemented via reinforcement learning, and to specify the features of this approach

2. Problem Statement

2.1 Model of spacecraft relative dynamics. To describe the motion of the servicing spacecraft (SSC) relative to an orbital object (OO), we use the orbital coordinate frame (OCF). The OCF origin is fixed at the SSC center of mass (Fig. 1). The x-axis is directed along the geocentric radius vector from Earth’s center to the SSC ($\hat{\imath}$). The z-axis is normal to the plane spanned by the x-axis and the SSC orbital-velocity vector ($\hat{k}$) and is oriented toward the positive orbital

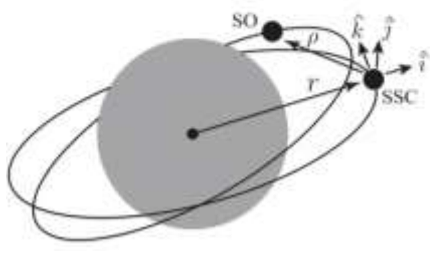


Fig. 1 – Axes directions of the OCF

angular momentum. The y-axis completes a right-handed triad ($\hat{j}$).

The relative dynamics of the SSC–OO system can be written as the following linearized system of equations [18]:

$$\ddot{x} - \omega^2 x - 2\omega\dot{y} - \dot{\omega}y - kx = \frac{f_x^d}{m^d} - \frac{f_x^s}{m^s}, \quad (1)$$

$$y - \omega^2 y + 2\omega\dot{x} + \dot{\omega}x + ky = \frac{f_y^d}{m^d} - \frac{f_y^s}{m^s}, \quad (2)$$

$$z + kz = \frac{f_z^d}{m^d} - \frac{f_z^s}{m^s}, \quad (3)$$

where x , y , z are the OCF projections of the relative position vector r; m^s and m^d are the masses of the SSC and OO, respectively; f_x^d, f_y^d, f_z^d are the OCF projections of the total force f^d, acting on the OO; f_x^s, f_y^s, f_z^s are the OCF projections of the total force f^s, acting on the SSC. The total force f^s includes both control input and external disturbances acting on the SSC. The disturbances f^d and f^s may also account for the J2 perturbation, third-body gravity (Sun and Moon), aerodynamic drag, and solar radiation pressure. The parameters in Eqs (1) – (3) are defined as follows:

$$\omega = \sqrt{\frac{\mu}{p^3}}(1 + \varepsilon\cos v)\omega, \dot{\omega} = -2\varepsilon\sqrt{\frac{\mu}{p^3}}\sin v\,(1 + \varepsilon\cos v)\omega,$$

$$p = a(1 - \varepsilon^2), k = \frac{\mu}{r^3}, r = \frac{a(1-\varepsilon^2)}{1+\varepsilon\cos v},$$

where μ is the Earth's gravitational constant, ε is the orbital eccentricity, v is the true anomaly, a is the semi-major axis, and r is the geocentric distance.

Equations (1) – (2) describe in-plane orbital motion, while Eq (3) describes out-of-plane orbital motion.

Neglecting the external disturbances and introducing the state vectors $X_{in} = [x, y, \dot{x}, \dot{y}\,]^T, X_{out} = [z, \dot{z}\,]^T$ and the control vectors $U_{in} = \left[u_x, u_y\right]^T$, $U_{out} = u_z$, the model (1) can be written in the standard linear state-space form:

$$\dot{X}_{in} = A_{in}X_{in} + B_{in}U_{in}, \dot{X}_{out} = A_{out}X_{out} + B_{out}U_{out} , \quad (4)$$

where

$$A_{in} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \omega^2 + 2k & \dot{\omega} & 0 & 2\omega \\ -\dot{\omega} & \omega^2 - k & -2\omega & 0 \end{bmatrix}, B_{in} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -\frac{1}{m^s} & 0 \\ 0 & -\frac{1}{m^s} \end{bmatrix},$$

$$A_{out} = \begin{bmatrix} 0 & 1 \\ -k & 0 \end{bmatrix}, B_{out} = \begin{bmatrix} 0 \\ -\frac{1}{m^s} \end{bmatrix}.$$

The magnitudes of individual state components differ significantly, which may complicate neural-network (NN) training. To mitigate this issue, the state vector is normalized as follows:

$$X_{in}^n = \left[\frac{x}{x_m}, \frac{y}{y_m}, \frac{\dot{x}}{\dot{x}_m}, \frac{\dot{y}}{\dot{y}_m}\right]^T, X_{out}^n = \left[\frac{z}{z_m}, \frac{\dot{z}}{\dot{z}_m}\right]^T, \quad (5)$$

where x_m, y_m, z_m, $\dot{x}_m$, $\dot{y}_m$, $\dot{z}_m$ are the maximum magnitudes of the corresponding state components. For the normalized state, the dynamic model can be given as

$$\dot{X}^n = A^n X^n + B^n U,$$

where $A^n = N^{-1}AN, B^n = N^{-1}B, N = \text{diag}(x_m, \ y_m, z_m, \ \dot{x}_m, \ \dot{y}_m, \dot{z}_m)$.

Since the onboard controller is implemented using a digital computer, we use the discrete-time form of the model:

$$X_{k+1} = A_k X_k + B_k U_k, \tag{6}$$

where $A_k = (I + A^n T)$; $B_k = B^n T$; T is the sampling period, and k is the time index. We also assume that the full state vector is measurable and that these measurements are not corrupted by noise.

2.2. Sliding-Mode Control Sliding-mode control is a variable-structure control in which the input is deliberately shaped to cause the trajectory to reach a sliding surface and then move along it [19]. The control law is not a continuous function of time, because the control structure switches depending on the current location in the state space. The switching nature of SMC suits well naturally to the task of spacecraft relative-motion control because reaction thrusters operate in the on/off mode.

Motion along the sliding hypersurface is described by a reduced-order equation and can be compared to the evolution along eigenmodes of linear systems. To keep the system sliding requires a high-gain switching feedback.

The SMC design includes the following steps:

selecting a sliding hypersurface $s(x) = 0$ such that motion along it yields the desired system behavior;

synthesizing a feedback law with sufficient gains that guarantees both reaching the surface and its invariance - i.e., the trajectory intersects the surface and then remains on it.

The switching function $s(x)$ is defined as a measure of the deviation of the state x from the sliding hypersurface: the state belongs to the hypersurface when $s(x) = 0$ and out of it when $s(x) \neq 0$. The control has a switching structure and changes sign according to the sign of the s(x), e.g., $u = u^+$ when $s(x) > 0$ and $u = u^-$), thus ensuring reaching and subsequent sliding along the surface.

