Using Reinforcement Learning to Design a Low-Thrust Approach into a Periodic Orbit in a Multi-Body System

Christopher J. Sullivan* and Natasha Bosanac†
University of Colorado Boulder, Boulder, CO, 80303

Exploring multi-body systems with low-thrust-enabled small satellites opens up a new frontier of scientific opportunities. However, the limited control authority of these spacecraft introduces significant challenges during trajectory and maneuver design. In this paper, a deep reinforcement learning method is used to design a control profile for a low-thrust-enabled small satellite to approach a periodic orbit from a nearby initial condition and over a short time horizon. Multiple neural networks are trained to recover controlled paths that meet a variety of objectives while tracking to a reference trajectory on a stable manifold. Monte Carlo simulations for three distinct formulations of a reward function are performed using a control scheme derived from trained neural networks.

I. Introduction

With the recent success of the Mars Cube One (MarCO) mission and advancements in miniaturized technologies, SmallSats are no longer limited to operating in low Earth orbit (LEO). Rather, deep space exploration, technology demonstrations, and targeted science missions via low-thrust-enabled SmallSats may soon become a reality. In fact, upcoming missions such as Lunar IceCube, LunaH-map and NEA Scout will feature SmallSats deployed to a variety of locations within multi-body gravitational environments after riding as secondary payloads onboard Artemis 1 [1–3]. However, trajectory and maneuver design for spacecraft in chaotic multi-body systems is an inherently high-dimensional problem complicated moreso by incorporating the constraints associated with low-thrust-enabled SmallSats: limited propulsive capabilities, operational scheduling constraints, and fixed, yet uncertain, initial conditions. While several numerical approaches from optimal control and dynamical systems theory (DST) exist for constructing low-thrust trajectories and maneuver profiles in approximate dynamical models of multi-body systems, the development of autonomous and robust design strategies necessitate an alternative approach.

One class of algorithms that is of increasing interest in the astrodynamics community for enabling autonomy in trajectory and maneuver design is reinforcement learning (RL). RL algorithms typically involve an agent interacting with an environment by undertaking actions for dynamical states to maximize a reward function. The agent explores the environment until the policy that determines the optimal action at each state has been identified. When formulated appropriately, these algorithms may explore many state-action pairs to determine the optimal actions while limiting exploration of suboptimal actions. RL methods have been used in astrodynamics for trajectory and maneuver design in a variety of applications and dynamical models. For instance, Dachwald has explored designing a transfer to Mercury for a low-thrust-equipped spacecraft using artificial neural networks and an evolutionary algorithm [4]. A more recent approach by Das-Stuart, Howell, and Folta leverages RL along with fundamental dynamical structures to design complex transfer trajectories between periodic orbits in the Circular Restricted Three-Body Problem (CR3BP) [5]. In addition, Scorsoglio, Furfaro, Linares, and Massari also use an actor-critic, deep reinforcement learning (DRL) approach to develop docking maneuvers for spacecraft in Near Rectilinear Orbits in cislunar space [6]. More recently, Miller and Linares apply the well-known Proximal Policy Optimization (PPO) algorithm to design transfers between distant retrograde orbits in the Earth-Moon system, modeled via the CR3BP [7]. The success of each of these studies has created a valuable foundation for the astrodynamics community to continue exploring and extending the use of RL in multi-body trajectory design strategies. Specifically, this paper builds upon these previous works by focusing on one important component of implementing an RL-based approach to trajectory design: formulation of a reward function that reflects both the design objectives and the constraints that influence the operational feasibility of the recovered maneuver profile. This analysis is performed in the context of trajectory design for a low-thrust-enabled SmallSat to quickly access a nearby reference trajectory that lies on a stable manifold associated with a periodic orbit in the CR3BP.

*Ph.D. Student, Smead Department of Aerospace Engineering Sciences, University of Colorado Boulder, Boulder CO, 80303.

†Assistant Professor, Colorado Center for Astrodynamics Research, Smead Department of Aerospace Engineering Sciences, University of Colorado Boulder, Boulder CO, 80309.

In this preliminary analysis, DRL is used to design a control profile for a SmallSat to approach a periodic orbit within a multi-body system modeled via the CR3BP. The advantage of using DRL and neural networks is that, once fully trained, they may be used to rapidly produce locally optimal results for any state within the training environment since evaluating a trained neural network takes a negligible amount of time. In the CR3BP, invariant manifold structures naturally guide motion towards and away from periodic and quasi-periodic orbits within the system. Manifolds emanating from a target orbit tend to enable the design of trajectories with low propellant requirements. In this investigation, PPO is used to train sets of neural networks corresponding to three distinct reward functions to guide a low-thrust-enabled SmallSat towards a nearby trajectory that lies on the stable manifold associated with a periodic orbit over a short time horizon. The first reward function is designed to reward the spacecraft for possessing a state that is close to a state that lies along the reference trajectory; this reference state is defined using an isochronous correspondence. A second reward function is formulated to reward matching a reference state that is calculated via a normal correspondence, i.e., by locating the nearest state along the reference trajectory. The third reward function then incorporates an additional term to minimize the change in the thrust vector over each time step, reflecting a significant operational constraint. Once the neural networks are trained for each reward function, the control schemes are evaluated on a common set of perturbed initial conditions to examine the properties of the recovered solutions and associated maneuver profiles.

II. Dynamical Environment

For trajectory and maneuver design in multi-body gravitational environments, dynamical models of varying levels of complexity are often employed. In this analysis, the CR3BP is used to model the motion of a SmallSat of negligible mass in cislunar space. The CR3BP admits a variety of fundamental dynamical structures that are approximately retained in higher-fidelity models. Thus, the CR3BP serves as a useful environment for constructing an initial guess for a trajectory and maneuver profile. The equations of motion for the natural CR3BP are then augmented to include the acceleration due to the low-thrust engine of the SmallSat. Thus, the dynamical environment for a PPO implementation is instantiated using a low-thrust-enabled CR3BP.

A. Circular Restricted Three-Body Problem

The CR3BP is an autonomous dynamical model that approximates the higher-fidelity gravitational environment in multi-body systems. This model reflects the dynamics governing a spacecraft of negligible mass under the gravitational influence of two larger primary bodies, the Earth and Moon, assumed to follow circular orbits about their mutual barycenter. Under these assumptions, a rotating frame, $(\hat{x}\hat{y}\hat{z})$, is constructed and displayed in Fig. 1 with the $\hat{x}$ axis directed from the Earth to the Moon, the $\hat{z}$ axis is aligned with the angular momentum vectors of the primaries, and $\hat{y}$ completes the right-handed coordinate frame. Then, length, time and mass quantities are nondimensionalized using the following characteristic quantities: l^* is set equal to the distance between the two primaries, t^* is defined to set the mean motion of the primary system to unity, and the sum of the masses of the primaries is $m^* = M_1 + M_2$. Finally, in the Earth-Moon system, the mass ratio is defined as $\mu = \frac{M_2}{m^*} \approx 0.01215$. This quantity significantly influences the underlying dynamical structures governing the solution space in the CR3BP. Although nondimensionalization is

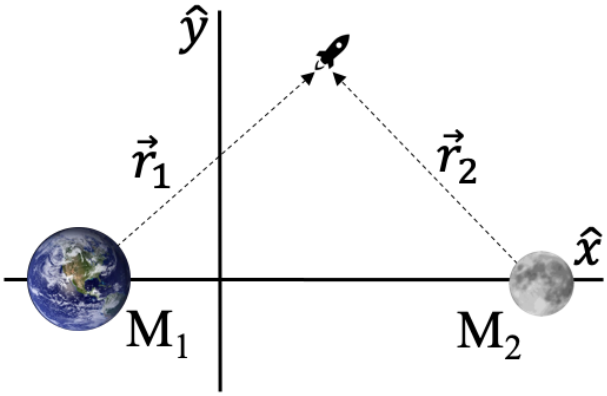


Fig. 1 Definition of the rotating frame $(\hat{x}\hat{y}\hat{z})$ in the Earth-Moon CR3BP.

typically used to reduce the potential for ill-conditioning during numerical integration and to enable straightforward comparison between systems with a similar mass ratio, it also offers an additional benefit in an RL-based approach: reducing the potential for ill-conditioning during neural network evaluations. The nondimensional state of the spacecraft in the CR3BP is then defined as $\bar{x} = [x, y, z, \dot{x}, \dot{y}, \dot{z}]$ relative to the system barycenter and in the rotating frame. Then, the equations of motion that govern the natural dynamics of the spacecraft in the CR3BP are written as:

$$\ddot{x} - 2\dot{y} = \frac{\partial U^*}{\partial x} \qquad \ddot{y} + 2\dot{x} = \frac{\partial U^*}{\partial y} \qquad \ddot{z} = \frac{\partial U^*}{\partial z} \tag{1}$$

where the pseudo-potential function $U^* = \frac{1}{2}(x^2 + y^2) + \frac{1-\mu}{r_1} + \frac{\mu}{r_2}$ and the distance of the spacecraft to each primary is $r_1 = \sqrt{(x+\mu)^2 + y^2 + z^2}$ and $r_2 = \sqrt{(x-1+\mu)^2 + y^2 + z^2}$ [8]. Using these equations of motion, an energy-type quantity denoted the Jacobi Constant is equal to $C_J = 2U^* - \dot{x}^2 - \dot{y}^2 - \dot{z}^2$ [8]. In the CR3BP, the value of the Jacobi constant is constant along a single natural trajectory. This parameter also provides insight into the relative difference in energy between two states of the system. For instance, if two states in the CR3BP possess distinct Jacobi constant values, then energy must be added or removed via a propulsion system for a spacecraft to transfer between the states.

