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

IAC–22–C1.6.2

Adaptive Closed-Loop Maneuver Planning for Low-Thrust Spacecraft using Reinforcement Learning

Nicholas B. LaFarge[a]*, Kathleen C. Howell[b], David C. Folta[c]

[a] *Ph.D. Candidate, School of Aeronautical and Astronautical Engineering, Purdue University, West Lafayette, IN, United States*, nlafarge@purdue.edu
[b] *Hsu Lo Distinguished Professor of Aeronautics and Astronautics, School of Aeronautical and Astronautical Engineering, Purdue University, West Lafayette, IN, United States*, howell@purdue.edu
[c] *Senior Fellow, Navigation and Mission Design Branch, Code 595, NASA Goddard Space Flight Center, Greenbelt, MD, United States*, david.c.folta@nasa.gov
* Corresponding Author

Abstract

Autonomy is an increasingly essential component of future space missions, and new technologies are necessary to accommodate off-nominal occurrences onboard that may pose risks to mission success. In search of a satisfactory maneuver plan to correct for an unanticipated event, the initial identification of a suitable basin of convergence remains challenging for onboard, low-thrust mission applications in cislunar space. This investigation addresses this challenge by demonstrating artificial neural networks as promising tools in estimating accurate startup solutions for a conventional iterative guidance and control approach, thus producing a robust 'hybrid' architecture that simultaneously benefits from the computational simplicity of neural networks, and the robustness of a targeting scheme to satisfy accuracy requirements and to ensure mission success. In this paradigm, differential corrections is incorporated directly into a reinforcement learning process, tasking the resulting neural network controller with initial guess identification for trajectory recovery. Rapid low-thrust maneuver planning is demonstrated in a 'runaway' spacecraft scenario, where deviation over time from a planned near rectilinear halo orbit path renders stationkeeping ineffective, and demands an alternative approach to determine an effective recovery plan.
Keywords: Spacecraft autonomy, Reinforcement learning, Low-thrust, Cislunar space, Neural network control

1. Introduction

To enable fully autonomous spaceflight and to establish a sustained human and robotic presence in deep space, rapid onboard maneuver planning is an essential technology. In complex multi-body dynamical environments, such as in the Earth-Moon neighborhood, onboard applications for low-thrust spacecraft are particularly challenging. Traditionally, Guidance and Control (G&C) techniques involve communicating with ground-based resources. However, susceptibility to communications failure, time delays, limitations in data transmission, sensor tasking complexity, and operations costs all motivate moving G&C functionality to the flight computer. While many recent advancements in trajectory design exploit improvements in computer hardware, relatively few are practical for autonomous onboard implementation given the limited computational resources. Reinforcement Learning (RL), a subset of Machine Learning (ML), has emerged in recent years as a promising tool in onboard spaceflight G&C applications. This investigation proposes an RL-based G&C process, with demonstrated applicability to a low-thrust mission recovery scenario along a Near Rectilinear Halo Orbit (NRHO).

Reinforcement learning excels in sequential decision-making problems that lack a-priori knowledge of an effective control strategy, with recent applications frequently leveraging a Neural Network (NN) to parameterize the control function. In G&C problems, NN controllers are demonstrated as potentially effective in overcoming challenging dynamical regions of space. However, the safety-critical nature of spaceflight poses practical barriers to onboard ML implementations, where NN accuracy and explainability issues introduce risk to mission success. This investigation addresses the safety-criticality of incorporating a NN into the G&C architecture by blending ML with traditional methods. In this paradigm, rather than directly controlling the spacecraft, the NN is instead trained to estimate startup solutions for a traditional iterative approach, an architecture that is here termed 'Neural Network-Initialized Targeting' (NNIT). Directly incorporating traditional iterative methods into the training process broadens the available solution space, reduces the impact of chaotic dynamics on the training process, and ensures all mission criteria are satisfied by the corresponding solution.

Several recent flight software prototype investigations leverage iterative numerical methods to construct or update maneuver schedules while ensuring the necessary mission constraints remain satisfied. For example, the autoNGC flight software system, developed by NASA

Goddard Space Flight Center, implements a forward shooting scheme for onboard maneuver planning optimization [1]. Similarly, the onboard GN&C architecture for Orion includes a differential corrections, or targeting, algorithm that is capable of automatically altering maneuver plans based on navigation states [2]. Furthermore, targeting methods are commonly employed in multi-body stationkeeping (SK) approaches [3, 4], with demonstrated applicability in NRHO applications for the upcoming Gateway program [5, 6]. While targeting methods are effective in converging to feasible trajectories, they rely on sufficiently accurate startup solutions to both achieve convergence and to maintain current maneuver schedules. In the presence of large deviations or low-thrust propulsion options, an originally planned baseline reference trajectory may prove insufficient for achieving targeting convergence. This investigation addresses this limitation by introducing the NNIT architecture, and details the associated RL-based training process to construct a NN that rapidly and autonomously generates accurate initial conditions for a targeter despite unexpected, prolonged deviations from a pre-planned mission scenario.

A key challenge in enabling onboard autonomy is handling off-nominal situations that are normally addressed ad hoc by a team of specialists. For example, many prior investigations evaluate various SK methods along multi-body orbits, including techniques based on crossing control [6, 7] and dynamical systems theory [4], RL [8, 9], among others [10]. However, there is substantially less research on autonomous recovery given an inadvertent departure from a libration point orbit, with prior research efforts regarding NRHOs focusing primarily on intentional departure for both heliocentric disposal [11] and lunar impact [12]. In this chaotic region of space, large deviations over time rapidly render many traditional SK approaches ineffective. This investigation leverages RL to determine accurate initial guesses for low-thrust recovery maneuver sequences, both lengthening the possible recovery window for a potential onboard application and demonstrating the utility of blending RL with traditional G&C techniques.

In the absence of a pre-determined contingency plan, inadvertent departure from a desired orbit path poses serious risk to mission success. For spacecraft on an NRHO, regular SK maneuvers are necessary to maintain the orbit [6]. However, in a missed-thrust scenario where multiple SK maneuvers are skipped, determining a recovery path is challenging, with significant additional complexity introduced for autonomous, low-thrust scenarios. While employing an NRHO as a nearly stable baseline orbit is advantageous due to a slow departure rate, it is correspondingly challenging to re-enter this region of near-stability. In addressing this challenge, Zimovan-Spreen et al. present a thorough analysis of NRHO departure characteristics and offer several potential recovery methods [13]. In their analysis, the suggested recovery response is determined by which of two 'regimes' the departing flow is categorized. Regime 1 signifies that a standard SK approach readily converges to a recovery plan, and Regime 2 encompasses situations where the departing flow remains in the lunar vicinity, but is not recoverable by SK maneuvers (a third regime for cases that depart the lunar vicinity entirely is identified as future work). The authors leverage known structures and dynamical systems theory to formulate an impulsive recovery framework for trajectories in 'Regime 2'. While offering many potential recovery options, the computational footprint, human-in-the-loop interaction, and impulsive engine assumption present significant barriers to any potential onboard, autonomous implementation for low-thrust spacecraft. This investigation addresses a subset of recovery scenarios within a low-thrust analogue to Zimovan-Spreen et al.'s 'Regime 2' by employing a deep neural network to aid low-thrust recovery planning for cases where SK no longer maintains the desired NRHO.

While research into NN control stability in aerospace applications is ongoing [14], several previously suggested G&C approaches simultaneously benefit from NN function approximation while still ensuring that the associated approximation errors do not impact mission safety. In these paradigms, rather than the NN directly controlling the spacecraft, safety assurances are accomplished by instead tasking the NN with estimating startup solutions for conventional iterative algorithms that incorporate mission constraints. The majority of previous applications of these NN startup identification methods involve leveraging behavior cloning with NNs to estimate costates for indirect optimal control problems [15–17]. Furthermore, prior investigations introduce a 'hybrid' G&C targeting architecture that employs post-processing in conjunction with a pre-trained NN to compute initial conditions for targeting [18, 19]. This investigation similarly employs targeting, but offers significant improvement on the hybrid approach by 1) directly incorporating targeting into the learning process, and 2) detailing a time-varying training scheme that results in a remarkably adaptive maneuver planning process.

Recent investigations demonstrate RL as a promising tool in spaceflight G&C applications with potential for onboard use, with several previous research efforts demonstrating RL for G&C along known reference paths in multi-body dynamical regimes. Investigations in the Earth-Moon system include guidance along heteroclinic transfers between Lyapunov orbits [20], as well as "multi-objective" RL to guide a spacecraft into a Lyapunov orbit [21] and between Lyapunov and halo orbits [22]. Furthermore, several investigations employ RL to compute or adjust SK maneuvers that maintain libration point or-

bits, with approaches including Q-Learning [23], Floquet-model control augmentation [24], momentum unload offsets [8], and low-thrust maneuver identification and placement [9]. Another branch of relevant and promising research involves path-planning, where, rather than following a pre-selected trajectory, the RL training process is instead tasked with uncovering connections between two distinct orbits [25–27]. This investigation builds on previous contributions in multi-body RL by demonstrating the utility of incorporating targeting directly into training.

2. Problem Formulation

This investigation focuses on low-thrust mission recovery given an inadvertent departure from an NRHO. The low-thrust Circular Restricted Three Body Problem (CR3BP) is employed in this analysis because it accurately represents the NRHO region while remaining sufficiently low fidelity for initial analysis of the NNIT scheme. Next, to demonstrate applicability in a realistic mission scenario, the N-body ephemeris force model is introduced along with specific characteristics of the modeled Lunar IceCube spacecraft. Furthermore, a flexible multiple shooting algorithm is detailed in this analysis to demonstrate the utility of employing a NN in an initial guess generation procedure.

