Designing Sun-Earth L2 Halo Orbit Stationkeeping Maneuvers via Reinforcement Learning

Stefano Bonasera*, Natasha Bosanac†, Christopher J. Sullivan*, Ian Elliott*, Nisar Ahmed‡ and Jay W. McMahon§
University of Colorado Boulder, Boulder, CO, 80309

Reinforcement learning (RL) is used to design impulsive station-keeping maneuvers for a spacecraft operating near an L_2 quasi-halo trajectory in a Sun-Earth-Moon point mass ephemeris model with solar radiation pressure. This scenario is translated into an RL problem that reflects the desired station-keeping goals, variables, and dynamical model. An algorithm from proximal policy optimization is used to train a policy that generates station-keeping maneuvers while transfer learning is used to reduce the computational time required for training. The trained policy successfully generates station-keeping maneuvers that result in boundedness to the vicinity of the selected reference trajectory with low total maneuver requirements, producing comparable results to a traditionally-formulated constrained optimization scheme.

I. Introduction

Autonomous maneuver planning in multi-body systems is a key capability that could enable and enhance future space missions. One type of maneuver that may occur throughout the lifetime of a mission is a station-keeping maneuver that maintains bounded motion near a desired path. Existing strategies for planning station-keeping maneuvers for spacecraft in a multi-body environment often leverage a combination of dynamical systems theory, optimization, and support from a human [1–3]. These strategies successfully recover efficient maneuvers at the expense of significant workload and computational resources. Developing approaches to autonomously plan these station-keeping maneuvers may be invaluable for reducing the operational cost and complexity of large observatories, formations, and small satellites as well as in scenarios that require either a rapid response or experience a communications delay.

Reinforcement learning (RL) algorithms have emerged as a tool for autonomously designing a control policy in complex dynamical environments. A general RL problem is formulated using an observation vector that represents the information the agent perceives from the environment. Then, a policy produces the action to apply given an observation,

*Ph.D. Graduate, Colorado Center for Astrodynamics Research (CCAR), Smead Aerospace Engineering Sciences, 3775 Discovery Dr., Boulder, CO 80303. CS and IE are AIAA Student Members.

†Assistant Professor, Colorado Center for Astrodynamics Research (CCAR), Smead Aerospace Engineering Sciences, 3775 Discovery Dr., Boulder, CO 80303. AIAA Senior Member. natasha.bosanac@colorado.edu

‡Associate Professor, Research and Engineering Center for Unmanned Vehicles (RECUV), Smead Aerospace Engineering Sciences, 3775 Discovery Dr., Boulder, CO 80303. AIAA Member

§Associate Professor, Colorado Center for Astrodynamics Research (CCAR), Smead Aerospace Engineering Sciences, 3775 Discovery Dr., Boulder, CO 80303. AIAA Associate Fellow.

An earlier version of this paper was presented as paper AAS 21-216 at the virtual AAS/AIAA Space Flight Mechanics Meeting, February 1-4, 2021.

This version of the paper has been accepted for publication by AIAA's Journal of Guidance, Control, and Dynamics after peer review but does not reflect any changes due to copy-editing and formatting by AIAA. The published manuscript is available online at: https://arc.aiaa.org/doi/10.2514/1.G006783

and an environment governs the transition between two specific observation-action pairs. RL problems also require a mathematical reward function that encodes the immediate reward for selecting an optimal action at a given observation. Using these definitions, the goal is to uncover a policy that maximizes the long-term reward, i.e., the discounted cumulative reward of successive observation-action pairs, also labeled as the value [4].

The use of RL to train policies in a wide variety of environments has inspired recent applications to astrodynamics [5–7]. In transfer design, Das-Stuart et al. use RL, graph searches, and optimization for low-thrust trajectory design [8], Miller and Linares train an RL agent for low-thrust enabled transfers between two distant retrograde orbits [9], Sullivan and Bosanac use multi-reward RL to explore a segment of a low-thrust trajectory design space [10], LaFarge et al. use RL to design continuous-thrust transfers between libration point orbits in the Earth-Moon system [11], Scorsoglio et al. apply RL to relative motion around periodic orbits [12], and Boone et al. explore incorporating uncertainties into training an RL agent for transfer design in the Sun-Earth system [13]. In the context of maneuver design within low-fidelity dynamical models, Bonasera et al. and Bosanac et al. apply RL to station-keeping and reconfiguration maneuver design near Sun-Earth L_2 halo orbits [14, 15], Guzzetti studies the performance of a tabular Monte Carlo implementation of RL for planar orbit maintenance [16], Molnar uses RL with dynamical systems theory to analyze and design station-keeping maneuvers near a spatial orbit [17], and LaFarge et al. use an off-policy RL framework to generate station-keeping maneuvers in the Earth-Moon system [18]. These applications motivate continued exploration of the use of RL in astrodynamics activities that currently rely heavily on large optimization problems and a human-in-the-loop.

In this paper, RL is used to generate impulsive station-keeping maneuvers for a spacecraft operating near a Sun-Earth L_2 quasi-halo trajectory. This scenario is modeled after a previous design of the Nancy Grace Roman Telescope [1]. During the multi-year mission, uncertainties and regular momentum unloads perturb the spacecraft to deviate from a desired reference path. Accordingly, regular station-keeping maneuvers must be designed to adjust the path of the spacecraft to maintain boundedness within the vicinity of a reference path while limiting control effort. RL is used in this paper to design station-keeping maneuvers because: 1) the underlying problem is a Markov decision process where a long-term objective function is optimized; 2) the exploration and exploitation tradeoff inherent to RL enables efficient exploration of the continuous state and action spaces; and 3) the trained policy produces a complex mapping between a broad range of values of the problem variables and the associated maneuvers that may be rapidly evaluated without an initial guess, albeit at the expense of higher initial computational effort during training [4]. First, an RL problem and training process are formulated in this paper to recover a policy that generates impulsive station-keeping maneuvers in the selected scenario. A policy that solves this problem is then trained using an algorithm from the proximal policy optimization (PPO) family implemented using PyTorch in Python [19]; PPO is employed due to previously successful applications to spacecraft trajectory design in a multi-body gravitational system [9–11]. Bayesian optimization is used to guide the selection of suitable hyperparameters and neural network structures that govern the policy and training process. Finally, transfer learning is used to progressively train the policy in dynamical models of increasing fidelity to

reduce the computational burden associated with training [20].

The trained policy is evaluated to generate impulsive station-keeping maneuvers for a spacecraft near a Sun-Earth L_2 quasi-halo trajectory in a Sun-Earth-Moon point mass ephemeris model with solar radiation pressure (SRP). First, the evolution of the policy during training is examined to both verify a successful training process and examine the utility of transfer learning. Then, the station-keeping maneuvers, generated by the trained policy, are compared to the results of a constrained optimization problem that minimizes control effort while directly constraining the position deviation from the reference at each maneuver location. The trained policy produces maneuvers with slightly lower magnitudes relative to the nearby locally optimal solutions that solve the constrained optimization problem. Minor differences in position deviation along the resulting trajectories exist due to distinct but comparable problem formulations for the two methods. Finally, the trained policy is evaluated for a single trajectory in the presence of regular momentum unloads. In this application, the trained policy results in a total maneuver magnitude of 2.285 m/s to bound the spacecraft to predominantly lie within 100 km and 0.05 m/s of the reference path over 10.33 years; comparable to the results produced by Bosanac et al. by solving a similar station-keeping maneuver design problem using constrained optimization [1].

This paper offers several contributions to the application of RL to maneuver design in multi-body gravitational systems. These contributions include: 1) demonstrating the use of RL to train policies for station-keeping in an ephemeris model; 2) formulating an RL problem that does not require a priori knowledge of the dynamical environment; 3) using Bayesian optimization to guide hyperparameter selection; and 4) demonstrating the use of transfer learning to significantly reduce the time to train the policy without introducing instability into the policy updates. Together, these contributions offer a foundation for continued exploration of the use of RL in autonomous maneuver planning for spacecraft in multi-body gravitational environments and, eventually, in more complex maneuver design scenarios.

II. Background: Dynamical Models

In this paper, a spacecraft station-keeping near a reference trajectory in the Sun-Earth system is governed by two dynamical models of increasing fidelity: the CR3BP and an ephemeris model. The configuration and equations of motion for each of these models are presented in this section. A periodic orbit that exists in the Sun-Earth CR3BP is then used in a multiple-shooting method to recover a nearby, continuous trajectory in an ephemeris model; this procedure is briefly summarized in this section. The periodic orbit that exists in the CR3BP and the nearby continuous trajectory that exists in an ephemeris model serve as reference paths during station-keeping maneuver design in each dynamical model.

A. Circular Restricted Three-Body Problem

The CR3BP describes the motion of a spacecraft of negligible mass subject to the point mass gravitational attraction of two significantly larger primaries that follow circular orbits about their mutual barycenter [21]. The equations of motion for the CR3BP are usually formulated in a reference frame with the origin at the system barycenter and axes

$(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}})$ that rotate with the two primaries, P_1 and P_2: the $\hat{\boldsymbol{x}}$-axis is directed from P_1 to P_2, the $\hat{\boldsymbol{z}}$-axis is parallel to the orbital angular momentum vector of the system, and the $\hat{\boldsymbol{y}}$-axis completes the orthogonal right-handed triad. In addition, normalization is often employed: the characteristic length is set equal to the distance between P_1 and P_2, the characteristic mass equals the sum of the masses of the primaries, and the characteristic time produces a nondimensional period of the primary system that is equal to 2π. Following normalization, μ represents the mass ratio, equal to $\mu \approx 3.00348064 \times 10^{-6}$ in the Sun-Earth CR3BP. Then, the nondimensional state vector for the spacecraft relative to the P_1-P_2 barycenter is defined in the rotating frame as $\boldsymbol{x} = [x,\ y,\ z,\ \dot{x},\ \dot{y},\ \dot{z}]^T \in \mathbb{R}^6$. Using these definitions, the nondimensional equations of motion for the spacecraft in the CR3BP and in the rotating frame 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 $U^* = (x^2 + y^2)/2 + (1-\mu)/d_1 + \mu/d_2$, $d_1 = \sqrt{(x+\mu)^2 + y^2 + z^2}$, and $d_2 = \sqrt{(x-1+\mu)^2 + y^2 + z^2}$. When formulated in the rotating frame, this dynamical model is autonomous and admits an energy-like integral of motion, the Jacobi constant, equal to $C_J = 2U^* - \dot{x}^2 - \dot{y}^2 - \dot{z}^2$. The CR3BP also admits a variety of solutions, including: five equilibrium points L_i for integers $i \in [1, 5]$, periodic and quasi-periodic trajectories, as well as chaotic motion [21].

