Low-Thrust Trajectory Design in Multi-Body Systems via Multi-Objective Reinforcement Learning

by

Christopher J. Sullivan

B.S., University of Texas at Austin, 2017

M.S., University of Colorado Boulder, 2019

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Aerospace Engineering Sciences

2022

Committee Members:

Natasha Bosanac, Chair

Nisar Ahmed

Rodney Anderson

Alinda Mashiku

Jorge Poveda

Daniel Scheeres

Sullivan, Christopher J. (Ph.D., Aerospace Engineering Sciences)

Low-Thrust Trajectory Design in Multi-Body Systems via Multi-Objective Reinforcement Learning

Thesis directed by Prof. Natasha Bosanac

As the number of missions in multi-body gravitational environments continues to increase, new trajectory design techniques that enable autonomous maneuver and trajectory design may decrease the effort required of human trajectory designers. This benefit may be particularly apparent with the increased usage of low-thrust propulsion systems and exploration of chaotic regimes with limited dynamical insight to guide the trajectory designer. Exploring the multi-objective trade space for a trajectory design scenario provides insights to trajectory designers, scientists, and mission designers during preliminary mission concept development. One approach, multi-objective reinforcement learning, enables the autonomous generation of optimal low-thrust trajectories with varying geometries, flight times, and propellant mass usages in multi-body environments. A multi-objective reinforcement learning algorithm, designated Multi-Reward Proximal Policy Optimization (MRPPO), is introduced to simultaneously train multiple policies with unique objective prioritizations to maximize distinct regions of the multi-objective solution space. To implement this approach, this dissertation presents a formulation of the reinforcement learning problem that reflects the trajectory design problem. This approach is applied to three trajectory design scenarios to develop low-thrust transfers that balance flight time and propellant mass usage and thereby provide insights into the multi-objective solution spaces for each scenario. The results are then compared with the solutions recovered via a nonlinear, constrained, gradient descent optimization scheme and Proximal Policy Optimization (PPO) to validate the performance of MRPPO. Compared to PPO, MRPPO consistently develops policies with higher total rewards and develops similar solutions to the transfers produced by the optimization scheme.

Dedication

To my parents. Thank you for your never-ending, unconditional support, and raising me to always pursue my dreams.

To Linh. Thank you for your endless love and support. This is just the conclusion of one of the many adventures we will go on together.

Acknowledgements

I would first love to express my gratitude towards my family for their support throughout my years at CU. Linh, Miso, Mom, Dad, Colleen, Dan, Caroline, and everyone else, none of this would have been possible without your encouragement, laughs, and love. Next, I want to thank my advisor, Dr. Natasha Bosanac, for her constant support, feedback, and willingness to challenge me to be a better researcher. You have taught me so much about the research process, astrodynamics, and exploring new research ideas I would have never considered before CU. I would also like to thank my fellow lab members, Ian, Thomas, Stefano, Kristen, Renee, and Giuliana for asking questions, giving constructive feedback on this research, and for creating such a fun lab environment these past few years. On a similar note, Ken, Mark, Spencer, Erica, Hermann, Shota, Kieran, Lubna, and Jonathan, thank you for all of the fun and productive conversations we've had the past few years since we all started at CU. Whether they happened on a rock wall, ski slope, or in the dungeons of the engineering center, they contributed to this research and made CU feel like home. I'd also like to thank Ben, Dan, Andrew, Luke, Don, Alex, CK, and Felix for showing me the ropes and slopes and welcoming me as a member of a brand new lab into the department. Outside of CU, I cannot express enough gratitude towards my NASA advisors Dr. Alinda Mashiku, Dr. Rodney Anderson, and Dr. Jeff Stuart for facilitating my work with NASA and for providing guidance and constructive feedback throughout the years. From UT-Austin, I would like to thank Dr. Brandon Jones, Dr. Humboldt Mandell, Dr. Hans Mark, and Dr. Lori Magruder for providing avenues for me to perform research as an undergraduate student. Finally, I would like to acknowledge the NASA Space Technology Research Fellowship (Grant 80NSSC19K1147) for funding this work.

Contents

Chapter

Tables

Table

Figures

Figure

Chapter 1

Background and Motivation

Missions operating in the chaotic regimes of multi-body gravitational environments have long been at the forefront of NASA's goal to better humanity's scientific understanding of the universe. Further, with the significant increase in access to space provided by the private sector of the aerospace industry, entities such as research laboratories, private companies, and universities have begun exploring mission concepts outside of the near-Earth vicinity. The destinations for these missions include the cislunar regime, Sun-Earth equilibrium points, and further regions where the gravitational influence from multiple celestial bodies is significant. However, a major challenge with designing trajectories within these systems is that even with insights gained from dynamical systems theory into the underlying dynamics of the system, scenarios exist where these is limited intuition into the solution space. Additionally, once a single solution is identified, a common challenge is sufficiently exploring nearby solutions of varying geometries and characteristics. Finally, current methods for trajectory design in multi-body environments are often effort intensive for a human trajectory designer, necessitating that new techniques be developed to enable autonomous exploration of the solution space.

Multi-objective optimization is typically used to explore a solution space with competing objectives and constraints. Such problems exist in astrodynamics when designing feasible thrust-enabled trajectories for a spacecraft in multi-body systems. For example, there is a well-known trade-off between minimizing the flight time, minimizing propellant mass usage, minimizing changes in the thrust direction, and satisfying additional objectives. These multi-objective optimization

problems are often studied via analysis of the multi-objective solution space, and in particular, the nondominated solutions that maximize two or more objectives in an environment. In trajectory design problems, generating a single point solution may be time and resource intensive; as a result, generating the global solution space, much less a local solution space, may be impractical.

In multi-body systems, one step towards reducing the complexity of trajectory design is to use the Circular Restricted Three-Body Problem (CR3BP) for preliminary analysis. The CR3BP closely approximates the dynamics of three-body systems, such as the Sun-Earth and Earth-Moon systems, yet it is chaotic and exhibits a large variety of fundamental solutions that tend to be retained in higher-fidelity models. By leveraging insights from the CR3BP into the motion of a spacecraft within the system, solutions may be constructed in the CR3BP by an expert trajectory designer and then transferred to a higher-fidelity point mass ephemeris model using numerical techniques such as multiple shooting or collocation. However, with additional continuous or significant constraints, this approach can be difficult to automate rendering it insufficient when needing rapid trajectory or maneuver development.

Machine learning, a field of artificial intelligence, offers one solution for enabling autonomous trajectory and maneuver design for spacecraft without requiring extensive input from a human trajectory designer. In fact, machine learning has already revolutionized many industries. One of the goals of machine learning is to map high-dimensional relationships between inputs and outputs and uncover optimal behavior that may be nonintuitive to a human designer. This goal may be leveraged within the trajectory design process to uncover the high-dimensional relationships between the dynamics, spacecraft, constraints, and objectives for a trajectory design scenario.

Reinforcement learning is a distinct class of machine learning algorithms that employ one or more agents to interact with an unknown dynamical environment to develop optimal solutions that satisfy a single or multi-objective optimization problem. Specifically, reinforcement learning algorithms are used to construct a policy that maps states to optimal actions within the environment that aim to maximize the long-term reward generated for a single trajectory. Additionally, multiple policies may be trained that maximize unique objective prioritizations corresponding to uncovering

distinct regions of the multi-objective solution space. Policies trained using reinforcement learning may autonomously recover nonintuitive solutions within the solution space while also reducing the effort required of a human trajectory designer. Reinforcement learning algorithms that train neural networks to learn optimal behavior within an environment have been successfully applied across a variety of industries, most notably achieving acclaim in Atari games and Go, demonstrating the applicability and robustness of these algorithms across multiple disciplines [98, 129].

Multi-objective reinforcement learning algorithms construct multiple optimal solutions simultaneously to develop a subset of solutions within the multi-objective solution space. In this dissertation, a multi-objective reinforcement learning algorithm designated Multi-Reward Proximal Policy Optimization (MRPPO) is constructed and used to recover a subset of the multi-objective solution space for low-thrust trajectory design within a multi-body system. This research synthesizes the recent advancements in reinforcement learning algorithms with trajectory design techniques to create a framework for autonomously uncovering solutions to a multi-objective optimization problem. This approach is applied to three trajectory design scenarios to recover a multitude of low-thrust transfers that distinctly balance propellant mass usage and flight time while targeting the final periodic orbit. Additionally, the solutions are approximately validated and compared to solutions produced by Proximal Policy Optimization (PPO) and a nonlinear constrained optimization scheme. This approach demonstrates the applicability of using a multi-objective reinforcement learning methodology to autonomously recover optimal solutions within a low-thrust, multi-body trajectory design scenario.

1.1 Motion in Multi-Body Gravitational Environments

Throughout history, motion in a multi-body gravitational environment has been studied, explored, and used to develop innovative mathematical and scientific observations. Many of these observations have been formed in the context of a three-body gravitational system. In a three-body system, the three bodies gravitationally interact with one another and influencing the path of the bodies within the system. Three-body systems are inherently chaotic, generally possess no closed

form analytical solution, and are valuable approximations for the motion of a spacecraft in the Earth-Moon system, Sun-Earth system, and other regions of the solar system.

Many researchers have focused on the three-body problem to develop insights into the motion of the bodies in the system throughout time. This insight has been used as a foundation for trajectory design centuries later. Notable mathematicians studying three-body systems include both Euler and Lagrange who explored the three-body problem leading to the discovery of periodic solutions and equilibrium points under certain assumptions about the system [11, 148]. Jacobi followed on this work by formulating an integral of motion for the system [74]. The mathematical insights provided by these investigations have been used as a valuable foundation for mission design in multi-body systems providing humanity with new scientific knowledge about the universe.

Many past, current, and future missions require spacecraft to operate in chaotic regimes where multiple celestial bodies are significantly influencing the trajectory. For instance, the International Sun-Earth Explorer (ISEE-3), was the first spacecraft to enter into a three-body orbit in the vicinity of the Sun-Earth L_1 equilibrium point [117]. Other notable missions that have or will operate in three-body dynamical regimes include Lunar IceCube, the Solar and Heliophysics Observatory (SOHO), and Near-Earth Asteroid (NEA) Scout in addition to numerous other missions [26, 118, 95]. Most recently, the James Webb Space Telescope (JWST) was inserted into a Lissajous orbit at the Sun-Earth L_2 equilibrium point [88]. Outside of the Earth vicinity, numerous mission concepts have been designed for missions to the Sun-Earth L_5 equilibrium point including proposals by Akioka et al., Schmidt and Bothmer, and Vourlidas due to its favorable properties for heliophysics observations [4, 123, 160]. These missions demonstrate the interest in developing new and innovative mission concepts that leverage multi-body dynamics to enable new scientific observations.

1.2 Low-Thrust Trajectory Design Methods

A variety of techniques based on insights from dynamical systems theory have been developed to construct low-thrust-enabled transfers in multi-body systems. For instance, Anderson and Lo explored propellant-optimal, low-thrust transfers and found that the low-thrust transfers often

follow invariant manifold arcs in a three-body system [5]. Both Bosanac et al. and Ozimek and Howell develop low-thrust transfers using invariant manifold arcs and dynamical systems theory in the cislunar regime [26, 108]. More recently, Cox, Howell, and Folta and Prado Pino employed dynamical structures within a low-thrust CR3BP dynamical model and insights from dynamical systems theory to develop low-thrust transfers [36, 114]. These techniques often use insights from a human trajectory designer to recover low-thrust transfers that leverage the natural flow of motion within a system.

Low-thrust trajectory design in multi-body environments is a challenging and complex problem due to the need to balance achieving the objectives while also satisfying the constraints with often limited dynamical insight into the solution space. This problem is compounded when developing solutions for SmallSats, which pose additional constraints on the available solution space. Trajectory designers must contend with the additional challenges that the SmallSat form factor imposes including: low maneuverability due to a limited propulsive capability, constrained deployment conditions, and operational scheduling constraints due to limited power. These challenges all impact the availability and geometry of feasible transfer paths that tend to exist in a high-dimensional design space.

1.3 Low-Thrust Trajectory Optimization

Various optimization methods have been used in the astrodynamics community to recover optimal solutions to a low-thrust, multi-body trajectory design problem. These problems are characterized by their high dimensionality, sensitivity due to the chaotic environment, and in some cases, limited dynamical insight. Previous investigations have focused on using both traditional optimization and heuristic methods to recover solutions in the multi-objective solution space [15]. Many investigations have implemented these methods in the two-body problem, however, this dissertation focuses on transfers in a multi-body environment. Traditional optimization schemes applied to develop transfers in trajectory design scenarios may be broadly categorized into three types of approaches: 1) indirect optimization methods, 2) direct optimization methods, and 3)

hybrid optimization methods [137]. However, state-of-the-art traditional optimization methods often require an initial guess in low-thrust, multi-body trajectory design scenarios to converge on an optimal solution. Generating this initial guess may be time-consuming for a human trajectory designer and may strongly influence the local basin of convergence that the optimal solution falls within [164]. Conversely, heuristic methods, such as evolutionary algorithms, may struggle to find optimal solutions and are challenging to properly design [77].

1.3.1 Nonlinear Constrained Optimization Methods

Indirect optimization methods leverage calculus of variations to define a two-point boundary value problem (2PBVP) and recover the optimality conditions [15]. This approach has been used to generate both low-thrust and impulsive optimal trajectories. Investigations in the astrodynamics community that incorporated indirect optimization include Colasurdo and Pastrone employing indirect optimization for minimum fuel impulsive transfers, Pérez-Palau and Epenoy applying indirect optimization for developing low-thrust transfers in the Sun-Earth-Moon bicircular four-body problem, Taheri, Kolmanovsky, and Atkins investigating smoothing functions in an indirect optimization scheme for low-thrust interplanetary transfers, and Tang and Jiang using indirect optimization for capturing near-Earth objects [34, 113, 149, 150]. Indirect optimization methods generally require less computational time and possess low-dimensionality, but possess less robust convergence properties and require defining non-intuitive costates [115].

Direct methods use nonlinear programming to solve the optimal control problem by discretizing the trajectory into a sequence of arcs that may be iteratively updated. A number of investigations have applied direct optimization to solve optimization problems in the astrodynamics community. These approaches include an investigation by Falck and Dankanich using direct optimization to compute low-thrust spiral trajectories, Graham and Rao applying direct optimization to develop low-thrust transfer between Earth orbits, and Scheel and Conway employing direct optimization for multi-revolution, low-thrust trajectories [50, 56, 121]. In distinct approaches, Aziz, Scheeres, and Lantione and Boone and McMahon employed differential dynamic programming to

develop optimal controls schemes for low-thrust transfers [8, 21, 22]. Direct optimization methods are generally more robust to initial guess selection and do not require defining costates but possess higher-dimensionality and lower accuracy compared to indirect optimization methods [115].

Hybrid optimization methods combine elements from indirect and direct optimization methods to recover locally optimal trajectories. In a hybrid optimization approach, indirect procedures may be combined with direct methods to retain low-dimensionality while increasing the robustness of the convergence process. Insights from calculus of variations may be used to define optimal engine operation and determine the values of the control parameters along a trajectory prior to using a gradient-based optimization to recover an optimal solution. Within the astrodynamics community, hybrid optimization has been used by Stuart, Ozimek, and Howell for designing low-thrust transfers between halo orbits using invariant manifolds as an initial guess [139]. Sullivan et al. followed this investigation by applying this method for low-thrust transfers to high-inclination science orbits in the Sun-Earth system [144]. By combining the advantages from the direct and indirect approaches, solutions may be developed using optimization techniques that are robust to the initial guess.

For each approach, the optimization scheme develops an optimal solution, which may be a local or global optima, with respect to a defined objective function by updating the parameters of the initial guess. This objective function captures the objectives and goals of the trajectory optimization problem, and generally must be continuous and possess continuous first order derivatives. Additionally, boundary and path constraints may be introduced that the optimization scheme must satisfy when developing the optimal solution. Both equality and inequality constraints may be incorporated, but the constraints generally must also be continuous and have continuous first order derivatives. When constraints are introduced, the problem is redefined as a constrained optimization problem. Finally, nonlinear optimization methods refer to optimization methods that recover optimal solutions when at least one objective or constraint is a smooth, nonlinear function. Nonlinear constrained optimization methods may possess a direct, indirect, or hybrid procedures, recover local or global optima, and possess single or multi-objective formulations.

1.3.2 Heuristic Methods

Heuristic methods are a distinct class of methods from classical optimization methods that enable the generation of optimal spacecraft trajectories. Well-known heuristic methods include evolutionary algorithms, genetic algorithms, ant colony optimization, artificial bee colony optimization, and particle swarm optimization. Many of these algorithms are inspired by the behavior of biological organisms evolving throughout time in response to feedback from an environment and may also be considered swarm-based optimization methods. In a swarm-based optimization approach, a population of candidates, i.e. trajectories, are evaluated, evolved, and combined over many generations or iterations until an optimal solution is recovered or the population is no longer significantly changing. Often, characteristics from members of the population that generate a higher value in the environment are incorporated into the weaker members of the population to boost their value and ensure the population continues to explore the environment. Heuristic methods may be used to solve both unconstrained and constrained nonlinear optimization problems such as those seen in the astrodynamics community.

Previous investigations in the astrodynamics community have recovered both impulsive and low-thrust trajectories using a variety of heuristic methods. Abraham, Spencer, and Hart employed particle swarm optimization to explore the design space for a low-thrust transfer from an Earth orbit to an Earth-Moon libration point orbit [1]. Gurfil and Kasdin constructed out-of-ecliptic trajectories in the Sun-Earth CR3BP using deterministic crowding genetic algorithms [60]. Finally, an investigation by Cage, Braun, and Kroo developed interplanetary spacecraft trajectories using a genetic algorithm approach [28]. These investigations demonstrated that heuristic methods may be used to recover optimal solutions in trajectory design scenarios, but more recent investigations have employed evolutionary algorithms.

Evolutionary algorithms, a type of heuristic method, possess many parallels with reinforcement learning algorithms. In both evolutionary and reinforcement learning algorithms, a population of candidate solutions may explore an unknown dynamical environment and use feedback from the

environment and objectives to update the population. In fact, neuroevolutionary algorithms, which combine neural networks and evolutionary algorithms, possess similar structures and goals to both evolutionary and reinforcement learning algorithms [73]. Neuroevolutionary algorithms update a population composed of neural networks, each with distinct weights and biases, by fusing, evolving, and perturbing the neural networks throughout training to recover an optimal solution in the environment that maximizes a set of objectives. However, while neuroevolutionary algorithms do not require computing the gradients of the neural network parameters for updates, thus eliminating a costly computational time requirement, they often do not perform as well as reinforcement learning algorithms in complex environments [82]. Neuroevolutionary algorithm have been applied by a number of investigations within the astrodynamics community. Dachwald trained evolutionary neurocontrollers to generate low-thrust transfers to a near-Earth asteroid and solar sail transfers to a variety of interplanetary destinations [37, 38]. Carnelli, Dachwald, and Vasile also applied evolutionary neurocontrollers to develop low-thrust interplanetary transfers incorporating gravity assists [29]. Additionally, Huber applied NeuroEvolution Augmenting Topologies (NEAT) to develop interplanetary trajectories [71].

Alternatively, heuristic methods may be combined with optimization methods to leverage the benefits of both approaches and recover optimal solutions. For instance, Stuart, Howell, and Wilson combined ant colony optimization and hybrid optimization to explore the design space for an asteroid tour while minimizing propellant mass usage [138]. Often, these approaches are combined to explore and uncover solutions within a multi-objective solution space.

1.3.3 Multi-Objective Optimization

Multi-objective optimization algorithms aim to generate multiple optimal solutions within a multi-objective solution space that distinctly balance two or more objectives. A nondominated optimal solution is recovered if a solution balances two or more objectives without being dominated by another solution, i.e. a solution cannot improve a single objective without decreasing a different objective. Generating multiple, global, nondominated, optimal solutions enables the recovery of a

Pareto front for a multi-objective optimization problem [157]. However, since guaranteeing recovery of global nondominating solutions, rather than just local optima, along the entire Pareto front in an unknown, complex solution space is challenging, in this research, the goal is limited to recovering multiple locally optimal solutions within a multi-objective solution space that may potentially lie along or approximate the Pareto front in a trajectory design scenario.

In a trajectory design scenario, common objectives that are balanced in a multi-objective optimization problem include propellant mass usage, flight time, uncertainty, scientific observations, and other mission objectives. Often, there is an inverse relationship between flight time and propellant mass usage where longer flight times correspond to lower propellant mass usages and lower flight times require higher propellant mass usages. Additionally, for preliminary mission concept development, uncovering the relationship between flight time and propellant mass usage for a trajectory design scenario enables insights into the trade space for mission designers and scientists. There are numerous methods for recovering solutions that span the multi-objective solution space. One approach is to perform many sequential runs of a single objective optimization method while varying the weighting of objectives in the objective function [31]. Alternatively, a multi-objective optimization approach may generate multiple optimal solutions that span a multi-objective solution space. Traditional multi-objective optimization approaches to spacecraft trajectory design may be categorized as classical optimization schemes, heuristic methods, or a combination of the two.

Classical multi-objective optimization schemes have been applied to a variety of trajectory design scenarios balancing distinct objectives and may incorporate direct, indirect, or hybrid optimization algorithms. For instance, Russell explored multi-revolution, low-thrust transfers via indirect optimization in the Jupiter–Europa and Earth–moon restricted three body problems [120]. Dellnitz et al. employed a three-step multi-objective optimization process with dynamical systems theory and direct optimization to generate transfers from Earth to an Earth-Moon L_2 halo orbit that balance flight time and total ΔV [45]. Jenson and Scheeres applied an indirect multi-objective optimization method to develop orbit transfers and phasing maneuvers at Bennu balancing minimizing propellant mass usage and uncertainty [75].

Multi-objective optimization approaches that combine optimization methods and heuristic methods have also been applied to a variety of trajectory design scenarios and are used within NASA software packages such as Evolutionary Mission Trajectory Generator (EMTG) to discover distinct local minima in distinct regions of the solution space [102]. Often, the heuristic method is used in an outer loop to enable a population of candidates to explore the multi-objective solution space while the optimization method composes the inner loop to optimize each candidate with respect to a defined objective function [136]. Notable investigations in the astrodynamics community using this combination of methods include approaches by Lee et al. which combine genetic algorithms with a Lyapunov feedback control law to design low-thrust transfers balancing propellant mass usage and flight time [85] Similar investigations by Coverstone-Carroll, Hartmann, and Mason and Hartmann, Coverstone-Carroll, and Williams synthesize genetic algorithms with a calculus of variations optimization techniques to produce a multi-objective solution space for interplanetary transfers [35, 64]. Other approaches employ Non-dominated Sorting Genetic Algorithm II (NSGA-II), a multi-objective optimization algorithm that combines a sequential quadratic programming (SQP) optimization algorithm for the inner loop with a genetic algorithm for the outer loop, to construct transfers spanning a multi-objective solution space [43]. Deb et al. uncovered the multi-objective solution space for interplanetary trajectories using NSGA-II [42]. Englander, Vavrina, and Ghosh used NSGA-II in EMTG to generate insights into the flight time and propellant mass usage trade space for a main-belt asteroid tour [49]. Finally, Shah, Beeson, and Coverstone recovered the flight time and ΔV trade space for transfers to the Earth-Moon L_2 equilibrium point [128]. While these methods may be used to recover transfers in chaotic gravitational environments spanning a multi-objective solution space, the algorithms often require an initial guess or dynamical knowledge input from a human expert to converge on solutions in complex environments. Additionally, genetic algorithms, including NSGA-II, often require tuning non-intuitive parameters to recover a diverse set of solutions for a multi-objective optimization problem [49, 77].

1.4 Reinforcement Learning

Machine learning has been a rapidly growing field since its renaissance in the 2000's due to increases in computation power and the derivation of more efficient, robust algorithms. Machine learning techniques have been described as a disruptive technology with the ability to reduce cost, lower design time, and reduce human biases in the design process in a variety of fields [78]. While there exists a large variety of machine learning algorithms, they may be broadly classified into three categories: supervised learning, unsupervised learning, and reinforcement learning. Each classification has advantages and disadvantages depending on the problem the algorithm is being applied to. Then, the design engineer must make an informed choice on which method to apply.

Reinforcement learning is a field of machine learning that enables the recovery of optimal behavior in dynamical environments without a priori domain knowledge. Reinforcement learning algorithms use an agent or multiple agents to autonomously interact with an environment and generate data to develop solutions that satisfy a single or multi-objective optimization problem. First explored in the 1950's and 1960's, the theory behind reinforcement learning has developed alongside the increases in computational power [97]. As computational power has increased, new techniques have been developed and applied to numerous applications. However, one of the most significant steps that has led to a renewal of interest in reinforcement learning was the incorporation of neural networks in combination with traditional reinforcement learning methods. This development facilitated the recovery of control schemes for Atari games that surpassed human experts [98]. Since neural networks have been incorporated, reinforcement learning has been applied to autonomous vehicles, Go, robotics, and a variety of applications and industries [72, 96, 129, 159]. However, traditional reinforcement learning approaches typically focus on recovering a single policy maximizing a single objective necessitating multiple sequential runs to recover solutions spanning a multi-objective solution space.

1.4.1 Applications of Reinforcement Learning to Trajectory Design

Several researchers have applied reinforcement learning for multi-body trajectory and maneuver design. For instance, Das-Stuart, Howell, and Folta used reinforcement learning to construct low-thrust transfers by exploring and combining arcs computed from periodic orbit families in the CR3BP [40]. Miller and Linares applied Proximal Policy Optimization (PPO) to develop low-thrust transfers between distant retrograde orbits in the Earth-Moon CR3BP [96]. LaFarge et al. expanded on this work by using PPO to generate low-thrust transfers between distinct libration point orbits following a pre-computed, continuous solution constructed by a human designer [84]. Additionally, LaFarge, Howell, and Folta and Scorsoglio et al. employed reinforcement learning for targeting periodic orbits in the Earth-Moon CR3BP [83, 127]. Federici et al. developed low-thrust transfers between distinct libration point orbits in the Earth-Moon CR3BP and using a sparse reward function formulation that only depends on the final state along a trajectory [51]. Bonasera et al. and Bosanac et al. constructed impulsive maneuvers using PPO for station-keeping and reconfiguration scenarios in the Sun-Earth CR3BP [19, 25]. Boone et al. built upon this work by incorporating uncertainty measures into the state vector for a reinforcement learning environment to explore the influence of the uncertainty on the robustness of a controls scheme for an impulsive orbit transfer scenario [20].

1.4.2 Multi-Objective Reinforcement Learning

In multi-objective reinforcement learning, algorithms train multiple policies simultaneously distinctly balancing multiple objectives to uncover optimal solutions spanning a multi-objective solution space. Researchers have used a variety of multi-objective reinforcement learning approaches, with and without neural networks, to train policies that recover solutions spanning a multi-objective optimization problem. Multi-objective reinforcement learning approaches are generally more efficient compared to single objective approaches while developing the multi-objective solution space and provide deeper insights into the trade-offs between the objectives. A variety of multi-objective

reinforcement learning algorithms have been developed using distinct frameworks, neural network and non-neural network implementations, algorithm characteristics, and for numerous applications.

These multi-objective reinforcement learning approaches are broadly assigned into two categories: (1) the reward function for each policy returns a scalar sum composed of terms representing each objective and (2) the reward function for each policy returns a vector where each component directly represents an objective. The first type of approach assumes that the multi-objective space may be described by multiple scalar reward functions formed using shared reward function components that are regulated using linearly-related, scalar weights. This approach generally results in a less complex formulation and retains the ability to recover nondominated solutions. Well-known approaches have focused on using Q-learning in a multi-objective framework to develop solutions for problems including water management, the deep sea treasure problem, electrical power management, and molecule optimization [30, 100, 152, 158, 167].

Alternative approaches have specified the reward function as a vector valued function. These approaches also tend to use Q-learning in combination with a vector-valued reward function to construct solutions spanning a multi-objective solution space in problems including resource gathering, clinical trials, the network routing problem, and the deep sea treasure problem [10, 91, 103, 105]. While both types of approaches may generate nondominated solutions within the global solution space, the presented research falls within the first category of multi-objective reinforcement learning algorithms whereby the reward function outputs a scalar.

1.5 Path Planning Approaches

Path planning may be incorporated into a reinforcement learning algorithm to aid exploration in a complex environment and recover optimal solutions. Within the reinforcement learning community, prior investigations in a variety of path planning applications have noted the inability of state-of-the-art reinforcement learning algorithms to converge on locally optimal trajectories in complex environments without either a priori knowledge defined by a domain expert or the incorporation of path planning algorithms. This phenomenon has been noted by researchers including

Panov, Yakovlev, and Suvorov, Spandan et al., and Trott et al. [109, 135, 155]. While domain knowledge may be incorporated into state-of-the-art reinforcement learning algorithms as demonstrated by Silver et al., this process requires time and effort from a human expert [129]. Instead, path planning approaches may be incorporated into a reinforcement learning algorithm to improve the convergence behavior and performance of the policies.

Reinforcement investigations that leverage path planning have incorporated a variety of structures to enable a policy to recover optimal behavior. For instance, Trinh, Vu, and Kimura applied PPO with a point-of-conflict path planning prediction model for pedestrian path planning [154]. Zhang et al. employed geometric reinforcement learning for path planning of uncrewed aerial vehicles (UAVs) [165]. Guo et al. combined reinforcement learning and an artificial potential field to develop paths for uncrewed ships [59]. Lastly, Park, Kim, and Song developed a motion planning methodology for a robot by leveraging probabilistic roadmaps and reinforcement learning [110].

Other investigations focused on using reinforcement learning for path planning incorporate waypoints enabling a policy to follow a set of waypoints towards an objective whereby the waypoints may be fixed or iteratively adjusted by the reinforcement learning algorithm or a path planning algorithm. Ebrahimi et al., Li et al., and Yan, Xiang, and Wang applied reinforcement learning for designing paths for UAVs using waypoints [47, 87, 163]. Within the astrodynamics community, Gaudet and Furfaro employed reinforcement learning for spacecraft hovering at waypoints surrounding an asteroid [54]. In a distinct investigation, Furfaro and Linares combined a Zero-Effort-Miss/Zero-Effort-Velocity (ZEM/ZEV) feedback guidance law with waypoints determined via reinforcement learning for spacecraft entry trajectories for Mars [53]. However, the waypoints incorporated for reinforcement learning algorithms to follow are generally points in position space that are defined a priori or autonomously generated without consideration for the natural flow of motion within the dynamical model.

1.6 Dissertation Overview

This dissertation focuses on using to autonomously recover subsets of the multi-objective solution space for low-thrust trajectory design in multi-body environments. First, a multi-objective reinforcement learning algorithm is formulated using a state-of-the-art single objective reinforcement learning algorithm as a foundation. Then, the trajectory design problem is translated into a reinforcement learning problem to enable the multi-objective reinforcement learning algorithm to recover a subset of solutions spanning the multi-objective solution space. A nonlinear constrained optimization scheme is also formulated to validate the low-thrust transfers produced by the multi-objective reinforcement learning approach. These methodologies are applied to three trajectory design scenarios of varying complexity to demonstrate the applicability of the multi-objective reinforcement learning approach and generate insight into the performance of the approaches.

1.6.1 Contributions

The contributions of this research to the technical community are summarized as:

- A multi-objective reinforcement learning algorithm, designated MRPPO, is developed using PPO as a foundation to train multiple policies simultaneously. MRPPO is empirically shown to be more stable in the explored trajectory design scenarios and recover solutions more efficiently than PPO thereby reducing the required computational time and effort.

- A reinforcement learning problem is formulated to reflect a low-thrust trajectory design problem in a multi-body system. This formulation incorporates a path planning approach, inspired by the concept of waypoints, to enable the policies to autonomously develop low-thrust transfers to a periodic orbit without requiring an initial guess.

- MRPPO is used to solve the formulated reinforcement learning problem in three trajectory design scenarios. A subset of the multi-objective solution space is therefore generated and analyzed to provide insights into the propellant mass usage and flight time trade space.

These results are compared to a nonlinear, constrained optimization scheme to generate insights into the performance of MRPPO in these trajectory design scenarios.

1.6.2 Organization

The research introduced in this investigation is structured as follows

- Chapter 2: The dynamical models that are used to model the motion of a spacecraft within a multi-body gravitational environment are derived and explored. First, the equations of motion and relevant assumptions for the CR3BP are formulated. Additionally, natural dynamical structures including equilibrium points, periodic orbits, and invariant manifolds are developed. Then, equations of motion for the higher-fidelity point mass ephemeris model and for incorporating low-thrust into the dynamical models are derived. Finally, transformations between inertial and rotating frames are formulated.

- Chapter 3: The general formulation of a reinforcement learning algorithm and classifications for existing algorithms are introduced. Then, the theory behind neural networks, actor-critic methods, and policy gradient methods is summarized. PPO synthesizes these components to train policies to recover optimal behavior in an environment, which may then be used in transfer learning to update the policies for related dynamical environments and objectives.

- Chapter 4: MRPPO is developed using PPO as a foundation for training multiple policies simultaneously in a multi-objective framework. Then, parameters that influence the training process are explored. Finally, a path planning approach incorporating a continuous set of waypoints in the environment is integrated into MRPPO to aid the policies in converging on optimal solutions in complex environments.

- Chapter 5: A multi-objective low-thrust trajectory design optimization problem is formulated as a reinforcement learning problem. The agent properties, episode characteristics, state vector, action vector, time step, reward function, and termination conditions are

delineated for the three trajectory design scenarios explored in this research. Lastly, the transfer learning implementation enabling policies to be trained in a point mass ephemeris model is introduced.

- Chapter 6: To validate the results of MRPPO, the multi-objective reinforcement learning problem is translated into a multi-objective optimization problem that is solved using a nonlinear constrained optimization scheme. The free variables, objectives, constraints, and initial guess construction are defined, and differences between the results produced by the optimization scheme and MRPPO due to non-identical implementations are noted.

- Chapter 7: The results produced by MRPPO and the optimization scheme are presented and explored for three trajectory design scenarios. Additionally, an analysis comparing the results produced by PPO and MRPPO for the first trajectory design scenario is included. Finally, an investigation into the influence of the hyperparameter selection and the number of reference trajectories per policy is presented.

- Chapter 8: A summary of the methodologies and results of this investigation is presented and followed by recommendations for future work in the areas of reinforcement learning, multi-objective reinforcement learning, and applying reinforcement learning approaches for low-thrust trajectory design in multi-body systems.

Chapter 2

Dynamical Models

Dynamical models of varying fidelity are often leveraged to generate an initial guess for a transfer prior to transitioning the solution to a more complex ephemeris model. First, the equations of motion for the CR3BP, which approximates the motion of a spacecraft in the vicinity of two larger primary bodies are presented. Additionally, the CR3BP admits natural dynamical structures that are approximately retained in higher-fidelity models [48]. Then, locally optimal solutions in the CR3BP often serve as valuable initial guesses for recovering locally optimal transfers in a point mass ephemeris model [23]. An example of a higher-fidelity point mass ephemeris model is presented in this section. In this chapter, the CR3BP and point mass ephemeris equations of motion for natural trajectories are augmented to incorporate the acceleration imparted by a low-thrust engine. Finally, the transformations between the rotating and inertial frames are specified for defined state and thrust vectors.

2.1 Circular Restricted Three-Body Problem

The motion of a spacecraft in multi-body regime may be modeled using the CR3BP. The CR3BP models the paths of three bodies, denoted as P_1, P_2, and P_3, with masses $\tilde{M}_1$, $\tilde{M}_2$, and $\tilde{M}_3$, respectively, orbiting a common barycenter, represented as O. In this derivation, tilde notation denotes dimensional quantities and variables with overhead bars indicate vector quantities. In this model, the third body is assumed to have negligible mass in comparison to the other bodies such that $\tilde{M}_3 \approx 0$, and the motion of the third body does not influence the motion of the two larger

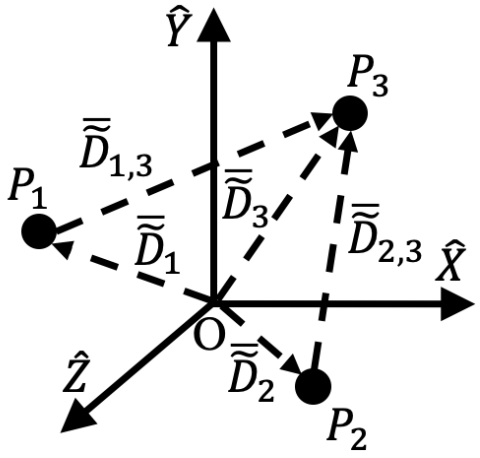


Figure 2.1: Inertial coordinate frame with axes $(\hat{X}, \hat{Y}, \hat{Z})$ centered at the barycenter O with bodies P_1, P_2, and P_3.

primary bodies nor affect the location of the barycenter. Additionally, both primary bodies are assumed to be point masses with constant mass where P_1 is assumed to possess a larger mass than P_2, i.e. $\tilde{M}_1 > \tilde{M}_2$. Then, P_1 and P_2 may be treated as an isolated two-body problem and are assumed to follow coplanar, circular orbits about their mutual barycenter.

Using these assumptions, an inertial frame may be constructed with orthogonal axes $(\hat{X}, \hat{Y}, \hat{Z})$ that is centered upon the system barycenter. Both $\hat{X}$ and $\hat{Y}$ are defined in the orbital plane of the primary bodies where the $\hat{Y}$ is set perpendicular to $\hat{X}$. Then, $\hat{Z}$ completes the right-handed triad and directed out of the orbital plane and aligned with the angular momentum vector of bodies P_1 and P_2. Figure 2.1 depicts an inertial frame defined for bodies P_1, P_2, and P_3. The position and velocity of P_3 in the inertial frame are designated $\tilde{\bar{D}}_3 = [\tilde{X}_3, \tilde{Y}_3, \tilde{Z}_3]^T$ and $\tilde{\bar{D}}'_3 = [\tilde{X}'_3, \tilde{Y}'_3, \tilde{Z}'_3]^T$, respectively. In the inertial frame, $\tilde{\bar{D}}_i$ denotes the vector directed from the origin of the inertial frame to each body i and is written as $\tilde{\bar{D}}_i = \tilde{X}_i\hat{X} + \tilde{Y}_i\hat{Y} + \tilde{Z}_i\hat{Z}$. The relative vectors from body i to body j are expressed as $\tilde{\bar{D}}_{i,j} = (\tilde{X}_j - \tilde{X}_i)\hat{X} + (\tilde{Y}_j - \tilde{Y}_i)\hat{Y} + (\tilde{Z}_j - \tilde{Z}_i)\hat{Z}$.

In this conservative, multi-body system, the only force acting upon the objects is their mutual gravitational attractions. Then, the force per unit mass vector acting on P_3 is written as $\tilde{\bar{F}}_3 = \bar{\nabla}_3\tilde{U}_3 = \tilde{\bar{D}}''_3$ where $\tilde{U}_3$ is the potential function describing the gravitational forces of P_1 and P_2 acting

on P_3. The potential function is composed of two terms which are the gravitational potentials of bodies P_1 and P_2 measured with respect to P_3. The potential function is computed via $\tilde{U}_3 = (\tilde{G}\tilde{M}_1)/(\tilde{D}_{1,3}) + (\tilde{G}\tilde{M}_2)/(\tilde{D}_{2,3})$ where $\tilde{G}$ is the universal gravitational constant defined as $\tilde{G} = 6.67408 \times 10^{-11}$ m^3 kg^{-1} s^{-2}. Additionally, the magnitudes of the relative vectors, $\tilde{D}_{13}$ and $\tilde{D}_{23}$, are computed via $\tilde{D}_{i,3} = \sqrt{(\tilde{X}_3 - \tilde{X}_i)^2 + (\tilde{Y}_3 - \tilde{Y}_i)^2 + (\tilde{Z}_3 - \tilde{Z}_i)^2}$ where i denotes primaries P_1 and P_2, respectively. Then, the partial derivatives of $\tilde{U}_3$ with respect to $\tilde{\bar{D}}_3$ are written as

$$\frac{\partial \tilde{U}_3}{\partial \tilde{X}_3} = \tilde{X}_3'' = \frac{-\tilde{G}\tilde{M}_1(\tilde{X}_3 - \tilde{X}_1)}{\tilde{D}_{1,3}^3} - \frac{\tilde{G}\tilde{M}_2(\tilde{X}_3 - \tilde{X}_2)}{\tilde{D}_{2,3}^3} \tag{2.1}$$

$$\frac{\partial \tilde{U}_3}{\partial \tilde{Y}_3} = \tilde{Y}_3'' = \frac{-\tilde{G}\tilde{M}_1(\tilde{Y}_3 - \tilde{Y}_1)}{\tilde{D}_{1,3}^3} - \frac{\tilde{G}\tilde{M}_2(\tilde{Y}_3 - \tilde{Y}_2)}{\tilde{D}_{2,3}^3} \tag{2.2}$$

$$\frac{\partial \tilde{U}_3}{\partial \tilde{Z}_3} = \tilde{Z}_3'' = \frac{-\tilde{G}\tilde{M}_1(\tilde{Z}_3 - \tilde{Z}_1)}{\tilde{D}_{1,3}^3} - \frac{\tilde{G}\tilde{M}_2(\tilde{Z}_3 - \tilde{Z}_2)}{\tilde{D}_{2,3}^3} \tag{2.3}$$

This formulation produces eighteen equations of motion for the three bodies. However, only 10 integrals of motion exist: six from the conservation of linear momentum, three from the conservation of angular momentum, and one from the conservation of energy. Instead, to reduce the dimensionality of the equations of motion, the position and velocity of the bodies are nondimensionalized using the system's characteristic quantities and the assumptions of the CR3BP are employed.

For the CR3BP, a rotating coordinate system may be conveniently defined using the assumption that the two primary bodies possess circular orbits about a common barycenter. The rotating coordinate system is centered at the system barycenter O with axes $(\hat{x}, \hat{y}, \hat{z})$: the $\hat{x}$ axis is directed from P_1 to P_2, the $\hat{z}$ axis is directed along the orbital angular momentum vectors of the two primaries, and the $\hat{y}$ axis is defined to create a right-handed coordinate frame. In this rotating frame, the primaries are held fixed at these locations at $\tilde{\bar{\rho}}_1 = [\tilde{x}_1, \tilde{y}_1, \tilde{z}_1]^T$ and $\tilde{\bar{\rho}}_2 = [\tilde{x}_2, \tilde{y}_2, \tilde{z}_2]^T$ for P_1 and P_2, respectively, with position vectors measured with respect to P_3 denoted as $\tilde{\bar{d}}_1$ and $\tilde{\bar{d}}_2$, respectively. Figure 2.2 depicts the CR3BP rotating coordinate system defined using P_1 and P_2. This rotating frame admits a set of dimensional autonomous coordinates for P_3 denoted $\tilde{\bar{d}}_3 = \tilde{\bar{d}} = [\tilde{x}, \tilde{y}, \tilde{z}]^T$ and $\tilde{\bar{d}}_3' = \tilde{\bar{d}}' = [\tilde{x}', \tilde{y}', \tilde{z}']^T$ for the position and velocity components, respectively.

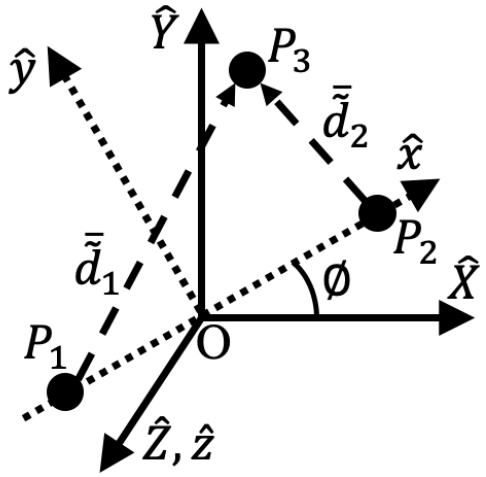


Figure 2.2: Rotating coordinate frame with axes $(\hat{x}, \hat{y}, \hat{z})$ centered at the barycenter O with bodies P_1, P_2, and P_3.

This formulation enables the conversion between inertial and rotating coordinates to be defined as

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{bmatrix} = \begin{bmatrix} \cos(\phi) & \sin(\phi) & 0 \\ -\sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{X} \\ \tilde{Y} \\ \tilde{Z} \end{bmatrix} \tag{2.4}$$

where $\phi = \tilde{n}\tilde{t}$ is the angle of rotation and $\tilde{n}$ is the mean motion.

Then, the characteristic quantities of a multi-body system are defined to nondimensionalize the position, velocity, time, and mass components. First, the characteristic length, $\tilde{\ell}^*$, is set equal to the distance between P_1 and P_2 and written as $\tilde{\ell}^* = \tilde{\rho}_1 + \tilde{\rho}_2$. Then, the mass of the primaries is nondimensionalized using the characteristic mass of the system computed via $\tilde{m}^* = \tilde{M}_1 + \tilde{M}_2$. The characteristic mass is also used to determine the mass ratio of the system via $\mu = \tilde{M}_2/\tilde{m}^*$ such that mass ratios approaching 0.5 signify primary bodies with near equal mass. The characteristic time is defined to set the mean motion of the system to equal unity via $n = \tilde{n}t^*$, equivalent to setting the orbital period of the primaries equal to 2π. Then, the characteristic time is computed via $\tilde{t}^* = \sqrt{(\tilde{\ell}^*)^3/(\tilde{G}\tilde{m}^*)}$. Table 2.1 lists the characteristic quantities used for the Earth-Moon and Sun-Earth CR3BP, the two systems explored in this investigation.

Using the characteristic quantities, Eqs. (2.1) - (2.3) may be updated to develop equations

Table 2.1: Characteristic quantities of the Earth-Moon and Sun-Earth CR3BP.

Characteristic Quantity	Earth-Moon	Sun-Earth
Length ($\tilde{\ell}^*$), km	3.8440×10^5	1.4960×10^8
Time ($\tilde{t}^*$), s	3.7513×10^5	5.0219×10^6
Mass ($\tilde{m}^*$), s	6.0477×10^{24}	1.9891×10^{30}
Mass Ratio (μ)	1.2151×10^{-2}	3.0035×10^{-6}

of motion in the inertial frame for nondimensional coordinates. The simplified equations of motion for body P_3 in the inertial frame using nondimensional coordinates are

$$\frac{\partial U_3}{\partial X_3} = X_3'' = -\frac{(1-\mu)(X_3 - X_1)}{D_{1,3}^3} - \frac{\mu(X_3 - X_2)}{D_{2,3}^3} \tag{2.5}$$

$$\frac{\partial U_3}{\partial Y_3} = Y_3'' = -\frac{(1-\mu)(Y_3 - Y_1)}{D_{1,3}^3} - \frac{\mu(Y_3 - Y_2)}{D_{2,3}^3} \tag{2.6}$$

$$\frac{\partial U_3}{\partial Z_3} = Z_3'' = -\frac{(1-\mu)(Z_3 - Z_1)}{D_{1,3}^3} - \frac{\mu(Z_3 - Z_2)}{D_{2,3}^3} \tag{2.7}$$

These equations of motion are associated with a nonautonomous system dependent on time. Further, by incorporating Eq. (2.11), the equations of motion defined in Eqs. (2.5) - (2.7) may be updated to only require the position and velocity of P_3. The final set of equations of motion in the nondimensional inertial frame are written as

$$X_3'' = -\frac{(1-\mu)(X_3 + \mu\cos(t))}{D_{1,3}^3} - \frac{\mu(X_3 - (1-\mu)\cos(t))}{D_{2,3}^3} \tag{2.8}$$

$$Y_3'' = -\frac{(1-\mu)(Y_3 + \mu\sin(t))}{D_{1,3}^3} - \frac{\mu(Y_3 - (1-\mu)\sin(t))}{D_{2,3}^3} \tag{2.9}$$

$$Z_3'' = -\frac{(1-\mu)Z_3}{D_{1,3}^3} - \frac{\mu Z_3}{D_{2,3}^3} \tag{2.10}$$

which are used to propagate P_3 in the inertial frame under the assumptions of the CR3BP.

This nondimensionalization scheme enables the position of the primary bodies to be written as $\bar{\rho}_1 = [-\mu, 0, 0]^T$ and $\bar{\rho}_2 = [1-\mu, 0, 0]^T$ in the rotating frame. Additionally, the nondimensional position and velocity components of P_3 are then defined as $\bar{d}_3 = \bar{d} = [x, y, z]^T$ and $\bar{d}_3' = \bar{d}' = [x', y', z']^T$,

respectively. Eq. (2.4) is then redefined for the nondimensional coordinates transformation as

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \cos(t) & \sin(t) & 0 \\ -\sin(t) & \cos(t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.11}$$

where t is the nondimensionalized time. Finally, the nondimensional position vectors of P_3 with respect to P_1 and P_2 are $\bar{d}_1 = \sqrt{(x+\mu)^2 + y^2 + z^2}$ and $\bar{d}_2 = \sqrt{(x+\mu-1)^2 + y^2 + z^2}$, respectively.

To develop the equations of motion for P_3 in the rotating frame defined by P_1 and P_2, each side of Eqs. (2.8) - (2.10) are updated to be expressed using a new set of axes. Note, derivatives with respect to time for a rotating observer will be written with the symbol ($\dot{}$) to differentiate from derivatives with respect to time in the inertial frame. First, Eqs. (2.8) - (2.10) are updated using the position of bodies P_1, P_2, and P_3 in the rotating frame and written as

$$X_3'' = -\frac{(1-\mu)(x+\mu)}{D_{1,3}^3} - \frac{\mu(x+\mu-1)}{D_{2,3}^3} \tag{2.12}$$

$$Y_3'' = -\frac{(1-\mu)y}{D_{1,3}^3} - \frac{\mu y}{D_{2,3}^3} \tag{2.13}$$

$$Z_3'' = -\frac{(1-\mu)z}{D_{1,3}^3} - \frac{\mu z}{D_{2,3}^3} \tag{2.14}$$

Then, the derivative of the position vector in the rotating coordinate system is defined as

$$\bar{d}' = \dot{\bar{d}} + {}^I\bar{\omega}^R \times \bar{d} \tag{2.15}$$

where

$${}^I\bar{\omega}^R = [0, 0, 1]^T \tag{2.16}$$

and the rotating frame derivative of the position vector is defined as

$$({}^R d\bar{d})/(dt) = \dot{x}\hat{x} + \dot{y}\hat{y} + \dot{z}\hat{z} \tag{2.17}$$

Then, the second derivative of the position vector in the inertial frame is found similarly as

$$\bar{d}'' = (\ddot{x} - 2\dot{y} - x)\hat{x} + (\ddot{y} + 2\dot{x} - y)\hat{y} + \ddot{z}\hat{z} \tag{2.18}$$

Substituting in Eq. (2.18) and rewriting the position vector in the rotating frame defined by P_1 and P_2 produces a new set of equations written as

$$\ddot{x} = 2\dot{y} + x - \frac{(1-\mu)(x+\mu)}{d_1^3} - \frac{(\mu)(x-1+\mu)}{d_2^3} \tag{2.19}$$

$$\ddot{y} = -2\dot{x} + y - \frac{(1-\mu)y}{d_1^3} - \frac{\mu y}{d_2^3} \tag{2.20}$$

$$\ddot{z} = \frac{-(1-\mu)z}{d_1^3} - \frac{\mu z}{d_2^3} \tag{2.21}$$

This autonomous system of equations may be transformed into a set of six first-order differential equations and used within a numerical integration scheme to develop solutions to the CR3BP in the rotating frame.

2.1.1 Jacobi Constant

The equations of motion of the CR3BP in the rotating frame admit a constant of integration, denoted the Jacobi constant, which reflects the relative energy between states in the CR3BP [74]. First, Eqs. (2.19) - (2.21) may be formulated in terms of the pseudo-potential function as

$$\ddot{x} - 2\dot{y} = \frac{\partial U^*}{\partial x} \tag{2.22}$$

$$\ddot{y} + 2\dot{x} = \frac{\partial U^*}{\partial y} \tag{2.23}$$

$$\ddot{z} = \frac{\partial U^*}{\partial z} \tag{2.24}$$

where the pseudo-potential function is

$$U^* = \frac{1}{2}(x^2 + y^2) + \frac{1-\mu}{d_1} + \frac{\mu}{d_2} \tag{2.25}$$

composed of the gravitational influences of P_1 and P_2 on P_3 and additional elements due to the rotation of the rotating frame. Then, the dot product of the velocity and acceleration vectors of P_3 in the rotating frame produces

$$\dot{x}\ddot{x} + \dot{y}\ddot{y} + \dot{z}\ddot{z} = \dot{x}\frac{\partial U^*}{\partial x} + \dot{y}\frac{\partial U^*}{\partial y} + \dot{z}\frac{\partial U^*}{\partial z} \tag{2.26}$$

which may be analytically integrated and written as

$$\frac{1}{2}(\dot{x}^2 + \dot{y}^2 + \dot{z}^2) = \int dU^* - \frac{dU^*}{dt} \tag{2.27}$$

The Jacobi constant exists in an autonomous system such that $dU^*/dt = 0$, and results from the constant of integration from $\int dU^*$. Then, Eq. (2.27) may be updated such that

$$\frac{1}{2}(\dot{x}^2 + \dot{y}^2 + \dot{z}^2) = U^* - C \tag{2.28}$$

Substituting in Eq. (2.25), the Jacobi constant is set equal to

$$C_J = x^2 + y^2 + \frac{2(1-\mu)}{d_1} + \frac{2\mu}{d_2} - \dot{x}^2 - \dot{y}^2 - \dot{z}^2 \tag{2.29}$$

which remains constant along natural trajectories in the CR3BP.

2.1.2 Equilibrium Points

The CR3BP admits a variety of fundamental solutions that are useful in the trajectory design process [81]. The CR3BP is known to possess five equilibrium, or Lagrange points, which are commonly denoted as L_1, L_2, and L_3 for the collinear points and L_4 and L_5 for the triangular points [156]. The locations of the equilibrium points are computed by setting velocity and acceleration terms in the CR3BP equations of motion to zero which allows Eqs. (2.19) - (2.21) to be written as

$$0 = x - \frac{(1-\mu)(x+\mu)}{d_1^3} - \frac{(\mu)(x-1+\mu)}{d_2^3} \tag{2.30}$$

$$0 = y - \frac{(1-\mu)y}{d_1^3} - \frac{\mu y}{d_2^3} \tag{2.31}$$

$$0 = \frac{-(1-\mu)z}{d_1^3} - \frac{\mu z}{d_2^3} \tag{2.32}$$

Eq. (2.32) indicates that the z position for all five points is zero requiring all five points to be located in the orbital plane of P_1 and P_2.

To compute the locations of the collinear equilibrium points, the y position is set equal to zero. Then, the x coordinate of the three collinear equilibrium points is solved according to the location

along the x-axis where each point is located. In order of increasing x position, the equilibrium points are located in the intervals

$$(x+\mu) < 0, \quad (x+\mu-1) < 0 \tag{2.33}$$

$$(x+\mu) > 0, \quad (x+\mu-1) < 0 \tag{2.34}$$

$$(x+\mu) > 0, \quad (x+\mu-1) > 0 \tag{2.35}$$

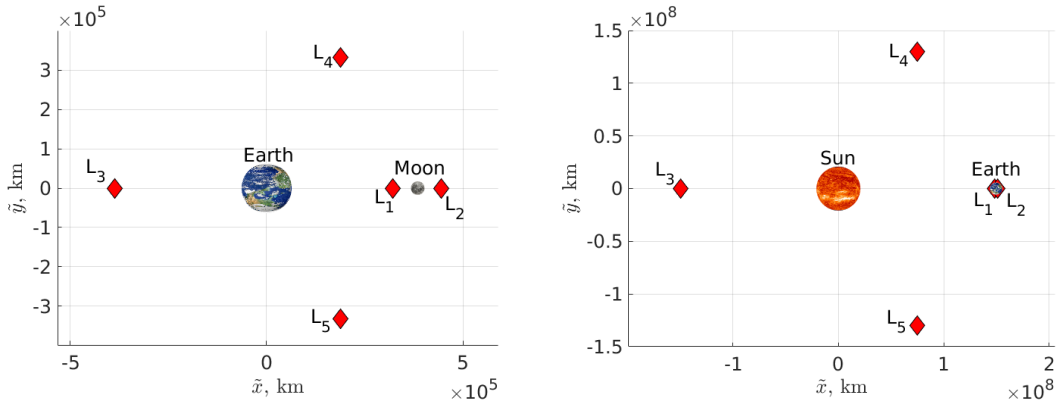
corresponding to having an x position less than both primary bodies, an x position between both primaries, and an x position greater than both primary bodies, respectively. The x coordinate of each point is solved via a numerical corrections scheme where initial guesses are placed according to the region along the x-axis where the equilibrium point is located. For mass ratios corresponding to the Earth-Moon and Sun-Earth systems, L_1, L_2, and L_3 possess stable and unstable modes indicating that motion naturally approaches or departs these equilibrium points as well as oscillatory modes corresponding to the existence of periodic orbits.

Unlike the collinear equilibrium points, the triangular equilibrium points possess a non-zero y coordinate. These two equilibrium points are known to by symmetric across the x-axis and may be computed numerically via

$$x = \frac{1}{2} - \mu \tag{2.36}$$

$$y = \pm\frac{\sqrt{3}}{2} \tag{2.37}$$

where L_4 and L_5 correspond to positive and negative y coordinates, respectively. In the Earth-Moon and Sun-Earth CR3BP, L_4 and L_5 possess oscillatory modes indicating stability and the existence of periodic orbits. Figures 2.3a and 2.3b depict the locations of the equilibrium points in the Earth-Moon CR3BP and Sun-Earth CR3BP, respectively. The stability of each equilibrium point indicates the existence of nearby periodic orbit families, which may be leveraged in the trajectory design process.



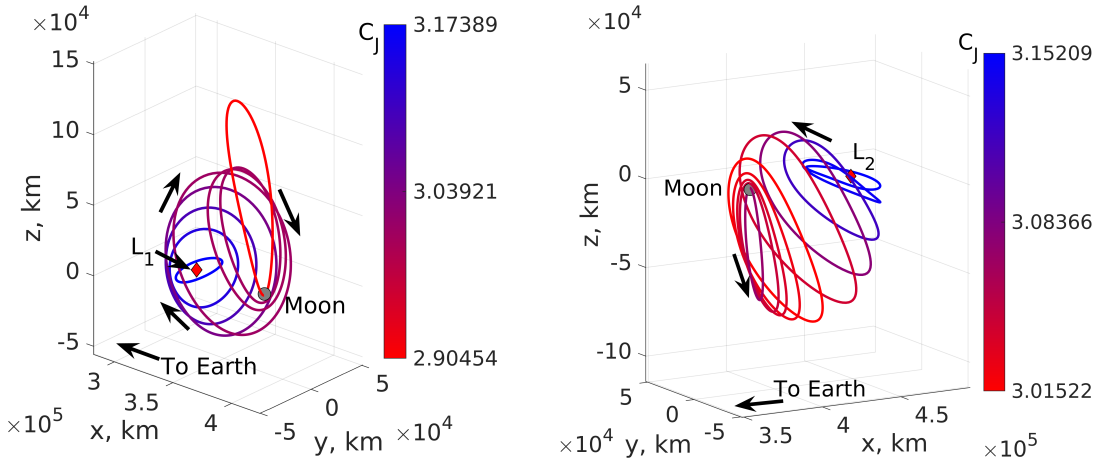
(a) Earth-Moon CR3BP (b) Sun-Earth CR3BP

Figure 2.3: Locations of equilibrium points in the Earth-Moon CR3BP and Sun-Earth CR3BP (primary bodies not to scale).

2.1.3 Periodic Orbits

In the CR3BP, periodic orbits exist as naturally repeating motions in the rotating frame defined by P_1 and P_2 that may be leveraged in the trajectory design process [23, 70]. These periodic orbits exist in continuous families with variations in Jacobi constant, geometry, and stability [57]. While periodic orbit families may emanate from each of the equilibrium points, families also exist centered at both primaries and in resonance with the motion of the secondary primary. One method for computing the members of a periodic orbit families in a multi-body system is to use a corrections scheme to compute individual orbits and a pseudo-arclength continuation method to find nearby members along the family. In this method, an initial guess for each member is corrected; then, pseudo-arclength continuation method uses the tangent of the solution curve to predict a nearby member while constraining the new member to possess a certain distance from the current member of the periodic orbit family [23]. The initial guess may be constructed using the stability information from an equilibrium point, a two-body orbit approximation, following a bifurcation from another periodic orbit family, or from initial conditions obtained from a library of periodic orbits [23]. This method is implemented using the procedure explained by Bosanac [23].