2.1 Dynamical Model: CR3BP with Low-Thrust

The CR3BP [28] is a model for the motion of a spacecraft with infinitesimal mass moving under the mutual gravitational influence of two celestial bodies. In this model, as depicted in Fig. 1, the two spherically symmetric gravitational bodies, assumed in this investigation to be the Earth (P_1) and the Moon (P_2), form the primary system as they move in circular orbits about their common barycenter, B. The relative size of the primaries is represented by the mass ratio $\mu = m_2/(m_1+m_2)$. Nondimensionalization is leveraged for flexibility in applications as well as numerical stability. The characteristic length, l^*, represents an average distance between the Earth and the Moon, and characteristic time, t^*, is defined such that the nondimensional gravitational constant is equal to one. The nondimensional position and velocity three-dimensional vectors for P_3 with respect to B comprise the six-dimensional vector $\boldsymbol{\rho} = [x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}]^T$. The vector components are propagated with respect to the system barycenter, B, in a reference frame that rotates with the motion of P_2, denoted by dashed lines in Fig. 1. The nondimensional CR3BP equations of motion in the rotating frame are expressed as,

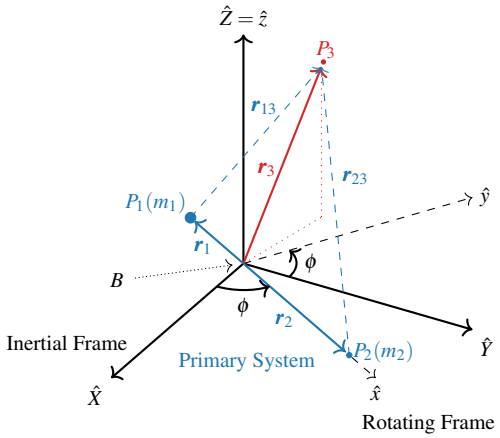


Fig. 1. Vector definitions in the CR3BP.

$$\ddot{x} - 2\dot{y} = \Omega_x \qquad \ddot{y} + 2\dot{x} = \Omega_y \qquad \ddot{z} = \Omega_z \tag{1}$$

where Ω_i are derivatives of a pseudo-potential function,

$$\Omega = \frac{(1-\mu)}{r_{13}} + \frac{\mu}{r_{23}} + \frac{x^2+y^2}{2} \tag{2}$$

The distances between P_3 and the first and second primary body are defined as $r_{13} = \left((x+\mu)^2+y^2+z^2\right)^{1/2}$ and $r_{23} = \left((x-1+\mu)^2+y^2+z^2\right)^{1/2}$, respectively. The natural CR3BP equations of motion yield a well-known integral, i.e., the Jacobi constant of integration (C). This single integral of motion is evaluated as,

$$C = 2\,\Omega - \left(\dot{x}^2+\dot{y}^2+\dot{z}^2\right) \tag{3}$$

The Jacobi constant is related to orbital energy as observed in the rotating frame and remains constant throughout any natural propagation.

Many mission architectures benefit from the inclusion of low-thrust Solar Electric Propulsion (SEP). In contrast to traditional chemical engines, electric propulsive engines are much more efficient, but deliver energy changes over longer time intervals. Low-thrust engines using SEP include ion thrusters that are powered through solar panels on the spacecraft. Currently, ion thrusters are successfully employed on various missions [29], e.g., Deep Space 1 and Dawn. Building on this progress, upcoming low-thrust missions include Lunar IceCube, Psyche, and the Lunar Gateway.

In this analysis, over any low-thrust CR3BP integration segment, the thrust direction is assumed fixed in the rotating frame. Two simple coordinate definitions are leveraged to represent thrust direction in this investigation. First, $\hat{u}$ is a unit vector in the rotating frame. Second, spherical coordinates are employed in targeting applications, and are related to $\hat{u}$ via the transformation,

$$\hat{u} = \begin{bmatrix} \cos\theta\sin\kappa & \sin\theta\sin\kappa & \cos\kappa \end{bmatrix}^T \tag{4}$$

where θ and κ represent angles between $\hat{u}$ and the rotating $\hat{x}$ axis and $\hat{x}$-$\hat{y}$ plane, respectively. Motion in the CR3BP

is nonlinear in a notably sensitive dynamical regime, thus, the proposed NNIT strategy exploits the impact of the low-thrust terms to achieve desired behavior. While a continuously changing inertial thrust direction may not be practical, precession of the rotating frame during thrusting time intervals is addressed when transferring CR3BP solutions into a higher-fidelity model

This investigation bases its low-thrust spacecraft model on the Lunar IceCube CubeSat, and assumes a Constant Specific Impulse (CSI) engine. The impact of low-thrust propulsion on the motion of the spacecraft is modeled by defining a nondimensional thrust magnitude, f, and acceleration magnitude, a^{lt},

$$f = \frac{Ft^{*2}}{l^* M_{3,0}}, \qquad a^{\text{lt}} = \frac{f}{m} \tag{5}$$

where F and $M_{3,0}$ are thrust magnitude and initial mass, respectively [30]. Here, capital letters signify dimensional values, where lowercase letters reflect nondimensional quantities. The spacecraft mass is defined as $m = M_3/M_{3,0}$, with M_3 representing the mass of the spacecraft at the beginning of the thrusting segment. The low-thrust CR3BP equations of motion are formulated by augmenting the natural CR3BP equations of motion, Eq. (1), with low-thrust terms, and introducing a linear mass flow rate,

$$\begin{cases} \ddot{x} - 2\dot{y} &= \Omega_x + a^{\text{lt}} u_x \\ \ddot{y} + 2\dot{x} &= \Omega_y + a^{\text{lt}} u_y \\ \ddot{z} &= \Omega_z + a^{\text{lt}} u_z \\ \dot{m} &= \dfrac{-fl^*}{I_{\text{sp}} g_0 t^*} \end{cases} \tag{6}$$

Propulsive capability is inversely related to spacecraft mass and, hence, as propellant is expended, the spacecraft is capable of higher acceleration values. Furthermore, the equivalent ΔV for a low-thrust maneuver is derived from the ideal rocket equation,

$$\Delta V_{\text{equiv.}} = I_{\text{sp}} g_0 \log\left(\frac{m_0}{m_f}\right) \tag{7}$$

and provides an intuitive measure of the change in velocity for a CSI engine.

2.2 N-Body Ephemeris Model

While the present investigation focuses primarily on the CR3BP to demonstrate the proposed NNIT process, and corresponding RL training framework, an N-Body simulation is employed to explore validity in a higher-fidelity ephemeris force model. The N-Body differential equations describe the motion of a massless particle m_i with respect to a primary body m_q, with the remaining $N-2$ bodies included as perturbing gravitational forces. In this investigation, the Moon serves as the central body, with the Sun, Earth, and Jupiter comprising the perturbing bodies. The Moon-centered J2000 inertial reference frame is employed to represent and propagate state variables; rotating and J2000 reference frame transformations are detailed in [31]. As in the CR3BP, a low-thrust acceleration vector augments the natural N-body system of differential equations, resulting in the vector expression,

$$\ddot{\boldsymbol{r}}_{qi} = -G\frac{m_i + m_q}{r_{qi}^3}\boldsymbol{r}_{qi} + G\sum_{\substack{j=1 \\ j\neq i,q}}^{N} m_j \left(\frac{\boldsymbol{r}_{ij}}{r_{ij}^3} - \frac{\boldsymbol{r}_{qj}}{r_{qj}^3}\right) + a^{\text{lt}}\hat{u}_j \tag{8}$$

where G is the universal gravitational constant, $\boldsymbol{r}_{[a][b]}$ represents the relative position of body $[b]$ with respect to body $[a]$, $\hat{u}_j$ denotes a J2000-fixed thrust direction, and a^{lt} is defined in Eq. (5). Planetary ephemerides at specific points in time are obtained using the DE440 SPICE kernel, retrieved from JPL's Navigation and Ancillary Information Facility (NAIF) [32].

2.3 Spacecraft Model: Lunar IceCube

To more accurately model a realistic mission scenario, this investigation bases its spacecraft model and mission application on the upcoming Lunar IceCube mission. The Lunar IceCube spacecraft is a 6U CubeSat equipped with a Busek Ion Thruster 3-cm (BIT-3) low-thrust propulsion system [33], with specific spacecraft and engine characteristics listed in Table 1.

Table 1. Lunar IceCube spacecraft data

Parameter	Symbol	Value
Initial mass	$M_{3,0}$	13.487 kg
Specific impulse	I_{sp}	2156 s
Maximum thrust	$F_{\max}$	1.1 mN
	$f_{\max}$	0.02992 nondim

Lunar IceCube is on-course to become the first low-thrust spacecraft to access an NRHO (and the second spacecraft overall after the CAPSTONE mission [34]). Set to launch as a secondary payload on the Artemis 1 mission, the primary science objective is to prospect, locate, and study ice, vapor, and liquid water features on the Moon from a highly-inclined 100-km perilune low-lunar science orbit [35, 36]. Lunar IceCube's transit trajectory incorporates a 9:2 lunar synodic resonant L_2 southern NRHO, depicted in Fig. 2, as a staging orbit prior to the spiral-down leg into its science orbit [37], and performs at least one SK maneuver to maintain the NRHO. The 9:2 NRHO is of particular recent interest as the planned baseline orbit for the Lunar Gateway [38]. This investigation simulates a scenario in which a disturbance causes Lunar IceCube to significantly deviate from the desired NRHO, and a low-thrust maneuver plan is sought to return the spacecraft to the NRHO region. While Lunar IceCube

is leveraged to model a realistic scenario, the proposed methodology is potential applicable to additional spacecraft, thrust capabilities, and reference trajectories.

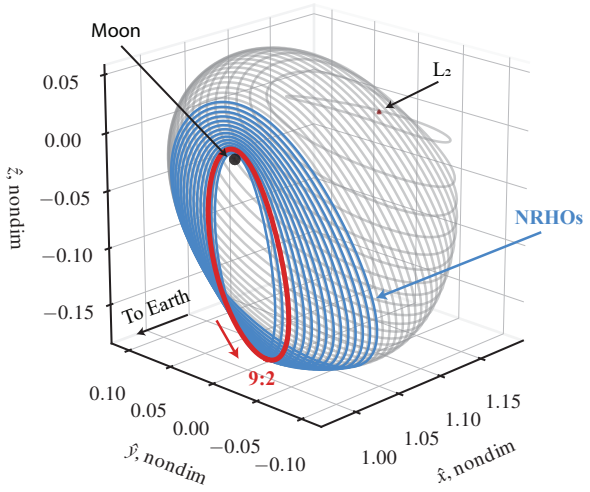


Fig. 2. The Earth-Moon L_2 southern halo orbit family, where blue denotes the NRHO region, and red signifies the NRHO at 9:2 lunar synodic resonance.

2.4 Differential Corrections

The implemented RL recovery training process for NNIT results in a NN that estimates initial guesses for a targeting algorithm. Targeting is formulated as a multidimensional generalization of a Newton-Raphson method [39] and is a common numerical technique in astrodynamics for constructing feasible trajectories. In particular, this investigation employs a direct multiple shooting strategy to simultaneously accommodate numerous types of arcs and control variables across a maneuver plan, to correct discontinuities, and to ensure mission criteria are met. Targeting is selected as the underlying G&C strategy in this research due to onboard use for Orion [2], and applicability in low-thrust problems [17].