Within the SMC paradigm, the control action is always directed to reduce the magnitude of $|s(x)|$ and to "push" the trajectory toward the sliding surface. When the surface is reached, the trajectory slides along it and, depending on how the surface is chosen, moves toward the target, typically an equilibrium (e.g., the origin).

The switching function is defined as follows:

$$\sigma(X) = S^T X, \tag{7}$$

Let the sliding surface be a simplex defined by $\sigma(X) = 0$. Then, for the stability analysis, we consider the following Lyapunov candidate:

$$V\big(\sigma(X)\big) = 0.5\sigma(X)^T \sigma(X)$$

with the time derivative

$$\dot{V}\big(\sigma(X)\big) = 0.5\sigma(X)^T \dot{\sigma}(X).$$

The control is chosen so that $\dot{\sigma}(X) < 0$ when $\sigma(X) > 0$ and $\dot{\sigma}(X) > 0$ when $\sigma(X) < 0$ to ensure that $\dot{V}\big(\sigma(X)\big) < 0$. The evolution of the switching function can be given as

$$\dot{\sigma}(X) = S^T (AX + BU).$$

The control input is formed as

$$U = (S^T B)^{-1} \left(-S^T AX + h\big(\sigma(X)\big)\right),$$

where $h\big(\sigma(X)\big) = -K_i \frac{\sigma(X)}{|\sigma(X)|+\varepsilon}$, and ε sets a dead zone that prevents excessive control chattering.

Since the actuators operate in on/off mode, the SMC output is approximated by a sequence of pulses of variable duration using a PWM. The pulse duration within each sampling period is determined as follows:

$$t_f = \frac{u_k}{u_f} T,\ t_f \le T,$$

where u_f is the nominal thrust level.

As the control performance metric, we use the following quadratic cost function [20]:

$$J = \sum_{k=0}^{N} \left({X_k}^T Q X_k + {U_k}^T R U_k\right), \tag{8}$$

where Q and R are the weighting matrices penalizing the states X_k and the control U_k , respectively.

To obtain an optimal sliding-mode controller, it is necessary to determine the parameters K, S, and ε that minimize the quadratic cost (8). This optimization task is addressed via reinforcement learning.

3. Reinforcement-Learning-Based Control

In the RL paradigm, the controller learns by analyzing the consequences of its own actions [21]. These consequences are evaluated by a scalar reinforcement signal received from the plant with which the controller interacts. The reinforcement can be interpreted as a criterion that enables an intelligent system to adjust its control policy to achieve a long-term objective.

The generic RL loop (Fig. 2) includes the following steps:

1. At the time t_k the plant is in the state X_k. In this state, the controller (CS) selects an action U_k from the available action set.
2. The chosen action is applied, causing a transition to a new state X_{k+1}, and the system receives a reinforcement signal C_k.
3. The algorithm repeats from step 2, taking into account the received reinforcement; if the new state is terminal, the episode ends.

Let χ denote the state space and A the action space. The reinforcement C_k results from applying the action U_k in state X_k. Formally, it is a function on $\chi \times A$: $C_k = C(X_k, U_k)$

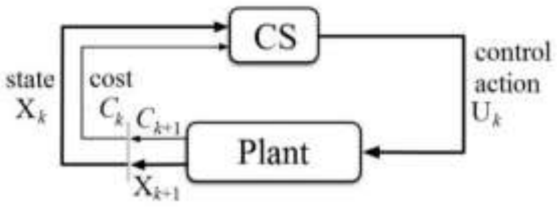


Fig. 2 – RL setup

The CS selects actions to minimize the return, which is the discounted sum of instantaneous costs:

$$G_k = C_k + \gamma C_{k+1} + \gamma^2 C_{k+2} + \ldots = \sum_{i=0}^{\infty} \gamma^i C_{k+i}, 0 \le \gamma \le 1,$$

where γ is the discount factor and C_k is the instantaneous cost at step k.

The factor γ determines how strongly future costs influence current action selection.

A key element of RL is the value function. Let the controller apply actions according to a policy π:

$$U_k = \pi(X_k).$$

Then the value function $V^{\pi}(X_k)$ gives the total discounted cumulative cost when starting from the state X_k and following the policy π:

$$V^{\pi}(X_k) = \sum_{i=0}^{\infty} \gamma^k C_{k+i}(X_{k+i}, U_{k+i}) = C_k(X_k, U_k) + \gamma V^{\pi}(X_{k+1}).$$

RL can be implemented using an actor-critic architecture. In this case, the critic estimates the value function for each state, while the actor maps states to control actions.

In this study, the actor and critic are implemented as neural networks that approximate, respectively, the policy and the value function:

$$V^{\pi}(X_k, \phi),\ \pi(X_k, \phi\ ),$$

where θ and ϕ are the parameter vectors of the critic and actor, respectively.

Proximal Policy Optimization (PPO) [22] is chosen among different RL algorithms because it provides a tradeoff between implementation complexity and training stability. This includes the following steps:

Cumulative cost estimation, which is the sum of the cost for this time step and the discounted future cost [23]:

$$G_t = \sum_{k=t}^{ts+m}\left(\gamma^{k-t} C_k\right) + b\gamma^{N-t+1} V(X_{ts+N}, \theta),$$

where $b = 0$ if X_{t_s+N} is terminal and $b = 1$ otherwise (i.e., when the rollout segment does not end at a terminal state, a bootstrap term from the critic $V(\cdot;\theta)$ is added); $\gamma \in [0,1]$ is the discount factor.

1. Advantage estimation:

$$D_t = G_t - V(X_t, \theta),$$

2. Critic update via value loss minimization

$$L_{critic}(\theta) = \frac{1}{M}\sum_{i=1}^{M}\left(G_i - V(X_i, \theta)\right)^2,$$

3. Actor update using clipped policy objective

$$L_{actor}(\phi) = \frac{1}{M}\sum_{i=1}^{M}\left(-min(r_i(\phi)\cdot D_i, c_i(\phi)\cdot D_i)\right), \quad r_i(\phi) = \frac{\pi(U_i|X_i,\phi)}{\pi(U_i|X_i,\phi_{old})},$$

$$c_i(\phi) = max(min(r_i(\phi), 1+\varepsilon), 1-\varepsilon)\ ,$$

where D_i is the advantage for the i-th sample, G_i is the corresponding return, $\pi(U_i|X_i, \phi)$ is the probability of action U_i in state X_i under the current policy parameters ϕ, $\pi(U_i|X_i, \phi_{old})$ is the probability under the pre-update policy, ε is the clip factor, M is the minibatch size.