B. Point Mass Ephemeris Model with Solar Radiation Pressure

In this paper, an ephemeris model is formulated to include the point mass gravitational influences of the Sun, Earth, and Moon following their actual paths, as well as SRP. The mass of the spacecraft is assumed to be negligible compared to the masses of the celestial bodies. Length, time and mass quantities are also normalized consistent with the CR3BP. The nondimensional position vector of body P_j relative to an inertially-fixed origin, O, is defined as $\boldsymbol{R}_j = X_j\hat{\boldsymbol{X}} + Y_j\hat{\boldsymbol{Y}} + Z_j\hat{\boldsymbol{Z}}$, where $(\hat{\boldsymbol{X}}, \hat{\boldsymbol{Y}}, \hat{\boldsymbol{Z}})$ are the axes of the Geocentric Celestial Reference Frame (GCRF). Then, the nondimensional position vector of P_i relative to P_j is denoted as $\boldsymbol{R}_{j,i} = \boldsymbol{R}_i - \boldsymbol{R}_j$. Using these definitions, the vector differential equation governing the motion of the spacecraft in the ephemeris model is written in the GCRF as

$$\ddot{\boldsymbol{R}}_{E,s/c} = -\frac{GM_E}{R^3_{E,s/c}}\boldsymbol{R}_{E,s/c} + G\sum_{i=L,S} M_i\left(\frac{\boldsymbol{R}_{s/c,i}}{R^3_{s/c,i}} - \frac{\boldsymbol{R}_{E,i}}{R^3_{E,i}}\right) + \frac{kAS_0}{cM_{s/c}}\frac{R_0^2}{R^3_{S,s/c}}\boldsymbol{R}_{S,s/c} \tag{2}$$

where G is the nondimensional universal gravitational constant; the subscripts E, S, L, and s/c correspond to the Earth, Sun, Moon, and spacecraft, respectively; and the nondimensional mass of body i is M_i. The final term is the nondimensional acceleration due to SRP, reflecting the force exerted by photons impacting the surface of an assumed spherical spacecraft [1]. In this term, k is the reflectivity coefficient, A is the spacecraft cross-sectional area, S_0 is the solar flux at the nominal Sun-Earth distance R_0, and c is the speed of light [1]; these quantities are nondimensionalized using the characteristic quantities of the Sun-Earth CR3BP. The position vectors of the celestial bodies in the GCRF are

accessed using the Jet Propulsion Laboratory DE421 ephemerides, via the SPICE toolkit [22, 23].

C. Recovering a Continuous Trajectory in an Ephemeris Model

Multiple shooting is used to recover a continuous trajectory that exists in an ephemeris model and retains the geometry of a nearby periodic solution from the CR3BP. The multiple shooting scheme is formulated following the procedure outlined by Bosanac et al. and briefly summarized here [1]. First, an initial guess is constructed for a desired initial epoch using multiple revolutions of a periodic orbit that exists in the CR3BP. The initial guess is formed by discretizing this path into multiple arcs: the description of each arc includes the initial state that is expressed in the GCRF following a coordinate transformation, the initial epoch, and the integration time along the arc. This information is combined into a free variable vector. A constraint vector is then formed to reflect state and time continuity between each pair of neighboring arcs. The free variable vector is then iteratively updated via Newton's method until the norm of the constraint vector equals zero to within a small tolerance. This approach produces the continuous trajectory, for a specific initial epoch, that serves as a reference path for a spacecraft operating in an ephemeris model in this paper.

III. Background: Reinforcement Learning via Proximal Policy Optimization

The goal of RL is to train a policy that maps a set of states to the actions that maximize the long-term reward for an agent interacting with its environment. Specifically, the observation vector at a time t is denoted as $\boldsymbol{o}_t$ and reflects the information the agent perceives from the environment. In this paper, the state space is assumed to be fully observable. At time t, the agent takes an action $\boldsymbol{u}_t$ determined by a policy function denoted $\pi_{\boldsymbol{\nu}}(\boldsymbol{u}_t|\boldsymbol{o}_t)$. When propagated forward in the environment across a single time step from t to $t+1$, this observation-action pair produces a subsequent observation $\boldsymbol{o}_{t+1}$. Successive observation-action pairs are generated across an episode, defined in this work to be composed of up to τ time steps [4]. Across each time step, the reward function $r_t(\boldsymbol{o}_t, \boldsymbol{u}_t, \boldsymbol{o}_{t+1})$ is evaluated using the observation-action pair at time t and the associated observation at time $t+1$ [4]. This reward is used to calculate the output from the value function for an observation at time t in the environment and is defined as

$$V^{\pi}(\boldsymbol{o}_t) = \mathbb{E}\left[\sum_{t=0}^{\tau} \gamma^t r_t(\boldsymbol{o}_t, \boldsymbol{u}_t, \boldsymbol{o}_{t+1})\right] \tag{3}$$

where $\gamma = [0, 1)$ denotes the discount factor which decreases the influence of future rewards compared to more immediate rewards and the superscript $(\cdot)^{\pi}$ indicates that each action is selected using the policy $\pi(\boldsymbol{u}_t|\boldsymbol{o}_t)$.

RL algorithms often leverage neural networks, which act as universal function approximators, to learn an optimal policy that maximizes the value function [5, 24]. A neural network is composed of nodes arranged in consecutive and connected layers, enabling outputs from a preceding layer to be fed into the following layer. Feed-forward neural networks, specifically, use an input layer, one or more hidden layers, and a final layer; weights and biases are assigned to

the connections between these layers. To incorporate nonlinearity into the network, each of the hidden nodes is assigned a nonlinear activation function, such as the hyperbolic tangent function [25]. The training parameters, comprising the weights and biases for each connection in this feed-forward neural network, are iteratively adjusted using a stochastic gradient descent optimization algorithm based on backpropagation. This approach captures, via the chain rule of derivatives, the influence of each training parameter on the gradient of a defined loss function [26, 27].

Actor-critic methods form the foundation of many state-of-the-art RL algorithms by combining the benefits of value-based and policy-based learning methods without significant penalties [28]. One common approach involves modeling each of the actor and critic via a feed-forward neural network. The actor neural network approximates the policy function $\pi_{\boldsymbol{\nu}}(\boldsymbol{u}_t|\boldsymbol{o}_t)$ and is parameterized by the weights and biases $\boldsymbol{\nu}$. The actor network is represented as a probability density function at each time t, conditioned by the observation $\boldsymbol{o}_t$, that samples the action $\boldsymbol{u}_t$ in the environment [4]. Concurrently, the critic approximates the value function, corresponding to the expected discounted cumulative reward obtained by starting from the observation $\boldsymbol{o}_t$ and continuing until the end of the episode while following the policy $\pi_{\boldsymbol{\nu}}(\boldsymbol{u}_t|\boldsymbol{o}_t)$ [4]. The explicit dependence of the critic neural network on the parameters $\boldsymbol{\xi}$ is expressed as $V^{\pi}_{\boldsymbol{\xi}}$. Together, the training parameters for both neural networks are summarized as $\boldsymbol{\theta} = [\boldsymbol{\nu}, \boldsymbol{\xi}]$.

In this work, the actor and critic neural networks are trained using an algorithm from the family of Proximal Policy Optimization that uses ratio clipping [29]; this specific algorithm is referred to as PPO throughout this paper. PPO is used in this paper due to its robust convergence towards optimal policies in chaotic environments and, as a result, successful application to trajectory design in multi-body systems [9–11, 29–31]. This algorithm uses a specified number of observation-action-reward experiences, collected from the environment, to form an update to the parameters of the neural networks. First, an advantage function is defined to determine the observation-action pairs in the environment that return the highest value. The advantages of each observation-action pair are estimated using Generalized Advantage Estimation (GAE) as $\hat{A}^{\pi}_t(\boldsymbol{o}_t, \boldsymbol{u}_t) = \sum_{\ell=0}^{\infty}(\gamma\lambda)^{\ell}\delta_{t+\ell}$ where δ_t is the temporal difference, defined as $\delta_t = r_t(\boldsymbol{o}_t, \boldsymbol{u}_t, \boldsymbol{o}_{t+1}) - V^{\pi}(\boldsymbol{o}_t) + \gamma V^{\pi}(\boldsymbol{o}_{t+1})$. The temporal difference represents the estimated advantage of the action $\boldsymbol{u}_t$ and λ is the GAE factor, which influences the bias-variance trade-off in the estimated advantages [32]. Then, PPO incorporates a trust region constraint inspired by another RL algorithm, Trust Region Policy Optimization, into the objective function to prevent large changes from destabilizing the neural networks [33]. In this paper, the change in the neural networks for an update j is measured using the difference between the probabilities of the old and new policies to generate $\boldsymbol{u}_t$ upon receiving $\boldsymbol{o}_t$, calculated as $R_t(\boldsymbol{\nu}) = \pi_{\boldsymbol{\nu},j}(\boldsymbol{u}_t|\boldsymbol{o}_t) - \pi_{\boldsymbol{\nu},j-1}(\boldsymbol{u}_t|\boldsymbol{o}_t)$ where values near zero correspond to smaller updates to the neural networks [10, 34, 35]. Then, a loss function is necessary to compute updates to the parameters of the neural networks using the stochastic gradient descent algorithm and is defined as $L^{\text{CLIP}}(\boldsymbol{\theta}) = \hat{\mathbb{E}}_t\left[\min(R_t(\boldsymbol{\nu})\hat{A}_t, \text{clip}(R_t(\boldsymbol{\nu}), -\varepsilon, \varepsilon)\hat{A}_t)\right]$ where the clipping function clip$(\cdot)$ constrains the probability difference to possess a magnitude less than or equal to the clipping parameter, ε [29]. The expected value $\hat{\mathbb{E}}_t[\cdot]$ represents an approximation of the true expectation $\mathbb{E}[\cdot]$, used to compute a loss over the sampled experiences. The

sampled loss is used to update $\boldsymbol{\theta}$ during the stochastic gradient descent step. Two additional terms are appended to the loss function to guide the networks to uncover the value function for the environment and to incentivize continued exploration in the environment [29]. The resulting modified objective function used in PPO is written as

$$L_t^{\text{CLIP+VF+S}}(\boldsymbol{\theta}) = \hat{\mathbb{E}}_t[L_t^{\text{CLIP}}(\boldsymbol{\theta}) - c_1 L_t^{\text{VF}}(\boldsymbol{\xi}) + c_2 S[\pi_{\boldsymbol{\nu},j}](\boldsymbol{o}_t)] \tag{4}$$

where $S[\pi_{\boldsymbol{\nu},j}](\boldsymbol{o}_t)$ is an entropy term influencing the amount of exploration within the environment, $L_t^{\text{VF}}(\boldsymbol{\xi}) = (V_{\boldsymbol{\xi}}^{\pi}(\boldsymbol{o}_t) - V_t^{\text{true}})^2$ is a squared-error loss term encouraging learning the true value function, c_1 is the squared-error loss coefficient, and c_2 is the entropy coefficient [29]. This objective function is used in a stochastic gradient descent optimization algorithm to train the neural networks using experiences within the environment; this paper employs the AdamW optimization algorithm, which has been demonstrated to efficiently and robustly converge in RL [29, 31, 36].