The proposed targeting algorithm is implemented by varying an array of design variables, $\boldsymbol{X}$, to satisfy a set of scalar constraints, $\boldsymbol{F}(\boldsymbol{X}) = \boldsymbol{0}$. Similar to other iterative approaches, targeting algorithms require sufficiently accurate startup solutions. Selecting an initial guess is challenging in complex dynamical regions of space and, hence, the NNIT approach benefits from the startup accuracy and speed offered by a NN controller. If sufficient computational resources are available, optimization techniques are a potential alternative or additional approach for the targeting step, however, this possibility is not investigated.

The recovery initial guess is comprised of: 1) $\mathtt{n}$ low-thrust arcs with control variable time history computed by a NN controller, and 2) $\mathtt{m}$ stacked ballistic revolutions of the desired periodic orbit. By employing a minimum norm update, a solution is sought that maintains the stacked orbit states close to their original value, typically producing motion in the close vicinity of the desired orbit without precisely constraining the final state. Constraining the terminal condition to match a desired six-dimensional state vector presents additional challenges that are not always necessary or practical in the presence of navigation uncertainty and maneuver execution errors. An alternative targeting approach is to constrain the final state to be on the stable manifold, as detailed by Haapala and Howell [40]. While the stable manifold constraint approach preserves energy and periodicity, it poses challenges for a rapid closed-loop system due to the sensitivities in response to initial time parameters. Stacking revolutions is a more flexible approach if consistent closed-loop convergence is required, and $\mathtt{m} = 4$ revolutions is employed in this investigation.

The free variables $\boldsymbol{X}$ and constraint vector $\boldsymbol{F}(\boldsymbol{X})$ of the proposed targeting process are defined as,

$$\boldsymbol{X} = \left(\underbrace{\left\{\boldsymbol{\rho}_i\right\}_{i=1}^{\mathtt{n}+\mathtt{m}}}_{\text{States}} \quad \underbrace{\left\{t_i \quad \eta_i\right\}_{i=0}^{\mathtt{n}+\mathtt{m}}}_{\text{Propagation times}} \quad \underbrace{\left\{\zeta_i \quad \theta_i \quad \kappa_i\right\}_{i=0}^{\mathtt{n}}}_{\text{Controls}} \right)^T \tag{9}$$

$$\boldsymbol{F}(\boldsymbol{X}) = \left(\underbrace{\left\{\boldsymbol{\rho}_{i,f} - \boldsymbol{\rho}_{i+1,0}\right\}_{i=0}^{\mathtt{n}+\mathtt{m}-1}}_{\text{State Continuity}} \quad \underbrace{\left\{t_i - \eta_i^2\right\}_{i=0}^{\mathtt{n}+\mathtt{m}}}_{\text{Ensure } t_i > 0} \right)^T \tag{10}$$

Throughout the differential corrections process, the state vector $\boldsymbol{\rho}_i$ at each intermediate patch point is allowed to vary; the initial state $\boldsymbol{\rho}_0$ is assumed fixed. Propagation times t_i are similarly included as design variables for both thrusting and coasting segments, with additional slack variables $\eta_i \in [-\infty, \infty]$ included to enforce the $t_i > 0$ inequality constraint. Thrust magnitude and direction are design variables for the initial $\mathtt{n}$ thrust segments, with direction represented in spherical coordinates (θ and κ are defined in Eq. (4)). Furthermore, to ensure that the thrust magnitude remains bounded, i.e., $f_i \in [0, f_{\max}]$, a parameterized thrust magnitude ζ_i is introduced such that, $f_i = f_{\max} \cdot (\sin(\zeta_i) + 1)/2$. The new unbounded design variable ζ_i is more easily included in the targeting process and guarantees $0 \leq f_i \leq f_{\max}$. An alternative approach introduces additional slack variables to enforce the inequality constraints, however, this method may diverge when f_i is close to either bound. One benefit, or drawback, of employing ζ_i is that no update is produced when $f_i = 0$ or $f_i = f_{\max}$. The design variables, together, comprise the $\boldsymbol{X}$ vector, which is updated iteratively to satisfy the $\boldsymbol{F}(\boldsymbol{X}) = \boldsymbol{0}$ constraints. In this investigation, the enforced constraints in Eq. (10) are simply state continuity constraints between successive patch points, and positive value inequality constraints on all time variables.

3. Reinforcement Learning Formulation

Reinforcement learning is a branch of ML that encompasses a broad range of goal-oriented algorithms that 'learn' to perform tasks by means of trial-and-error. Current state-of-the-art RL approaches frequently employ NNs to parameterize the control function for a given problem. Actor-critic methods are of particular recent interest due to their demonstrated ability in continuous control tasks. One such algorithm, Twin Delayed Deep Deterministic Policy Gradient (TD3) [41], is employed in this investigation.

3.1 Neural Networks

An artificial neural network (NN) is a class of nonlinear computational models that are frequently employed in ML tasks [42]. While ubiquitous in supervised learning applications, NNs are also employed extensively in modern RL algorithms due to their demonstrated ability in approximating nonlinear functions. Many traditional tabular RL approaches, such as Q-learning, rely on finely discretizing the state and action spaces, quickly becoming intractable as the number of dimensions in the problem increases. Thus, leveraging NNs allows modern algorithms to both access continuous state and action spaces and to easily incorporate additional dimensions.

Each node in a NN is computed by adding the linear combination of weighted values from the previous layer to a bias value, and processing the result through an activation function to incorporate nonlinearity into the model [42]. Without activation functions, the NN only models linear functions and, thus, the selection of the activation function is an important component in NN performance. Furthermore, bounded functions are often advantageous since they aid in normalizing the output of each neuron. To incorporate nonlinearity, this investigation employs the rectified linear unit (ReLU) function for all hidden layers, and hyperbolic tangent (tanh) to bound outputs for actor networks.

Neural networks are potentially well-suited for a flight computer, with demonstrated capability on hardware suitable for spaceflight [43]. While training is computationally complex, the evaluation process is relatively simple. Furthermore, current flight software prototype investigations leverage iterative G&C algorithms onboard [1, 2]. These repetitive operations pose challenges in both predicting the total number of CPU cycles, and in excessive iterations that result from an inaccurate initial guess. The NN employed in this investigation is evaluated in constant time and possesses a relatively small 0.5 MB memory footprint, thus offering a rapid means of identifying startup solutions. Augmenting iterative methods with a NN offers potential improvements in both extending the range of convergence and in reducing the total number of necessary iterations.

3.2 Actor-Critic Reinforcement Learning

Reinforcement learning is a class of algorithms in which a goal-seeking agent seeks to complete a task by means of interaction with an environment. As visualized in Fig. 3, three signals facilitate this communication cycle at time t: 1) a state signal feature vector, $\boldsymbol{s_t}$, that communicates relevant information about the environment at a specific point in time, 2) an action vector, $\boldsymbol{a_t}$, that allows the agent to affect its state when processed through the environment's dynamics, and 3) a scalar reward, $r(\boldsymbol{s_t}, \boldsymbol{a_t})$, that communicates the immediate benefit of a given action. This process repeats iteratively for a given task. Over many trials, the agent improves by seeking to maximize accumulated reward. In episodic tasks, terminal conditions exist that cease the learning process. In these cases, the environment typically resets to a randomly seeded initial configuration, and the process begins anew. While the RL literature prefers the terms *agent*, *environment*, and *action*, these expressions are analogous to the more common engineering terms *controller*, *controlled system* (or *plant*), and *control signal* [44].

An agent's policy $\pi : \mathcal{S} \rightarrow p(\mathcal{A})$ is defined as a mapping of the set of all possible states $\mathcal{S}$ to a probability distribution over the action space $\mathcal{A}$. Reinforcement learning aims to construct a policy to maximize the expected return, quantified as a balance between immediate and future rewards,

$$R_t = \sum_{i=t}^{T} \gamma^{(i-t)} r(\boldsymbol{s_i}, \boldsymbol{a_i}) \tag{11}$$

where $\gamma \in [0, 1]$ is the discount factor that determines the extent to which the agent prioritizes immediate rewards (typically near 1). This definition of expected return leads to the formalization of the action-value function Q_π, defined as the expected return of taking action $\boldsymbol{a_t}$ at $\boldsymbol{s_t}$, and subsequently following the policy π:

$$Q_\pi(\boldsymbol{s_t}, \boldsymbol{a_t}) = \mathbb{E}_{\boldsymbol{a_i} \sim \pi} \left[R_t | \boldsymbol{s_t}, \boldsymbol{a_t} \right] \tag{12}$$

In practice, many modern algorithms estimate $Q_\pi(\boldsymbol{s_t}, \boldsymbol{a_t})$ using a deep NN, termed a 'critic' network.

Policy optimization is a branch of RL methods that seek to directly 'learn' a parameterized policy, $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})$, where $\boldsymbol{\theta}$ represents a parameter vector. A particularly

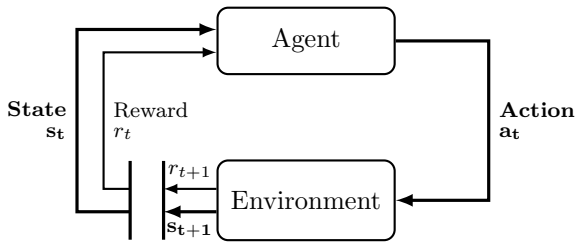


Fig. 3. The agent-environment communication process (reproduced from Sutton and Barto, Ref [44], p.48).

fruitful branch of policy optimization methods emerges from a class of hybrid algorithms, identified as actor-critic methods. These approaches seek both the policy (i.e., *actor*) and the value (i.e., *critic*) functions, where both the actor and critic are typically represented as NNs. In these algorithms, the actor's policy is stochastic during the training phase to enable exploration. A multivariate Gaussian distribution is constructed over the action space, with mean values delivered from the actor NN, and a constant variance applied to each variable (action variables are assumed to be uncorrelated). An optimal policy is gradually approached over the course of training. Once fully trained, exploration is no longer necessary, so the mean values are employed directly to form a deterministic controller. In contrast to some value optimization methods, such as the classical Q-Learning approach, policy optimization methods are well suited to higher-dimensional problems that require a continuous action space. Common state-of-the-art actor-critic schemes include TD3 [41], Soft Actor Critic (SAC) [45], and Proximal Policy Optimization (PPO) [46].