The NN architecture of the intelligent agent is given in Tables 1 and 2. Almost all network layers have the tanh activation, and only the actor's output layer employs SoftPlus function to provide positive outputs.

The actor NN takes the state vector as input and produces the mean and standard deviation of the parameters S, K, and ε. The structure of this NN is shown in Fig. 3. In conjunction with this actor, we use the same NN critic and the same quadratic performance index as in Ref [17].

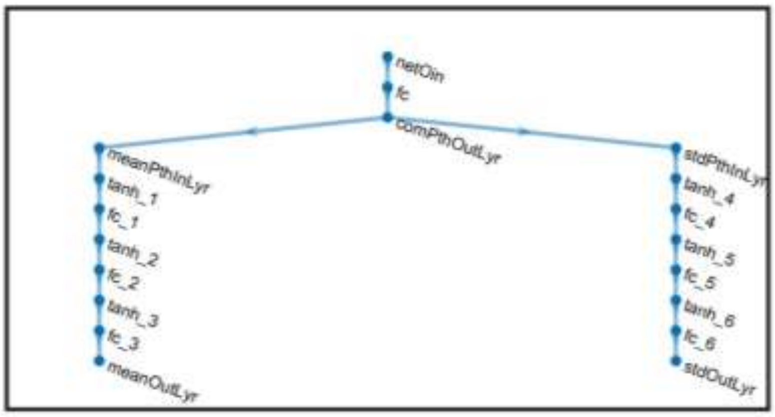


Fig. 3 – Architecture of the actor neural network

Table 1. Number of neurons in the actor layers

Case	**Input**	**fc hidden**	**fc_1/fc_4 hidden**	**fc_2/fc_5 hidden**	**fc_3/fc_6 hidden**	**Output**
in-plain	n_{in}	$10n_{in}$	$10n_{in}$	25	30	3
out-of-plain	n_{out}	$10n_{out}$	$10n_{out}$	50	60	6

Table 2. Number of neurons in the critic layers

Case	**Input**	**1-st hidden**	**2-nd hidden**	**3-d hidden**	**Output**
in-plain	n_{in}	$10n_{in}$	$\sqrt{100n_{in}}$	10	1
out-of-plain	n_{out}	$10n_{out}$	$\sqrt{100n_{out}}$	10	1

4. Numerical results

For training and evaluation of the intelligent agent, the following system parameters were used: a=7017 km; $m^s = 500$ kg; $m^d = 1575$ kg; T=200 sec; $T_f = 10$ sec. Maximum magnitudes of the state-vector components: $x_m = 800$ m, $y_m = 800$ m, $\dot{x}_m$=2 m/s, $\dot{y}_m$=2 m/s.

We analyzed four cases that differ in the thrust levels and weighting matrices (Table 3).

Table 3. Values of thrust levels and weighting matrices

	in-plain			**out-of-plain**		
	$\boldsymbol{u_f}$	**Q**	**R**	$\boldsymbol{u_f}$	**Q**	**R**
Case 1	8	[1 0; 0 0.001] ×1e-7	1e-3	8	[1 0 0 0; 0 1 0 0; 0 0 0.001 0; 0 0 0 0.001]×1e-7	[1e-3 0; 0 1e-3]
Case 2	4	[1 0; 0 0.001]×1e-7	1e-3	4	[1 0 0 0; 0 1 0 0; 0 0 0.001 0; 0 0 0 0.001]×1e-7	[1e-3 0; 0 1e-3]
Case 3	8	[1 0; 0 0.001]×1e-7	1e-4	8	[1 0 0 0; 0 1 0 0; 0 0 0.001 0; 0 (0 0.001]×1e-7	[1e-3 0; 0 1e-4]
Case 4	4	[1 0; 0 0.001]×1e-7	1e-4	4	[1 0 0 0; 0 1 0 0; 0 0 0.001 0; 0 0 0 0.001]×1e-7	[1e-3 0; 0 1e-4]

The reward function is designed as the weighted sum of state errors and control efforts:

$$r_t = -(x_t^T Q x_t + u_t^T R u_t). \quad (10)$$

Training was conducted in episodes, each running until a maximum step count was reached. The PPO hyperparameters were selected to ensure stable learning as follows: $\gamma = 0.98$, $\varepsilon = 0.2$. An experience horizon of 50 and a minibatch size of 50 were used for updating the NN parameters.

Figure 3 demonstrates system stability since all trajectories of x_5, x_6 converge to zero. The cases differ in how quickly the spacecraft reaches the target state and in the magnitude of residual errors. Lower thrust leads to a longer transition and more frequent small control actions, when the agent corrects the state more frequently using small pulses. After settling, the residual error is close to zero in all cases. The small remaining oscillations are due to the on-off control actions.

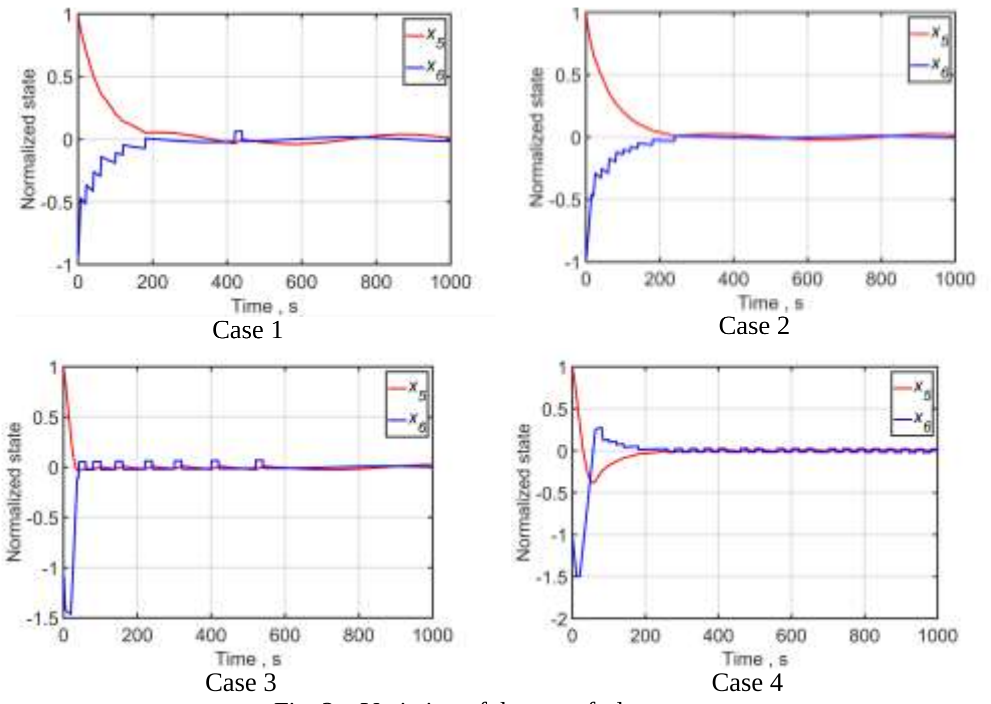
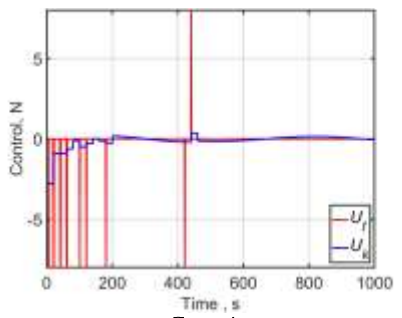
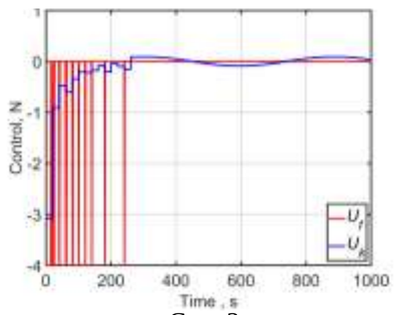