During training, PPO's performance is influenced by several hyperparameters [30, 31]. In this paper, hyperparameters of interest include the clipping parameter, the squared-error value loss coefficient, the entropy coefficient, the GAE factor, the discount factor, and the learning rate that governs how quickly the policy is updated by the AdamW optimizer. Two additional hyperparameters include the number of epochs, specifying how many times each observation-action pair is used to update the parameters, and the number of batches, defining the number of groups the observation-action pairs are divided into at each epoch [10]. Note that throughout this paper the definition of epoch depends on the context: in an RL framework, an epoch corresponds to one forward pass over the training experiences whereas in trajectory design, an epoch reflects an instant of time. The hyperparameters are typically selected to achieve an efficient training process that balances exploration and exploitation to converge on a policy that maximizes the expected discounted cumulative reward within a reasonable number of updates. Various strategies for selecting these quantities exist, including manual tuning, random search, grid search, Bayesian optimization, and reinforcement learning [37, 38].

IV. Approach to Station-Keeping Maneuver Design

An RL-based maneuver planner is formulated for a spacecraft station-keeping near a southern L_2 halo orbit in the Sun-Earth system. An overview of this scenario, modeled after the upcoming Nancy Grace Roman Space Telescope, is presented. Then, station-keeping maneuver design is translated into an RL problem. In addition, the use of transfer learning for efficient training is outlined. Next, the use of Bayesian optimization to guide selecting the hyperparameters governing PPO and the neural network structures is presented. Finally, a comparable constrained optimization problem is formulated to verify the maneuvers generated by the trained policy.

A. Scenario Overview

The reference path for the spacecraft is modeled after an early conceptual design of the Nancy Grace Roman Space Telescope [1]. Specifically, a maneuvering spacecraft is assumed to operate near a bounded and natural reference trajectory that resembles a Sun-Earth L_2 southern halo orbit. When modeling the dynamics via the Sun-Earth CR3BP, a periodic L_2 southern halo orbit with a period of $T_r \approx 180$ days and a Jacobi constant of $C_J = 3.00078$ is employed as a reference trajectory; this path is displayed in Figure 1(a) in the Sun-Earth rotating frame using dimensionless coordinates measured relative to the Earth. The correction scheme summarized in Section II.C is used to recover a nearby continuous trajectory that exists in the defined ephemeris model; the spacecraft is assumed to possess a dry mass of 6877 kg, a cross-sectional area of 49.6 m^2 and a coefficient of reflectivity equal to $k = 2$ [1]. Following corrections, a nearby continuous reference trajectory is generated from an initial epoch of 24 June 2021 19:22:22 UTC until a final epoch of 30 June 2042 14:02:41 UTC; this trajectory is displayed in Figure 1(b) in a pulsating Sun-Earth rotating frame using dimensionless coordinates measured from the Earth.

Impulsive station-keeping maneuvers are often used to mitigate the impact of uncertainties, momentum unloads, and off-nominal maneuvers on the path of a spacecraft. In this paper, regular momentum unloads are the only mechanism for perturbing the path of the spacecraft, and are modeled as an instantaneous change in velocity, denoted $\Delta \boldsymbol{v}_{MU}$; full state and model parameter knowledge are assumed with nominal maneuver implementation. Similar to the scenario presented by Bosanac et al., based on an early iteration of the Nancy Grace Roman Space Telescope mission parameters, momentum unloads are assumed to occur in a random direction with a magnitude of 8.7 mm/s every $t_{MU} = 130$ hours ≈ 5.417 days [1]. The station-keeping maneuvers and momentum unloads occur at evenly-spaced time intervals, forming momentum unload cycles [1]. Each cycle begins with an impulsive station-keeping maneuver; in this paper, there is no constraint on its magnitude or direction. This station-keeping maneuver is followed by a coast arc with a duration of t_{MU}. Three momentum unloads and three more coast arcs of duration t_{MU} then occur in alternating order to form a single momentum unload cycle. Following the scenario presented by Bosanac et al., the goal is to design station-keeping

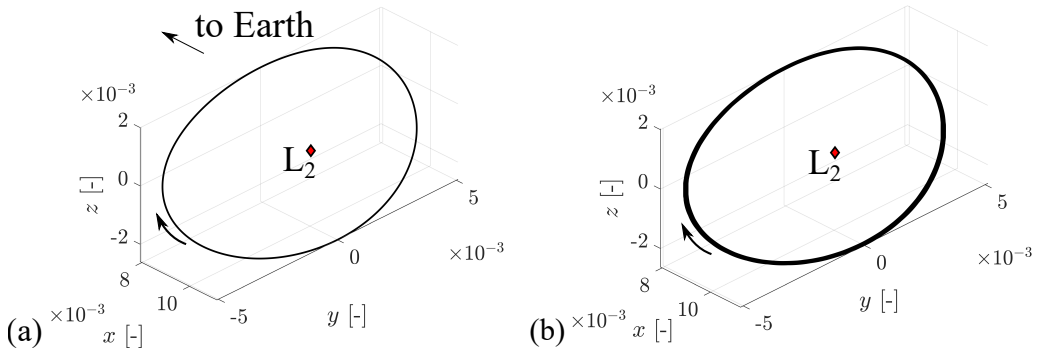

Fig. 1 Reference paths near Sun-Earth L_2: (a) southern halo orbit in the CR3BP and (b) 21-year trajectory in the ephemeris model with initial epoch of 29390.308 MJD.

maneuvers that, in the presence of regular momentum unloads, produce a spacecraft path that remains sufficiently bounded within the vicinity of the reference trajectory while requiring less than 5 m/s per year for at least 10 years [1].

B. Translating Station-Keeping Maneuver Design into an RL Problem

Each episode is defined to consist of up to $\tau \in \mathbb{N}$ time steps and, therefore, encompass τ station-keeping maneuvers and coast arcs. The τ arcs along the controlled path of the spacecraft appear as blue arcs in the conceptual representation of the station-keeping problem in Fig. 2 whereas the reference path is depicted in black. At the beginning of each time step, an observation-action pair is used to form a six-dimensional state vector $\boldsymbol{x}$ for the spacecraft, depicted as a gray circle in Fig. 2. The impulsive station-keeping maneuver at the beginning of the time step is labeled $\Delta\boldsymbol{v}$ and depicted with red arrows in Fig. 2. This maneuver is followed by a coast arc with a duration of Δt, equal to the duration of a momentum unload cycle, i.e., $\Delta t = 21.67$ days. Each arc is generated via numerical integration in the selected dynamical model without any perturbations during training. However, during evaluation of the trained policy, momentum unloads are also applied to regularly perturb the path of the spacecraft. Using these definitions, an RL implementation of the station-keeping maneuver design problem is specified in this subsection through the definition of an observation vector, action vector, and reward function; during this discussion, the remaining annotations in Fig. 2 are also defined.

Using a finite length episode is consistent with both applications of RL and station-keeping maneuver design. In RL, this approach prevents the generation of episodes of infinite length that may reduce the performance of the trained policy at the moment of deployment. In astrodynamics, one common station-keeping maneuver design strategy involves targeting a specific goal over a finite time horizon [1, 39, 40]. Selecting this finite time horizon as a limited number of revolutions near the reference trajectory reflects limitations in predicting the future path of the spacecraft due to momentum unloads, off-nominal maneuvers, modeling uncertainty, orbit determination errors, and numerical propagation errors. Furthermore, missions that involve placing spacecraft near the same reference trajectory may possess a variety of finite durations that may also evolve post-launch. In this paper, the finite length episode is selected to encompass approximately one revolution of the reference trajectory. However, examining the impact of a longer episode

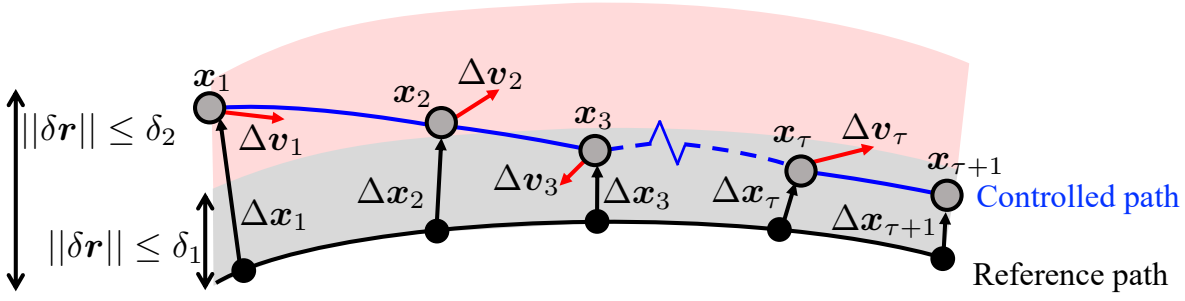

Fig. 2 Conceptual representation of station-keeping maneuver design problem variables and regions used to define the reward function.

on policy training and accuracy as well as the resulting maneuver magnitudes is an interesting avenue for future work.

The observation vector $\boldsymbol{o}$ is formulated to reflect the state of the spacecraft and relevant problem variables. To represent the state of the spacecraft in the dynamical environment, the following quantities are employed: $\boldsymbol{x}_{\text{ref}}$, the desired state vector along the reference trajectory; $\Delta\boldsymbol{x}$, a relative state vector that is measured isochronously from a state along the reference trajectory to the associated spacecraft state along the controlled trajectory; and t_E, the epoch that lies within the range $[e_0, e_f]$ and is expressed numerically as a modified Julian date (MJD). Both $\boldsymbol{x}_{\text{ref}}$ and $\Delta\boldsymbol{x}$ are specified in a pulsating Sun-Earth rotating frame. Although these two state vectors do not provide a minimal description of the spacecraft state, they do reflect both the nominal and perturbed states that describe the station-keeping problem. In addition, the epoch t_E is used to fully specify the state of the spacecraft in the epoch-dependent ephemeris model and support the use of transfer learning when training in the autonomous CR3BP. Finally, a quantity s is defined to reflect the number of remaining maneuvers within the episode: $s = 2i/\tau - 1$ for $i \in \{\tau, \tau - 1, \ldots, 0\}$. This quantity is observed to improve the training process by enabling the critic to distinguish between similar observation vectors and maneuvers that occur at distinct time steps along an episode. This information is combined to form the following observation vector:

$$\boldsymbol{o} = [\tilde{\boldsymbol{x}}_{\text{ref}}, \Delta\tilde{\boldsymbol{x}}, \tilde{t}_E, \tilde{s}] \in \mathbb{R}^{14} \tag{5}$$

where the tilde notation indicates that the components of this observation vector are scaled to generally remain within the range $[-1, 1]$ during training. This vector exists within a 14-dimensional continuous observation space.