A key delineation between modern actor-critic methods surrounds an assumption concerning the policy distribution from which the actions are sampled. In "on-policy" algorithms, e.g., PPO, update equations assume all actions are sampled from the most recent version of the parameterized policy, $\pi_{\boldsymbol{\theta}}$, causing the current data batch to be discarded following each optimization process. In contrast, "off-policy" approaches, actions may be sampled from any distribution, allowing for improved learning efficiency. While the state-of-the-art landscape is constantly evolving in RL, both on-policy (PPO) and off-policy (TD3 and SAC) schemes are demonstrated as effective in continuous control tasks, and this investigation employs TD3 due to favorable data efficiency properties.

3.3 Twin Delayed Deep Deterministic Policy Gradient

Various off-policy RL approaches leverage the state-action value function $Q(\boldsymbol{s_t}, \boldsymbol{a_t})$, updated by iterating on the Bellman equation, to formulate an effective policy. An early breakthrough for problems with discrete state and action spaces occurred in 1989 with Q-Learning: a temporal difference learning algorithm that allows powerful agents to be trained in an off-policy manner [47]. The core motivation for the Q-Learning approach is simple: if the $Q_\pi(\boldsymbol{s_t}, \boldsymbol{a_t})$ function is known, that is, the value of all state action pairs is accurately represented, then an optimal policy is available by simply selecting the available action that possesses the largest $Q_\pi(\boldsymbol{s_t}, \boldsymbol{a_t})$ value. While Q-Learning is very effective, it was limited, along with other RL algorithms at the time, by the assumption of discrete spaces due to the lack of effective nonlinear function approximators.

A major breakthrough in RL came in 2015 when the Q-Learning algorithm was extended to continuous state space problems with the Deep Q-Network (DQN) algorithm [48]. The DQN approach involves the same update rule as Q-Learning, but rather than direct computation, the Q-function is effectively estimated using a NN: alleviating the need for tabulated results. However, similar to Q-Learning, DQN still assumes a discrete action space. While acceptable for tasks with a small, finite number of controls (such as Atari games), the curse of dimensionality in the action space limits applicability for continuous control tasks, such as robotics.

To address the action-space dimensionality limitation with DQN, Lillicrap et al. introduce Deep Deterministic Policy Gradient (DDPG) [49]: an extension of DQN and Deterministic Policy Gradient (DPG) [50], that employs a second "actor" NN to construct a parameterized action function, and performs the $Q(\boldsymbol{s_t}, \boldsymbol{a_t})$ update with the new actor network. A key portion of DDPG is the inclusion of "target" networks for both the actor and critic networks. By including duplicate networks, the second "target" network may be held constant while the actual network is updated. A more stable process results because it doesn't involve continuously updating estimates to be used in the minibatch updates. While powerful, DDPG contains several notable limitations. In particular, the value function is often overestimated, which leads to a significant bias in learning.

Twin Delayed Deep Deterministic Policy Gradient (equivalently, Twin Delayed DDPG, or TD3) [41] is a state-of-the-art off-policy actor-critic algorithm that concurrently learns a parameterized policy $\pi_{\boldsymbol{\theta}}$ and action-value function $Q_\pi(\boldsymbol{s_t}, \boldsymbol{a_t})$ [41]. Twin Delayed DDPG addresses the instability of DDPG in several ways. First, two separate value networks are included. Incorporating the minimum value estimate from the two networks mitigates the effect of the value function overestimation bias present in DDPG (along with other actor-critic schemes). Next, DDPG involves bootstrapping, i.e., implementing updates based on estimates rather than true values. Noisy estimates cause noisy gradients that pose significant difficulty in NN optimization. The TD3 approach seeks to mitigate this error by delaying updates to the actor network in hopes that additional updates to the critic network provide more accurate estimates for the actor updates. Finally, to avoid peak overfitting in the policy network, policy smoothing is included, that introduces a small amount of clipped random noise to the output of the target policy. Together, these improvements form the TD3 variant of DDPG. While the inclusion of function approximations demands increased complexity, TD3 shares the same core motivation of Q-Learning: optimize a policy network to compute actions that maximize the Q-function. This investigation bases its implementation of TD3 on the open-source "Spinning Up" Tensorflow implementation pro-

vided by OpenAI [51], with actor-critic NN architectures detailed in Table 2.

Table 2. Configuration of actor and critic neural networks employed in this investigation.

	Actor Network		**Critic Network**	
Layer name	Size	Activation	Size	Activation
Input	16	-	21	-
Hidden 1	400	ReLU	400	ReLU
Hidden 2	300	ReLU	300	ReLU
Output	5	tanh	1	linear
Learning Rate	0.0001		0.001	

4. Learning Environment

Implementing a spaceflight problem as an RL environment involves formalizing the three signals in Fig. 3, as well as the environment's transition dynamics that propagate one state to the next. This investigation involves a mission recovery scenario, and each signal is understood in this context. The state signal provides the agent with relevant information about its dynamical state, the action defines low-thrust command variables, and the reward communicates information regarding the distance relative to the reference orbit, as well as terminal success/failure criteria. The environment's dynamics are governed by the low-thrust CR3BP equations of motion, Eq. (6).

4.1 Scenario Overview

In each episode, the agent is tasked with recovering in response to an induced departure from a reference NRHO. Prolonged departures are generated using similar methodology to [13], where a randomly-selected initial state along the orbit is corrupted, with perturbation values for each position and velocity component sampled from independent and identically distributed Gaussian distributions (3σ: 10 km, 10 cm/s). These perturbed states are then propagated forward in time for 20-30 days to simulate an inadvertent departure from the NRHO. This time range is selected to introduce significant deviations that still generally admit 'direct' returns to the NRHO. The end of the inadvertent departure trajectory serves as the initial condition for the RL episode.

Each episode consists of a fixed number of time-steps ($\mathtt{n} = 10$ in this investigation), after which the resulting low-thrust trajectory serves as an initial guess for the differential corrections algorithm defined by Eqs. (9) and (10). At each time step, the agent selects a thrust magnitude, direction, and time, which are then numerically integrated by the environment. The agent, therefore, seeks to determine a sequence of $\mathtt{n}$ low-thrust arcs that terminate sufficiently close to the NRHO to achieve subsequent targeting convergence. Furthermore, experiments demonstrate that this framework admits long-duration coasting segments through the selection of near-zero thrust magnitude values. In including coasting segments and allowing for large variations in propagation time, the proposed learning process demonstrates remarkable adaptability in selecting a maneuver plan. However, while targeting offers many advantages in the NNIT process, one drawback is the increase in training time. This investigation overcomes this limitation by leveraging parallel computing between NN updates to rapidly simulate training episodes.

Previous research efforts in implementing RL to reach a desired orbit frequently employ arbitrary heuristics to formalize termination conditions, including the application of pre-defined position and velocity thresholds [18, 26]. A thresholding approach is particularly challenging in NRHO problems, where larger velocity variations occur at the close perilune passes and, conversely, position deviations grow near apolune. If heuristic thresholds are employed in such a problem, the agent is unlikely to achieve satisfactory relative position and velocity values simultaneously due to their inverse relationship. Instead, this investigation leverages convergence properties of the NNIT targeter to produce a more informed terminal "success" metric for accessing a pre-selected destination orbit.

4.2 State Signal

The state signal is included as an input to both the actor and critic networks and, hence, proper selection of the feature variables greatly impacts learning performance. In the recovery scenario, the dynamical state $\boldsymbol{\rho}_t$ informs the agent of its position and velocity in space, while the relative vector $\boldsymbol{\delta\rho}_t$ provides information about the spacecraft behavior relative to the NRHO. The relative state $\boldsymbol{\delta\rho}_t$ is measured from the nearest neighbor along the NRHO, defined as

$$\boldsymbol{\delta\rho}_t = \boldsymbol{\rho}_t - \boldsymbol{\rho}_{\text{ref}} \quad \text{s.t.} \quad k = ||\boldsymbol{\delta\rho}_t||_2 \ \text{ is minimal} \tag{13}$$

where $\boldsymbol{\rho}_{\text{ref}}$ is a state along the reference orbit. A KD-tree is employed to facilitate rapid nearest neighbor searching, as detailed in [18]. Note that this definition of "nearest" does not include time and, hence, is a normal rather than an isochronous correspondence. Furthermore, an additional coordinate is employed to measure the location of $\boldsymbol{\rho}_{\text{ref}}$ along the NRHO, where t_{po} signifies the elapsed time along $\boldsymbol{\rho}_{\text{ref}}$ from a specified initial state (apolune in this investigation). As detailed in [9], time is a cyclic variable for periodic orbits and, hence, a trig encoding is leveraged to remove end-point discontinuities for the NN, i.e.,

$$t_{\text{po}} \rightarrow [\sin\xi, \cos\xi], \quad \text{where } \xi = \frac{t_{\text{po}} \cdot 2\pi}{\mathbb{P}} \tag{14}$$

where $\mathbb{P}$ is the period of the orbit. Finally, the difference in the Jacobi constant value between $\boldsymbol{\rho}_t$ and $\boldsymbol{\rho}_{\text{ref}}$ (δC_t), is

included to communicate the energy disparity. Together, the complete state signal is defined as,

$$\boldsymbol{s_t} = \begin{bmatrix} \boldsymbol{\rho}_t & m & \boldsymbol{\delta\rho}_t & \delta C_t & \sin(\xi_t) & \cos(\xi_t) \end{bmatrix} \in \mathbb{R}^{16} \quad (15)$$

While each component of the state signal is leveraged in the resulting policies, experiments demonstrate the relative state as especially critical in learning problems that involve reference motion.

4.3 Action Signal

The action signal quantifies the agent's ability to command the spacecraft. In this investigation, low-thrust arcs are fundamentally comprised of three components: thrust magnitude, direction, and propagation time, with each estimated separately by the NN. Each action value is bounded by $[-1, 1]$ due the selection of tanh as the output activation function in the actor NN. In this investigation, 'tilde' denotes the bounded output value retrieved directly from the NN. The action vector is defined as,

$$\boldsymbol{a_t} = \begin{bmatrix} \tilde{f} & \tilde{u}_x & \tilde{u}_y & \tilde{u}_z & \tilde{\tau} \end{bmatrix} \in \mathbb{R}^5 \quad (16)$$

where thrust time and magnitude variables are decoded as,

$$f = (\tilde{f}+1) f_{\max}/2, \qquad \tau = (\tilde{\tau}+1)\tau_{\max}/2 \quad (17)$$

Here, a minimal value for f results in nearly-ballistic coasting segments. While the maximum thrust magnitude, $f_{\max}$, is a characteristic of the spacecraft's engine, no obvious value emerges for the pre-selected maximum time, $\tau_{\max}$. If the selected value for $\tau_{\max}$ is too small, the spacecraft may not possess sufficient maneuver-planning authority to solve the given problem. If the value is too large, small errors in control estimation become exacerbated, and the agent is more likely to converge on a propellant-inefficient policy. This investigation employs $\tau_{\max} = 24$ hours to allow for long-duration thrust and coast arcs necessary in the recovery phase. However, this choice limits the ability to accurately estimate the short burns often employed for SK, thus, increasing the NNIT cost at low error levels.