Fig. 3 – Variation of the out-of-plane state

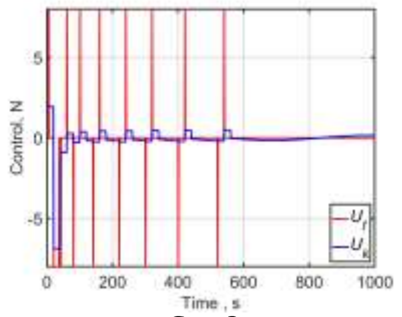
In Fig. 4, the actor NN output signal is plotted by a blue line and its on-off realization is shown by a red one. Initially, sequences of saturated same-sign pulses are applied to reduce the error rapidly. After that, the pulses become sparser and shorter, occasionally changing sign for fine corrections. In cases when higher thrust is available, fewer control pulses are applied than in cases with lower thrust. The discrepancy between the blue and red lines is due to a mismatch between the required and available thrusts and the discrete PWM realization of the control. In the steady state, both curves remain near zero with small oscillations due to the on-off actuation.



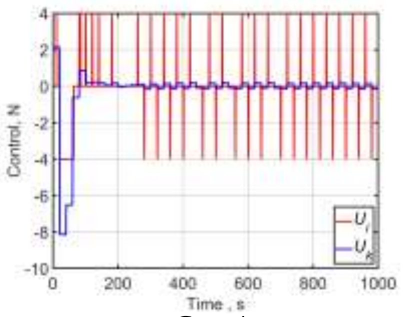
Case 1



Case 2



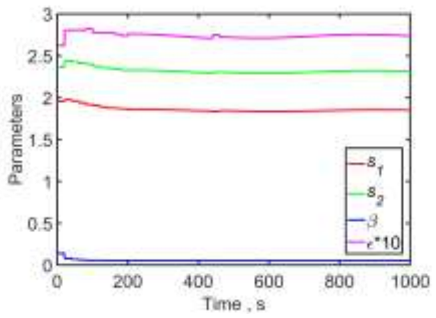
Case 3



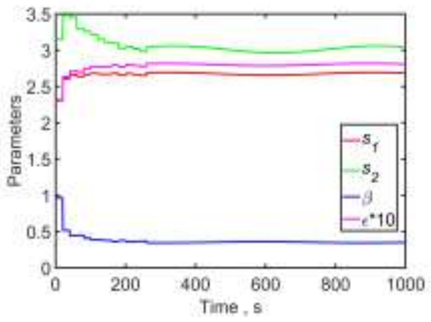
Case 4

Fig. 4 – Variation of the out-of-plane control actions

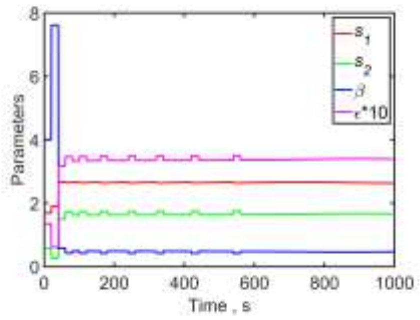
Figure 5 demonstrates a similar pattern across all four parameter plots, where after a short initial interval with possible sharp jumps, all curves vary slowly. Parameters s_1 and s_2 stabilize around 2–3 units. Typically s_2 slightly exceeds s_1. The parameter β quickly drops from a large initial value to a small one (< 1) and then varies slowly. The parameter ε shown as $\varepsilon \times 10$ in the plot, stays near 2–3 on the scale, i.e., $\varepsilon \approx 0.2 - 0.3$. Small "steps" on all curves reflect discrete parameter updates. In the steady state, deviations are minor.



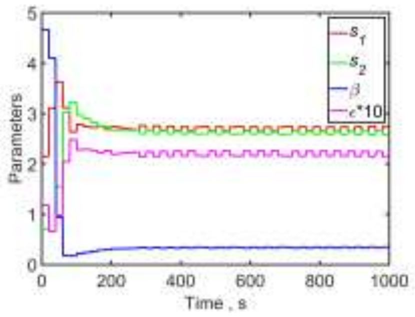
Case 1



Case 2



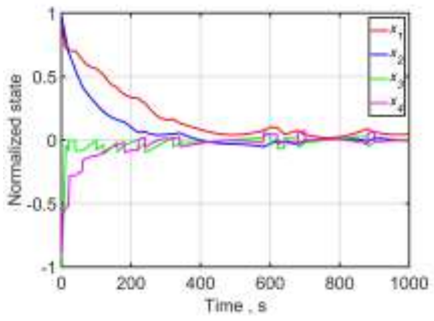
Case 3



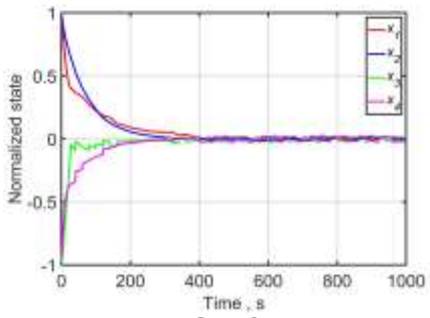
Case 4

Fig. 5 – Parameter variation for the out-of-plane channel

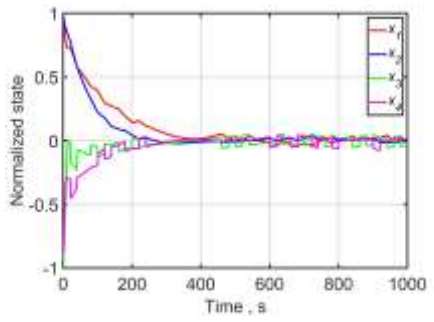
In Fig. 6, all state components converge to zero. State components x_1 and x_2 (red and blue) decrease smoothly with no noticeable overshoot. Velocities x_3 and x_4 (green and purple) rise to zero in “stair-steps,” reflecting discrete actions. After about (300–500) s, all curves remain near zero and only small periodic oscillations, which are typical for on-off control, are visible. These plots mainly differ in the convergence speed and the magnitude of steady-state errors.