Fundamental dynamical structures that exist within the CR3BP include equilibrium points, periodic orbits, quasi-periodic orbits, and invariant manifolds [9]. A multi-body system described by the CR3BP admits five equilibrium points: the collinear equilibrium points, L_1, L_2, and L_3, and the triangular points, L_4 and L_5. Due to the significant interest in placing the Lunar Orbital Platform-Gateway (LOP-G) into an L_2 Near Rectilinear Halo Orbit (NRHO), this investigation will construct solutions in the L_2 region of the Earth-Moon system [10]. A variety of orbit families emanate from the L_2 equilibrium point, including the planar L_2 Lyapunov family; Fig. 2 depicts a subset of this family in the xy-plane of the rotating frame and colored by the Jacobi constant of each orbit. Specifically, an L_2 Lyapunov orbit with a Jacobi constant of 3.17 is selected as the target orbit for the trajectory design scenario in this paper due to its proximity to the L_2 equilibrium point, a potentially valuable region for future scientific research and infrastructure placement. This member of the L_2 Lyapunov family admits stable and unstable invariant manifolds, capturing trajectories that, respectively, asymptotically approach and depart an unstable orbit. These structures are often used in constructing initial guesses for low-cost transfers within multi-body gravitational environments [11, 12]. For this reason, a trajectory that asymptotically approaches the selected L_2 Lyapunov orbit from the vicinity of the Moon is used to define a reference trajectory for the scenario examined in this paper. Figure 3 displays a segment of the stable manifold (blue) associated with the selected L_2 Lyapunov orbit (black). The red arc corresponds to the specific reference trajectory used in this analysis, which is sampled at and uniquely described by the following state: $\bar{x} = [1.120, -0.0442, 0, 0.0614, 0.0274, 0]$. This state is considered the initial condition along the reference trajectory. A spacecraft possessing this exact state at any time will asymptotically approach the selected L_2 Lyapunov orbit in the CR3BP. However, to reach this state – or another state along the associated reference trajectory – in the CR3BP from a nearby state, a controlled transfer is necessary.

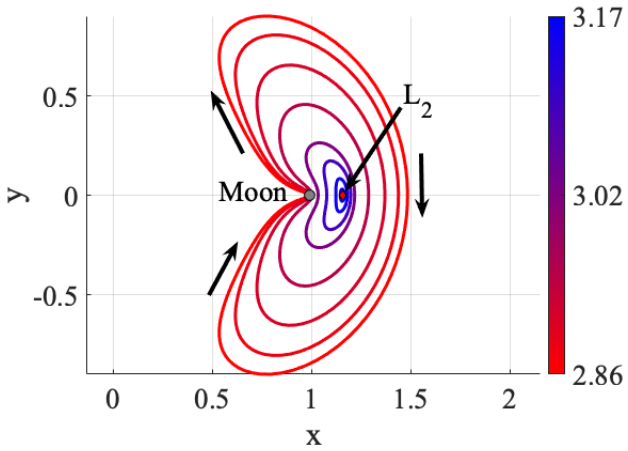


Fig. 2 Members of the L_2 Lyapunov periodic orbit family in the Earth-Moon CR3BP colored by the value of the Jacobi constant.

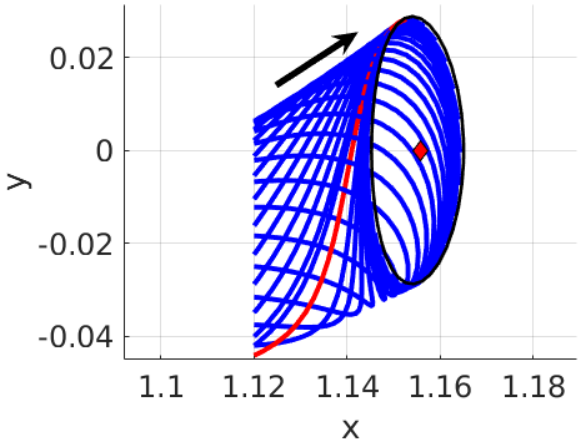


Fig. 3 A segment of the stable manifold (blue) associated with an L_2 Lyapunov periodic orbit (black). The specific reference trajectory used in this paper is displayed in red.

B. Low-Thrust-Enabled CR3BP

When the propulsion system is activated, both the mass of the SmallSat and the acceleration from the low-thrust engine must be incorporated into the dynamical model. The nondimensional acceleration imparted by an engine with a constant thrust and constant specific impulse, I_{sp}, is written as:

$$\bar{a} = \frac{T}{m_{s/c}}\hat{u} = a_x\hat{x} + a_y\hat{y} + a_z\hat{z} \tag{2}$$

where $m_{s/c}$ denotes the current spacecraft mass normalized by the initial wet mass, T is the thrust normalized by length and time characteristic quantities and the spacecraft initial wet mass, and $\hat{u}$ is a unit vector describing the thrust direction in the Earth-Moon rotating frame. Using these definitions, the equations of motion for a low-thrust-enabled spacecraft in the CR3BP are written as:

$$\ddot{x} - 2\dot{y} = \frac{\partial U^*}{\partial x} + a_x \qquad \ddot{y} + 2\dot{x} = \frac{\partial U^*}{\partial y} + a_y \qquad \ddot{z} = \frac{\partial U^*}{\partial z} + a_z. \tag{3}$$

Then, the state vector is augmented to include the nondimensionalized spacecraft mass, i.e., $\bar{x}_{LT} = [x, y, z, \dot{x}, \dot{y}, \dot{z}, m_{s/c}]$. Finally, an additional first-order differential equation is formulated to model the decrement in the spacecraft mass due to propellant mass usage. The mass flow rate equation for a constant thrust, constant I_{sp} low-thrust engine is written as $\dot{m}_{s/c} = \frac{-T}{I_{sp}g_0}$, where g_0 is the gravitational acceleration as measured from the surface of the Earth [13]. The resulting first-order differential equations are supplied to the RL algorithm to define the dynamical environment.

III. Deep Reinforcement Learning

DRL algorithms have been of much interest for applications where an optimization problem cannot be solved analytically. Fundamental RL algorithms typically involve an agent interacting with an environment, defined via a known or unknown dynamical model taking actions that maximize the long-term reward at a given state [14]. This returned reward is mathematically encapsulated within a scalar value function that denotes the long-term reward of a particular state within the state space [15]. Additionally, the set of actions that are performed throughout the state space to maximize the value function is often identified as a policy. Developing a policy for a state-action space that maximizes the value function admits the optimal actions for any state within the state space.

While the overall structure of RL-type problems has generally remained consistent over time, numerous methods have been introduced to solve these problems [16]. Of particular interest in this analysis is the Actor-Critic method which incorporates two independent neural networks to develop an optimal policy within an environment. Actor-Critic methods have gained significant traction in a wide variety of disciplines due to their computational efficiency and ability to stochastically model the optimal action for a given state [17]. Recent advancements in computational power and the development of more efficient algorithms have resulted in robust algorithms that are capable of discovering policies

for a continuous action space in highly chaotic environments [18]. The introduction of deep neural networks into RL algorithms has revolutionized the field of RL and enabled DRL algorithms to be applied to entirely new applications.

A. Neural Networks

Neural networks are composed of layers of nodes structured similarly to neurons in a brain. For the first layer, also known as the input layer, these nodes take as input the states associated with the environment. In this analysis, these states include the position and velocity vectors in the Earth-Moon rotating frame, the mass of the spacecraft and time measured from an initial epoch. The next layer is termed a hidden layer and consists of a set of nodes, each defined using an nonlinear activation function such as the hyperbolic tangent, rectified linear unit (ReLU), or sigmoid [19]. Each of the input nodes is fully connected to each node in the first hidden layer, and each connection is assigned a weight that signifies the magnitude of the connected input node in computing the result of the hidden layer node. Then, if working with a deep neural network, the following layer is another hidden layer with nodes fully connected to the preceding hidden layer. Multiple hidden layers with a specified number of nodes may be incorporated into a deep neural network. The final layer of the network is denoted the output layer. The output layer reflects the network's current estimate of the optimal behavior, i.e. actions, weights, or another parameter that the network is tasked with learning. Figure 4 displays a conceptual representation of a deep neural network where the blue nodes represent the input layer; two hidden layers are depicted in gold; the output layer is red; connections between the nodes are denoted using black arrows; the weights for each layer are encapsulated within $\bar{W}_i$; and the nodes within each layer are numbered using superscripts for the layer number and subscripts to enumerate the nodes within one layer. Although a neural network may be formulated to use the dynamical states to recover optimal actions, configuration of the neural network is nontrivial. In fact, assigning the weights to each connection within a neural network is often an iterative process due to the complex and nonlinear relationships within a network.

While neural networks may be useful in solving problems where analytical mappings between states and actions do not exist, they must be trained to replicate the behavior they are created to emulate. To train a neural network, updates to the weights of the network must be formulated. These updates to the network are designed in an RL problem by using experiences derived from the network interacting with the environment and receiving rewards based on previously performed actions and states. The commonly used Adams Optimizer method is leveraged in this analysis due to its well established success in state-of-the-art DRL algorithms [20]. For a well-trained neural network, these weights will asymptotically converge to values that produce trajectories that correspond to solutions that maximize the long term reward. Additionally, the selected activation function and number of hidden nodes in each layer often influences the convergence behavior of the algorithm [19]. A larger number of hidden nodes per layer may potentially reflect the complexity of the solution space for the chaotic dynamics of a multi-body system, but at the expense of longer training times. Furthermore, the choice of activation function for a specific problem is nontrivial and still an active area of research [19]. In this investigation, the neural networks are structured with three hidden layers, 256 hidden nodes per hidden layer, and each node using an ReLU activation function. This structure is observed through a trial-and-error approach to produce favorable results within an Actor-Critic method and based on its success in other applications with complex dynamics.

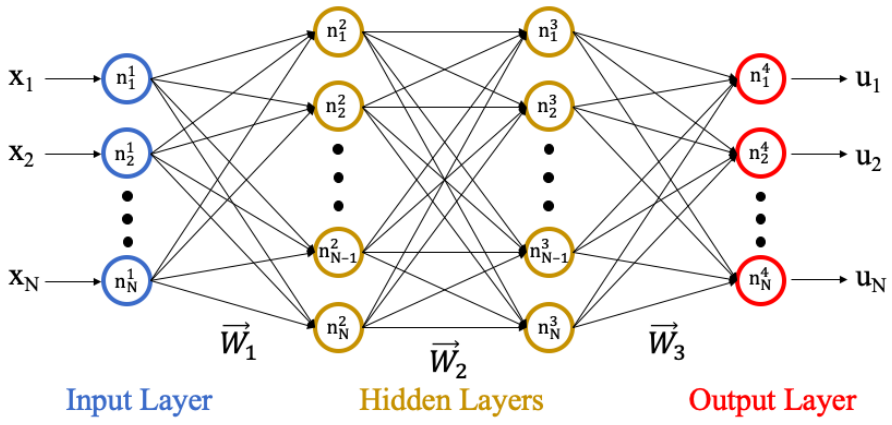


Fig. 4 Representation of a neural network with an input layer, two hidden layers, and an output layer.