The observation vector is used as an input to the policy to generate an action. The action, $\boldsymbol{u}$, is formulated as a 3×1 vector that exists in a continuous action space. This action vector is scaled by α to produce the components of the impulsive station-keeping maneuver, $\Delta\boldsymbol{v}$, in the Sun-Earth rotating frame, i.e., $\Delta\boldsymbol{v} = \alpha\boldsymbol{u}$. To encourage generating actions within the range $[-1, 1]$, the scaling factor is empirically selected as $\alpha \approx 0.14$ m/s. However, for relatively large displacements, $\|\Delta\tilde{\boldsymbol{x}}\|$, the policy may generate actions outside of the range $[-1, 1]$ if needed.

Initialization of each episode during training and evaluation depends on the specific dynamical model used. When propagating the spacecraft state in the Sun-Earth CR3BP, an initial state along the reference orbit, i.e., $\boldsymbol{x}_{\text{ref}}$, is randomly selected and the initial epoch is randomly selected within the range $[e_0, e_f]$ MJD. When propagating in the ephemeris model, an episode is initialized by randomly selecting the initial epoch t_E to produce the associated state vector $\boldsymbol{x}_{\text{ref}}$ along the reference trajectory. Then, each component of the relative state vector is initialized by first randomly sampling from a continuous uniform distribution in the range $[-1, 1]$. This quantity is multiplied by an order of magnitude randomly selected from a triangular distribution between -5 and 0.2 and centered at 0, as well as a scaling factor of 300 km and 5.96 mm/s in each position and velocity component. This sampling scheme ensures sufficient sampling of the relative state across each order of magnitude. The trained policy gains sufficient knowledge in solving the RL problem close to the reference path where the spacecraft would ideally be located when successfully station-keeping.

Similarly, the policy gains sufficient experience with initial states with a large deviation from the reference such as when the spacecraft is significantly impacted by perturbations. Finally, by definition, $\tilde{s} = 1$ at the beginning of an episode.

The reward function is formulated to reflect two goals of the spacecraft station-keeping near a Sun-Earth L_2 southern quasi-halo trajectory over a long time interval: minimizing the control effort and maintaining sufficient boundedness in the vicinity of the reference path, assessed isochronously. These objectives are written mathematically to ensure that achieving these goals corresponds to maximization of the long-term reward. However, in practical applications of RL, defining a suitable, non-sparse reward function that encourages efficient learning is a well-known challenge with no unique solution [41, 42]. In this paper, the reward is defined as a piecewise function that is formulated as follows:

1) When the relative position vector $\delta\boldsymbol{r}(\Delta t)$ at the end of an arc possesses a magnitude greater than δ_2, the spacecraft is no longer located within the vicinity of the reference; in Fig. 2, the spacecraft would be located beyond the red shaded region. Accordingly, a heavy penalty is applied via a constant, large negative reward and the episode is terminated. In this paper, $\delta_2 \approx 1.67 \times 10^{-5} = 2499$ km is selected for this position error threshold.
2) If the relative position vector $\delta\boldsymbol{r}(\Delta t)$ at the end of an arc possesses a magnitude between δ_2 and δ_1, with $\delta_1 < \delta_2$, the spacecraft is located in the vicinity of the reference path but not within the desired threshold, i.e., the red shaded region in Fig. 2. The goal is to further reduce the distance of the spacecraft from the reference path while also minimizing control effort. Thus, the position deviation is included in the reward function within a natural logarithm term, also labeled a logarithmic barrier, and the action vector appears within a linear term.
3) When the spacecraft relative position vector $\delta\boldsymbol{r}(\Delta t)$ at the end of an arc possesses a magnitude below δ_1, the spacecraft lies within the desired threshold for station-keeping; in Fig. 2, the spacecraft would be located within the gray shaded region. In this paper, $\delta_1 \approx 6.68 \times 10^{-7} = 100$ km, consistent with the threshold used by Bosanac et al. [1]. The goal is then to reduce the required control effort while maintaining a relative position vector with a magnitude that is less than δ_1. Thus, the deviation in the position vector, used within a natural logarithm term, is clipped to equal δ_1. In addition, a discontinuity in the reward function is introduced by scaling the clipped position deviation term using the constant coefficient $B = \ln(\delta_1/2)/\ln(\delta_1) \approx 1.05$. Incorporating this constant term, $B\ln(\delta_1)$, produces a discontinuity in the piecewise reward function. Selecting $B > 1$ produces an increase in the reward because the value of the unscaled position deviation term at the boundary $||\delta\boldsymbol{r}|| = \delta_1$ equals $-\ln(\delta_1) = 14.219$. This approach is empirically observed to encourage the policy to produce maneuvers that drive the position of the spacecraft to within the desired threshold of the reference trajectory. Increasing the value of B is observed to generally decrease the time steps required for the controlled trajectory to reach the desired region near the reference at the expense of increased control effort.

Based on these goals, the reward at a single time step is summarized mathematically as follows:

$$r = \begin{cases} -B\ln(\delta_1) + K(1 - \|\boldsymbol{u}\|) & \text{if } \|\delta\boldsymbol{r}(\Delta t)\| \leq \delta_1 \\ -\ln(\|\delta\boldsymbol{r}(\Delta t)\|) + K(1 - \|\boldsymbol{u}\|) & \text{if } \|\delta\boldsymbol{r}(\Delta t)\| \leq \delta_2 \wedge \|\delta\boldsymbol{r}(\Delta t)\| > \delta_1 \\ -100 & \text{otherwise} \end{cases} \tag{6}$$

In this paper, $K = 15$ is empirically selected to balance maintaining sufficient boundedness with minimizing control effort. Alternative values of K would result in a distinct relative weighting between these two goals and, therefore, lead to the generation of distinct policies.

To train a policy to solve the defined RL problem while limiting the required computational effort, transfer learning is employed: a policy trained in a simpler problem or model initializes a policy trained in a more complex problem or model [20]. In this paper, training occurs in a two-step process: a policy is trained in the CR3BP, where numerical integration for generating state transitions is fast, to form an initial guess for a policy that is trained in the ephemeris model, where numerical integration is more time-consuming. To facilitate a smooth transition during transfer learning, the observation vector, action vector, and reward function formulations are defined equivalently when training in each distinct dynamical model; the reference trajectories, however, that exist in each model slightly differ.

C. Selecting Hyperparameters and Neural Network Structures

Bayesian optimization is used to guide hyperparameter selection due to its tendency to exhibit sample efficiency for problems with expensive cost function evaluation, such as those that require numerical integration [37, 43]. Bayesian optimization describes a user-specified cost function via a surrogate model. The goal of the optimizer is then to identify the inputs, e.g., selected parameters governing the training process and policy, that maximize the cost function. An acquisition function uses the surrogate model to select the input set for the next cost function evaluation by balancing the exploration of areas with large uncertainty with the exploitation of regions where the expected mean cost function is large. This process continues until a specified termination condition is satisfied. This approach is implemented using the python toolbox *BayesianOptimization*, with the Matérn kernel and the upper confidence bound acquisition function [44].

Bayesian optimization is used to guide the selection of parameters governing both the training process and neural networks. The following hyperparameters that govern PPO are used as inputs to Bayesian optimization: discount factor, clipping rate, number of epochs and batches, value and entropy coefficients, and GAE factor. To aid the training process in identifying a local optimum, the learning rate evolves across one order of magnitude over updates using a sequence of step functions as $l_r(j) = l_{r,0} + \left\lfloor (jn_{steps})/(n_{upd}) \right\rfloor (l_{r,f} - l_{r,0})/(n_{steps})$ where n_{upd} is the total number of updates to be performed during training, j is an integer reflecting the current update, $l_{r,0}$ and $l_{r,f}$ are the initial and final learning rates, and n_{steps} is the total number of steps for the learning rate during training [15]. Thus, the initial learning rate and

number of steps also serve as hyperparameters that are input to Bayesian optimization. In addition, the depth and width of each of the actor and critic neural networks are also identified via Bayesian optimization; both neural networks use the hyperbolic tangent activation functions between consecutive layers and orthogonal initializers [10, 31].

To reduce the required computational effort, Bayesian optimization is implemented in the Sun-Earth CR3BP. The cost function used in Bayesian optimization to select these hyperparameters and neural network structures is defined as the sum of the following two terms: 1) the average discounted cumulative reward of the last update batch set and 2) the mean derivative of the value over the last 10 updates, approximated via forward finite differences. This cost function favors the hyperparameters and neural networks producing policies that are both effective in the final update and have exhibited improvements within at least the final several updates. A total of 130 iterations or, equivalently, runs of the training process are employed during optimization: the first 50 iterations correspond to evaluating random combinations of input parameters, whereas the subsequent 80 iterations are governed by the acquisition function. Training a policy for each combination of input parameters during Bayesian optimization terminates after a total of 2×10^6 time steps are generated. This total number of steps is selected at a lower value than used in later sections of this paper to limit the significant computational effort required for Bayesian optimization. Of course, convergence may not occur within that number of updates across all policies. Thus, the parameters identified by Bayesian optimization to maximize the cost function may not precisely predict the hyperparameters and neural network structure that produce the highest average discounted cumulative reward after convergence. However, the results of Bayesian optimization do offer a useful guide for further manual fine-tuning. Furthermore, this step uses the CR3BP for training because it offers a sufficient approximation of the dynamical environment near Sun-Earth L_2 in an ephemeris model, while significantly reducing the computational time required to train a large number of policies. It is assumed that the selected hyperparameters and neural network structures also result in a sufficiently effective training process in the ephemeris model.

The results of Bayesian optimization are used to guide manual tuning in the scenario outlined in Section IV.B and in the Sun-Earth CR3BP. Hyperparameters and neural network structures are selected using three criteria to:

1) Produce a high value of the cost function used in Bayesian optimization, corresponding to policies with a large average reward at the final update and potential for continued improvement with extended training
2) Lie in a region of the hyperparameter space that is well-explored during the last 80 iterations of optimization as the acquisition function exploits regions with reduced uncertainty near optimal values of the quantities of interest
3) Follow recommendations from authors applying PPO to various complex dynamical models [10, 29–32]

Guided by these three criteria, the selected hyperparameters and neural network structures are summarized in Table 1; the column labeled “B.O. Ranges" lists the allowable ranges of values for each quantity during Bayesian optimization.