Examples of periodic orbit families in multi-body systems include the L_1 Lyapunov, L_2



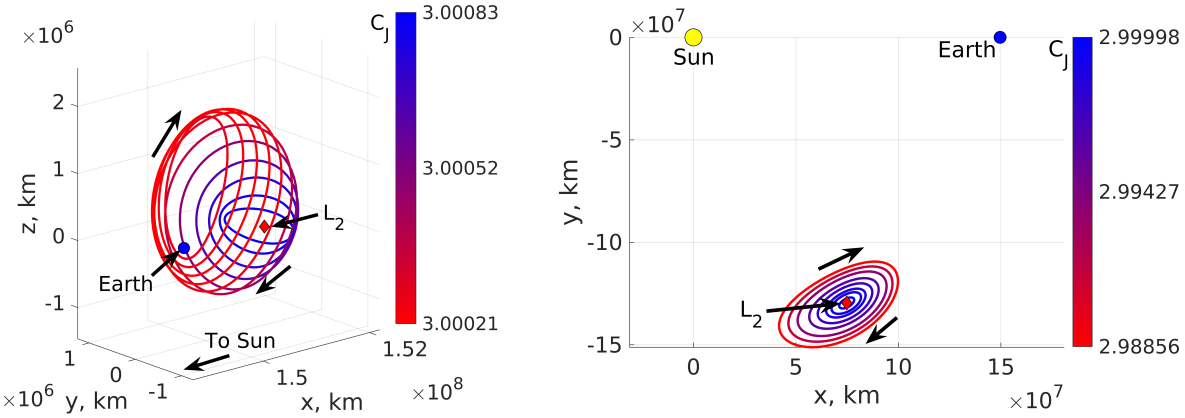
(a) L_1 northern halo (b) L_2 southern halo

Figure 2.4: Members of the Earth-Moon periodic orbit families shaded by Jacobi constant (Moon not to scale).

Lyapunov, L_1 halo, L_2 halo, and L_5 short period orbit families. The L_1 Lyapunov, L_2 Lyapunov, and L_5 short period orbit families are planar in the xy-plane of the CR3BP rotating frame while the L_1 halo and L_2 halo periodic orbit families are spatial in the configuration space. Further, the L_1 halo and L_2 halo orbit families are symmetric across the xy-plane requiring only the northern and southern segments of the family to be depicted [57]. Figure 2.4 depicts selected members of periodic orbit families in the Earth-Moon CR3BP shaded by the Jacobi constant. Similarly, Fig. 2.5 displays members from periodic orbit families generated in the Sun-Earth CR3BP shaded by Jacobi constant. Selected members of the periodic orbit families may then be used in the trajectory design process to construct low-thrust transfers throughout a multi-body system.

2.1.4 Invariant Manifolds

Periodic orbits that are unstable possess stable and unstable manifolds that guide natural motion towards and away from the orbit. Stable manifolds may be leveraged to access a periodic orbit while unstable manifolds may be used to guide a spacecraft away from a periodic orbit [81]. These manifold structures may be approximated by taking a small step in the direction of the stable



(a) L_2 northern halo

(b) L_5 short period

Figure 2.5: Members of the Sun-Earth periodic orbit families shaded by Jacobi constant (primaries not to scale).

or unstable eigenvector computed from the monodromy matrix, $M(0, P)$; the state transition matrix propagated forward in time for one orbital period. Figure 2.6 depicts a stable manifold guiding motion to an L_2 southern halo orbit in the Earth-Moon CR3BP. However, for periodic orbits without stable or unstable invariant manifolds, an additional acceleration is required in order to insert into these orbits. One method of inserting into these orbits is by using a low-thrust engine to gradually adjust the path of the spacecraft and guide a spacecraft into these stable orbits.

2.2 Point Mass Ephemeris Model

In a point mass ephemeris model, the celestial bodies are modeled to follow their actual paths, described by the ephemerides in NASA's SPICE toolkit [3]. Similar to the CR3BP, the celestial bodies are modeled as point masses and the mass of the body of interest, i.e. spacecraft, is negligible in comparison to the masses of the primary bodies. Additionally, the position, velocity, time, and mass values are nondimensionalized to aid the numerical integration scheme. The equations of motion for the point mass ephemeris model are formulated in an inertial frame, assumed to be the International Celestial Reference Frame (ICRF), with axes $(\hat{X}, \hat{Y}, \hat{Z})$. The equations of motion for a spacecraft relative to primary body P_1 in an inertial frame defined using the axes of the ICRF is

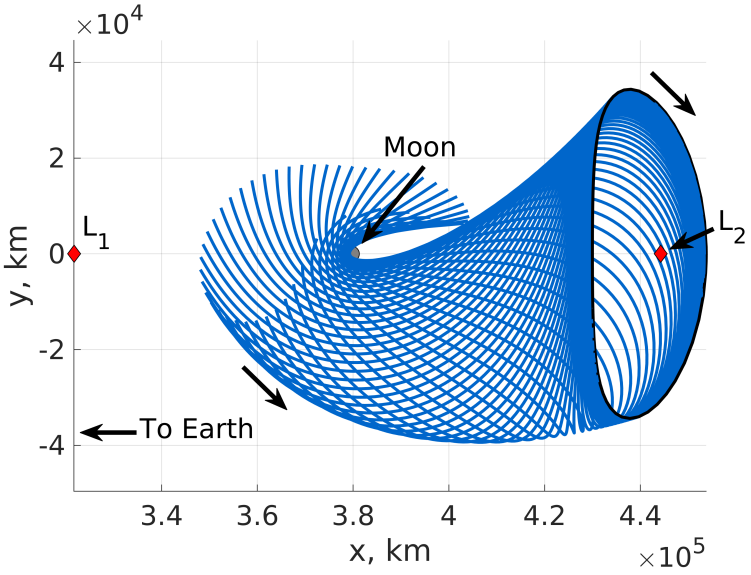


Figure 2.6: Stable manifold governing natural motion approaching an L_2 southern halo orbit in the Earth-Moon CR3BP (Moon not to scale).

written as

$$\ddot{\bar{D}}_{1,sc} = -\frac{GM_1\bar{D}_{1,sc}}{D_{1,sc}^3} + \sum_{i=2}^{N} GM_i\left(\frac{\bar{D}_{sc,i}}{D_{sc,i}^3} - \frac{\bar{D}_{1,i}}{D_{1,i}^3}\right) \tag{2.38}$$

where the 1 subscript denotes the primary influencing the motion of the spacecraft, $\bar{D}_{1,sc} = [X, Y, Z]$ is the nondimensional position of the spacecraft in the inertial frame measured with respect to the primary, and the summation incorporates the gravitational influences of N additional celestial bodies on the motion of the spacecraft. Then, the motion of a body of negligible mass may be numerically propagated in a point mass ephemeris model composed of multiple celestial bodies using the position and velocity of the bodies at defined epochs.

2.3 Incorporating Low-Thrust Acceleration into the Dynamical Models

The equations of motion for both the CR3BP and point mass ephemeris model are modified to incorporate the low-thrust acceleration imparted by a low-thrust engine and the associated decrement in the spacecraft mass. For these equations of motion, the spacecraft is assumed to possess a single low-thrust engine with a constant specific impulse. Additionally, the direction of the thrust vector output by the spacecraft is expressed in the rotating frame defined by the two

largest primaries. In the rotating frame, the nondimensional acceleration vector output by the low-thrust-enabled spacecraft is written as

$$\bar{a} = \frac{\tilde{T}\tilde{t}^{*2}}{1000\tilde{m}_{sc}\tilde{\ell}^{*}}\left(u_x\hat{x} + u_y\hat{y} + u_z\hat{z}\right) = a_x\hat{x} + a_y\hat{y} + a_z\hat{z} \tag{2.39}$$

where $\tilde{m}_{sc}$ is the wet mass of the spacecraft in kg, $\tilde{T}$ is the thrust magnitude in N, $\hat{u} = [u_x, u_y, u_z]^T$ is the thrust direction unit vector, and the multiplicative factor of 1000 is included to ensure dimensional consistency when $\tilde{\ell}^{*}$ is measured in km. Incorporating the low-thrust acceleration and propellant mass usage, the equations of motion for the low-thrust-enabled CR3BP are written as

$$\ddot{x} = 2\dot{y} + x - \frac{(1-\mu)(x+\mu)}{d_1^3} - \frac{\mu(x-1+\mu)}{d_2^3} + a_x \tag{2.40}$$

$$\ddot{y} = -2\dot{x} + y - \frac{(1-\mu)y}{d_1^3} - \frac{\mu y}{d_2^3} + a_y \tag{2.41}$$

$$\ddot{z} = \frac{-(1-\mu)z}{d_1^3} - \frac{\mu z}{d_2^3} + a_z \tag{2.42}$$

Then, the augmented state of the spacecraft in the low-thrust-enabled CR3BP is defined as $\bar{x} = [x, y, z, \dot{x}, \dot{y}, \dot{z}, m_{sc}]^T$ composed of the position, velocity, and mass quantities. Similarly, the equations of motion for the low-thrust-enabled point mass ephemeris model are written as

$$\ddot{\bar{D}}_{1,sc} = -\frac{GM_1\bar{D}_{1,sc}}{D_{1,sc}^3} + \sum_{i=2}^{N} GM_i\left(\frac{\bar{D}_{sc,i}}{D_{sc,i}^3} - \frac{\bar{D}_{1,i}}{D_{1,i}^3}\right) + {}^{I}\bar{a} \tag{2.43}$$

where ${}^{I}\bar{a}$ is the low-thrust acceleration imparted by the spacecraft expressed in the axes of the ICRF. In the inertial frame, the augmented state of the spacecraft may be defined as $\bar{X} = [X, Y, Z, X', Y', Z', m_{sc}]^T$ measured with respect to the barycenter or a primary body. Additionally, the decrement in the spacecraft mass due to propellant mass usage via the low-thrust engine is modeled via the mass flow rate equation, written as

$$\dot{m}_{sc} = \frac{-\tilde{T}\tilde{t}^{*}}{\tilde{I}_{sp}\tilde{g}_0\tilde{m}_{sc,0}} \tag{2.44}$$

where $\tilde{I}_{sp}$ is the specific impulse of the low-thrust engine in seconds and $\tilde{g}_0 = 9.81$ m/s^2 is the gravitational acceleration at the surface of the Earth [156]. This differential equation is included in a numerical integration scheme for both the low-thrust-enabled CR3BP and point mass ephemeris model equations of motion.

2.4 Transformation from Rotating to Inertial Frames

States and control vectors in an ephemeris model may be transformed from an inertial frame to a rotating frame defined by two primaries in a multi-body system [39]. One example of a transformation would be between the ICRF and the Sun-Earth rotating frame. To support the transformation at a defined epoch, the instantaneous characteristic length and time quantities, $\tilde{\ell}^\dagger$ and $\tilde{t}^\dagger$, and the instantaneous axes of the rotating frame relative to the largest primary, $(\hat{x}^*, \hat{y}^*, \hat{z}^*)$, are computed using ephemeris state information accessed via SPICE. First, $\tilde{\ell}^\dagger$ and $\tilde{t}^\dagger$ found using the instantaneous distance between P_1 and P_2 and $\tilde{t}^\dagger = \sqrt{(\tilde{\ell}^\dagger)^3/(\tilde{G}\tilde{m}^*)}$, respectively. Then, given the position and velocity of P_2 relative to P_1, the instantaneous unit vectors are defined as

$$\hat{x}^* = \frac{\bar{D}_{1,2}}{|\bar{D}_{1,2}|} \tag{2.45}$$

$$\hat{y}^* = \hat{z}^* \times \hat{x}^* \tag{2.46}$$

$$\hat{z}^* = \frac{\bar{D}_{1,2} \times \bar{D}'_{1,2}}{|\bar{D}_{1,2} \times \bar{D}'_{1,2}|} \tag{2.47}$$

where $\bar{D}_{1,2}$ denotes the position vector measured from P_1 to P_2. These instantaneous vectors are used to directly construct the rotation matrix from the instantaneous rotating frame to the inertial frame using the axes of the ICRF and written as

$$\boldsymbol{C}_{R,I} = [\hat{x}^*, \hat{y}^*, \hat{z}^*] \tag{2.48}$$

Additionally, the instantaneous angular velocity vector of the system is computed as

$$\bar{\omega} = \frac{(\bar{D}_{1,2} \times \bar{D}'_{1,2})\tilde{t}^\dagger}{(\tilde{\ell}^\dagger)^2} \tag{2.49}$$

The nondimensional state of the spacecraft in the rotating frame, $\bar{x}$, is transformed from being centered at the barycenter of the system to a primary since the ICRF and instantaneous rotating frame defined by P_1 and P_2 possess different origins. Then, $\bar{x}$ is dimensionalized using $\tilde{\ell}^\dagger$ and $\tilde{t}^\dagger$, denoted as $\tilde{\bar{x}}_{1,sc}$. The dimensional state of the spacecraft in the rotating frame relative to P_1 is then transformed to the inertial frame and denoted $\tilde{\bar{X}}_{1,sc}$. This state vector is calculated as

$$\tilde{\bar{X}}_{1,sc} = \begin{bmatrix} \boldsymbol{C}_{R,I} & \boldsymbol{0}_{3\times3} \\ \dot{\boldsymbol{C}}_{R,I} & \boldsymbol{C}_{R,I} \end{bmatrix} \tilde{\bar{x}}_{1,sc} \tag{2.50}$$

where

$$\dot{\boldsymbol{C}}_{R,I} = \begin{bmatrix} \omega C_{R,I,1,2} & -\omega C_{R,I,1,1} & 0 \\ \omega C_{R,I,2,2} & -\omega C_{R,I,2,1} & 0 \\ \omega C_{R,I,3,2} & -\omega C_{R,I,3,1} & 0 \end{bmatrix} \tag{2.51}$$

where $C_{R,I,i,j}$ denotes the (i,j) component of the $\boldsymbol{C}_{R,I}$ matrix. Finally, the dimensional inertial state of the spacecraft, $\tilde{\bar{X}}_{1,sc}$, is nondimensionalized via the CR3BP characteristic quantities, $\tilde{\ell}^*$ and $\tilde{t}^*$, to produce $\bar{X}_{1,sc}$. This transformation is also be used to convert the thrust direction from the CR3BP rotating frame to an inertial frame using the axes of the ICRF.

2.5 Transformation from Inertial to Rotating Frames

State and control vectors are also transformed from an inertial frame to the CR3BP rotating frame at a specified epoch using the CR3BP and instantaneous characteristic quantities. First, the nondimensional state vector in the inertial frame using the axes of the ICRF is converted to dimensional quantities using the CR3BP characteristic quantities, $\tilde{\ell}^*$ and $\tilde{t}^*$. Then, $\check{x}$, $\check{y}$, $\check{z}$, and $\bar{\omega}$ are computed using Eqs. 2.45 - 2.49. These quantities may be used to construct the rotation matrix and written as

$$\boldsymbol{C}_{I,R} = [\check{x}, \check{y}, \check{z}]^T \tag{2.52}$$

The dimensional state of the spacecraft in the instantaneous rotating frame defined by P_1 and P_2 is calculated via

$$\tilde{\bar{d}}_1 = \boldsymbol{C}_{I,R}\tilde{\bar{D}}_{1,sc} \tag{2.53}$$

$$\dot{\tilde{\bar{d}}}_1 = \boldsymbol{C}_{I,R}\tilde{\bar{D}}'_{1,sc} + \boldsymbol{C}_{I,R}(\tilde{\bar{\omega}} \times \tilde{\bar{D}}_{1,sc}) \tag{2.54}$$

Finally, the dimensional state of the spacecraft is transformed to be measured with respect to the barycenter of the system and nondimensionalized using the instantaneous characteristic quantities, $\tilde{\ell}^\dagger$ and $\tilde{t}^\dagger$. This process produces the nondimensional state of the spacecraft in the CR3BP rotating frame from a nondimensional inertial state using data from the NASA SPICE toolkit.

Chapter 3

Reinforcement Learning

Reinforcement learning algorithms have been of significant interest for applications where an optimization problem cannot be solved analytically and can be formulated as a Markov Decision Process (MDP). An MDP, as originally defined by Bellman, is formulated as a discrete set of time steps where state transitions between steps are governed by a probability distribution that is influenced by a control exerted at the time step [14]. The control may be selected to maximize goals within the environment. However, greedy control selections, which only maximize the immediate reward, may produce sub-optimal long-term behavior. Instead, there is often a need to consider the trade-off between the immediate reward and long-term reward generated for an MDP. Additionally, balancing the exploration vs. exploitation trade-off for an MDP requires insight into when to continue exploring to potentially find a better strategy compared to exploiting the existing, potentially sub-optimal strategy. The first explorations into solving MDP-like problems occurred during World War 2 analyzing multi-armed bandit problems, but found the problems to be intractable [55]. Latter explorations used dynamic programming, a precursor to reinforcement learning, to recursively break down a complex problem into simple problems to develop an optimal solution to the complex problem [13]. Well-known examples of dynamic programming algorithms include Dijkstra's algorithm and A* [46, 63]. However, disadvantages of dynamic programming algorithms include that they generally require the state transitions to be fully known a priori for each state in the environment. Consequently, dynamic programming algorithms suffer from the curse of dimensionality leading to exponentially growing computational requirements as the number of states

and actions in the environment increases [146]. Instead, reinforcement learning algorithms do not require a dynamical model to be known a priori to solve an MDP and may generate solutions in large state spaces without exponential memory requirements by using methods to approximate the state space.

Fundamentally, reinforcement learning algorithms possess an agent interacting with an environment, defined via a known or unknown dynamical model, taking actions that maximize the expected total reward at a given state. This returned reward is mathematically encapsulated within a scalar value function that denotes the expected long-term reward of a particular state within the state space [146]. Additionally, the set of actions that are performed throughout the state space to maximize the value function is often identified as the policy. Developing a policy that maximizes the value function requires mapping a locally optimal state-action relationship for any state within the state space.

While the overall structure of reinforcement learning problems has generally remained consistent over time, numerous methods have been introduced to solve these problems. State-of-the-art reinforcement learning methods generally model the policy using one or more neural networks, which act as universal function approximators [69]. Neural networks are able to recover the complex, high-dimensional state-action relationships within an environment much more efficiently and robustly compared to similar reinforcement learning structures, such as support vector machines and Q-tables [146]. Of particular interest in this analysis are actor-critic methods which incorporate 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 robust convergence properties [146]. Actor-critic methods separate learning the policy from learning the value function by using two distinct neural networks; the actor which models the policy and the critic which approximates the value function [80]. While actor-critic structures do not require neural networks to model the actor and critic, neural networks are used due to their robust convergence properties and ability to act as universal function approximators. Recent advancements in computational power and the development of more efficient algorithms have resulted

in robust algorithms that are capable of generating policies for a continuous action space in highly complex environments.

One approach for training neural networks in an actor-critic structure that has demonstrated strong performance in astrodynamics and other applications is PPO by Schulman et al. in combination with a stochastic gradient descent optimization scheme [126]. PPO is an on-policy, model-free, online, actor-critic reinforcement learning algorithm. Further, stochastic gradient descent optimization schemes are often used in reinforcement learning to reduce sensitivity to noisy data inputs, reduce computational times, and are well-suited for tuning the parameters in a neural network [146]. Additionally, PPO is a state-of-the-art reinforcement learning algorithm that has demonstrated strong performance and favorable convergence properties in chaotic environments [66]. PPO trains a single policy to maximize the expected total reward returned for an environment and reward function while limiting the size of updates from destabilizing the networks by enforcing a soft constraint within an objective function [126].

As an extension to reinforcement learning, multi-objective reinforcement learning methods simultaneously train multiple policies, each with a distinct reward function, by sharing environmental information across all policies. This approach generally enables a more efficient use of computational resources when recovering transfers within a region of a multi-objective solution space when compared with multiple, independent runs of a traditional reinforcement learning method [158]. PPO has been helpful for training a policy to maximize a single reward function and employed as a foundation for many related reinforcement learning algorithms. Then, based on its favorable convergence properties, PPO is a suitable candidate for developing a multi-objective reinforcement learning algorithm. A multi-objective framework may leverage the beneficial convergence properties of PPO in complex environments while also recovering solutions that span a multi-objective solution space.

Finally, transfer learning may be employed to use trained policies for one dynamical model and objective function as initial guesses for policies being trained for a more complex dynamical model or related objective function. The benefits of transfer learning include that transfer learning

significantly reduces the required training time and computational resources, increase the convergence rates of the neural networks, and produces policies in new scenarios that behave similarly to policies trained in more simple scenarios [168]. In this chapter, the framework of reinforcement learning is summarized including neural networks, actor-critic methods, policy gradient methods, PPO, and transfer learning.

3.1 Reinforcement Learning Formulation

Reinforcement learning is a type of machine learning that enables the autonomous recovery of optimal behavior that maximizes an objective function in an unknown dynamical environment. Reinforcement learning algorithms are typically composed of three elements: (1) the dynamical environment which governs state transitions and determines the reward for state-action pairs, (2) the agent which interacts with the environment and supplies the states and actions to propagate within the dynamical environment, and (3) the policy which controls the agent and determines the action to perform at a state in the environment. Figure 3.1 depicts a representation of the interplay between the three components demonstrating how states and actions are shared between the policy, agent, and environment. In this structure, the policy determines the action, $\bar{u}_t$, to perform at the state, $\bar{s}_t$, in the environment. The environment encapsulates both the dynamical model and the reward function which mathematically determines the instantaneous reward, r_t, of a state-action pair within the environment. The reward function must be formulated to capture the goals of a designer for a defined scenario [104]. Once the instantaneous reward is determined for a state-action pair, the data point is termed a state-action-reward experience. Then, the policy is updated once a set number of state-action-reward experiences have been gathered in the environment.

In many reinforcement learning implementations, multiple agents controlled by a single policy may be used in parallel to interact independently with the environment. This enables the agents to be dispersed across a server and more efficiently generate state-action-reward experiences. Ultimately, the goal of a reinforcement learning algorithm is to generate state-action-reward experiences in the environment while balancing exploration and exploitation in the environment to

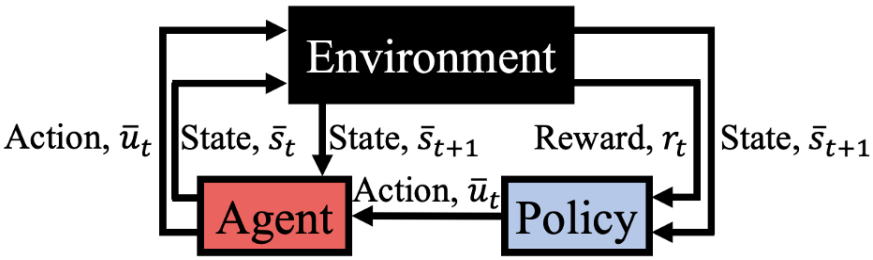


Figure 3.1: General reinforcement learning configuration including the environment, agent, and policy.

train the policy to produce optimal behavior [146]. Using this feedback loop, the policy may converge on optimal behavior for controlling one or more agents in the environment while maximizing the expected total reward throughout a trajectory.

3.2 Reinforcement Learning Classifications

Reinforcement learning algorithms may be categorized into many classifications, which provide insight into the advantages and disadvantages of the algorithms and aid a designer in determining which reinforcement learning algorithm to apply to a specific problem. Classifications of reinforcement learning algorithms including: (1) on-policy vs. off-policy, (2) model-free vs. model-based, (3) offline vs. online, and (4) policy-based vs. value-based vs. actor-critic methods [146]. Figure 3.2 depicts how a select number of reinforcement learning algorithms fall within the model-free vs. model-based classification. Additionally, an algorithm classified as being model-free may also be classified as on-policy, off-policy, online, offline, policy-based, value-based, or an actor-critic method. These overlapping classifications are valuable for generating preliminary insights into how an algorithm may behave and the advantages and disadvantages of an approach. However, the performance and behavior of reinforcement learning algorithms grouped within a single classification may vary significantly.

In on-policy algorithms, a policy is updated using the state-action-reward experiences that the policy produced within the environment [6]. Once the policy is updated, new state-action-reward

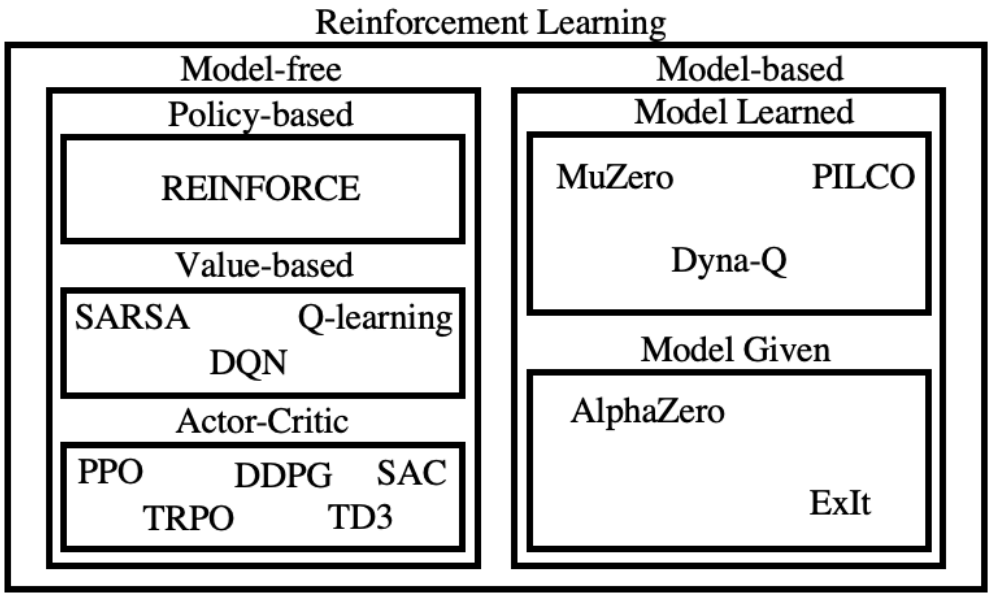


Figure 3.2: Taxonomy of selected reinforcement learning algorithms sorted by model-free vs. model-based approaches.

experiences are collected from the environment. Examples of on-policy algorithms include State-Action-Reward-State-Action (SARSA) by Rummery and Niranjan, Trust Region Policy Optimization (TRPO) formulated by Schulman et al., and PPO formulated by Schulman et al. [119, 124, 126]. On-policy algorithms are generally more stable and easier to formulate, but are often more inefficient since state-action-reward experiences are only used once to update the policy [58]. Conversely, off-policy methods may use two distinct policies to recover an optimal policy: one policy which is used to generate the data and a second policy that is improved using that data. For example, the policy interacting with the environment may be a greedy policy, i.e. the policy takes the action with the highest immediate reward, while the policy being updated may be non-greedy and attempting to maximize the total reward over an entire trajectory. Well-known off-policy algorithms include Twin Delayed Deep Deterministic Policy Gradient (TD3) devised by Fujimoto, Hoof, and Meger, Soft Actor-Critic (SAC) created by Haarnoja et al., Deep Deterministic Policy Gradient (DDPG) designed by Lillicrap et al., Deep Q-Networks (DQN) formulated by Mnih et al., and Q-learning developed by Watkins [52, 61, 89, 98, 161]. These algorithms benefit from using state-action-reward

experiences multiple times to update the policy leading to greater sample efficiency but at the cost of reduced stability guarantees and requiring more complex implementations [58].

In a model-free reinforcement learning algorithm, the policy focuses solely on maximizing the expected total reward within the environment while model-based algorithms attempt to learn both the dynamical model and the behavior that maximizes the expected total reward in that model [99]. More specifically, model-based methods predict the response output from the environment and use that prediction to formulate an action whereas model-free methods do not attempt to explicitly predict a response from the environment [99]. Notable model-based approaches include Expert Iteration (ExIt) by Anthony, Tian, and Barber, Probabilistic Inference for Learning Control (PILCO) by Deisenroth and Rasmussen, MuZero by Silver et al., AlphaZero by Silver et al., and Dyna-Q by Sutton [7, 44, 129, 130, 145]. Conversely, model-free approaches include TD3, SAC, DDPG, DQN, SARSA, TRPO, PPO, and Q-learning [52, 61, 89, 98, 119, 124, 126, 161]. Model-free approaches are generally simpler to implement and generate higher total rewards for most applications [99]. This has been theorized to be attributed to model-free methods only learning a single component in the environment, i.e. maximize the expected total reward, while model-based methods must learn both the value function and the dynamical model to a high-fidelity to be return optimal behavior [99]. However, the policies developed by model-based approaches have the potential to be more adaptable when used in a transfer learning scheme if the objectives within the environment change but the dynamics remain the same. Additionally, there are methods for reducing the complexity of model-based methods by providing the algorithm with the dynamical model, such as AlphaZero, rather than requiring the algorithm to learn an approximation of the dynamical model [130].

State-of-the-art reinforcement learning algorithms also tend to be online algorithms instead of offline algorithms. Offline algorithms use a precomputed data set to train a policy without introducing additional data throughout training [86]. Figure 3.3 illustrates how an offline reinforcement learning algorithm may interact with an environment. Initially, a policy, P_0, collects state-action-reward experiences within the environment. Once a set number of state-action-reward experiences

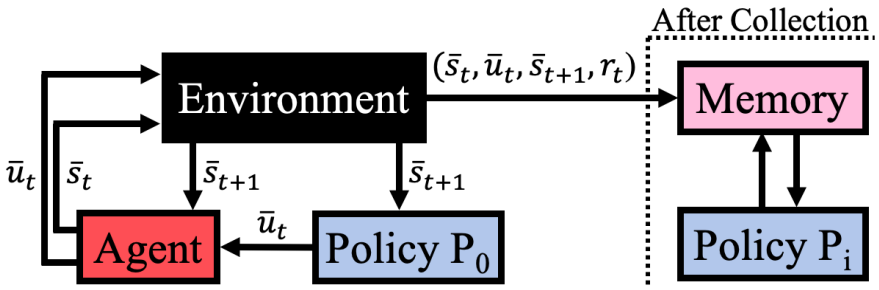


Figure 3.3: Structure of an offline reinforcement learning algorithm.

have been gathered, data collection ceases and the existing data is used to iteratively update policy P_i, to generate optimal behavior. This approach is most similar to supervised learning due to the use of an existing data set to map high-dimensional relationships between inputs and outputs. While having significant potential in applications with pre-existing data and limited insight into a complex dynamical model, a majority of offline reinforcement learning methods struggle to converge on optimal solutions without some online learning incorporated into the approach [86]. AlphaGo, a precursor to AlphaZero, used elements of offline learning to learn the basics of Go; a process that was discarded with more advanced versions of AlphaZero [129]. Conversely, online reinforcement learning methods, originally formulated by Rummery and Niranjan, allow a policy to continually interact and generate data from an environment [119]. TD3, SAC, DDPG, TRPO, PPO, AlphaZero, and Q-learning are all considered online reinforcement learning methods and follow the structure displayed in Fig. 3.1 [52, 61, 89, 124, 126, 130, 161].

Another distinction made between reinforcement learning algorithms is whether they primarily follow a policy-based, value-based, or actor-critic implementation. Policy-based methods operate on the policy function directly by using state-action-reward experiences to determine optimal actions for states in the environment [146]. These methods generally perform better in high-dimensional environments whereby a policy may be continually refined to generate optimal behavior [80]. For instance, Williams formulated REINFORCE, a policy-based reinforcement learning method, to solve simple reinforcement learning problems [162]. However, policy-based methods

are generally sample-inefficient and may experience high variance in estimating the influence of certain state-action-pairs. Conversely, value-based methods determine the value of states in the environment, and indirectly generate the policy by learning state-action transitions that produce the highest value. Value-based methods are generally more stable and more sample-efficient compared to policy-based algorithms [80]. Common value-based methods include DQN, SARSA, and Q-learning [98, 119, 161]. Additionally, while value-based methods may produce optimal policies in simple environments, policies in high-dimensional environments may fluctuate rapidly due to small changes in the approximation of the value function at each state. Actor-critic methods combine elements of both policy-based and value-based methods represented using the actor and critic, respectively, to leverage the advantages of each approach to produce stable, efficient learning in complex environments [80]. A majority of state-of-the-art reinforcement learning algorithms such as TD3, SAC, DDPG, TRPO, and PPO are primarily used with an actor-critic implementation [52, 61, 89, 124, 126].

3.3 Neural Networks

Neural networks offer one method for approximating nonlinear mappings between inputs and outputs via the construction of a web of nodes similar to the composition of neurons in a brain [69]. There are a variety of neural network structures including convolutional neural networks, recurrent neural networks, generative adversarial networks, and feed forward neural networks. However, for reinforcement learning approaches without image data input, feed forward neural networks are often leveraged to recover high-dimensional relationships between the inputs and outputs.

Feed forward neural networks accept the state of the agent into the first layer of the neural network, denoted the input layer. Then, each node in the input layer is fully connected to each node in the proceeding layer, termed the first hidden layer. Each of these connections is assigned a weight and bias, denoted $\bar{W}_i$ and $\bar{b}_i$, respectively, for the i^{th} hidden layer. The initial values assigned for the weights and biases are generated either randomly by an initialization scheme or using a method that takes in data generated by a human expert. While both approaches have

advantages and disadvantages, to reduce the time and effort required of a human designer, the initial values for the weights and biases are assigned randomly using an initialization scheme.

The nodes in the first hidden layer take in each of the weighted inputs and feed them through an activation function to add nonlinearity to the function approximation. The output from a node in a hidden layer is calculated via

$$y_{i,j} = f(\bar{W}_i^T \bar{y}_{i-1} + b_{i,j}) \tag{3.1}$$

where i is the current network layer, j is the current node in the network layer, $\bar{y}_{i-1}$ denotes the outputs from the previous network layer, and f represents the activation function. For deep neural networks, this process is continued through multiple hidden layers, each with weighted connections and biases until the final hidden layer which is connected to the output layer, again using a set of weighted connections. The training parameters for the neural network, denoted $\bar{\theta}$ are written as

$$\bar{\theta} = [\bar{W}_1, \bar{b}_1, \bar{W}_2, \bar{b}_2, \bar{W}_3, \bar{b}_3, \ldots, \bar{W}_{h+1}, \bar{b}_{h+1}]^T \tag{3.2}$$

where h is the number of hidden layers in the neural network. Then, the training parameters are updated throughout training via backpropagation [65]. Figure 3.4 displays a conceptual representation of a deep neural network with two hidden layers where the blue nodes represent the input layer; two hidden layers are depicted in purple; the output layer is pink; connections between the nodes are denoted using black arrows; and the nodes within each layer are numbered using superscripts for the layer number and subscripts to enumerate the nodes within one layer. Then, a reinforcement learning algorithm may update the parameters in one or more feed forward neural networks via backpropagation to recover optimal behavior in an environment.

Deep neural networks are distinct from single hidden layer neural networks by the inclusion of two or more hidden layers in the neural network. Deep neural networks often require less nodes per hidden layer to arrive at a local optimal solution and tend to perform better in more complex scenarios than single hidden layer neural networks [90]. Yet, deep neural networks are generally only guaranteed to converge to a local optimum [9, 147]. While constructing and evaluating feed forward

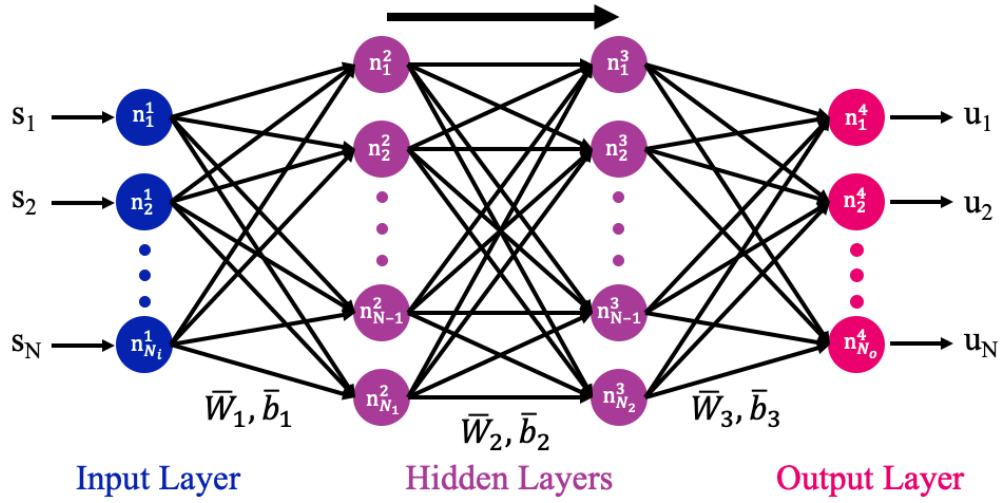


Figure 3.4: Feed forward neural network structure with an input layer, two hidden layers, and an output layer.

neural networks requires a trivial amount of computational resources, training neural networks may be time-consuming and highly sensitive to the structure of the network.

For instance, selecting a suitable activation function, which adds nonlinearity to the function approximation, is still an active areas of research in the reinforcement learning community [106]. The choice of activation function for the nodes of a feed forward neural network is nontrivial and may be problem dependent [106]. Neural networks may use a single activation function for all layers or incorporate distinct activation functions for each layer of the neural network. However, in this research, the neural networks are modeled using the same activation function for all hidden layers to simplify the construction and training of the neural networks. Table 3.1 lists seven activation functions commonly used in reinforcement learning and the associated equations for each function: hyperbolic tangent (Tanh), sigmoid, rectified linear unit (ReLU) formulated by Nair and Hinton, leaky rectified linear unit (LeakyReLU), softplus, exponential linear unit (ELU) introduced by Clevert, Unterthiner, and Hochreiter, and swish developed by Hendrycks and Gimpel [33, 67, 101]. For the LeakyReLU formulation in this research, c_{Leaky} denotes a scaling factor which is set to 0.1. Additionally, Fig. 3.5 depicts the outputs from each activation function for a range of inputs,

Table 3.1: Selection of activation functions commonly used in neural networks trained via reinforcement learning.

Activation Function	Formulation, $y = f(x)$
Hyperbolic Tangent	$y = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
Sigmoid	$y = \frac{1}{1+e^x}$
Rectified Linear Unit	if $x \leq 0$, $y = 0$ else $y = x$
Leaky Rectified Linear Unit	if $x \leq 0$, $y = -c_{Leaky}x$ else $y = x$
Softplus	$y = \ln(e^x + 1)$
Exponential Linear Unit	if $x \leq 0$, $y = e^x - 1$ else $y = x$
Swish	$y = \frac{x}{1+e^{-x}}$

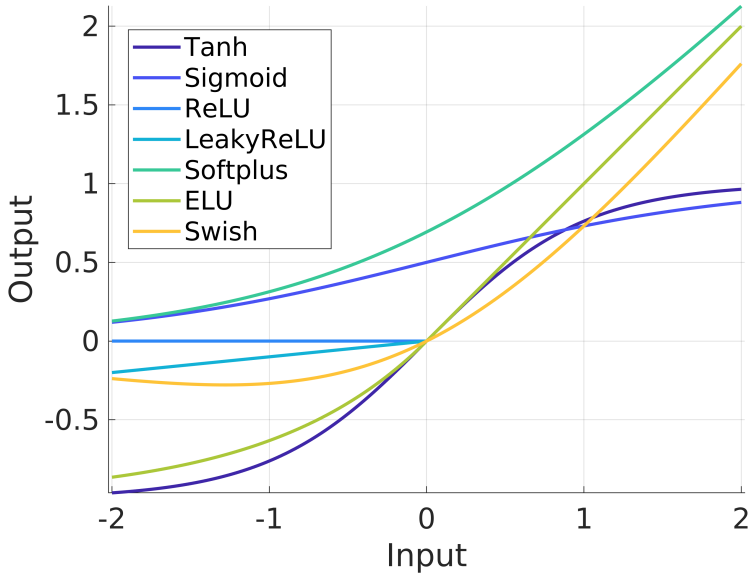


Figure 3.5: Common activation functions evaluated for a range of inputs.

$[-2, 2]$, demonstrating nonlinear behavior for the outputs. However, the most common activation function selections used in reinforcement learning investigations are the Tanh and ReLU functions, and ReLU is primarily used in this research.

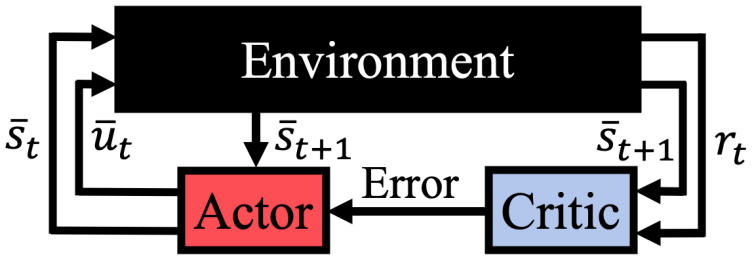
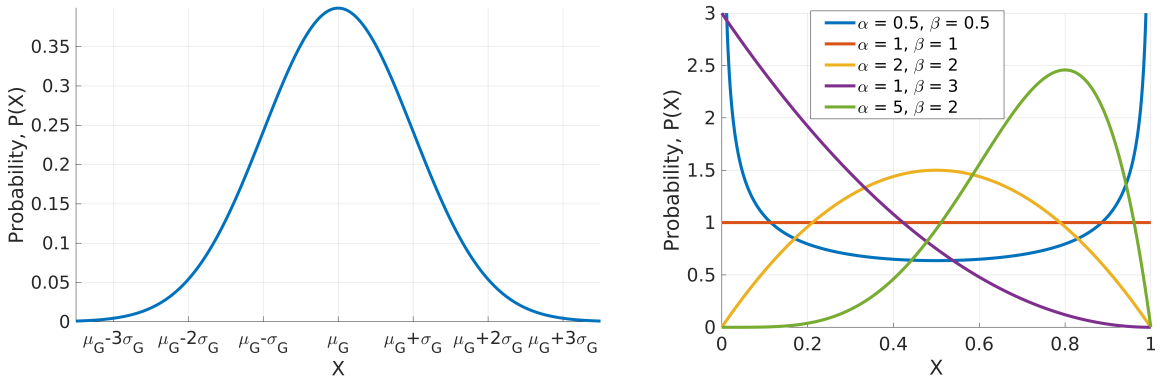


Figure 3.6: Structure of an Actor-Critic method with the environment of the system, actor, critic, and parameters that are passed between the components.

3.4 Actor-Critic Methods

State-of-the-art reinforcement learning algorithms often incorporate an actor-critic structure to uncover optimal behavior within an environment. Actor-critic methods form the foundation of many modern reinforcement learning algorithms due to their robust convergence properties and applicability to a wide variety of problems [66]. Using two distinct structures to learn the optimal behavior simplifies the learning process and leads to more robust convergence properties compared to singular structure methods such as policy iteration or value iteration methods [80]. These methods are typically implemented using two neural networks: one for the actor and one for the critic. While many function approximators exist, neural networks have demonstrated advantageous performance across many applications; thus, they are used to construct the actor and the critic. Figure 3.6 depicts a conceptual representation of an actor-critic method interacting with an environment. In an actor-critic structure, learning is separated into two components: (1) the actor maps states to actions in the environment, denoted as the policy function $\pi(\bar{s}_t, \bar{u}_t)$, and (2) the critic estimates the value function, denoted as $V^{\pi}(\bar{s}_t)$, for every state in the environment [146].

When determining the action to perform at a state in the environment using an actor neural network, the final output layer is formulated to output the components of an independent multivariate probability distribution. This probability distribution is dependent on the state input to the actor neural network. Further, during the training phase for the actor, action components are randomly drawn from the probability distributions while in the evaluation phase, action components



(a) Gaussian distribution

(b) Beta distribution

Figure 3.7: Gaussian distribution with mean μ_G and standard deviation σ_G and beta distributions with varying α and β shape parameters for an action vector component.

are taken from the mean of the distribution. State-of-the-art reinforcement learning approaches generally model the probability distribution as a Gaussian or beta probability distribution. For a Gaussian distribution, the influence of the action components are assumed to be independent, have a mean and standard deviation, μ_G and σ_G, respectively, that may be updated throughout training, and possess covariance values equal to zero. Figure 3.7a depicts a subset of a Gaussian distribution bounded between $\mu_G - 3\sigma_G$ and $\mu_G + 3\sigma_G$. However, a Gaussian distribution is unbounded such that any action drawn must be subsequently bounded according to the constraints of the reinforcement learning scenario. Bounding the action creates a gap between the unbounded action that the actor neural network believes was performed and the bounded action that the environment experiences. Conversely, for a beta distribution, the α and β shape parameters are updated throughout the training process, and the distribution is guaranteed to produce a bounded action [32]. Figure 3.7b displays beta distributions for a variety of α and β values. While using a Gaussian distribution to represent the action action is more common and was observed to produce better convergence behavior, the beta distribution possesses advantageous theoretical properties due to the ability to produce a bounded action [6].

The probability of drawing each action by the actor is stored in logarithmic form as a log-

likelihood value, which is used when estimating the impact of an update to the neural networks. As the neural networks converge on a locally optimal solution, the probability distribution at a state narrows as the networks become more certain of which action to take, i.e. exploiting prior knowledge more than continuing to explore. Then, when evaluating the trained neural networks, the mean of the probability distribution for each action component is used for generating the action to perform at a state.

For the critic, the value function models the expected total reward over the remaining trajectory for every state in the environment and is computed as

$$V^{\pi}(\bar{s}_t) = \mathbb{E}\left[\sum_{t=0}^{\tau} \gamma^t r_t(\bar{s}_t, \bar{u}_t, \bar{s}_{t+1})\right] \tag{3.3}$$

where γ denotes the discount factor which reduces the influence of future rewards and τ is the maximum number of time steps in the trajectory [125]. Unlike the actor neural network, the critic neural network does not approximate the value function via a probability distribution and instead outputs a scalar for each state in the environment. While actor-critic structures with neural networks are often used as the basis for learning the optimal policy and value functions, there are a variety of approaches available for updating the networks.

3.5 Policy Gradient Theorem

A policy gradient method is used to update the parameters of the actor and critic neural networks. In a policy gradient method, the training parameters of the neural networks, $\bar{\theta}$, are iteratively updated based on the collected experiences from the agent interacting with the environment to maximize the expected total reward written as

$$J(\bar{\theta}_j) = \mathbb{E}_t\left[\sum_{t=0}^{\tau} r_t(\bar{s}_t, \bar{u}_t, \bar{s}_{t+1})\right] \tag{3.4}$$

which is evaluated along a single trajectory where $\bar{\theta}_j$ denotes the neural network parameters at update j [125]. To maximize the returned reward, the gradient of this function, labeled $\hat{g}$, is

estimated as

$$\hat{g} = \nabla_{\bar{\theta},j} J(\bar{\theta}_j) = \nabla_{\bar{\theta},j} \, \mathbb{E}_t \left[\sum_{t=0}^{\tau} r_t(\bar{s}_t, \bar{u}_t, \bar{s}_{t+1}) \right] = \hat{\mathbb{E}}_t [\nabla_{\bar{\theta},j} \log \pi_{\bar{\theta},j}(\bar{u}_t|\bar{s}_t) \hat{A}_t] \tag{3.5}$$

where circumflex symbols denote estimated values and $\hat{A}_t$ is termed the estimated advantage function at a defined time step [125]. One approach for estimating the advantages of state-action pairs is using Generalized Advantage Estimation (GAE) developed by Schulman et al. [125]. GAE evaluates the advantage via the difference between the expected and returned rewards as

$$\hat{A}_t^{\pi}(\bar{s}_t, \bar{u}_t) = \sum_{\zeta=0}^{\tau} (\gamma\lambda)^{\zeta} \partial_{t+\zeta} \tag{3.6}$$

where

$$\partial_t = r_t(\bar{s}_t, \bar{u}_t, \bar{s}_{t+1}) + \gamma V_{\bar{\theta},j}^{\pi}(\bar{s}_{t+1}) - V_{\bar{\theta},j}^{\pi}(\bar{s}_t) \tag{3.7}$$

is the estimated advantage of action $\bar{u}_t$ at state $\bar{s}_t$, λ denotes the GAE factor which determines the bias-variance trade-off within the estimated advantages, and $V_{\bar{\theta},j}^{\pi}$ is the estimated value output by the critic with parameters $\bar{\theta}_j$ [125]. Using these definitions, a loss function is formulated as

$$L^{PG}(\bar{\theta}_j) = \hat{\mathbb{E}}_t [\log \pi_{\theta}(\bar{u}_t|\bar{s}_t) \hat{A}_t] \tag{3.8}$$

which may be differentiated to generate Eq. (3.5) and may be used within an optimization scheme to train the actor and critic neural networks [125]. However, the limitation of this approach is the potential for large updates to destabilize the neural networks under poorly sampled state, action, and reward data.

Instead, TRPO 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, i.e. difference, between two updates [124]. Using TRPO, the optimization problem for updating a policy may be written as

$$\text{maximize}_{\bar{\theta}}: \qquad \hat{\mathbb{E}}_t \left[\frac{\pi_{\bar{\theta},j}(\bar{u}_t|\bar{s}_t)}{\pi_{\bar{\theta},j-1}(\bar{u}_t|\bar{s}_t)} \hat{A}_t \right] \tag{3.9}$$

$$\text{subject to:} \qquad \hat{\mathbb{E}}_t [KL[\pi_{\bar{\theta},j-1}(\cdot|\bar{s}_t), \pi_{\bar{\theta},j}(\cdot|\bar{s}_t)]] \leq \delta \tag{3.10}$$

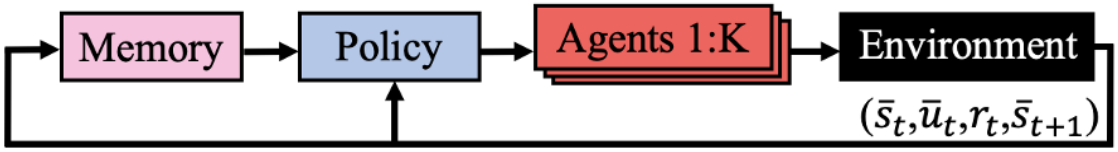


Figure 3.8: Representative diagram of the structure of PPO for a reinforcement learning problem.

where KL indicates the KL divergence function evaluation and δ is a bound on the KL divergence between the policies [124]. Additionally, the probability ratio, R_t, between the old and new policies is defined as

$$R_t(\bar{\theta}_j) = \frac{\pi_{\bar{\theta},j}(\bar{u}_t|\bar{s}_t)}{\pi_{\bar{\theta},j-1}(\bar{u}_t|\bar{s}_t)} \tag{3.11}$$

where probability ratios near unity denote smaller updates to the policy [124]. However, determining a suitable value for δ is nontrivial across multiple, distinct environments. Further, even within a single environment, adjusting parameters of the implementation may necessitate updating δ. Additionally, TRPO requires solving a nontrivial constrained optimization problem for each update to the neural networks. Subsequent policy gradient algorithms have been developed that improve on these limitations, but generally follow the structure defined above.

3.6 Proximal Policy Optimization

PPO was originally developed to overcome the limitations of TRPO, eliminating the necessity of predefining a trust region and reducing the complexity of solving the constrained optimization problem [126]. PPO trains a single policy to maximize the expected total reward returned for an environment and reward function while limiting the size of updates from destabilizing the networks by enforcing a soft constraint within the objective function [126]. The soft constraint is implemented via a clipping parameter, ε, defined using the ratio, $R_t(\theta)$; any update to the policy $\pi_{\theta,j}(\bar{u}_t|\bar{s}_t)$ that causes the ratio to diverge away from 1 is penalized. Figure 3.8 depicts a representative diagram demonstrating how PPO may be used to train a policy to maximize the expected total reward in a reinforcement learning environment.

The objective function used by PPO to update the neural networks consists of three elements: (1) the clipped objective which inhibits highly rewarding or penalizing state-action pairs from over correcting the neural networks, (2) a value error term that measures the error of the estimated value function by the critic with respect to the actual value computed using the training data, and (3) an entropy term that balances the exploration-exploitation trade-off in the environment [126]. First, the clipped objective function is written as

$$L_t^{CLIP}(\bar{\theta}_j) = \hat{\mathbb{E}}_t[\min(R_t(\bar{\theta}_j)\hat{A}_t, \text{clip}(R_t(\bar{\theta}_j), 1-\varepsilon, 1+\varepsilon)\hat{A}_t)] \tag{3.12}$$

which is formulated to ignore changes in the probability ratio if the update improves the objective, but considers the probability ratio when the update produces a decrease in the objective function [126]. Then, to encourage the critic to accurately approximate the value function, the value error objective is defined as

$$L_t^{VF}(\bar{\theta}_j) = (V_{\bar{\theta},j}^{\pi}(\bar{s}_t) - V_t^{target}(\bar{s}_t))^2 \tag{3.13}$$

where V_t^{target} is computed using the rewards generated from the environment [126]. Additionally, an entropy term is defined using the entropy of the probability distribution output by the actor, $S_t[\pi_{\bar{\theta},j}]$, to encourage exploration within the environment. Combining the three terms, the objective function that is used within PPO is then written as

$$L_t^{CLIP+VF+S}(\bar{\theta}_j) = \hat{\mathbb{E}}_t[L_t^{CLIP}(\bar{\theta}_j) - c_1 L_t^{VF}(\bar{\theta}_j) + c_2 S_t[\pi_{\bar{\theta},j}](\bar{s}_t)] \tag{3.14}$$

c_1 and c_2 are scalar coefficients for the value error and entropy terms, respectively [126].

Once a set number of state-action-reward experiences for a policy are evaluated using the objective function, the weights of the neural networks are optimized and updated using the AdamW optimizer. AdamW offers improved convergence performance compared to the traditionally used Adam optimizer by incorporating a weight decay term in the update to the neural networks [79, 92]. PPO training neural networks in an actor-critic structure using AdamW has demonstrated strong performance in chaotic dynamical environments compared to other reinforcement learning algorithms [6]. Using this approach, PPO trains a single policy to maximize the expected total reward generated over a trajectory for a defined dynamical model and reward function.

3.7 Transfer Learning

Transfer learning enables a set of neural networks trained in a dynamical model for a defined objective to initialize a policy in a more complex dynamical model or for a distinct objective. This enables policies to be trained more rapidly than re-initializing the policies with no a priori domain knowledge [151]. For example, transfer learning has been used to develop a controls scheme for a robot in a simulated environment prior to uploading and updating the controls scheme for a robot in a real-world environment [159]. In this research, transfer learning enables policies first trained in a lower-fidelity dynamical model to serve as initial guesses for policies in a higher-fidelity dynamical model with similar objectives. Then, once a policy recovers a locally optimal solution in the simpler dynamical model, the policy is used as an initial guess for a similar locally optimal solution in the more complex dynamical model. This is valuable for dynamical regimes where the propagation time for state-action pairs comprises a significant portion of the total training time, such as the point mass ephemeris model, or when a simulated environment cannot capture the complexity of a real-world environment. In this research, a C++ implementation of numerical integration in the point mass ephemeris model requires approximately $50\times$ the computational time compared to a similar numerical integration in the CR3BP. This increase in total computational time effectively prohibits the rapid training of policies in the point-mass ephemeris model. Instead, transfer learning is leveraged to first train policies to recover locally optimal transfers in the CR3BP with no a priori domain knowledge, and initialize policies trained in the point mass ephemeris model for a much smaller number of updates to the neural networks.

Chapter 4

Multi-Reward Proximal Policy Optimization

In this chapter, a multi-objective reinforcement learning algorithm, designated MRPPO, is presented to produce solutions to a multi-objective optimization problem. MRPPO leverages PPO as a building block for a multi-objective framework, and expands the structure of PPO to enable multiple policies, each with distinct reward function formulations, to be trained simultaneously by sharing state-action propagation information across all policies [140, 142]. The reward functions are composed of the same components with distinct weightings to enable the policies trained by MRPPO to recover locally optimal solutions distinctly balancing multiple objectives and spanning a multi-objective solution space. Additionally, sharing the propagation information with additional policies significantly reduces the required computational time and resources since numerical integration often accounts for over 90% of the total training time for a policy in multi-body gravitational environments. Further, sharing the propagation data also enables policies to learn from the innovative or disadvantageous actions undertaken by other policies, aiding the convergence and stability of all policies. Since multiple policies, each with unique objective prioritizations, are simultaneously trained on a single scenario, the required computational resources are decreased, stability is introduced, and a subset of the solutions spanning the multi-objective design space may be more efficiently generated. However, like PPO, the policies developed using MRPPO and the performance of the algorithm are influenced by hyperparameter selections.

4.1 Multi-Reward Proximal Policy Optimization

MRPPO leverages a multi-objective framework using PPO as a foundation to train multiple policies, each assigned a distinct reward function, to autonomously recover a subset of the multi-objective solution space for a defined dynamical model [140, 141]. The multi-objective framework allows the state-action pairs generated by one policy to be shared with every other policy throughout training. Sharing state-action pairs across policies significantly reduces the total computational time required to train multiple policies and recover solutions spanning the multi-objective solution space of a trajectory design problem. Additionally, potentially beneficial or disadvantageous actions undertaken by one policy may be used by other policies to improve their own performance leading to enhanced behavior across all policies. Using MRPPO, N policies, denoted $[\mathrm{P}_1 \dots \mathrm{P}_N]$, are trained in a dynamical model where each policy controls K_i independent agents gathering state-action pairs stored in a shared memory. During the training process, policies receive the current state at time t of each assigned agent for that policy. Then, the actor neural network of each policy outputs actions for each agent to perform that each policy controls. The dynamical model returns the state at time $t+1$ to generate a state-action transition, $(\bar{s}_t, \bar{u}_t, \bar{s}_{t+1})$. For one update to the neural networks, this process is repeated until the maximum number of state-action transitions, denoted Λ, have been generated for each policy. Then, the state-action transitions for one policy are shared with every other policy via saving in a shared memory; thus, each policy learns from an additional $(N-1)\Lambda$ transitions in the environment.

Once a set number of state-action transitions have been collected in the dynamical model, the shared state-action transitions are input to the reward function for each policy and the resulting state-action-reward experiences are used to update each policy. Specifically, the state-action transitions from all policies are input into the assigned reward function for each policy, designated $r_{i,t}(\bar{s}_t, \bar{u}_t, \bar{s}_{t+1})$ for policy P_i. The reward function assigned to each policy corresponds to the unique objective prioritization that each policy is attempting to maximize via recovery of a locally optimal set of parameters for the neural networks. The policies are continually updated throughout training

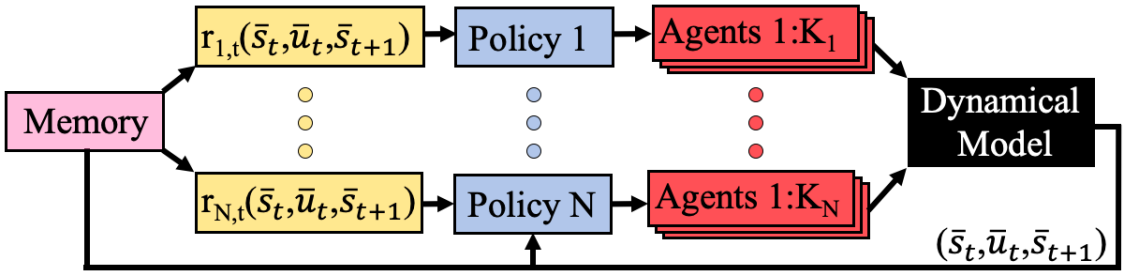


Figure 4.1: MRPPO architecture illustrating N policies, each with an assigned reward function $r_{i,t}(\bar{s}_t, \bar{u}_t, \bar{s}_{t+1})$, controlling K_i agents subject to the dynamical model.

until a terminating condition is reached; typically, terminating the training process is defined using a maximum number of updates. Figure 4.1 displays the structure of MRPPO for N policies in blue, each controlling K_i agents, depicted in red, the dynamical model is illustrated as a black box, the reward function for each policy is displayed in gold, and the memory object in pink stores state-action transitions prior to updating each policy. Additionally, while policies may be trained using distinct hyperparameters and neural network structures, in this research, policies are trained using the same set of hyperparameters and neural network structures. This assumption simplifies the training process and facilitates direct comparisons between the policies. While the training phase for MRPPO follows a similar configuration to PPO, the updates to the neural network parameters for MRPPO benefit from modifications to the objective function.