The final components of the thrust command are the direction values $\tilde{u}_i$, which are combined and normalized to form a unit vector $\hat{u}$ fixed in the rotating frame,

$$\hat{u} = [u_x \;\; u_y \;\; u_z] = \frac{[\tilde{u}_x \;\; \tilde{u}_y \;\; \tilde{u}_z]}{\sqrt{\tilde{u}_x^2 + \tilde{u}_y^2 + \tilde{u}_z^2}} \quad (18)$$

While this parameterization is demonstrated as successful in related tasks [9, 18], an alternative option in [27] is to locate the thrust direction using spherical coordinates, Eq. (4), coupled with an encoding method to handle the cyclic angle values, similar to the process employed in Eq. (14). Together, f, τ, and $\hat{u}$ form the complete thrust command for the spacecraft.

4.4 Reward Signal

For non-terminal states, as suggested by LaFarge et al. [18], the reward function is modeled as an exponential that grows rapidly as the agent's state nears the desired orbit in both position and velocity, with a small control penalty added to encourage propellant-efficient solutions and to allow near-ballistic segments to emerge. Trials are terminated when either a large deviation threshold is exceeded, or the agent reaches $\mathtt{n}$ time steps. Relative distance is measured from the nearest neighbor along the reference orbit, as defined in Eq. (13). Deviation is quantified when either the relative position or velocity magnitudes exceed pre-defined thresholds of 8000 km and 250 m/s, or the spacecraft impacts the Moon. Together, then, the reward is defined as,

$$r = \begin{cases} \beta_1 e^{-\lambda k} - \beta_2 \Delta V_{\text{equiv.}} & \text{not deviated} \\ b - \beta_3 \Delta V_{\text{tot.}} & \text{targeter converges} \\ p_1 & \text{deviation criteria met} \\ p_2 & \text{targeter does not converge} \end{cases} \quad (19)$$

Each term that comprises the total reward function is defined in Table 3 along with the corresponding suggested values employed in this investigation.

5. Mission Recovery Scenario: Lunar IceCube

An NRHO recovery problem is simulated in this investigation to illustrate the proposed NNIT framework as applied to the Lunar IceCube spacecraft. During training, recovery situations are simulated by introducing deviations from the reference orbit, and propagating those deviations forward in time. To verify the validity of this deviation modeling, a higher-fidelity simulation is introduced. This scenario assumes the Lunar IceCube spacecraft previously reached the 9:2 NRHO, and subsequently implemented several small SK maneuvers using the low-thrust variant of x-axis control (XAC) detailed in [6]. In reality, Lunar IceCube intends to access the NRHO for a shorter period of time; this scenario considers additional SK maneuvers and a longer recovery window to evaluate algorithmic performance. For this simulation, during the NRHO SK phase, 3σ errors of 10 km, 10 cm/s are modeled as an uncorrelated Gaussian process for each position and velocity component for both orbit determination and orbit insertion errors; a fixed 0.03 cm/s maneuver execution error is added in a random direction for each SK maneuver. Stationkeeping maneuvers are implemented at apolune at a once-per-orbit cadence. Leveraging XAC to initialize the departure simulation provides a more realistic scenario to evaluate NNIT performance.

5.1 Unintended Departure Simulation

To illustrate the RL recovery process in a practical situation, a single representative scenario is considered. In this sample problem, following its fifth orbit maintenance

Table 3. Components for reward functions leveraged in this investigation

Symbol	Description	Value(s)
β	Tuning coefficients that adjust the relative impact of each term	$\beta_1 = 1, \beta_2 = 0.03, \beta_3 = 0.45$
k	Relative state norm, defined in Eq. (13)	-
λ	Scaling factor that adjusts the gradient of the reward	300
b	Bonus applied for achieving targeting convergence	28
$\Delta V_{\text{tot.}}$	Shorthand for the sum of the $\mathbb{n}$ $\Delta V_{\text{equiv.}}$ values	-
p_1	Penalty for violating deviation criteria	-25
p_2	Penalty for targeter not converging in 10 iterations or fewer	-2

maneuver, an unanticipated thruster failure event is assumed for Lunar IceCube that causes SK to cease entirely. An inadvertent departure from the NRHO results, causing the 'runaway' spacecraft to slowly drift from its desired orbit path, as depicted in Fig. 4. Due to the quasi-stability of the 9:2 NRHO, departure slowly evolves over approximately 60 days until the spacecraft eventually leaves the vicinity of the orbit entirely. All times along the departure path are calculated from the time of the final SK maneuver.

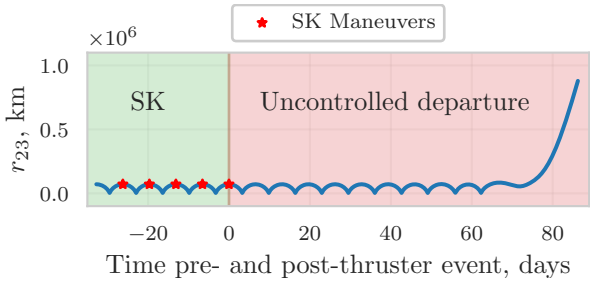


Fig. 4. Distance between the spacecraft and the Moon during both the SK and uncontrolled departure phases of the sample recovery simulation.

At various points throughout the departure segment, both XAC and NNIT are simulated to guide the spacecraft back to the 9:2 NRHO in the event that thruster functionality is suddenly restored. Leveraging XAC as a comparison metric for recovery is motivated by Zimovan-Spreen et al.'s study [13], and helps designate scenarios when recovery, rather than SK, is necessary. Several of the XAC recovery trends identified by Zimovan-Spreen et al. emerge in this study, though small differences arise in both XAC performance and in the reported departure times due to differences in dynamics modeling, assumed spacecraft propulsion, error sources, and XAC implementation. Zimovan-Spreen et al. employ a variant of XAC to compute impulsive maneuvers that accommodate phasing considerations by, at times, constraining future perilune passage times in an N-body ephemeris force model. In contrast, the implemented XAC algorithm for low-thrust, suggested in [6], targets only the downstream $\dot{x}$ value in the CR3BP rotating frame, and leverages the ideal rocket equation to convert impulsive SK maneuvers to finite-burn, low-thrust arcs. This transformation remains accurate for multiple hour burns, but begins to introduce significant error as thrust times grow. Furthermore, as also observed by by Zimovan-Spreen et al., a single converged XAC maneuver does not necessarily mean the orbit is 'stabilized'. Therefore, recovery costs for XAC are only reported if the algorithm continues to function and repeat the NRHO geometry for 20 subsequent periods of the orbit. As a SK algorithm, XAC is not expected to consistently function in a recovery process. Rather, XAC is simulated here to provide insight into recovery and conditions when it may be required.

Performance of the XAC (blue) and NNIT (orange) processes over time for the sample scenario, Fig. 4, are compared in Fig. 5, with vertical yellow lines denoting apolune times. Occasionally, for XAC, a large recovery cost is incurred at the second or third SK maneuver and, therefore, the reported recovery cost sums the equivalent ΔV of three sequential SK maneuvers. As expected, XAC initially offers lower cost maneuvers than NNIT. After 27 days, XAC still maintains the orbit, but is no longer dependably more propellant-efficient than NNIT. After 39 days, in this sample simulation, XAC no longer stabilizes the orbit. Conversely, during the initial departure segment, NNIT determines high recovery cost, indicating that it is inefficient to employ the NNIT method directly for SK (though it does maintain the orbit). However, as deviations grow, NNIT begins to perform comparably to XAC. Once outside the region of applicability for short-horizon SK, NNIT extends the recoverable window for approximately 3 orbital periods (about 21 days). This extended recovery period results from an autonomous process and offers low-thrust maneuver plans without the additional complexities of Zimovan-Spreen et al.'s approach [13]. As expected, recovery cost generally increases over time during this extended window. As visualized in Fig. 4, the spacecraft begins to significantly deviate from the NRHO path after 65 days, after which alternative recovery options must be considered.

Several key features of the NNIT recovery process are illustrated when examining maneuver plans at specific times. Recall that, in all presented cases, the NNIT ap-

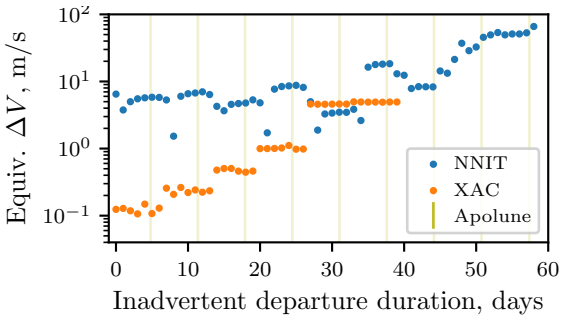


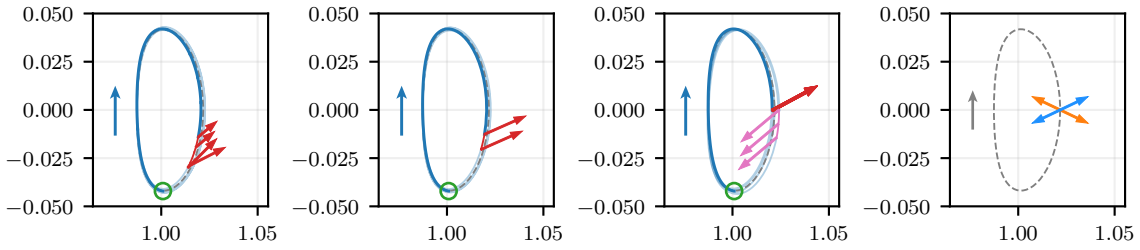
Fig. 5. Comparison of the cost to recover to the 9:2 NRHO reference motion using SK (orange), and NNIT (blue). Vertical yellow lines denote apolune times. The NNIT method extends the recoverable window by 21 days.