Table 1 Selected hyperparameters and neural network (NN) parameters.

Parameter	Value	B.O. Ranges
Initial learning rate, $l_{r,0}$	5×10^{-3}	$[10^{-5}, 10^{-1.5}]$
Final learning rate, $l_{r,f}$	5×10^{-4}	-
Learning rate schedule	8 steps	$[2, 8]$ steps
Number of epochs, N_e	5	[2, 10]
Number of batches, N_b	6	[2, 10]
Value coefficient, c_1	1×10^{-3}	$[10^{-4}, 10^{-0.3}]$
Entropy coefficient, c_2	7×10^{-3}	$[10^{-6}, 10^{-1}]$
Clipping parameter, ϵ	0.02	$[10^{-5}, 10^{-2}]$
Discount factor, γ	0.99	$[0.9, 0.999]$
GAE factor, λ	0.99	$[0.8, 0.999]$
Actor NN depth, D_{act}	3	$[1, 6]$
Actor NN width, W_{act}	16	$[2, 2^{10}]$
Critic NN depth, D_{cri}	1	$[1, 6]$
Critic NN width, W_{cri}	1024	$[2, 2^{10}]$

D. Translating Station-Keeping Maneuver Design into a Constrained Optimization Problem

The station-keeping maneuvers generated by the trained policy are compared to those generated by a well-known existing approach: constrained optimization. A general constrained optimization problem is expressed as

$$\min_{\boldsymbol{X}} f(\boldsymbol{X}) \quad \text{subject to} \quad \boldsymbol{F}(\boldsymbol{X}) = \boldsymbol{0},\ \boldsymbol{G}(\boldsymbol{X}) \leq \boldsymbol{0} \tag{7}$$

where $\boldsymbol{X}$ is the free variable vector, $f(\boldsymbol{X})$ is the scalar objective function, $\boldsymbol{F}(\boldsymbol{X})$ is an equality constraint vector, and $\boldsymbol{G}(\boldsymbol{X})$ is an inequality constraint vector. Although there are several useful formulations, this paper uses a constrained optimization problem that implements the station-keeping maneuver design strategy as follows: for a fixed initial spacecraft state that is perturbed from a reference trajectory, a multiple-shooting based approach is used to identify a sequence of τ maneuvers that minimize the cumulative control effort while directly constraining each maneuver location to lie within δ_1 of the associated position vector along the reference trajectory. Of course, translating the goal of maintaining boundedness in the vicinity of the reference into a direct constraint, as opposed to a term within the objective function, produces a difference between the constrained optimization problem and the presented RL problem. However, in a traditional station-keeping maneuver design approach, such a goal is often formulated as a direct constraint [1, 23]. Given that RL and constrained optimization use slightly different procedures and problem formulations in this paper, the two approaches are not expected to produce equivalent maneuvers and controlled trajectories. However, if the total maneuver magnitudes generated by the two methods are on the same order of magnitude and the relative position deviation along the two controlled trajectories possess similar time histories, then the RL-trained policy may offer an

alternative approach to producing useful solutions to the station-keeping maneuver design problem.

A free variable vector $\boldsymbol{X}$ is defined to capture the spacecraft states and maneuvers at the beginning of all τ arcs along a maneuver sequence. Accordingly, the free variable vector is defined as

$$\boldsymbol{X} = [\boldsymbol{x}_2, \ldots, \boldsymbol{x}_\tau, \boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_\tau]^T \in \mathbb{R}^{9\tau-6} \tag{8}$$

where $\boldsymbol{x}_i$ is the spacecraft state at the beginning of the i-th arc and immediately before a maneuver is applied. This state is expressed non-dimensionally in the pulsating Sun-Earth rotating frame and is equivalent to $\boldsymbol{x}_i = \boldsymbol{x}_{ref,i} + \Delta\boldsymbol{x}_i$. The initial spacecraft state, $\boldsymbol{x}_1$, and the epoch at each maneuver location are not included in the free variable vector to reflect that the initial reference state, perturbation, and epoch as well as the time between maneuvers are fixed.

The equality and inequality constraint vectors, $\boldsymbol{F}(\boldsymbol{X})$ and $\boldsymbol{G}(\boldsymbol{X})$, respectively, are defined to ensure that the solution to the constrained optimization problem corresponds to a continuous trajectory that lies within the desired region near the reference trajectory. First, the equality constraint vector is formulated to reflect state continuity as:

$$\boldsymbol{F}(\boldsymbol{X}) = [\boldsymbol{x}_{u,1}(\Delta t) - \boldsymbol{x}_2,\ \boldsymbol{x}_{u,2}(\Delta t) - \boldsymbol{x}_3, \ldots,\ \boldsymbol{x}_{u,\tau-1}(\Delta t) - \boldsymbol{x}_\tau]^T \in \mathbb{R}^{6(\tau-1)} \tag{9}$$

where $\boldsymbol{x}_{u,i}(\Delta t)$ is the spacecraft state at the end of the i-th arc, generated by propagating in the ephemeris model for a duration of Δt from the state $\boldsymbol{x}_i$ that is modified by the maneuver $\alpha\boldsymbol{u}_i$. Next, an inequality constraint vector is defined to ensure that the end of each arc along the controlled path remains within a relative distance of δ_1 from the reference path, measured isochronously. This nonlinear inequality constraint is defined mathematically as

$$\boldsymbol{G}(\boldsymbol{X}) = [\|\boldsymbol{r}_2 - \boldsymbol{r}_{\text{ref},2}\| - \delta_1, \|\boldsymbol{r}_3 - \boldsymbol{r}_{\text{ref},3}\| - \delta_1, \ldots, \|\boldsymbol{r}_\tau - \boldsymbol{r}_{\text{ref},\tau}\| - \delta_1, \|\boldsymbol{r}_{\tau+1} - \boldsymbol{r}_{\text{ref},\tau+1}\| - \delta_1]^T \in \mathbb{R}^\tau \tag{10}$$

where $\|\cdot\|$ represents the l^2-norm, $\boldsymbol{r}_i$ and $\boldsymbol{r}_{ref,i}$ are the nondimensional position vectors at the beginning of the i-th arc along the controlled and reference trajectories, respectively, in the pulsating Sun-Earth rotating frame, and $\boldsymbol{r}_{\tau+1}$ is equal to the position vector at the end of the τ-th arc along the controlled trajectory.

The objective function is formulated to minimize the magnitudes of the τ maneuvers within a maneuver sequence or episode. Thus, the objective function is specified as $f(\boldsymbol{X}) = \sum_{i=1}^{\tau} \boldsymbol{u}_i^T\boldsymbol{u}_i$. Although distinct from the reward function, the objective function uses the sum of the squared l^2 norms of the scaled maneuvers to provide a globally first-order continuous function. In a multi-body system, there may be multiple solutions to this constrained optimization problem. Thus, in a comparison to the results of the RL-based approach, the observation-action pairs that are generated by the trained policy are used to form an initial guess for the free variable vector in constrained optimization; this approach supports recovery of a nearby locally optimal solution to effectively compare the maneuver sequences and the generated trajectories constructed by each of the two methods. Constrained local optimization is then performed via sequential

quadratic programming in *fmincon* within MATLAB® using analytical derivatives [45]. An optimal solution must also satisfy the constraints to within a tolerance of 5×10^{-14} in fewer than 100 iterations.

V. Results: Station-Keeping Maneuver Design in an Ephemeris Model via Reinforcement Learning

A policy is trained using PPO to generate impulsive station-keeping maneuvers for a spacecraft operating near a Sun-Earth L_2 quasi-halo trajectory in an ephemeris model. In this application, an episode is modeled to encompass a sequence of $\tau = 8$ maneuvers every 21.67 days, spanning approximately one revolution of the quasi-halo trajectory. The policy is trained to design this sequence of station-keeping maneuvers for minimization of control effort while maintaining sufficient boundedness to the vicinity of the reference trajectory displayed in Figure 1(b) by solving the RL problem specified in Section 4. During training, regular perturbations due to momentum unloads are not modeled. This section examines the evolution of this policy, initialized without any domain knowledge, throughout the training process as well as the use of transfer learning. The policy trained via transfer learning is then compared to the results of the constrained optimization problem. Once trained, impulsive station-keeping maneuvers are generated for a spacecraft operating in the vicinity of the reference trajectory for approximately 10 years while subject to regular momentum unloads. This application is used to demonstrate that the maneuvers generated by the trained policy successfully adjust the path of the spacecraft to remain close to the reference trajectory with a low total maneuver magnitude.

A. Examining the Training Process

The training process is analyzed to ensure that: 1) the policy is improving throughout training until converging to a large value and 2) the use of transfer learning produces a similar or better policy than training solely in the ephemeris model. In this subsection, two policies are trained to solve the same RL problem using two distinct approaches. The first policy is trained solely in the ephemeris model without any a priori domain knowledge; training is governed by the hyperparameters presented in Section IV.C and continues for up to 1×10^7 time steps. In this case, training requires approximately 72.76 hours on an i7-2600K 3.40GHz processor using 6 logical cores. The second policy, however, uses transfer learning to speed-up a computationally expensive training process. First, the policy is trained in the autonomous Sun-Earth CR3BP for a maximum of 5×10^6 steps using the halo orbit in Figure 1(a) as a reference trajectory along with the hyperparameters presented in Section IV.C. Then, the trained neural networks are used to initialize the actor and critic networks for training in the ephemeris model using the Sun-Earth L_2 quasi-halo trajectory in Figure 1(b) as a reference path. To avoid abrupt modifications of the policy during the second phase of training, a constant learning rate of $l_r = 5 \times 10^{-4}$, i.e., the final learning rate used during training in the CR3BP, is used throughout the 5×10^6 steps of the second training step in the ephemeris model. All remaining hyperparameters are selected as listed in Section IV.C. For this second policy, training requires approximately 37 minutes and 36.25 hours for the phases that use the CR3BP

and point mass ephemeris model, respectively, on an i7-2600K 3.40GHz processor using 6 logical cores.

To verify the use of transfer learning, consider the value over each update for each policy during training. This information is displayed in Fig. 3 for a) the policy trained in the ephemeris model and b) the policy trained using transfer learning. In Fig. 3(a), the value of the policy trained solely in the ephemeris model gradually increases until converging to a high average of 129.54 over the final 20 updates, without any destabilizing updates. Such behavior indicates that the policy is successfully learning to achieve the goals encapsulated within the reward function. Figure 3(b) displays in red and blue the value for the policy when trained in the Sun-Earth CR3BP and the ephemeris model, respectively. The update at which the policy is transferred is captured in the zoomed-in view. The value for this second policy gradually increases until converging towards an average of 129.69 over the last 20 updates; this quantity is similar to the average value produced by training solely in the ephemeris model. Across the transfer, the value does not significantly change, indicating the absence of destabilization due to the change in the dynamical model and reference trajectory.