Due to the limited machine precision available and the statistical nature of certain actions, the objective function used by PPO must be adjusted for MRPPO to enable a smooth training process. For instance, actions selected by one policy must be evaluated to determine their value and log-likelihood for every other policy during training. Throughout the training process, policies become more deterministic and exploitative as they converge towards locally optimal solutions. Then, actions selected by one policy may become statistically unlikely for other policies, which is encapsulated within the log-likelihood action probabilities. However, the limitations imposed by machine precision cause some of these log-likelihood values to equal zero thereby causing the denominator of the probability ratio defined in Eq. (3.11) to produce near-infinite values. This

causes the backpropagation process to destabilize the neural networks during the update phase of training. Instead, to combat this destabilizing effect, the probability ratio is reformulated as

$$R_{i,t}(\bar{\theta}_{i,j}) = \pi_{\bar{\theta},i,j}(\bar{u}_t|\bar{s}_t) - \pi_{\bar{\theta},i,j-1}(\bar{u}_t|\bar{s}_t) \tag{4.1}$$

where $\bar{\theta}_{i,j}$ denotes the neural networks parameters corresponding to policy P_i at update j [68, 166]. This formulation inhibits statistically improbable actions from derailing the training process. Using the redefined probability ratio, values near zeros indicate negligible changes between updates to a policy. The clipping objective used by PPO defined in Eq. (3.12) is then rewritten as

$$L_{i,t}^{CLIP}(\bar{\theta}_{i,j}) = \hat{\mathbb{E}}_t[\min(R_{i,t}(\bar{\theta}_{i,j})\hat{A}_{i,t}, \mathrm{clip}(R_{i,t}(\bar{\theta}_{i,j}), -\varepsilon, \varepsilon)\hat{A}_{i,t})] \tag{4.2}$$

to include the new bounds, $[-\varepsilon, \varepsilon]$ on the probability ratio [166]. Then, the objective function used by MRPPO is updated to

$$L_{i,t}^{CLIP+VF+S}(\bar{\theta}_{i,j}) = \hat{\mathbb{E}}_{i,t}[L_{i,t}^{CLIP}(\bar{\theta}_{i,j}) - c_1 L_{i,t}^{VF}(\bar{\theta}_{i,j}) + c_2 S_{i,t}[\pi_{\bar{\theta},i,j}](\bar{s}_t)] \tag{4.3}$$

Using these equations, MRPPO is implemented in PyTorch, an open-source machine learning library, but may be formulated in other open-source machine learning libraries [112]. Finally, MRPPO has been observed to retain PPO's beneficial convergence properties in chaotic environments as demonstrated in Ch. 7.

4.2 Hyperparameters

MRPPO is governed by multiple hyperparameters that influence the resulting behavior of the trained policies. Using PPO as a foundation, MRPPO generally retains PPO's robustness to hyperparameter selection compared to other state-of-the-art reinforcement learning algorithms [6, 66]. Generally, the hyperparameter selection process explores a high-dimensional, multi-modal distribution where the effects of certain hyperparameters are coupled. Hyperparameters that influence the policies trained by MRPPO include:

- Environmental steps taken during training, Λ: sets the number of state-action-reward experiences gathered using each agent prior to updating the parameters of the neural networks.

Note that the total number of state-action-reward experiences used when updating the neural networks, Ψ, is a function of both the number of agents, K, and the number of environmental steps such that $\Psi = K \times \Lambda$.

- Epochs, E: determines the number of times the state-action-reward experiences are used to update the parameters.

- Mini batches, M: divides the total number of experiences into smaller sets of data, where each subset is then used to update the parameters. Figure 4.2 illustrates the relationship between epochs and mini batches demonstrating the total number of updates to the parameters of a neural network for a set of experiences is $E \times M$.

- Update phases, Φ: determines the number of times state-action-reward experiences are gathered in the environment to update the neural networks.

- Discount factor, γ: determines how heavily future rewards are discounted when computing the estimated value and advantage functions.

- GAE factor, λ: affects how states are considered when calculating the advantages and influences the bias-variance trade-off in the estimated advantage computation using GAE.

- Learning rate, l_r: affects how quickly the neural networks are updated from a set of state-action-reward experiences using the AdamW optimizer, set equal for both the actor and critic neural networks. Note, the learning rate is modified throughout training following a triangular cycle increasing from the initial learning rate to the maximum learning rate and then decreasing back to the initial learning rate.

- Clipping parameter, ε: limits the size of the updates to the parameters of the neural networks in the objective function.

- Value function coefficient, c_1: scales the value loss error in the objective function.

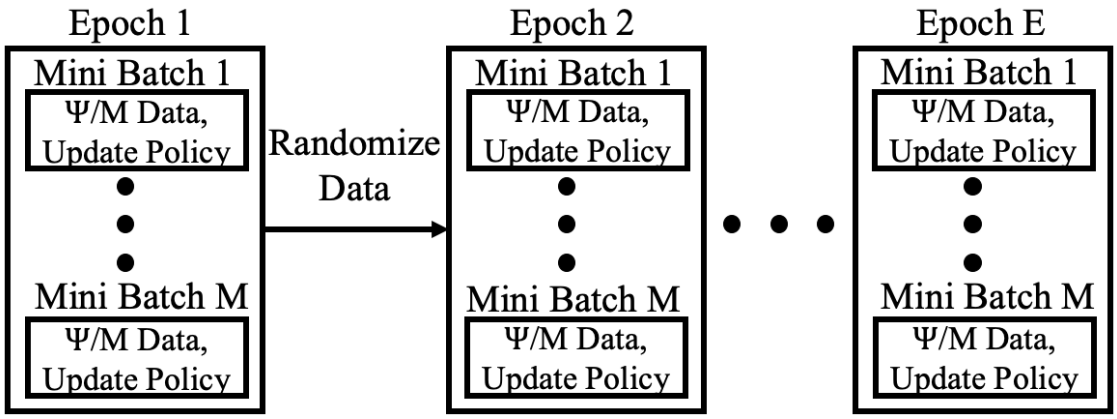


Figure 4.2: Epoch and mini batches used to update the neural networks for Ψ experiences where the data is randomized between each epoch to aid in convergence.

- Entropy coefficient, c_2: determines the amount of entropy inclucded in the objective function calculation.
- Maximum gradient norm, c_{max}: constrains the estimated gradients of the objective function to lie within specified bounds.

These hyperparameters directly influence the policy training, but additional factors may also affect the policies and their performance.

Construction parameters are also defined that influence the resulting behavior of the policies during and after the training phase. Construction parameters that influence the policies recovered by MRPPO in this research include:

- Actor hidden layers, l_a: determines the number of hidden layers in the actor neural network.
- Critic hidden layers, l_c: sets the number of hidden layers in the critic neural network.
- Actor hidden nodes, n_a: dictates the number of hidden nodes per hidden layer in the actor neural network.
- Critic hidden nodes, n_c: governs the number of hidden nodes per hidden layer in the critic neural network.

- Activation function: determines the activation function used within each hidden layer of the actor and critic neural networks.
- Initialization scheme: governs how the weights and biases of the neural networks are initialized prior to training.
- Action space representation: dictates how the probability distribution for the actor neural network is parameterized.

Both the hyperparameters and construction parameters may be adjusted to improve the performance of policies trained by MRPPO in this research.

Chapter 5

Low-Thrust Trajectory Design as a Reinforcement Learning Problem

MRPPO is used to recover low-thrust transfers between periodic orbits in multi-body gravitational environments without the need for a predefined reference trajectory or initial guess. First, the trajectory design problem is formulated as a reinforcement learning problem necessitating definition of the spacecraft parameters, state vector, action formulation, reward function, initial and final periodic orbits, engine configuration, and scenario parameters. Policies are initialized without any a priori knowledge of the environment or objectives and are trained to guide a low-thrust-enabled SmallSat from an initial periodic orbit to a final periodic orbit in a multi-body system. Transfer learning may be employed in certain trajectory design scenarios to update the policies to construct locally optimal low-thrust transfers in a point mass ephemeris model.

A path planning approach similar to a set of continuous waypoints is also incorporated into MRPPO to enable policies to autonomously generate low-thrust transfers from an initial periodic orbit to a final periodic orbit in a multi-body system. Specifically, reference trajectories are selected using the best trajectories generated throughout training to guide the policies from the initial periodic orbit to the final periodic orbit. As observed in prior work, reference trajectories often serve as valuable guides for training policies to recover transfers between two periodic orbits in complex trajectory design scenarios; without a reference trajectory, training in both PPO and MRPPO may fail to produce successful policies [141, 143]. As constructed, updating the reference trajectories throughout training requires no initial guess to be defined prior to training and enables the best trajectories generated throughout training for each policy to serve as a reference trajectory until a

Table 5.1: Initial and final periodic orbits for each transfer scenario and the associated nondimensional characteristics.

Scenario	Orbit	x	y	z	$\dot{x}$	$\dot{y}$	$\dot{z}$	Period	C_J
# 1	L_2 Southern Halo	1.1676	0	-0.1029	0	-0.1973	0	3.3216	3.11
	L_2 Southern Halo	1.1484	0	-0.1494	0	-0.2192	0	3.1808	3.07
# 2	L_1 Northern Halo	0.8687	0	-0.0451	0	-0.1881	0	2.7614	3.15
	L_2 Southern Halo	1.1676	0	-0.1029	0	-0.1973	0	3.3216	3.11
# 3	L_2 Northern Halo	1.0081	0	-0.0016	0	0.0108	0	3.0983	3.0008
	L_5 Short Period	0.5945	-0.8660	0	-0.0591	-0.0916	0	6.2832	2.995

better trajectory is generated. Using MRPPO with the updating reference trajectories enables the design of transfers in high-dimensional, multi-objective solution spaces.

5.1 Trajectory Design Scenario Overview

In this research, trajectory design scenarios are defined for low-thrust transfers between periodic orbits in the Earth-Moon and Sun-Earth CR3BP. Specifically, three types of low-thrust transfers are designed: (1) from an L_2 southern halo orbit to a higher-energy L_2 southern halo orbit in the Earth-Moon CR3BP, (2) from an L_1 northern halo orbit to an L_2 southern halo orbit in the Earth-Moon CR3BP, and (3) from an L_2 northern halo orbit to an L_5 short period orbit in the Sun-Earth CR3BP. The selected periodic orbits are members of the periodic orbit families depicted in Figs. 2.4 and Figs. 2.5. Table 5.1 presents initial conditions and characteristics of the periodic orbits for each trajectory design scenario. Each of these scenarios possesses an increasing level of complexity for transfer design and also reflects the differences in sensitivity between the Earth-Moon and Sun-Earth systems. However, in each of these scenarios, the transfers are designed to balance minimizing propellant mass usage and flight time by adjusting the transfer geometry and thrust profile.

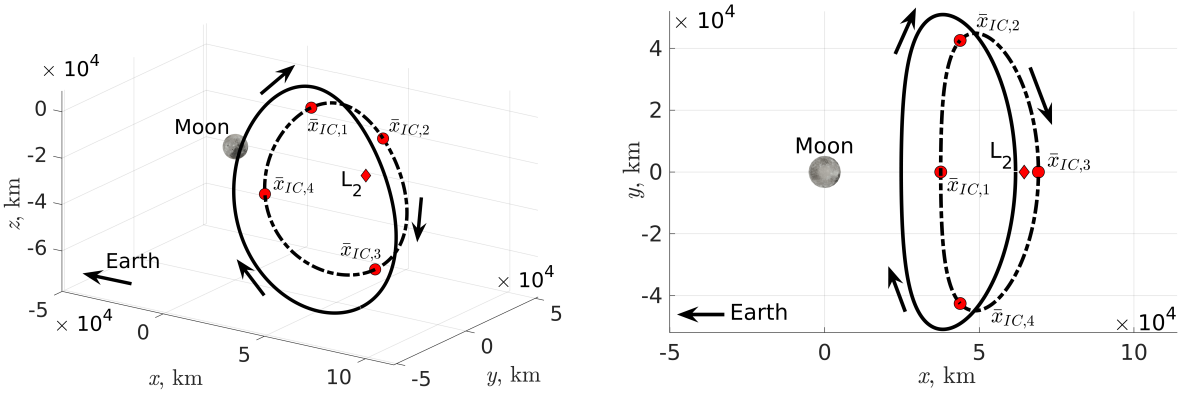
Scenario # 1: The goal of the first trajectory design scenario is to construct low-thrust transfers between neighboring periodic orbits in the Earth-Moon CR3BP. In this scenario, four

Table 5.2: Departure locations along the Earth-Moon L_2 southern halo orbit for the first trajectory design scenario.

Location	x	y	z	$\dot{x}$	$\dot{y}$	$\dot{z}$
1	1.0855	0	0.0626	0	0.2735	0
2	1.1018	0.1107	0	0.0645	0.0713	-0.1576
3	1.1676	0	-0.1029	0	-0.1973	0
4	1.1018	-0.1107	0	-0.0645	0.0713	0.1576

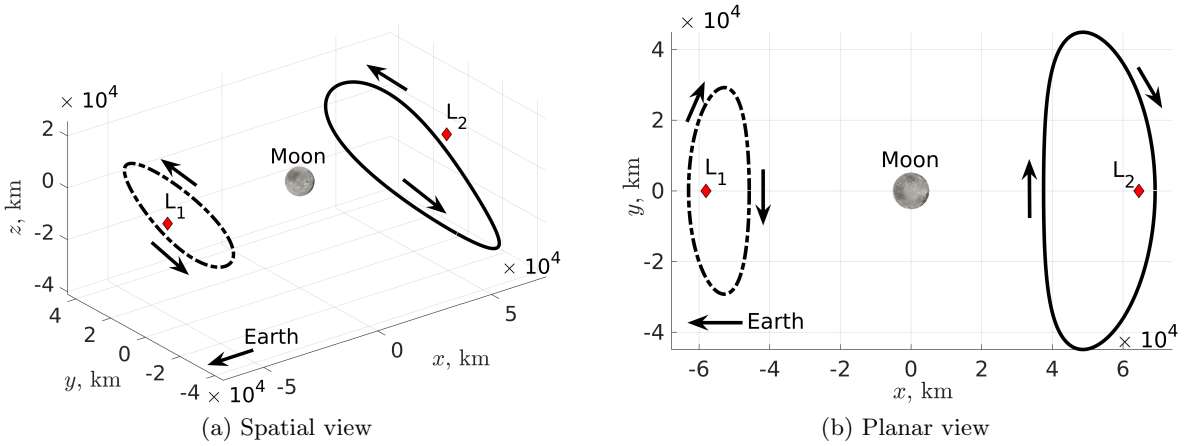
policies are trained to develop low-thrust transfers. The spacecraft is assumed to begin in the vicinity of a fixed point along the initial halo orbit, and target any state along the final L_2 southern halo orbit. Specifically, four departure locations are explored where the initial conditions of the spacecraft are drawn near one of four locations around the initial orbit defined in Table 5.2. Figure 5.1 depicts both a spatial and planar view of this trajectory design scenario and the locations of the initial conditions for each of the four departure locations. In these figures, the black dashed arc denotes the initial periodic orbit, the solid black arc represents the final periodic orbit, and the equilibrium points are displayed using red diamonds. This trajectory design scenario serves as a valuable test case for performing a hyperparameter exploration and generating insights into the influence of hyperparameter selections for MRPPO prior to moving onto more complex scenarios incorporating flybys of primary bodies.

Scenario # 2: In the second trajectory design scenario, low-thrust transfers are explored between periodic orbits near the L_1 and L_2 equilibrium points of the Earth-Moon CR3BP. Four policies are trained to generate low-thrust transfers spanning the propellant mass usage and flight time trade space. Figure 5.2 displays the initial and final periodic orbits defined for this trajectory design scenario using the same color configuration as Fig. 5.1. Specifically, the spacecraft are initialized along an L_1 northern halo orbit and follow a reference trajectory that is autonomously generated and updated by a policy to target an L_2 southern halo orbit. This scenario demonstrates that the neural networks may recover transfers in dynamically sensitive regions of the Earth-Moon



(a) Spatial view (b) Planar view

Figure 5.1: Initial and final periodic orbits and initial condition locations defined for the first trajectory design scenario.

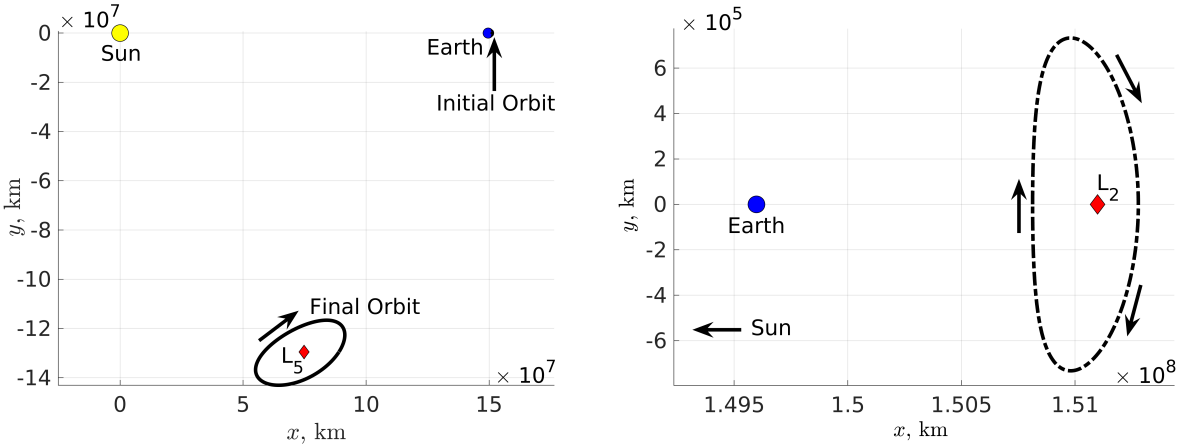


(a) Spatial view (b) Planar view

Figure 5.2: Initial and final periodic orbits defined for the second trajectory design scenario.

CR3BP with close lunar flybys, similar to those planned for the upcoming Lunar Gateway and Artemis program [41, 132].

Scenario # 3: The final scenario requires guiding a low-thrust-enabled SmallSat from a Sun-Earth L_2 northern halo orbit to a Sun-Earth L_5 short period orbit in the Sun-Earth CR3BP. Recall, the Sun-Earth L_5 region is of significant interest to the scientific community. In fact, spacecraft positioned at the Sun-Earth L_5 equilibrium point may serve as early warning systems



(a) Planar view (b) Zoomed-in view

Figure 5.3: Initial and final periodic orbits defined for the third trajectory design scenario.

for detecting coronal mass ejection events while also providing a unique perspective for studying the solar environment that is currently unavailable to the scientific community [123]. For this scenario, the number of policies trained is increased from four to eight to more thoroughly span the multi-objective solution space. Once policies are trained to recover transfers in the Sun-Earth CR3BP, similar transfers are recovered in a higher-fidelity ephemeris model via transfer learning. By using the policies trained in the low-thrust CR3BP dynamical model as an initial guess for training in the higher-fidelity model, policies may be updated to produce optimal low-thrust transfers for significantly less computational time and resources. Figure 5.3 depicts the periodic orbits in the Sun-Earth CR3BP defined for this trajectory design scenario using the same color configuration as Fig. 5.1. Additionally, a comparison is performed to the investigation by Elliott et al., which explored transfers with similar spacecraft and deployment conditions to a Sun-Earth L_5 short period orbit with $C_J = 2.995$ [48].

In these three trajectory design scenarios, due to the difference in energy between the orbits and the potential for no natural transfers, maneuvers are required to facilitate these transfers. Further, each trajectory design scenario necessitates the exploration of motion outside of the Earth-Moon and Sun-Earth xy-planes prohibiting planar simplifications. This limits the application of

certain dynamical systems theory techniques that rely upon the planar simplification. Additionally, for a constant thrust engine configuration, MRPPO may be used to autonomously locate locally optimal segments to activate the low-thrust engine along a trajectory rather than require a human expert, with limited insight, to determine where to place the low-thrust segments. Finally, MRPPO may recover locally optimal solutions for discontinuous constraints and objectives, which would violate the assumptions of many traditional trajectory design techniques. Reinforcement learning, and by extension MRPPO, may serve a new trajectory design technique that possesses advantages compared to traditional trajectory design techniques for low-thrust transfers in multi-body systems.

5.1.1 Spacecraft Properties

In all three trajectory design scenarios, the agents in the environment are modeled as ESPA-class SmallSats equipped with a single constant specific impulse engine with full control over thrust direction in the CR3BP rotating frame. However, for transfers in the Sun-Earth-Moon point mass ephemeris model, the thrust direction is specified in the Sun-Earth instantaneous rotating frame. While SmallSats may provide a viable low-cost option for achieving a targeted science return, they introduce significant challenges into the trajectory design process. Trajectory designers must contend with the additional challenges that the SmallSat form factor imposes including: low maneuverability due to a limited propulsive capability, constrained deployment conditions, and operational scheduling constraints due to limited power. These challenges all impact the availability and geometry of feasible transfer paths that tend to exist in a high-dimensional design space [26].

The spacecraft is assumed to have an initial wet mass of $\tilde{m}_{sc,0} = 180$ kg and a specific impulse of $\tilde{I}_{sp} = 3000$ s, both within the current or near-term technological capabilities of low-thrust engines [24, 76]. However, the nominal maximum thrust for each scenario is varied according to the maximum thrust required to complete the periodic orbit transfer. Specifically, the nominal maximum thrust is determined by running simulations for a range of maximum thrust values and selecting the lowest value in which the policies develop transfers that consistently converge on the final periodic orbit. In the first trajectory design scenario, the spacecraft is assigned a nominal

maximum thrust of $\tilde{T}_{max} = 250$ mN while the second and third trajectory design scenarios are assigned maximum thrusts of $\tilde{T}_{max} = 150$ mN and $\tilde{T}_{max} = 25$ mN, respectively. Additionally two engine models are explored: (1) a variable thrust engine where the spacecraft may output a thrust magnitude between $\tilde{T} = [0, \tilde{T}_{max}]$ and (2) a constant thrust engine where the spacecraft may only coast or thrust at the maximum thrust level. These two engine models are reflective of distinct technological capabilities available for SmallSats. Note, the constant thrust engine configuration is only implemented within the third trajectory design scenario.

5.1.2 Episode Initialization

Throughout training for each trajectory design scenario, the initial conditions for the spacecraft are randomly drawn from along the initial periodic orbit and perturbed in both position and velocity components. For the first trajectory design scenario, the departure location along the initial periodic orbit is fixed for a single training run. However, for the second and third trajectory design scenarios, the location along the initial periodic orbit is drawn from a uniform probability distribution such that the entire periodic orbit is equally sampled for initial conditions. Additionally, for all three trajectory design scenarios, the induced perturbations are drawn from a Gaussian distribution with zero mean and a standard deviation of σ_{IC} nondimensional units for both the position and velocity. Perturbing the spacecraft off of the initial orbit enables the policies to experience a wider array of environmental information within the vicinity of the initial orbit and reflects off-nominal initial conditions the spacecraft may encounter. For the first and second trajectory design scenarios, which are in the Earth-Moon CR3BP, the standard deviation is set to $\sigma_{IC} = 1 \times 10^{-3}$, while for the third trajectory design scenario in the Sun-Earth CR3BP, the standard deviation is set equal to $\sigma_{IC} = 1 \times 10^{-8}$. Due to the difference in distance scales between the Earth-Moon and Sun-Earth CR3BP, a smaller standard deviation is required to prevent spacecraft from being initialized far outside of the initial periodic orbit in the Sun-Earth CR3BP.

Three additional values for the input state to the neural networks are also initialized with the spacecraft: κ, the initial time, and the initial mass. For the first trajectory design scenario,

$\kappa = 0$ at the initial time step since the reference trajectory is always the nearest dynamical arc. For the second and third trajectory design scenarios, the initial κ value is determined depending on if the spacecraft is closer to the initial orbit or reference trajectory where κ is set to either -1 or 0, respectively. The initial time is always set to 0. Finally, the initial mass is set equal to the initial wet mass of the spacecraft, $m_{sc,0} = 180$ kg for all three trajectory design scenarios.

Once the training process is complete, the trained neural networks are evaluated on a common set of initial conditions to facilitate equivalent comparisons between the policies. For the first trajectory design scenario and hyperparameter exploration, the trained neural networks are evaluated on 25 initial conditions. However, for the second and third trajectory design scenarios, the policies are evaluated on 1,000 initial conditions. This initialization ensures equivalent comparisons between all of the policies and that enough initial conditions are evaluated to analyze the behavior of the policies. Once a spacecraft is initialized near the initial orbit, the state and action vectors for the policies are formulated, and a transfer is developed to target the final periodic orbit.

5.2 Autonomous Trajectory Design with MRPPO

To enable policies trained by MRPPO to autonomously recover solutions in complex trajectory design scenarios, a path planning approach similar to a set of continuous waypoints is incorporated to facilitate low-thrust transfers between periodic orbits in a multi-body system. Specifically, reference trajectories are used to guide the policies from the initial periodic orbit to the final periodic orbit since reference trajectories often serve as valuable guides for training policies to recover transfers between two periodic orbits. Prior investigations in a variety of path planning applications have noted the limitations of DRL algorithms to converge on locally optimal trajectories in complex environments without either a priori domain knowledge or the use of path planning algorithms [154, 134]. To address this challenge, some applications of reinforcement learning for path planning have successfully leveraged a set of waypoints that guides an agent through the configuration space while maximizing the reward function; these waypoints may be fixed or iteratively adjusted by the reinforcement learning algorithm or a path planning algorithm [154, 134, 163]. These waypoints

are often states or position vectors that are defined a priori or autonomously generated. Directly applying this waypoint concept to low-thrust transfer design in a multi-body system would require time and support from a human with sufficient knowledge of the solution space; an expectation that is not always reasonable, particularly in the low-thrust-enabled CR3BP. Rather, this paper builds upon this approach by formulating a reinforcement learning problem for transfer design that uses the best transfer generated by each policy during training directly in the state and reward definitions. The best transfer for each policy, labeled within this paper as the policy's 'reference trajectory', is autonomously generated as the agents controlled by that policy explore the environment and updated as the policies learn better control profiles. This approach is empirically observed to facilitate each policy uncovering locally optimal behavior that leverage the natural flow of motion within complex environments while reducing both the bias induced by a human trajectory designer as well as the time and computational resources needed to manually identify a reference trajectory or sequence of waypoints a priori.

Throughout the training process, the reference trajectories are autonomously updated, which aids policies in autonomously recovering solutions in a complex trajectory design scenario [143]. Each policy may construct and follow one or more reference trajectories from the initial orbit to the final orbit that are initially undefined [141]. The policies each autonomously generate multiple reference trajectories such that if a spacecraft is initialized along one segment of the initial periodic orbit, the transfer may follow a distinct geometry than if the spacecraft is initialized at a different segment of the initial periodic orbit. Updating the reference trajectory is defined by the following steps during the training of each policy:

(1) Generate a trajectory for policy P_i that follows reference trajectory $\mathfrak{R}_{i,k}$

(2) Evaluate as reference trajectory $\mathfrak{R}_{i,k}$

- If first trajectory: set as $\mathfrak{R}_{i,k}$
- Else: compare to $\mathfrak{R}_{i,k}$ and replace if better

where $\mathfrak{R}_{i,k}$ is the reference trajectory generated by policy P_i for initial conditions initialized along the k^{th} segment of the initial periodic orbit. Figure 5.4 displays a representation of updating a

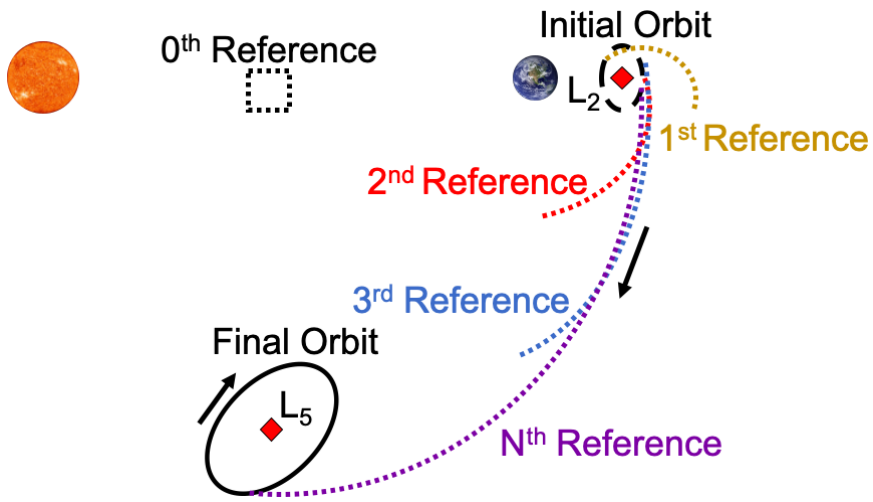


Figure 5.4: Updating the reference trajectory throughout training in a transfer design scenario.

reference trajectory for a single policy during training to recover transfers between two periodic orbits. In this figure, the initial periodic orbit is represented by the black dashed arc, the final periodic orbit is denoted by the solid black arc, the initial reference trajectory is expressed as an empty black dashed box to indicate that it is undefined at the start of the training process, and the reference trajectories are depicted in a variety of hues to demonstrate how the reference trajectory for this policy is updated throughout the training process. As training progresses, the reference trajectory converges on a locally optimal solution. Additionally, Fig. 5.5 displays a zoomed-in view demonstrating that for a single policy, reference trajectories with distinct geometries may be generated for each region of the initial periodic orbit. Then, the best trajectories generated throughout the training process are used as the reference trajectories for each policy.

Determining whether a new training trajectory supplies a better reference may depend on the properties of interest, e.g., via criteria defined in the phase space, evaluation of the reward function, or a combination of factors. Throughout training, a reference trajectory is continually updated based on a set of criteria that may include closest approach to the final orbit, highest total reward, lowest propellant mass usage, or other objectives and constraints significant to the trajectory design scenario. To recover transfers in multi-body trajectory design scenarios with

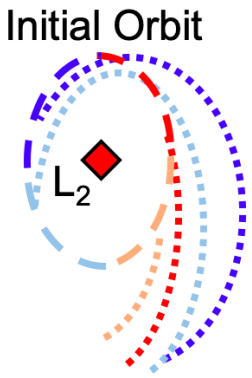


Figure 5.5: Zoomed-in view of reference trajectories with unique geometries assigned to distinct segments of an initial periodic orbit.

minimal propellant mass usage, a combination of criteria is used to determine when a training trajectory should be used as a reference trajectory. Initially, reference trajectories are determined by the trajectories that approach the target orbit closest in configuration space. Since trajectories may approach and then depart the final periodic orbit prior to the policies learning to converge on the orbit early on in the training, an average of the k closest states in position space to states along the final orbit is computed. To facilitate an accurate approximation without an excessive computational requirement, the final orbit is finely discretized into 1,000 states equally spaced in time. If the average of the closest states is less than the average computed for the current reference trajectory that this trajectory is following, than this training trajectory is updated to be the new reference trajectory. However, once the reference trajectories converge on the target orbit, defined using a set of hyperspheres for the position and velocity deviations, the reference trajectories are then updated based on minimizing

$$J_{ref} = \sum_{t=0}^{t_{max}} \left(-1 - c_m \Delta \tilde{m}_t \right) \tag{5.1}$$

where c_m is the propellant mass coefficient assigned to each policy and Δm is the propellant mass usage caused by the most action $\bar{u}_t$. This function primarily discourages high propellant mass usages for the policies, but also inhibits long flight times for the reference trajectories. Other combinations of criteria have been explored, but minimizing the position deviations followed by

minimizing propellant mass usage in combination with minimizing the flight time has been found to produce reference trajectories with flight times and propellant mass usages comparable to transfers recovered via an optimization scheme.

This approach enables each policy to develop transfers with unique geometries subject to where the transfers depart the initial periodic orbit. Specifically, each policy produces unique reference trajectories corresponding to the distinct objective prioritization assigned to the policy through the reward function. Additionally, employing multiple reference trajectories facilitates faster transfer times, a wider variety of transfer geometries, and policies to develop locally optimal solutions for multiple segments of the initial periodic orbit. However, while many reference trajectories may be assigned for each policy, too many reference trajectories may not allow the policies to update and produce locally optimal solutions for each region of the periodic orbit. Using MRPPO while updating the reference trajectories enables policies to uncover locally optimal behavior in complex environments without the need for predefined reference trajectories, thereby reducing both the bias induced by a human trajectory designer and the time and computational resources needed to develop the reference trajectories for each policy.

5.2.1 State Vector Formulation

Once an initial condition is selected for an agent, the spacecraft information is used to form a state vector that is input to the neural networks to generate an action. Note, the assumptions of reinforcement learning, that the optimization problem may be formulated as a MDP, requires that the input state to the neural networks contains all information about the agent necessary to formulate a locally optimal action. In this research, the state vector for the neural networks is defined as a 15×1 vector written as

$$\bar{s}_t = [\bar{d}_t^T, \bar{v}_t^T, m_{sc}, \Delta\bar{d}_t^T, \Delta\bar{v}_t^T, \kappa_t, t]^T \tag{5.2}$$

where the components are standardized to be approximately in the range of [-1,1]. In this state formulation, $\bar{d}_t$ and $\bar{v}_t$ denote the nondimensionalized position and velocity vectors of the spacecraft

with respect to the barycenter of the CR3BP rotating frame; m_{sc} is the spacecraft mass; $\Delta\bar{d}_t$ and $\Delta\bar{v}_t$ represent the relative states of the spacecraft measured with respect to the initial orbit, final orbit, or reference trajectory; κ_t is an index that signifies whether the relative states are measured with respect to the initial orbit, final orbit, or reference trajectory, and t is the current nondimensional time. However, for the first trajectory design scenario, only the final orbit and reference trajectory are used to determine the relative states. Additionally, for low-thrust transfers in the Sun-Earth-Moon point mass ephemeris model, $\bar{d}_t$ and $\bar{v}_t$ are measured with respect to the Sun-Earth instantaneous rotating frame.

Each component of the state vector provides information about the spacecraft and environment to the neural networks when determining a locally optimal action. Including $\bar{d}_t$ and $\bar{v}_t$ in the state vector enables the neural networks to determine the dynamics acting on the spacecraft at the current time in the system. Meanwhile, the relative states, $\Delta\bar{d}_t$ and $\Delta\bar{v}_t$, are measured with respect to the closest periodic orbit or reference trajectory in position space unless the spacecraft is within some distance of the final orbit. Note, $\Delta\bar{d}_t$ and $\Delta\bar{v}_t$ are non-unique and supply the neural networks with a nonminimal state description. Providing the relative state information to the neural networks enables the selection of actions that minimize the state deviation from the dynamical structures, facilitates convergence on the final periodic orbit, and was observed to improve the performance of the neural networks. For the three trajectory design scenarios investigated, the position threshold is defined as 5,000 km, 10,000 km, and 7.5 $\times 10^6$ km. This position threshold with respect to the final orbit is used to prevent chattering in the relative states between a reference trajectory and final orbit. Then, κ_t corresponds to how the relative states are measured and is set as -1, 0, or 1 for the initial orbit, reference trajectory, or final orbit, respectively. Incorporating κ_t into the input state provides the neural networks with information about how the relative state is measured, and the expected reward output from the reward function formulations defined in Section 5.2.4. Finally, t is the current time measured from a defined epoch, which generates insights into the dynamics acting on a spacecraft in a point mass ephemeris model and any termination conditions that may be time dependent. This state formulation provides the neural networks information about the spacecraft

and enables the trained neural networks to draw locally optimal actions for the spacecraft.

Each of the state components input to the neural networks is gathered from the agent information and is standardized to be approximately between [-1,1]. Standardizing the state components input to the neural networks aids the convergence of the neural networks on optimal behavior and generally leads to improved performance [146]. The state vector components are standardized via

$$\breve{s}_{t,i} = \frac{2(\bar{s}_{t,i} - \breve{s}_{i,min})}{\breve{s}_{i,max} - \breve{s}_{i,min}} - 1 \tag{5.3}$$

where $\breve{s}_{t,i}$ indicates agent states after standardization, $\breve{s}_{i,max}$ represents the upper standardization value for a neural network state component, and $\breve{s}_{i,min}$ denotes the lower standardization value for a neural network state component. The standardization values are defined to ensure that a majority of the states encountered within the environment may be standardized to be between [-1,1].

The standardization values are selected using the characteristics of the trajectory design scenario. For the position and velocity components, the standardization values are computed using the maximum and minimum state components of the initial and final periodic orbits. For example, the lower standardization value of the x state component is defined using the minimum x-coordinate of either the initial or final periodic orbit, i.e. whichever orbit possesses a smaller x-coordinate. A similar process using the two periodic orbits is employed for the other position and velocity standardization values. For the mass component, the upper standardization value is set as the initial wet mass of the spacecraft while the lower standardization value is defined as approximately 75% of the spacecraft mass at the conclusion of the transfer if the spacecraft used the maximum thrust magnitude throughout the transfer. This was empirically observed across the trajectory design scenarios to produce strong and consistent behavior. For the relative state components, the standardization values are typically defined using the radii of the convergence hyperspheres set at the final orbit. For instance, if the spacecraft enters within 5×10^{-3} in position and velocity space of a state along the final orbit, the spacecraft is considered to have converged on the final orbit. Then, the standardization values for the relative position states are typically set equal to the radii of the convergence hyperspheres while the standardization values of the relative velocity

Table 5.3: Standardization values for the input vector to the neural networks for the three trajectory design scenarios.

Quantity	Scenario #1		Scenario #2		Scenario #3	
	$\breve{s}_{i,min}$	$\breve{s}_{i,max}$	$\breve{s}_{i,min}$	$\breve{s}_{i,max}$	$\breve{s}_{i,min}$	$\breve{s}_{i,max}$
x	1.0522	1.1676	0.8240	1.1676	0.3681	1.0112
y	-0.1325	0.1325	-0.1168	0.1168	-0.9562	0.0049
z	-0.1494	0.0751	-0.1029	0.0626	-0.0016	0.0021
$\dot{x}$	-0.0993	0.0993	-0.0843	0.0843	-0.1209	0.1220
$\dot{y}$	-0.2192	0.3667	-0.1973	0.2735	-0.0916	0.0841
$\dot{z}$	-0.2278	0.2278	-0.1576	0.1576	-0.0037	0.0037
$m_{s/c}$	179	180	179	180	150	180
Δx	-5×10^{-3}	5×10^{-3}	-5×10^{-3}	5×10^{-3}	-5×10^{-3}	5×10^{-3}
Δy	-5×10^{-3}	5×10^{-3}	-5×10^{-3}	5×10^{-3}	-5×10^{-3}	5×10^{-3}
Δz	-5×10^{-3}	5×10^{-3}	-5×10^{-3}	5×10^{-3}	-5×10^{-3}	5×10^{-3}
$\Delta \dot{x}$	-1×10^{-2}	1×10^{-2}	-1×10^{-2}	1×10^{-2}	-1×10^{-2}	1×10^{-2}
$\Delta \dot{y}$	-1×10^{-2}	1×10^{-2}	-1×10^{-2}	1×10^{-2}	-1×10^{-2}	1×10^{-2}
$\Delta \dot{z}$	-1×10^{-2}	1×10^{-2}	-1×10^{-2}	1×10^{-2}	-1×10^{-2}	1×10^{-2}
κ	-1	1	-1	1	-1	1
t	0	3	0	9	0	25.3

states are defined as twice the radii of the convergence hyperspheres. The standardization values of the relative velocity states are twice as large because the relative velocity states are typically an order of magnitude larger than the relative position states for the transfers explored in this research. The standardization values for the index for how the relative states are measured are defined as -1 and 1 for a periodic orbit transfer scenario. If additional dynamical structures are included, the standardization values for the index may need to be altered to reflect more than three dynamical structures. Finally, the standardization values for the time are set to zero and the maximum time allotted for the transfer. Table 5.3 displays the values for $\breve{s}_{i,max}$ and $\breve{s}_{i,min}$ for each state component for each trajectory design scenario. Using these standardization values for the neural network input state enables the networks to efficiently learn locally optimal actions for every state in the environment.

5.2.2 Action Vector Formulation

Once the current standardized state, $\bar{s}_t$, is input to the neural networks, the actor neural network generates an action that aims to maximize the long-term reward along a trajectory in the environment. Specifically, the action vector output from the actor neural network enables the spacecraft to maneuver towards the final orbit while maximizing the expected total reward in the dynamical model. The action vector is defined as a 4×1 vector written as $\bar{u}_t = [u_x, u_y, u_z, u_T]$ where the first three components determine the thrust direction and the fourth component determines the thrust magnitude. These action components are drawn from the probability distributions output by the actor neural network, which depend on the input state. Recall, throughout the training phase, action components are randomly drawn from the probability distributions while during the evaluation phase, the means of the probability distributions are used for each action component. In this research, the actor neural network is parameterized to output a Gaussian distribution to determine the action vector where the actor neural network learns and outputs the means of the Gaussian distributions. The standard deviation of the Gaussian distribution for each action component is initialized to be 0.5 to ensure sufficient exploration early in the training process. The standard deviation is then annealed throughout training via a linear decrease following the update to the policies until reaching a value of zero at the conclusion of training. Figure 5.6 depicts the Gaussian distribution for a single action component for an input state where the action component may be drawn randomly from the distribution during training and set equal to the mean during the evaluation phase. Once an action vector is drawn, the action components are bounded depending on the quantity that each component represents.

Although the actor neural network may produce an unbounded action vector, it must be bounded to conform to the hardware and operational constraints of the spacecraft. The first three components of an action vector are normalized to produce a unit vector, $\hat{u}_t$, to determine the thrust direction in the CR3BP rotating frame or Sun-Earth instantaneous rotating frame. The

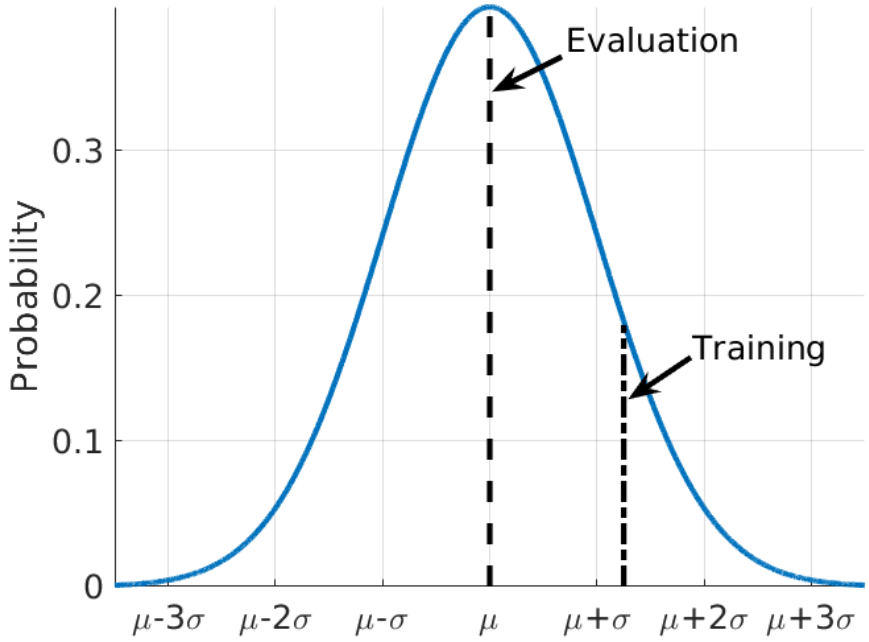


Figure 5.6: Sample Gaussian distributions for an action component output by an actor neural network from an input state.

thrust direction unit vector is computed via

$$\hat{u}_t = \frac{[u_x, u_y, u_z]}{\sqrt{u_x^2 + u_y^2 + u_z^2}} \tag{5.4}$$

and used in Eqs. (2.40)-(2.42). Then, the thrust magnitude component is interpreted differently depending on whether the spacecraft is equipped with a variable thrust or constant thrust engine. For a variable thrust engine, the thrust magnitude $\tilde{T}$ used in Eq. (2.39) is computed via

$$\tilde{T} = \tilde{T}_{max}\left(\frac{\tanh(u_T) + 1}{2}\right) \tag{5.5}$$

where the hyperbolic tangent function first constrains the thrust magnitude component, u_T to be between -1 and 1. Then, the component is normalized to be between 0 and 1, which is then multiplied by $\tilde{T}_{max}$ to produce the actionable thrust magnitude for the specified thrust direction in the CR3BP rotating frame or Sun-Earth instantaneous rotating frame. Conversely, for a constant thrust engine configuration, the sign of the thrust magnitude component, u_T determines whether the low-thrust engine is activated; if the thrust magnitude component is greater than 0, than the

low-thrust engine is activated, else the spacecraft engine is deactivated and the spacecraft is in coast mode. This formulation helps the neural networks determine when to activate the low-thrust engine since the outputs from the actor neural network are typically in the range of [-1,1]. Then, the actor neural network may shift the mean of the Gaussian distribution for the fourth action component to either side of zero depending on whether the agent should perform a maneuver. For a constant thrust engine configuration, the thrust magnitude is determined via

$$\tilde{T} = \begin{cases} \tilde{T}_{max}, & \text{if } (u_T > 0) \\ 0, & \text{if } (u_T \leq 0) \end{cases} \tag{5.6}$$

Once a state-action pair is generated, the spacecraft is propagated for a defined time step using the equations of motion specified for the trajectory design scenario.

5.2.3 Time Step Definition

Since reinforcement learning is formulated in the context of an MDP, the continuous time trajectory design problem must be transformed into a discrete time process. In a discrete time formulation, a state-action pair for an agent is propagated for a defined time step, denoted Δt, and then subsequently assigned a reward, new state vector, and new action vector to perform. A discrete time formulation may not recover the precise action to perform at every point in time that a continuous time formulation for a classical optimization scheme may recover. However, a discrete time formulation may more closely match operational requirements only allowing the spacecraft to perform a new maneuver after a set amount of time. These operational requirements may stem from discouraging large changes in the thrust direction vector over a small amount of time, limited communication windows, or preventing maneuvers from interfering with scientific observations. Further, it has been observed that neural networks learn more rapidly when Δt is larger and there are less steps in the environment due to the need to learn less combinations of state-action pairs to find an optimal solution that converges on the final orbit. However, smaller time steps may allow the recovered solution to sufficiently resemble a continuous time solution recovered via a classical

Table 5.4: Propagation time and maximum number of time steps for the trajectory design scenarios.

Scenario	Δt	t_{max}
# 1	3×10^{-2}	100
# 2	6×10^{-2}	150
# 3	0.22	115

optimization scheme, and enable the neural networks to slightly reduce the required propellant mass and flight time to arrive at the final orbit. Thus, the size of Δt must be balanced to enable the neural networks to learn the control profiles that solve the reinforcement learning problem, converge on the final orbit, and limit the required propellant mass usage and flight time.

In this research, Δt is generally set to allow the neural networks a maximum of 100-150 time steps to converge on the final orbit. This maximum number of time steps for a trajectory, denoted t_{max}, is empirically determined to enable the neural networks to learn to converge on the final orbit with a high degree of control without requiring the neural networks to learn a significant number of states in the environment. Additionally, the neural networks generally only reach the maximum number of time steps early on in training prior to the networks learning to converge on the final orbit. Table 5.4 denotes the discrete time step size and maximum number of time steps for the three trajectory design scenarios explored in this research where the propagation time is nondimensionalized using the CR3BP characteristic time, $\tilde{t}^*$. Then, each state-action pair is propagated in the dynamical model forward in time for Δt nondimensional time units and state-action pairs are continually generated until one or more of the termination conditions is activated.

5.2.4 Reward Function Formulation

Each policy trained by MRPPO attempts to maximize the expected total reward for a trajectory returned from a reward function. The reward functions are structured to encourage the

policies to balance multiple objectives in the dynamical model; specifically targeting the final orbit while minimizing propellant mass usage and flight time. Unlike in constrained optimization techniques, in reinforcement learning, reaching the target orbit is formulated as a goal that is maximized rather than a hard constraint in this proof of concept approach. Additionally, the terms in the reward functions are scaled to be approximately on the order of 10^0 throughout the transfer for the three trajectory design scenarios. This scaling enables the terms in the reward functions to have approximately similar influences throughout the environment. Then, by selecting the relative weighting between propellant mass usage and flight time, the policies recover distinct regions of the multi-objective solution space associated with propellant mass usage and flight time. However, unlike in a conventional optimization problem, the reward function cannot be set to maximize the final mass or to have too high of a propellant mass usage penalty because the policies would then be incentivized to not perform any actions and stay along the initial orbit. While the reward function formulations for each trajectory design scenario are similar, there are notable differences due to the spacecraft parameters, propagation time, and scenario assumptions. The specific reward function definitions are outlined in the following sections.

Scenario # 1: For the first trajectory design scenario, the neural networks are tasked with constructing a transfer from an L_2 southern halo orbit to a higher-energy L_2 southern halo orbit in the Earth-Moon CR3BP, which is reflected in the reward function formulation. The reward function for this trajectory design scenario is written as

$$r_t(\bar{s}_t, \bar{u}_t) = \begin{cases} -1 - c_m(\tilde{m}_{sc,t} - \tilde{m}_{sc,t+\Delta t}) + \Omega & \text{Reference Trajectory} \\ -10|\Delta\bar{d}_{t+\Delta t}| - |\Delta\bar{v}_{t+\Delta t}| - c_m(\tilde{m}_{sc,t} - \tilde{m}_{sc,t+\Delta t}) + \Omega & \text{Final Orbit} \end{cases} \tag{5.7}$$

where c_m denotes the mass coefficient which scales the penalty on propellant mass usage, Ω is the bonus or penalty associated with the termination conditions, and the reward depends on whether the location of the spacecraft is measured with respect to the reference trajectory or the final orbit. When the location of the spacecraft is measured with respect to the reference trajectory, the reward is composed of a propellant mass usage term, termination condition terms, and a constant

penalty of -1 to incentivize the policies to rapidly approach the final orbit. Although the policy is not incentivized to follow the reference trajectory closely, the reference trajectory does still supply information about targeting the final orbit through the position and velocity deviations terms in the state vector. Once the spacecraft enters the vicinity of the final orbit, i.e., $\kappa = 1$, the reward function incentivizes the policy to decrease the state discontinuities with respect to the final orbit while also conserving propellant mass. In both cases, the propellant mass usage term is calculated using the most recent action, $\boldsymbol{u}_t$, and is scaled by c_m to reflect a distinct relative weighting between minimizing propellant mass and minimizing flight time for each policy while reaching the final orbit. Specifically, four policies are trained with mass coefficients of $c_m = [0, 13.33, 26.66, 40]$, corresponding to increasingly larger penalties for propellant mass usage along a low-thrust-enabled transfer. With this formulation of the reward function, the variable mass coefficient enables recovery of solutions spanning the multi-objective trade space for this trajectory design scenario.

Scenario # 2: For the second trajectory design scenario, the reward functions are structured based on whether the location of the spacecraft is measured with respect to the initial orbit, final orbit, or reference trajectory. In this scenario, it was observed that rewarding the policies to decrease state deviations with respect to the reference trajectory aided convergence in this more complex scenario. The general reward function formulation for this trajectory design scenario is written as

$$r_t(\bar{s}_t, \bar{u}_t) = \begin{cases} -8 - 10|\Delta\bar{d}_{t+\Delta t}| - |\Delta\bar{v}_{t+\Delta t}| - c_m(\tilde{m}_{s/c,t} - \tilde{m}_{s/c,t+\Delta t}) + \Omega & \text{Initial Orbit} \\ -4 - 10|\Delta\bar{d}_{t+\Delta t}| - |\Delta\bar{v}_{t+\Delta t}| - c_m(\tilde{m}_{s/c,t} - \tilde{m}_{s/c,t+\Delta t}) + \Omega & \text{Reference} \\ -100|\Delta\bar{d}_{t+\Delta t}| - 10|\Delta\bar{v}_{t+\Delta t}| - c_m(\tilde{m}_{s/c,t} - \tilde{m}_{s/c,t+\Delta t}) + \Omega & \text{Final Orbit} \end{cases} \tag{5.8}$$

which includes an additional case if the location of the spacecraft is measured with respect to the initial orbit. Specifically, the four policies are assigned the propellant mass usage coefficients, $c_m = [0, 83.33, 166.66, 250]$, respectively, whereby higher coefficients increase the penalty associated with propellant mass usage. Additionally, the scalar penalties at each time step, i.e. -8, -4, and 0, decrease as the spacecraft transfers from the initial orbit to the reference trajectory and then to

the final orbit to encourage targeting the final orbit. Finally, the coefficients on the state deviation terms are lower for the initial orbit and reference trajectory components to penalize large deviations while the coefficients for the state deviation terms for the final orbit component are increased to encourage convergence to the final orbit. This formulation enables the policies to learn to follow the dynamical arcs within the system to target the final orbit while also limiting flight time and propellant mass usage.

Scenario # 3: The third trajectory design scenario uses a similar reward function formulation as the second trajectory design scenario defined in Eq. (5.8) but with distinct propellant mass usage coefficients. Instead, the eight policies trained for each dynamical and engine model are assigned $c_m = [0, 9/7, 18/7, \ldots, 9]$, respectively, whereby higher coefficients increase the penalty associated with propellant mass usage. The propellant mass usage coefficients are altered to reflect the increase in propagation time between state-action pairs, the smaller maximum thrust magnitude of the low-thrust engine, and the distinct dynamics of the Sun-Earth system. This general formulation encourages the policies to target the final orbit while balancing flight time and propellant mass usage for a variety of dynamical and engine models. The resulting policies are used to uncover regions of the multi-objective solution space that balance minimizing flight time and propellant mass usage.

To aid the convergence behavior of the neural networks in the trajectory design scenarios, the rewards generated for each policy are standardized to possess a mean of zero and standard deviation equal to unity. Standardizing the rewards before every update to the neural networks encourages the neural networks to target the best state-action pairs generated throughout training and has been shown to improve the performance of policies [146]. Since each policy for each trajectory design scenario generates distinct rewards for identical state-action pairs due to the unique propellant mass usage coefficients, the standardization is performed for each policy. The rewards for each policy are standardized via

$$r_t = \frac{r_t - \text{mean}(r_{0:\Psi})}{\text{stddev}(r_{0:\Psi})} \tag{5.9}$$

where $r_{0:\Psi}$ denotes the rewards for policy P_i associated with the state-action pairs collected by all policies for one update phase for the policies, and mean($r_{0:\Psi}$) and stddev($r_{0:\Psi}$) compute the mean and standard deviation of the rewards for that policy, respectively. The standardized rewards are then input to the estimated advantage and objective functions to update the neural networks.

5.2.5 Termination Conditions

Termination conditions are defined for each trajectory design scenario corresponding to the termination of a trajectory in the environment. Once a termination condition has been activated, a bonus or penalty is applied depending on whether the spacecraft has converged upon the final orbit or a constraint has been violated. In each trajectory design scenario, policies and agents are rewarded for converging on the final orbit within defined position and velocity hyperspheres of the closest state in position space along the final periodic orbit. The final periodic orbit in each trajectory design scenario is discretized into 1,000 states equally spaced in time. The time step for each discretization is significantly less than the propagation time for each step of the agent enabling the spacecraft to determine that it has converged on the final orbit. However, discretizing the final orbit into too many states significantly increases the required computational time to determine the closest state along the final orbit in position space. Since this must be performed for every state generated by the policies across all updates, a balanced approach is used to limit the required computational time while still enabling an accurate determination of convergence. Additionally, for each trajectory design scenario, the agent is allowed a maximum number of time steps to converge on the final orbit. If the agent does not reach the final orbit, the trajectory is terminated and a penalty is applied.

Scenario # 1: For the first trajectory design scenario, four termination conditions for a trajectory are defined:

(1) The spacecraft approaches within 1×10^{-3} and 5×10^{-3} in position and velocity space, or 384 km and 5.1 m/s respectively, of the closest state in position space of the final halo orbit where $\Omega = 100$.

(2) The maximum number of time steps, 100, is reached and $\Omega = 0$.

(3) The spacecraft drifts farther than 7,500 km from both the final orbit and reference trajectory, selected to allow the policies to explore the environment but not so large as to depart the vicinity of the initial and final orbits, where $\Omega = -1000$.

(4) The spacecraft impacts the Moon where $\Omega = -1000$.

These termination conditions enable the policies to explore the environment and exploit multiple geometries to reach the final orbit while limiting exploration of the state space that is significantly outside the region of the periodic orbits.

Scenario # 2: Similarly, the termination conditions for a trajectory in the second trajectory design scenario are:

(1) The spacecraft is within 5×10^{-3} in position and velocity of the closest state in position space of the final halo orbit and $\Omega = 1000$.

(2) The maximum number of time steps, 150, is reached and $\Omega = 0$.

(3) The spacecraft strays over 12,500 km from the initial orbit, final orbit, and reference trajectory and $\Omega = -1000$.

(4) The spacecraft enters within 5 Moon radii of the Moon and $\Omega = -1000$.

These penalties discourage the spacecraft from departing the system or impacting the Moon while the bonus rewards convergence to the final orbit. Both the first and second trajectory design scenarios are within the Earth-Moon CR3BP so are assigned similar termination conditions.

Scenario # 3: For the final trajectory design scenario in the Sun-Earth system, the termination conditions are defined as:

(1) The spacecraft converges within a defined hypersphere in position and velocity of the closest state in position space along the L_5 short period orbit and $\Omega = 1000$.

(2) The maximum number of time steps, 115, is reached and $\Omega = 0$.

(3) The spacecraft diverges farther than 0.205 in nondimensional position units from the initial orbit, final orbit, and reference trajectory and $\Omega = -1000$.

(4) The spacecraft impacts the Earth or Moon and $\Omega = -1000$.

For the Sun-Earth CR3BP, the position and velocity hyperspheres are set as 5×10^{-3} while for the Sun-Earth-Moon point mass ephemeris model, the position and velocity hyperspheres are defined to be 1×10^{-2}. The position and velocity hyperspheres are increased for the ephemeris model to aid the neural networks in converging on the final orbit in the more complex dynamical model while still achieving bounded motion near the Sun-Earth L_5 equilibrium point.

5.2.6 Hyperparameter Selection

Values are assigned to the hyperparameters and construction parameters that influence the behavior of the policies in each trajectory design scenario. Recall, the hyperparameters influence the training process for the policies while the construction parameters determine how the actor and critic neural networks are initialized. Further, Figs. 5.7 and 5.8 depict the roles of the actor and critic in a trajectory design scenario determining the action to perform and value at every state in the environment.

Scenario # 1: In the first trajectory design scenario, a set of baseline hyperparameters is defined to examine the behavior of MRPPO and serve as a comparison prior to performing a hyperparameter exploration. Table 5.5 summarizes the hyperparameter values used as a baseline for the first trajectory design scenario, prior to hyperparameter exploration. In the hyperparameter exploration, a one dimensional search along a single hyperparameter is performed while all other hyperparameters are set to the baseline values. This exploration provides insights into the influence of certain hyperparameters on the behavior of the policies throughout training. Notably, the learning rate is adjusted throughout training following a triangular cycle which was demonstrated by Andrychowicz et al. and Smith and Topin to aid the convergence of the policies [6, 131].

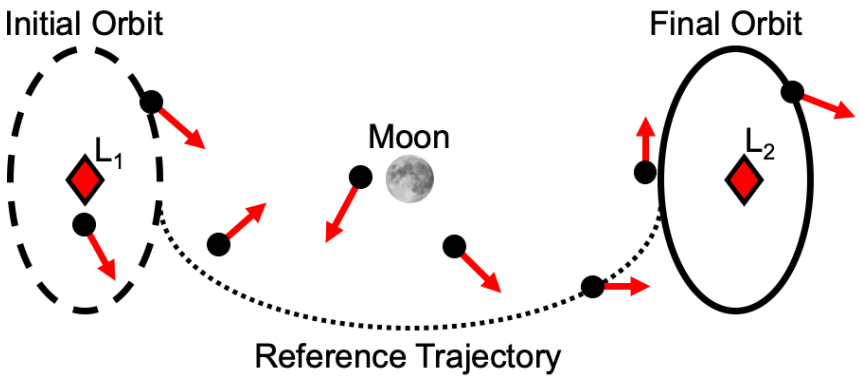


Figure 5.7: Demonstration of an actor outputting action vectors for a spacecraft to perform in a representative trajectory design problem.