proach consists of 10 NN-identified low-thrust arcs that serve as an initial guess for the targeting algorithm define in Eqs. (9) and (10). Recovery paths originating at 29 days since the final stationkeeping maneuver, Fig. 4, are depicted in Fig. 6 to both highlight the NNIT maneuver planning process, and to illustrate a representative example of the region where NNIT and XAC perform comparably. In this case, at 29 days, NNIT suggests a thrust plan with $\Delta V_{\text{equiv.}} = 2.8$ m/s compared to $\Delta V_{\text{equiv.}} = 4.4$ m/s for XAC across two maneuvers. The NNIT and XAC maneuver plans (represented as x-y projections for clarity) are plotted in Figs. 6(a) and 6(c), respectively, where green circles signify the departing trajectory state at 29 days and the gray dashed lines correspond to the underlying 9:2 NRHO. Notably, the NNIT maneuver plan independently identifies apolune to be an effective location to place its recovery maneuver. While each of the 10 control choices for the NNIT process are implemented as low-thrust arcs, the first 6 maneuvers in this plan each possess a thrust magnitude less than 0.1% of maximum thrust, thus rendering the low-thrust effect on the system insignificant for the first 4.66 days. A challenge in previous NN-based investigations for low-thrust G&C is the assumption of constant thrust. However, a notable feature of the NN trained in this investigation is its consistent suggestions of coasting segments when the spacecraft is near perilune. This knowledge that apolune maneuver placement is effective along NRHOs is consistent with previous studies [5, 7]. The ability of the RL training process to independently uncover an effective known solution suggests further applicability of RL in problems where such a-priori knowledge is absent.

In the 29-day post-failure simulation for NNIT, Fig. 6(a), the 4.66-day coasting segment suggested by the NN is followed by four successive thrust arcs (red) ranging from 1 to 8 hours. While thrust direction and magnitude instantaneously vary between the four arcs, their combined effect is well-approximated by a single 8.6-hour fixed-direction thrust profile, depicted in Fig. 6(b). The thrust direction and time of this combined maneuver is determined via weighted averaging and scaling techniques suggested by LaFarge et al. [19], with the resulting discontinuity corrected using the same differential corrections technique employed in NNIT. Furthermore, tasked with recovering from the same deviation, XAC produces the bold red maneuver in Fig. 6(c). This single 28-minute finite thrust arc is not sufficient to fully recover to the NRHO and, upon propagating to the next apolune, a much larger 14.5 hour thrust segment (pink) is suggested. Furthermore, both the NNIT and XAC solutions produce thrust arcs that approximate the direction of the stable eigenvector associated with the 9:2 NRHO at apolune, plotted in Fig. 6(d) (XAC is within 2 degrees of the stable direction whereas the single NNIT arc differs by 30 degrees in the positive z direction). Implementing maneuvers in the direction of the stable eigenvector forms the basis of the powerful Floquet-mode SK strategy for unstable multi-body orbits, particularly in the Sun-Earth system [4]. This maneuver direction is known to be effective, and is independently uncovered by the RL training process through this investigation.

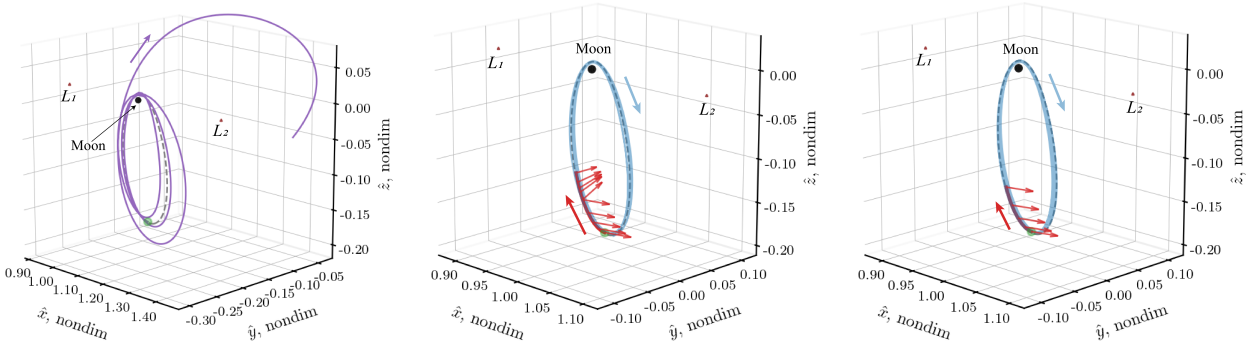
If spacecraft function is restored at 46.4 days into the inadvertent departure period, XAC fails to converge on a solution that stabilizes the orbit, and recovery is deemed necessary. The state at 46.4 days, plotted green in Fig. 7, rapidly departs the NRHO region over the subsequent month when left uncontrolled, as depicted in Fig. 7(a). To return to the NRHO, NNIT suggests the maneuver sequence in Fig. 7(b), consisting of: 1) an immediate 31.3-hour thrust arc at maximum throttle, 2) five smaller thrust arcs ranging from 1-3 hours, and 3) four near-ballistic coasting segments totaling 32 hours. A feasible trajectory emerges from the targeting process resulting in a total cost equal to $\Delta V_{\text{equiv.}} = 12.36$ m/s. Each of the thrust arcs are in similar directions and, therefore, their combined effect on the trajectory is well-approximated by the single fixed-attitude thrusting segment in Fig. 7(c).

Recovery paths originating at later points in time are depicted in Fig. 8. As the spacecraft departs from the desired NRHO motion, longer thrust segments are employed to achieve recovery targeting convergence. At 49 days, Fig. 8(a), the NNIT process still avoids thrusting near perilune with a 21-hour coast arc, followed by several successive thrust arcs that occur over a combined 5.19 days with a total cost equal to $\Delta V_{\text{equiv.}} = 35.2$ m/s. By 51 days, Fig. 8(b), the agent immediately thrusts for 4.6 days, and subsequently alternates between thrusting and coasting segments, with total $\Delta V_{\text{equiv.}} = 45.4$ m/s. Finally, by 55 days, Fig. 8(c), the deviation is sufficiently large that the NNIT maneuver path no longer avoids perilune, and instead immediately thrusts for 7.3 days continuously for a combined cost of $\Delta V_{\text{equiv.}} = 51.3$ m/s. At this error level, while the recovery targeting algorithm still achieves con-



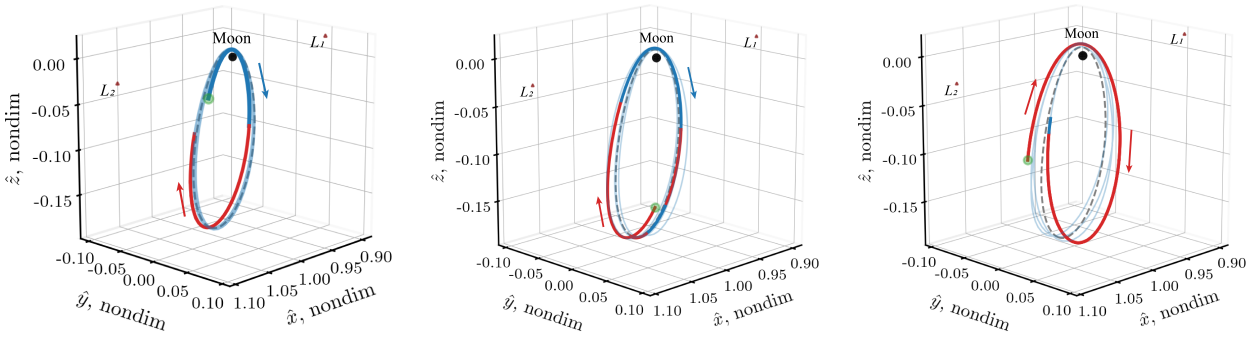
(a) Full NNIT recovery (b) NNIT single maneuver (c) Two XAC SK maneuvers (d) Eigenvector directions

Fig. 6. Recovery simulated at 29 days post-failure event (x-y projections). Green signifies the initial state, dashed gray denotes the NRHO, and light blue represents the ballistic trajectory following the final low-thrust maneuver. The NNIT recovery path (a) coasts until several successive thrust arcs in similar directions are implemented immediately following apolune. The NNIT arcs are then combined into a single representative thrust segment (b). Two XAC SK maneuvers are implemented in (c), where the first maneuver (red) does not adequately stabilize the orbit, necessitating a much larger second SK maneuver (pink). Thrust directions occur near the stable (blue) eigenvector directions in (d).



(a) Departing trajectory from 46.4 to 80 days (b) NNIT-identified recovery (c) Single thrust-arc recovery

Fig. 7. Recovery maneuver plan initiated at 46.4 days after thruster failure event with initial state denoted by a green circle. The uncontrolled path rapidly deviates from NRHO over a 33-day period (a). The NNIT maneuver plan (b) stabilizes the orbit and is well-approximated by a single fixed-direction thrusting segment (c).



(a) 49 days (b) 51 days (c) 55 days

Fig. 8. Recovery trajectories determined by NNIT at additional times after the thruster failure event. By 55 days (c), the targeting algorithm achieves converges, but the resulting post-recovery ballistic trajectory begins to diverge from the NRHO.

vergence, the resulting ballistic path initialized from the NRHO (plotted in light blue) begins to deviate from the desired NRHO motion. However, recall that the goal of recovery is not necessarily to return the spacecraft precisely to the NRHO, but instead achieve a maneuver plan that guides the spacecraft to a region where short-horizon SK may be resumed. In this scenario, both XAC and NNIT achieve convergence when simulated on the final state in Fig. 8(c). As deviations continue to grow further, convergence is eventually no longer achieved, and an alternative recovery approach or mission plan is deemed necessary. In these cases, longer-term low-thrust transfers that traverse further from the NRHO region must be considered, and alternate long-horizon baselines or intermediate orbit options may be necessary [13]; the feasibility of applying NNIT in such scenarios remains as future work.

5.1.1 *Higher-fidelity ephemeris recovery*

To initially explore the validity of NNIT recovery plans in a higher-fidelity force model, a single recovery path is considered. Recall, Fig. 7 depicts a NNIT maneuver plan in the CR3BP at 46.4 days post-thruster failure, with Fig. 7(b) depicting the full 42.8-day NNIT trajectory, and Fig. 7(c) representing a simplified control strategy that fixes the maneuver direction in the rotating reference frame for a 36.7-hour burn. These simulations do not consider additional perturbing forces and, hence, a higher-fidelity simulation provides insight into the practical applicability of the proposed methodology. Furthermore, while helpful for preliminary analysis, it may not be practical to fix the spacecraft attitude in a rotating frame and, therefore, an inertially fixed direction is employed.