To further examine the use of transfer learning during training, both policies are evaluated to produce a sequence of 8 impulsive station-keeping maneuvers using the same set of 100 initial conditions in the ephemeris model without regular momentum unloads. During evaluation, the two policies produce two separate actions and, therefore, two separate trajectories for each initial condition. After evaluation, the following quantities are evaluated for each policy: the average of the total maneuver magnitude along each maneuver sequence; the average position and velocity deviations, assessed by sampling the trajectory at 32 locations from the end of the first time step onward, thereby mitigating the influence of the initial perturbation magnitude on this quantity; the average of the total reward, defined as the non-discounted sum of the rewards at each time step along a trajectory; and the percentage of episodes that result in the controlled trajectory remaining within δ_2 of the reference trajectory in the configuration space. These quantities are displayed in this order in Table 2 for each of the policies across the 100 eight-maneuver sequences. These results reveal that the policy trained with transfer learning generates station-keeping maneuvers with comparable but slightly higher average rewards as

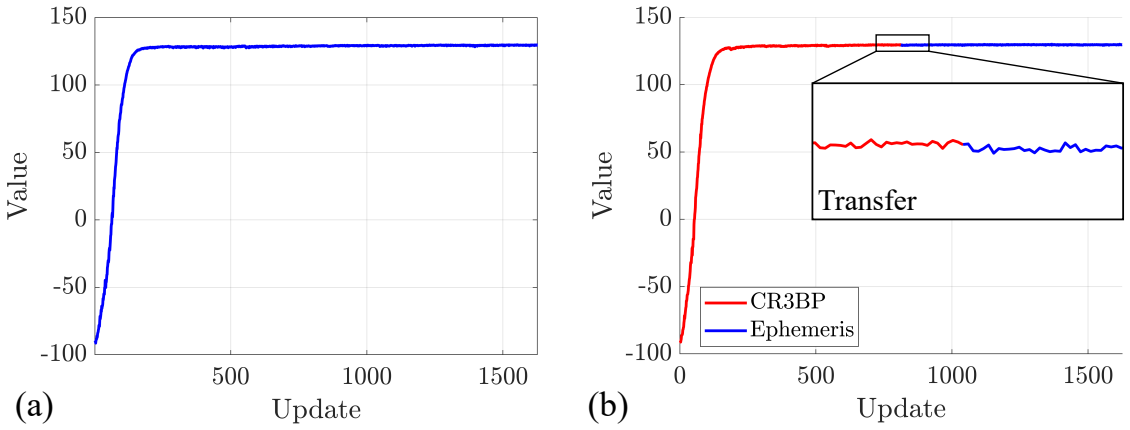

Fig. 3 Discounted cumulative reward, i.e., the value, over updates for policies trained in (a) only the ephemeris model and (b) using transfer learning.

well as lower total maneuver magnitudes and average position and velocity deviations when compared to the policy generated without transfer learning. Both policies generate station-keeping maneuvers that do not cause any of the 100 trajectories to deviate outside of the acceptable region of the reference trajectory defined by δ_2. Thus, a transfer learning approach to training produces a policy that designs station-keeping maneuvers to achieve the desired goals with comparable and slightly better performance in terms of the control effort and position deviation than the policy trained without transfer learning. As a direct result of these observations and the benefit of reducing the computational time by a factor of two, the policy trained via transfer learning is used throughout the remainder of this paper.

Table 2 Characteristics of 8-maneuver sequences and associated trajectories generated using policies trained with and without transfer learning (TL) and evaluated using 100 common initial conditions.

Policy trained:	Without TL	With TL
Average total Δv [m/s]	0.050	0.044
Average $\|\|\delta \boldsymbol{r}\|\|$ [km]	29.652	26.713
Average $\|\|\delta \boldsymbol{v}\|\|$ [m/s]	0.01245	0.01189
Average total reward	233.64	234.23
Success rate [%]	100	100

B. Comparison to Traditional Constrained Optimization

A detailed comparison is performed between the maneuver sequences and trajectories generated by each of the trained policy and constrained optimization for a single initial condition and without momentum unloads. The trained policy is first evaluated using a randomly-generated initial condition, possessing a deviation of 250.10 km and 0.0016 m/s from a state along the reference path on Nov 2, 2040 at 22:34:08.183 UTC, to produce an 8-maneuver sequence. Then, the trajectory produced by evaluating the trained policy is used to construct an initial guess for constrained optimization. The following properties of the two maneuver sequences and trajectories are plotted in Fig. 4 for each of the RL (blue) and constrained optimization (red) methods: (a) the magnitude of the position vector along the controlled trajectory measured isochronously from the reference path and (b) the magnitude of each station-keeping maneuver. In Fig. 4(a), the circles indicate the position deviation at the beginning of each time step and the black dashed line corresponds to the $\delta_1 = 100$ km threshold that defines sufficient boundedness in the vicinity of the reference trajectory.

The maneuvers generated by the trained policy and constrained optimization possess similar magnitudes and produce trajectories with similar position deviations. First, the two time histories of the position deviation possess similar geometric characteristics. However, the trajectory generated via constrained optimization approach always achieves a relative position vector with a magnitude below 100 km at the end of every time step. As a result, constrained optimization produces slightly larger maneuvers than the trained policy for the first three time steps to more quickly place the spacecraft within 100 km of the reference path and satisfy the constraints. Then, the remaining five maneuvers generated via optimization are consistently lower than those generated by the trained policy as the spacecraft remains

close to the reference. The trained policy, however, slightly exceeds this desired position deviation at the end of the first and fourth time steps in a trade-off with reducing the total maneuver magnitude. Across the entire episode, the total maneuver magnitudes generated by the trained policy and constrained optimization are $\Delta v_{tot,RL} = 0.118$ m/s and $\Delta v_{tot,Const.Opt.} = 0.142$ m/s, respectively. Notably, the first three maneuvers contribute most significantly to the larger total maneuver magnitudes for constrained optimization. However, these minor differences between the maneuvers and relative trajectories are a direct consequence of the slight differences in problem formulation across the two methods.

Comparison between maneuver sequences generated via the trained RL policy and constrained optimization is expanded from the previous analysis to encompass 500 randomly-generated initial conditions that lie near the reference trajectory at various initial epochs. The following two metrics are evaluated across 500 maneuver sequences and trajectories and then visualized using the histograms in Fig. 5: (a) the average magnitude of the position deviation from the end of the first step to the end of the last step and (b) the sum of the maneuver magnitudes over each episode. The histograms associated with the RL policy and constrained optimization appear as blue and red bars, respectively. The same coloring scheme is used to locate the mean values of each quantity, displayed using dashed lines.

Figure 5(a) reveals that the average position deviation from the reference path is reasonably low for both methods. First, the histogram summarizing the results of constrained optimization possesses a compact distribution around a mean of 69.97 km with extrema of 93.62 km and 26.22 km. Conversely, the average position deviations of the controlled trajectories produced by the RL policy possess a mean of 77.13 km and span a wider range of values between 9.59 km and 166.19 km; 21.2% of the 500 trajectories exceed the desired 100 km at least once. This difference in the histograms is due to constrained optimization enforcing a 100 km constraint on the position deviation at the end of every time step whereas RL incorporates minimizing the position deviation into the reward function. As a result, the trajectory

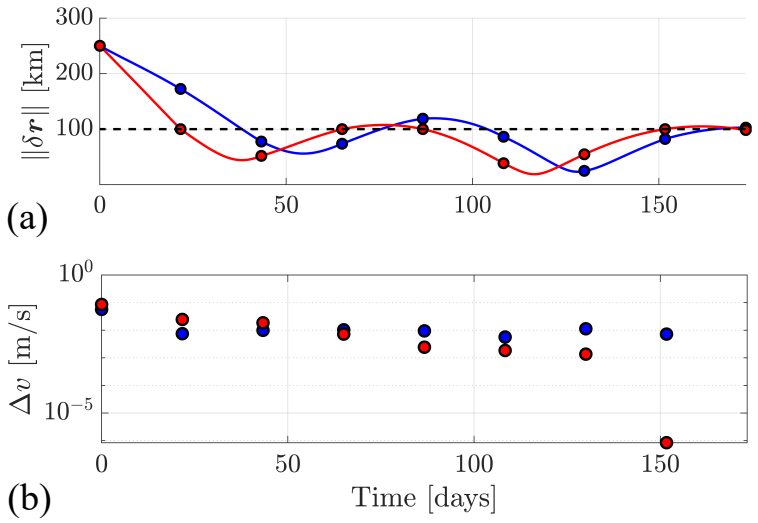

Fig. 4 Station-keeping maneuvers generated by the trained policy (blue) and constrained optimization (red) for one initial condition via: (a) isochronous position deviation and (b) maneuver magnitudes

generated by the trained policy often lies slightly outside of the desired 100 km region around the reference for one or more time steps. However, these deviations tend to be reasonably small across the entire maneuver sequence.