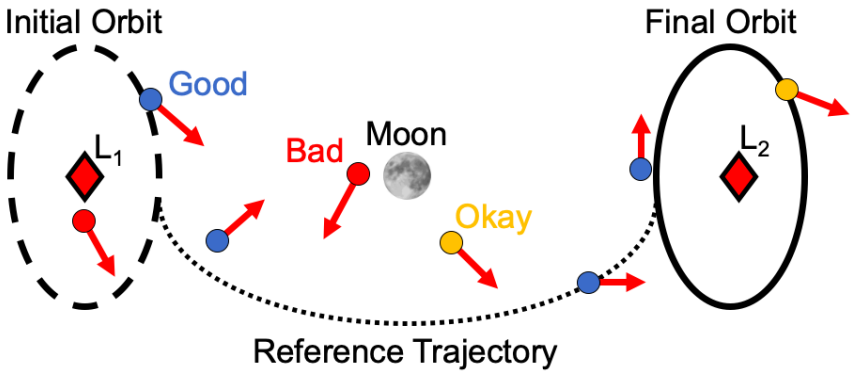


Figure 5.8: Demonstration of a critic generating values for spacecraft states in a representative trajectory design problem.

The baseline hyperparameters and construction parameters are selected due to their tendency to produce relatively stable performance during training across a variety of applications and many trajectory design scenarios as demonstrated by Andrychowicz et al., Bonasera et al., Henderson et al., and Sullivan et al. [6, 19, 66, 141].

Scenario # 2: Following the hyperparameter exploration detailed in Section 7.1.4, the hyperparameters are modified for the second and third trajectory design scenarios. Table 5.6 displays the hyperparameters and construction parameters used for the second trajectory design scenario. Namely, three parameters are altered. The GAE factor is increased from 0.85 to 0.9 which increases the variance and lowers the bias in the bias-variance trade-off. Additionally, the number

Table 5.5: MRPPO baseline hyperparameter and construction parameter values for the first trajectory design scenario.

Quantity	Value
Update Phases, Z	500
Environmental Steps, Λ	4096
Epochs, E	5
Mini Batches, M	4
Discount Factor, γ	0.95
GAE Factor, λ	0.85
Initial Learning Rate, l_r	1×10^{-3}
Maximum Learning Rate, l_r	5×10^{-3}
Clipping Parameter, ε	2×10^{-3}
Value Function Coefficient, c_1	0.5
Entropy Coefficient, c_2	1×10^{-3}
Maximum Gradient Norm, c_{max}	0.5
Actor Hidden Layers, l_a	2
Actor Hidden Nodes per Layer, n_a	64
Critic Hidden Layers, l_c	2
Critic Hidden Nodes per Layer, n_c	1024
Initialization Scheme	Orthogonal
Activation Function	Tanh

of critic hidden layers is decreased from 2 to 1 decreasing the total number of training parameters for estimating the value function. Finally, the activation function is modified to ReLU instead of the hyperbolic tangent since ReLU is found to generate higher and most consistent total rewards.

Scenario # 3: In the final trajectory design scenario, the hyperparameters are adjusted for this more complex trajectory design scenario. Table 5.7 displays the hyperparameters and construction parameters used for the third trajectory design scenario. The number of hidden layers and hidden nodes per hidden layer for the critic neural network are updated to 2 and 64, respectively, which produced more stable convergence behavior. Additionally, although the number of policies is increased from four to eight while the number of agents per policy is held constant. This enables each policy to collect a significant amount of information in the environment via agents the policy controls while still learning from the state-action-reward experiences gathered by other policies.

Table 5.6: MRPPO baseline hyperparameter and construction parameter values for the second trajectory design scenario.

Quantity	Value
Update Phases, Ψ	500
Environmental Steps, Λ	4096
Epochs, E	5
Mini Batches, M	4
Discount Factor, γ	0.95
GAE Factor, λ	0.9
Initial Learning Rate, l_r	1×10^{-3}
Maximum Learning Rate, l_r	5×10^{-3}
Clipping Parameter, ε	2×10^{-3}
Value Function Coefficient, c_1	0.5
Entropy Coefficient, c_2	1×10^{-3}
Maximum Gradient Norm, c_{max}	0.5
Actor Hidden Layers, l_a	2
Actor Hidden Nodes per Layer, n_a	64
Critic Hidden Layers, l_c	1
Critic Hidden Nodes per Layer, n_c	1024
Initialization Scheme	Orthogonal
Activation Function	ReLU
Agents per Policy	4

5.3 Transfer Learning Implementation

Transfer learning is a technique used within the machine learning community that enables neural networks trained for one environment to be used as an initial guess for policies in a new environment [151, 168]. Transfer learning has been employed in a variety of domains and applications included autonomous driving, guidance and control for drones, biological systems, and supervised learning [159, 168]. When applied to a reinforcement learning problem, the environment may be updated to possess a distinct dynamical model, a new set of objectives, changes to the agents, or a combination of the three components. While transfer learning typically assumes that the new environment is similar to the original environment that the policies are first trained on, this is not a requirement. However, more similar environments typically require less updates to the policies

Table 5.7: MRPPO baseline hyperparameter and construction parameter values for the third trajectory design scenario.

Quantity	Value
Update Phases, Ψ	500
Environmental Steps, Λ	4096
Epochs, E	5
Mini Batches, M	4
Discount Factor, γ	0.95
GAE Factor, λ	0.9
Initial Learning Rate, l_r	1×10^{-3}
Maximum Learning Rate, l_r	5×10^{-3}
Clipping Parameter, ε	2×10^{-2}
Value Function Coefficient, c_1	0.5
Entropy Coefficient, c_2	1×10^{-3}
Maximum Gradient Norm, c_{max}	0.5
Actor Hidden Layers, l_a	2
Actor Hidden Nodes per Layer, n_a	64
Critic Hidden Layers, l_c	2
Critic Hidden Nodes per Layer, n_c	64
Initialization Scheme	Orthogonal
Activation Function	ReLU
Agents per Policy	4

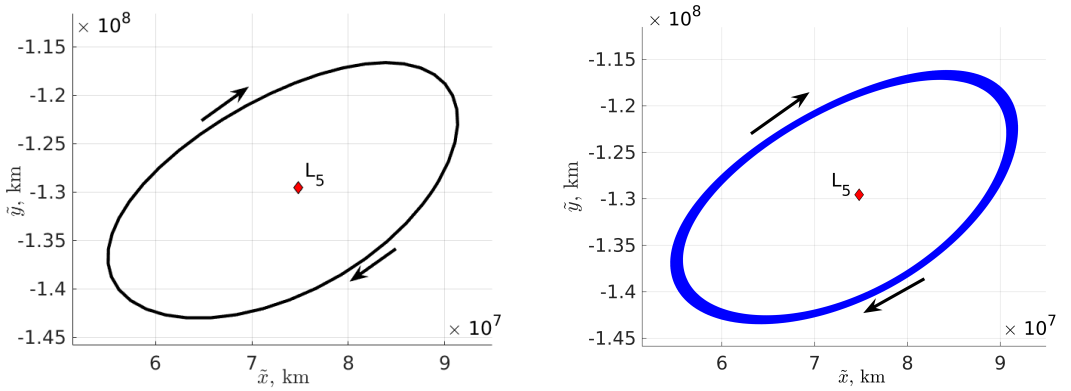
to recover a locally optimal solution. Further, in environments where significant data collection may be time- and cost-prohibitive, transfer learning may be applied to enable policies trained in an approximate, cheaper model to serve as valuable initial guesses for the more complex environment. Transfer learning may be employed to reduce the required computational times, facilitate the recovery similar locally optimal solutions in more complex environments, and iteratively update policies to maximize the objectives in complex trajectory design scenarios.

In this research, transfer learning is used to train policies in a higher-fidelity ephemeris model using policies trained in the CR3BP as an initial guess similar to the approach used by Bonasera et al. [18]. This approach is similar to employing continuation across dynamical models, a technique commonly used when designing transfers first in a lower-fidelity model prior to updating the transfer

for a higher-fidelity model. However, since the weights, biases, and structure of the neural networks are held constant in the transfer learning approach, the state and action vector formulations must also be identical between dynamical models. An important note when applying this to trajectory design is that some input state variables do not influence the actions during the training in the lower-fidelity model, but they are necessary to include for the higher-fidelity dynamical model. Then, while the time component in the state vector formulation in Eq. (5.2) for the CR3BP, which is an autonomous dynamical model, may not influence the dynamics of the spacecraft or the decision of the neural networks in the environment, the time is necessarily included for the neural network structure for the ephemeris model dynamics.

Specifically, transfer learning is used to train policies to recover low-thrust transfers in the Sun-Earth-Moon point mass ephemeris model using policies from the Sun-Earth CR3BP as an initial guess for the third trajectory design scenario. First, policies are trained in the Sun-Earth CR3BP to recover low-thrust transfers to a Sun-Earth L_5 short period orbit. Then, the neural networks constituting the policies are used as initial guesses for developing transfers in the Sun-Earth-Moon point mass ephemeris model. Once policies have been trained for both the variable thrust and constant thrust engine configurations in the Sun-Earth CR3BP, these policies are used as an initial guess to develop locally optimal trajectories in the Sun-Earth-Moon point mass ephemeris model using transfer learning.

To limit the changes between the dynamical models in this transfer learning approach, the final periodic orbits in both models are assumed to be similar. However, in this approach, the Sun-Earth L_5 short period orbit is no longer periodic in the Sun-Earth-Moon point mass ephemeris model. Instead, 1,000 states discretized along the Sun-Earth L_5 short period orbit are used to formulate natural trajectories in the Sun-Earth-Moon point mass ephemeris model that are generated to resemble the Sun-Earth L_5 short period orbit in the higher-fidelity model at each epoch. The states and rewards are measured with respect to the closest state in configuration space corresponding to the current epoch along each natural trajectory. Figure 5.9 depicts the 1,000 trajectories in the Sun-Earth-Moon point mass ephemeris model for the 115 epochs defined for the reinforce-



(a) Sun-Earth L_5 short period orbit

(b) Approximated Sun-Earth L_5 short period orbit

Figure 5.9: Optimized trajectories in the Sun-Earth-Moon point mass ephemeris model approximating the Sun-Earth L_5 short period orbit.

ment learning environment. Section 6.4 details the optimization process for generating the natural transfers that approximate the Sun-Earth L_5 short period orbit in the Sun-Earth-Moon point mass ephemeris model.

The weights and biases of the neural networks used to construct the policies in the CR3BP are used to initialize the neural networks in the Sun-Earth-Moon point mass ephemeris model. Then, the neural networks are updated using state-action-reward experiences generated in the Sun-Earth-Moon point mass ephemeris model. While the state, action, and reward function formulations are equivalent between the two dynamical models, the original MRPPO hyperparameters in Table 5.7 are modified. Instead, the policies are trained for 25 updates in the Sun-Earth-Moon point mass ephemeris model with an initial learning rate of 1×10^{-4} and a maximum learning rate of 1.1×10^{-4} following a cyclic annealing process. The learning rate is decreased for the transfer learning approach to prevent the state-action-reward experiences generated in the Sun-Earth-Moon point mass ephemeris model from destabilizing the policies trained in the Sun-Earth CR3BP.

Chapter 6

Transforming a Reinforcement Learning Problem to a Nonlinear Constrained Optimization Problem

Two nonlinear constrained optimization schemes are formulated to: 1) validate the low-thrust transfers generated by policies trained using MRPPO and 2) develop natural trajectories in the Sun-Earth-Moon point mass ephemeris model that approximate the configuration of a Sun-Earth L_5 short period orbit. For the first task, the local optima recovered by reinforcement learning approaches have been validated in a variety of domains and applications using distinct optimization techniques. For example, reinforcement learning approaches have surpassed human experts in domains such as Go, Atari, and Chess [27, 130]. Additionally, reinforcement learning has compared favorably against dynamic programming, genetic algorithms, and classical optimization techniques in other applications and industries [12, 82, 143]. A variety of optimization techniques have been employed within the astrodynamics community including classical optimization approaches and genetic algorithms, both of which possess single and multi-objective formulations. For this research, a nonlinear, constrained, gradient descent optimization technique employing a multiple shooting algorithm in combination with SQP implemented in *fmincon* in MATLAB© is used.

The gradient descent optimization algorithm is assigned a single objective to maximize which may be composed of multiple objectives via a weighted sum, similar to the reward function for a policy trained using MRPPO. The optimization problem for validating the transfers developed by MRPPO is formulated to match as closely as possible to the reinforcement learning problem to facilitate a fair and near-equivalent comparison. Generating locally optimal solutions using the

optimization scheme enables the validation of the results produced by MRPPO for each trajectory design scenario and comparisons between the solution spaces uncovered by the two methodologies.

6.1 Multiple Shooting

Shooting methods are one approach to solving a two point boundary value problem by converting the boundary value problem into an initial value problem [107]. One method for implementing a shooting method is through a free variable and constraint vector formulation where the free variables may be altered and constraints which must be satisfied. A multiple shooting method discretizes a trajectory into multiple arcs, which are simultaneously updated to both satisfy boundary constraints and continuity and other constraints [23]. This is a common technique used by a variety of researchers for recovering trajectories in multi-body systems [23, 48, 137]. Multiple shooting methods are more computationally expensive to evaluate and possess more complex implementations, but are generally more robust in complex scenarios. In this research, a multiple shooting scheme is used to correct a trajectory due to its: 1) robust convergence properties in chaotic environments, 2) suitability for defining equality constraints used within SQP in *fmincon*, and 3) similarity to the reinforcement learning episode definition.

In a multiple shooting approach, a trajectory is discretized into a sequence of arcs which are simultaneously updated to satisfy a set of predefined constraints. Figure 6.1 depicts a representation of using multiple shooting to correct a discontinuous initial guess into a continuous trajectory where blue arcs denote coast segments and red arcs denote low-thrust segments. Each arc along an trajectory is described by an initial state, propagation time, and any other required parameters. The initial set of arcs, i.e. the initial guess, may be continuous or discontinuous and is defined using a set of free variables. The free variables describing each arc are encapsulated within the free variable vector, denoted $\bar{V}$, which is iteratively updated using a quasi-Newton approach. Further, the free variables are updated to satisfy a set of constraints, which are defined in the constraint vector, designated $\bar{F}(\bar{V})$. These constraints may include full state continuity between each arc, targeting boundary conditions, enforcing the thrust direction to be a unit vector, and other constraints

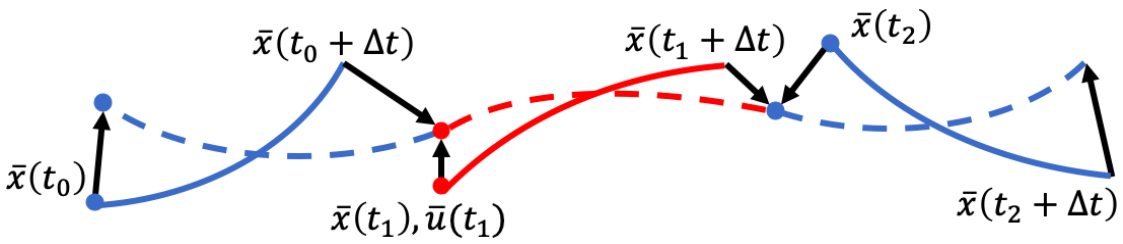


Figure 6.1: Representative diagram of a trajectory updated via multiple shooting.

required for a specific trajectory design scenario. The update to the free variable vector for when there are fewer constraints than free variables is written as

$$\bar{V}_{i+1} = \bar{V}_i - D\bar{F}(\bar{V}_i)^T (D\bar{F}(\bar{V}_i) D\bar{F}(\bar{V}_i)^T)^{-1} \bar{F}_i(\bar{V}_i) \tag{6.1}$$

where i denotes the current iterative update and $D\bar{F}(\bar{V}_i)$ is a matrix composed of the partial derivatives of the constraints taken with respect to each free variable. This matrix is computed using a combination of analytical and numerically-approximated derivatives, calculated via a first-order central difference.

6.2 Constrained Optimization via Sequential Quadratic Programming

Within *fmincon*, multiple optimization techniques may be used to recover locally optimal solutions including (1) SQP, (2) Interior Point, and (3) Trust Region Reflective [94]. In this paper, SQP in *fmincon* is used due to the demonstrated effectiveness of this method in a variety of trajectory design scenarios by Betts, Parrish and Scheeres, and Hargraves and Paris [16, 111, 62]. SQP methods formulate the optimization problem as a quadratic programming problem and solutions are iteratively updated until an optimal solution is found [17]. Additionally, SQP algorithms may handle nonlinear constraints and do not require the initial guess to be feasible prior to generating an optimal solution [17].

6.3 Constrained Optimization of Low-Thrust Transfers

A nonlinear constrained optimization scheme is formulated to validate the reference trajectories generated by policies trained using MRPPO in each trajectory design scenario. Since the multiple shooting algorithm inside of SQP requires an initial guess, the solutions developed by MRPPO serve as the initial guess prior to optimization. Note, only transfers generated using a variable thrust engine in the CR3BP are compared to the optimization scheme. This provides validation for the results produced by MRPPO on all three trajectory design scenarios and insight into the solution space expected for transfers employing a constant thrust engine configuration or in the Sun-Earth-Moon point mass ephemeris model.

To validate the results produced by MRPPO, the principal components of a multiple shooting scheme employed in combination with SQP in *fmincon* may be formulated to replicate the reinforcement learning environment. First, the free variable vector is constructed to resemble the sequential decision making process a policy follows in a reinforcement learning environment. Then, the constraint vector is designed to match the constraints inherent to a reinforcement learning environment, namely enforcing full state continuity between arcs and enforcing the thrust direction specified at each arc to be a unit vector. Finally, when formulating a multiple shooting scheme with SQP, an objective function is also defined. The objective function is minimized for a trajectory, and is used within SQP to update the free variables and recover a locally optimal solution.

In a multiple shooting algorithm, nodes are distributed at the beginning of arcs, which is similar to the states along a trajectory in a reinforcement learning environment where a policy determines an action to perform. In this optimization scheme, the characteristics of the initial guess including the low-thrust propagation time, maximum available thrust, and thrust vector formulation, are consistent between the MRPPO and SQP implementations. However, a completely equivalent implementation between MRPPO and SQP is not possible because of several notable differences. Namely, SQP implemented in *fmincon* requires the objective function and constraints to be continuous and possess continuous first order derivatives. The reward function used by MRPPO

in each trajectory design scenario violates these requirements. Then, two objective functions similar to the reward function maximized by MRPPO are explored using SQP implemented in *fmincon* for each trajectory design scenario. The objective functions are formulated to solely maximize either: 1) an augmented total reward and 2) the final mass. In both formulations, state and action continuity is enforced via constraints. Note, in a reinforcement learning algorithm, state and action continuity is guaranteed along a trajectory due to sequential propagation leading to another difference between implementations. While the recovered solution spaces are expected to be slightly different due to the difference in objective functions and implementation distinctions, the comparison facilitates insights into the validity of the multi-objective solution space recovered by MRPPO.

6.3.1 Free Variables

The free variable vector is parameterized to describe two distinct components of a trajectory: (1) a low-thrust segment based on a reference trajectory produced by MRPPO and (2) a coast segment along the final periodic orbit. The first component of the trajectory models the low-thrust transfer produced by MRPPO specified in the reinforcement learning environment. The coast segment is included to determine whether the optimization scheme converged on the final orbit. Specifically, the coast segment is propagated from the termination of the low-thrust segment and targets a defined state along the final periodic orbit. By targeting a state along the final periodic orbit using a coast segment, the low-thrust transfer is considered to have converged on the final periodic orbit.

Both the low-thrust and coast segments are discretized into multiple arcs that are described by nodes located at the initial states of the arcs. The transfers are constrained to begin from the same initial conditions as the reference trajectories generated by MRPPO to facilitate direct comparisons between the solutions. The position and velocity for the initial conditions are randomly perturbed from a state along the initial periodic orbit in all three trajectory design scenarios. Then, the position, velocity, and mass of the initial condition are not included as free variables in the

free variable vector for the initial node. Instead, only the action components for this node, $\bar{u}_1$, are included as free variables for the initial node and written as

$$\bar{N}_{LT,1} = [u_x, u_y, u_z, u_T]^T \tag{6.2}$$

where u_x, u_y, u_z are the components of the thrust direction unit vector specified in the CR3BP rotating frame and u_T is the thrust magnitude component when multiplied by $\tilde{T}_{max}$ along the thrust direction. Note, u_T is bounded between $[0, 1]$.

For all proceeding low-thrust nodes, the position, velocity, and mass of the spacecraft at the beginning of the low-thrust arc are included as free variables for the node in addition to the control components of the low-thrust engine. Then, all proceeding low-thrust nodes are written as

$$\bar{N}_{LT,i\geq 2} = \left[x, y, z, \dot{x}, \dot{y}, \dot{z}, m_{s/c}, u_x, u_y, u_z, u_T\right]^T \tag{6.3}$$

enabling the optimization scheme to significantly alter the solution from the initial guess constructed using MRPPO. Note, the propagation time is not included as a free variable to ensure an equivalent comparison to the reinforcement learning environment, which assumes discrete time steps with the same propagation time. Additionally, this may be representative of operational constraints on the spacecraft allowing a change in the thrust direction only after a specified time.

For the coast segment that enables determining if the solution has converged on the final orbit, the position, velocity, and propagation time are all included as free variables for each node. The free variables for the coast nodes are written as

$$\bar{N}_C = [x, y, z, \dot{x}, \dot{y}, \dot{z}, \Delta t]^T \tag{6.4}$$

Unlike the low-thrust arcs, the free variables for the coast arcs do include the propagation time to enable the optimization scheme to alter where the solution arrives along the final periodic orbit. Additionally, the mass of the spacecraft is not included as a free variable because in the CR3BP equations of motion, the mass of the spacecraft is assumed to not influence the ballistic trajectory. Finally, the entire free variable vector is written as

$$\bar{V} = \left[\bar{N}_{LT,1}, \bar{N}_{LT,2}, \ldots, \bar{N}_{LT,n}, \bar{N}_{C,1}, \ldots \bar{N}_{C,q}\right]^T \tag{6.5}$$

where n denotes the number of low-thrust arcs and q is the number of coast arcs along the final periodic orbit. This free variable vector formulation facilitates comparisons between the solutions produced by MRPPO and an optimization scheme combining a multiple shooting corrections scheme with SQP in *fmincon*.

6.3.2 Equality Constraints

Unlike the MRPPO implementation where soft constraints are incorporated via penalties in the reward function, hard constraints are explicitly defined and satisfied by the solutions to the constrained optimization problem. The constraint vectors for the three trajectory design scenarios are formulated to ensure full state continuity between all arcs, mass continuity between low-thrust arcs, and unity magnitude for the thrust direction for each low-thrust arc. Full state continuity is enforced via

$$\bar{c}_i = \bar{x}_{i,F} - \bar{x}_{i+1,0} \tag{6.6}$$

where i includes all low-thrust and coast arcs except for the final coast arc, $\bar{x}_{i,F}$ is the initial condition of the ith arc propagated forward in time for Δt nondimensional time units, and $\bar{x}_{i+1,0}$ is the initial condition of the $i+1$th arc. The full state continuity constraint for the final coast arc to ensure insertion into the final orbit is written as

$$\bar{c}_F = \bar{x}_{p+n,F} - \bar{x}_{IC,Fin} \tag{6.7}$$

where $\bar{x}_{IC,Fin}$ is set equal to the initial condition vector for the final orbit specified in Table 5.1. Note, the multiple shooting algorithm is constrained to target the final orbit initial conditions to machine precision. The partial derivatives of these constraints taken with respect to the free variables are either found via the state transition matrix or a first order central difference. Similarly, the mass continuity constraints for the low-thrust arcs are defined as

$$\Delta m_i = m_{i,F} - m_{i+1,0} \tag{6.8}$$

while the thrust direction vector constraints for the low-thrust arcs are written as

$$U_i = u_{i,x}^2 + u_{i,y}^2 + u_{i,z}^2 - 1 \tag{6.9}$$

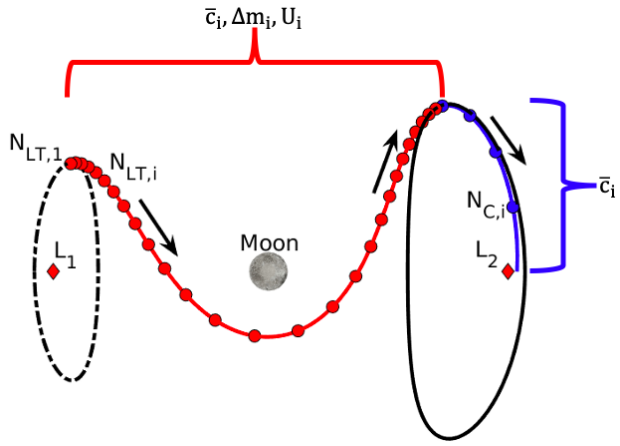


Figure 6.2: Free variables and constraints describing the segments along a transfer.

These formulations facilitate straightforward analytical partial derivatives for the constraints when taken with respect to the free variable vector for the $D\bar{F}(\bar{V})$ matrix.

Then, the entire constraint vector is written using these components as

$$\bar{F}(\bar{V}) = [\bar{c}_1, \ldots \bar{c}_F, \Delta m_1, \ldots, \Delta m_n, U_1, \ldots, U_n]^T \tag{6.10}$$

Figure 6.2 depicts a representative diagram for the second trajectory design scenario incorporating a free variable vector constructed using the results from MRPPO where spherical markers denote the initial state of an arc, red arcs correspond to low-thrust arcs, blue arcs represent coast arcs, and constraints are noted between each arc connection. In this research, the tolerance on satisfying the constraints is set as 1×10^{-10} nondimensional units measured using the norm of the constraint vector. These constraints are used to ensure that the solution recovered by SQP is continuous in the state, mass, and thrust directions to within a specified tolerance.

6.3.3 Initial Guess Construction

Each initial guess for the free variable vector input to SQP in *fmincon* is formulated using a reference trajectory developed by MRPPO. First, the low-thrust arcs are straightforwardly converted from each reference trajectory into low-thrust arcs for the corresponding initial guess. For the final orbit coast arcs, the end state of the final low-thrust arc is naturally propagated forwards

in time until the trajectory intersects the x-axis. This natural trajectory is then discretized into a specified number of coast nodes to form the final segment of the initial guess supplied to the SQP optimization scheme in *fmincon*. In this research, four coast arcs are used to ensure that the free variable vector is robust to numerical sensitivities.

6.3.4 Objective Functions

Once the initial guesses have been constructed, they are input to SQP in *fmincon* to generate locally optimal trajectories. However, SQP in *fmincon* requires that both the constraints and objective function be continuous and possess continuous first-order derivatives [94]. While the constraint vector formulation satisfies these requirements, the reward functions defined for the trajectory design scenarios are generally discontinuous and violate the requirements of SQP in *fmincon* as formulated in the reinforcement learning environment. This necessitates the definition of alternative objective functions that approximately capture the objectives that the reinforcement learning policies are trying to maximize.

In this research, two objective functions with distinct formulations are explored to validate the solution space recovered by MRPPO on three trajectory design scenarios. The first objective function formulation resembles the reward functions maximized by the policies trained using MRPPO in Eqs. (5.7) and (5.8) but with a negative sign multiplied into each component since SQP aims to minimize the objective function rather than maximize the objective function. The reward function enables policies generate low-thrust transfers that target the final orbit while minimizing flight time and propellant mass usage. The first objective function for the first trajectory design scenario is defined as

$$J_1 = 100 + \sum_{i=1}^{n} \begin{cases} 1 + c_m(\tilde{m}_{sc,i,0} - \tilde{m}_{sc,i,F}) & \text{Reference Trajectory} \\ 10|\Delta\bar{d}_{i,F}| + |\Delta\bar{v}_{i,F}| + c_m(\tilde{m}_{sc,i,0} - \tilde{m}_{sc,i,F}) & \text{Final Orbit} \end{cases} \tag{6.11}$$

and for the second and third trajectory design scenarios as

$$J_{2,3} = 1000 + \sum_{i=1}^{n} \begin{cases} 8 + 10|\Delta\bar{d}_{i,F}| + |\Delta\bar{v}_{i,F}| + c_m(\tilde{m}_{s/c,i,0} - \tilde{m}_{s/c,i,F}) & \text{Initial Orbit} \\ 4 + 10|\Delta\bar{d}_{i,F}| + |\Delta\bar{v}_{i,F}| + c_m(\tilde{m}_{s/c,i,0} - \tilde{m}_{s/c,i,F}) & \text{Reference} \\ 100|\Delta\bar{d}_{i,F}| + 10|\Delta\bar{v}_{i,F}| + c_m(\tilde{m}_{s/c,i,0} - \tilde{m}_{s/c,i,F}) & \text{Final Orbit} \end{cases} \tag{6.12}$$

where the 100 and 1000 constant offsets do not affect the optimization results, but facilitate direct comparisons to the transfers produced by MRPPO that receive bonuses for converging on the final orbit. Both formulations are only evaluated for the low-thrust arcs composing the trajectory. In each case, the propellant mass usage coefficient, c_m is assigned depending on the policy that this initial guess corresponds to. Additionally, the soft constraints encapsulated within the Ω term are omitted, however, the convergence bonus is included to enable comparison between the objective function evaluation and the sum of the rewards for the optimization scheme and MRPPO transfers, respectively. In this formulation, each node is assigned a reward function formulation associated with a dynamical arc prior to optimization, i.e. if low-thrust node i in the initial guess is closest to the reference trajectory, the objective function evaluation for that node is always measured with respect to the reference trajectory even if the node is updated to be closer to a different dynamical arc. This assumption prevents discontinuities in the objective function evaluation and facilitates a closer comparison between the results generated by the optimization scheme and MRPPO.

This objective function closely approximates the value function that the policies trained by MRPPO are maximizing, but does not include the discount factor. However, since the rewards are scaled using Eq. 5.9 to have a mean of zero and standard deviation of one prior to each update to the neural networks, these scaled rewards influence the approximation of the value function and advantage function. Then, the critic neural network is no longer directly estimating the value function, but instead, a normalized value function that is continually updated throughout training as more state-action-reward experiences are generated and the policies converge on behavior that returns higher rewards. While this objective function may be modified to include the discount factor, this would no longer reflect the ultimate goal of the policies, which is to return the cumulative reward

over a trajectory within the environment. Instead, this alternative objective function formulation, without the discount factor, based on the reward function is used to measure the performance of the policies in maximizing the total reward over a trajectory.

While the first formulation of the objective function enables a more direct comparison between the two methodologies, an alternative objective function formulation is used to explore the characteristics of the flight time and propellant mass usage trade space recovered by MRPPO. Thus, a distinct but comparable objective function is specified for use in SQP implemented in *fmincon* to maximize the final mass of the spacecraft as

$$J_m = -\tilde{m}_{n,F} \tag{6.13}$$

where the final mass of the spacecraft is measured at the termination of the final propagated low-thrust node. This objective function enables the recovery of locally optimal, minimal propellant solutions for a fixed flight time. Then, this objective function maximizes distinct objectives compared to the objective functions defined in Eqs. 6.11 and 6.12, which balance flight time, propellant mass usage, and minimizing state deviations from the dynamical arcs. The recovered solutions necessarily are part of a distinct multi-objective solution space compared to the solutions recovered by MRPPO. However, for the purposes of uncovering insights into the flight time vs. propellant mass usage trade space, both methodologies and objective functions are valuable. In this research, the tolerance on satisfying the optimality in *fmincon* is set as 1×10^{-8} nondimensional units.

6.3.5 Summary of Differences in Implementation from Reinforcement Learning

A number of differences must be noted between the implementations of the two methodologies, MRPPO and SQP, and the influence on the resulting multi-objective solution spaces. As noted in Section 6.3.4, the solutions recovered by the optimization scheme are maximizing distinct objectives compared to the MRPPO reward functions. While the objective functions defined in Eqs.6.11 and 6.12 are similar to the reward function used by MRPPO to maximize the cumulative reward encapsulated within the value function, as noted, the MRPPO implementation incorporates

a discount factor and reward scaling altering the estimation of the value function by the critic neural network. Then, the policies trained by MRPPO are no longer maximizing the default value function, but instead, a normalized value function at each update that incentivizes the policies to continue to improve. However, since the goal of MRPPO is to maximize the cumulative reward returned from the reward function, MRPPO and SQP may be compared in how well they accomplish that goal.

A second difference results from the approach used by both methods. In MRPPO, policies learn from many training trajectories originating at unique initial conditions surrounding the initial periodic orbit. Thus, when evaluating the policies, the policies are only able to generate a trajectory from a unique initial condition once in the environment and are not able to iterate on that trajectory to produce a better solution. Conversely, SQP may iterate thousands of times on a trajectory originating from a single initial condition to develop an optimal solution.

Another difference in implementations occurs due to how the transfers are constructed in both environments. In MRPPO, transfers are guaranteed to be continuous throughout the entire transfer while in the optimization scheme, full state continuity must be enforced between all arcs via constraints. While the constraints are satisfied to within a small tolerance, that tolerance is always larger than the numerical integration error that the transfers generated by MRPPO possess. However, for targeting the final orbit, the optimization scheme is constrained to target a smaller hypersphere along the orbit than the solutions produced by MRPPO. Since the optimization scheme may iterate thousands of times to develop a locally optimal solution, the size of the hyperspheres is decreased.

A final difference in both implementations is how the transfer geometries are selected. MRPPO autonomously explores the solution space prior to selecting a transfer geometry to exploit, a consequence of the exploration vs. exploitation trade-off inherent to reinforcement learning algorithms. Conversely, the reference trajectories generated by MRPPO serve as initial guesses for the optimization scheme constraining the optimization scheme to develop optimal solutions within the local basin of convergence surrounding the initial guess. The optimization scheme may recover a better

solution for a distinct objective function within the local basin of convergence, but a similar solution would validate that MRPPO recovered the optimal solution within the local basin of convergence. While a global optimization scheme may uncover a better solution by extensively exploring the entire solution space, both MRPPO and SQP in *fmincon* only guarantee recovery of a local optima. The noted differences in implementations between the two methodologies guarantee that the recovered solution spaces cannot be identical, however, the comparison is valuable for generating insights into the validity of the multi-objective solution spaces recovered by MRPPO and exploring the differences in the transfer geometries for the three trajectory design scenarios.

6.4 Recovering a Trajectory that Resembles a Periodic Orbit in an Ephemeris Model

In addition to validating the low-thrust transfers generated by policies trained using MRPPO, a similar nonlinear optimization scheme is employed to develop natural trajectories in the Sun-Earth-Moon point mass ephemeris model that resemble a nearby Sun-Earth L_5 short period orbit for the third trajectory design scenario. While the CR3BP serves as a valuable initial guess for approximating motion in higher-fidelity dynamical models, corrections must be introduced to account for the non-circular orbits of the Sun and Earth and the gravitational influence of the Moon. However, implementing a corrections scheme within a reinforcement learning environment is computationally expensive. Instead, the 1,000 states discretized along the L_5 short period orbit in the Sun-Earth CR3BP are propagated in the Sun-Earth-Moon point mass ephemeris model and used to seed initial guesses for trajectories that resemble a Sun-Earth L_5 short period orbit. Specifically, optimization is used to minimize the cumulative position difference from the Sun-Earth L_5 short period orbit modeled in the Sun-Earth CR3BP [23, 133]. Then, at each time step in the reinforcement learning environment, 1,000 states at the defined epoch for each ephemeris trajectory are drawn and used to approximate the L_5 short period orbit. A spacecraft approaching to within specified position and velocity hyperspheres along any of the 1,000 states is considered converged and remains in bounded, natural motion around the Sun-Earth L_5 equilibrium point. A nonlinear

constrained optimization scheme is employed to facilitate the development of the 1,000 natural ephemeris trajectories.

6.4.1 Free Variables

The initial guess for a natural ephemeris trajectory is composed of only natural arcs. For each node of the initial guess, the epoch and propagation time are fixed to match the epochs defined for the reinforcement learning ephemeris environment. Then, the free variables for each arc of the ephemeris trajectory are

$$\bar{N}_E = [x, y, z, \dot{x}, \dot{y}, \dot{z}]^T \tag{6.14}$$

where the position and velocity are allowed to vary in the non-pulsating Sun-Earth instantaneous rotating frame. Then, the free variable vector is written as

$$\bar{V} = \left[\bar{N}_{E,1}, \ldots, \bar{N}_{E,t_{max}}\right]^T \tag{6.15}$$

where $t_{max} = 115$ for the third trajectory design scenario. This formulation of the free variable vector enables the development of natural ephemeris trajectories that may be optimized to retain bounded motion near a Sun-Earth L_5 short period orbit.

6.4.2 Equality Constraints

The constraint vector for generating the natural ephemeris trajectories that resemble a Sun-Earth L_5 short period orbit is formulated to ensure full state continuity between neighboring arcs. Eq. (6.6) ensures full state continuity between neighboring natural nodes and the full constraint vector is written as

$$\bar{F}(\bar{V}) = [\bar{c}_1, \ldots \bar{c}_{t_{max}-1}]^T \tag{6.16}$$

where continuity is not enforced at the beginning of the initial arc nor at the termination of the final arc. This enables the initial and final arcs to not be precisely on the Sun-Earth L_5 short period orbit, but instead, be optimized remain as close as possible in the configuration space to the

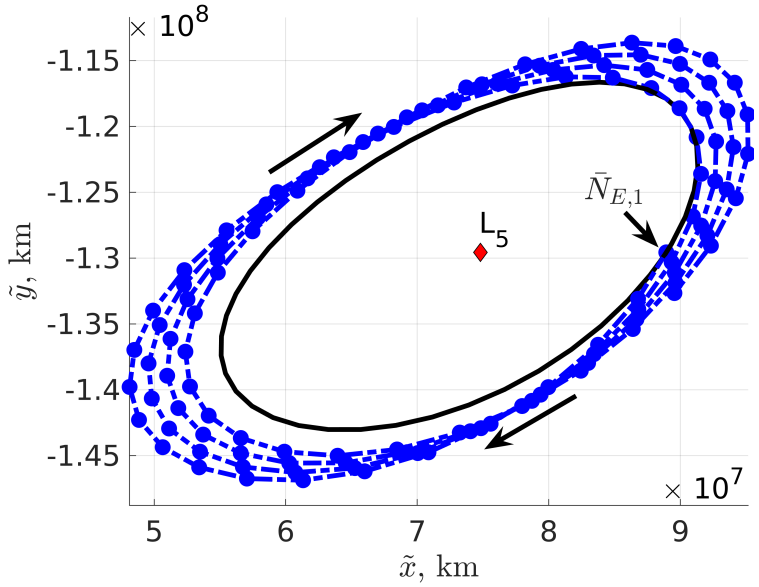


Figure 6.3: Initial guess for an ephemeris trajectory used to approximate a Sun-Earth L_5 short period orbit.

Sun-Earth L_5 short period orbit. Within SQP, the tolerance on satisfying the constraints is set as 1×10^{-10} nondimensional units.

6.4.3 Initial Guess Construction

The initial guess for the ephemeris trajectory is generated by propagating the specified state along the Sun-Earth L_5 short period orbit in the Sun-Earth-Moon point mass ephemeris model for 4.02 years. This natural trajectory is discretized evenly in time every 0.22 nondimensional time units to formulate $t_{max} = 115$ natural arcs composing the initial guess. Figure 6.3 displays one trajectory propagated in the Sun-Earth-Moon point mass ephemeris model from a state along the Sun-Earth L_5 short period orbit and discretized to formulate an initial guess. In this figure, the black arc denotes the Sun-Earth L_5 short period orbit, the blue dotted arc represents the ephemeris trajectory, and the blue spheres designate nodes used to construct the initial guess for the optimization scheme. This same process is used to develop initial guesses for all 1,000 states along the Sun-Earth L_5 short period orbit.

6.4.4 Objective Function

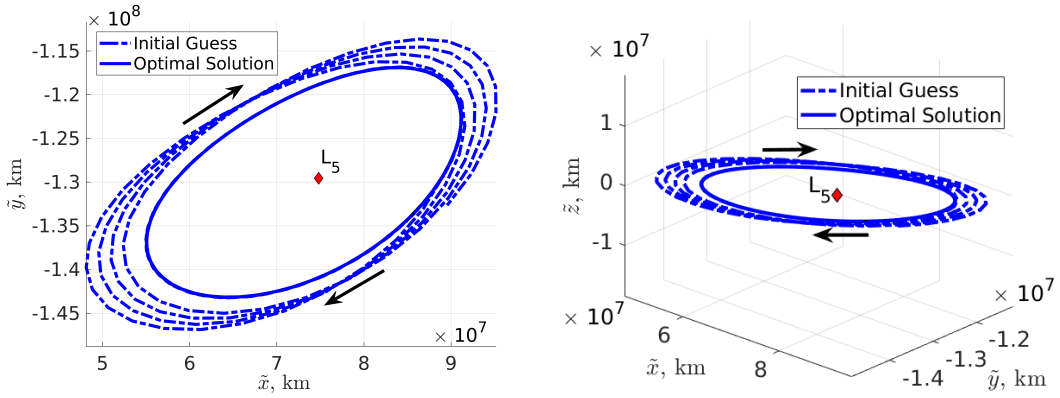
An objective function is defined to optimize the 1,000 trajectories near a Sun-Earth L_5 short period orbit at the defined epochs by minimizing the cumulative position difference from the Sun-Earth L_5 short period orbit in the Sun-Earth CR3BP. Then, one formulation for an objective function that prioritizes minimizing the position difference near the Sun-Earth L_5 short period orbit is written as

$$J = \sum_{i=1}^{t_{max}} |\bar{d}_{i,0} - \bar{d}_{i,0,SPO}| \tag{6.17}$$

where $\bar{d}_{i,0}$ is the initial position of each node and $\bar{d}_{i,0,SPO}$ is the position of a state selected a priori along the Sun-Earth L_5 short period orbit. The tolerance on satisfying the optimality condition in *fmincon* is set as 1×10^{-8} nondimensional units. This formulation minimizes the position deviation between the natural ephemeris trajectory and the Sun-Earth L_5 short period orbit at each epoch.

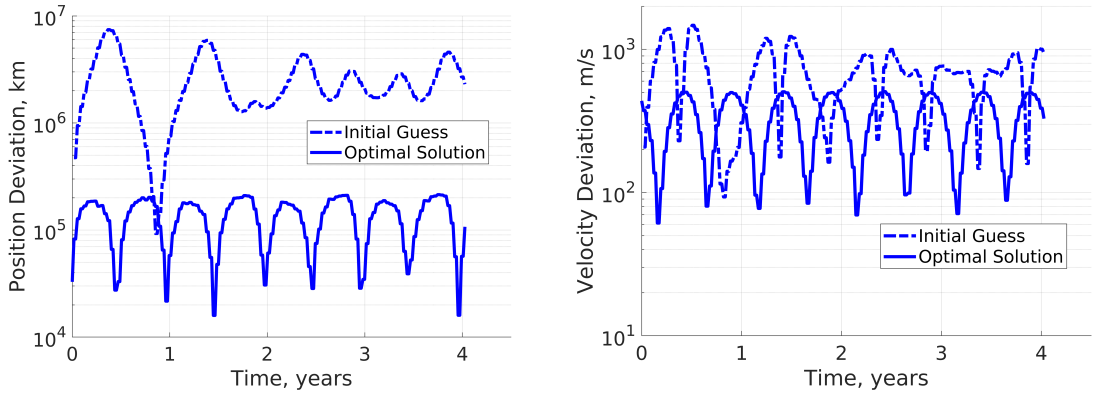
6.4.5 Example of an Ephemeris Trajectory Resembling a Periodic Orbit

The optimization scheme is used to produce natural ephemeris trajectories that approximate the motion of a Sun-Earth L_5 short period orbit in the Sun-Earth-Moon point mass ephemeris model. First, lets examine the trajectory produced from the initial guess initial guess depicted in Fig. 6.3. Figure 6.4 displays the optimized ephemeris trajectory using the solid blue arc, the initial ephemeris trajectory is displayed using the dashed blue arc, and the black arc represents the Sun-Earth L_5 short period orbit in the Sun-Earth CR3BP. In this figure, the optimized ephemeris trajectory closely resembles the L_5 short period orbit and envelopes the orbit preventing visualization. Further, Fig. 6.5a demonstrates that the initial ephemeris trajectory possesses over an order of magnitude larger position deviation measured from the closest state along the Sun-Earth L_5 short period orbit compared to the optimized ephemeris trajectory. Similarly, Fig. 6.5b depicts the velocity deviation measured from the closest state along the Sun-Earth L_5 short period orbit for both the initial and optimized ephemeris trajectories. Since the objective function does not include velocity components, the optimized ephemeris trajectory possesses velocity deviations closer to the



(a) Planar View (b) Spatial View

Figure 6.4: Optimized ephemeris trajectory and initial guess used to approximate a Sun-Earth L_5 short period orbit.



(a) Position deviation (b) Velocity deviation

Figure 6.5: State deviations for an optimized ephemeris trajectory approximating a Sun-Earth L_5 short period orbit.

initial guess. This demonstrates one approach to rapidly recovering ephemeris trajectories that closely resemble a periodic orbit in an ephemeris model.

Chapter 7

Application: Low-Thrust Transfers in Multi-Body Systems

MRPPO is used to train multiple policies, each with a distinct reward function, to recover unique controls schemes for low-thrust transfers in the Earth-Moon and Sun-Earth systems. Once trained, each policy is evaluated to examine a region of the multi-objective solution space and the balance between propellant mass usage and flight time for each trajectory design scenario. Then, the reference trajectories associated with each policy for each trajectory design scenario supply initial guesses to the constrained optimization scheme for approximate verification of the results generated via multi-objective reinforcement learning. The resulting solution spaces from both methodologies are used to validate the results produced by MRPPO with the moving reference modification. Further, the results generated by MRPPO for the first trajectory design scenario are also verified and compared to results developed using PPO. Additionally, the impact of hyperparameter selection on the policies and training is examined for the first trajectory design scenario. Finally, for the third trajectory design scenario, the low-thrust transfers are recovered using a variable thrust and a constant thrust engine configuration. For both engine models, the policies are first trained to recover locally optimal solutions in the Sun-Earth CR3BP prior to incorporating a transfer learning approach to update the policies to recover locally optimal solutions in the Sun-Earth-Moon point mass ephemeris model. This exploration is used to demonstrate the use of MRPPO to autonomously recover a subset of the multi-objective solution space in a higher-fidelity dynamical model using transfer learning.

7.1 Low-Thrust Transfers from an Earth-Moon L_2 Halo Orbit to a Neighboring Earth-Moon L_2 Halo Orbit

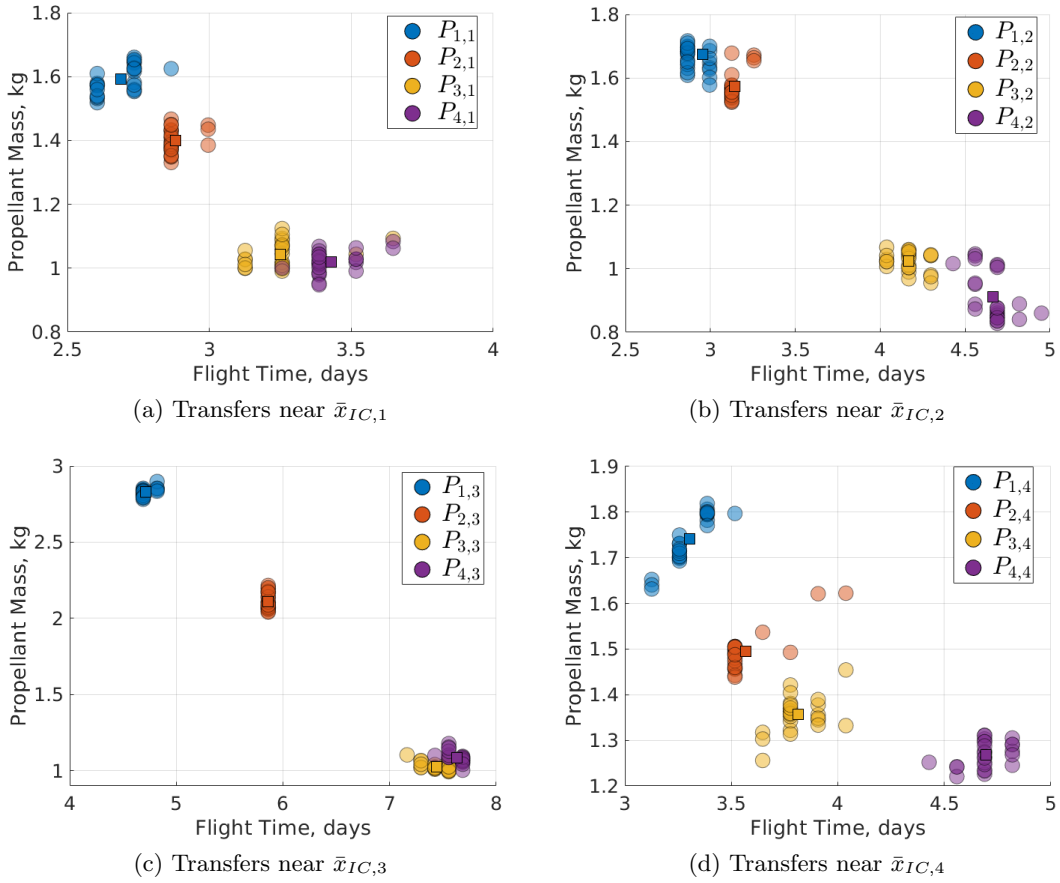
Multiple policies, each with a distinct reward function, are trained to recover unique controls schemes for a transfer between two L_2 southern halo orbits in the Earth-Moon CR3BP starting at one of four departure locations along the initial orbit. In this trajectory design scenario, MRPPO is run once to train four policies for each initial condition. Specifically, for the i-th initial reference location along the periodic orbit, policy $P_{1,i}$ seeks to maximize the reward function with no penalty on propellant mass usage, i.e. $c_m = 0$; the proceeding policies, $P_{j,i}$ for $j = [2, 3, 4]$, correspond to larger propellant mass penalties with $c_m = [13.33, 26.66, 40]$. Once trained, each policy is evaluated to examine a region of the multi-objective solution space and the balance between propellant mass usage and flight time. Characteristics from the evaluation trajectories are used to extract the mean propellant mass usage and flight times for each controls scheme in the multi-objective solution space. Then, the reference trajectories associated with each policy for each initial condition supply initial guesses to the constrained optimization scheme for verification of the results generated via multi-objective reinforcement learning. The initial guesses are optimized using two distinct objective functions to validate the transfers recovered by MRPPO. The first objective function is similar to the reward function used by MRPPO while the second objective function solely minimizes propellant mass usage. The multi-objective solution spaces and characteristics of the solutions recovered by SQP are compared to the results produced by MRPPO. Finally, one initial condition is selected and the effect of hyperparameter selection on the policies throughout training is evaluated using the total reward summed across all policies.

7.1.1 Transfers Recovered via MRPPO with Baseline Hyperparameters

Four policies are simultaneously trained using MRPPO to develop controls schemes that guide a low-thrust-enabled SmallSat from initial conditions near one of four departure locations along an L_2 southern halo orbit to a higher-energy L_2 southern halo orbit in the Earth-Moon system

using the baseline set of hyperparameters defined in Table 5.5. Figure 7.1 depicts a subset of the recovered multi-objective solution spaces for each departure location using the characteristics from each evaluation trajectory for each policy. In these figures, the evaluation trajectory characteristics are displayed as semi-transparent circles and the mean values as squares, each shaded corresponding to the associated policy. Policies $P_{4,i}$ recover solutions requiring approximately 1 kg of propellant mass while policies $P_{1,i}$ generate transfers requiring approximately 1.7 kg of propellant mass except for policy $P_{1,3}$ which necessitates approximately 2.8 kg of propellant mass. Additionally, policy $P_{2,3}$ develops transfers with significantly higher propellant mass usage than policies $P_{2,1}$, $P_{2,2}$, and $P_{2,4}$. Further, while $P_{3,3}$ and $P_{4,3}$ produce transfers with lower propellant mass usages, these transfers possess significantly longer flight times. This insight suggests that $\bar{x}_{IC,3}$ may not be a favorable departure location along the initial periodic orbit compared to the other departure locations. However, each solution space indicates, as expected, that a longer flight time corresponds to a lower required propellant mass for each policy at each departure location.

The trajectories controlled by each policy at each departure location are also examined. Figures 7.2 - 7.5 depict both the reference trajectories and the trajectories generated by evaluating $\mathrm{P}_{1,i}$, $\mathrm{P}_{2,i}$, $\mathrm{P}_{3,i}$, and $\mathrm{P}_{4,i}$ for $i = [1, 4]$. The evaluation trajectories are shaded using a variety of hues to aid in differentiation and the final reference trajectory generated by each policy is depicted as a dotted arc and completely enveloped by the evaluation trajectories. In addition to the variations in flight time and propellant mass across these solutions, there are noticeable differences in geometry corresponding to the distinct basins of convergence for each policy, supporting recovery of the distinct regions of the solution space. Transfers originating near $\boldsymbol{x}_{IC,1}$, $\boldsymbol{x}_{IC,3}$, and $\boldsymbol{x}_{IC,4}$ possess similar geometries which converge to the final orbit near the maximum $\tilde{y}$ amplitude that are distinct from the transfers generated near $\boldsymbol{x}_{IC,2}$ which approach near the maximum $\tilde{x}$ amplitude. Further, for $\boldsymbol{x}_{IC,3}$, the evaluation trajectories produced by $\mathrm{P}_{3,3}$, and $\mathrm{P}_{4,3}$ tend to follow the initial periodic orbit until approaching the vicinity of $\boldsymbol{x}_{IC,4}$ leading to the increased flight time for these transfers. Conversely, policies $\mathrm{P}_{1,3}$, and $\mathrm{P}_{2,3}$ develop transfers that immediately depart the vicinity of the initial periodic orbit and arrive at distinct locations along the final periodic orbit corresponding



(a) Transfers near $\bar{x}_{IC,1}$

(b) Transfers near $\bar{x}_{IC,2}$

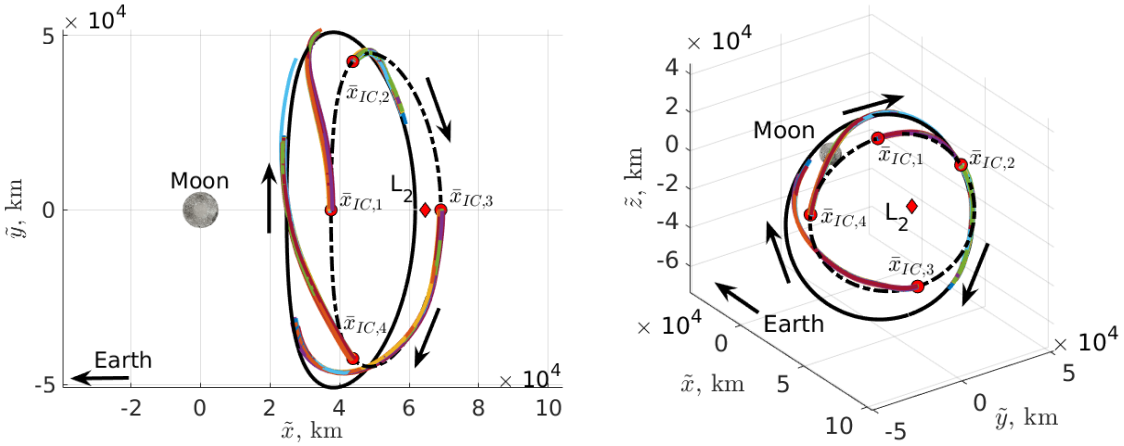
(c) Transfers near $\bar{x}_{IC,3}$

(d) Transfers near $\bar{x}_{IC,4}$

Figure 7.1: Characteristics of the evaluation trajectories constructed by policies trained using MRPPO.

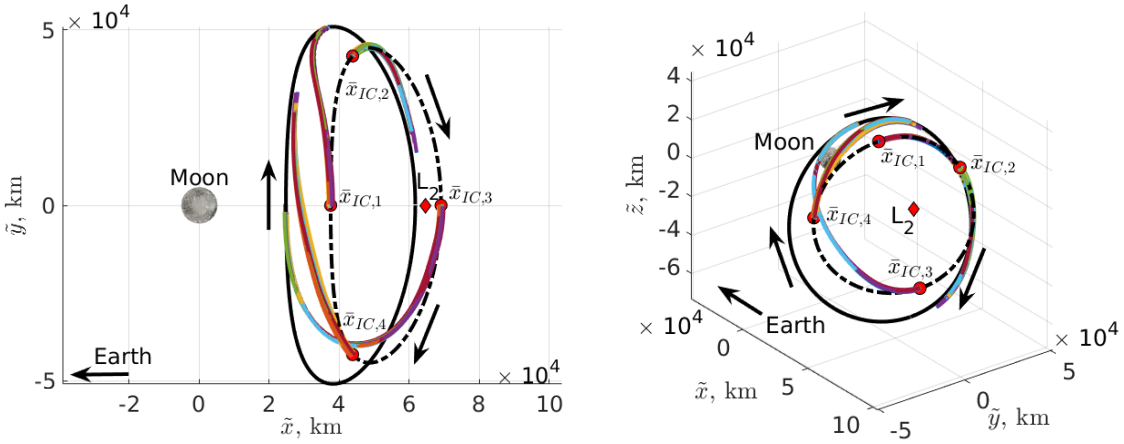
to significantly higher propellant mass usages but lower flight times. This further evidences that $\boldsymbol{x}_{IC,3}$ is not a favorable departure location compared to $\boldsymbol{x}_{IC,1}$, $\boldsymbol{x}_{IC,2}$, and $\boldsymbol{x}_{IC,4}$.

The differences in the transfer geometries are due to the distinct thrust magnitude profiles developed by each policy. Figure 7.6 depicts the thrust magnitude profiles for the reference trajectories developed by $\mathrm{P}_{1,i}$, $\mathrm{P}_{2,i}$, $\mathrm{P}_{3,i}$, and $P_{4,i}$ at each departure location along the initial periodic orbit. The reference trajectories developed by policies $\mathrm{P}_{1,i}$ and $\mathrm{P}_{2,i}$ possess similar thrust magnitude profiles with thrust levels between 50 mN and 200 mN throughout the transfers. However, the thrust magnitude profiles of the reference trajectories developed by policies $\mathrm{P}_{3,i}$ and $\mathrm{P}_{4,i}$ exhibit much lower thrust magnitudes throughout the transfers except for near the end of the transfers



(a) Planar view (b) Spatial view

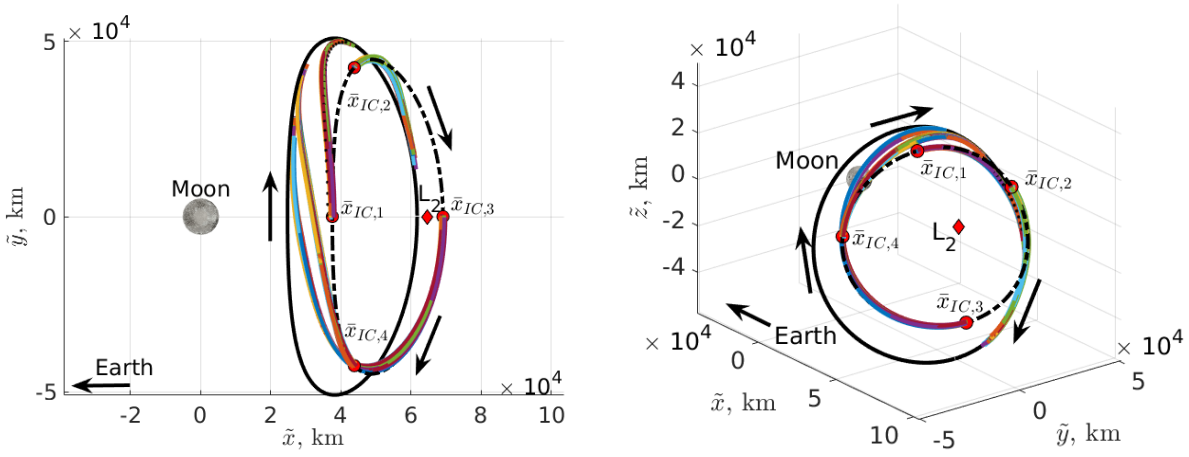
Figure 7.2: Evaluation and reference trajectories controlled using policies $\mathrm{P}_{1,i}$ trained using MRPPO (Moon not to scale).