B. Actor-Critic Methods

Actor-Critic methods form the foundation of many modern DRL algorithms due to their robust convergence properties and applicability to a wide variety of problems. These methods are characterized by their implementation of two neural networks: one for an actor and one for a critic [21]. An actor neural network produces what it believes to be an optimal action for a given state while the critic neural network evaluates the actor and updates the networks accordingly. These two neural networks are independent and both attempt to learn and optimize a distinct reward function within the environment. Specifically, the actor neural network is used to learn the policy function, $\pi_\theta(\bar{u}_t|\bar{x}_t)$, that produces the optimal action, $\bar{u}_t$, for a given state, $\bar{x}_t$, at time t [22]. Note, both $\bar{x}_t$ and $\bar{u}_t$ may be set to scalar values in an Actor-Critic method – however, in this analysis, they are each defined as vector quantities. Meanwhile, the critic neural network is used to learn the value function, $V_\theta^\pi(\bar{x}_t)$, defined as:

$$V_\theta^\pi(\bar{x}_t) = \sum_{t=0}^{T} \gamma^t r_t(\bar{x}(t), \bar{u}(t)) \tag{4}$$

where γ denotes the discount factor, T represents the maximum number of time steps for the agent along a trajectory within the environment, π is the current policy believed by the networks to optimize the value function, r_t is the reward at each time step t, and θ signifies the the current set of policy parameters [23]. More precisely, θ is formulated as a vector containing all of the weights within both the actor and critic neural networks. Equivalently, the policy may be generated directly from the actor neural network while the critic neural network ensures that the policy converges on a long-term, locally optimal solution. Figure 5 depicts a conceptual representation of an Actor-Critic method with the environment: the current state is denoted by $\bar{x}$, the reward of that state-action pair by r, the derivative of the state by $\dot{\bar{x}}$, the action $\bar{u}$, and the weights of each neural network by $\bar{W}_c$ and $\bar{W}_a$ formulated as vectors for the critic and actor neural networks respectively. The environment contains both the dynamics of the system and the reward function for the algorithm. While deep neural networks are contained within both the actor and critic blocks, updates to those networks may be developed via a variety of approaches including policy gradient, least squares, and projected gradient descent methods.

Policy gradient algorithms are commonly employed within state-of-the-art actor-critic methods because the bounds on the weights do not need to be predefined. Rather, the weights of the connections in the neural networks are iteratively updated based on the collected experiences from the agent interacting with the environment to maximize the following expected discounted total reward function:

$$J(\theta) = \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t r_t(\bar{x}(t), \bar{u}(t)) \right] \tag{5}$$

which is evaluated along a single trajectory [22]. To maximize the returned reward, the gradient of this function, labeled $\hat{g}$, is estimated as:

$$\hat{g} = \nabla_\theta J(\theta) = \nabla_\theta \, \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t r_t \right] = \hat{\mathbb{E}}_t [\nabla_\theta \log \pi_\theta(\bar{u}_t|\bar{x}_t) \hat{A}_t] \tag{6}$$

where circumflex symbols denote estimated values and $\hat{A}_t$ is termed the estimated advantage function at a given time step [24]. The advantage function is set equal to:

$$\hat{A}_t = r_t + \gamma V_\theta^\pi(\bar{x}_{t+1}) - V_\theta^\pi(\bar{x}_t) \tag{7}$$

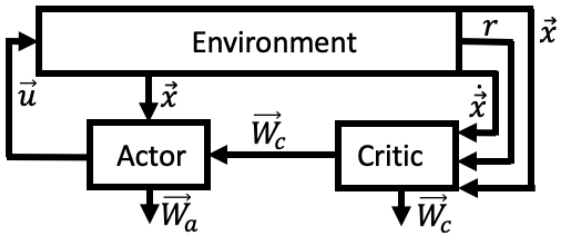


Fig. 5 Structure of an Actor-Critic method with the environment of the system, the actor and critic neural networks, and parameters that are passed between the components.

This estimated advantage function converges to the true advantage function as the value function is iteratively updated [23]. Using these definitions, a loss function is formulated via:

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t[\log \pi_\theta(\bar{u}_t|\bar{x}_t)\hat{A}_t] \tag{8}$$

and represents the loss function for the current policy and value functions [23]. The update to the neural networks and policy is calculated by formatting the update as an unconstrained optimization problem. However, the limitation of this approach is the potential for large updates that destabilize the system under poorly sampled state, action, and reward data. Schulman, Moritz, Levine, Jordan, and Abbeel have developed the trust region policy optimization (TRPO) algorithm which mitigates this issue by setting a hard constraint on the size of the update through the Kullback-Leibler (KL) divergence evaluation, a measure of the entropy or difference between two updates [23]. Using this algorithm, the optimization problem for developing a policy takes the following form:

$$\text{maximize}_\theta: \hat{\mathbb{E}}_t\left[\frac{\pi_{\theta,i}(\bar{u}_t|\bar{x}_t)\hat{A}_t}{\pi_{\theta,i-1}(\bar{u}_t|\bar{x}_t)}\right] \tag{9}$$

$$\text{subject to: } \hat{\mathbb{E}}_t[KL[\pi_{\theta,i-1}(\cdot|\bar{x}_t), \pi_{\theta,i}(\cdot|\bar{x}_t)]] \leq \delta \tag{10}$$

where θ_i denotes the policy parameters at update i, KL indicates the KL divergence function evaluation, and δ is a bound on the KL divergences between the policies [23, 24]. Additionally, the ratio, R_t, between the old and new policies is defined as:

$$R_t(\theta) = \frac{\pi_{\theta,i}(\bar{u}_t|\bar{x}_t)}{\pi_{\theta,i-1}(\bar{u}_t|\bar{x}_t)} \tag{11}$$

When the policy converges on an optimal solution, this ratio approaches 1. However, determining a correct value for δ is nontrivial as new environments and changing the parameters of the implementation may necessitate drastically different values [24]. Subsequent Actor-Critic, policy gradient algorithms have been developed that improved on these limitations, but generally follow the structure defined above.

C. Proximal Policy Optimization

PPO was originally developed to overcome the limitations of TRPO, eliminating the necessity of predefining a trust region. Rather, the objective function is now clipped through a clipping parameter, ε, defined using the ratio, $R_t(\theta)$; any update to the policy $\pi_{\theta,i}(\bar{u}_t|\bar{x}_t)$ that causes the ratio to diverge away from 1 is penalized [24]. Using this information, a new objective function is written as:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[\min(R_t(\theta)\hat{A}_t, \text{clip}(R_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t)] \tag{12}$$

This loss function ignores any change in the policy ratio if it improves the policy parameters, but considers the policy ratio when the update produces a decrease in the objective function [24]. A squared-error loss term and an entropy term are both added to the above objective function and, together, encourages continued exploration of the environment while converging towards a policy. The objective function that is used within a PPO algorithm is then written as:

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](\bar{x}_t)] \tag{13}$$

where c_1 and c_2 are the value function and entropy scalar coefficients, the superscript VF refers to the squared-error loss term, the superscript S represents the entropy term within the objective function, and S is the entropy bonus which naturally degrades as the algorithm experiences more updates [25]. The squared-error loss term in this expression is defined as:

$$L_t^{VF}(\theta) = (V_\theta(\bar{x}_t) - V_t^{target})^2 \tag{14}$$

The value of the clipping parameter originally proposed by the authors, $\varepsilon = 0.2$, tends to perform well on a wide variety of problems and is, therefore, used in this analysis [24]. Applying PPO within a dynamical environment enables the development of a policy via trained neural networks to maximize a reward function; Miller and Linares have demonstrated that this algorithm can be successfully applied to a trajectory design scenario with the CR3BP used to define the dynamical environment [7].

IV. Implementing PPO in the CR3BP

PPO is used to design a trajectory and maneuver profile for a low-thrust-enabled SmallSat to reach a nearby reference trajectory on the stable manifold associated with an L_2 Lyapunov orbit in the Earth-Moon CR3BP. First, the low-thrust-enabled CR3BP is employed as the dynamical environment within the PPO algorithm. Specifically, a SmallSat equipped with one Busek BHT-8000 Hall Effect thruster set in the maximum thrust mode with $T = 0.449N$ and a specific impulse of $I_{sp} = 2210s$ is assumed [26]. In addition, an ESPA-class SmallSat with an initial wet mass of 180 kg is assumed in this analysis, consistent with current SmallSat technological capabilities [27]. Then, the PPO algorithm is structured such that when the neural networks receive an observed state, an action is selected and the reward is determined by analyzing the effect of the action on the state. In this analysis, the goal captured by each reward function is to encourage the SmallSat to approach a specified trajectory by minimizing the position and velocity difference between the current state and a reference state – and, in one reward function formulation, also minimizing the change in the thrust vector direction. The action output from the neural networks, $\bar{u}(t)$, is a 3×1 unit vector that specifies the thrust direction in the rotating frame; although each component is originally selected from a continuous space between $[-1, 1]$, this vector is subsequently normalized to possess a unity magnitude. Formulating the action as a unit vector reflects a constant thrust applied at each time step. Furthermore, throughout each time step, Δt, the thrust direction of the SmallSat is held constant. By defining a constant Δt term to propagate forward the state of a SmallSat in the low-thrust-enabled CR3BP, the subsequent state and resulting reward is distinctly determined and returned to the PPO algorithm. In selecting Δt, a balance is required to enable the neural networks to identify the effect of an action on a state without preventing the algorithm from discovering feasible trajectories. Thus, a value of $\Delta t = 0.001$ is selected in nondimensional units to define for each time step of the PPO algorithm, corresponding to approximately six minutes. The algorithm is also constrained to identify a solution for the low-thrust-enabled SmallSat to reach the reference trajectory from a specified initial condition within 50 time steps.