Figure 5(b) reveals similar total maneuver magnitudes over each episode. These histograms possess similar means of 0.117 m/s and 0.129 m/s for the trajectories generated with constrained optimization and the trained RL policy, respectively, as well as similar distributions; these observations indicate a comparable total maneuver magnitude across the 500 episodes. For a closer comparison, Fig. 6 summarizes the difference between the magnitudes of the (a) first and (b) final maneuvers along each pair of trajectories generated from the same initial condition. The dashed red line locates the mean of the distributions. The histogram in Fig. 6(a) possesses a mean of 9.12×10^{-3} m/s, indicating that, on average, constrained optimization produces a slightly larger first maneuver along an episode than the trained policy. This result is consistent with the position constraint enforced during optimization and the subsequent smaller dispersion of average position deviations from the reference trajectories in Fig. 5(a). Conversely, the histogram in Fig. 6(b) summarizes the difference in magnitude between the final maneuvers along each pair of trajectories. This histogram spans only negative values with a mean of -5.5×10^{-3} m/s, indicating that constrained optimization consistently produces final maneuvers with a lower magnitude than those generated by the trained policy. Thus, both methods generate maneuver sequences with similar total maneuver magnitudes, but slightly different distributions across the individual maneuvers due to slight differences in problem formulation. These results have some potential implications for applying the trained policy to station-keeping maneuver design: when regularly evaluating the policy by generating only the first maneuver in the 8-maneuver sequence, the station-keeping maneuvers may possess comparable or smaller magnitudes to those

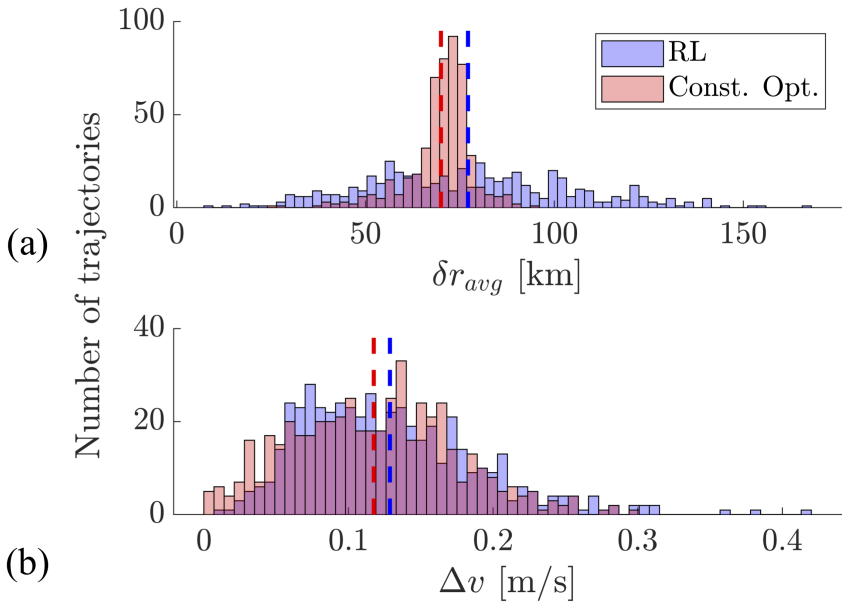

Fig. 5 Histograms describing 500 trajectories generated by the RL policy (blue) and constrained optimization (red) by: (a) average position deviation and (b) total maneuver magnitude.

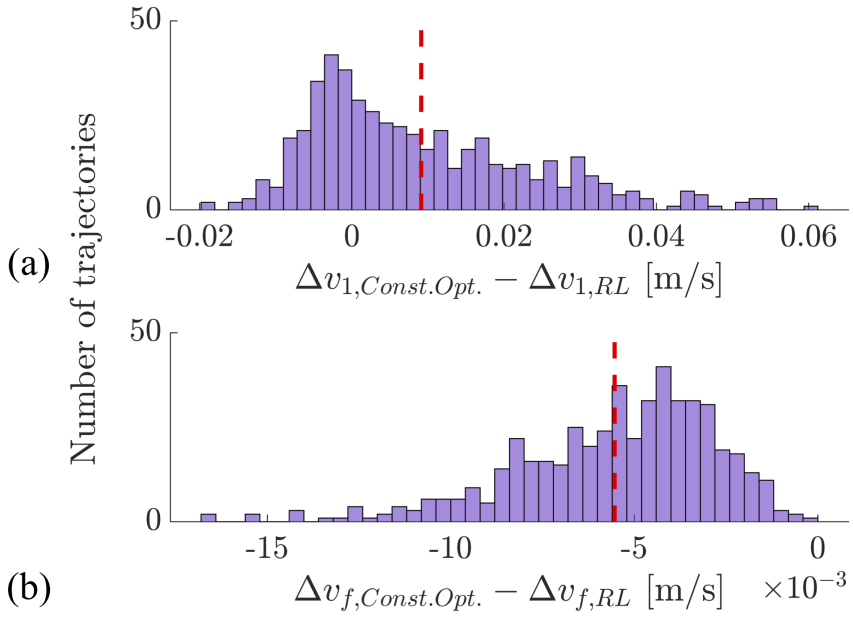

Fig. 6 Histograms summarizing the magnitude of the difference between the (a) first and (b) final maneuvers generated by the RL policy and constrained optimization.

generated by a traditional approach at the expense of slightly higher position deviations. Of course, constrained and safe RL methods could be incorporated in future work to directly constrain the position deviation and enforce any other requirements. Nevertheless, the trained policy appears to offer alternative approach to producing useful and reasonable solutions to the station-keeping maneuver design problem.

C. Applying the Trained Policy

The policy trained using transfer learning is evaluated to design station-keeping maneuvers for a spacecraft operating near the selected L_2 quasi-halo trajectory in the ephemeris model for 10.33 years. During evaluation, regular momentum unloads are applied to perturb the path of the spacecraft, necessitating the use of station-keeping maneuvers. After each momentum unload cycle, a maneuver is generated by evaluating the policy to produce the first maneuver from the τ-maneuver sequence, i.e., with $\tilde{s} = 1$. This approach is inspired by station-keeping strategies that select individual maneuvers according to the properties of the subsequent trajectories generated for one or more revolutions around the reference path [46, 47]. However, evaluating the policy with $\tilde{s} = 1$ corresponds to designing each individual maneuver to produce a subsequent controlled trajectory, spanning the τ-maneuver sequence, that would remain bounded within the vicinity of reference and cumulatively minimize control effort if there were no momentum unloads or other perturbations.

In this application of the trained policy, a single initial state for the spacecraft is randomly generated by applying a perturbation from the reference path at an epoch of June 24, 2021, 09:42:02 UTC. The station-keeping maneuvers generated by evaluating the trained policy every 21.67 days for 10.33 years produce a controlled trajectory that is

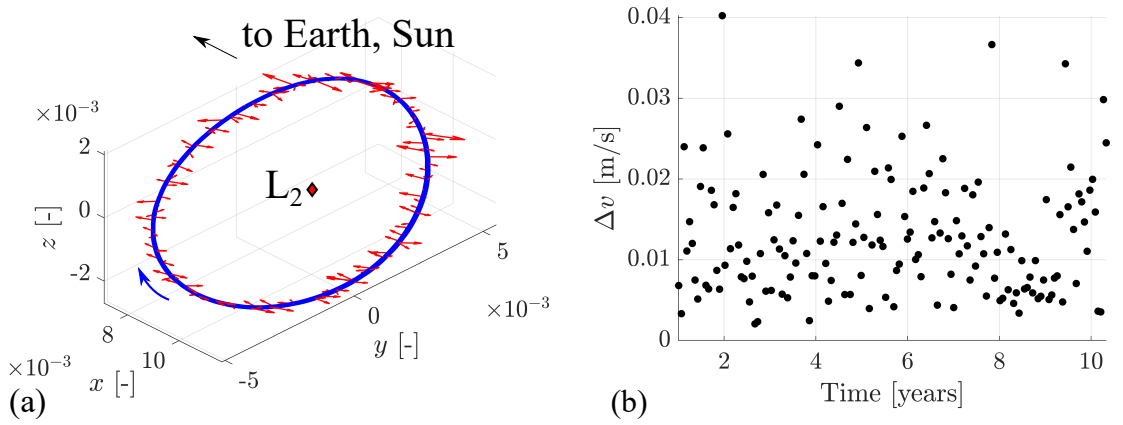

Fig. 7 Station-keeping maneuvers in the ephemeris model with regular momentum unloads over 10.33 years: (a) single controlled trajectory and (b) associated maneuver magnitudes.

displayed in blue in Figure 7(a) in the pulsating Sun-Earth rotating frame with dimensionless coordinates relative to the Earth. In this figure, the direction and relative magnitudes of the impulsive station-keeping maneuvers are reflected by the red arrows. Figure 7(b) displays the associated maneuver magnitudes. The 175 impulsive station-keeping maneuvers that occur over 10.33 years require a total maneuver magnitude of $\Delta v_{tot} = 2.285$ m/s; this total maneuver magnitude is comparable to the 2.22 m/s over approximately 10 years as calculated by Bosanac et. al for a similar spacecraft and reference trajectory using constrained optimization [1]. In addition, Figure 8 displays the time history of the magnitudes of the deviations in (a) position and (b) velocity between the reference and controlled trajectories. At the initial epoch, the spacecraft is located outside of the desired 100 km boundary from the reference trajectory. However, the controlled trajectory rarely exceeds the desired 100 km distance threshold or 0.05 m/s in velocity from the reference trajectory at the end of each momentum unload cycle after the first 0.5 years. These results indicate that trained policy successfully generates station-keeping maneuvers for a spacecraft to remain near the reference Sun-Earth L_2 quasi-halo trajectory in the ephemeris model with a low total maneuver magnitude, even as momentum unloads regularly perturb the path.

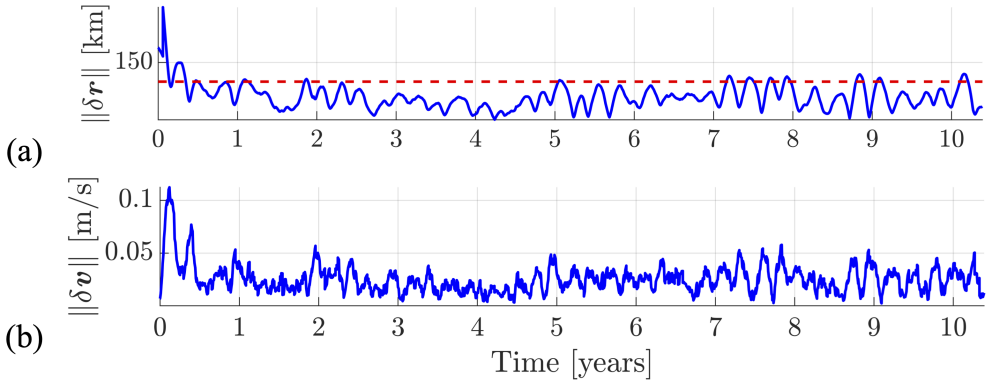

Fig. 8 Time history of magnitude of position (a) and velocity (b) deviations of the controlled trajectory in Figure 7, measured relative to the reference quasi-halo trajectory.

Although outside the scope of this paper, these results supply a foundation for investigating RL-based maneuver design in more complex models of a mission scenario by incorporating safety and other constraints into the policy, incorporating uncertainty during training, achieving more complex maneuvering goals, and using a wider variety of reference paths.

VI. Conclusions

A policy that is approximated by a feed-forward neural network can be trained using reinforcement learning to design station-keeping maneuvers for a spacecraft operating near a Sun-Earth L_2 southern quasi-halo trajectory in an ephemeris model. Training via proximal policy optimization and guided parameter selection via Bayesian optimization can result in a policy that produces a high value in a deterministic model. In addition, the use of transfer learning across models of increasing fidelity can reduce the computational time for this training process in approximately half while producing similar results to training the policy solely in the ephemeris model. The controlled trajectories generated by the trained policy require low maneuver magnitudes and remain within close proximity of the reference quasi-halo trajectory, with similar characteristics to the solutions generated via constrained optimization.