(a) Planar view (b) Spatial view

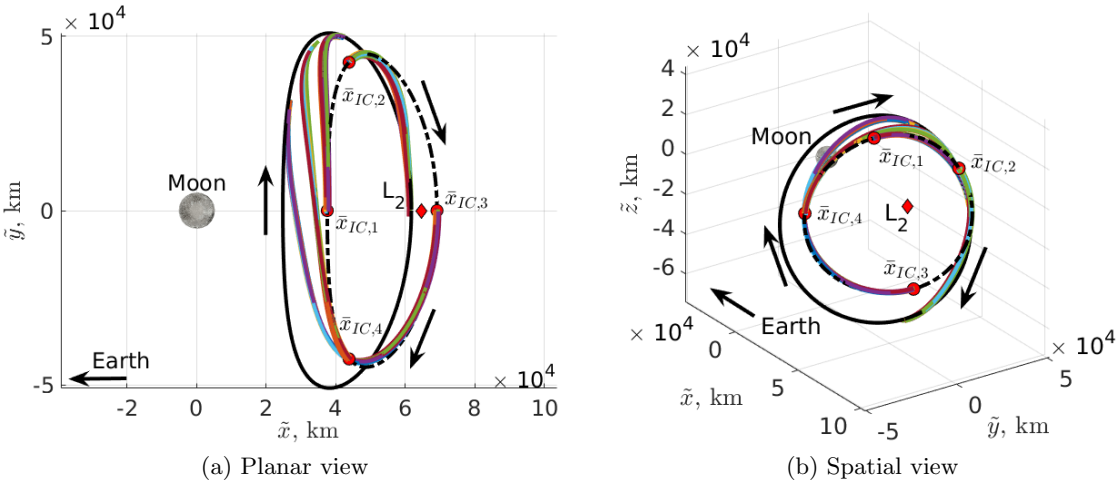
Figure 7.3: Evaluation and reference trajectories controlled using policies $\mathrm{P}_{2,i}$ trained using MRPPO (Moon not to scale).

when the spacecraft is approaching the final periodic orbit. This effect is attributable to the higher propellant mass usage penalty assigned to policies $\mathrm{P}_{3,i}$ and $\mathrm{P}_{4,i}$, which also influences the increase in flight time across the reference trajectories. Further, the reference trajectories constructed by $\mathrm{P}_{3,3}$ and $\mathrm{P}_{4,3}$ possess near-zero thrust magnitudes for a majority of the transfers while the reference trajectories follow the initial periodic orbit demonstrating that $\boldsymbol{x}_{IC,3}$ is not an ideal departure



(a) Planar view (b) Spatial view

Figure 7.4: Evaluation and reference trajectories controlled using policies $P_{3,i}$ trained using MRPPO (Moon not to scale).

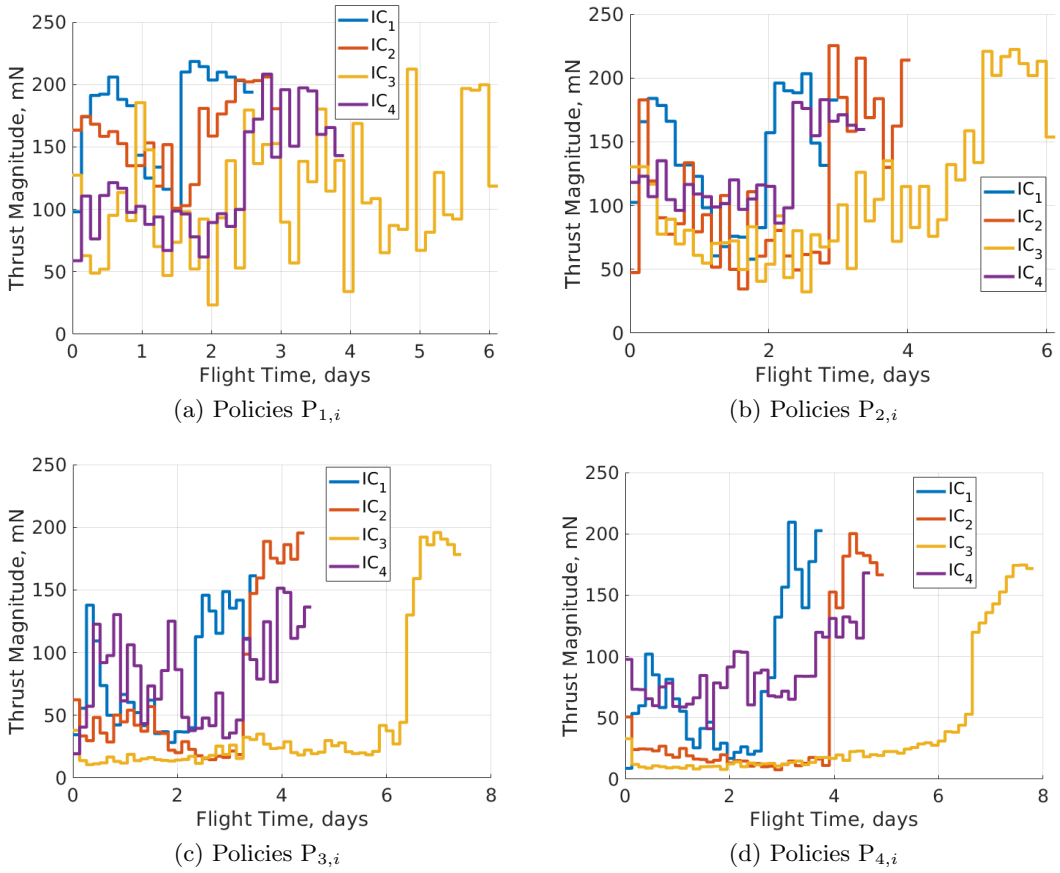


(a) Planar view (b) Spatial view

Figure 7.5: Evaluation and reference trajectories controlled using policies $P_{4,i}$ trained using MRPPO (Moon not to scale).

location since alternative reference trajectories with shorter flight times and less propellant mass usages were not recovered.

Lastly, Fig. 7.7 displays the Jacobi constant throughout the reference trajectories developed by policies $P_{1,i}$, $P_{2,i}$, $P_{3,i}$, and $P_{4,i}$, respectively where the black dashed line and the black solid line



(a) Policies $P_{1,i}$ (b) Policies $P_{2,i}$

(c) Policies $P_{3,i}$ (d) Policies $P_{4,i}$

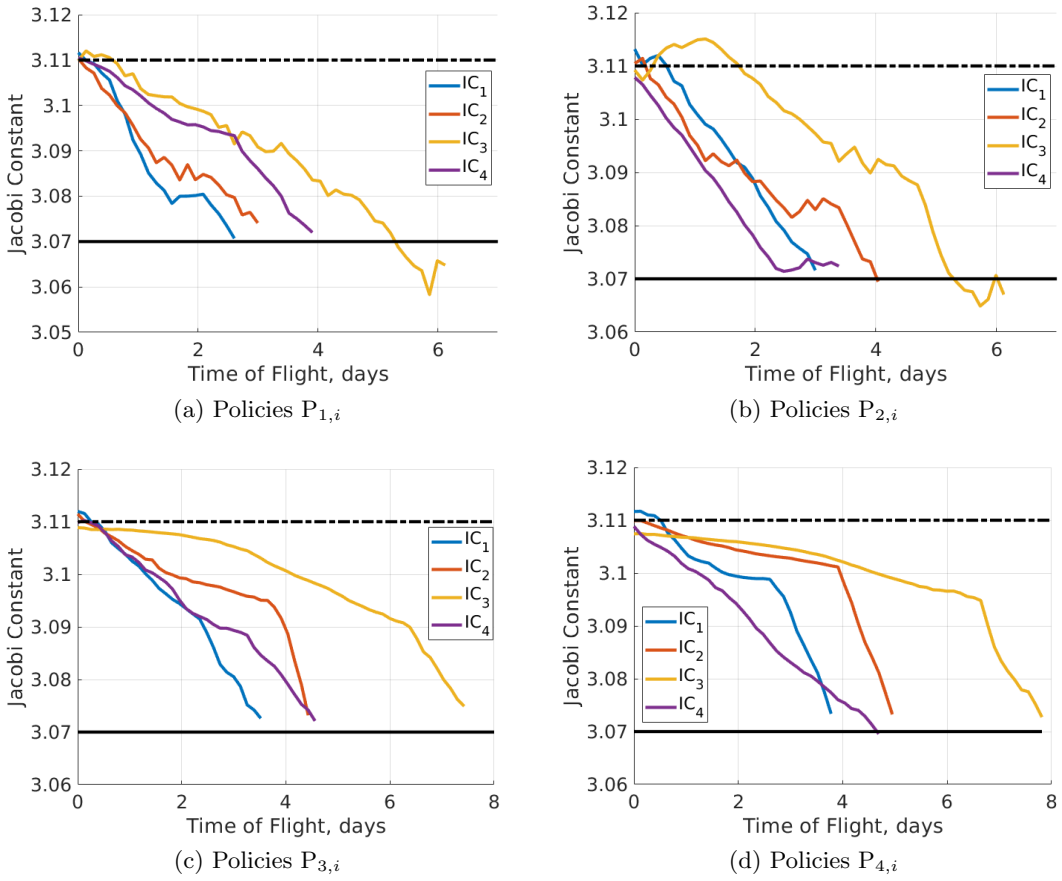
Figure 7.6: Thrust magnitude profiles of the reference trajectories controlled using policies $P_{1,i}$, $P_{2,i}$, $P_{3,i}$, and $P_{4,i}$ trained using MRPPO.

denote the Jacobi constants of the initial and final periodic orbits, respectively. For all reference trajectories, the Jacobi constant decreases throughout the transfer to match the Jacobi constant of the final orbit. Similar to the thrust magnitude profiles, the Jacobi constants of the reference trajectories produced by policies $P_{1,i}$ and $P_{2,i}$ possess similar geometries as they decrease towards the Jacobi constant of the final periodic orbit. However, the Jacobi constants of the reference trajectories developed by $P_{1,3}$ and $P_{2,3}$ overshoot the Jacobi constant of the final periodic orbit further explaining the high propellant mass usage required for these transfers. The reference trajectories developed by policies $P_{3,1}$, $P_{3,4}$, $P_{4,1}$, and $P_{4,4}$ also possess Jacobi constants that approximately linearly decrease towards the Jacobi constant of the final periodic orbit. However, the reference

trajectories developed by policies $P_{3,2}$, $P_{3,3}$, $P_{4,2}$, and $P_{4,3}$ only slightly decrease the Jacobi constant until approaching the final orbit whereby the Jacobi constant significantly decreases which corresponds to the increase in the thrust magnitude. Further, the Jacobi constants at the end of the reference trajectories do not exactly match the Jacobi constant of the final orbit due to the non-zero position and velocity hyperspheres defined to determine convergence to the final orbit. These position and velocity deviations do not guarantee that the Jacobi constant of the reference trajectory matches the Jacobi constant of the final periodic orbit, which is reflected in the plots. Similarly, the perturbations induced on the initial condition locations selected along the initial periodic orbit influence the Jacobi constant at the beginning of the reference trajectories. Each policy distinctly balances flight time and propellant mass usage for each departure location allowing the recovery of transfers that span the multi-objective solution space.

7.1.2 Transfers Recovered via PPO with Baseline Hyperparameters

The low-thrust transfers generated by policies trained using MRPPO are compared to policies trained via PPO with a nearly-identical set of baseline hyperparameters and set of reward functions. Recall, PPO trains a single policy to maximize the cumulative reward returned from a single reward function in an unknown dynamical environment without learning from other policies exploring the environment. Table 5.5 lists the set of hyperparameters used throughout the training process for PPO with the exception of the number of agents per policy which is updated from 4 to 16 agents per policy. This increase in number of agents per policy theoretically should provide the same amount of information to each policy trained by MRPPO and PPO, but with more agents per policy, the PPO policies possess more agency in the environment. This should provide improved performance for PPO policies compared to MRPPO policies since each PPO policy has more agents, i.e. spacecraft, to explore unknown regions of the solution space. However, since a policy trained via PPO only has access to the actions that policy performs, the policy may not be able to learn from potentially innovative advantageous or disadvantageous actions taken by another policy exploring a distinct region of the solution space. Additionally, since PPO is generally robust to hyperparameter



(a) Policies $\mathrm{P}_{1,i}$ (b) Policies $\mathrm{P}_{2,i}$

(c) Policies $\mathrm{P}_{3,i}$ (d) Policies $\mathrm{P}_{4,i}$

Figure 7.7: Jacobi constants of the reference trajectories controlled using policies $\mathrm{P}_{1,i}$, $\mathrm{P}_{2,i}$, $\mathrm{P}_{3,i}$, and $\mathrm{P}_{4,i}$ trained using MRPPO.

selection, these results assume that the selecting hyperparameters lie close to the optimal set of hyperparameters for this trajectory design scenario. Then, MRPPO and PPO are used to train policies to construct transfers departing from initial conditions in the vicinity of $\bar{x}_{IC,1}$ along an L_2 southern halo orbit to a higher-energy L_2 southern halo orbit in the Earth-Moon CR3BP.

Five simulations with four policies each are trained using PPO and MRPPO to enable insights into the consistency and training behavior of PPO and MRPPO on this trajectory design scenario. In a single simulation for PPO, four policies, denoted ρ_j for $j = [1, 2, 3, 4]$, are trained sequentially and assigned the reward function defined in Eq.(5.7) where $c_m = [0, 13.33, 26.66, 40]$ is varied across the four policies. Then, this set of four policies is evaluated on a common set of 25 initial conditions,

grouped together, and compared to the policies trained simultaneously using MRPPO for the same trajectory design scenario and set of reward functions. Five simulations are performed due to the stochastic nature of reinforcement learning.

The total reward summed across all four policies for all 25 evaluation trajectories for each simulation provides insights into how well MRPPO and PPO trained the policies to maximize the cumulative reward returned from an assigned set of reward functions. The total reward is computed for every tenth update to the policies to demonstrate how the total reward evolves throughout the training process. Then, three values are computed across each simulation trained using PPO and MRPPO to determine the effectiveness of using PPO and MRPPO to maximize the cumulative reward returned from the set of reward functions: (1) the maximum total reward simulation, (2) the minimum total reward simulation, and (3) the mean of the total reward of all five simulations. Figure 7.8 depicts the total reward throughout the training process for the five simulations trained using MRPPO and PPO in blue and red, respectively. While the simulations trained using MRPPO and PPO perform similarly at the start of the training process, the MRPPO simulations by the end of training outperform and return higher total rewards compared to the PPO simulations. Additionally, the MRPPO simulations return a more consistent total reward across simulations, approximately 250, compared to the five PPO simulations which return total rewards between -2050 and 70. Then, not only does MRPPO return a higher maximum, minimum, and mean total reward across all simulations compared to PPO, but MRPPO returns more consistent total rewards also on this trajectory design scenario.

Since the PPO policies are trained individually, another method of comparison is to examine the PPO policies that return the highest cumulative reward for each reward function regardless of the simulation that the policies are categorized within. These four best PPO policies may then also be compared to a single run of MRPPO, in this case the MRPPO results shown in Section 7.1.1. The total reward returned by the best four PPO policies at the end of training is 254 while the highest total reward generated by a MRPPO simulation at the conclusion of training is 255. While the total rewards may be similar between the MRPPO and PPO simulations, directly examining

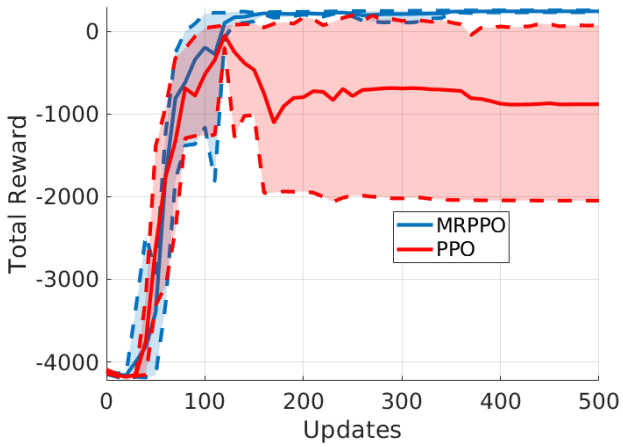
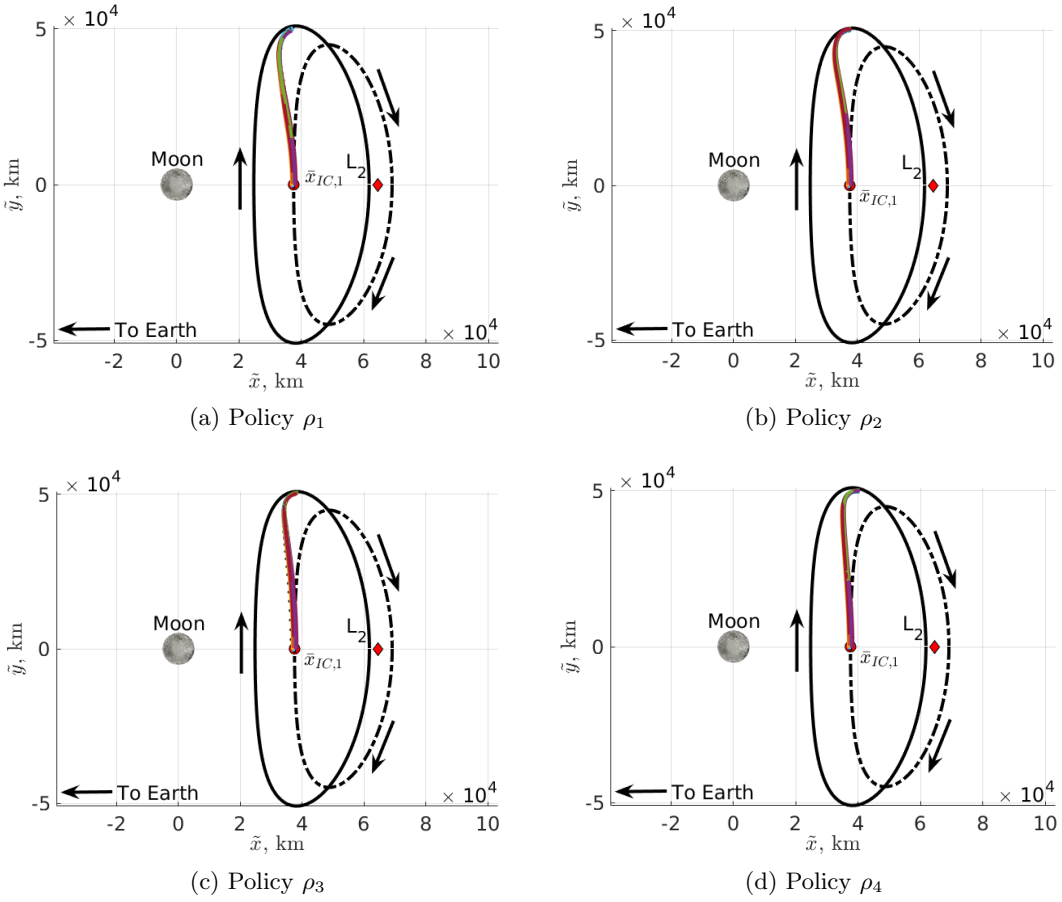


Figure 7.8: Total reward for policies trained using MRPPO and PPO applied for transfer design between Earth-Moon L_2 halo orbits.

the evaluation trajectories highlights that there are still significant differences between the two simulations. Figure 7.9 displays evaluation trajectories generated by the best PPO policies for ρ_1-ρ_4 using the same color configuration as Fig. 7.2. The evaluation trajectories produced by policies trained using PPO possess similar geometries to the results produced by MRPPO and depicted in Fig. 7.2 with minimal difference across policies for this departure location.

However, an inspection of the evaluation trajectory solution space in Fig. 7.10a reveals that MRPPO produces a more consistent solution space across the four policies compared to the policies trained using PPO. In this figure, the characteristics of the evaluation trajectories produced by PPO are depicted using stars while the characteristics of the evaluation trajectories produced by MRPPO are displayed using circles that are shaded out to aid in visualization. Specifically, policy ρ_2 trained using PPO produces nearly identical evaluation trajectory characteristics as policy ρ_3 also trained using PPO. This may be due to both ρ_2 and ρ_3 independently recovering similar locally optimal solutions without learning from other policies, as in MRPPO, that better solutions exist. Instead, policy ρ_2 trained using MRPPO recovers solutions that lie between the evaluation trajectories recovered using policies ρ_1 and ρ_3. Additionally, for policy ρ_4, MRPPO recovers evaluation trajectories with lower propellant mass usages for equivalent flight times compared to PPO.



(a) Policy ρ_1

(b) Policy ρ_2

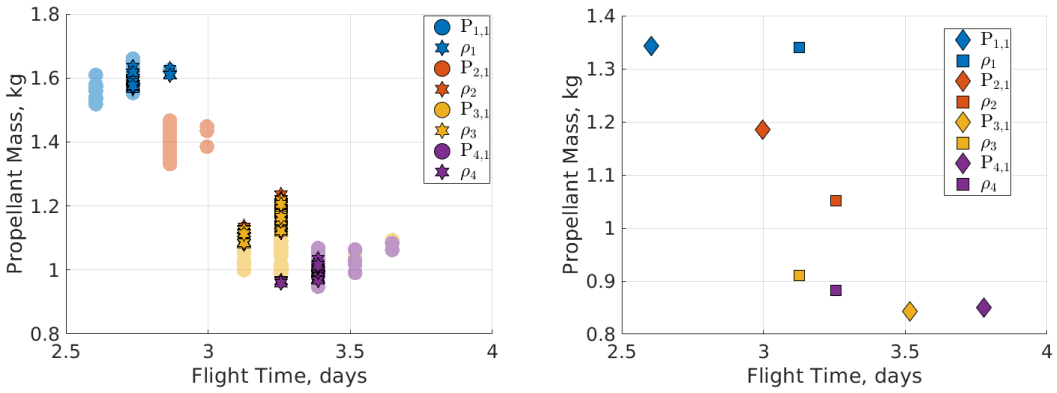
(c) Policy ρ_3

(d) Policy ρ_4

Figure 7.9: Evaluation and reference trajectories controlled using the best policies for ρ_1 - ρ_4 trained using PPO (Moon not to scale).

The only policy that produce evaluation trajectories with similar flight time and propellant mass usages for both MRPPO and PPO is policy ρ_1. Similarly, Fig. 7.10b depicts the flight time and propellant mass usage solution space for the reference trajectories constructed by policies trained using MRPPO and PPO. Again, MRPPO produces a more consistent relationship between flight time and propellant mass usage across the four policies with solutions demonstrating a consistent decrease in propellant mass usage and increase in flight time as the assigned c_m values in the reward functions increase.

In addition to the recovered solution spaces, MRPPO and PPO may also be compared via the computational time required to train the policies. The required computational time for both



(a) Evaluation trajectories (b) Reference trajectories

Figure 7.10: Characteristics of the evaluation and reference trajectories departing near $\bar{x}_{IC,1}$ constructed by policies trained using MRPPO and PPO.

Table 7.1: Machine specifications for measuring the computational time required to train policies using MRPPO and PPO.

Component	Model
Central Processing Unit	AMD Ryzen 7 2700X
Graphics Processing Unit	EVGA GeForce RTX 2070
Power Supply Unit	Corsair RMX850X
Motherboard	Asus Prime X470-Pro AM4

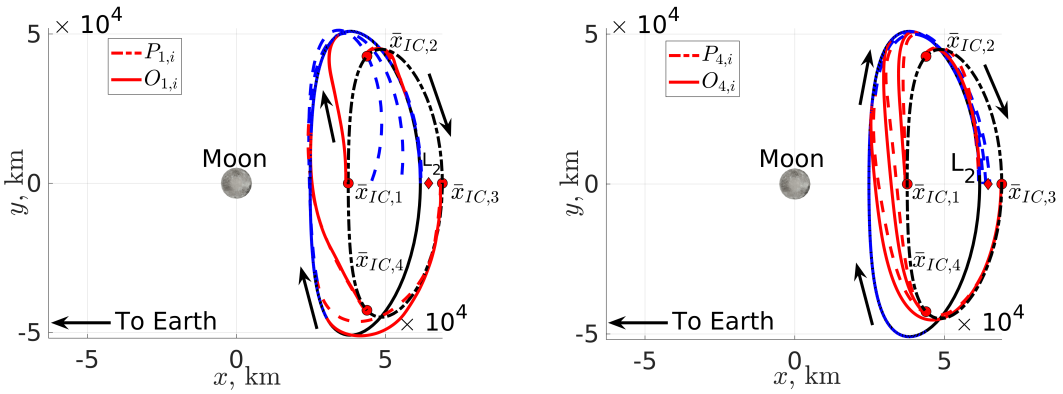
algorithms is measured on the same computational environment to facilitate a direct and fair comparison between the algorithms. Table 7.1 lists the specifications of the computational environment used to evaluate both algorithms. For this trajectory design scenario, PPO necessitated 49.3 hours to train four policies sequentially while MRPPO required 16.3 hours to train four policies simultaneously. If the four policies trained by PPO are run simultaneously on the same computational environment, the required computational time decreases to 20.6 hours. In either case, the computational time required by PPO is significantly greater than the computational time required by MRPPO and produces demonstrably worse results for this trajectory design scenario.

On this trajectory design scenario, MRPPO and PPO recover solutions spanning the propel-

lant mass usage and flight time trade space. However, the solutions produced by MRPPO more evenly span the multi-objective solution space, return a higher total reward, and possess lower propellant mass usages for equivalent flight times. Additionally, across the five simulations, MRPPO returns a consistently high total reward across the four policies while PPO produces lower rewards and more inconsistent behavior. MRPPO also requires significantly less computational time than PPO. Further exploration into the hyperparameter selection process for PPO may yield improved results, however, when using nearly identical sets of hyperparameters, MRPPO outperforms PPO on this trajectory design scenario for this specific set of simulations. By enabling policies to share data generated from the environment with one another, policies trained using MRPPO learn from the advantageous and disadvantageous state-action pairs explored by other policies with distinct objective prioritizations leading to overall better performance across the policies. While MRPPO returns better results compared to PPO on this trajectory design scenario, the results generated by MRPPO are also validated using a classical optimization scheme.

7.1.3 Transfers Recovered via Classical Optimization

The results autonomously generated by MRPPO are used as initial guesses for constrained optimization, supporting verification of the results produced using multi-objective reinforcement learning. In this section, the reference trajectories generated by policy $P_{j,i}$ are used as initial guesses to recover a single locally optimal solution, labeled $O_{j,i}$, by solving the constrained optimization process outlined in Chapter 6 and implemented using SQP in *fmincon*. This process is followed for all the reference trajectories from policies $P_{1,i}$ to $P_{4,i}$ for $i = [1, 4]$, producing a total of 16 locally optimal transfers. Then, the initial guesses are optimized with respect to two distinct objective functions: 1) maximizing the total reward and 2) minimizing the propellant mass usage. The first objective function provides insights into the effectiveness of using MRPPO to maximize the set of defined reward functions for this trajectory design scenario and the second objective function generates insights into the optimality of the solutions developed using MRPPO in the propellant mass usage and flight time trade space. However, neither objective function is mathematically



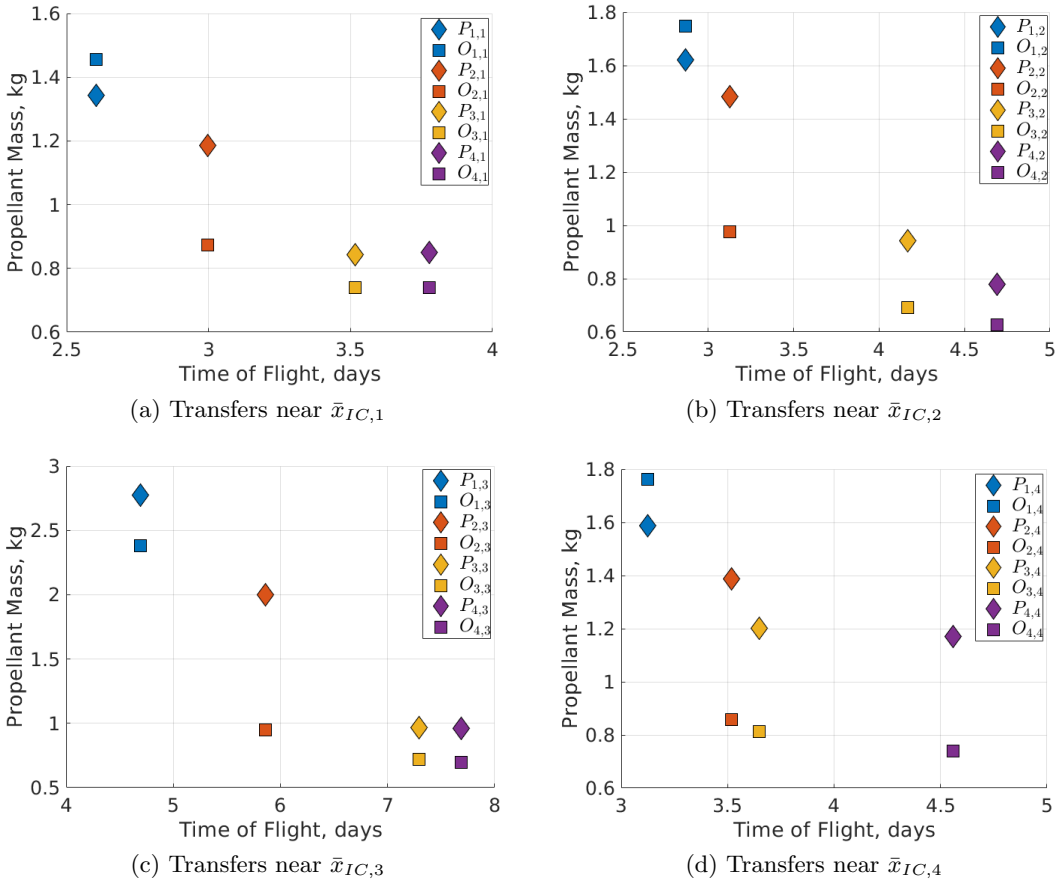
(a) Policies $P_{1,i}$ (b) Policies $P_{4,i}$

Figure 7.11: Low-thrust transfers maximizing the total reward associated with policies $P_{1,i}$ and $P_{4,i}$ from an L_2 southern halo orbit to a higher-energy L_2 southern halo orbit in the Earth-Moon CR3BP (Moon not to scale).

identical to the reward function defined in Eq. (5.7) for this trajectory design scenario. Using *fmincon* with SQP, locally optimal low-thrust transfers are developed from an L_2 southern halo orbit to a higher-energy L_2 southern halo orbit in the Earth-Moon CR3BP while adhering to the constraints defined for this trajectory design scenario.

The first objective function resembles the reward function maximized for each trajectory using MRPPO, and serves as a closer comparison to the results produced by MRPPO. Figures 7.11a and 7.11b display the results of this process with the initial guesses represented using dashed arcs, the optimal transfers depicted with solid arcs, the black dashed arc denoting the initial periodic orbit, and the black solid arc representing the final periodic orbit. The transfers of the policies at the extrema of the solution space are pictured since these transfers and policies provide insight into the overall behavior of the policies as the penalty on propellant mass usage increases. At each departure location, the low-thrust segments of the optimal transfers $O_{1,i}$ for $i = [1, 4]$ remain close to the initial guess and, therefore, the reference trajectories generated by $P_{1,i}$. However, for $P_{4,i}$, the optimal solutions, $O_{4,i}$ for $i = [1, 4]$, exhibit slight deviations from the initial guess and, therefore, the reference trajectories generated by $P_{4,i}$.

The flight time and propellant mass usage trade spaces generated from the MRPPO and SQP



(a) Transfers near $\bar{x}_{IC,1}$ (b) Transfers near $\bar{x}_{IC,2}$

(c) Transfers near $\bar{x}_{IC,3}$ (d) Transfers near $\bar{x}_{IC,4}$

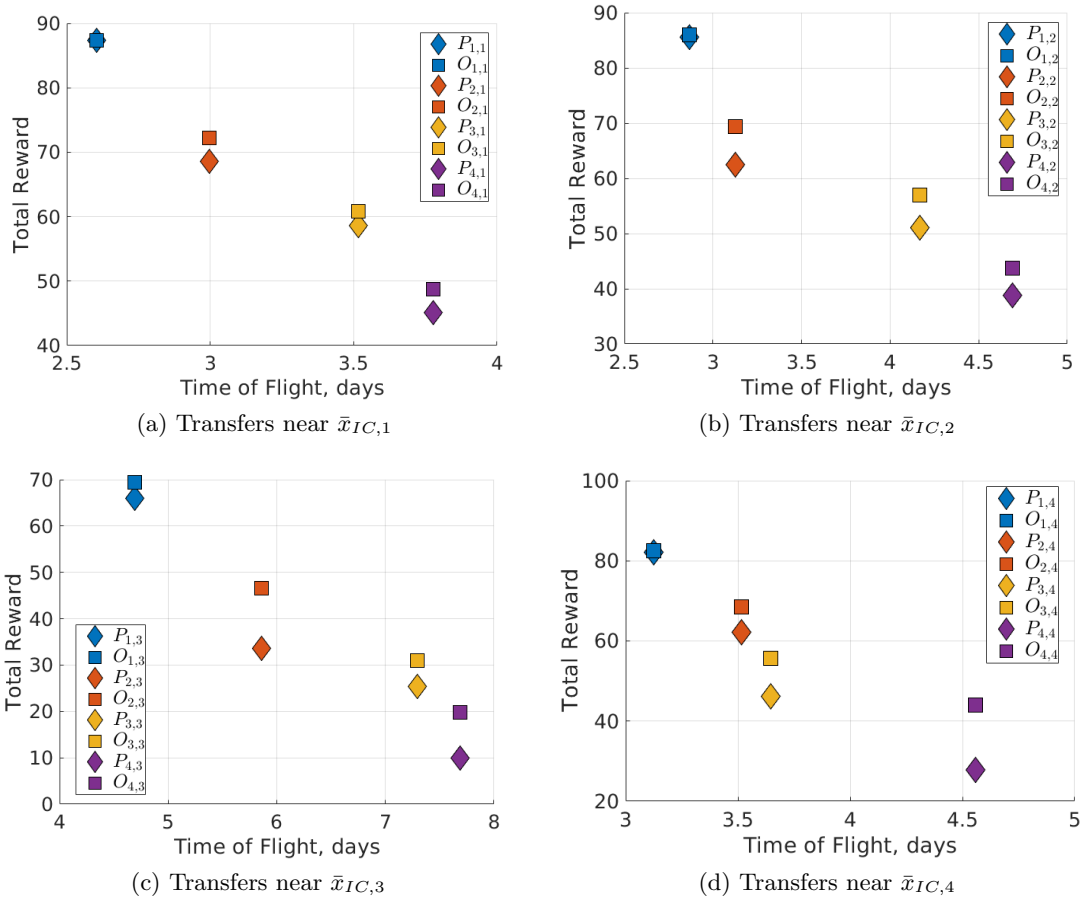
Figure 7.12: Solution space for transfers recovered via optimization while maximizing the total reward from initial guesses developed using MRPPO.

transfers for each departure location are also examined. Figures 7.12 depicts the flight time and propellant mass usage for the low-thrust transfers associated with each policy at each departure location where MR denotes the solutions recovered by MRPPO and SQP represents solutions recovered using *fmincon* with SQP. For the low-thrust transfers associated with policies $\mathrm{P}_{2,i}$, $\mathrm{P}_{3,i}$, and $\mathrm{P}_{4,i}$, SQP recovers solutions with lower required propellant mass usages. For policies $\mathrm{P}_{1,i}$, MRPPO develops transfers with lower propellant mass usages than SQP for all departure locations except $\bar{x}_{IC,3}$. However, for policies with a higher penalty on propellant mass usage and longer flight times, the gap in propellant mass usage decreases between the low-thrust transfers produced by MRPPO and SQP.

The optimization results may also be viewed in a flight time and total reward solution space reflecting the effectiveness of the optimization scheme in recovering the maximum total reward for a low-thrust transfer with a fixed flight time. Figure 7.13 displays the flight time and total reward for the low-thrust transfers associated with each policy at each departure location along the initial periodic orbit. For each initial condition, policies $P_{1,i}$ recover solutions with nearly identical total rewards as the optimization scheme. However, as the penalty on propellant mass usage increases, the optimization scheme tends to produce transfers that require slightly less propellant mass usage, which is reflected in the total reward. These differences are due to the differences in implementations; namely, *fmincon* may extensively iterate on a single solution to limit the propellant mass usage while MRPPO learns from training data to generate transfers for not previously seen initial conditions. These results demonstrate that the policies trained using MRPPO generate a similar solution space to those generated with constrained local optimization for the specified set of reward functions. Accordingly, the results produced by MRPPO are approximately verified via the optimization scheme demonstrating that MRPPO enables the recovery of a solution space that is similar to a solution space produced by an optimization scheme without requiring a predefined initial guess to construct transfers.

The initial guesses constructed by MRPPO are also optimized with respect to the second objective function, minimizing propellant mass usage, using SQP in *fmincon*. Figures 7.14a and 7.14b exhibit the initial guesses and optimal transfers from SQP for policies $\mathrm{P}_{1,i}$ and $\mathrm{P}_{4,i}$, respectively, using the same color configuration as Fig. 7.11. The optimal transfers developed by SQP possess similar geometries as the optimal transfers generated for the first objective function; namely, the transfers associated with policies $\mathrm{P}_{1,i}$ resemble the initial guess while transfers associated with policies $\mathrm{P}_{4,i}$ tend to be shifted to have smaller $\tilde{x}$-positions. Moreover, the recovered solutions seem to possess similar geometries for both objective functions.

Examining the multi-objective solution space via the characteristics of the transfers developed by both methodologies, Fig. 7.15 depicts the propellant mass usage and flight time characteristics of the initial guesses and optimal trajectories for each policy at each departure location. These

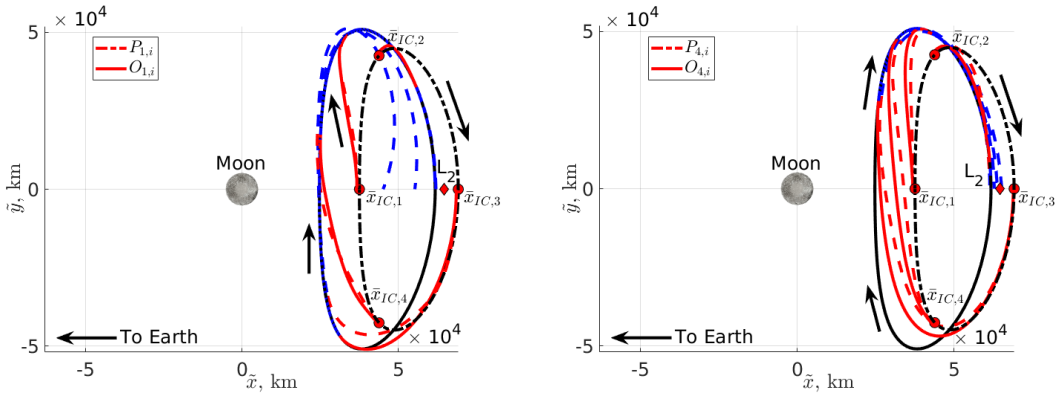


(a) Transfers near $\bar{x}_{IC,1}$ (b) Transfers near $\bar{x}_{IC,2}$

(c) Transfers near $\bar{x}_{IC,3}$ (d) Transfers near $\bar{x}_{IC,4}$

Figure 7.13: Solution space for transfers recovered via optimization while maximizing the total reward from initial guesses developed using MRPPO.

figures demonstrate that the optimal transfers developed for the second objective function require much less propellant mass usage than the transfers generated by MRPPO. However, the difference in propellant mass usage between the initial guess and optimal solution decreases as the flight time increases suggesting that MRPPO recovers solutions resembling propellant-optimal solutions at longer flight times. Since policy $\mathrm{P}_{1,i}$ is not incentivized to decrease the propellant mass usage with $c_m = 0$, this results in a large gap from the propellant-optimal solution which aims to solely minimize propellant mass usage while satisfying the constraints. Then, as the penalty on propellant mass usage increases, the transfers tend to more closely match the optimization results. Further, the differences in implementations also contribute to the differences between the two solution spaces.

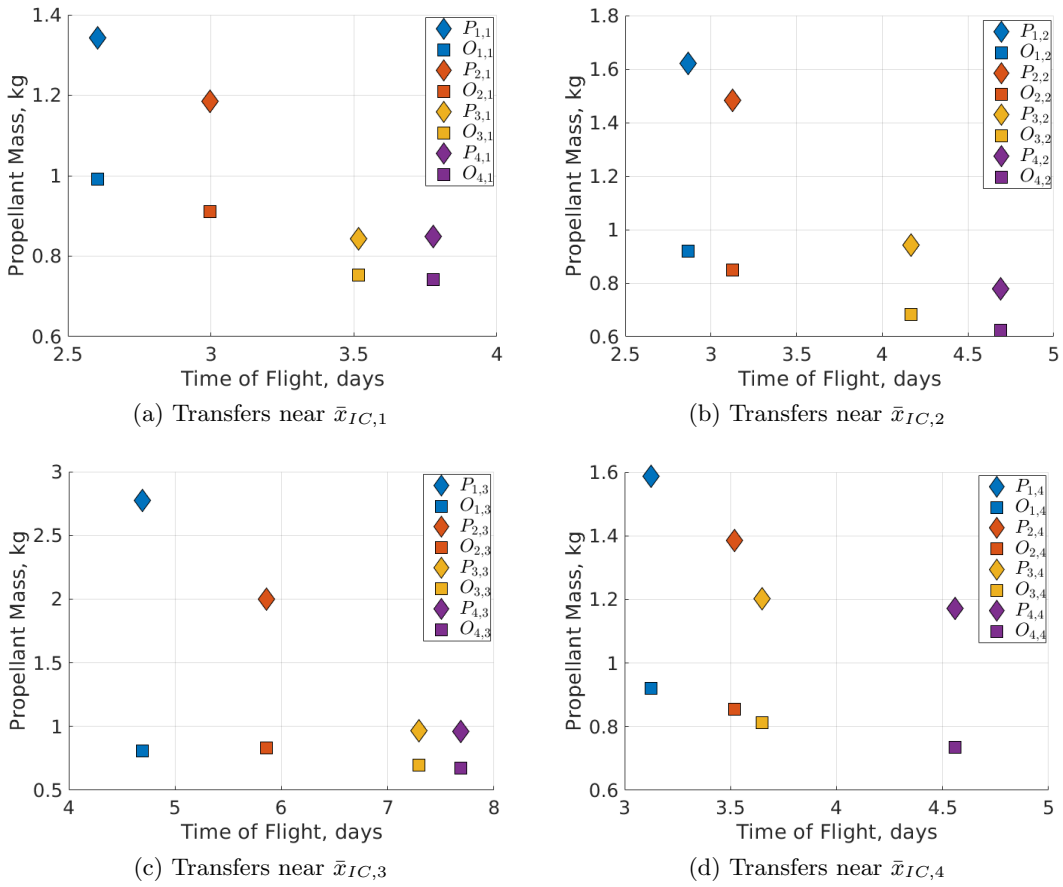


(a) Policies $P_{1,i}$ (b) Policies $P_{4,i}$

Figure 7.14: Low-thrust transfers maximizing the total reward associated with policies $P_{1,i}$ and $P_{4,i}$ from an L_2 southern halo orbit to a higher-energy L_2 southern halo orbit in the Earth-Moon CR3BP (Moon not to scale).

Additionally, the thrust magnitude profiles for both the initial guesses and optimal transfers provide insights into the discrepancy in propellant mass usage requirements. Generally, propellant-optimal solutions possess a bang-bang thrust magnitude profile [111]. However, the thrust magnitude profiles depicted in Figs. 7.6a and 7.6d do not possess perfectly bang-bang control profiles. Bang-bang control profiles are not expected since the reward functions used by MRPPO are composed of both propellant mass usage terms and state deviation terms. Instead, the reference trajectories developed by MRPPO may serve as valuable initial guesses for an optimization scheme for recovering propellant-optimal solutions with a fixed flight time.

Using the transfers developed by the optimization scheme, Figs. 7.16 and 7.17 display the thrust magnitude profiles of the initial guesses and optimal low-thrust trajectories associated with policies $P_{1,i}$ and $P_{4,i}$, respectively. Additionally, Figs. 7.18 and 7.19 depict the thrust direction profiles, which are minimally updated by the optimization scheme from the initial guess. The thrust magnitude profiles for all of the optimal transfers are updated to resemble a bang-bang control profile. However, recovering a perfect bang-bang control profile is often challenging and may only slightly decrease the required propellant mass usage. Instead, these propellant-optimal transfers possess bang-bang-like profiles suggesting that these solutions may only be locally optimal. Since



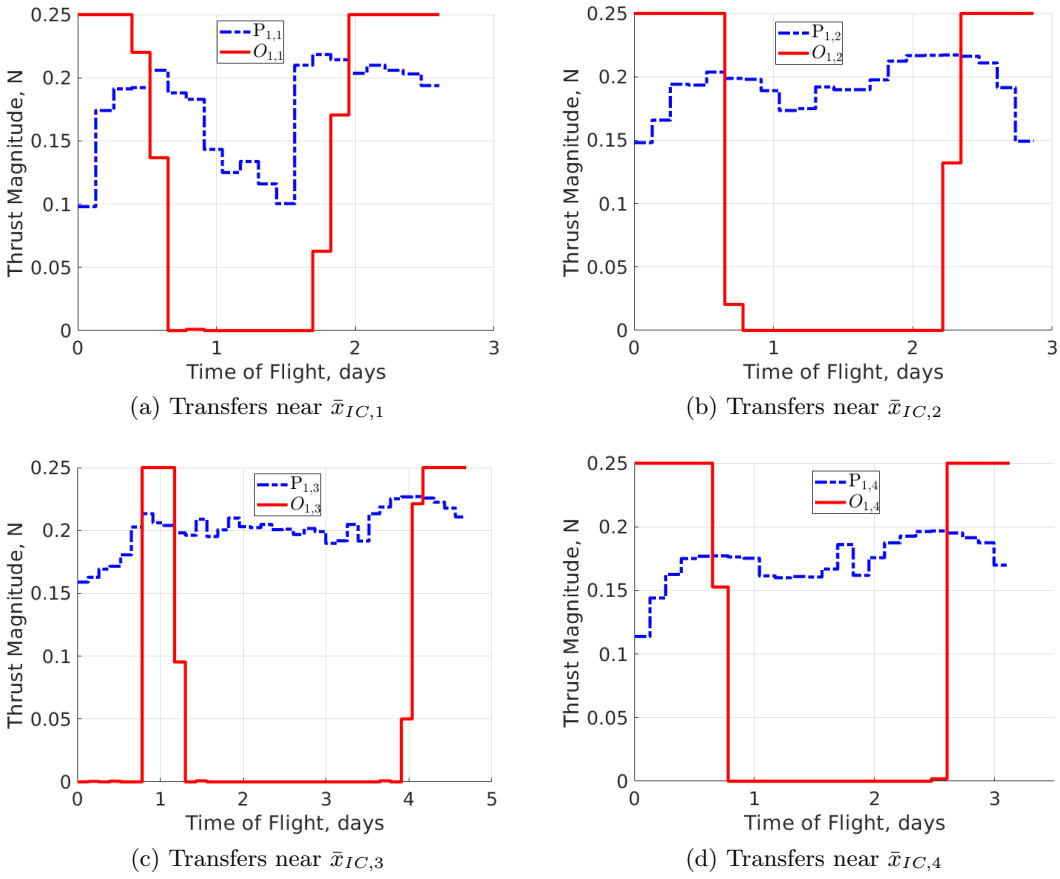
(a) Transfers near $\bar{x}_{IC,1}$ (b) Transfers near $\bar{x}_{IC,2}$

(c) Transfers near $\bar{x}_{IC,3}$ (d) Transfers near $\bar{x}_{IC,4}$

Figure 7.15: Solution space for transfers recovered via optimization while minimizing the propellant mass usage from initial guesses developed using MRPPO.

SQP in *fmincon* only guarantees recovery of a locally optimal solution based on the local basin of convergence for the initial guess, it can be inferred that the initial guess provided by MRPPO is close to the propellant-optimal solution and resulted in the successful recovery of a locally optimal solution with a bang-bang-like control profile. However, further optimization techniques are necessary to recover a purely bang-bang control profile.

The MRPPO results are approximately validated compared to transfers developed by an optimization scheme that both maximize the total reward and minimize the propellant mass usage. For both objective functions, the policies at each departure locations develop transfer geometries that are approximately retained by the optimization scheme. However, the optimal solutions with

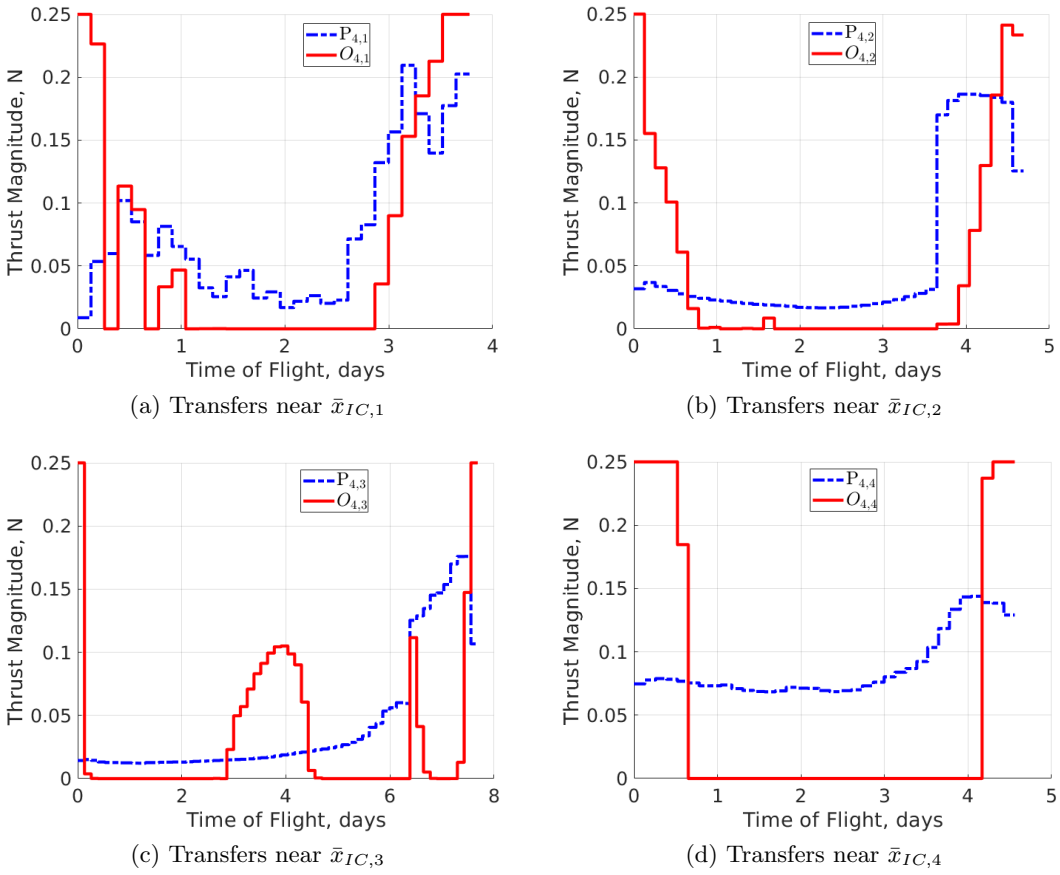


(a) Transfers near $\bar{x}_{IC,1}$ (b) Transfers near $\bar{x}_{IC,2}$

(c) Transfers near $\bar{x}_{IC,3}$ (d) Transfers near $\bar{x}_{IC,4}$

Figure 7.16: Thrust magnitude profiles for low-thrust transfers minimizing the propellant mass usage associated with $\mathrm{P}_{1,i}$ from an L_2 southern halo orbit to a higher-energy L_2 southern halo orbit in the Earth-Moon CR3BP.

shorter flight times tend to require less propellant mass usage than the reference trajectories. Specifically, as the penalty on propellant mass usage increases, the policies recover transfers with characteristics that more closely resemble the optimization solutions. This may be attributed to policies $\mathrm{P}_{3,i}$ and $\mathrm{P}_{4,i}$ prioritizing conserving propellant mass while policies $\mathrm{P}_{1,i}$ and $\mathrm{P}_{2,i}$ have limited incentive to decrease propellant mass usage. Further, while the thrust magnitude profiles of the reference trajectories are updated to bang-bang-like profiles in the optimal solutions, the thrust direction profiles of the reference trajectories produced by MRPPO are minimally updated by the optimization scheme.



(a) Transfers near $\bar{x}_{IC,1}$

(b) Transfers near $\bar{x}_{IC,2}$

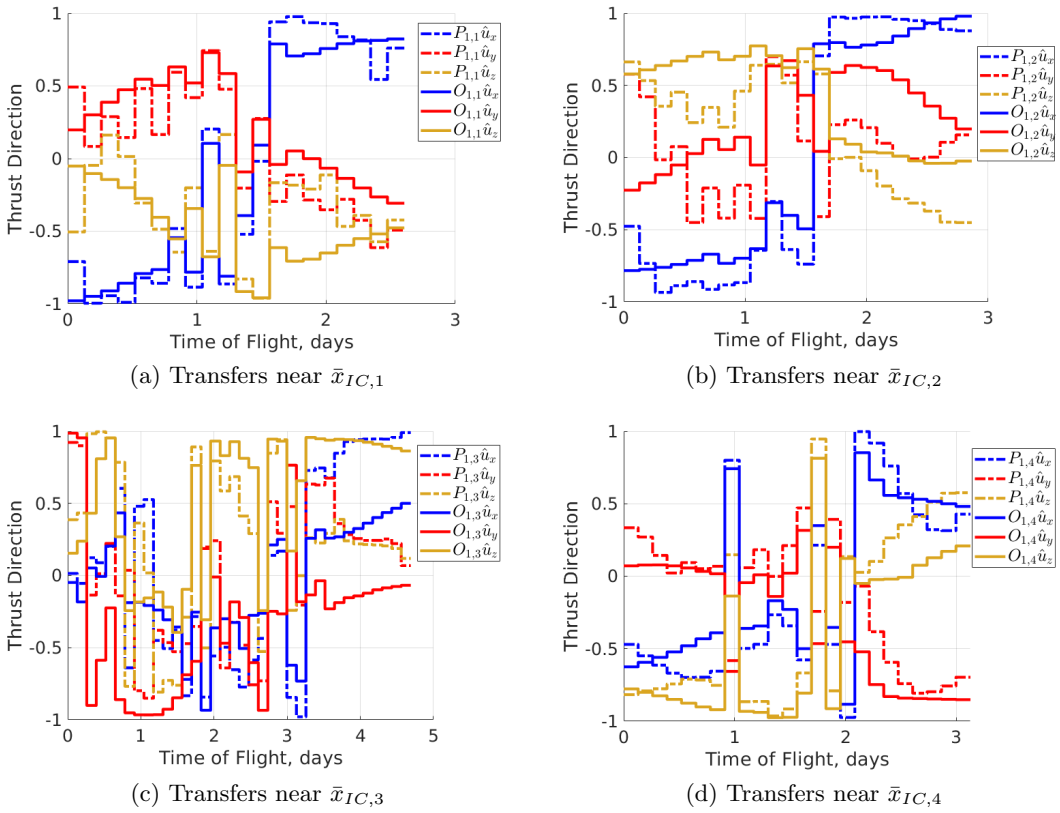
(c) Transfers near $\bar{x}_{IC,3}$

(d) Transfers near $\bar{x}_{IC,4}$

Figure 7.17: Thrust magnitude profiles for low-thrust transfers minimizing the propellant mass usage associated with $\mathrm{P}_{4,i}$ from an L_2 southern halo orbit to a higher-energy L_2 southern halo orbit in the Earth-Moon CR3BP.

7.1.4 MRPPO Hyperparameter Exploration

Throughout training, the selected hyperparameters for MRPPO significantly influence the resulting policies: poor selections may prevent policies from converging to better locally optimal solutions. For the hyperparameter selection exploration, MRPPO is used to train policies using the initial conditions associated with Scenario #1 defined in Table 5.2 and with the baseline hyperparameters defined in Table 5.5. In this exploration, the hyperparameters are individually modified to explore the influence of the hyperparameter on the behavior of the policies throughout training. Then, the trained policies for each set of hyperparameters are evaluated on a common set of per-



(a) Transfers near $\bar{x}_{IC,1}$

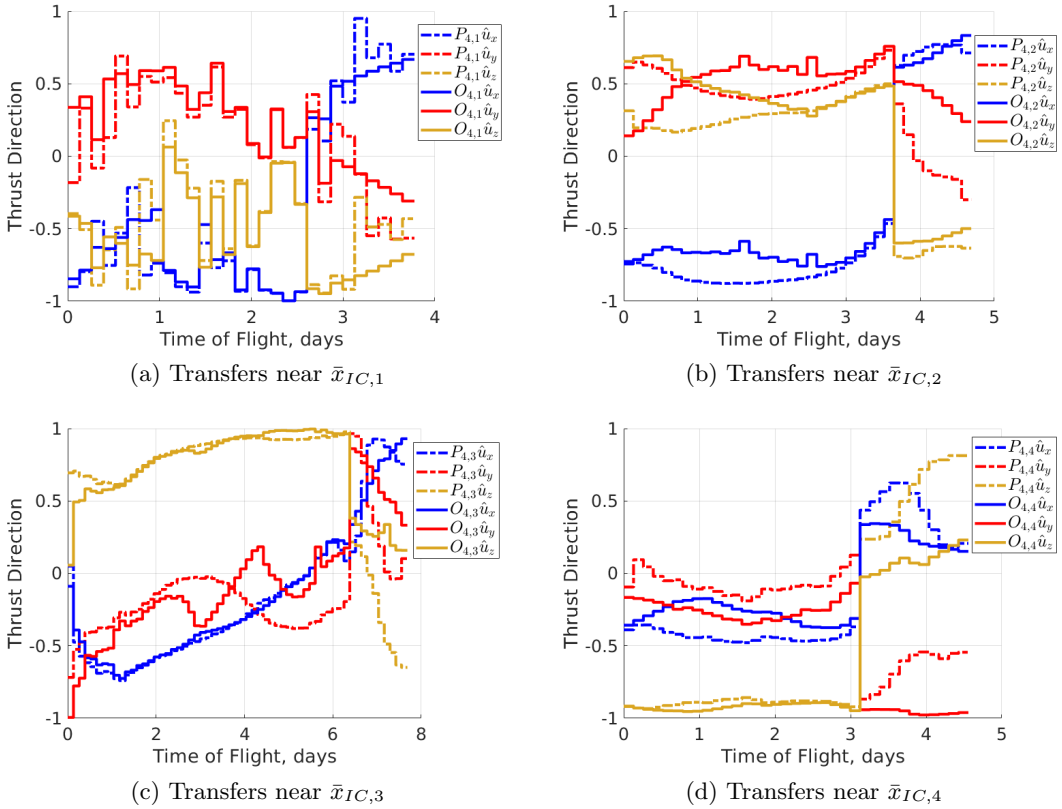
(b) Transfers near $\bar{x}_{IC,2}$

(c) Transfers near $\bar{x}_{IC,3}$

(d) Transfers near $\bar{x}_{IC,4}$

Figure 7.18: Thrust direction profiles for low-thrust transfers minimizing the propellant mass usage associated with $\mathrm{P}_{1,i}$ from an L_2 southern halo orbit to a higher-energy L_2 southern halo orbit in the Earth-Moon CR3BP.

turbed initial conditions to generate fundamental insights into the influence of each hyperparameter selection. Their effectiveness is evaluated using the total reward which is measured by summing the rewards from the evaluation trajectories for all four policies. Similar to Section 7.1.2, the total reward is computed for every tenth update to the policies to study the evolution of the total reward throughout training. Additionally, three values are again used to determine the influence of the hyperparameter selection on the behavior of MRPPO: (1) the maximum total reward simulation, (2) the minimum total reward simulation, and (3) the mean of the total reward of all five simulations. These values are calculated to generate insight into how each hyperparameter affects the training of policies on this scenario using MRPPO.