To evaluate a reward function that reflects the deviation between the current spacecraft state and the reference trajectory, two definitions of a correspondence are employed: an isochronous and a normal correspondence [8]. An isochronous correspondence between two trajectories is calculated as the difference between two states occurring at the same time as measured from a common initial epoch. Thus, in the scenario used in this analysis, the spacecraft state is compared at each time step to a state that is simultaneously integrated along the reference trajectory from the specified initial condition. Alternatively, for a normal correspondence, the reward function is calculated using the difference between the current spacecraft state and the closest point along the reference trajectory. In this analysis, this location is calculated as the closest point in configuration space from a set of 500 states discretized along the reference trajectory. When evaluating the reward function using an isochronous correspondence, the spacecraft is tracking towards a moving target – a common scenario when trajectory and maneuver design strategies must incorporate timing or phasing constraints. However, when using a normal correspondence, the goal is simply to reach any location along the reference trajectory. This formulation for comparing the spacecraft state to a reference trajectory may be applicable to scenarios where the goal is simply to access a specific dynamical structure rather than a single state at a specific epoch.

The initial conditions for generating low-thrust-enabled trajectories via the neural networks are defined using a Gaussian distribution in the x, y, z positions relative to a specified state along the reference trajectory. Specifically, the Gaussian distribution is defined using a standard deviation of 38.4 km. Although this value represents only a small deviation from the reference trajectory, the vicinity of the L_2 equilibrium point in the Earth-Moon CR3BP is sensitive. In fact, trajectories generated from initial conditions defined using this small deviation from the reference trajectory are depicted in Fig. 6 via a projection onto the (x, y)-plane. The reference trajectory lies along the stable manifold associated with an L_2 Lyapunov orbit – a spacecraft on this reference would asymptotically approach the periodic orbit. However, Fig. 6 reveals that the perturbed solutions either return to the lunar vicinity or pass through the L_2 gateway. During the training phase of each neural network, initial conditions are randomly drawn from the Gaussian distribution, producing a new set at each iteration. However, once the neural networks are trained using a specific reward functions, the trained neural networks are all evaluated using the same set of 100 perturbed initial conditions to facilitate a clear comparison of the resulting trajectories. Training the networks on random initial conditions and then testing on the evaluation set of initial conditions may prevent overfitting to the data.

Structuring a neural network prior to training is specific to the environment and reward function. Typically, larger neural networks can better approximate highly nonlinear functions – but at the expense of training time. In this application, the neural network used in the PPO algorithms for the actor and critic are each structured with three hidden layers consisting of 256 nodes in each hidden layer. The state input to the neural networks includes the position and velocity vectors in the Earth-Moon rotating frame, spacecraft mass, and time along the trajectory. Additionally, for the implementation of the PPO algorithm in this investigation, the associated hyperparameters are selected according

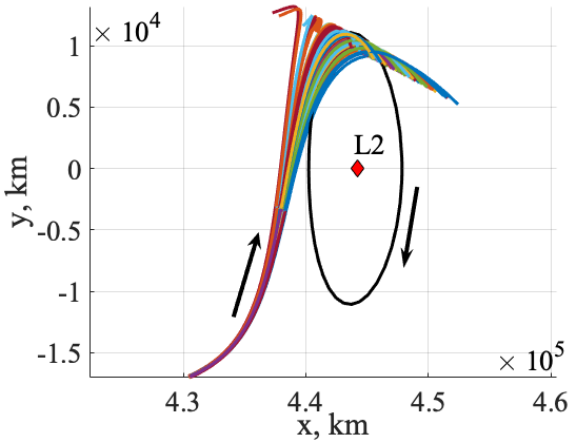


Fig. 6 Trajectories associated with initial conditions generated via perturbations from a reference trajectory along the stable manifold associated with an L_2 Lyapunov orbit.

to trial-and-error observations of the convergence behavior of the algorithm; Table 1 summarizes the values used in this investigation. These hyperparameters are used within the PPO2 implementation accessed via the Stable Baselines open-source library, built using Google's open-source TensorFlow library [28, 29]. Using these parameters within the PPO algorithm, neural networks are trained on the low-thrust-enabled CR3BP environment to develop an optimal policy for tracking to a reference trajectory over a short time horizon following a small perturbation.

The neural networks used in this analysis are trained using three specific formulations of a reward function. Fundamentally, a general scalar reward function is written as a sum of terms reflecting the difference between the current spacecraft state and the reference trajectory in position and velocity as well as the change in the thrust unit vector. Then, each of the three examined reward functions is formulated by varying the scalar coefficients of these terms. The general reward function, r_t, is written as a function of the state and action pair as:

$$r_t(\bar{x}_t, \bar{u}_t) = -c_d|\bar{d}_{ref}(t+\Delta t) - \bar{d}_{s/c}(t+\Delta t)| - c_v|\bar{v}_{ref}(t+\Delta t) - \bar{v}_{s/c}(t+\Delta t)| - c_\theta\theta \tag{15}$$

where $\bar{d}(t)$ represents the position vectors at a given time step for either the reference trajectory or the current spacecraft state relative to the system barycenter and in the rotating frame, the velocity vectors are $\bar{v}(t)$ for either the reference or the current spacecraft state; the subscript 'ref' corresponds to a quantity evaluated along the reference trajectory while the 's/c' subscript corresponds to the current spacecraft state. Recall that the position and velocity difference terms of the reward function may be evaluated using either an isochronous or normal correspondence. Then, θ is measured as the angle between the current action unit vector and the previous action unit vector calculated as:

$$\theta = cos^{-1}\left(\frac{\hat{u}(t)\cdot\hat{u}(t-\Delta t)}{|\hat{u}(t)|\cdot|\hat{u}(t-\Delta t)|}\right)$$

Then, c_d, c_v, and c_θ denote coefficients for the position, velocity, and control direction terms, respectively. The negative

Table 1 Parameters for the PPO implementation

Parameter	Value
Learning rate, α	2.5E-4
Discount factor, γ	0.99
Coefficient of squared-error loss, c_1	0.5
Coefficient of entropy term, c_2	0.01
Clipping parameter, ε	0.2
Time steps per update	65,536
Total time steps	315,000,000

signs scaling each term reflect a penalty on large state deviations and large changes in thrust direction between each step. By altering the coefficients of each term, the neural networks weight the importance of each of the three objectives to produce a distinctly different reward function.

V. Results

Neural networks are trained using three distinct reward functions to generate three sets of trajectories and maneuver profiles. Once trained, the neural networks are tested on the same set of initial conditions corresponding to perturbations in x, y, z from a state along the reference trajectory. The computed set of controlled trajectories and maneuver profiles is examined to gain further insight into the influence of the reward function on the results of the PPO algorithm.

A. Tracking to a Reference Trajectory via an Isochronous Correspondence

The first reward function formulation reflects the goal of tracking a reference trajectory via an isochronous correspondence without any penalty on changes in the thrust vector direction. Thus, both the position and velocity differences between the current spacecraft state and a state integrated along the reference trajectory must be included in the reward function. However, the perturbed initial conditions are formulated to possess the same velocity at the initial epoch as the corresponding state along the reference trajectory. Thus, the reward function must be modified to encourage the networks to decrease the position prior to decreasing the velocity. Not incorporating this modification could result in the networks not performing any actions because decreasing the position difference inherently increases the velocity difference. To discourage this type of behavior, an exponential-like term is included in the velocity coefficient to increase the weight of the velocity error penalty over time. Initially, the reward function only depends on the position term while the magnitude of the velocity term tends to dominate at later time steps. Then, the reward function simplifies to:

$$r_1(\bar{x}_t, \bar{u}_t) = -10{,}000 * |\bar{d}_{ref}(t + \Delta t) - \bar{d}_{s/c}(t + \Delta t)| - 1{,}000 * F(t) * |\bar{v}_{ref}(t + \Delta t) - \bar{v}_{s/c}(t + \Delta t)| \tag{16}$$

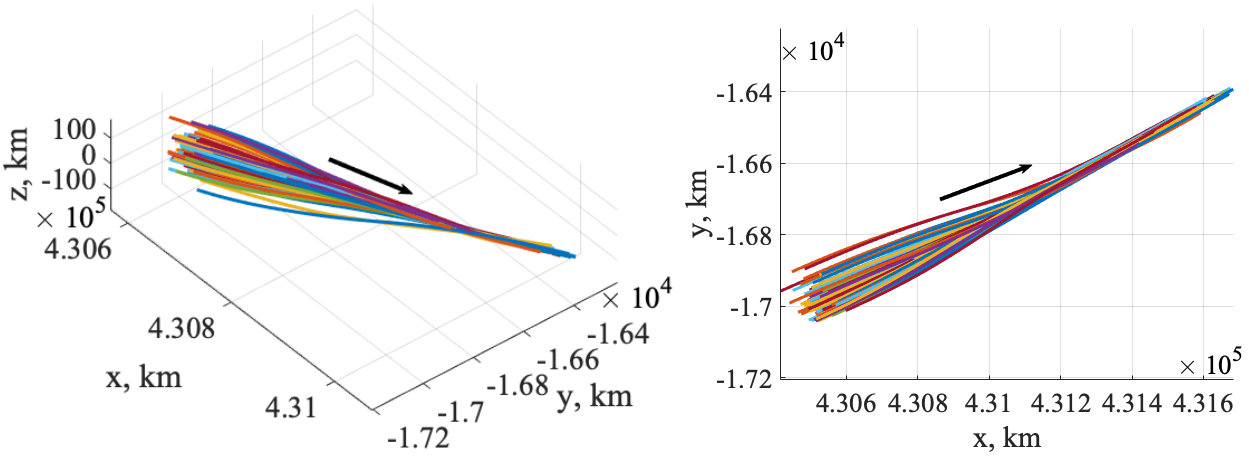
where c_d is set to 10,000 to scale the position difference term to be approximately -1 when the sampled positions are within 1E-4 nondimensional distance units, or 38.4 km, of the state along the reference trajectory at the current time step and c_v is set equal to 1,000 to similarly scale the velocity difference term to approximately -1. These coefficients aid convergence of the neural networks by scaling the rewards to be the same order of magnitude as the states and actions, which are both typically in the range [-1,1]. Then, an exponential-like term, $F(t)$, is defined as:

$$F(t) = \frac{k^{t/0.05} - 1}{k - 1} \tag{17}$$

and k is set equal to 1E10. Additionally, the state along the reference trajectory is propagated simultaneously with the spacecraft state consistent with the use of an isochronous correspondence. Using this reward function, the neural networks are trained for 315,000,000 time steps within the low-thrust-enabled CR3BP environment and then evaluated on a set of 100 perturbed initial conditions. The evaluation set of trajectories are depicted in the Earth-Moon rotating frame in Fig. 7a in the three-dimensional configuration space while Fig. 7b displays a projection onto the plane of the primaries. The trajectories are colored using a variety of hues for clear differentiation. Using the controls scheme derived from the neural networks, all of the generated solutions reach the reference trajectory within the specified upper bound of 50 time steps.