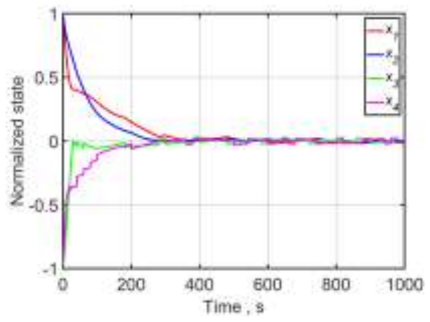
Case 1



Case 2

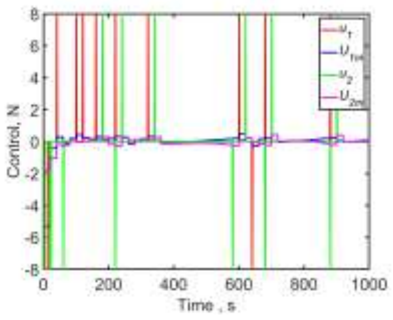


Case 3

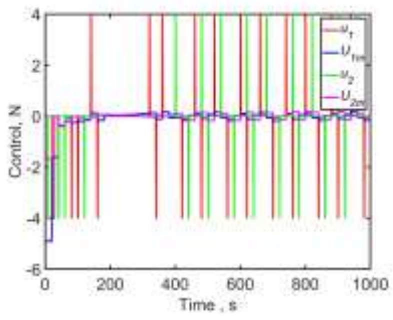


Case 4

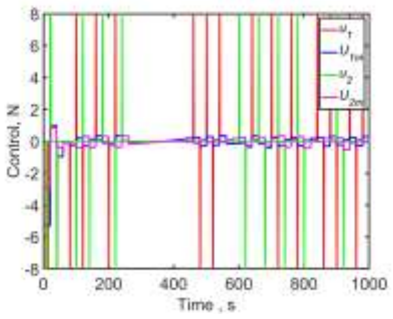
Fig. 6 – Variation of the in-plane state



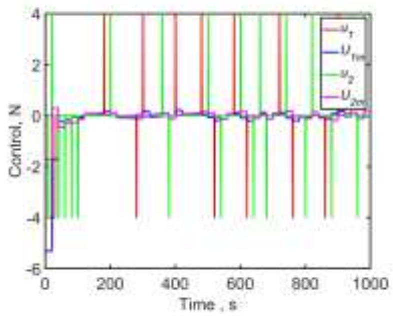
Case 1



Case 2

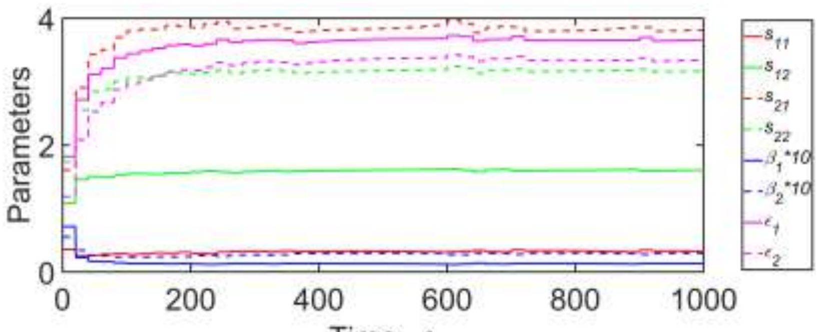


Case 3

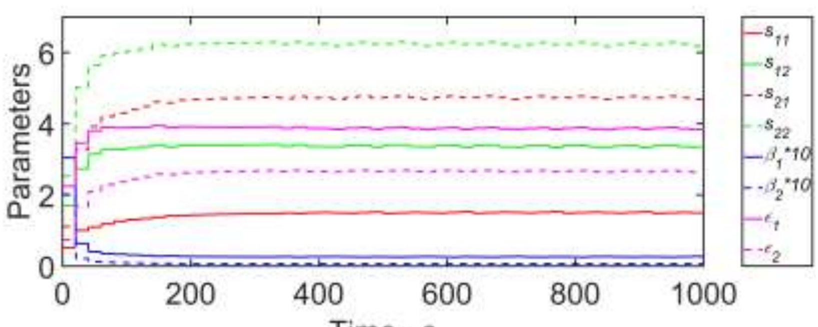


Case 4

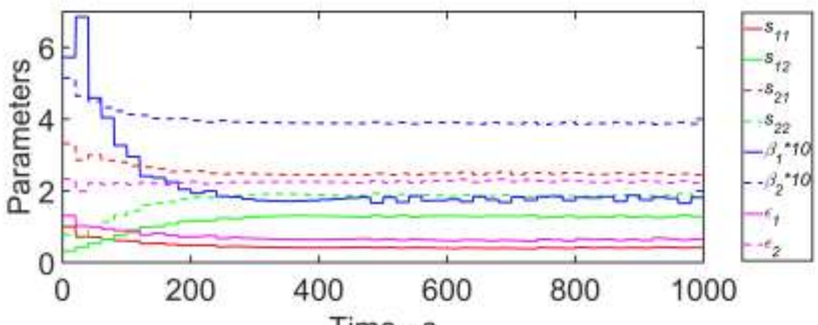
Fig. 7 – Variation of the in-plane control actions