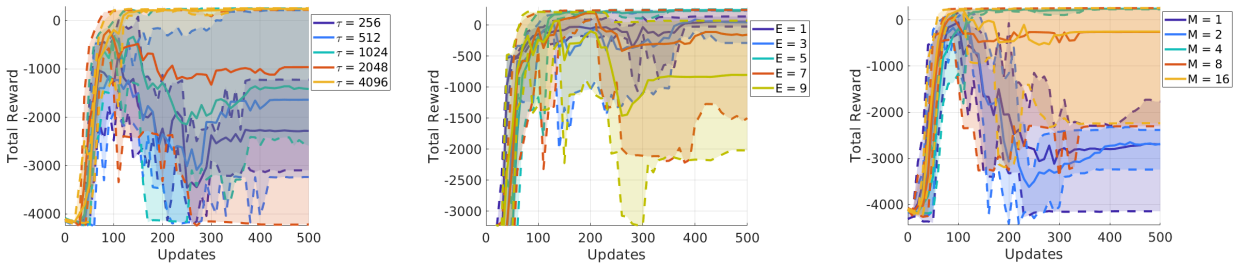


(a) Transfers near $\bar{x}_{IC,1}$ (b) Transfers near $\bar{x}_{IC,2}$

(c) Transfers near $\bar{x}_{IC,3}$ (d) Transfers near $\bar{x}_{IC,4}$

Figure 7.19: Thrust direction profiles for low-thrust transfers minimizing the propellant mass usage associated with $\mathrm{P}_{4,i}$ from an L_2 southern halo orbit to a higher-energy L_2 southern halo orbit in the Earth-Moon CR3BP.

First, Fig. 7.20a depicts the reward for distinct values of the number of environmental steps per update used during training. Generally, as the number of environmental steps used during training increases, the mean total reward increases. Additionally, higher numbers of environmental steps also produce higher minimum and maximum total rewards throughout the training process. This trend generally demonstrates the benefit of incorporating more information into the updates of the neural networks at the expense of increased training times and memory requirements. The number of epochs and mini batches, however, displayed in Figs. 7.20b and 7.20c respectively, used during the update phase also influences the increase in the total reward throughout training. The best scenario is found by not setting the hyperparameters to the smallest or largest values, but

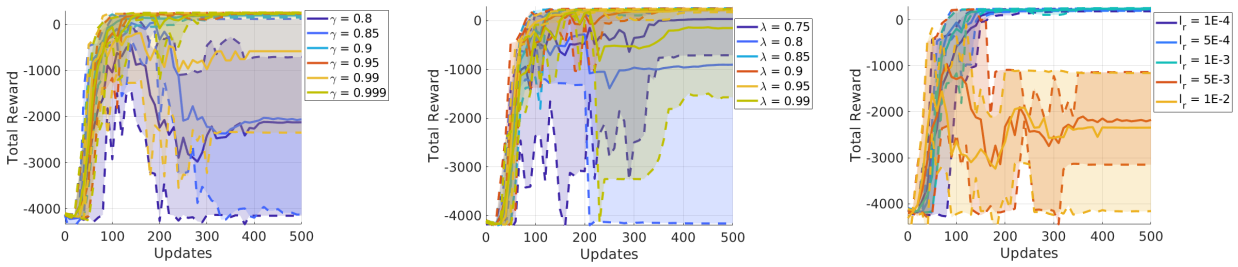


(a) Environmental steps (b) Epochs (c) Mini batches

Figure 7.20: Total reward for policies trained using MRPPO applied for transfer design between Earth-Moon L_2 halo orbits.

instead, setting the hyperparameters to intermediate values. This observation may be attributed to both the sensitive environment and the data gathered in a single update being used to update the networks too many times, thereby biasing the training and forcing the policies to converge to sub-optimal results. Instead, a balanced approach is more suitable where the data is used to update the networks multiple times, but not too many so as to produce substandard policies.

Further insights are generated by examining the hyperparameters used within each update to the neural networks, including the discount factor, GAE factor, and learning rate, which influence how the data is used within a single update to the neural networks. Figure 7.21a depicts the influence of the discount factor, using six values. For this scenario, setting $\gamma \geq 0.9$ produces high mean, maximum, and minimum total reward and enables future rewards to be considered without under-estimating their influence. However, the GAE factor exploration displayed in Fig. 7.21b reveals that selecting the GAE factor necessitates a balanced approach such that $0.85 \leq \lambda \leq 0.95$ produces high mean total rewards across multiple runs. While the values for the discount factor and GAE factor do influence the training, the value for the learning rate impacts training more significantly. In fact, high learning rate values result in policy divergence and suboptimal behavior. Figure 7.21c depicts the total reward for five learning rate values. The higher learning rate produces policies that significantly diverge while lower learning rates yield more consistent total rewards across policies. Specifically, setting $l_r = 1 \times 10^{-3}$ produces stable behavior for



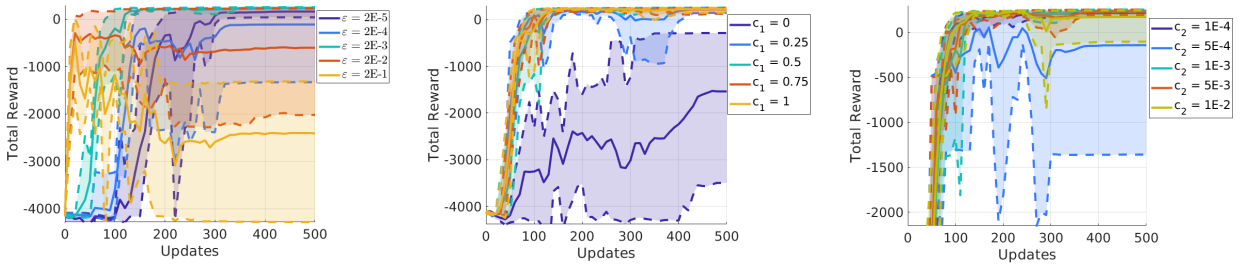
(a) Discount factor (b) GAE factor (c) Learning rate

Figure 7.21: Total reward for policies trained using MRPPO applied for transfer design between Earth-Moon L_2 halo orbits.

the policies while returning slightly higher mean, maximum, and minimum total rewards. These three hyperparameter explorations demonstrate that properly selecting these values that govern the updates to the neural networks is a necessity for training policies with efficient and stable convergence properties and for generating a high, final total reward.

Three additional hyperparameters are used within the objective function defined in Eqs. (3.14) and (4.2): the clipping parameter, value function coefficient, and entropy coefficient. Figure 7.22a depicts the results of five clipping parameter values. Higher values for the clipping parameter produce a faster initial convergence but significant potential for divergence while low values result in a slower and smoother convergence of the policies due to the update size being limited. The clipping parameter exploration demonstrates a balanced approach is necessary such that setting $\varepsilon = 2 \times 10^{-3}$ in this scenario provides efficient and smooth convergence while also leading to the highest mean total reward. Then, Fig. 7.22b displays the influence of the value function coefficient with $c_1 \geq 0.5$ producing high mean, maximum, and minimum total rewards. Similarly, for the entropy coefficient, Fig. 7.22c illustrates that the policies converge to high total reward with no significant difference in performance except when $c_2 = 5 \times 10^{-4}$. Each of these hyperparameters directly affects the evaluation of the objective function during training leading to distinct behavior when evaluated on the same initial conditions.

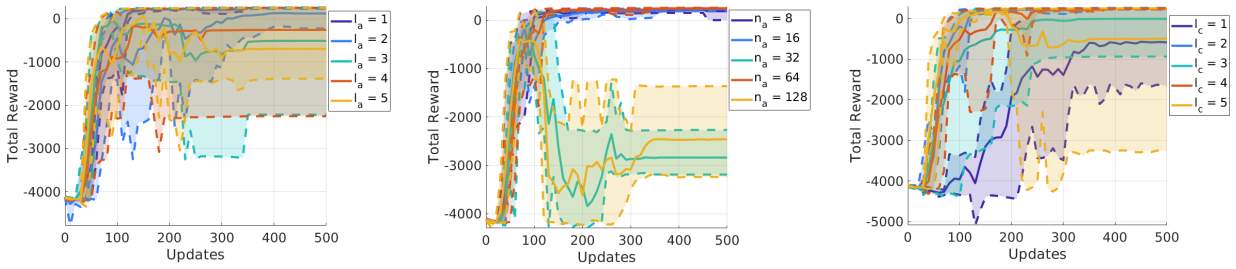
Six construction parameters are also explored to analyze their influence on the trained poli-



(a) Clipping parameter (b) Value function coefficient (c) Entropy coefficient

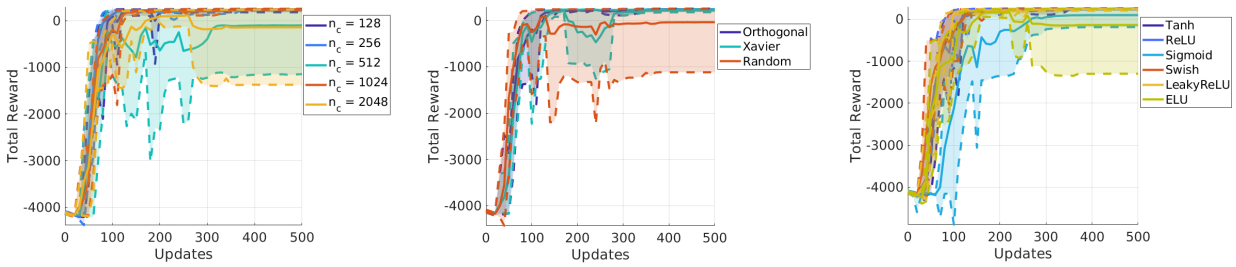
Figure 7.22: Total reward for policies trained using MRPPO applied for transfer design between Earth-Moon L_2 halo orbits.

cies. Figure 7.23a depicts the total reward for five values of the number of hidden layers in the actor neural network for each policy. The exploration demonstrates that less hidden layers in the actor neural network tends to produce higher total rewards. Similarly, Fig. 7.23b displays the results for the number of hidden nodes per actor hidden layer demonstrating that 8, 16, and 64 nodes per actor hidden layer produce high maximum, minimum, and mean total rewards. This suggests that less actor nodes per hidden layer may produce more consistent behavior across runs while higher values may be more inconsistent. However, the exploration into the number of hidden layers in the critic neural network in Fig. 7.23c produces distinct results with $2 \leq l_c \leq 4$ delivering significantly higher mean, maximum, and minimum total rewards than $l_c = 1$ and $l_c = 5$. Due to similar higher maximum, minimum, and mean total rewards, selecting the critic neural network to have two hidden layers reduces the memory requirements compared to a higher number of critic hidden layers. The number of hidden nodes per hidden layer for the critic neural network displayed in Fig. 7.24a also demonstrates a trend whereby smaller values tend to produce higher mean total rewards although all selections produce at least one run with a mean total reward above 200. The results indicate that smaller numbers of hidden layers and hidden nodes for both the actor and critic neural networks tend to produce better results for this specific trajectory design scenario and set of runs, potentially due to the number of parameters in the neural networks being a much smaller multi-dimensional space.



(a) Actor layers (b) Actor nodes (c) Critic layers

Figure 7.23: Total reward for policies trained using MRPPO applied for transfer design between Earth-Moon L_2 halo orbits.



(a) Critic nodes (b) Weight initialization method (c) Activation function

Figure 7.24: Total reward for policies trained using MRPPO applied for transfer design between Earth-Moon L_2 halo orbits.

The final two construction parameters, the weight initialization method and the activation function selection affect both the actor and critic neural networks. Figure 7.24b displays the results of using different initialization strategies for the weights of the neural networks. This figure reveals that the initial weights of each network slightly influence the resulting behavior of the policy in this scenario where orthogonal and Xavier initialization schemes produce high maximum, minimum, and mean total rewards for all five simulations. Finally, Fig. 7.24c depicts the results of varying activation function used within the networks. All activation functions except ELU and sigmoid produce high total maximum, minimum, and mean total rewards across the five simulations.

Overall, the baseline hyperparameter set listed in Table 5.5 produces stable and consistent behavior throughout the training process across five simulations for this simple trajectory design

scenario. These results indicate the influence of altering each hyperparameter on the training process. While some hyperparameters may be altered to produce slightly better performance, the baseline hyperparameter set serves as the foundation for the hyperparameters selected in the more complex trajectory design scenarios explored in this research.

7.2 Low-Thrust Transfers from an Earth-Moon L_1 Halo Orbit to an Earth-Moon L_2 Halo Orbit

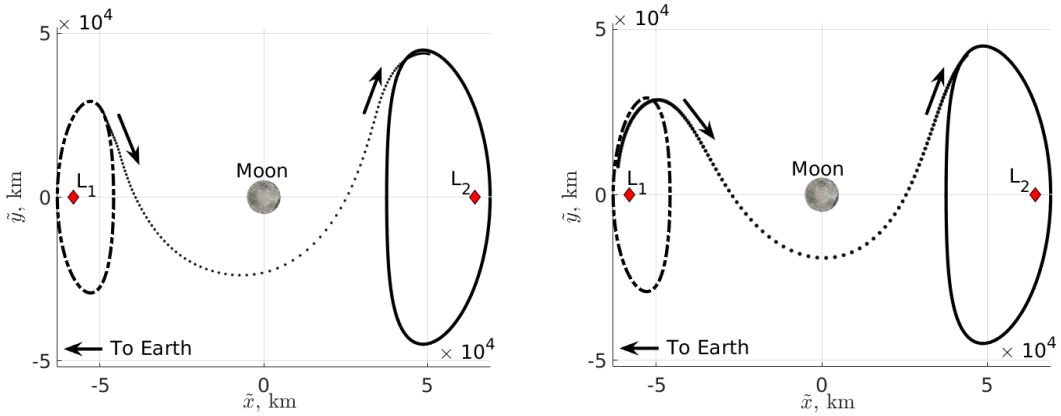
MRPPO is used to train four policies that recover low-thrust transfers between halo orbits at distinct libration points in the Earth-Moon system. Specifically, a low-thrust enabled SmallSat is initialized near an L_1 northern halo orbit and targets an L_2 southern halo orbit in the Earth-Moon CR3BP. Recall, the initial conditions are drawn from anywhere along the initial periodic orbit and randomly perturbed in both position and velocity. The four policies trained by MRPPO are assigned distinct reward functions to encourage the policies to uncover distinct solutions within the multi-objective solution space balancing flight time and propellant mass usage. Once the four policies are trained, the policies are evaluated on 1,000 initial conditions to facilitate direct comparisons between the policies and generate insights into the flight time and propellant mass usage trade space. Characteristics of both the evaluation and reference trajectories are explored to generate further insights. Then, the reference trajectories associated with each policy are used to construct initial guesses that are subsequently input to the optimization problem solved via *fmincon* for both objective functions. The resulting solution spaces from both methodologies are used to validate the results produced by MRPPO.

7.2.1 Transfers Recovered via MRPPO

Four policies are trained using MRPPO to construct transfers for low-thrust-enabled SmallSats from an L_1 northern halo orbit to an L_2 southern halo orbit in the Earth-Moon CR3BP. Specifically, policy P_1 has no penalty on propellant mass usage while policy P_4 is assigned the highest penalty on propellant mass usage to encourage a variety of solutions within the multi-

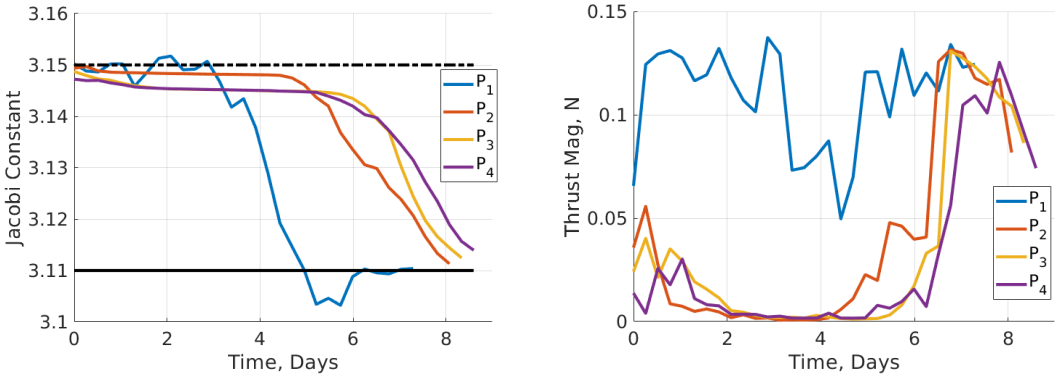
objective solution space. Once trained, both the policies and resulting reference trajectories for each policy are evaluated to uncover a subset of solutions within the propellant mass usage and flight time trade space.

The reference trajectories are the best trajectories generated throughout the training process for each policy according to the evaluation criteria. Figures 7.25a and 7.25b depict the reference trajectories developed by policies P_1 and P_4 respectively, where the initial orbit is denoted by a black dashed arc, the final orbit by a solid black arc, and the reference trajectories are represented by dotted black arcs. Note,each policy develops a single reference trajectory and the reference trajectories begin near a state along the initial periodic orbit due to the perturbations induced on the state drawn along the orbit. While the trajectories may possess similar geometries, Figs. 7.26a and 7.26b demonstrate that the Jacobi constant and thrust magnitude profiles of each reference trajectory differ reflecting increases in the propellant mass usage penalty from policy P_1 to policy P_4. Specifically, policy P_1 produces a higher thrust magnitude throughout the beginning of the transfer corresponding to an increase in the energy of the spacecraft. Conversely, policy P_4 initially produces negligible thrust and coasts until the spacecraft approaches the vicinity of the final periodic orbit. Policies P_2 and P_3 exhibit intermediary behavior with smaller levels of thrust than P_1 but larger than P_4. However, when the reference trajectories for policies P_2, P_3, and P_4 do approach the final periodic orbit, the thrust magnitude significantly increases resembling a bang-bang-like profile. This behavior suggests that the reference trajectories are following motion that resembles an unstable invariant manifold trajectory departing the initial periodic orbit until the reference trajectories approach the final orbit. This phenomenon has been explored by Anderson and Lo and suggests that the policies are autonomously recovering near-optimal behavior that models this known dynamical phenomenon [5]. Additionally, the reference trajectory developed by policy P_2 possesses a significantly distinct thrust magnitude and Jacobi constant profile compared to policies P_2, P_3, and P_4 demonstrating that the policies recovered distinct solutions. Finally, since the policies are only trained to enter within a 5×10^{-3} hypersphere in position and velocity around the final orbit, a small difference in the Jacobi constant at the conclusion of the transfer is



(a) Policy P_1 (b) Policy P_4

Figure 7.25: Reference trajectories developed by policies P_1 and P_4 trained using MRPPO (Moon not to scale).

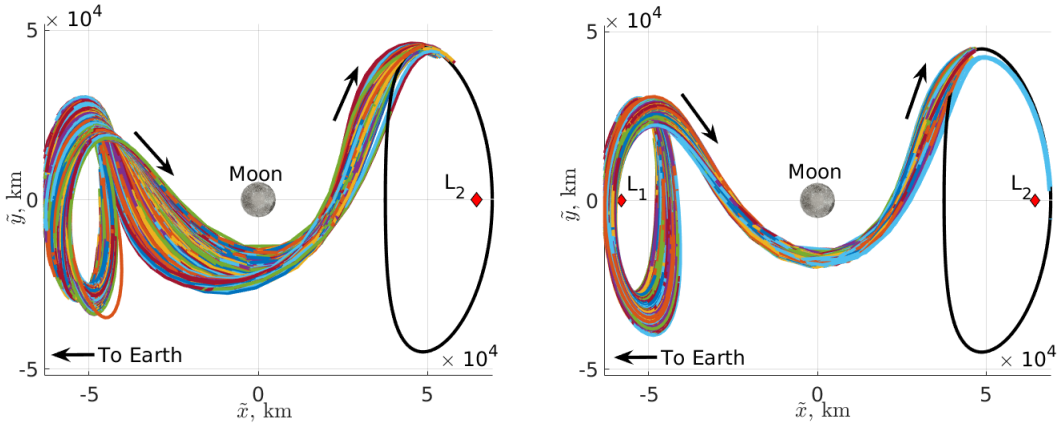


(a) Jacobi constant (b) Thrust magnitude

Figure 7.26: Characteristics of the reference trajectories generated by each policy during training.

evident. These reference trajectories are selected from the best trajectories generated during the training phase and serve as guides for the policies to construct transfers between the orbits.

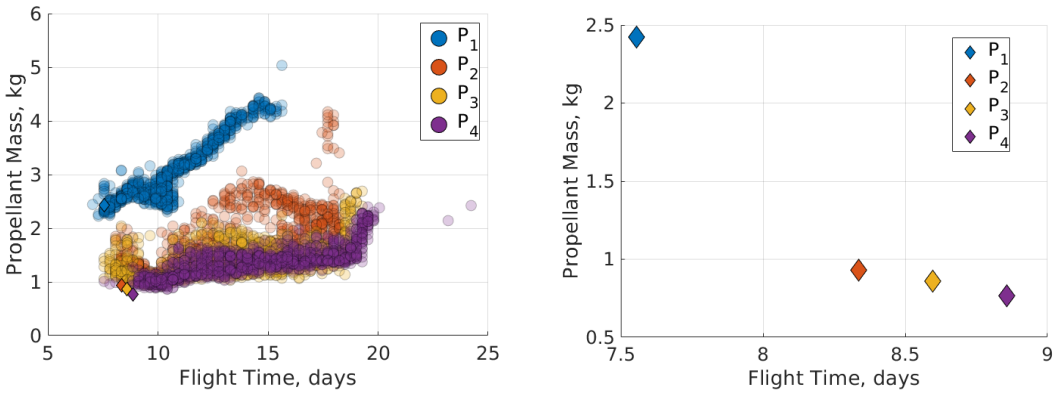
The trained policies are also evaluated on a common set of 1,000 initial conditions perturbed from the initial periodic orbit to facilitate direct comparisons between the policies. Figures 7.27a and 7.27b depict the 1,000 evaluation trajectories constructed by policies P_1 and P_4, respectively, and shaded in a variety of hues to aid in differentiation between the transfers. Both policies produce transfers that resemble the generated reference trajectories for each policy prior



(a) Policy P_1 (b) Policy P_4

Figure 7.27: Evaluation trajectories developed by policies P_1 and P_4 trained using MRPPO (Moon not to scale).

to converging on the final periodic orbit. Further, all of the policies tend to converge on the same transfer geometry. Additionally, the propellant mass usage and flight time is calculated for each evaluation trajectory. Figure 7.28a portrays the propellant mass usage and flight time values for each evaluation trajectory as circles and reference trajectory as diamonds constructed by each policy. The large dispersion in flight times and propellant mass usages may be attributed to the requirement for certain evaluation trajectories to follow the initial periodic orbit for nearly an entire orbital period prior to departing along the reference trajectory. The resulting trade space demonstrates a clear trend across the policies whereby propellant mass usage decreases at the expense of an increase in flight time as the propellant mass usage penalty increases. Similarly, Fig. 7.28b focuses on the propellant mass usage and flight time for each reference trajectory mirroring the trend in the evaluation trajectories. The reference trajectories for each policy accordingly correspond to the best trajectories generated and serve as a valuable basis for the initial guesses used in the optimization scheme.



(a) Evaluation and reference trajectories

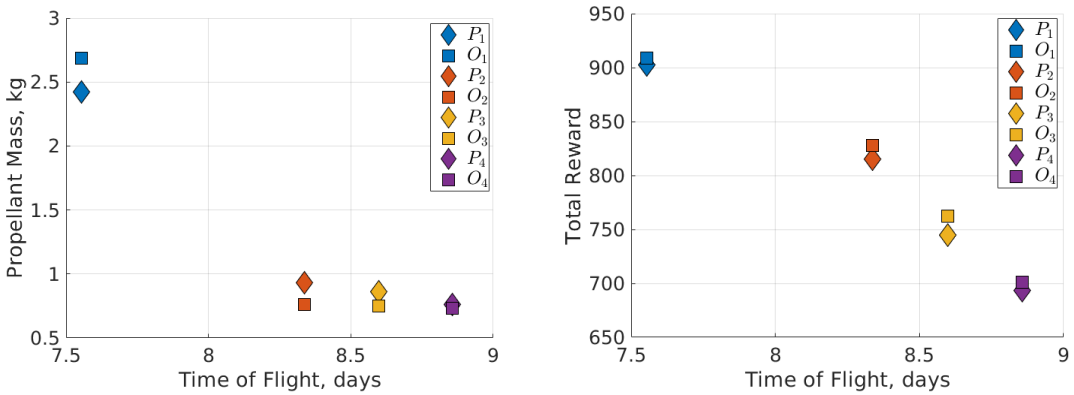
(b) Reference trajectories

Figure 7.28: Propellant mass usage and flight time characteristics for the evaluation trajectories and reference trajectories generated by policies trained using MRPPO.

7.2.2 Transfers Recovered via Classical Optimization

The results autonomously generated by MRPPO are used to construct initial guesses for optimization in SQP implemented in *fmincon*. Similar to Section 7.1.3, the locally optimal solutions developed by SQP are denoted O_j, corresponding to policies P_j for $j = [1, 2, 3, 4]$. Recall, both objective functions explored for this optimization scheme are not equivalent to but only similar to the reward function used in MRPPO to train the policies. However, exploring both objective functions further validates the results produced by MRPPO. The initial guesses are optimized to minimize both objective functions while continuity is ensured via hard constraints. Then, *fmincon* with SQP is used to recover a distinct solution space.

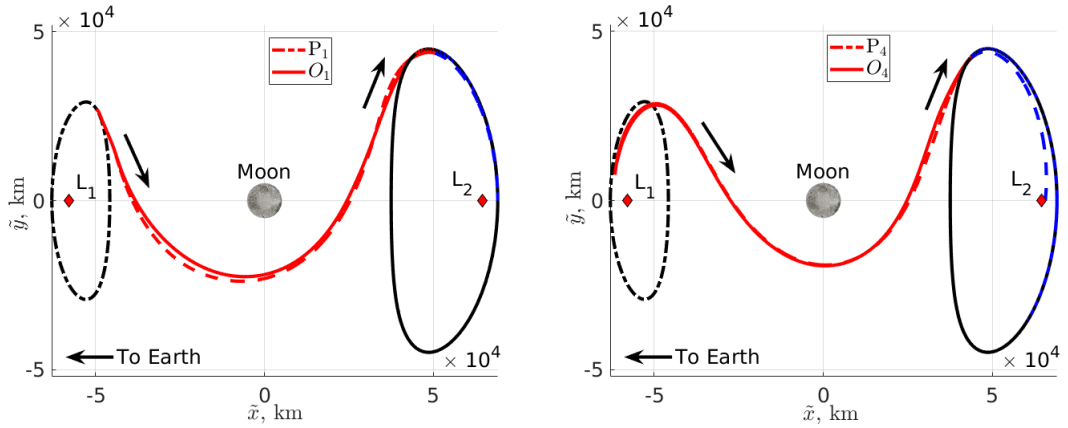
Reference trajectories produced by MRPPO form the initial guess to the optimization algorithm for the first objective function. Figures 7.29a and 7.29b depict the flight time and propellant mass usage and flight time and total reward trade spaces, respectively, generated by MRPPO and *fmincon* with SQP. For this objective function, which aims to maximize the total reward across a trajectory, SQP recovers solutions with slightly higher total rewards for all four transfers. Further, with the exception of the solutions associated with policy P_1, SQP develops transfers with lower propellant mass usages.



(a) Flight time and propellant mass usage

(b) Flight time and total reward

Figure 7.29: Solution space recovered via optimization while maximizing the total reward from initial guesses developed using MRPPO.

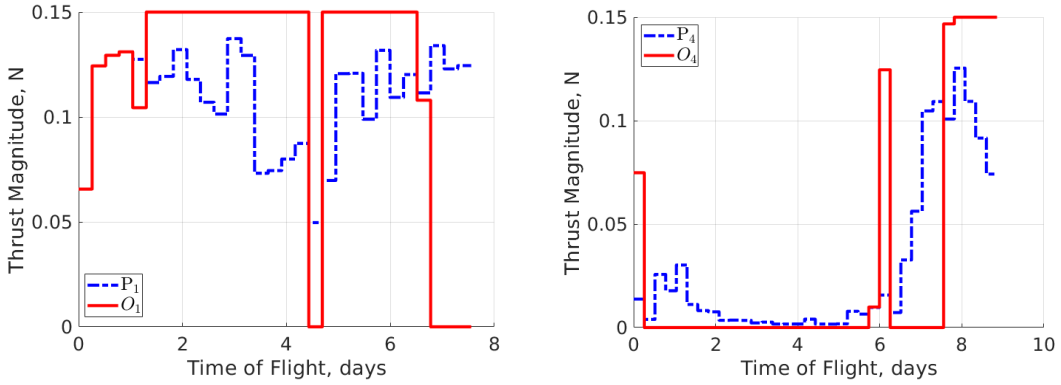


(a) Policy P_1 transfers

(b) Policy P_4 transfers

Figure 7.30: Low-thrust transfers maximizing the total reward associated with policies P_1 and P_4 from an L_1 northern halo orbit to an L_2 southern halo orbit in the Earth-Moon CR3BP (Moon not to scale).

Examining the solutions produced by both algorithms further enables the validation of the results produced by MRPPO. Figure 7.30a depicts the initial guess and optimal trajectory developed by SQP for policies P_1 and P_4. In these figures, the red arcs correspond to low-thrust nodes while the blue arcs denote coast nodes. The geometry of the two transfers closely resemble one another with small differences in the solutions produced by SQP due to the differences in the implementations.

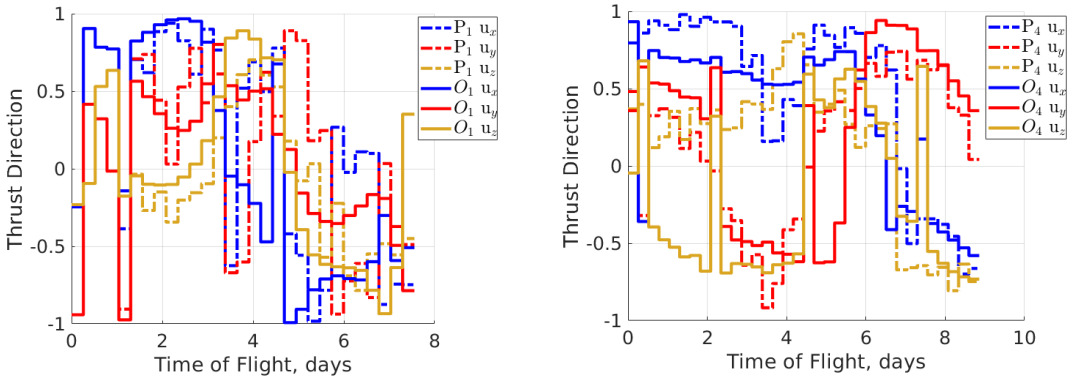


(a) Policy P_1 (b) Policy P_4

Figure 7.31: Thrust magnitude profiles for low-thrust transfers maximizing the total reward from an L_1 northern halo orbit to an L_2 southern halo orbit in the Earth-Moon CR3BP.

The thrust profiles are also examined to generate insights into the difference in transfers recovered by the two methodologies for this objective function. Figure 7.31 displays the thrust magnitude profiles for the initial guess and optimal trajectory developed by SQP for policies P_1 and P_4. Additionally, Fig. 7.32 depicts the thrust direction profiles for both policies and trajectories. Both the thrust magnitude and direction profiles for the optimal trajectories resemble the profiles for the initial guess. This further demonstrates that MRPPO recovers near-optimal solutions compared to SQP for objective functions maximizing the total reward across the trajectory, an objective function that more closely reflects the reward functions used in MRPPO.

Using the second objective function, SQP may be used to update the initial guesses to maximize the final spacecraft mass via the second objective function. Figure 7.33 displays the propellant mass usage and flight time trade spaces recovered by *fmincon* with SQP using the initial guesses constructed by MRPPO. For all four policies, *fmincon* with SQP recovers transfers with lower propellant mass usage than the transfers recovered by MRPPO. However, except for policy P_1 which has no incentive to decrease propellant mass usage, the difference in propellant mass usage is minimal and decreases as the flight time increases. The differences in propellant mass usage may be attributed to the differences in implementations; namely, the methodologies are recovering



(a) Policy P_1 (b) Policy P_4

Figure 7.32: Thrust direction profiles for low-thrust transfers maximizing the total reward from an L_1 northern halo orbit to an L_2 southern halo orbit in the Earth-Moon CR3BP.

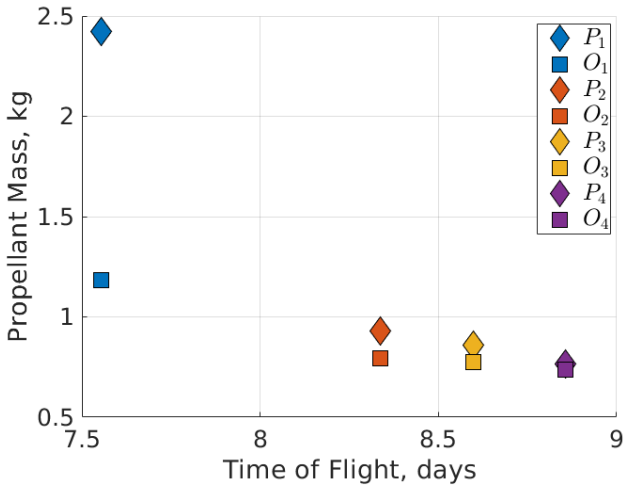
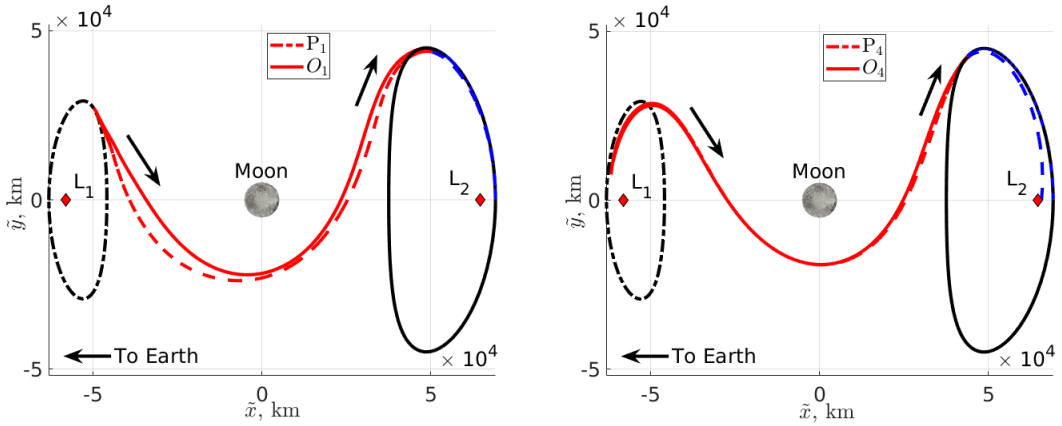


Figure 7.33: Solution space recovered via optimization while minimizing the propellant mass usage from initial guesses developed using MRPPO.

distinct but related solution spaces. Both MRPPO and SQP develop related solution spaces and demonstrate that as the flight time increases, the required propellant masses for both local minima approach similar values.

The low-thrust transfers associated with policies P_1 and P_4 provide additional insight into solutions produced by SQP. Figure 7.34 depicts two transfers associated with policies P_1 and P_4 using the same color configuration as Fig. 7.30. Both optimized transfers possess geometries



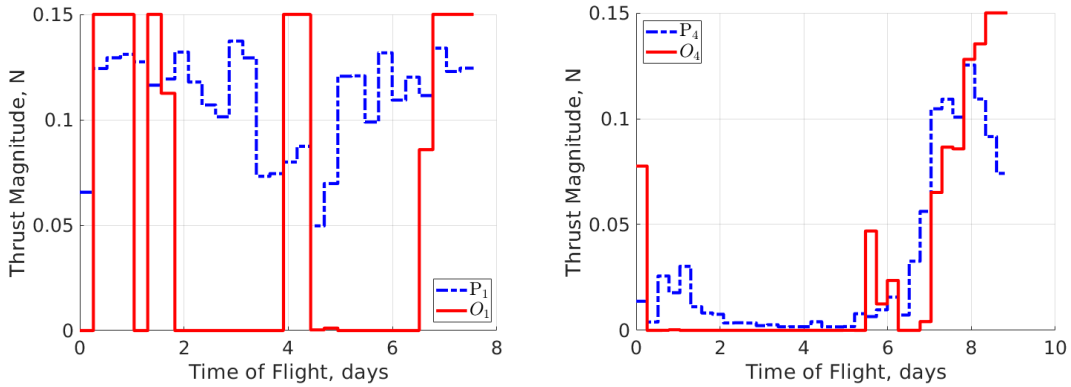
(a) Policy P_1 transfers

(b) Policy P_4 transfers

Figure 7.34: Low-thrust transfers minimizing the propellant mass usage associated with policies P_1 and P_4 from an L_1 northern halo orbit to an L_2 southern halo orbit in the Earth-Moon CR3BP (Moon not to scale).

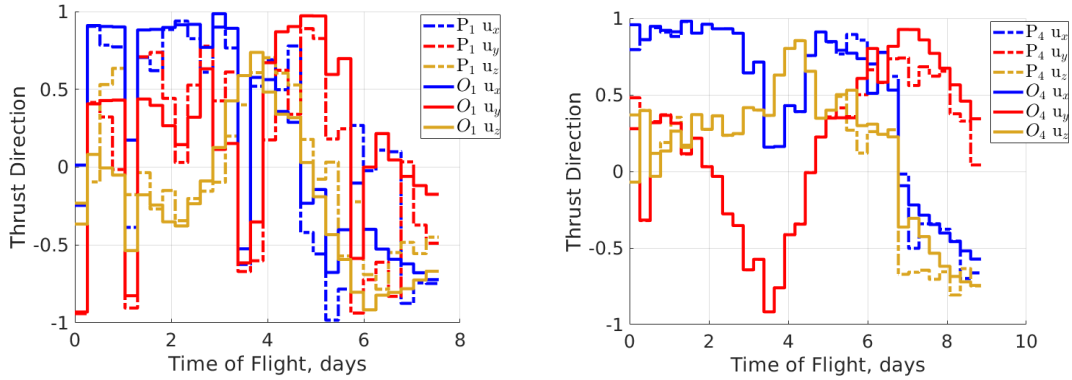
that closely resemble the initial guesses developed by MRPPO. This observation demonstrates that MRPPO generates initial guesses that sufficiently predict optimal solutions produced by an optimization scheme.

Figures 7.35 and 7.36 depict the thrust magnitude and direction profiles associated with the MRPPO and SQP transfers for policies P_1 and P_4, respectively. In these figures, the thrust magnitude profile associated with policy P_1 is altered by SQP to resemble a bang-bang like profile with additional changes to the thrust direction. These changes may be due to the differences in the objective functions between the two methodologies. However, the thrust profile for policy P_4 is negligibly altered by SQP. The results produced by the optimization scheme suggest that the results generated by MRPPO lie close to the local optima in the propellant mass usage and flight time solution space. While *fmincon* with SQP may recover transfers with a lower propellant mass usage for transfers with lower flight times, the difference in propellant mass usage decreases as the flight time increases. Both sets of results validate that MRPPO is autonomously recovering near-optimal solutions for low-thrust transfers from an L_1 northern halo orbit to an L_2 southern halo orbit in the Earth-Moon CR3BP.



(a) Policy P_1 (b) Policy P_4

Figure 7.35: Thrust magnitude profiles for low-thrust transfers minimizing the propellant mass usage from an L_1 northern halo orbit to an L_2 southern halo orbit in the Earth-Moon CR3BP.



(a) Policy P_1 (b) Policy P_4

Figure 7.36: Thrust direction profiles for low-thrust transfers minimizing the propellant mass usage from an L_1 northern halo orbit to an L_2 southern halo orbit in the Earth-Moon CR3BP.

For this trajectory design scenario, MRPPO produces solutions that more closely resemble the transfers developed by an optimization scheme. This may be attributed to the ability of the policies to generate transfers with bang-bang-like control profiles in this trajectory design scenario whereas in the previous trajectory design scenario, the policies produce a more continuous control profile throughout the reference trajectories.

7.2.3 Varying Number of Reference Trajectories per Policy

The number of reference trajectories designed by each policy is also varied to generate insights into the behavior of the policies as the number of reference trajectories increases. Recall, the initial periodic orbit is segmented into arcs of equal time and a policy updates a reference trajectory originating from each segment of the initial periodic orbit. More reference trajectories per policy may lead to the recovery of a wider variety of transfer geometries for the evaluation trajectories. However, too many reference trajectories may prohibit a policy from updating each reference trajectory consistently leading to poor behavior for evaluation trajectories initialized along certain segments of the initial periodic orbit.

The number of reference trajectories per policy is varied from one to sixty four and twenty simulations with four policies are run for each set number of reference trajectories. Figure 7.37 depicts a box plot exhibiting the median, quartiles, and outliers of the total reward for the set of twenty simulations evaluated on the common set of 1,000 initial conditions for low-thrust transfers originating near an L_1 northern halo orbit to an L_2 southern halo orbit in the Earth-Moon CR3BP. In this trajectory design scenario, as the number of reference trajectories per policy increases, the total reward across policies tends to decrease. However, simulations using one, two, and four reference trajectories per policy achieve total rewards greater than 2000 demonstrating that some simulations with more reference trajectories per policy are recovering strong behavior in this trajectory design scenario. These simulations are examined to generate further insights into how the number of reference trajectories per policy influences the behavior of the trained policies.

The simulations with the highest total reward for each number of reference trajectories are used to examine the influence of the number of reference trajectories per policy on the recovered low-thrust transfers. Figure 7.38 depicts the reference trajectories constructed by policies P_1 and P_4 for the simulations with the highest total reward using between two and sixty four reference trajectories while the results from using one reference trajectory per policy are exhibited in Section 7.2.1. The reference trajectories and the originating arcs along the initial orbit are depicted using the same

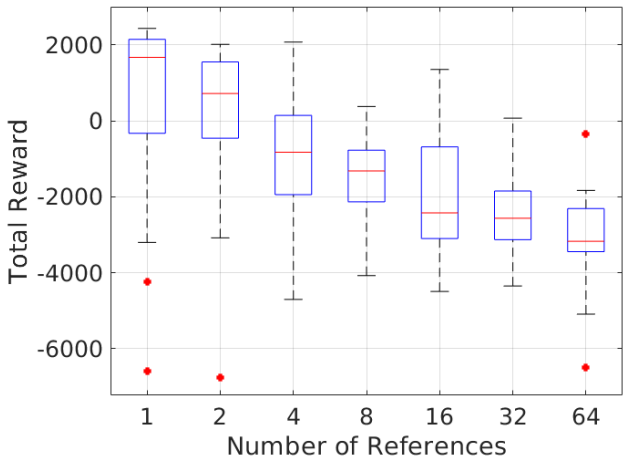
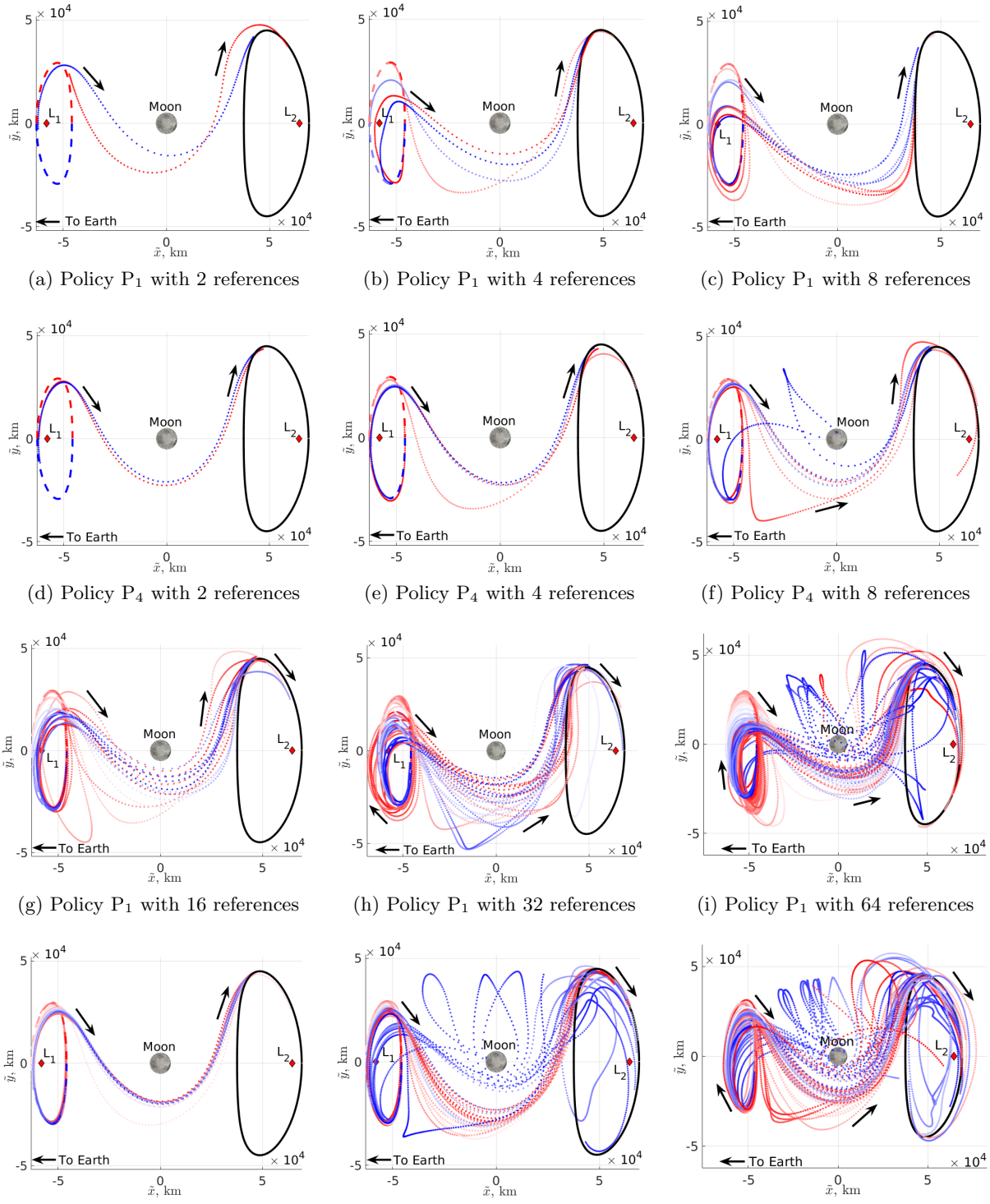


Figure 7.37: Box plot of total reward across twenty simulations while varying the number of reference trajectories per policy.

color composed of shades of red and blue to map the reference trajectory that the policy developed to each segment of the initial periodic orbit. Policies P_1 and P_4 are able to consistently recover transfers for one, two, four, eight, and sixteen reference trajectories per policy. In fact, while most of the reference trajectories target the final orbit with similar direct geometries, policy P_4, when assigned eight reference trajectories, is able to recover a reference trajectory that leverages a revolution about the Moon prior to targeting the final orbit. This demonstrates how using more than one reference trajectory per policy may produce a wider variety of transfer geometries. However, with the update procedure, for both policies P_1 and P_4 with thirty two and sixty four reference trajectories per policy, a significant number of the reference trajectories fail to converge on the final orbit demonstrating that the policies did not update the reference trajectories sufficiently to converge on the final orbit.

Additional insights into the influence of the number of references trajectories per policy are gathered from examining the propellant mass usage and flight time solution spaces of the evaluation and reference trajectories. Figure 7.39 depicts the flight time and propellant mass usage solution space for each number of reference trajectories while Fig. 7.40 displays the evaluation trajectories developed by policies P_1 and P_4. Note, these figures depict only the evaluation trajectories that



(a) Policy P_1 with 2 references (b) Policy P_1 with 4 references (c) Policy P_1 with 8 references

(d) Policy P_4 with 2 references (e) Policy P_4 with 4 references (f) Policy P_4 with 8 references

(g) Policy P_1 with 16 references (h) Policy P_1 with 32 references (i) Policy P_1 with 64 references

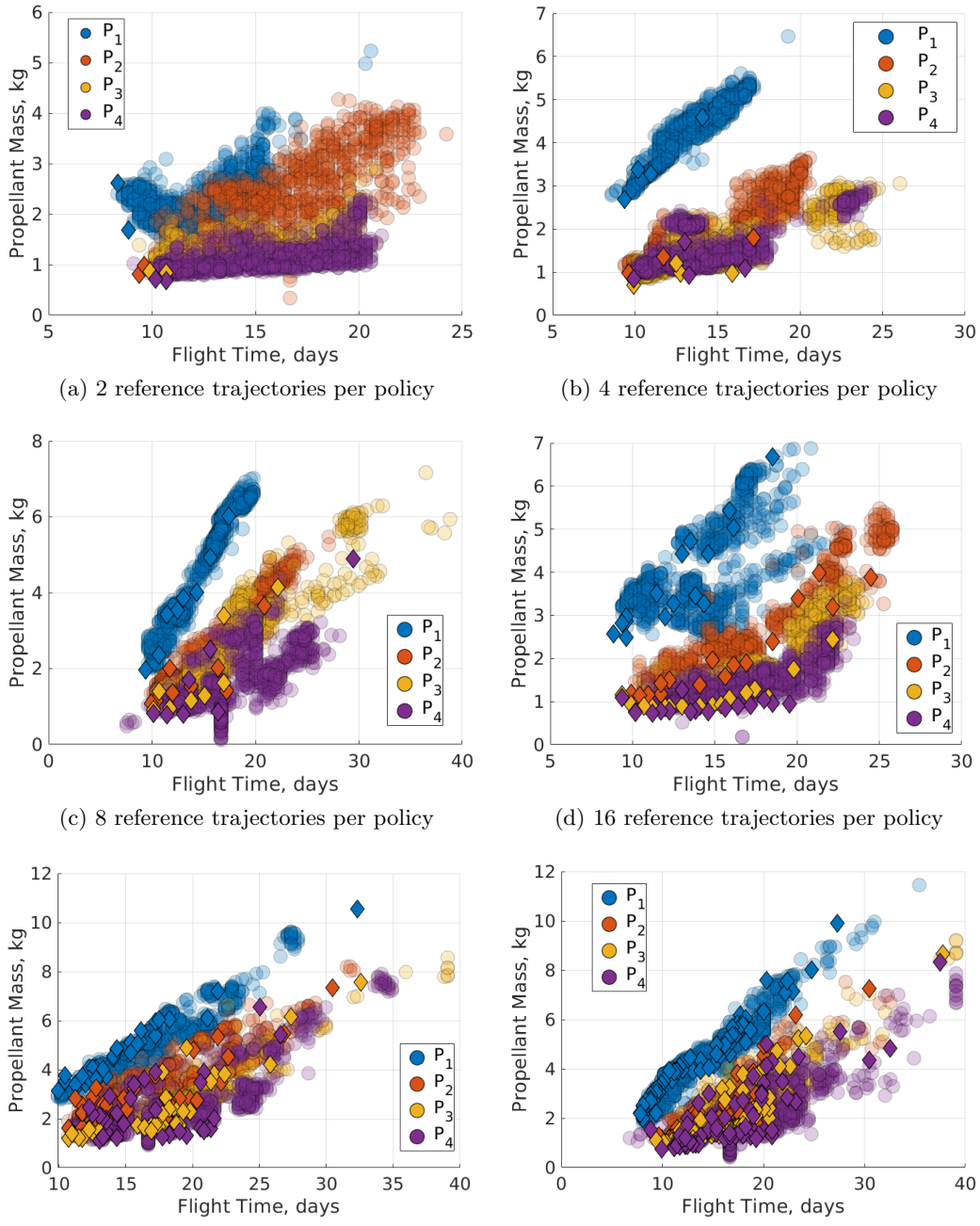
(j) Policy P_4 with 16 references (k) Policy P_4 with 32 references (l) Policy P_4 with 64 references

Figure 7.38: Reference trajectories developed by policies P_1 and P_4 from an L_1 northern halo orbit to an L_2 southern halo orbit in the Earth-Moon CR3BP (Moon not to scale).

converge on the final orbit. As the number of reference trajectories per policy increases, the number of evaluation trajectories that fail to converge on the final orbit also increases. Additionally, policy P_4 with eight reference trajectories, which recovered a reference trajectory with a revolution about the Moon, is unable to replicate that behavior in the evaluation trajectories as many of the evaluation trajectories following that reference trajectory diverge. This divergence may be due to the sensitive dynamics associated with the lunar flyby. Further, the flight time and propellant mass usage characteristics of the reference trajectories tend to be captured within the characteristics of the evaluation trajectories. Then, for this trajectory design scenario, there are limited advantages for assigning more than four reference trajectories per policy since the simulations tend to return a lower total reward, possess more evaluation trajectories that fail to converge to the final orbit, and tend to recover similar propellant mass usage and flight time solution spaces. However, a distinct set of hyperparameters or new methods for updating the reference trajectories may address the convergence issues for including a high number of reference trajectories per policy in this trajectory design scenario.

7.3 Low-Thrust Transfers from a Sun-Earth L_2 Halo Orbit to a Sun-Earth L_5 Short Period Orbit

Eight policies are trained via MRPPO to construct low-thrust transfers to the vicinity of Sun-Earth L_5 and recover a subset of solutions in the multi-objective solution spanning flight time and propellant mass usage. Two dynamical models are employed, the Sun-Earth CR3BP and the Sun-Earth-Moon point mass ephemeris model. Additionally, two sets of policies are trained to develop low-thrust transfers; one set constructing transfers using a variable thrust engine and a second, distinct set of policies designing transfers using a constant thrust engine. These two engine configurations are reflective of distinct technological capabilities for low-thrust-enabled SmallSats. For both engine models, the policies are first trained to recover locally optimal solutions in the Sun-Earth CR3BP prior to incorporating a transfer learning approach to update the policies to recover locally optimal solutions in the Sun-Earth-Moon point mass ephemeris model. Once trained,



(a) 2 reference trajectories per policy

(b) 4 reference trajectories per policy

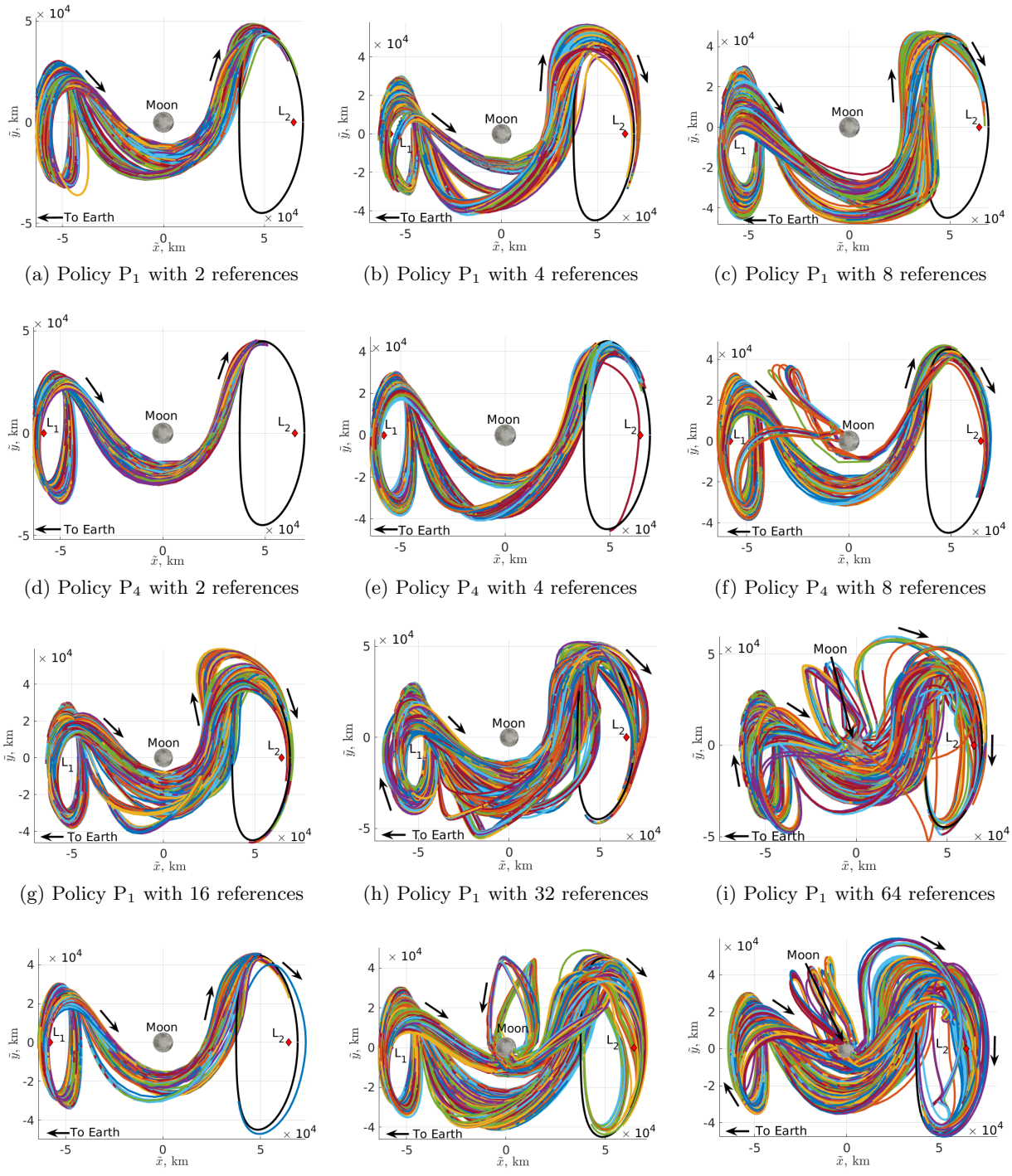
(c) 8 reference trajectories per policy

(d) 16 reference trajectories per policy

(e) 32 reference trajectories per policy

(f) 64 reference trajectories per policy

Figure 7.39: Solution spaces recovered while varying the number of reference trajectories per policy in MRPPO for transfers from an L_1 northern halo orbit to an L_2 southern halo orbit in the Earth-Moon CR3BP.



(a) Policy P_1 with 2 references (b) Policy P_1 with 4 references (c) Policy P_1 with 8 references

(d) Policy P_4 with 2 references (e) Policy P_4 with 4 references (f) Policy P_4 with 8 references

(g) Policy P_1 with 16 references (h) Policy P_1 with 32 references (i) Policy P_1 with 64 references

(j) Policy P_4 with 16 references (k) Policy P_4 with 32 references (l) Policy P_4 with 64 references

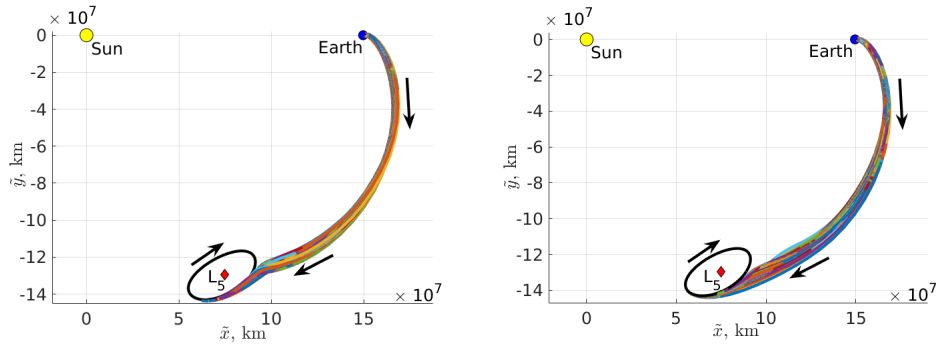
Figure 7.40: Evaluation trajectories developed by policies P_1 and P_4 from an L_1 northern halo orbit to an L_2 southern halo orbit in the Earth-Moon CR3BP (Moon not to scale).

the policies for each dynamical model and engine combination are evaluated on a common set of 1,000 initial conditions to facilitate direct comparisons between the policies and develop the flight time and propellant mass usage trade space. Additionally, the reference trajectories developed for variable thrust transfers in the Sun-Earth CR3BP are used to seed initial guesses for an optimization scheme to validate the results produced by MRPPO. Finally, an exploration into the influence of the number of reference trajectories per policy on the behavior of the trained policies is performed for variable thrust transfers in the Sun-Earth CR3BP.