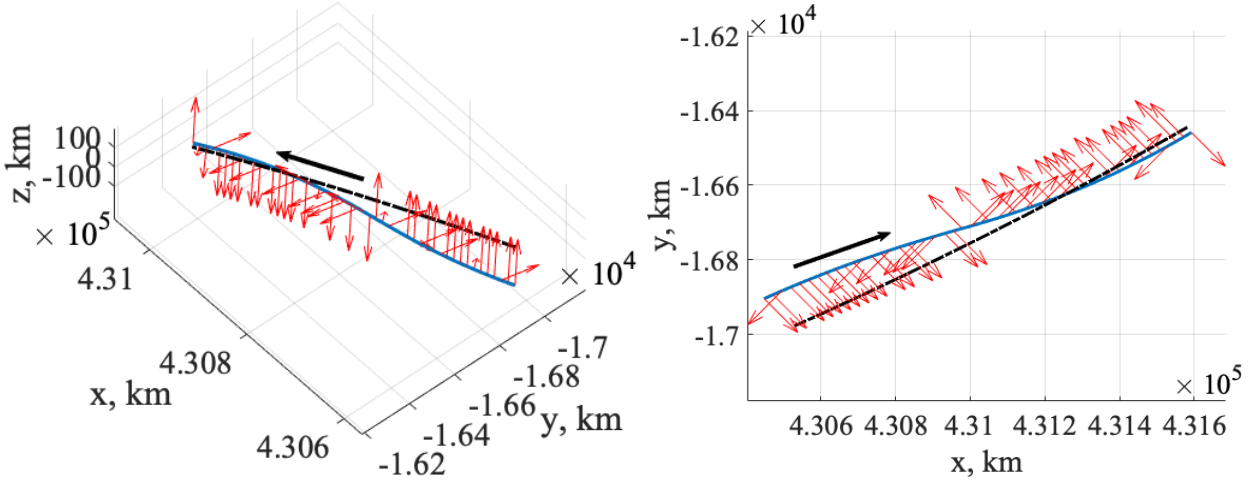
To further analyze the results of the trained networks, a single trajectory is examined. The initial condition for this trajectory is located approximately 122 km away from the specified initial condition on the reference trajectory. Through the generated maneuver profiles, the spacecraft reaches the reference trajectory to within 17.61 km and 1.77 m/s in position and velocity, respectively, within 5 hours and 12 minutes. This trajectory is presented in Fig. 8a with the controlled trajectory depicted in blue, the reference trajectory displayed as the black dashed line, and the red arrows representing the thrust unit vectors output from the neural networks at each time step. A planar view of the trajectory is also displayed in Fig. 8b. Both figures demonstrate a consistent understanding by the neural networks to direct the spacecraft towards the reference trajectory as quickly as possible. Once the SmallSat has approached the reference trajectory, the neural networks appropriately begin applying counter accelerations to avoid overshooting the reference. The large and regular changes in the direction of the thrust vector associated with this trajectory may be infeasible for a SmallSat over small time steps. Thus, modification of the reward function to penalize this behavior may be necessary.

Examining the state differences along the trajectories associated with the perturbed initial condition set offers further insight into the results of the neural networks. The magnitude of the position differences over time for all



(a) Spatial view (b) Planar view

Fig. 7 100 trajectories guided to a reference trajectory (black) using neural networks trained to minimize position and velocity differences, evaluated using an isochronous correspondence.



(a) Spatial view (b) Planar view

Fig. 8 Low-thrust trajectory from a highly perturbed initial condition in blue guided towards the reference trajectory, denoted by the black dashed line, with thrust directions depicted by red arrows, derived from the neural networks trained to reduce state differences, evaluated using an isochronous correspondence.

generated trajectories is displayed in Fig. 9a. All trajectories – with the exception of four that start from highly perturbed initial conditions – are guided to between 393 m and 10.78 km of the reference trajectory after 50 time steps. The magnitude of the velocity difference between the generated solution set and the reference is displayed in Fig. 9b. The maximum velocity difference at the final time step is 4.91 m/s due to a trajectory emanating from a highly perturbed initial condition. Furthermore, the mean of the final velocity difference is 1.21 m/s while the minimum is 0.28 m/s. Oscillations in the velocity differences through time are due to the neural networks performing an action to decrease the position difference and subsequently counteracting that increase in velocity at the proceeding time step.

Across all trajectories in the generated set, the change in the thrust vector direction directly influences the practicality of the maneuver profile. In fact, large and frequent changes in the direction of the thrust vector within the six minute time steps are largely impractical operationally or highly discouraged – particularly for a SmallSat. However, due to the definition of this dynamical environment where a constant thrust at each time step is applied, often the generated trajectories require large thrust vector direction changes to avoid drifting away from the reference trajectory. In practice,

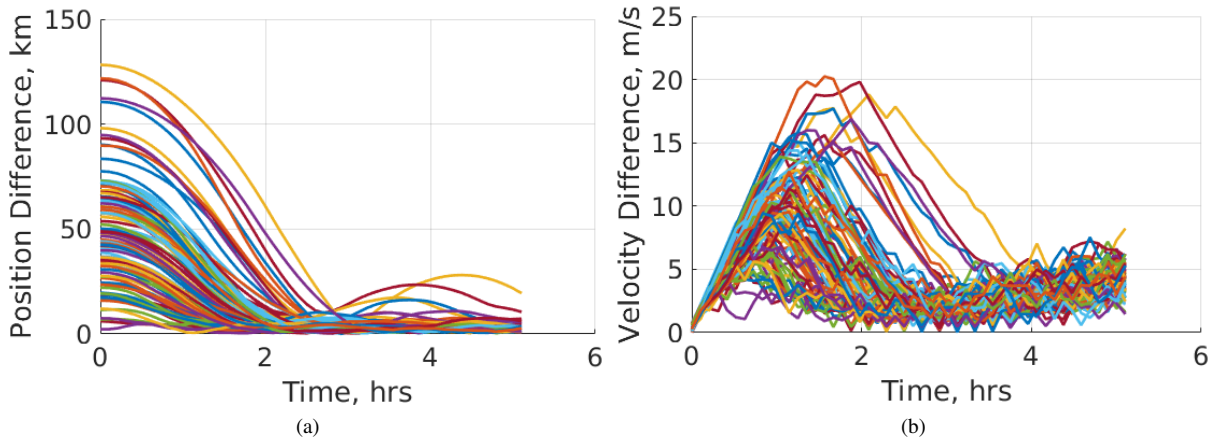


Fig. 9 Comparison between the reference and the controlled trajectories generated by the trained neural networks to decrease the state differences via an isochronous correspondence for (a) position and (b) velocity components of the state.

the SmallSat would likely coast once it is within a specified hypersphere of the associated state along the reference trajectory. Thus, post-processing is used. First, the hypersphere is defined with a radius of 10 km in position and 2.5 m/s in velocity. If the spacecraft passes within this hypersphere, the low-thrust engine is deactivated. Following this post-processing procedure, the angle changes occurring along the trajectories associated with 100 perturbed initial conditions are displayed in Fig. 10a. In this figure, the solid black line indicates the mean angle change across all non-converged trajectories, the dashed black lines represent one standard deviation, denoted σ, above and below the mean angle change, and the gray dots are the individual angle changes for each trajectory at each point in time. The high mean values of approximately 75 degrees indicate that on average, there are large thrust vector direction changes that would likely be infeasible for a SmallSat to perform. Furthermore, there are four angles that the neural networks repeatedly select, likely due to normalization of the actions to produce a unit vector. The thrust vector angle change for the single trajectory depicted in Fig. 8 is also displayed in Fig. 10b. This figure highlights the large and frequent angle changes associated with the recovered maneuver profile, thereby motivating the incorporation of a penalty on large changes in the thrust direction.

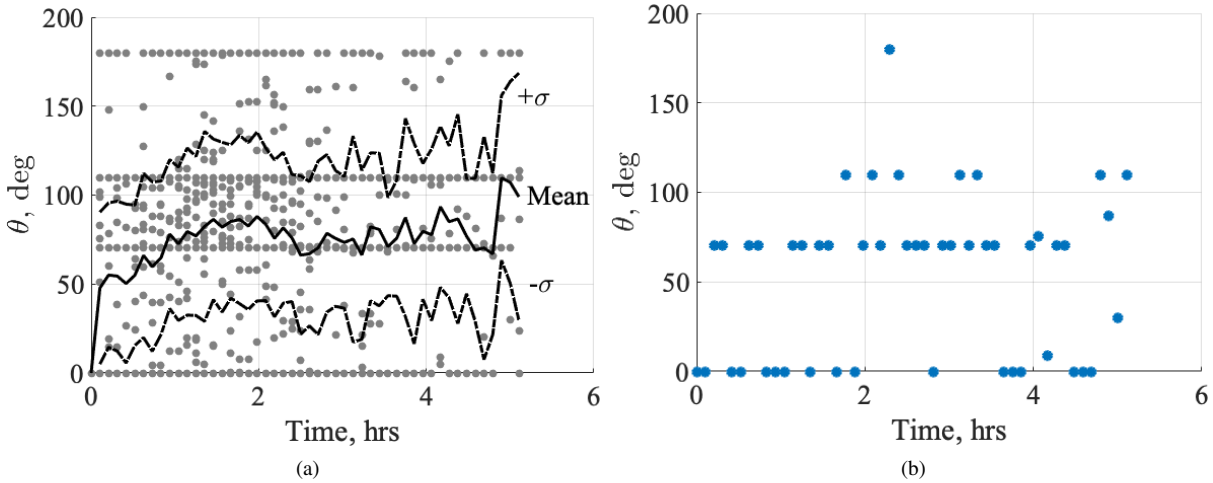


Fig. 10 Thrust vector direction changes along (a) all 100 trajectories and (b) a single trajectory generated by the trained neural networks to decrease the state differences evaluated using an isochronous correspondence.