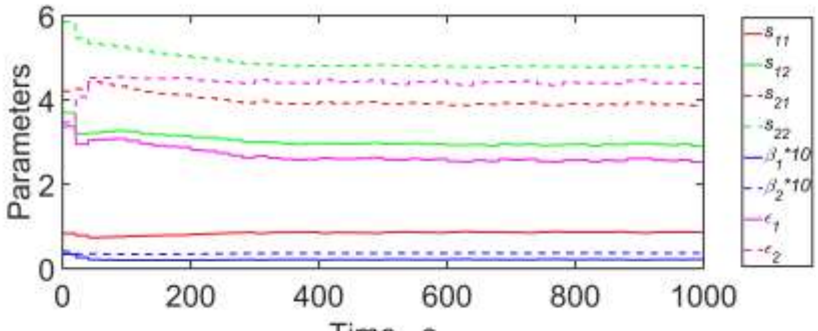
Case 1



Case 2



Case 3

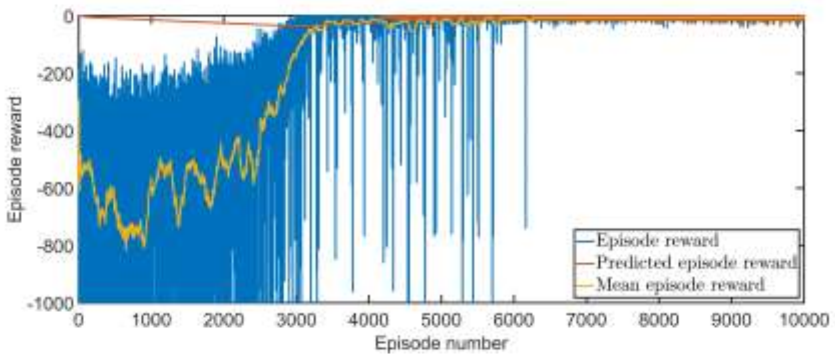


Case 4

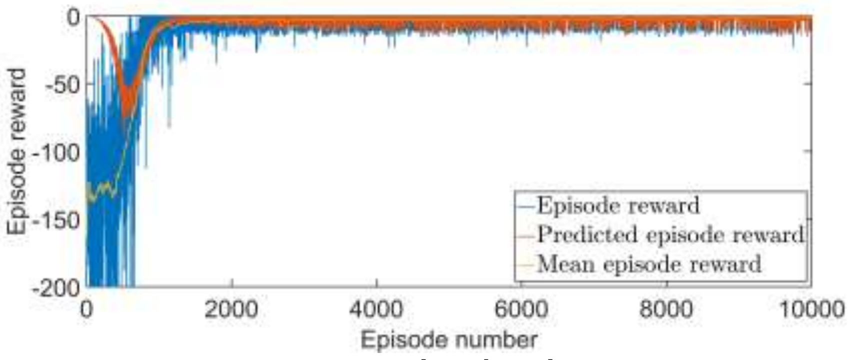
Fig. 8 – Parameter variation for the in-plane channel

As can be seen in Fig. 7, the sequence of dense same-sign pulses rapidly reduces the state errors at the beginning of the episodes. After that, the pulses become sparser and shorter, occasionally changing sign for accurate corrections. As stabilization proceeds, both commands stay near zero, and the pulses have short duration and low frequency, resulting in state chartering around equilibrium due to on-off actuation. The difference between the channels appears in pulse phasing and polarity, but the errors converge to small magnitudes in both channels.

Figure 8 demonstrates that after a short initial jump, the parameters change slowly. The parameters s_{ij} are stabilized with a clear ordering, when s_{12}, s_{22} remain higher than s_{11}, s_{21}. The parameters β_1, β_2 (plotted $\times$ 10) typically decrease from high initial values and then oscillate around small values. The parameters $\varepsilon_1, \varepsilon_2$ settle down quickly and vary little over the whole interval. In both channels, the trajectories are similar in shape and level, i.e., the settings converge to close steady values with only minor drift.



Out-of-plane channel



In-plane channel

Fig. 9 – Evolution of the cumulative return during training (Case 2)

Figure 9 shows a typical evolution of the cumulative return during training. At the beginning of training, returns have a large value with high variance. After a while, the mean return improves monotonically and approaches zero. The critic's estimates of the return initially are biased, but once the policy forms, it tracks the mean curve well. In the first plot, convergence is faster, while in the second one, it takes longer training with deeper occasional drops. However, the mean return also moves toward zero. After convergence, some episodes may still result in relatively high negative returns, but the overall trend remains stable.

5. Robustness analysis

In practice, real model parameters may deviate from their nominal values, so we assess the robustness of the SMC-based intelligent controllers against model parameter variations and compare it with the robustness of the intelligent agents from Ref. [17]. In this article, two RL-based agents for relative spacecraft control were proposed that differ in their input vector. The first agent (IA1) receives the standard state vector X_k, while the input of the second agent (IA2) is augmented with the previous step control action, i.e.,$[X_k,\ u_{k-1}]^T$.

Table 4 presents the description of test cases, which differ in the percentage deviation of the considered parameter from its nominal value.

Table 4. Description of test cases

Symbol	Nom.	B. 1	B. 2	B. 3	B. 4	B. 5	B. 6
Description	Nominal parameter value	+10 %	+20 %	+30 %	–10 %	–20 %	–30 %

Figure 10 shows the state variations for IA1, when SSC mass m^s deviates from the nominal one. As can be seen from these plots, the agent remains robust when the spacecraft mass decreases, but control performance degrades as mass increases (Case B.1). In Cases B.2 and B.3 the agent fails to maintain stability. This occurs because a greater spacecraft mass requires larger control amplitudes than those considered during training.