To transform the CR3BP maneuver plan, both the initial state (green) and thrust direction (red) are rotated into the Moon-centered J2000 reference frame, assuming an initial epoch of Jan. 1, 2025. As visualized in the rotating frame in Fig. 9, the J2000-fixed-direction thrust arc (red) is propagated for 46.4 hours in the low-thrust augmented N-body ephemeris force model described in Eq. (8). A fixed inertial orientation implies subtle direction changes in a rotating frame, as apparent in Fig. 9. The resulting final state is then propagated ballistically in the ephemeris model for an additional 26.2 days (blue), or four periods of the 9:2 NRHO. In this sample scenario, the 9.6-hour increase in thrusting time from the CR3BP initial guess results in slightly smaller x excursions during the final coasting segments, though both simulations strongly resemble the target NRHO. As demonstrated in Fig. 9, this ephemeris simulation achieves close adherence to the desired NRHO motion and suggests practical applicability for the proposed NNIT methodology.

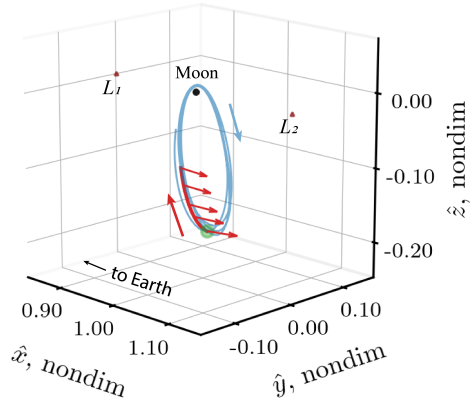


Fig. 9. Recovery path from NNIT at 46.4 days, transformed to a higher-fidelity simulation in an ephemeris force model (Jan. 1, 2025) with a J2000-fixed thrust direction. The corresponding CR3BP recovery path is visualized in Fig. 7(b).

5.2 *Monte-Carlo Results*

To evaluate NNIT efficacy across a wider range of simulations, a Monte Carlo analysis approach is employed. As in the previous recovery scenario, trials originate with a simulated SK phase. After five low-thrust XAC maneuvers, a thruster failure is introduced at apolune, and an inadvertent departure commences. Each trial employs NNIT to return the spacecraft to the NRHO, and data is gathered at various times across each departure, where time is measured from the final SK maneuver. The convergence rate for the NNIT targeting algorithm across the 5,000 trials is depicted in Fig. 10. The algorithm convergences in more than 99% of trials for 46 days following the simulated failure, and in more than 98% of trials up to 58 days. The convergence rate falls quickly over the proceeding 32 days, resulting in only 20.1% after 80 days, and 6.4% after 90 days, indicating that, by this point, an alternative algorithm or mission objective is necessary.

Propellant consumption statistics across the Monte Carlo trials are visualized in Fig. 11, where box plots are employed to represent interquartile ranges. As expected, propellant usage, measured by $\Delta V_{\text{equiv.}}$, increases substantially over the 90 days, beginning in the 4-7 m/s range for

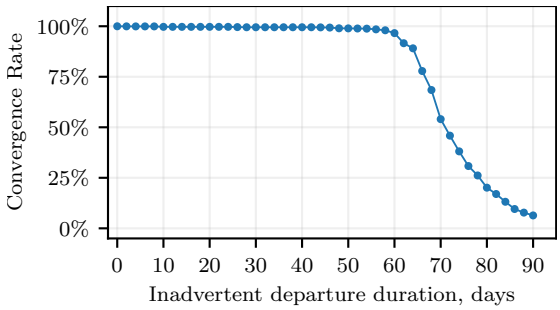


Fig. 10. Convergence rates across Monte Carlo trials

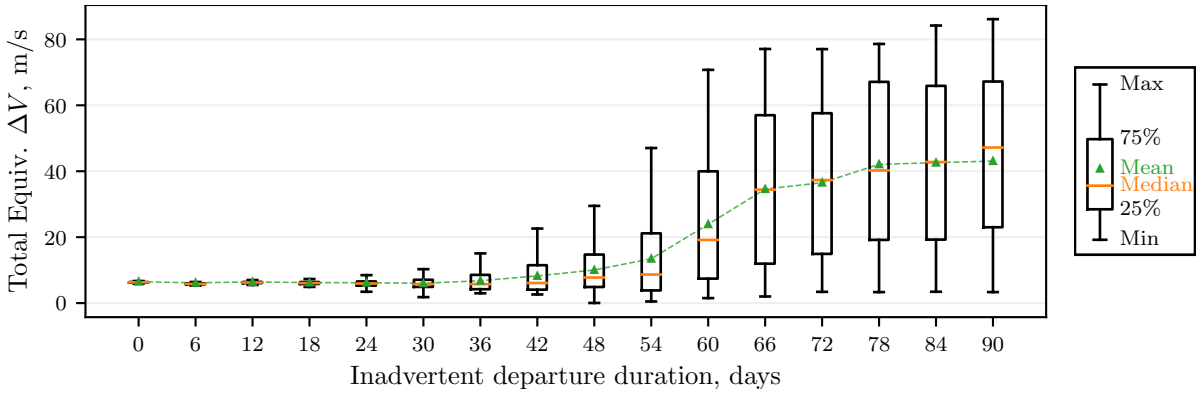


Fig. 11. Propellant consumption statistics across 5,000 Monte Carlo trials

the initial four weeks, increasing to 24 m/s by 60 days, and eventually reaching an average greater than 40 m/s for each time duration after 74 days. Furthermore, Fig. 11 demonstrates that the spread across trials increases precipitously toward the end of the inadvertent departure timeframe. This increase is primarily due to variations in the initial error and is similarly observed by Zimovan-Spreen et al. [13].

6. Concluding Remarks

In merging targeting and neural network control paradigms, the proposed NNIT method offers a realistic G&C architecture with demonstrated functionality in a low-thrust orbit recovery scenario. Neural network behavior is often considered unpredictable, and this opaqueness introduces error and risk into potential future NN-based G&C concepts. In contrast, traditional iterative methods produce exact solutions, but are computational expensive and require an accurate initial guess. The proposed NNIT approach mitigates risk incurred by the NN by employing a differential corrections process to ensure convergence and, conversely, the NN speeds up the targeter by rapidly offering accurate startup solutions, thus simultaneously expanding the solution space, and reducing the impact of chaotic dynamics on the RL training process. The proposed NNIT architecture offers promising results in an inadvertent departure scenario from a planned NRHO trajectory, resulting in a multi-week extension of the recovery window compared to a conventional crossing-control stationkeeping method. Onboard maneuver planning will enable future spacecraft to better react to unexpected events in challenging regions of space, and RL-trained neural networks provide an exciting potential component in facilitating this autonomy.

Acknowledgements

The authors thank the Purdue University School of Aeronautics and Astronautics and NASA Goddard Space Flight Center for facilities and support in conducting this research. This work was supported by a NASA Space Technology Research Fellowship, NASA Grant 80NSSC19K1175.

References

[1] Shoemaker, M. A., Hur-Diaz, S., Liounis, A. J., Van Eepoel, J. M., Romeo, M. J., Wu, V. C., Dangelo, S. D., Hatten, N., Schlenker, L. G., Hughes, S. P., Price, S. R., , and Winternitz, L. B., "Terrain Relative Navigation in a Lunar Landing Scenario Using autoNGC," *AIAA Scitech Forum*, AIAA, San Diego, CA, 2022. URL https://ntrs.nasa.gov/citations/20210024481.

[2] Marchand, B. G., Weeks, M. W., Smith, C. W., and Scarritt, S., "Onboard Autonomous Targeting for the Trans-Earth Phase of Orion," *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 3, 2010, pp. 943–956. https://doi.org/10.2514/1.42384.

[3] Dunham, D., and Roberts, C., "Stationkeeping Techniques for Libration-Point Satellites," *The Journal of the Astronautical Sciences*, Vol. 49, No. 1, 2001, pp. 127–144. https://doi.org/10.1007/bf03546340.

[4] Folta, D. C., Pavlak, T. A., Haapala, A. F., Howell, K. C., and Woodard, M. A., "Earth–Moon libration point orbit stationkeeping: Theory, modeling, and operations," *Acta Astronautica*, Vol. 94, No. 1, 2014, pp. 421–433. https://doi.org/https://doi.org/10.1016/j.actaastro.2013.01.022.

[5] Davis, D., Bhatt, S., Howell, K., Jang, J.-W., Whitley, R., Clark, F., Guzzetti, D., Zimovan, E., and Barton, G., "Orbit Maintenance and Navigation of Human Spacecraft at Cislunar Near Rectilinear Halo Orbits," *27th AAS/AIAA Space Flight Mechanics Meeting*, American Astronautical Society, San Antonio, Texas, 2017.

[6] Newman, C. P., Davis, D. C., Whitley, R. J., Guinn, J. R., and Ryne, M. S., "Stationkeeping, Orbit Determination, and Attitude Control for Spacecraft in Near Rectilinear Halo Orbits," *AAS/AIAA Astrodynamics Specialist Conference*, American Astronautical Society, Snowbird, Utah, 2018, pp. 1–20.

[7] Guzzetti, D., Zimovan, E. M., Howell, K. C., and Davis, D. C., "Stationkeeping Analysis for Spacecraft in Lunar Near Rectilinear Halo Orbits," *27th AAS/AIAA Space Flight Mechanics Meeting*, American Astronautical Society, San Antonio, Texas, 2017.

[8] Bonasera, S., Elliott, I., Sullivan, C. J., Bosanac, N., Ahmed, N., and McMahon, J., "Designing Impulsive Station-Keeping Maneuvers Near a Sun-Earth L2 Halo Orbit via Reinforcement Learning," *31st AAS/AIAA Spaceflight Mechanics Meeting*, AAS/AIAA, Charlotte, North Carolina (Virtual), 2021.

[9] LaFarge, N. B., Howell, K. C., and Folta, D. C., "An Autonomous Stationkeeping Strategy for Multi-Body Orbits Leveraging Rein-

forcement Learning," *AIAA Scitech Forum*, AIAA, San Diego, CA, 2022. https://doi.org/10.2514/6.2022-1764.