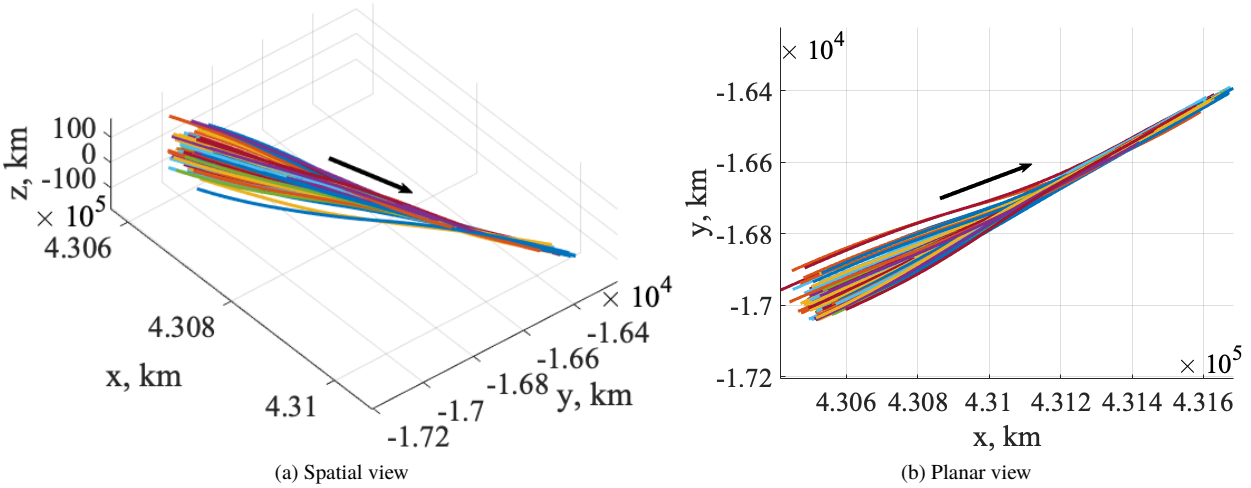
B. Tracking to a Reference Trajectory via a Normal Correspondence

In this second example, the reward function is formulated to ensure that the spacecraft tracks to any location along the reference trajectory. Since the CR3BP is autonomous and manifolds are time invariant structures, the SmallSat may access the reference trajectory located on a stable manifold at any time and still approach the associated periodic orbit. Thus, the reward function is reformulated to use a state on the reference trajectory that is closest in position space to the current spacecraft state, i.e., implementing a normal correspondence. In this case, the reward simplifies to:

$$r_2(\bar{x}_t, \bar{u}_t) = -10{,}000 * |\bar{d}_{ref,p} - \bar{d}_{s/c}(t + \Delta t)| - 1{,}000 * F(t) * |\bar{v}_{ref,p} - \bar{v}_{s/c}(t + \Delta t)| \tag{18}$$

where the subscript p denotes the state along the reference trajectory that is closest in position space to the SmallSat. The coefficient of the term penalizing changes in the control direction is set equal to zero and the agent is encouraged to reach any state on the reference trajectory by any means necessary. The neural networks are trained using this reward function for 315,000,000 time steps to understand the environment and maximize the returned reward. Figures 11a and 11b depict the trajectories associated with 100 perturbed initial conditions evaluated using the trained neural networks with a color scheme consistent with Fig. 7a. All of the perturbed trajectories approach the reference trajectory in configuration space within the maximum number of time steps. This behavior is highlighted in Figs. 12a and 12b which depict a single trajectory associated with the same highly perturbed initial condition as displayed in Figs. 8a and 8b using a similar color configuration. Analysis of these figures reveals that there are slight differences between the controls derived from each neural network attributable to the difference in the comparison to the reference trajectory at each time step. This particular trajectory corresponds to maneuvers that do not change in direction as frequently as in the previous case. The thrust direction changes and state differences that do occur across all trajectories must still be examined.

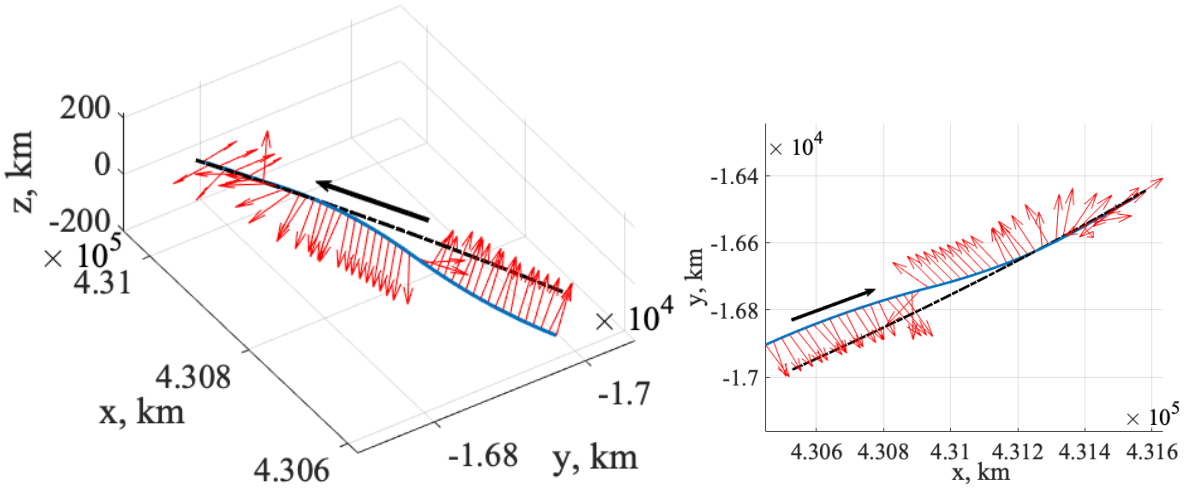
Examining the thrust vector direction changes and the state differences for the evaluated set of trajectories supplies insight into the characteristics of trajectories under policies that maximize the constructed reward function. The thrust vector direction changes for all trajectories are displayed in Fig. 13a using the same configuration as in Fig. 10a with a similar post-processing strategy. Similarly, the angle changes for the highlighted trajectory are depicted in Fig. 13b. Under the current reward function, formulated using a normal correspondence, lower mean changes in the thrust direction occur compared to the previous case. For the single trajectory presented, the thrust vector direction changes are also much smaller than those in the previous case. Additionally, analysis of the state differences between the perturbed solutions and the reference trajectory is useful. The magnitude of the position differences are displayed in Fig. 14a while the magnitude of the velocity differences are depicted in Fig. 14b using a color scheme consistent with Fig. 9a. When using a normal correspondence to formulate the reward function, the final position difference significantly decreases to between 80 m and 2.61 km. Furthermore, the velocity differences at the final time are much lower than that of the first reward function with a maximum velocity difference of 0.87 m/s: 80% lower than in the previous case. Overall, the mean position and velocity differences at the final time are 0.78 km and 0.49 m/s respectively, indicating the recovery of solutions that effectively reach the reference trajectory.



(a) Spatial view

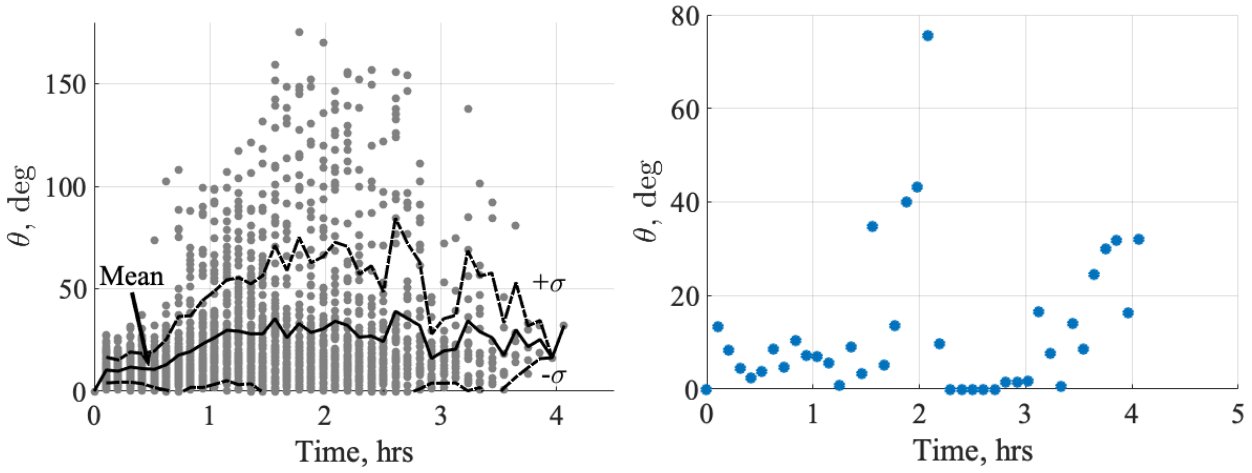
(b) Planar view

Fig. 11 100 trajectories guided to a reference trajectory (black) using neural networks trained to minimize position and velocity differences, evaluated using a normal correspondence.



(a) Spatial view (b) Planar view

Fig. 12 Low-thrust trajectory from a highly perturbed initial condition in blue guided towards the reference trajectory, denoted by the black dashed line, with thrust directions depicted by red arrows, derived from the neural networks trained to reduce state differences, evaluated using a normal correspondence.



(a) (b)

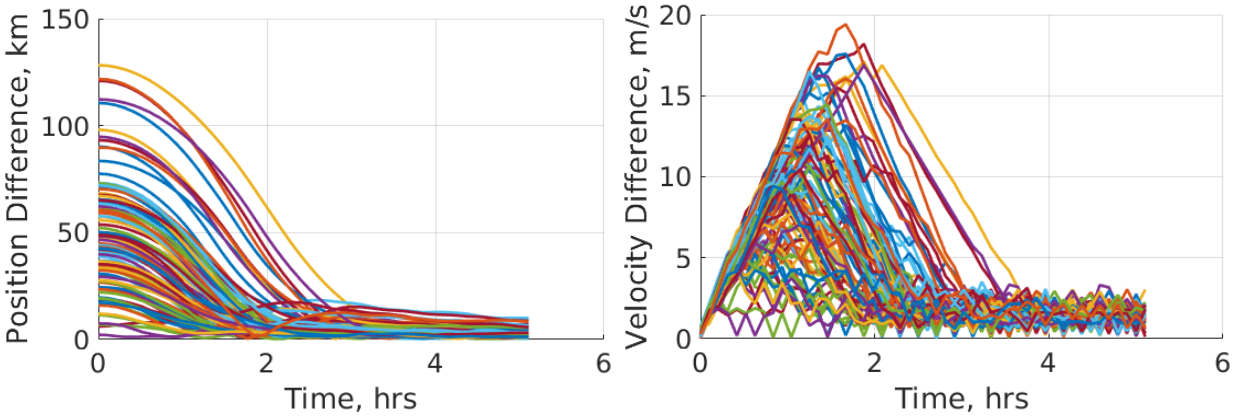
Fig. 13 Thrust vector direction changes along (a) all 100 trajectories and (b) a single trajectory generated by the trained neural networks to decrease the state differences, evaluated using a normal correspondence.