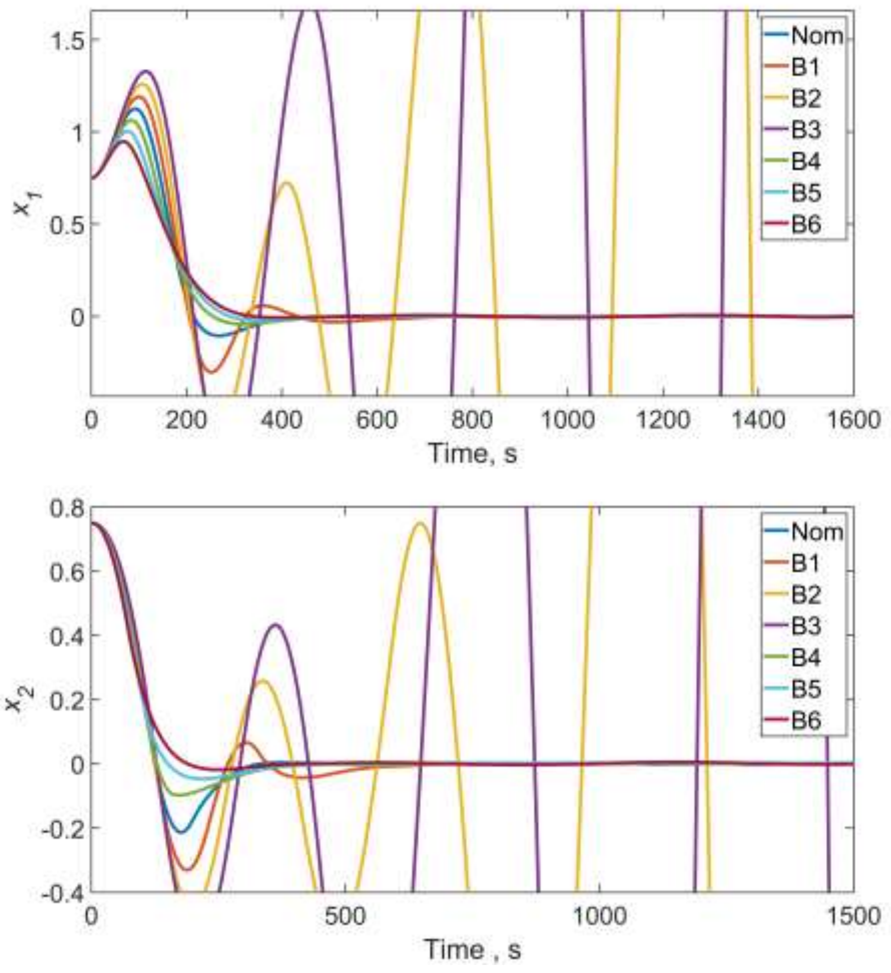


Fig. 10 – State variations for IA1 and different spacecraft masses

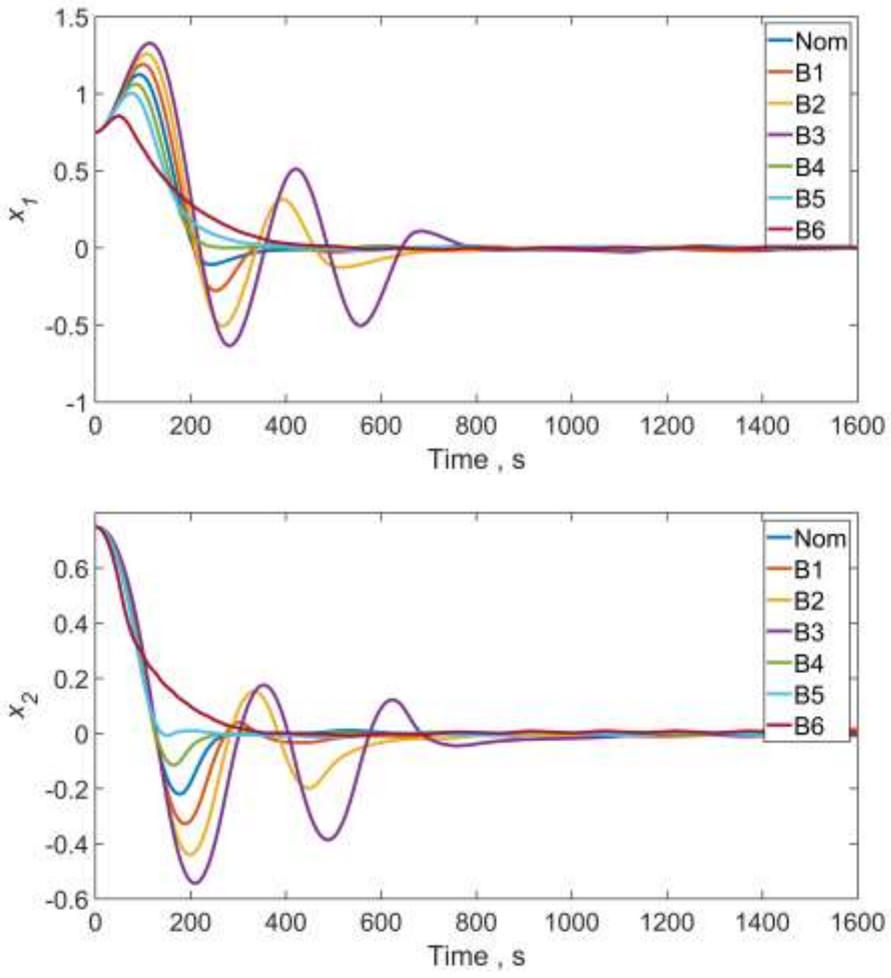


Figure 11 – State variations for IA2 and different spacecraft masses

Fig. 11 shows the state variations for IA2, when SSC mass m^s deviates from the nominal one. The agent remains robust, but control performance degrades as mass increases (Case B.1). Unlike IA1, IA2 maintains stability in Cases B.2 and B.3, although the performance degrades in these cases.

Figure 12 presents the state trajectories for the SMC-based agent and various cases of the SSC masses. Stability is preserved in all cases with a similar transient shape pattern. However, in some cases, the steady-state error increases.

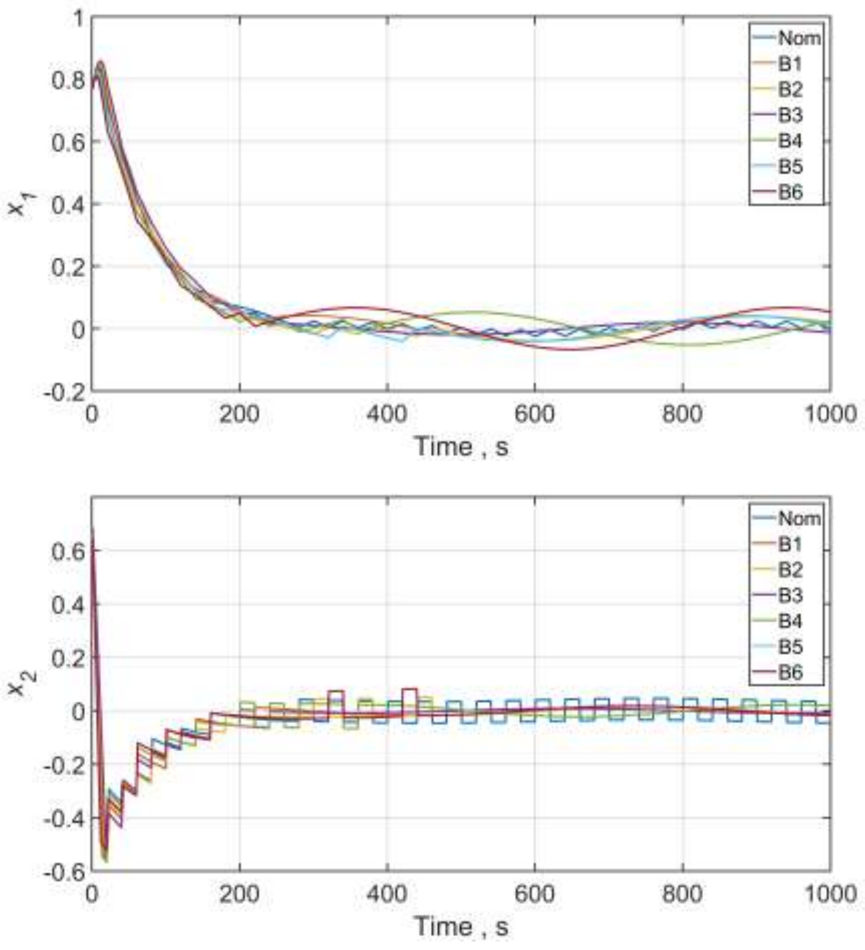


Fig. 12 – State variations for SMC-based agent and different spacecraft masses.