VII. Acknowledgment

This work was supported by an Early Stage Innovations grant from NASA's Space Technology Research Grants Program, under NASA grant 80NSSC19K0222. The third author acknowledges support from a NASA Space Technology Research Fellowship. The authors also thank the anonymous reviewers for their feedback. An earlier version of this paper was presented as AAS Paper 21-216 at the virtual AAS/AIAA Space Flight Mechanics Meeting in February 2021. The work in this paper also appears in the first author's Ph.D. dissertation titled "Incorporating Machine Learning into Trajectory Design Strategies in Multi-Body Systems," submitted in April 2022 to the University of Colorado Boulder.

References

[1] Bosanac, N., Webster, C., Howell, K., and Folta, D., "Trajectory Design for the Wide Field Infrared Survey Telescope Mission," *Journal of Guidance, Control and Dynamics*, Vol. 42, No. 9, September 2019, pp. 1899 – 1911. https://doi.org/doi:10.2514/1.G004179.

[2] Folta, D., and Beckman, M., "Libration Orbit Mission Design: Applications of Numerical and Dynamical Analysis," *Libration Point Orbits and Applications*, Aiguablava, Spain, June 2002.

[3] Simó, C., Gómez, G., Llibre, J., Martinez, R., and Rodriguez, L., "On the Optimal Station Keeping Control of Halo Orbits," *Acta Astronautica*, Vol. 15, No. 6, 1987, pp. 391–397. https://doi.org/doi:10.1016/0094-5765(87)90175-5

[4] Sutton, R. S., and Barto, A. G., *Reinforcement learning: An introduction*, MIT Press, 2018, Chaps. 1, 2, 3, 13.

[5] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A., Veness, J., Bellemare, M., Graves, A., Riedmiller, M., Fidjeland, A., 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, pp. 529–33. https://doi.org/doi:10.1038/nature14236.

[6] Mnih, V., Badia, A., Mirza, M., Graves, A., Harley, T., Lillicrap, T. P., Silver, D., and Kavukcuoglu, K., "Asynchronous Methods for Deep Reinforcement Learning," *Proceedings of The 33rd International Conference on Machine Learning*, New York, NY, 2016, pp. 1928–1937.

[7] Silver, D., Huang, A., Maddison, C., Gueza, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D., "Mastering the Game of Go with Deep Neural Networks and Tree Search," *Nature*, Vol. 529, 2016, pp. 484–489. https://doi.org/doi:10.1038/nature16961.

[8] Das-Stuart, A., Howell, K., and Folta, D., "Rapid Trajectory Design in Complex Environments Enabled by Reinforcement Learning and Graph Search Strategies," *Acta Astronautica*, Vol. 171, 2020, pp. 172–195. https://doi.org/10.1016/j.actaastro.2019.04.037.

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

[10] Sullivan, C. J., and Bosanac, N., "Using Multi-Objective Deep Reinforcement Learning to Uncover a Pareto Front in Multi-Body Trajectory Design," *AAS/AIAA Astrodynamics Specialist Virtual Conference*, August 9-13, 2020.

[11] LaFarge, N., Miller, D., Howell, K., and Linares, R., "Guidance for Closed-Loop Transfers using Reinforcement Learning with Application to Libration Point Orbits," *Acta Astronautica*, Vol. Volume 186, September 2021, pp. 1–23. https://doi.org/10.1016/j.actaastro.2021.05.014.

[12] 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*, January, 2019.

[13] Boone, S., Bonasera, S., McMahon, J., Bosanac, N., and Ahmed, N., "Incorporating Observation Uncertainty into Reinforcement Learning-Based Spacecraft Guidance Schemes," *AIAA Scitech 2022 Forum*, San Diego, CA, 2022. https://doi.org/10.2514/6.2022-1765.

[14] Bonasera, S., Elliott, I., Sullivan, C. J., Bosanac, N., McMahon, J., and Ahmed, N., "Designing Impulsive Station-Keeping Maneuvers Near a Sun-Earth L2 Halo Orbit via Reinforcement Learning," *AAS/AIAA Space Flight Mechanics Virtual Meeting*, February 2021.

[15] Bosanac, N., Bonasera, S., Sullivan, C. J., McMahon, J., and Ahmed, N., "Reinforcement Learning for Reconfiguration Maneuver Design in Multi-Body Systems," *AAS/AIAA Astrodynamics Specialist Virtual Conference*, August 2021.

[16] Guzzetti, D., "Reinforcement Learning and Topology of Orbit Manifolds for Stationkeeping of Unstable Symmetric Periodic Orbits," *AAS/AIAA Astrodynamics Specialist Conference, Portland, ME*, August 2019.

[17] Molnar, A., "Hybrid Station-Keeping Controller Design Leveraging Floquet Mode and Reinforcement Learning Approaches," Master's thesis, School of Aeronautics & Astronautics, Purdue University, West Lafayette, IN, December 2020.

[18] LaFarge, N., Howell, K., and Folta, D. C., "An Autonomous Stationkeeping Strategy for Multi-Body Orbits Leveraging Reinforcement Learning," *AIAA SciTech Forum*, San Diego, CA, 2022. https://doi.org/10.2514/6.2022-1764.

[19] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," *arXiv prePrint arXiv:1912.01703*, 2019.

[20] Taylor, M. E., and Stone, P., "Transfer Learning for Reinforcement Learning Domains: A Survey," *Journal of Machine Learning Research*, Vol. 10, No. 7, 2009.

[21] Szebehely, V., *Theory of Orbits: The Restricted Problem of Three Bodies*, Academic Press, London, UK, 1967, Chaps. 1, 5.

[22] "SPICE Toolkit," https://naif.jpl.nasa.gov/naif/aboutspice.html, 2016. Accessed December 2020.

[23] Pavlak, T., "Trajectory Design and Orbit Maintenance Strategies in Multi-body Dynamical Regimes," Ph.D. thesis, School of Aeronautics & Astronautics, Purdue University, West Lafayette, IN, May 2013.

[24] Hornik, K., Stinchcombe, M., and White, H., "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, Vol. 2, No. 5, 1989, pp. 359–366. https://doi.org/10.1016/0893-6080(89)90020-8.

[25] Nwankpa, C. E., Ijomah, W., Gachagan, A., and Marshall, S., "Activation Functions: Comparison of Trends in Practice and Research for Deep Learning," *arXiv:1811.03378v1*, 2018.

[26] Hecht-Nielsen, R., "Theory of the Backpropagation Neural Network," *Neural Networks for Perception*, Elsevier, 1992, pp. 65–93. https://doi.org/10.1016/B978-0-12-741252-8.50010-8.

[27] Bertsekas, D. P., *Reinforcement learning and optimal control*, Athena Scientific, Belmont, MA, 2019, Chap. 3.

[28] Sewak, M., *Deep Reinforcement Learning - Frontiers of Artificial Intelligence*, Springer, Singapore, 2019, Chap. 11.

[29] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O., "Proximal Policy Optimization Algorithms," *arXiv:1707.06347*, 2017.

[30] Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D., "Deep Reinforcement Learning that Matters," *Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA*, 2018.

[31] Andrychowicz, M., Raichuk, A., Stańczyk, P., Orsini, M., Girgin, S., Marinier, R., Hussenot, L., Geist, M., Pietquin, O., Michalski, M., Gelly, S., and Bachem, O., "What Matters in On-Policy Reinforcement Learning? A Large-Scale Empirical Study," *Google Research, Brain Team*, 2020. ArXiv:2006.05990.

[32] Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P., "High-Dimensional Continuous Control Using Generalized Advantage Estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[33] Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P., "Trust Region Policy Optimization," *International Conference on Machine Learning*, 2015, pp. 1889–1897.

[34] Zhang, Z., Luo, X., Liu, T., Xie, S., Wang, J., Wang, W., Li, Y., and Peng, Y., "Proximal Policy Optimization with Mixed Distributed Training," *2019 IEEE 31st International Conference on Tools with Artificial Intelligence*, 2019, pp. 1452–1456.

[35] Holubar, M., and Wiering, M., "Continuous-Action Reinforcement Learning for Playing Racing Games: Comparing SPG to PPO," *arXiv:2001.05270*, 2020.

[36] Loshchilov, I., and Hutter, F., "Decoupled Weight Decay Regularization," *International Conference on Learning Representations*, 2019.

[37] Young, M., Hinkle, J., Kannan, R., and Ramanathan, A., "Distributed Bayesian optimization of deep reinforcement learning algorithms," *Journal of Parallel and Distributed Computing*, Vol. 139, May 2020, pp. 43–52. https://doi.org/10.1016/j.jpdc.2019.07.008.

[38] Jomaa, H. S., Grabocka, J., and Schmidt-Thieme, L., "Hyp-RL : Hyperparameter Optimization by Reinforcement Learning," *arXiv:1906.11527v1*, 2019.

[39] Folta, D., Pavlak, T., Haapala, A., Howell, K., and Woodard, M., "Earth-Moon Libration Point Orbit Stationkeeping: Theory, Modeling, and Operations," *Acta Astronautica*, Vol. 91, No. 1, 2014, pp. 421–433. https://doi.org/10.1016/j.actaastro.2013.01.022.

[40] Dichmann, D., Alberding, C., and Yu, W., "Stationkeeping Monte Carlo Simulation for the James Webb Space Telescope," *International Symposium on Space Flight Dynamics*, 2014.

[41] Ng, A. Y., Harada, D., and Russell, S., "Policy invariance under reward transformations: Theory and application to reward shaping," *Proceedings of the 16th International Conference on Machine Learning (ICML'99)*, 1999, pp. 278—287.

[42] Zavoli, A., and Federici, L., "Reinforcement Learning for Robust Trajectory Design of Interplanetary Missions," *Journal of Guidance, Control, and Dynamics*, Vol. 44(8), August 2021. https://doi.org/10.2514/1.G005794.

[43] Brochu, E., Cora, V., and de Freitas, N., "A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning," *University of British Columbia, Department of Computer Science*, December 2010. Http://arXiv.org/abs/1012.2599.

[44] Nogueira, F., "Bayesian Optimization: Open source constrained global optimization tool for Python," , 2014. URL https://github.com/fmfn/BayesianOptimization, accessed November 2020.

[45] MathWorks, "MATLAB," , 2021. Natick, MA, USA.

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

[47] T.A. Pavlak, K.C. Howell, "Strategy for Optimal, Long-Term Stationkeeping of Libration Point Orbits in the Earth-Moon System," *AIAA/AAS Astrodynamics Specialist Conference*, Minneapolis, MN, August 13-16, 2012. https://doi.org/10.2514/6.2012-4665