C. Tracking to a Reference Trajectory and Penalizing Thrust Vector Direction Changes

The third reward function encourages the spacecraft to track to any location on the reference trajectory while also minimizing changes in the direction of the thrust unit vector between time steps. In this case, the reward function simplifies to:

$$r_3(\bar{x}_t, \bar{u}_t) = -10{,}000 * |\bar{d}_{ref,p} - \bar{d}_{s/c}(t + \Delta t)| - 1{,}000 * F(t) * |\bar{v}_{ref,p} - \bar{v}_{s/c}(t + \Delta t)| - \theta \tag{19}$$

where the c_θ term is set equal to unity to scale the position, velocity, and thrust direction terms approximately equally; in this case, the position and velocity differences are calculated via a normal correspondence. The control profile developed by the neural networks trained on this reward function is evaluated using the same set of 100 perturbed initial conditions as in the first two cases. Figure 15a depicts a three-dimensional view of the resulting trajectories in configuration space while Fig. 15b displays a planar projection of the trajectories onto the plane of the primaries using a configuration that is consistent with Fig. 7a. The trajectories all approach the reference trajectory in configuration space within the maximum

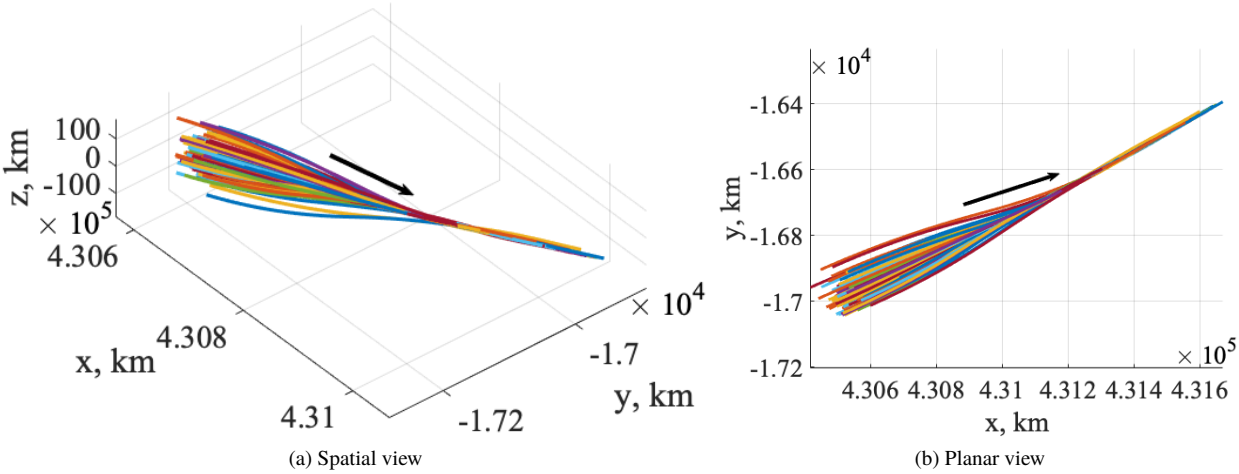


(a) (b)

Fig. 14 Comparison between the reference and the controlled trajectories generated by the trained neural networks to decrease the state differences via a normal correspondence for (a) position and (b) velocity components of the state.

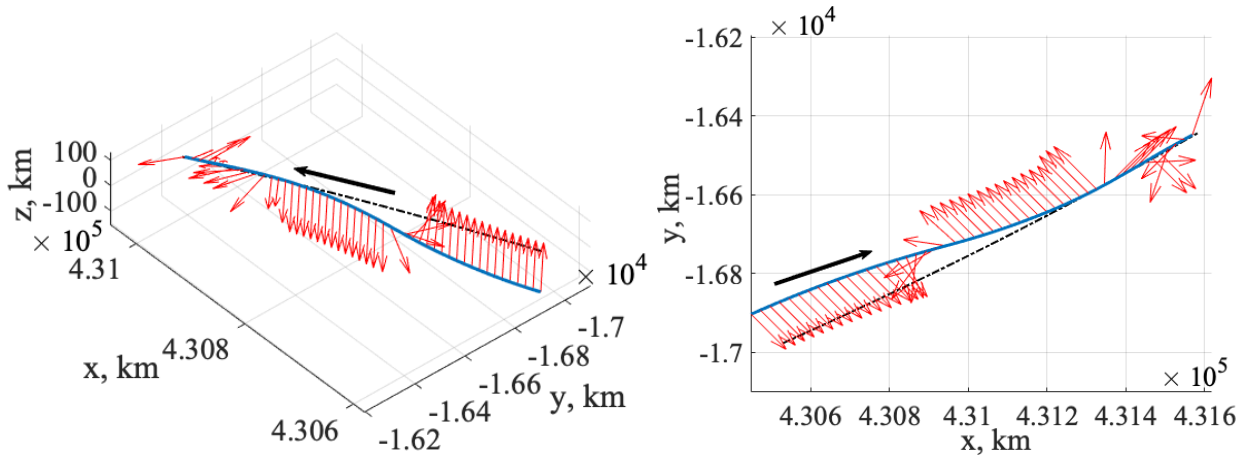
number of time steps with a geometry similar to that of the first two cases – despite the additional penalty on changes in the thrust direction. Figure 16a depicts the trajectory associated with a single, highly perturbed initial condition in blue with the thrust directions displayed with red vectors along with the manifold trajectory represented as a black dashed line; a planar perspective appears in Fig. 16b. The corresponding maneuver profile resembles those produced by the second set of trained neural networks where the spacecraft initially matches the velocity of the reference. However, in this case, smaller changes in the thrust direction occur across each time step.

Examining the thrust direction changes for all 100 perturbed trajectories supplies further insight into the influence of the thrust direction term in the reward function and on the developed policy. Using the same post-processing strategy as in the previous two cases, the change in the thrust direction for all trajectories is displayed in Fig. 17a along with the mean and one standard deviation values. Additionally, Fig. 17b depicts the thrust angle change throughout the selected trajectory. In both figures, there is a consistent decrease in the thrust vector direction change compared to the previous two cases. For instance, the mean thrust direction change for the evaluation set of trajectories is approximately 20° compared to the 30° thrust direction change for the second reward function. Further, the maximum thrust direction



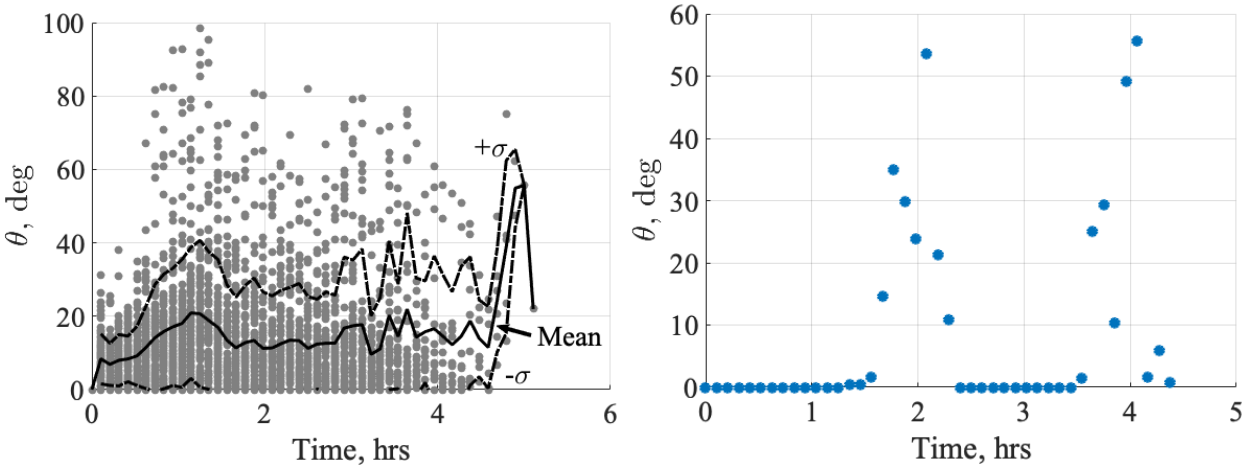
(a) Spatial view (b) Planar view

Fig. 15 100 trajectories guided to a reference trajectory (black) using neural networks trained to minimize position and velocity differences, evaluated using a normal correspondence, along with a penalty on changes in the thrust direction between time steps.



(a) Spatial view (b) Planar view

Fig. 16 Low-thrust trajectory from a highly perturbed initial condition in blue guided towards the reference trajectory, denoted by the black dashed line, with thrust directions depicted by red arrows, derived from the neural networks trained to reduce state differences evaluated using a normal correspondence and minimize changes in the thrust direction over time.



(a) (b)

Fig. 17 Thrust vector direction changes along (a) all 100 trajectories and (b) a single trajectory generated by the trained neural networks to decrease the state differences via a normal correspondence and with a penalty on changes in the thrust direction.

change in the previous two cases is 180° while this control profile has a maximum change of 100°. For the single trajectory examined, the maximum thrust direction change is limited to under 55° while previous cases admit maximum changes of 180° and 75°, respectively. Furthermore, the thrust direction in the first 15 time steps is nearly constant. These improvements demonstrate the impact of incorporating a penalty on the change in the thrust direction in the reward function along with the use of a normal correspondence.