7.3.1 Variable Thrust Transfers in the Sun-Earth CR3BP Recovered via MRPPO

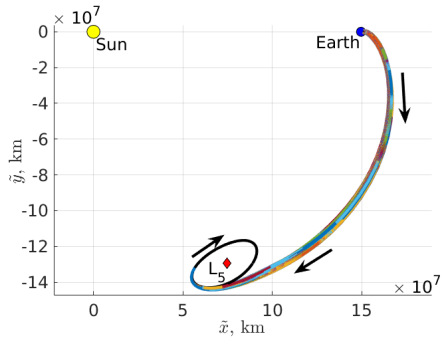
Low-thrust transfers are first constructed using a spacecraft with a variable thrust engine in the Sun-Earth CR3BP. Eight policies are trained on this scenario using MRPPO with the policies each using and updating four reference trajectories to follow depending on where along the initial periodic orbit the spacecraft is initialized. Once trained, the policies are evaluated on a common set of 1,000 initial conditions to generate low-thrust trajectories spanning a region of the propellant mass usage and flight time multi-objective solution space. Figure 7.41 displays the evaluation trajectories for policies P_1 - P_8, respectively, where the initial L_2 northern halo orbit is denoted by a black dashed arc, the final L_5 short period orbit by a solid black arc, and the evaluation trajectories are depicted in a variety of hues to aid in differentiation.

Each evaluation trajectory resembles one of the four reference trajectories developed by each policy, which are depicted in Fig. 7.42 in shades of reds and blues to differentiate the segments along the initial periodic orbit that the reference trajectories originate from. Additionally, Fig. 7.43 displays a zoomed-in view of the reference trajectories developed by policies P_1 and P_8 illustrating the segments of the periodic orbit that each shade of red and blue corresponds to. Moving from policy P_1 to P_2 and P_2 to P_3, an additional periapsis with respect to the Sun is incorporated into the reference trajectories. This raises the flight time by up to a year and modifies the transfer geometry in the rotating frame. Then, policies P_3 - P_8 possess two periapses with respect to the Sun, but distinct locations for those periapses. For instance, policies P_3 - P_6 construct reference

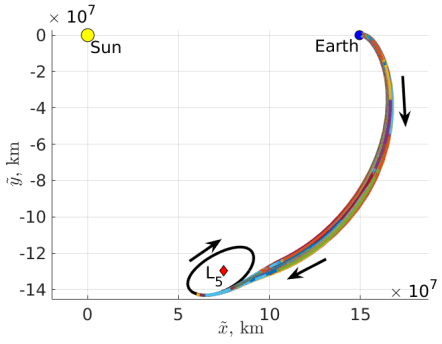


(a) Policy P_1

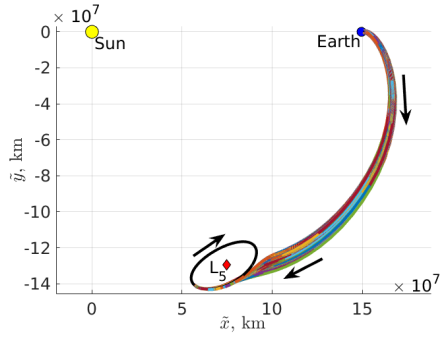
(b) Policy P_2



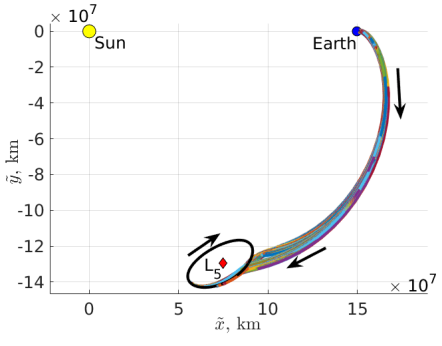
(c) Policy P_3



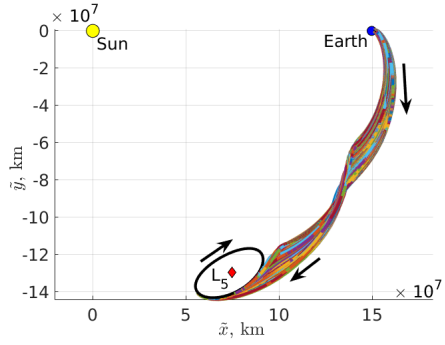
(d) Policy P_4



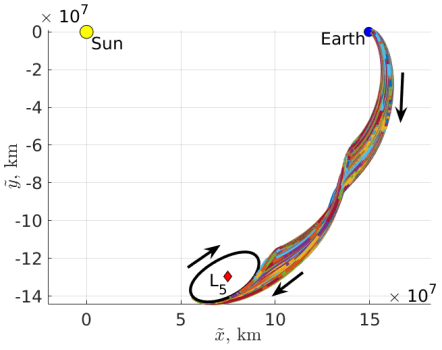
(e) Policy P_5



(f) Policy P_6



(g) Policy P_7



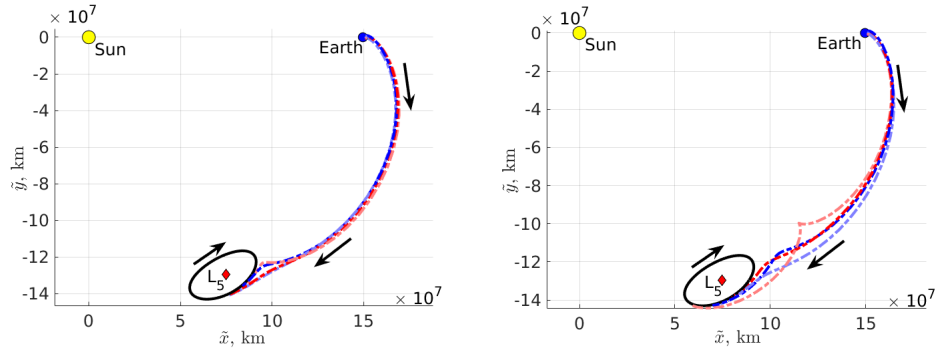
(h) Policy P_8

Figure 7.41: Evaluation trajectories developed for policies P_1 - P_8 using MRPPO for a variable thrust configuration, Sun-Earth CR3BP transfer (primaries not to scale).

trajectories with loops where the velocity in the rotating frame changes directions, but the reference trajectories generated by policies P_7 and P_8 do not possess the same loops. Further, policies P_2 and P_6 both develop the reference trajectories shaded in light red that possess significantly distinct geometries from the other reference trajectories developed by these policies.

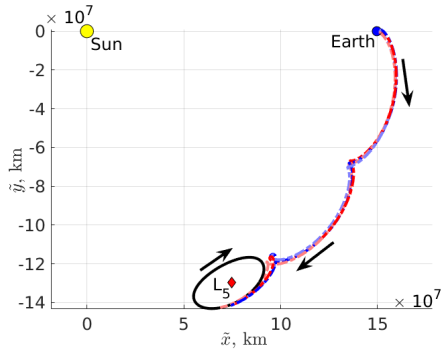
The thrust magnitude profiles and Jacobi constants throughout the reference trajectories are also examined to generate further insights into the recovered solution space. Figure 7.44 displays the thrust magnitudes of the reference trajectories developed by policies P_1 - P_8. The policies construct reference trajectories with thrust magnitude profiles that gradually decrease across policies as the penalty on propellant mass usage increases. In fact, policy P_1 produces reference trajectories that require near maximum thrust output from the low-thrust engine while policy P_8 generates reference trajectories that possess a comparatively lower thrust for a majority of the transfer until the spacecraft approaches the final orbit; such solutions may be used to recover trajectories with coast arcs, which may be valuable for opportunistic science observations [48]. Further, policies P_3-P_8 recover bang-bang-like control profiles, which are known to be propellant-optimal demonstrating how the policies are autonomously generating optimal behavior without a priori domain knowledge. Specifically, as the penalty on propellant mass usage increases, the trajectories approach a natural transfer arc in the CR3BP resembling an invariant manifold trajectory, which may be used to depart or arrive into a target orbit.

This trend is also reflected in Fig. 7.45 which depicts the Jacobi constants throughout the reference trajectories developed by policies P_1 - P_8. Policies P_1 and P_2 constructs reference trajectories with Jacobi constants that significantly oscillate about the Jacobi constants of the two periodic orbits. Conversely, policies P_3 - P_8 develop references trajectories with Jacobi constants that remain near $C_J = 3.0008$, the Jacobi constants of the initial periodic orbit, until the transfer approaches the final periodic orbit. Then, the Jacobi constants are decreased to match the Jacobi constant of the final periodic orbit. This result suggests that an arc from the unstable invariant manifold departing the Sun-Earth L_2 northern halo orbit may serve as a good initial guess for a transfer to a Sun-Earth L_5 short period orbit [48]. Additionally, this agrees with the conclusions

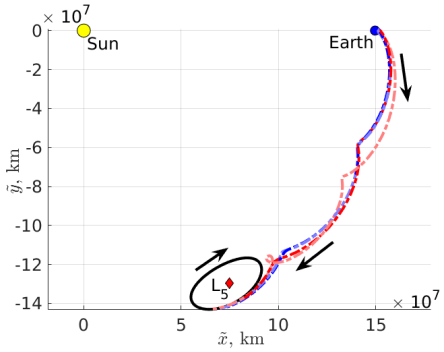


(a) Policy P_1

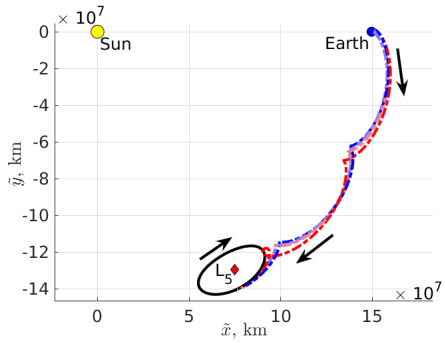
(b) Policy P_2



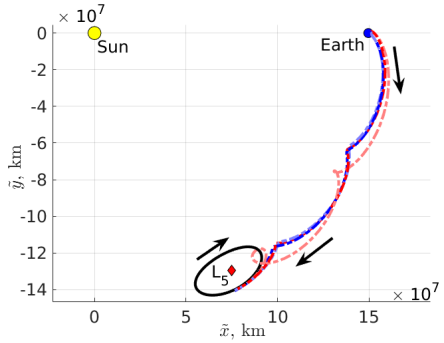
(c) Policy P_3



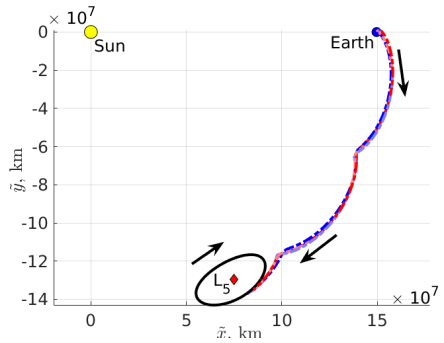
(d) Policy P_4



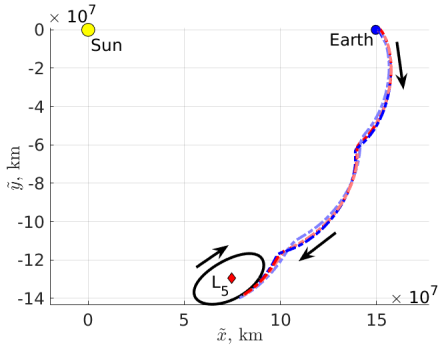
(e) Policy P_5



(f) Policy P_6

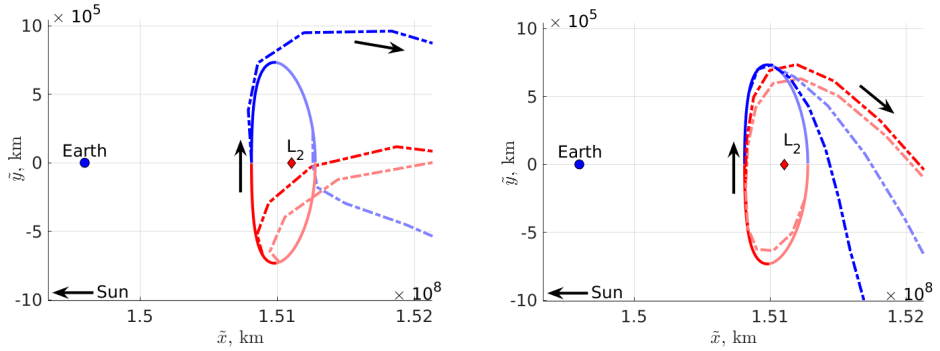


(g) Policy P_7



(h) Policy P_8

Figure 7.42: Reference trajectories developed for policies P_1 - P_8 using MRPPO for a variable thrust configuration, Sun-Earth CR3BP transfer (primaries not to scale).

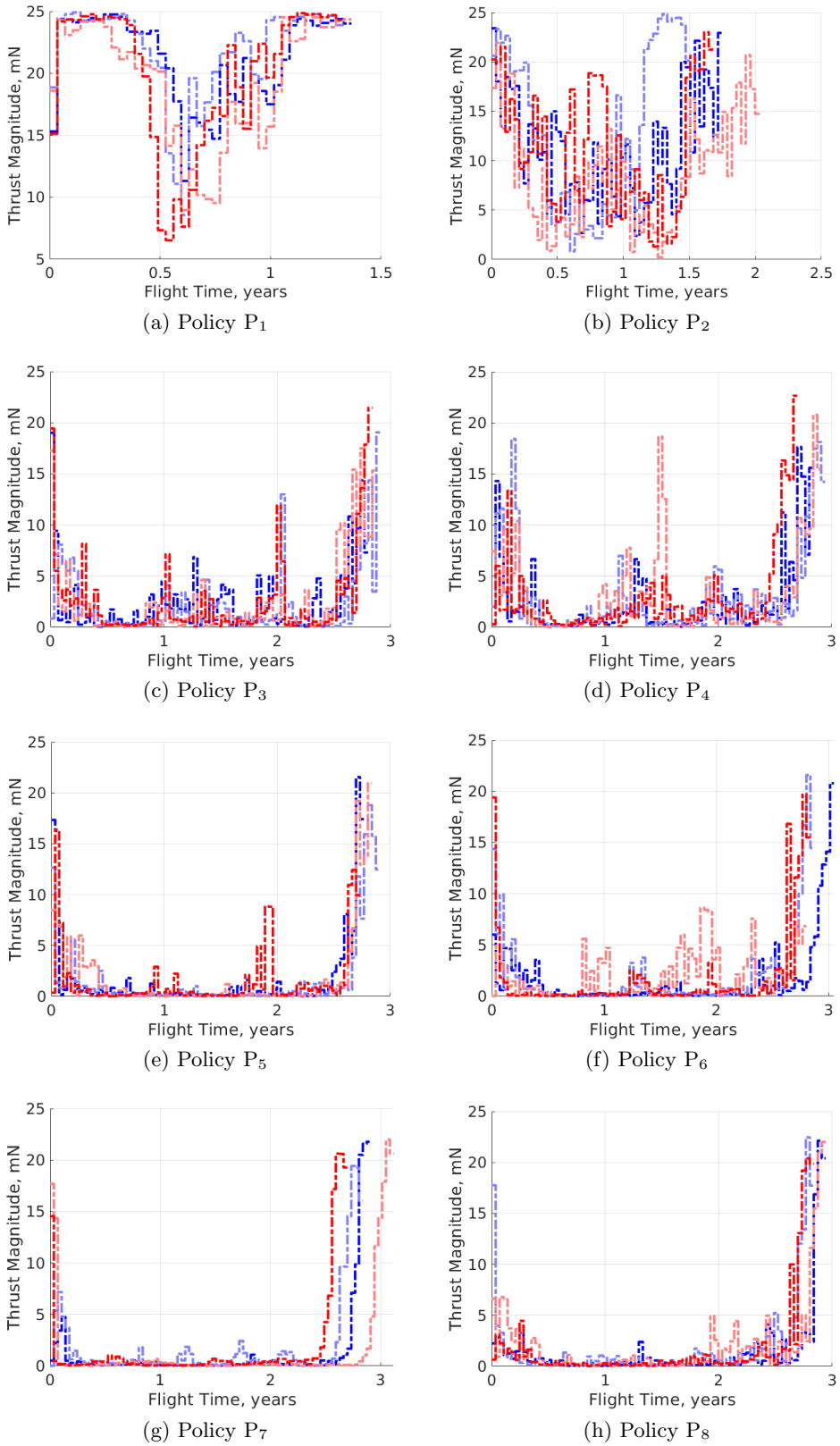


(a) Policy P_1 (b) Policy P_8

Figure 7.43: Zoomed-in view of the reference trajectories developed for policies P_1 and P_8 using MRPPO for a variable thrust configuration, Sun-Earth CR3BP transfer (Earth not to scale).

reached by Anderson and Lo, that low-thrust transfers in multi-body systems tend to resemble invariant manifold arcs [5]. However, due to the non-zero position and velocity hyperspheres used to define convergence to the final orbit, the Jacobi constants at the conclusion of the trajectories do not reach the Jacobi constant of the final orbit. This may be attributed to the size of the hyperspheres specified to define convergence at the final periodic orbit. In this proof of concept approach, the hyperspheres are defined with a large size to enable the policies to recover solutions that consistently target the final orbit. However, if the size of the hyperspheres is decreased, the policies tend to struggle to converge on the final orbit. While the policies develop transfers that converge on the final orbit within the specified hyperspheres, they are not incentivized to converge to a defined Jacobi constant near that of the final periodic orbit.

Finally, Figs. 7.46a and 7.46b depict the solution spaces associated with the evaluation trajectories and reference trajectories, respectively. Low-thrust transfers are recovered with flight times as low as 1.33 years while other transfers require only 4.11 kg of propellant mass. For this trajectory design scenario, policy P_1 produces transfers with lower flight times than policy P_8 at the expense of increased propellant mass usage while policies P_2 - P_7 possess intermediate values between the two extrema, consistent with the assigned values of c_m. Further, the reference trajectories developed by policies P_3 - P_8 possess similar transfer geometries, propellant mass usages,



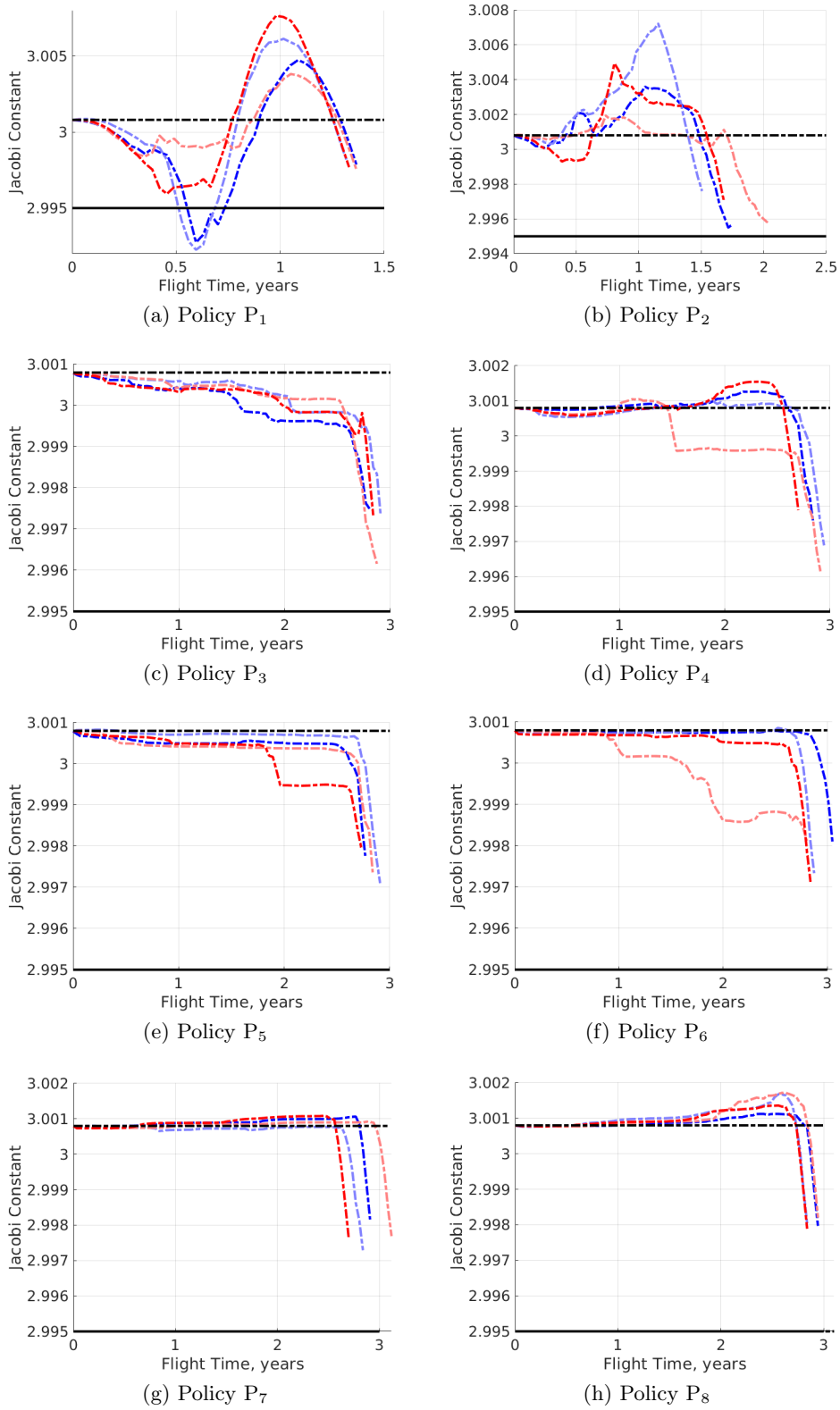
(a) Policy P_1 (b) Policy P_2

(c) Policy P_3 (d) Policy P_4

(e) Policy P_5 (f) Policy P_6

(g) Policy P_7 (h) Policy P_8

Figure 7.44: Thrust magnitude profiles of the reference trajectories developed for policies P_1 - P_8 using MRPPO for a variable thrust configuration, Sun-Earth CR3BP transfer.



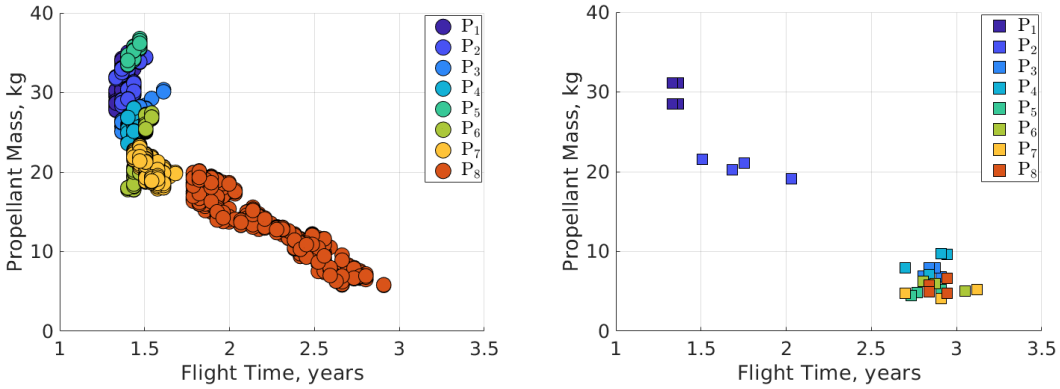
(a) Policy P_1 (b) Policy P_2

(c) Policy P_3 (d) Policy P_4

(e) Policy P_5 (f) Policy P_6

(g) Policy P_7 (h) Policy P_8

Figure 7.45: Jacobi constants of the reference trajectories developed for policies P_1 - P_8 using MRPPO for a variable thrust configuration, Sun-Earth CR3BP transfer.



(a) Evaluation solution space

(b) Reference solution space

Figure 7.46: Evaluation and reference solution spaces developed using MRPPO for a variable thrust configuration, Sun-Earth CR3BP transfer.

and flight times, which lie near the maximum final mass solution. This may be attributed to the update equation in Eq. 5.1 used to determine when to update the reference trajectories primarily encouraging propellant mass usage. These variable thrust, low-thrust transfers are produced in the Sun-Earth CR3BP dynamical model which are valuable approximations for low-thrust transfers in higher-fidelity models.

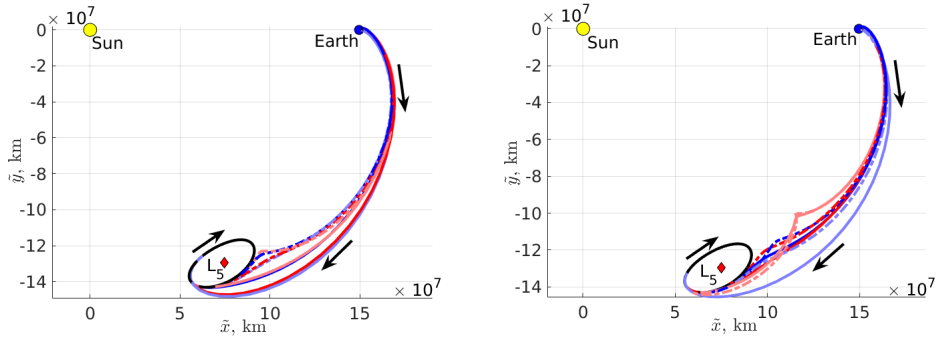
7.3.2 Variable Thrust Transfers in the Sun-Earth CR3BP Recovered Using Classical Optimization

The variable thrust transfers from an L_2 northern halo orbit to an L_5 short period orbit in the Sun-Earth CR3BP produced by MRPPO are compared to similar transfers generated by SQP in *fmincon*. The initial guesses are seeded using the reference trajectories developed by each policy and optimized with respect to two objective functions: 1) maximizing the total reward and 2) maximizing the final mass. Using both objective functions validates the performance of MRPPO in maximizing the total reward over a single trajectory, which is the goal of each policy, and how effective MRPPO is in uncovering solutions that lie along the front of the propellant mass usage and flight time trade space. However, as noted, differences in implementation prevent a completely

equivalent comparison between the two methodologies. For both objective functions, the solutions developed by SQP are designated O_j, corresponding to policies P_j for $j = [1, \ldots, 8]$ with increasing penalties on propellant mass usage. Additionally, the four reference trajectories constructed by each policy are used to see initial guesses for SQP producing thirty two optimal solutions for each objective function.

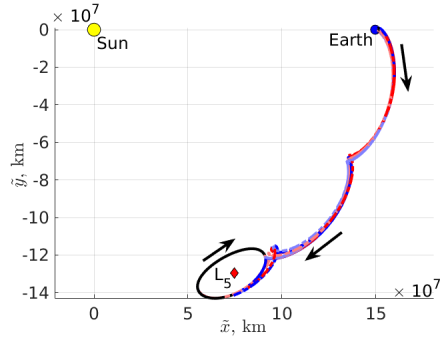
The first objective function more closely aligns with the reward function formulation used by MRPPO in this trajectory design scenario. Figure 7.47 depicts the initial guess and optimal solutions developed by SQP associated with policies P_1 - P_8 for variable thrust transfers from an L_2 northern halo orbit to an L_5 short period orbit in the Sun-Earth CR3BP. In this figure, the initial guesses are designated using dashed arcs, the optimal transfers are represented with solid arcs shaded according to the reference trajectory used to see the initial guess, the initial periodic orbit is illustrated using a black dashed arc, and the final periodic orbit is depicted using the solid black arc. A majority of the solutions developed by SQP closely resemble the initial guesses, however, the optimal solutions generated for P_1 possess noticeable deviation from the initial guesses near the Sun-Earth L_5 short period orbit. Additionally, a single locally optimal solution for both policies P_2 and P_5 differs significantly from the initial guess. However, the geometries of the initial guesses produced by MRPPO are generally retained in the locally optimal solutions developed by SQP.

In addition to the transfer geometries, the thrust magnitude and thrust direction profiles are explored for the optimal solutions that maximize the total reward. Figure 7.48 depicts the thrust magnitude profiles for the optimal solutions generated using each reference trajectory for each policy. Similar to the previous trajectory design scenarios, the thrust magnitude profiles are updated during optimization to resemble a bang-bang-like control scheme. This is most pronounced for the solutions associated with P_1 which previously had continuous control profiles. However, this is less evident for the latter policies, which developed reference trajectories that already possessed bang-bang-like control profiles. Additionally, Fig. 7.49 displays the thrust direction profiles for the reference trajectories and solutions generated using SQP associated with policies P_1 and P_8. In each case, the thrust direction profiles are only minimally updated by SQP demonstrating how the

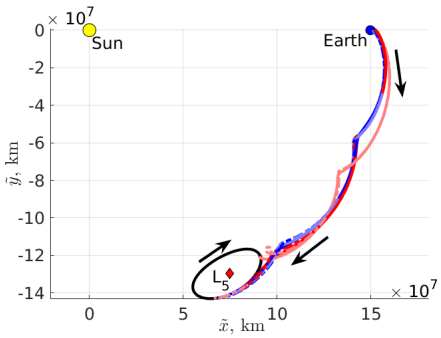


(a) Policy P_1 transfers

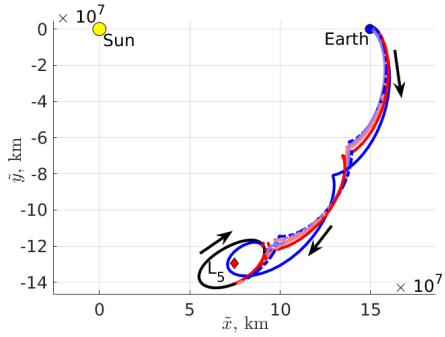
(b) Policy P_2 transfers



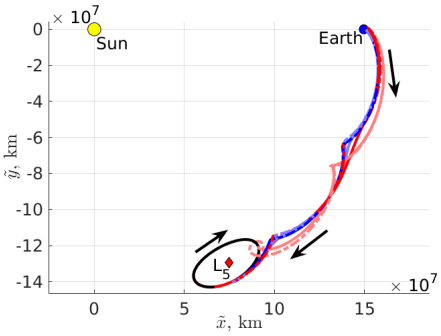
(c) Policy P_3 transfers



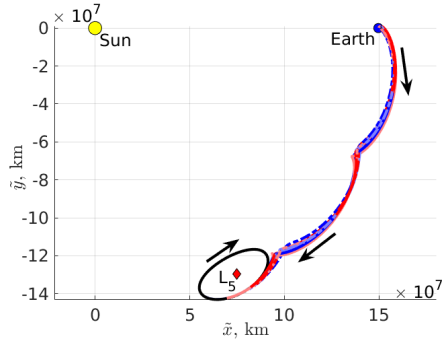
(d) Policy P_4 transfers



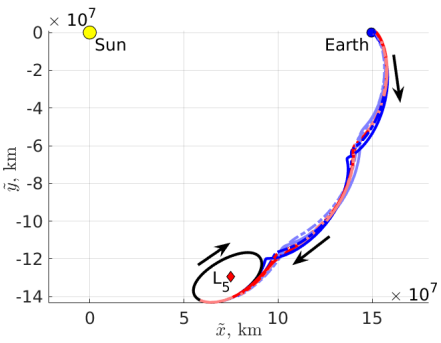
(e) Policy P_5 transfers



(f) Policy P_6 transfers



(g) Policy P_7 transfers



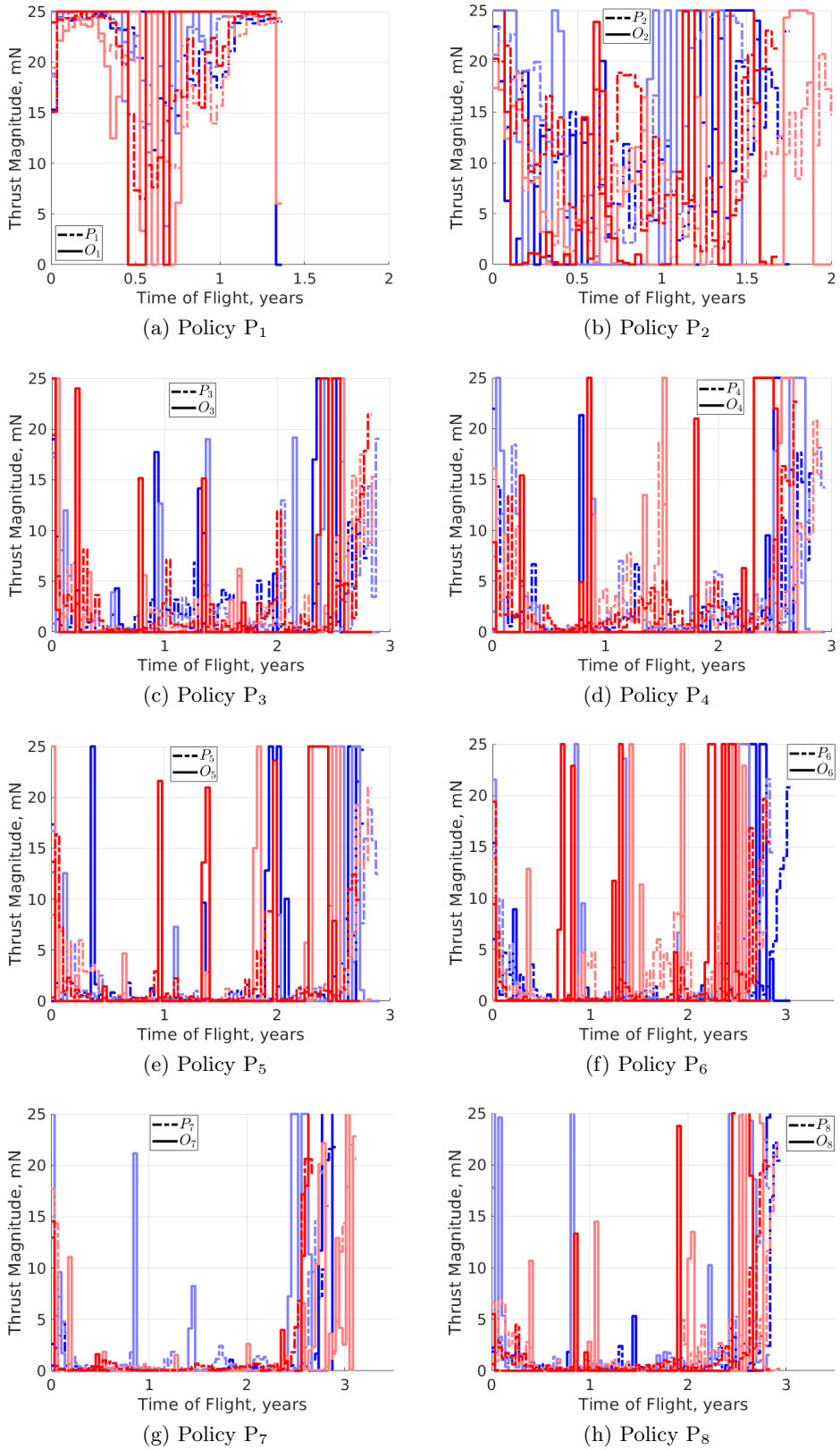
(h) Policy P_8 transfers

Figure 7.47: Variable thrust transfers produced by SQP maximizing the total reward associated with policies P_1 - P_8 from an L_2 northern halo orbit to an L_5 short period orbit in the Sun-Earth CR3BP (primaries not to scale).

thrust direction profiles generated by MRPPO lie close to the locally optimal solutions found by SQP in *fmincon*.

The multi-objective solution spaces are also examined to generate insights into the characteristics of the optimal solutions that maximize the total reward. Figure 7.50a depicts the propellant mass usage and flight time characteristics of the initial guesses and optimal solutions produced by SQP. In this figure, SQP, with the exception of transfers associated with P_2, develops solutions with similar propellant mass usages to MRPPO while maximizing the total reward. Specifically, the transfers constructed by SQP associated with policy P_1, which has no incentive to decrease propellant mass usage, and policies P_3 - P_8, which already lie close to the maximum final mass solution, recover similar propellant mass usages to MRPPO. Additionally, Fig. 7.50b illustrates the flight time and total reward solution space for the transfers developed by SQP that maximize the total reward. While some differences in the transfer geometry are apparent for the initial guesses and optimal solutions associated with P_1, P_2, and P_5, the transfers possess similar total rewards. These figures demonstrate that both SQP and MRPPO are recovering similar solutions that maximize the total reward validating the results produced by MRPPO.

The initial guesses developed by MRPPO for variable thrust transfers in the Sun-Earth CR3BP are also optimized to maximize the final mass. Figure 7.51 displays the initial guesses developed by MRPPO and the locally optimal solutions designed by SQP that maximize the final mass for policies P_1 - P_8 for variable thrust transfers from an L_2 northern halo orbit to an L_5 short period orbit in the Sun-Earth CR3BP using the same color configuration as Fig. 7.47. Compared to the first objective function, the optimal solutions developed by SQP possess a wider variety of transfer geometries that do not resemble the initial guesses. This may be attributed to the objective function, which no longer incentivizes following the reference. However, a significant number of solutions still retain the geometry of the initial guesses demonstrating how MRPPO may be recovering transfers near propellant-optimal solutions. Specifically, the transfers associated with policies P_3 and P_8 retain similar transfer geometries for all four optimal solutions while policies P_4, P_5, P_6, and P_7 retain similar transfer geometries for three of the four optimal solutions.



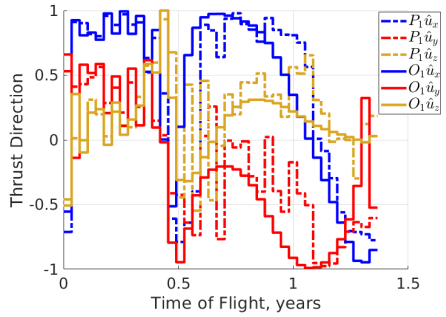
(a) Policy P_1 (b) Policy P_2

(c) Policy P_3 (d) Policy P_4

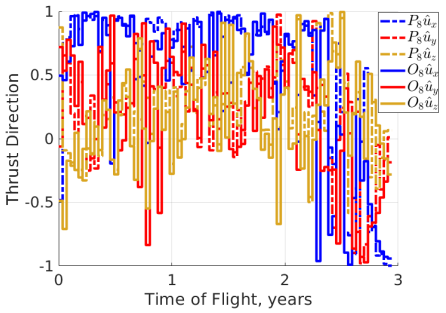
(e) Policy P_5 (f) Policy P_6

(g) Policy P_7 (h) Policy P_8

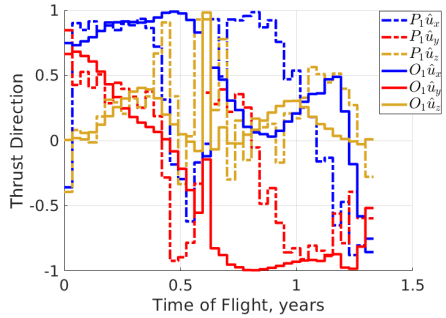
Figure 7.48: Thrust magnitude profiles for low-thrust transfers maximizing the total reward from an L_2 northern halo orbit to an L_5 short period orbit in the Sun-Earth CR3BP.



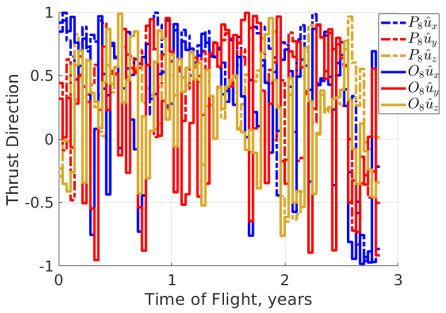
(a) Policy P_1 blue reference



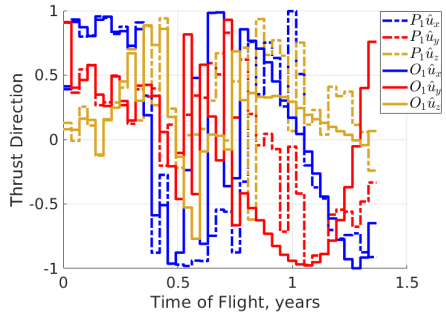
(b) Policy P_8 blue reference



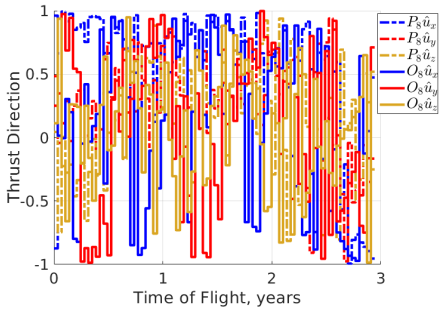
(c) Policy P_1 light blue reference



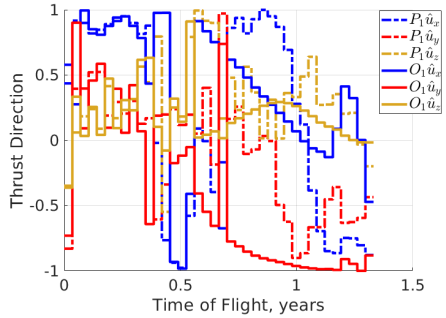
(d) Policy P_8 light blue reference



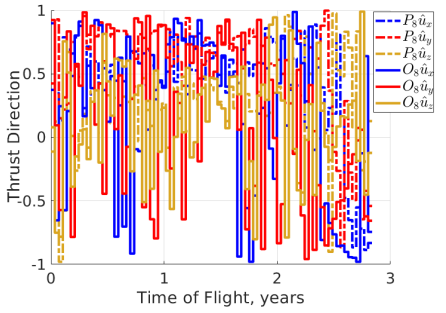
(e) Policy P_1 light red reference



(f) Policy P_8 light red reference

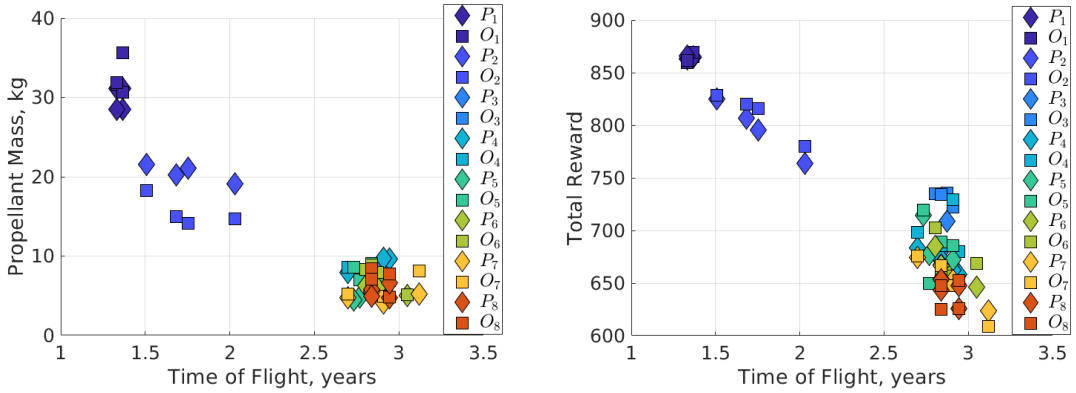


(g) Policy P_1 red reference



(h) Policy P_8 red reference

Figure 7.49: Thrust direction profiles of the reference trajectories and solutions developed by SQP maximizing the total reward associated with policies P_1 and P_8 using MRPPO for a variable thrust configuration, Sun-Earth CR3BP.



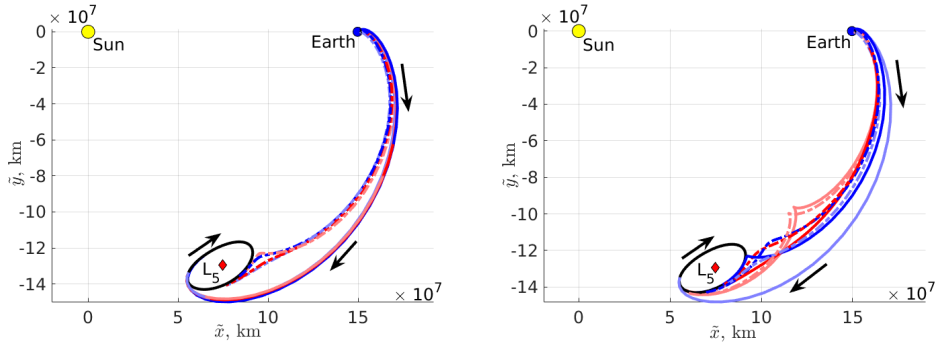
(a) Flight time and propellant mass usage

(b) Flight time and total reward

Figure 7.50: Solution spaces recovered via optimization while maximizing the total reward from initial guesses developed using MRPPO for a variable thrust configuration, Sun-Earth CR3BP transfer from an L_2 northern halo orbit to an L_5 short period orbit.

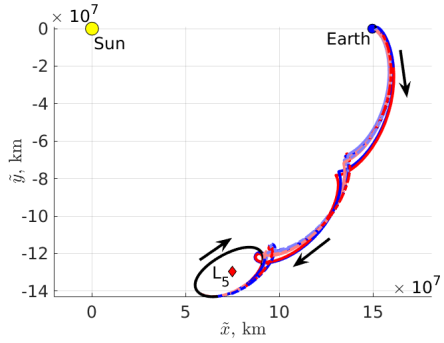
The thrust magnitude and thrust direction profiles of the solutions developed by SQP are also examined to generate insights into the solutions that maximize the final mass. Figure 7.52 displays the thrust magnitude profiles for the initial guesses and optimal solutions associated with each policy. For each transfer, the thrust-magnitude profiles are updated to a bang-bang-like control profile, which is known to correspond to propellant-optimal solutions. Additionally, each transfer tends to have a thrust arc at the beginning to depart the initial periodic orbit, a coast segment during the transit leg, and a thrust arc once the transfer approaches the vicinity of the final periodic orbit. This trend also supports the use of invariant manifold arcs for constructing an initial guess for a transfer to the Sun-Earth L_5 short period orbit. However, MRPPO autonomously recovered this behavior, which was replicated within the optimization scheme. Further, Fig. 7.53 depicts the thrust direction profiles for the initial guesses and optimal solutions associated with policies P_1 and P_8. Similarly to the first objective function, the thrust direction profiles are minimally updated for each reference trajectory further demonstrating that MRPPO recovers near-propellant-optimal solutions in this trajectory design scenario.

Finally, the propellant mass usage and flight time trade space is explored for the solutions developed by SQP that maximize the final mass. Figure 7.54 displays the characteristics of the

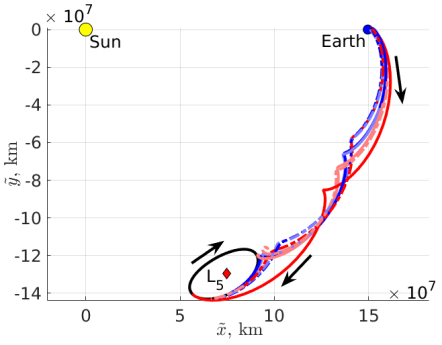


(a) Policy P_1 transfers

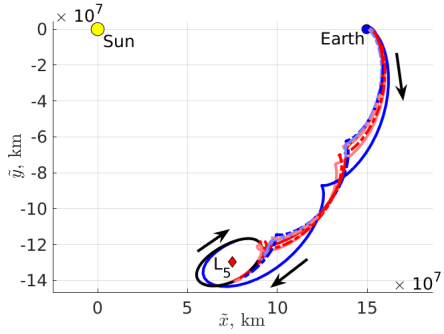
(b) Policy P_2 transfers



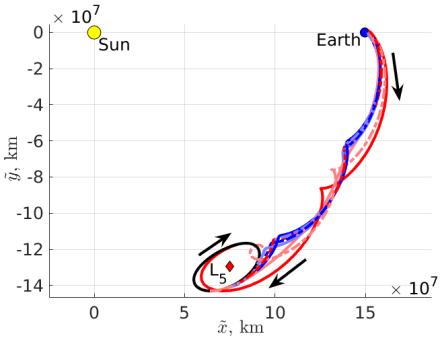
(c) Policy P_3 transfers



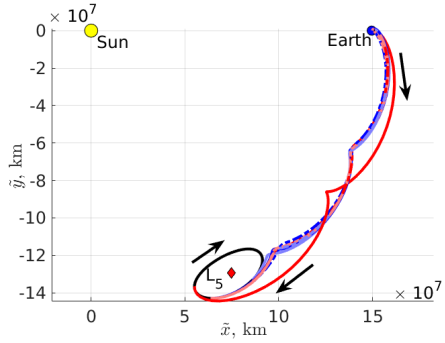
(d) Policy P_4 transfers



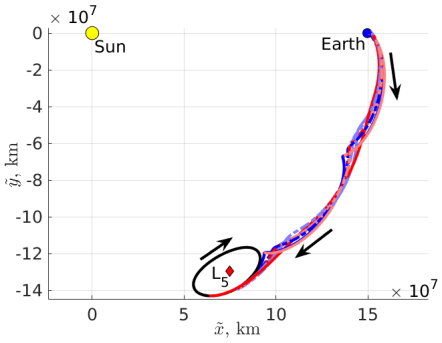
(e) Policy P_5 transfers



(f) Policy P_6 transfers

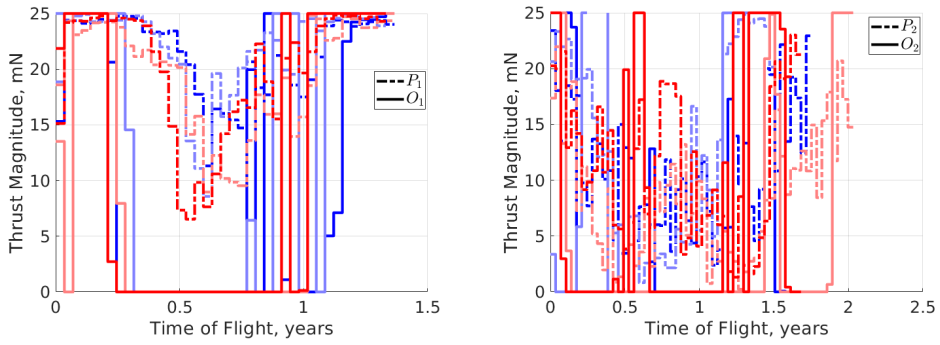


(g) Policy P_7 transfers



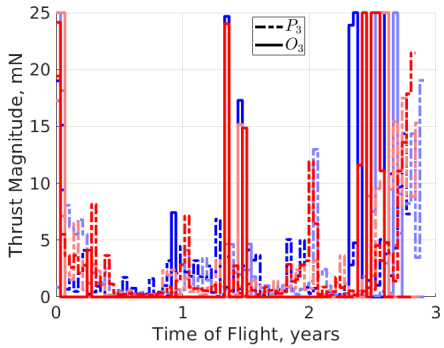
(h) Policy P_8 transfers

Figure 7.51: Variable thrust transfers produced by SQP maximizing the final mass associated with policies P_1 - P_8 from an L_2 northern halo orbit to an L_5 short period orbit in the Sun-Earth CR3BP (primaries not to scale).

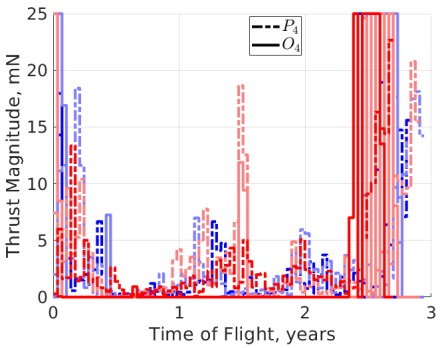


(a) Policy P_1

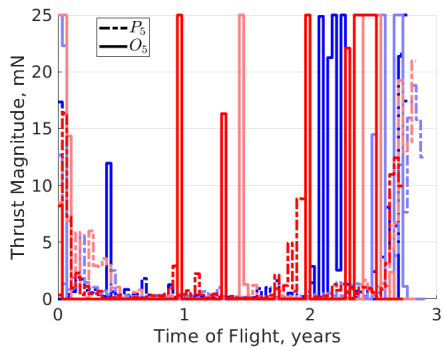
(b) Policy P_2



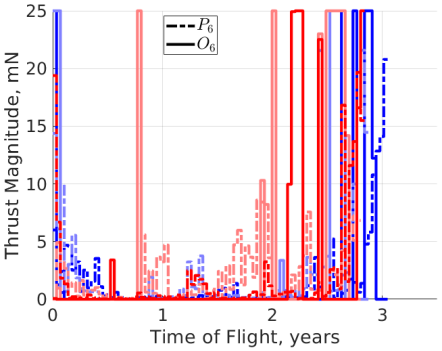
(c) Policy P_3



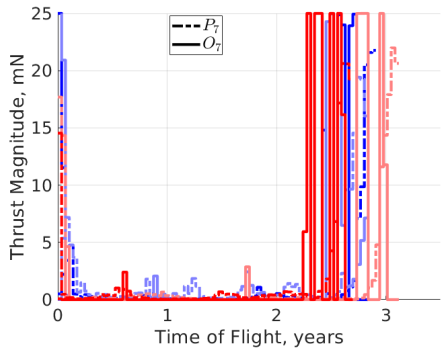
(d) Policy P_4



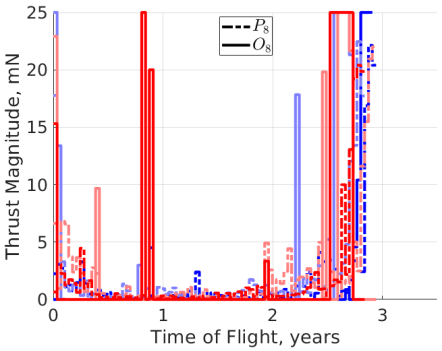
(e) Policy P_5



(f) Policy P_6

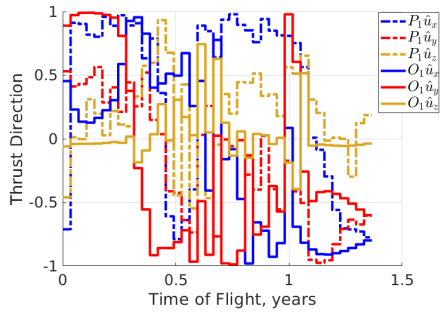


(g) Policy P_7

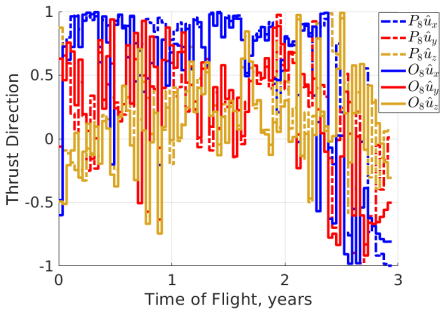


(h) Policy P_8

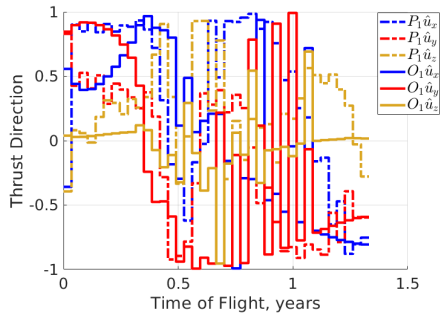
Figure 7.52: Thrust magnitude profiles for low-thrust transfers maximizing the final mass from an L_2 northern halo orbit to an L_5 short period orbit in the Sun-Earth CR3BP.



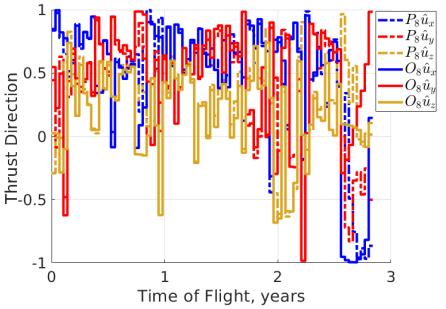
(a) Policy P_1 blue reference



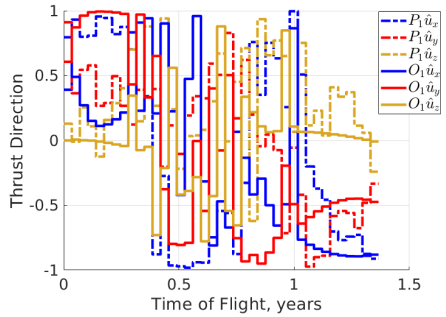
(b) Policy P_8 blue reference



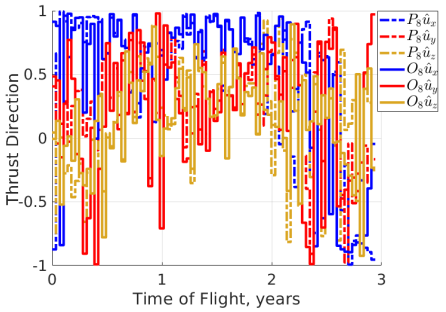
(c) Policy P_1 light blue reference



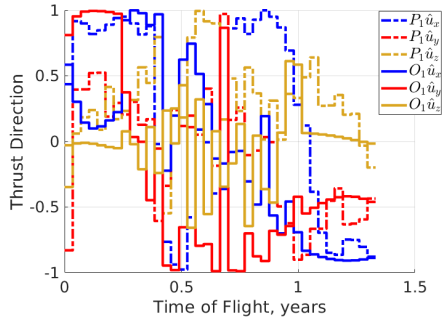
(d) Policy P_8 light blue reference



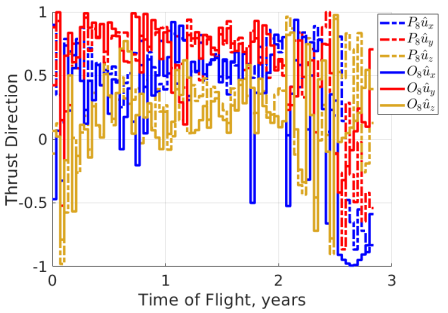
(e) Policy P_1 light red reference



(f) Policy P_8 light red reference



(g) Policy P_1 red reference



(h) Policy P_8 red reference

Figure 7.53: Thrust direction profiles of the reference trajectories and solutions developed by SQP maximizing the final mass associated with policies P_1 and P_8 using MRPPO for a variable thrust configuration, Sun-Earth CR3BP.

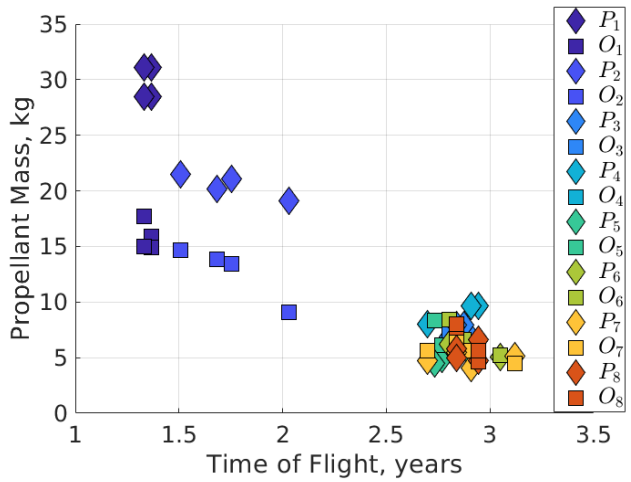


Figure 7.54: Propellant mass usage and flight time solution space recovered via optimization while maximizing the total reward from initial guesses developed using MRPPO for a variable thrust configuration, Sun-Earth CR3BP transfer from an L_2 northern halo orbit to an L_5 short period orbit.

solutions developed by SQP and MRPPO for variable thrust transfers from an L_2 northern halo orbit to an L_5 short period orbit in the Sun-Earth CR3BP. Generally, SQP recovers solutions with lower propellant mass usages than MRPPO. However, as the flight time and penalty on propellant mass usage increases, the MRPPO and SQP results tend to converge near the maximum final mass solution. Specifically, transfers associated with policies P_3 - P_8 recover similar propellant mass usages using both MRPPO and SQP. However, due to the differences in implementations and the large convergence hyperspheres used by MRPPO, the solutions produced by SQP use similar propellant mass usages, but converge within a much tighter hypersphere at the final periodic orbit. Nevertheless, this demonstrates that in this trajectory design scenario, MRPPO trains policies with higher propellant mass usage penalties that generate near-propellant-optimal transfers that are approximately validated using SQP.