[10] Shirobokov, M., Trofimov, S., and Ovchinnikov, M., "Survey of Station-Keeping Techniques for Libration Point Orbits," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 5, 2017, pp. 1085–1105.

[11] Boudad, K. K., Howell, K. C., and Davis, D. C., "Departure and Escape Dynamics from the Near Rectilinear Halo Orbits in the Earth-Moon-Sun System," *The Journal of the Astronautical Sciences*, 2022. https://doi.org/10.1007/s40295-022-00328-w.

[12] Davis, D. C., Power, R. J., Howell, K. C., and Gutkowski, J. P., "Lunar Impact Probability for Spacecraft in Near Rectilinear Halo Orbits," *31st AAS/AIAA Spaceflight Mechanics Meeting*, AAS/AIAA, Charlotte, North Carolina (Virtual), 2021.

[13] Zimovan-Spreen, E. M., Davis, D. C., and Howell, K. C., "Recovery Trajectories for Inadvertent Departures from an NRHO," *31st AAS/AIAA Spaceflight Mechanics Meeting*, AAS/AIAA, Charlotte, North Carolina (Virtual), 2021.

[14] Izzo, D., Tailor, D., and Vasileiou, T., "On the Stability Analysis of Deep Neural Network Representations of an Optimal State Feedback," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 57, No. 1, 2021, pp. 145–154. https://doi.org/10.1109/TAES.2020.3010670.

[15] Li, H., Baoyin, H., and Topputo, F., "Neural Networks in Time-Optimal Low-Thrust Interplanetary Transfers," *IEEE Access*, Vol. 7, 2019, pp. 156413–156419. https://doi.org/10.1109/ACCESS.2019.2946657.

[16] Cheng, L., Wang, Z., Jiang, F., and Li, J., "Fast Generation of Optimal Asteroid Landing Trajectories Using Deep Neural Networks," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 56, No. 4, 2020, pp. 2642–2655. https://doi.org/10.1109/TAES.2019.2952700.

[17] Parrish, N. L. O., "Low Thrust Trajectory Optimization inCislunar and Translunar Space," Ph.D. dissertation, University of Colorado Boulder, 2018.

[18] LaFarge, N. B., Miller, D., Howell, K. C., and Linares, R., "Autonomous Closed-Loop Guidance using Reinforcement Learning in a Low-Thrust, Multi-Body Dynamical Environment," *Acta Astronautica*, Vol. 186, 2021. https://doi.org/https://doi.org/10.1016/j.actaastro.2021.05.014.

[19] LaFarge, N. B., Howell, K. C., and Linares, R., "A Hybrid Close-Loop Guidance Strategy for Low-Thrust Spacecraft Enabled by Neural Networks," *31st AAS/AIAA Spaceflight Mechanics Meeting*, AAS/AIAA, Charlotte, North Carolina (Virtual), 2021.

[20] LaFarge, N. B., Miller, D., Howell, K. C., and Linares, R., "Guidance for Closed-Loop Transfers using Reinforcement Learning with Application to Libration Point Orbits," *20th AIAA Scitech Forum*, AIAA, Orlando, Florida, 2020. https://doi.org/10.2514/6.2020-1910.

[21] Miller, D., and Linares, R., "Low-Thrust Optimal Control via Reinforcement Learning," *29th AAS/AIAA Space Flight Mechanics Meeting*, American Astronautical Society, Ka'anapali, Hawaii, 2019, pp. 1–18.

[22] Sullivan, C. J., Bosanac, N., Anderson, R. L., Mashiku, A. K., and Stuart, J. R., "Exploring Transfers between Earth-Moon Halo Orbits via Multi-Objective Reinforcement Learning," *2021 IEEE Aerospace Conference*, IEEE, 2021, pp. 1–13. https://doi.org/10.1109/AERO50100.2021.9438267.

[23] Guzzetti, D., "Reinforcement Learning And Topology Of Orbit Manifolds For Station-keeping Of Unstable Symmetric Periodic Orbits," *AAS/AIAA Astrodynamics Specialist Conference*, American Astronautical Society, Portland, Maine, 2019, pp. 1–20.

[24] Molnar, A. B., "Hybrid Station-Keeping Controller Design Leveraging Floquet Mode and Reinforcement Learning Approaches," Master's thesis, Purdue University, Dec. 2020.

[25] Das-Stuart, A., Howell, K. C., and Folta, D. C., "Rapid Trajectory Design in Complex Environments Enabled by Reinforcement Learning and Graph Search Strategies," *Acta Astronautica*, Vol. 171, 2020, pp. 172–195.

[26] Sullivan, C. J., Bosanac, N., Mashiku, A. K., and Anderson, R. L., "Multi-Objective Reinforcement Learning for Low-Thrust Transfer Design between Libration Point Orbits," *AAS/AIAA Astrodynamics Specialist Conference*, American Astronautical Society, Big Sky, Montana (Virtual), 2021.

[27] Federici, L., Scorsoglio, A., Zavoli, A., and Furfaro, R., "Autonomous Guidance for Cislunar Orbit Transfers via Reinforcement Learning," *AAS/AIAA Astrodynamics Specialist Conference*, American Astronautical Society, Big Sky, Montana (Virtual), 2021.

[28] Szebehely, V. G., *Theory of Orbits: The Restricted Problem of Three Bodies*, Academic Press, New York, 1967.

[29] Brophy, J. R., "Perspectives on the success of electric propulsion," *Journal of Electric Propulsion*, Vol. 1, No. 1, 2022.

[30] Cox, A., Howell, K., and Folta, D., "Dynamical structures in a low-thrust, multi-body model with applications to trajectory design," *Celestial Mechanics and Dynamical Astronomy*, Vol. 131, No. 3, 2019, pp. 1–34.

[31] Park, B., and Howell, K. C., "Leveraging Intermediate Dynamical Models for Transitioning From the Circular Restricted Three-Body Problem to an Ephemeris Model," *AAS/AIAA Astrodynamics Specialist Conference*, American Astronautical Society, Charlotte, North Carolina, 2022.

[32] Acton, C., Bachman, N., Semenov, B., and Wright, E., "A look towards the future in the handling of space science mission geometry," *Planetary and Space Science*, Vol. 150, 2018, pp. 9–12. https://doi.org/https://doi.org/10.1016/j.pss.2017.02.013.

[33] Clark, P. E., Malphrus, B., Brown, K., Reuter, D., Folta, R. M. D., Mandell, A., Hurford, T., Brambora, C., Patel, D., Banks, S., Farrell, W., Petro, N., Tsay, M., Hruby, V., Brandon, C., and Chapin, P., "Lunar Ice Cube Mission: Determining Lunar Water Dynamics with a First Generation Deep Space CubeSat," *47th Lunar and Planetary Science Conference*, The Woodlands, Texas, 2016.

[34] Elwood, A., Hunter, R., and Cheetham, B., "Cislunar Autonomous Positioning System Technology Operations and Navigation Experiment (CAPSTONE)," *2021 CubeSat Developers Workshop*, Virtual, 2021.

[35] Folta, D. C., Bosanac, N., Cox, A., and Howell, K. C., "The Lunar IceCube Mission Design: Construction of Feasible Transfer Trajectories with a Constrained Departure," *26th AAS/AIAA Space Flight Mechanics Meeting*, American Astronautical Society, Napa Valley, CA, 2016.

[36] Bosanac, N., Cox, A. D., Howell, K. C., and Folta, D. C., "Trajectory design for a cislunar CubeSat leveraging dynamical systems techniques: The Lunar IceCube mission," *Acta Astronautica*, Vol. 144, 2018, pp. 283–296.

[37] Park, B., Howell, K. C., and Folta, D. C., "Design of Low-Thrust Transfers from an NRHO to Low Lunar Orbits: Applications for Small Spacecraft," *AAS/AIAA Astrodynamics Specialist Conference*, American Astronautical Society, Big Sky, Montana (Virtual), 2021.

[38] Lee, D. E., "Gateway Destination Orbit Model: A Continuous 15 Year NRHO Reference Trajectory," White paper, NASA Johnson Space Center, Aug. 2019.

[39] Keller, H. B., *Numerical solution of two point boundary value problems*, Society for Industrial and Applied Mathematics, Philadelphia, Pa., 1976.

[40] Haapala, A. F., and Howell, K. C., "A Framework for Constructing Transfers Linking Periodic Libration Point Orbits in the Spatial Circular Restricted Three-Body Problem," *International Journal of Bifurcations and Chaos*, Vol. 26, No. 5, 2016.

[41] Fujimoto, S., van Hoof, H., and Meger, D., "Addressing Function Approximation Error in Actor-Critic Methods," *35th International Conference on Machine Learning (ICML)*, 2018. URL https://arxiv.org/pdf/1802.09477.pdf.

[42] Hastie, T., Tibshirani, R., and Friedman, J., *The Elements of Statistical Learning Data Mining, Inference, and Prediction*, 2nd ed.,

Springer Series in Statistics, Springer, 2009.

[43] Wilson, C., and Riccardi, A., "Enabling intelligent onboard guidance, navigation, and control using reinforcement learning on near-term flight hardware," *Acta Astronautica*, Vol. 199, 2022, pp. 374–385. https://doi.org/https://doi.org/10.1016/j.actaastro.2022.07.013, URL https://www.sciencedirect.com/science/article/pii/S0094576522003502.

[44] Sutton, R. S., and Barto, A. G., *Reinfrocement Learning: An Introduction*, 2nd ed., The MIT Press, 2018.

[45] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S., "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," *Proceedings of the 35th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 80, edited by J. Dy and A. Krause, PMLR, Stockholmsmässan, Stockholm Sweden, 2018, pp. 1861–1870. URL http://proceedings.mlr.press/v80/haarnoja18b.html.

[46] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O., "Proximal Policy Optimization Algorithms," *CoRR*, Vol. abs/1707.06347, 2017.

[47] Watkins, C. J. C. H., and Dayan, P., "Q-learning," *Machine learning*, Vol. 8, No. 3-4, 1992, pp. 279–292.

[48] 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, 2015.

[49] 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. URL https://arxiv.org/pdf/1509.02971.pdf.

[50] Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M., "Deterministic Policy Gradient Algorithms," *Proceedings of the 31st International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 32, edited by E. P. Xing and T. Jebara, PMLR, Bejing, China, 2014, pp. 387–395. URL https://proceedings.mlr.press/v32/silver14.html.

[51] Achiam, J., "Spinning Up in Deep Reinforcement Learning," 2018. URL https://spinningup.openai.com/.