The entire set of trajectories reveals solutions that successfully track to the reference trajectory within the specified time horizon. The magnitude of the position difference between each perturbed trajectory and the reference is displayed in Fig. 18a. The maximum final position difference for any trajectory is 13.87 km while the mean final position difference is 2.14 km. The mean and maximum final position differences are both much lower than the first case, but slightly higher than the second case; discouraging large thrust direction changes tends to limit how closely the perturbed

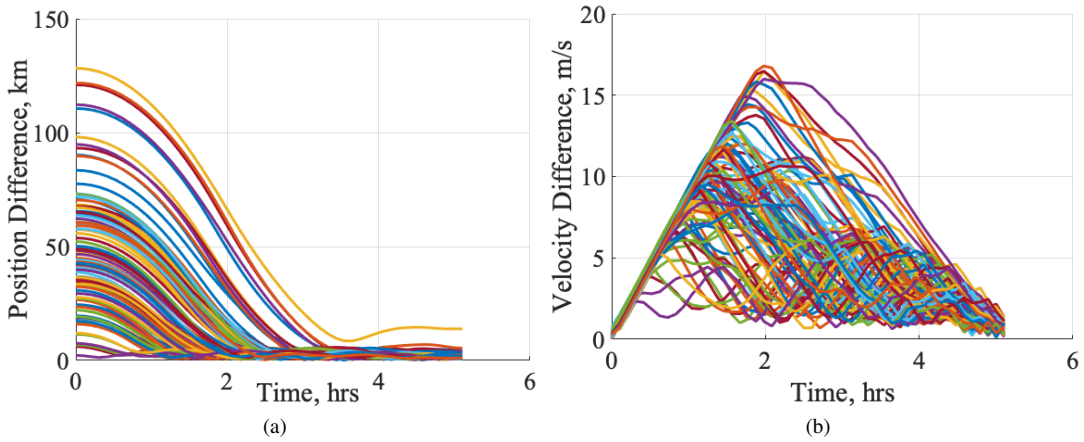


Fig. 18 Comparison between the reference and the controlled trajectories generated by the trained neural networks to decrease the state differences via a normal correspondence with a penalty on changes in the thrust direction for (a) position and (b) velocity components of the state

trajectories approach the reference trajectory within the specified time frame. Furthermore, the minimum final position difference is raised to 722 m. The magnitude of the velocity difference over time is depicted in Fig. 18b with similar characteristics as the solutions evaluated using the second trained neural networks. For instance, the average velocity difference at the final time is 1.31 m/s, the maximum difference is 4.51 m/s, and the minimum difference is 0.13 m/s. However, unlike the previous cases, these solutions do not maintain a low velocity difference for much time before the end of the simulation period.

VI. Concluding Remarks

A low-thrust-enabled SmallSat may be guided towards a periodic orbit by tracking to a reference trajectory that lies along an associated stable manifold. However, approaching a reference trajectory with limited propulsive capabilities poses challenges for trajectory and maneuver designers. One approach to designing trajectories and associated maneuver profiles under these constraints is to employ a DRL method to iteratively learn the dynamics and objectives to formulate a suitable control profile that produces a locally optimal solution. While training neural networks within a DRL method requires many iterations, reevaluating a trained neural network is computationally trivial. This paper focuses on one important component of implementing a DRL-based approach to trajectory design: formulation of a reward function that reflects both the design objectives and the constraints that influence the operational feasibility of the recovered maneuver profile. This analysis is performed in the context of trajectory design for a low-thrust-enabled small satellite to access a nearby reference trajectory that lies on a stable manifold associated with a periodic orbit in the CR3BP over a short time horizon.

In this analysis, neural networks are trained using three distinct reward functions and evaluated on a common set of initial conditions perturbed from the reference trajectory. The first reward function is designed to reward the spacecraft for possessing a state that is close to a state that lies along the reference trajectory; this reference state is defined using an isochronous correspondence. A second reward function is formulated to reward matching a reference state that is calculated via a normal correspondence. The third reward function then incorporates an additional term to minimize the change in the thrust direction over each time step, reflecting a significant operational constraint. All three neural networks generate maneuver profiles that enable a nearby spacecraft to track towards the reference in phase space – however, the reward function that leverages a normal correspondence produces solutions with tighter convergence towards the reference trajectory. Including a penalty on changes in the thrust direction in the reward function significantly improves the potential feasibility of the maneuver profile at the expense of higher position and velocity differences between the perturbed trajectories and reference solution. Across these three formulations of a reward functions, neural networks are used to successfully recover solutions for a low-thrust-enabled SmallSat to track to a reference trajectory on the stable manifold associated with an L_2 Lyapunov orbit over a short time horizon.

Acknowledgments

This research was performed at the University of Colorado Boulder. This work was supported by a NASA Space Technology Research Fellowship and the authors thank Dr. Alinda Mashiku for serving as the research collaborator under this fellowship.

References

[1] Bosanac, N., Cox, A., Howell, K., and Folta, D. C., "Trajectory Design for a Cislunar CubeSat Leveraging Dynamical Systems Techniques: The Lunar IceCube Mission," *Acta Astronautica*, 2018, pp. 283–296.

[2] Genova, A. L., and Dunham, D. W., "Trajectory Design for the Lunar Polar Hydrogen Mapper Mission," *27th AAS/AIAA Space Flight Mechanics Meeting, San Antonio, TX*, 2017.

[3] Johnson, L., Castillo-Rogez, J., Dervan, J., and McNutt, L., "Near Earth Asteroid (NEA) Scout," *4th International Symposium on Solar Sailing, Kyoto Japan*, 2017.

[4] Dachwald, B., "Evolutionary Neurocontrol: A Smart Method for Global Optimization of Low-Thrust Trajectories," 2004.

[5] Das-Stuart, A., Howell, K. C., and Folta, D. C., "Rapid Trajectory Design in Complex Environments Enabled via Supervised and Reinforcement Learning Strategies," *69th International Astronautical Congress*, 2018.

[6] Scorsoglio, A., Furfaro, R., Linares, R., and Massari, M., "Actor-Critic Reinforcement Learning Approach to Relative Motion Guidance in Near-Rectilinear Orbit," *29th AAS/AIAA Space Flight Mechanics Meeting, Ka'anapali, HI*, 2019.

[7] Miller, D., and Linares, R., "Low-Thrust Optimal Control via Reinforcement Learning," *29th AAS/AIAA Space Flight Mechanics Meeting, Ka'anapali, HI*, 2019.

[8] Szebehely, V., *Theory of Orbits: The Restricted Problem of Three Bodies*, London, UK: Academic Press, 1967.

[9] Koon, W. S., Lo, M. W., Marsden, J. E., and Ross, S. D., *Dynamical Systems, The Three-Body Problem and Space Mission Design*, Marsden Books, 2006.

[10] Davis, D. C., Phillips, S. M., Howell, K. C., Vutukuri, S., and McCarthy, B. P., "Stationkeeping and Transfer Trajectory Design for Spacecraft in Cislunar Space," *2017 AAS/AIAA Astrodynamics Specialist Conference, Stevenson, WA*, 2017.

[11] Oshima, K., and Yanao, T., "Jumping mechanisms of Trojan asteroids in the planar restricted three- and four-body problems," *Celestial Mechanics and Dynamical Astronomy*, Vol. 122, 2015. https://doi.org/10.1007/s10569-015-9609-4.

[12] Anderson, R., and Lo, M., "The Role of Invariant Manifolds in Low Thrust Trajectory Design (Part II)," *Journal of Guidance, Control, and Dynamics*, Vol. 32, 2009, pp. 1921–1930. https://doi.org/10.2514/6.2004-5305.

[13] Goebel, D. M., and Katz, I., *Fundamentals of Electric Propulsion: Ion and Hall Thrusters*, John Wiley Sons, 2008.

[14] Barto, A. G., "Reinforcement Learning and Dynamic Programming," *6th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design and Evaluation of Man-Machine Systems, Cambridge, MA*, Vol. 28, No. 15, 1995, pp. 407–412.

[15] Boyan, J. A., and Moore, A. W., "Generalization in Reinforcement Learning: Safely Approximating the Value Function," *Advances in Neural Information Processing Systems 7*, edited by G. Tesauro, D. S. Touretzky, and T. K. Leen, MIT Press, 1995, pp. 369–376. URL http://papers.nips.cc/paper/1018-generalization-in-reinforcement-learning-safely-approximating-the-value-function.pdf.

[16] Minsky, M., "Steps toward Artificial Intelligence," *in Proceedings of the IRE*, Vol. 49, No. 1, 1961, pp. 8–30. Doi: 10.1109/JRPROC.1961.287775.

[17] Singh, S. P., Jaakkola, T. S., and Jordan, M. I., "Learning Without State-Estimation in Partially Observable Markovian Decision Processes," *ICML*, 1994.

[18] Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D., "Deep Reinforcement Learning that Matters," , 2017.

[19] Nwankpa, C. E., Ijomah, W., Gachagan, A., and Marshall, S., "Activation Functions: Comparison of Trends in Practice and Research for Deep Learning," *CoRR, abs/1811.03378*, 2018.

[20] Kingma, D. P., and Ba, J., "Adam: A Method for Stochastic Optimization," , 2014.

[21] Konda, V. R., and Tsitsiklis, J. N., "Actor-Critic Algorithms," *Advances in Neural Information Processing Systems 12: Proceedings of the 1999 Conference, Denver, CO*, 2000, pp. 1008–1014.

[22] Wu, Y., Mansimov, E., Grosse, R. B., Liao, S., and Ba, J., "Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation," *Advances in Neural Information Processing Systems 30*, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Curran Associates, Inc., 2017, pp. 5279–5288. URL http://papers.nips.cc/paper/7112-scalable-trust-region-method-for-deep-reinforcement-learning-using-kronecker-factored-approximation.pdf.

[23] Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P., "High-Dimensional Continuous Control Using Generalized Advantage Estimation," , 2015.

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

[25] Williams, R. J., and Peng, J., "Function Optimization using Connectionist Reinforcement Learning Algorithms," *Connection Science*, Vol. 3, No. 3, 1991, pp. 241–268. https://doi.org/10.1080/09540099108946587, URL https://doi.org/10.1080/09540099108946587.

[26] Busek Co. Inc., "BHT-8000 Busek Hall Effect Thruster," , Online; accessed November 12th, 2019. URL http://www.busek.com/index_htm_files/70000703A%20BHT-8000.pdf.

[27] Bosanac, N., Alibay, F., and Stuart, J. R., "A Low-Thrust Enabled SmallSat Heliophysics Mission to Sun-Earth L5," *IEEE Aerospace Conference, Big Sky, MT*, 2018.

[28] Hill, A., Raffin, A., Ernestus, M., Gleave, A., Kanervisto, A., Traore, R., Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., and Wu, Y., "Stable Baselines," https://github.com/hill-a/stable-baselines, 2018.

[29] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," , 2015. URL https://www.tensorflow.org/, software available from tensorflow.org.