6. Conclusions

This paper presents an approach to relative on-off spacecraft control that combines sliding-mode control with reinforcement learning. The results confirm the potential of this approach for autonomous spacecraft control under constraints and uncertainty. The trained agent demonstrated superior robustness to uncertainty in plant parameters compared to the reported baselines, which is due to the strong inherent robustness of sliding-mode control. However, a large number of episodes is required to train the agent, which restricts the practical implementation of this approach in its current form. Accordingly, future work should explore ways to improve the training efficiency of the agent.

1. *Chandra A., Kalita H., Furfaro R., Thangavelautham J.* End to End Satellite Servicing and Space Debris Management. arXiv:1901.11121, 2019. 15 p.
2. *Li W., Cheng D., Liu X*., et al. On-orbit service (OOS) of spacecraft: A review of engineering developments. Progress in Aerospace Sciences. 2019. Vol. 108. P. 32–120. https://doi.org/10.1016/j.paerosci.2019.01.004
3. *Khosravi A., Sarhadi P.* Tuning of pulse-width pulse-frequency modulator using PSO: An engineering approach to spacecraft attitude controller design. Automatika. 2016. No. 57. P. 212–220. https://doi.org/10.7305/automatika.2016.07.618
4. *Anthony T., Wie B., Carroll S.* Pulse-Modulated Control Synthesis for a Flexible Spacecraft. Journal of Guidance. 1989. Vol. 13(6). P. 1014–1022. https://doi.org/10.2514/3.20574
5. *Alpatov A., Khoroshylov S., Lapkhanov E.* Synthesizing an Algorithm to Control the Angular Motion of Spacecraft Equipped with an Aeromagnetic Deorbiting System. Eastern-European Journal of Enterprise Technologies. 2020. 1(5(103)). P. 37–46. https://doi.org/10.15587/1729-4061.2020.192813

6. *Goodfellow I., Bengio Y., Courville A*. Deep Learning. MIT Press, 2016. 800 p.
7. *Krizhevsky A., Sutskever I., Hinton G. E.* ImageNet classification with deep convolutional neural networks. Communications of the ACM. 2017. 60(6). P. 84–90. https://doi.org/10.1145/3065386
8. *Pierson H., Gashler M*. Deep learning in robotics: a review of recent research. Advanced Robotics. 2017. 31(16). P. 821–835. https://doi.org/10.1080/01691864.2017.1365009
9. *Sallab A. E., Abdou M., Perot E., Yogamani S.* Deep reinforcement learning framework for autonomous driving. Electronic Imaging. 2017. Issue 19. P. 70–76. https://doi.org/10.2352/ISSN.2470-1173.2017.19.AVM-023
10. *Silver D., Schrittwieser J., Simonyan K.* Mastering the game of Go without human knowledge. Nature. 2017. 550. P. 354–359. https://doi.org/10.1038/nature24270
11. *Izzo D., Märtens M., Pan B.* A survey on artificial intelligence trends in spacecraft guidance dynamics and control. Astrodynamics. 2019. 3. P. 287–299. https://doi.org/10.1007/s42064-018-0053-6
12. *Khoroshylov S. V., Redka M. O.* Deep learning for space guidance, navigation, and control. Space Science and Technology (Космічна наука і технологія). 2021. 27(6/133). P. 38–52. https://doi.org/10.15407/knit2021.06.038
13. *Oestreich C. E., Linares R.,* Gondhalekar R. Autonomous six-degree-of-freedom spacecraft docking maneuvers via reinforcement learning. Journal of Aerospace Information Systems. 2021. 18(7). https://doi.org/10.2514/1.I010914
14. *Gaudet B., Linares R., Furfaro R.* Six Degree-of-Freedom Hovering using LIDAR Altimetry via Reinforcement Meta-Learning. Acta Astronautica. 2020. 172. P. 90–99. https://doi.org/10.1016/j.actaastro.2020.03.026
15. *Gaudet B., Linares R., Furfaro R.* Seeker based Adaptive Guidance via Reinforcement Meta-Learning Applied to Asteroid Close Proximity Operations. Acta Astronautica. 2020. 171. P. 1–13. https://doi.org/10.1016/j.actaastro.2020.02.036
16. *Redka M. O., Khoroshylov S. V.* Determination of the force impact of an ion thruster plume on an orbital object via deep learning. Space Science and Technology (Космічна наука і технологія). 2022. 28(5/138). P. 15–26. https://doi.org/10.15407/knit2022.05.015
17. *Khoroshylov S. V., Wang C.* Spacecraft relative on-off control via reinforcement learning. Space Science and Technology (Космічна наука і технологія). 2024. 30(2/147). P. 3–14. https://doi.org/10.15407/knit2024.02.003
18. *Khoroshylov S. V.* Relative motion control system of SC for contactless space debris removal. Sci. innov. (Наука та інновації). 2018. 14(4). P. 5–16. https://doi.org/10.15407/scine14.04.005
19. *Steinberger M., Horn M., Fridman L. (eds).* Variable-Structure Systems and Sliding-Mode Control. Springer-Verlag, London, 2020. (Studies in Systems, Decision and Control; Vol. 271). https://doi.org/10.1007/978-3-030-36621-6
20. *Bryson A. E., Ho Y. C.* Applied Optimal Control: Optimization, Estimation, and Control. Washington: Hemisphere Publishing, 1975. P. 224–235.
21. *Sutton R. S., Barto A. G.* Reinforcement Learning: An Introduction. 2nd ed. MIT Press, 2018. P. 47–65.
22. *Schulman J., Wolski F., Dhariwal P., Radford A., Klimov O.* Proximal Policy Optimization Algorithms. arXiv:1707.06347, 2017. 13 p.
23. *Mnih V., Badia A., Mirza M., Graves A., Lillicrap T., Harley T., Silver D.* Asynchronous Methods for Deep Reinforcement Learning. arXiv:1602.01783, 2016.

Received on October 14, 2025,
in final form on December 3, 2025