7.3.3 Varying Number of Reference Trajectories per Policy

Similar to Section 7.2.3, the number of reference trajectories per policy is explored to determine the influence on the trained policies, recovered transfer geometries, and resulting multi-

objective solution space. Recall, each reference trajectory originates from an assigned segment along the initial periodic orbit and is updated throughout training to incentivize decreasing propellant mass usage and flight time. Additionally, in this trajectory design scenario, policies are incentivized to follow the reference trajectory to the final orbit via state deviation penalty terms in the reward function formulation. For transfers from an L_1 northern halo orbit to an L_2 southern halo orbit, increasing the number of reference trajectories per policy had a detrimental effect on the trained policies and recovered multi-objective solution spaces. However, this trajectory design scenario demonstrates the benefits of including more reference trajectories per policy.

First, twenty simulations are run for each number of reference trajectories where the reference trajectories are modified from one to sixty four. Then, the trained policies are evaluated on a common set of 1,000 initial conditions for low-thrust transfers from an L_2 northern halo orbit to an L_5 short period orbit. Figure 7.55 displays a box plot illustrating the median, quartiles, and outliers of the total reward for each set of twenty simulations. In this trajectory design scenario, the maximum total reward recovered is approximately 5950 regardless of the number of reference trajectories per policy. Similarly, the median total reward for each set of simulations is equal to or greater than 5500. This demonstrates that the number of reference trajectories per policy does not have a detrimental influence on maximizing the total reward in this trajectory design scenario.

The simulation with the highest total reward for each number of reference trajectories per policy is explored to generate further insights into the behavior of the policies and the recovered multi-objective solution spaces. Figure 7.56 displays the reference trajectories generated by policies P_1 and P_8 as the number of reference trajectories per policy is varied from one to sixty four. Note, the results for employing four reference trajectories per policy are depicted in Section 7.3.1. These figures demonstrate that the reference trajectories for both policies P_1 and P_8 are all able to converge on the final periodic orbit and recover similar transfer geometries regardless of the number of reference trajectories per policy. However, for 16, 32, and 64 reference trajectories per policy, a wider variety of transfer geometries are uncovered evidencing one benefit of increasing the number of reference trajectories per policy. Further, Fig. 7.57 depicts the evaluation trajectories produced

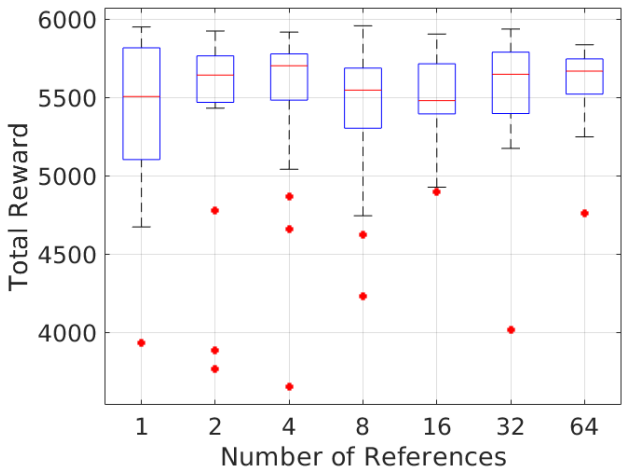
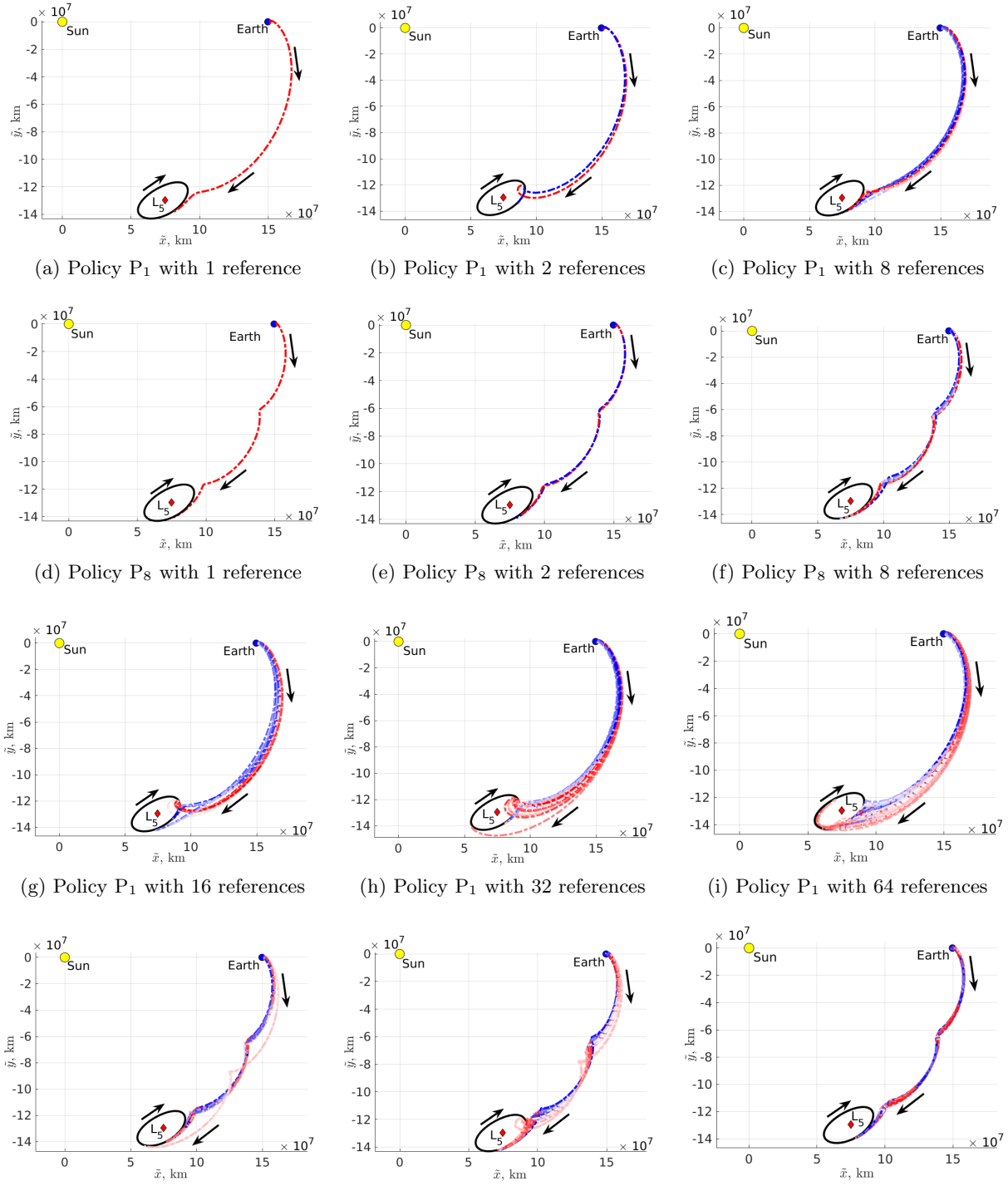


Figure 7.55: Box plot of total reward across twenty simulations while varying the number of reference trajectories per policy.

by policies P_1 and P_8 demonstrating that the policies are able to consistently follow the reference trajectory and target the final periodic orbit.

Finally, the propellant mass usage and flight time of the reference trajectories are analyzed to investigate the influence of increasing the number of reference trajectories per policy on the resulting multi-objective solution space. Recall, the objective function used to determine when to update the reference trajectories defined in Eq.5.1 balances minimizing both propellant mass usage and flight time for policies P_2-P_8 while for P_1, the function prioritizes decreasing propellant mass usage. Then, Fig. 7.58 exhibits the propellant mass usage and flight time solution space for the reference trajectories developed by the simulations with the highest total reward for each number of reference trajectories per policy. As the number of reference trajectory per policy increases, the solution space becomes more filled. Whereas the solution space associated with one reference trajectory per policy is sparse and does not provide a trajectory designer with extensive information, the solution space associated with sixty four reference trajectories per policy more clearly depicts the propellant mass usage and flight time trade space and helps uncover additional regions of the solution space. Then, while in some trajectory design scenarios, increasing the number of reference trajectories per policy has a detrimental influence on the trained policies, for low-thrust transfers



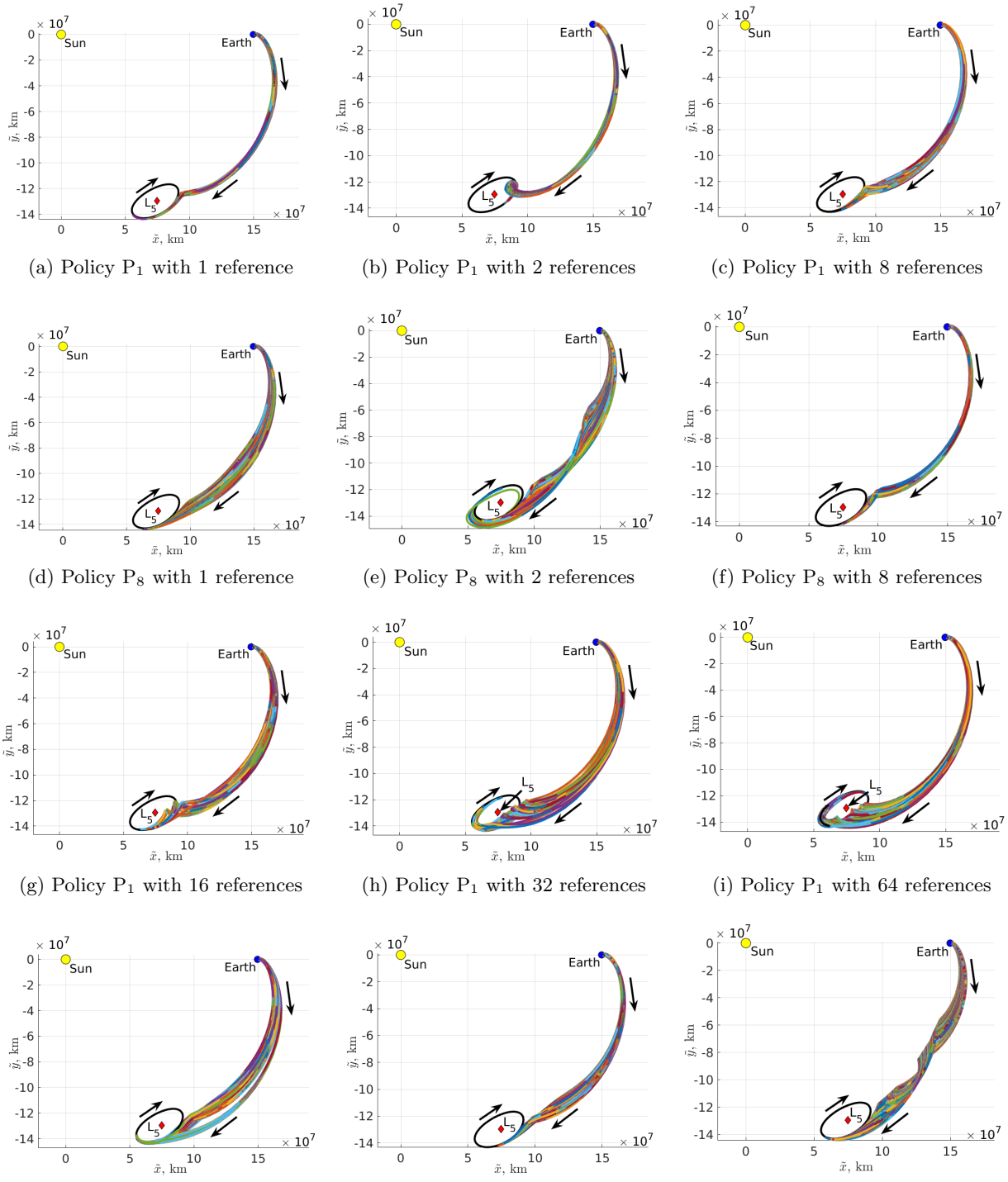
(a) Policy P_1 with 1 reference (b) Policy P_1 with 2 references (c) Policy P_1 with 8 references

(d) Policy P_8 with 1 reference (e) Policy P_8 with 2 references (f) Policy P_8 with 8 references

(g) Policy P_1 with 16 references (h) Policy P_1 with 32 references (i) Policy P_1 with 64 references

(j) Policy P_8 with 16 references (k) Policy P_8 with 32 references (l) Policy P_8 with 64 references

Figure 7.56: Reference trajectories developed by policies P_1 and P_8 from an L_2 northern halo orbit to an L_5 short period orbit in the Sun-Earth CR3BP (primaries not to scale).



(a) Policy P_1 with 1 reference (b) Policy P_1 with 2 references (c) Policy P_1 with 8 references

(d) Policy P_8 with 1 reference (e) Policy P_8 with 2 references (f) Policy P_8 with 8 references

(g) Policy P_1 with 16 references (h) Policy P_1 with 32 references (i) Policy P_1 with 64 references

(j) Policy P_8 with 16 references (k) Policy P_8 with 32 references (l) Policy P_8 with 64 references

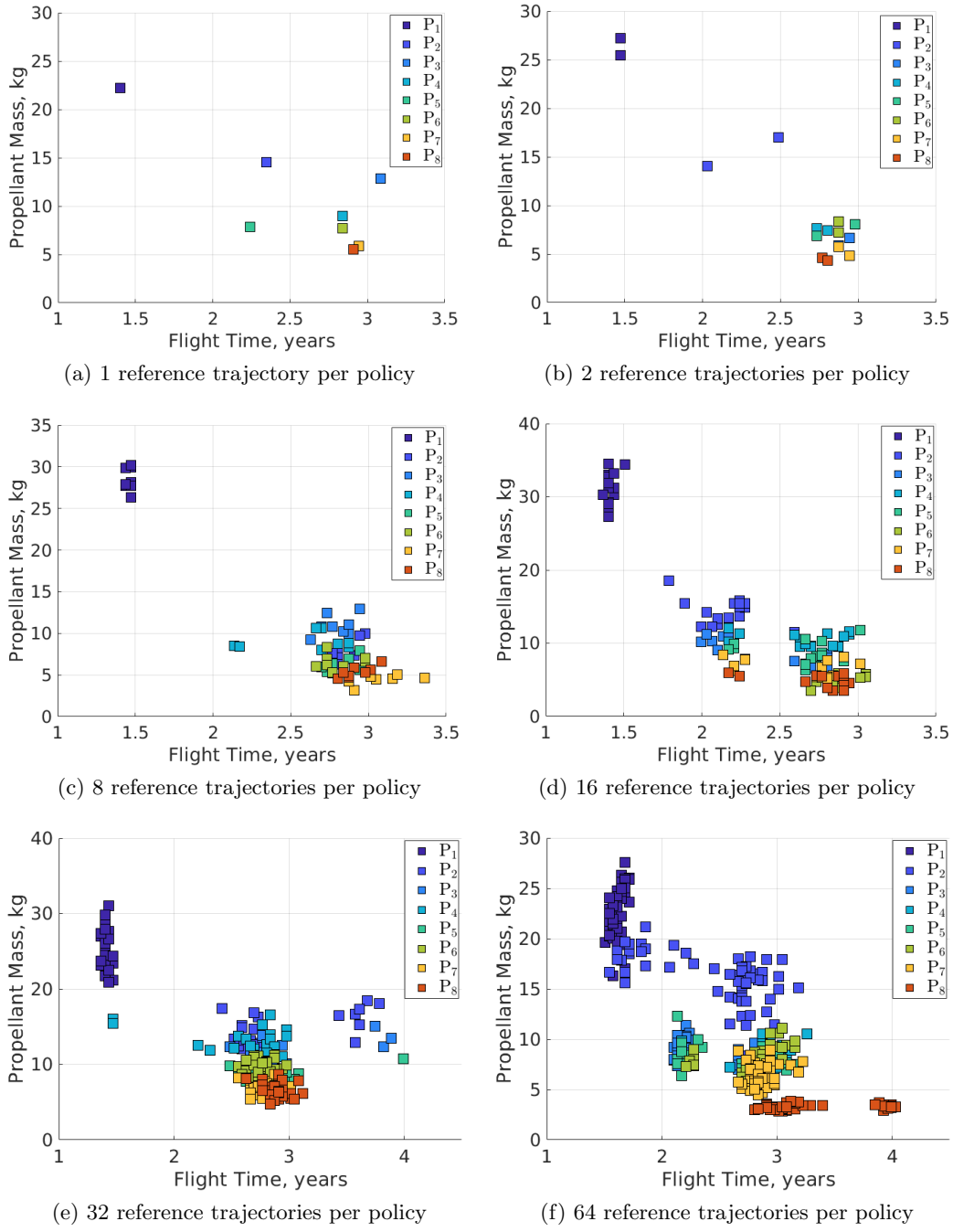
Figure 7.57: Evaluation trajectories developed by policies P_1 and P_8 from an L_2 northern halo orbit to an L_5 short period orbit in the Sun-Earth CR3BP (primaries not to scale).

from an L_2 northern halo orbit to an L_5 short period orbit in the CR3BP, increasing the number of reference trajectories per policy provides the trajectory designer with significantly more information about the multi-objective solution space.

7.3.4 Variable Thrust Transfers in the Sun-Earth-Moon Point Mass Ephemeris Model

Using a transfer learning approach similar to Bonasera et al., the policies trained by MRPPO for a variable thrust engine model in the Sun-Earth CR3BP initialize the policies trained in the Sun-Earth-Moon point mass ephemeris model [18]. The state, action, and reward formulations are consistent between dynamical models to minimize the updates required for the neural networks. To demonstrate the policies have converged in the Sun-Earth-Moon point mass ephemeris model, the total reward, which is the summed reward for each policy across all 1,000 evaluation trajectories, is evaluated to determine if the total reward converges to a steady state value at the end of the training process. Figure 7.59 depicts the total reward for the eight policies demonstrating improvement such that after updating the policies for the Sun-Earth-Moon point mass ephemeris model, the policies produce a total reward of 3152 from an initial total reward of 1866. This increase in total reward indicates that the policies have improved due to the additional training in the Sun-Earth-Moon point mass ephemeris model.

In addition to improving the total reward across the eight policies, the propellant mass usage and flight time characteristics of the evaluation and reference trajectories evolve in the Sun-Earth-Moon point mass ephemeris model. Figures 7.60a and 7.60b display the evaluation trajectory and reference trajectory solution spaces, respectively, where the non-dominating solutions exhibit similar propellant mass usage and flight time characteristics as the non-dominating solutions computed in the Sun-Earth CR3BP and depicted in Figs. 7.46a and 7.46b. The maximum final mass transfer requires 5.21 kg of propellant mass usage while the minimum flight time transfer recovered is 1.26 years. However, in Fig. 7.60a, only the characteristics associated with transfers that converge on the target orbit are included. These characteristics are distinct from the characteristics of the eval-



(a) 1 reference trajectory per policy

(b) 2 reference trajectories per policy

(c) 8 reference trajectories per policy

(d) 16 reference trajectories per policy

(e) 32 reference trajectories per policy

(f) 64 reference trajectories per policy

Figure 7.58: Solution spaces recovered while varying the number of reference trajectories per policy in MRPPO for transfers from an L_2 northern halo orbit to an L_5 short period orbit in the Sun-Earth CR3BP.

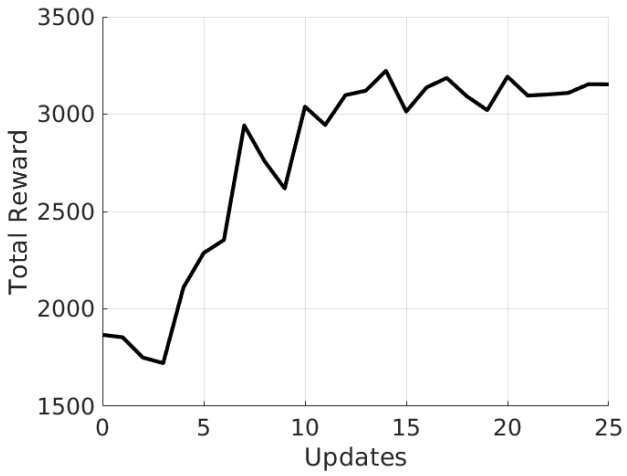
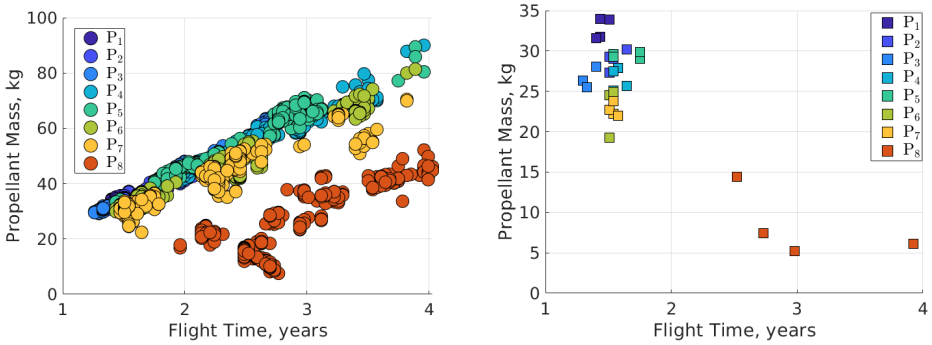


Figure 7.59: Total reward developed using transfer learning for a variable thrust, Sun-Earth-Moon point mass ephemeris model transfer.



(a) Evaluation solution space

(b) Reference solution space

Figure 7.60: Evaluation and reference solution spaces developed using transfer learning for a variable thrust configuration, Sun-Earth-Moon point mass ephemeris model transfer.

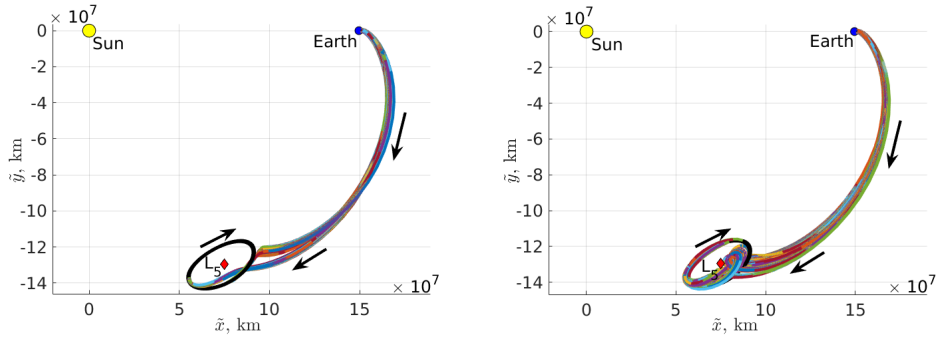
uation trajectories displayed in Fig. 7.46a for variable thrust transfers in the Sun-Earth CR3BP due to the policies inconsistently targeting the final periodic orbit in the Sun-Earth-Moon point mass ephemeris model.

Figure 7.61 depicts the evaluation trajectories for policies P_1 - P_8 using the same color configuration as Fig. 7.41. Note, only the converged solutions are plotted for each policy. Approximately 10% of transfers for policies P_3 - P_6 and 30% of transfers for policies P_2, P_7, and P_8 fail to converge on the final orbit. Further, of the transfers that do converge on the final orbit, many require addi-

tional revolutions about the Sun-Earth L_5 short period orbit significantly increasing the flight time of these solutions. While a majority of the trajectories for each policy possess similar flight times on the transfer leg, the inconsistent convergence on the final orbit leads to a large dispersion in the flight time and propellant mass usage solution space for the evaluation trajectories. This may be attributed to a number of factors: 1) the method for modeling the Sun-Earth L_5 short period orbit causes small discontinuities between epochs in the reinforcement learning environment, 2) the tight convergence hyperspheres at the Sun-Earth L_5 short period orbit, and 3) certain policies with neural network parameters that are not easily adaptable to the Sun-Earth-Moon point mass ephemeris model.

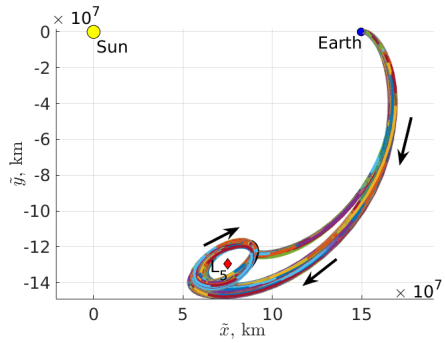
In addition to the evaluation trajectories, Fig. 7.62 displays the reference trajectories for policies P_1 - P_8. While the policies develop evaluation trajectories that struggle to converge on the final orbit, this trend is not exhibited in the reference trajectories because the references are updated using transfers that converge quickly on the final periodic orbit. Additionally, Fig. 7.63 depicts a zoomed-in view of the reference trajectories departing the initial periodic orbit using the same color configuration as Fig. 7.43. While the reference trajectories developed by P_1 rapidly depart the initial periodic in both the Sun-Earth CR3BP and Sun-Earth-Moon point mass ephemeris model, one of the reference trajectories developed by P_8 performs a flyby of the Earth prior to departing through the L_2 gateway. This indicates significant changes in the trajectory geometry during policy updates for a Sun-Earth-Moon point mass ephemeris model. However, the other three reference trajectories constructed by P_8 follow similar paths in both the Sun-Earth CR3BP and Sun-Earth-Moon point mass ephemeris model.

Finally, Fig. 7.64 displays the thrust magnitude profiles of the reference trajectories for policies P_1 - P_8, respectively, in the Sun-Earth-Moon point mass ephemeris model. These figures illustrate similarities between the thrust magnitude profiles associated with the reference trajectories generated for the Sun-Earth CR3BP and Sun-Earth-Moon point mass ephemeris model. Namely, that as the penalty on propellant mass usage increases, the policies tend to produce reference trajectories with smaller thrust magnitudes. However, unlike the reference trajectories developed by

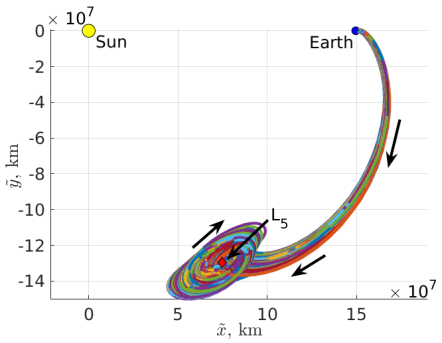


(a) Policy P_1

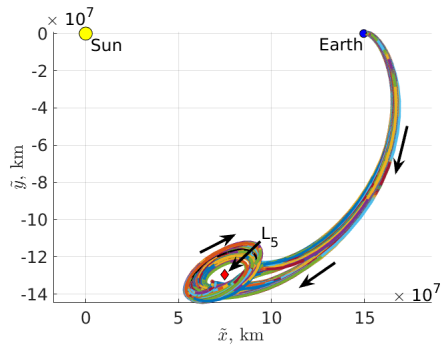
(b) Policy P_2



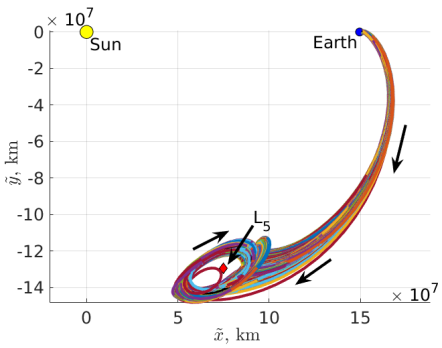
(c) Policy P_3



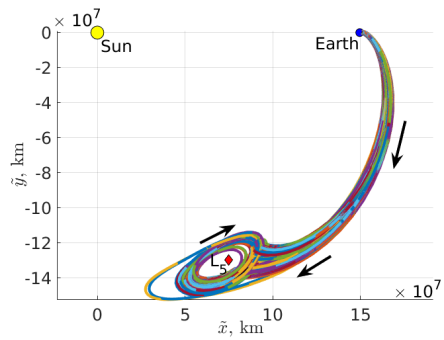
(d) Policy P_4



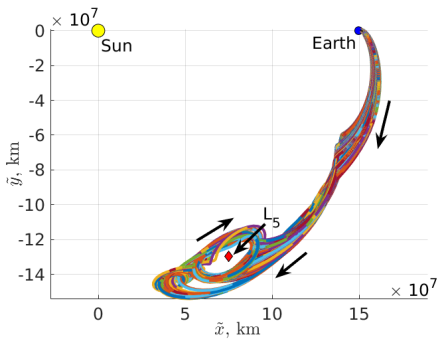
(e) Policy P_5



(f) Policy P_6

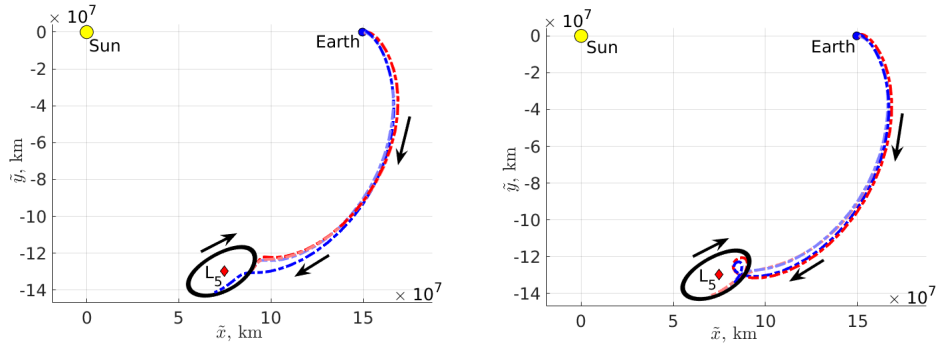


(g) Policy P_7



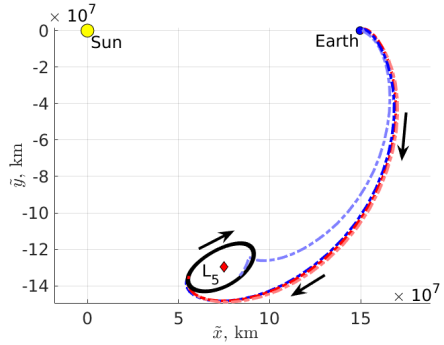
(h) Policy P_8

Figure 7.61: Evaluation trajectories developed by policies P_1 - P_8 using MRPPO for a variable thrust configuration, Sun-Earth-Moon point mass ephemeris model transfer (primaries not to scale).

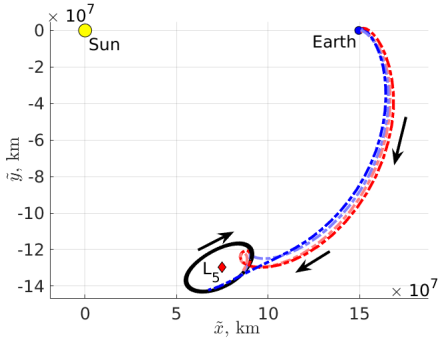


(a) Policy P_1

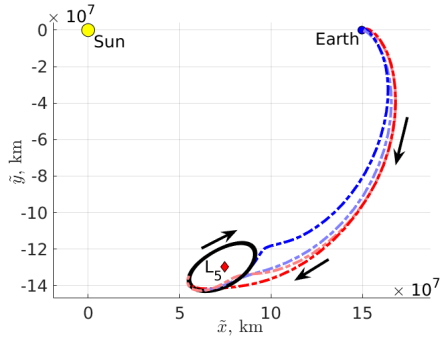
(b) Policy P_2



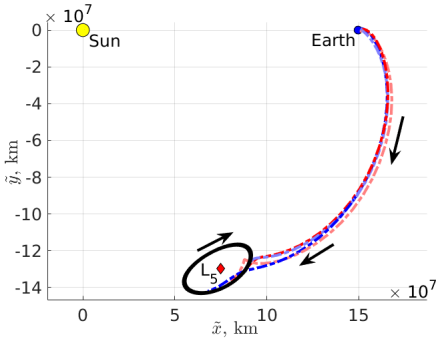
(c) Policy P_3



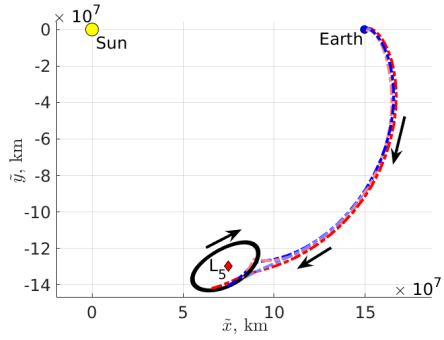
(d) Policy P_4



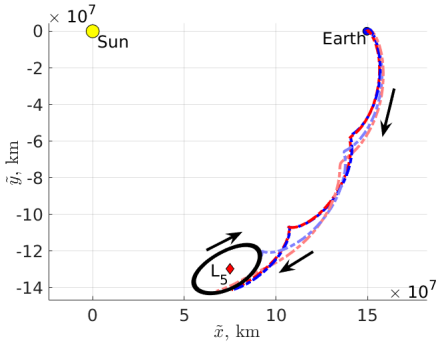
(e) Policy P_5



(f) Policy P_6

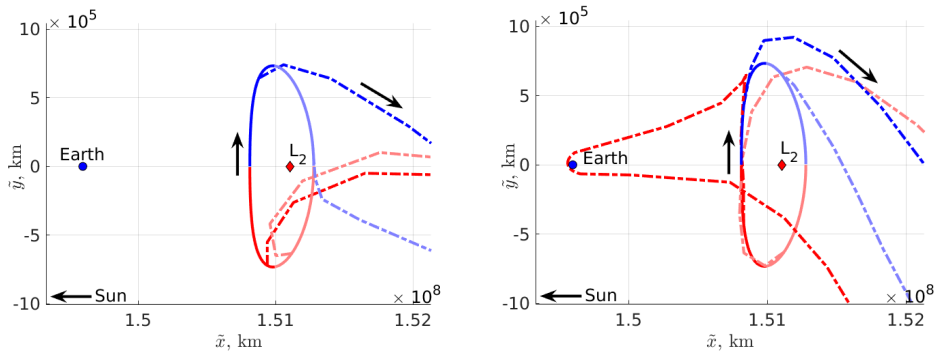


(g) Policy P_7



(h) Policy P_8

Figure 7.62: Reference trajectories developed by policies P_1 - P_8 using MRPPO for a variable thrust configuration, Sun-Earth-Moon point mass ephemeris model transfer (primaries not to scale).



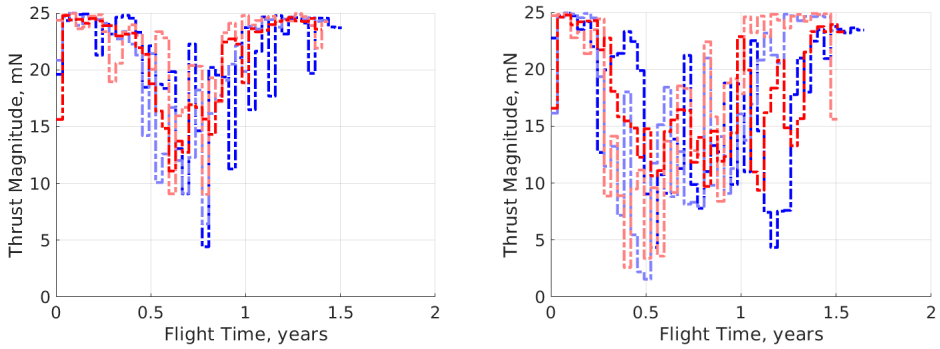
(a) Policy P_1 (b) Policy P_8

Figure 7.63: Zoomed-in view of the reference trajectories developed for policies P_1 and P_8 using MRPPO for a variable thrust configuration, Sun-Earth-Moon point mass ephemeris model transfer (Earth not to scale).

policies P_3 - P_7 for variable thrust transfers in the Sun-Earth CR3BP that possess a bang-bang-like thrust profile, in the Sun-Earth-Moon point mass ephemeris model, policies P_3 - P_7 generate reference trajectories that have a more continuous thrust profile. Additionally, Fig. 7.65 depicts the thrust direction profiles for the reference trajectories by policies P_1 and P_8. Across the reference trajectories, the policies tend to produce similar thrust direction profiles. These low-thrust transfers demonstrate the potential for transfer learning to facilitate training policies in complex dynamical models in a manner that reduces the required computational resources.

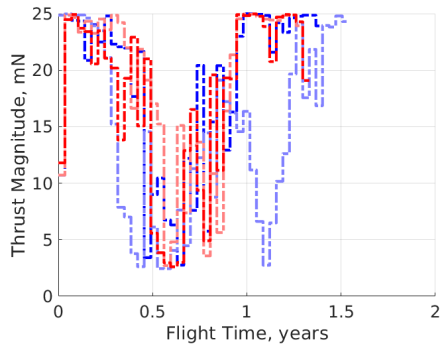
7.3.5 Constant Thrust Transfers in the Sun-Earth CR3BP

MRPPO is also used to train policies to construct low-thrust transfers for a constant thrust engine configuration in the Sun-Earth CR3BP to generate insights into the propellant mass usage and flight time multi-objective solution space. Figure 7.66 depicts the evaluation trajectories for policies P_1-P_8 using the same color configuration as Fig. 7.41. For this constant thrust configuration and dynamical model, the policies are evaluated to produce 1,000 transfers that converge to the final orbit. Additionally, the reference trajectories generated by policies P_1 - P_8 are displayed in Fig. 7.67. The evaluation trajectories developed by policy P_1 possess similar geometries to the

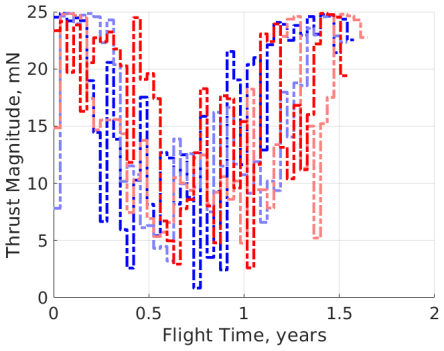


(a) Policy P_1

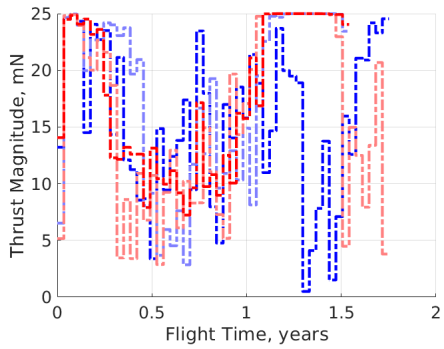
(b) Policy P_2



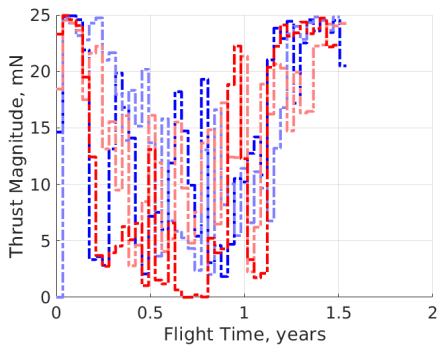
(c) Policy P_3



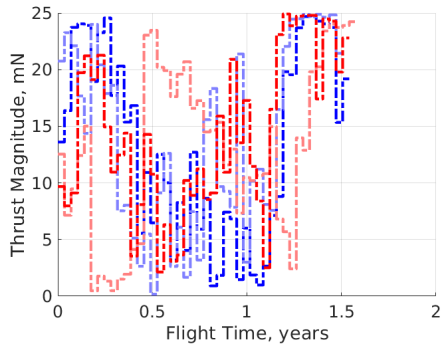
(d) Policy P_4



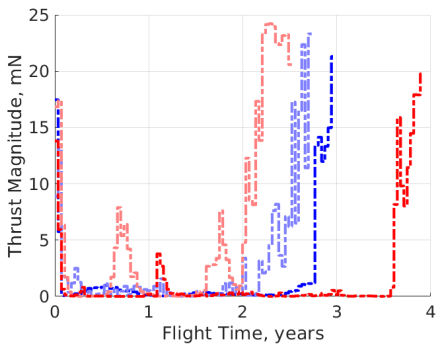
(e) Policy P_5



(f) Policy P_6

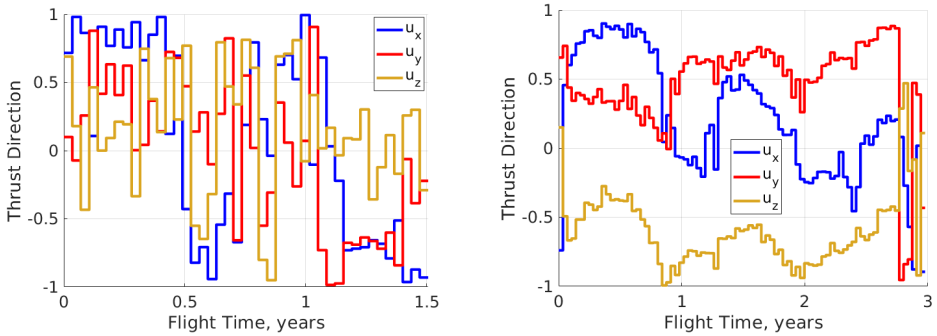


(g) Policy P_7



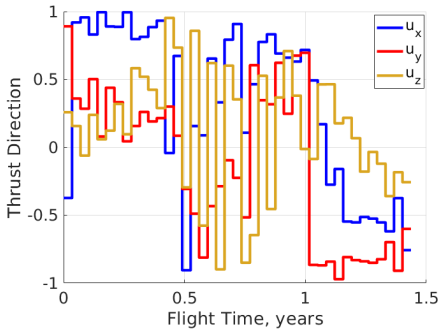
(h) Policy P_8

Figure 7.64: Thrust magnitude profiles of the reference trajectories developed by policies P_1 - P_8 using MRPPO for a variable thrust configuration, Sun-Earth-Moon point mass ephemeris model transfer.

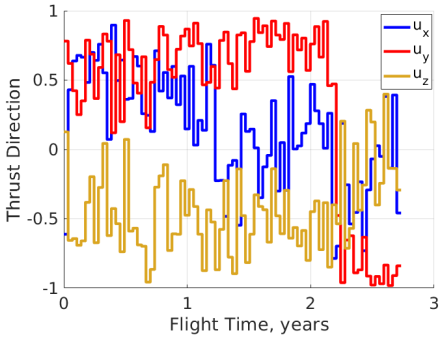


(a) Policy P_1 blue reference

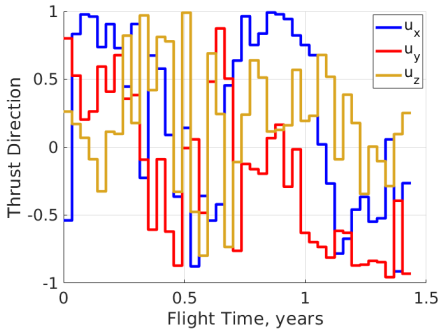
(b) Policy P_8 blue reference



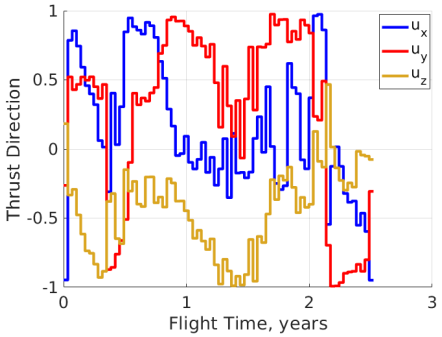
(c) Policy P_1 light blue reference



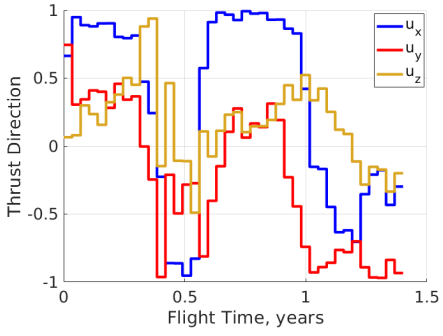
(d) Policy P_8 light blue reference



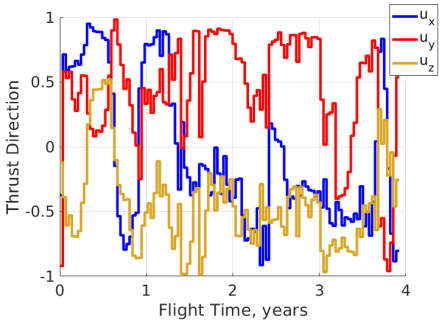
(e) Policy P_1 light red reference



(f) Policy P_8 light red reference



(g) Policy P_1 red reference

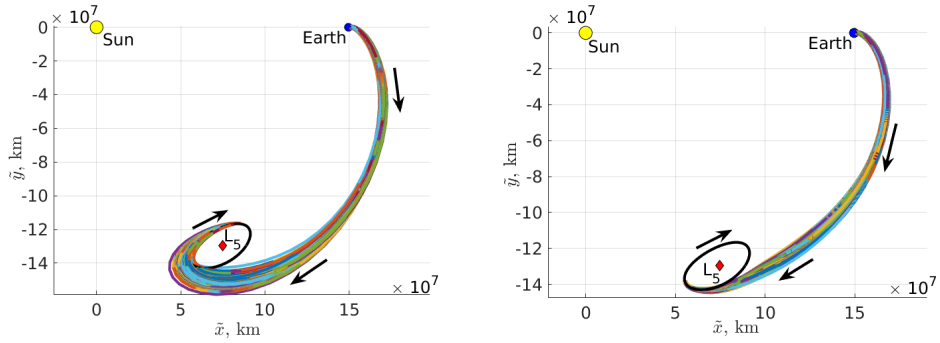


(h) Policy P_8 red reference

Figure 7.65: Thrust direction profiles of the reference trajectories developed by policies P_1 and P_8 using MRPPO for a variable thrust configuration, Sun-Earth-Moon point mass ephemeris model transfer.

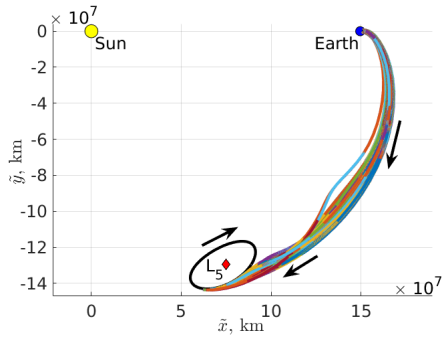
reference trajectories, which more directly target the final orbit. Conversely, the evaluation and reference trajectories developed by policies P_2 - P_8 possess increasing numbers of periapses with respect to the Sun. The policies develop distinct transfer geometries depending on where the trajectories depart the initial periodic orbit, which are reflective of the penalty on propellant mass usage that each policy is assigned.

The characteristics of the reference trajectories are further examined to generate insights into the recovered solutions. Figure 7.68 depict the thrust magnitude profiles of the reference trajectories generated by policies P_1 - P_8. The figures demonstrate that the reference trajectories developed by policy P_1 thrust nearly the entire transfer while the reference trajectories developed by policies P_2 - P_8 coast for a majority of the transfer. This may be attributed to the objective function used to determine when to update the reference trajectories which prioritizes conserving propellant mass. Figure 7.69 exhibits the change in the Jacobi constant due to the activation of the low-thrust engine throughout the reference trajectories developed by policies P_1 - P_8. The reference trajectories developed by P_1 possess Jacobi constants that vary significantly depending on where along the initial orbit the reference trajectory departs. While all four reference trajectories have similar Jacobi constants at the beginning of the trajectories, the Jacobi constants diverge approximately halfway through the transfer before ultimately approaching the Jacobi constant of the final periodic orbit. In fact, the light blue and light red Jacobi constant profiles are approximately mirrored across the axis of the Jacobi constant of the initial periodic orbit. Conversely, the reference trajectories developed by P_2 - P_8 possess Jacobi constants that remain near the Jacobi constant of the initial periodic orbit until the trajectories enter into the vicinity of the final periodic orbit. Once the low-thrust engine is activated, the Jacobi constants of the reference trajectories developed by P_2 - P_8 are decreased to approach the Jacobi constant of the final periodic orbit. However, similar to the variable thrust transfers, there is a noticeable gap between the Jacobi constant at the conclusion of the reference trajectories using a constant thrust engine and the Jacobi constant of the final orbit. This is due to the transfers reaching the non-zero position and velocity hyperspheres, which determine convergence to the final orbit. Reducing the size of the hyperspheres would reduce the

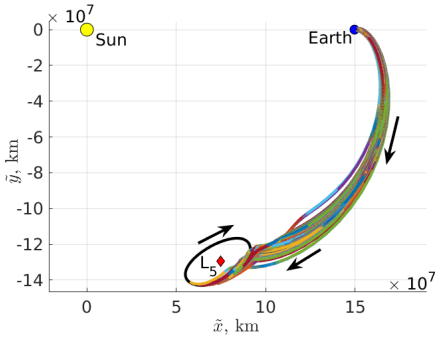


(a) Policy P_1

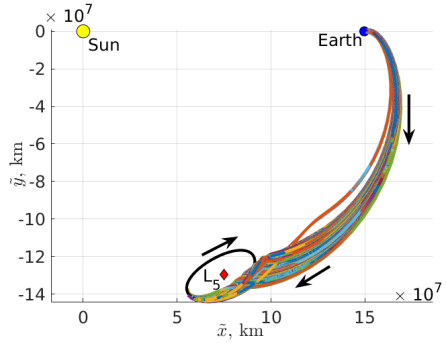
(b) Policy P_2



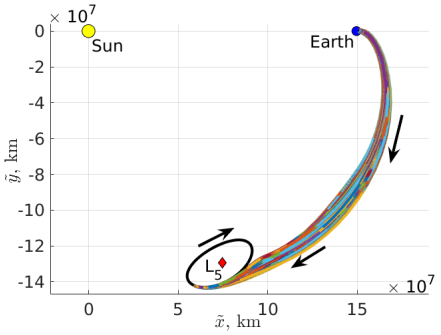
(c) Policy P_3



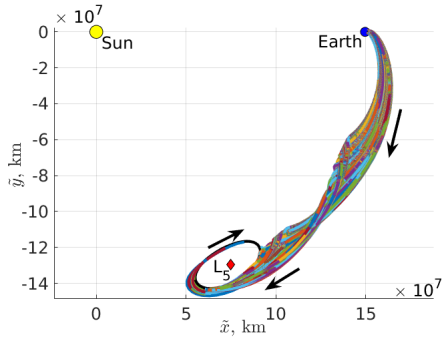
(d) Policy P_4



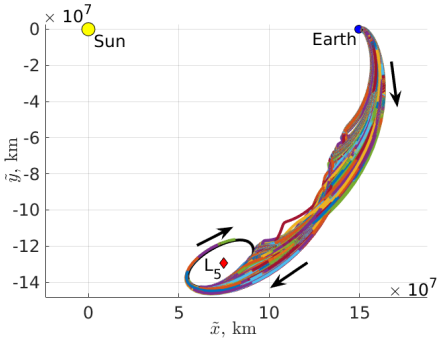
(e) Policy P_5



(f) Policy P_6

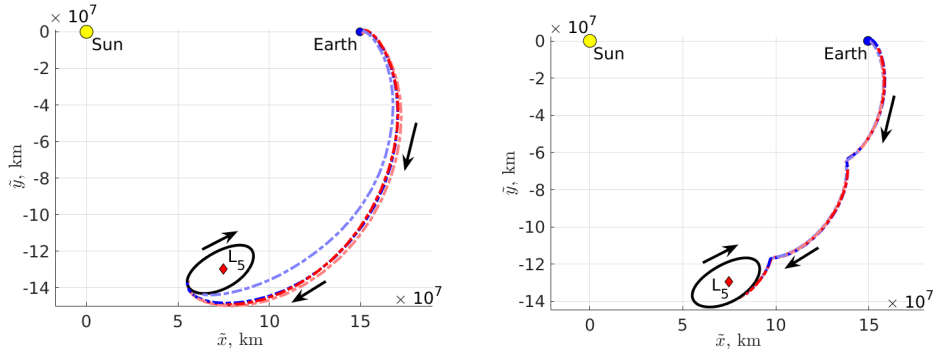


(g) Policy P_7

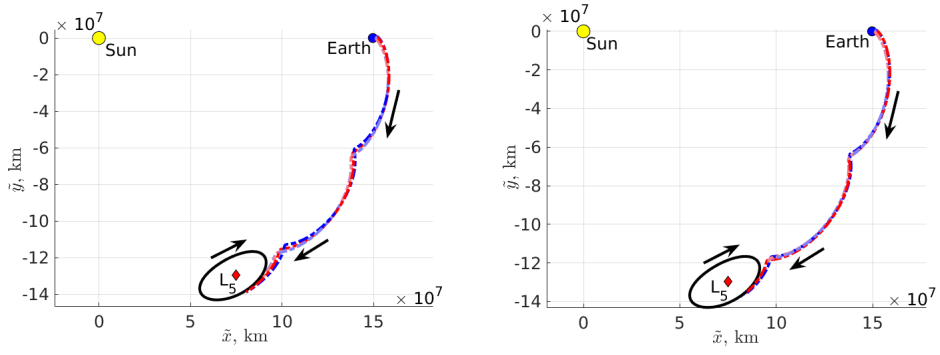


(h) Policy P_8

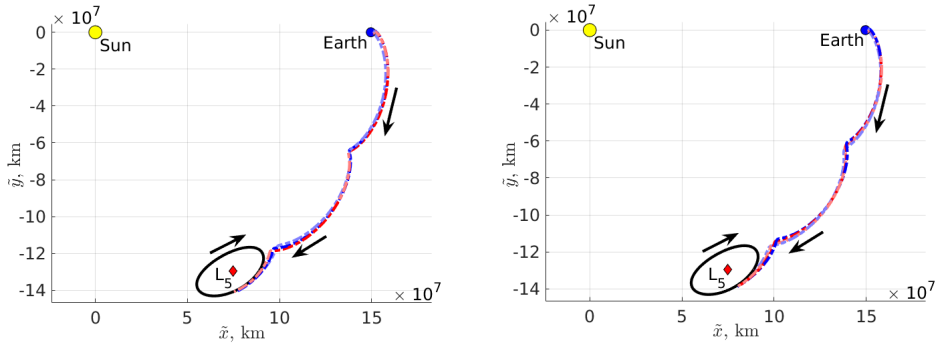
Figure 7.66: Evaluation trajectories developed for policies P_1 - P_8 using MRPPO for a constant thrust configuration, Sun-Earth CR3BP transfer (primaries not to scale).



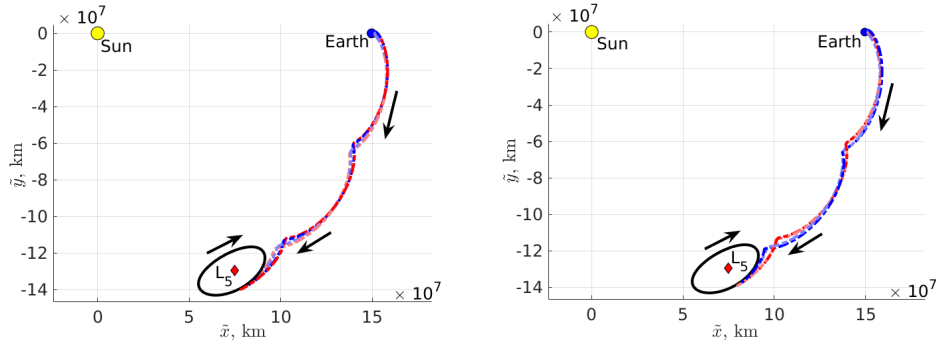
(a) Policy P_1 (b) Policy P_2



(c) Policy P_3 (d) Policy P_4



(e) Policy P_5 (f) Policy P_6



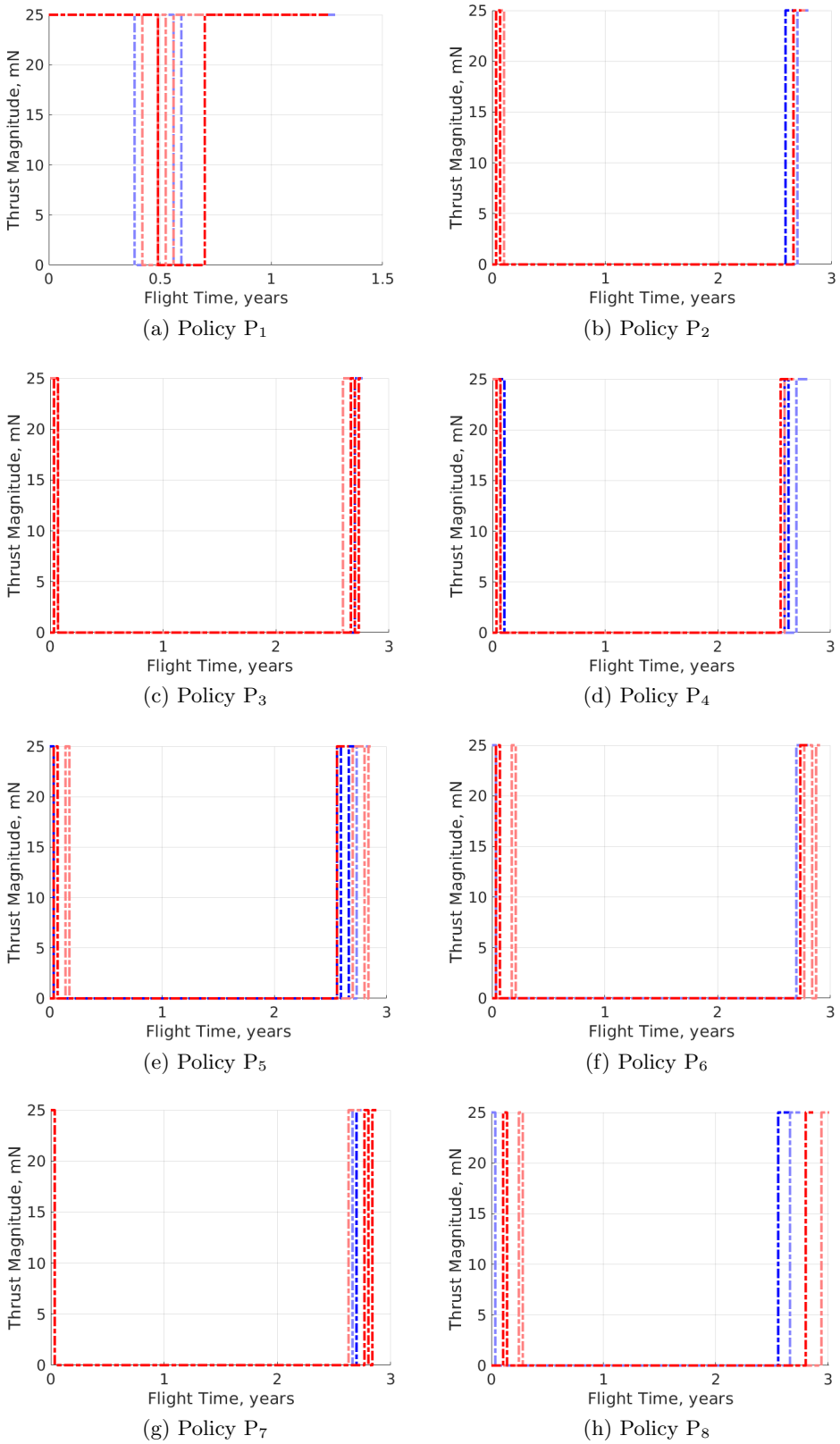
(g) Policy P_7 (h) Policy P_8

Figure 7.67: Reference trajectories developed for policies P_1 - P_8 using MRPPO for a constant thrust configuration, Sun-Earth CR3BP transfer (primaries not to scale).

gap between the Jacobi constants at the risk of causing the policies to struggle to consistently target the hyperspheres. An alternative approach may incorporate a controls scheme, potentially distinct neural networks, that specialize in minimizing the state deviations when the spacecraft is near the final orbit.

The propellant mass usage and flight time characteristics of the evaluation and reference trajectories further demonstrate that MRPPO successfully recovers a wide variety of solutions in this trajectory design scenario. Figures 7.70a and 7.70b depict the characteristics of the evaluation trajectories and reference trajectories, respectively, developed by the eight policies for the propellant mass usage and flight time trade space. While the policies developed by MRPPO produce evaluation trajectories that span the propellant mass usage and flight time trade space, the reference trajectories are instead clustered into two distinct groupings. The reference trajectories constructed by P_1 possess similar characteristics to the evaluation trajectories generated by P_1. However, the reference trajectories constructed by the other policies are close to the maximum final mass solution rather than spanning the propellant mass usage and flight time trade space. This again may be attributed to the function defined in Eq. 5.1 and used to determine when to update a reference trajectory. This function generally prioritizes limiting propellant mass usage rather compared to minimizing the flight time. The maximum final mass transfer requires 2.82 kg of propellant mass usage while the minimum flight time transfer recovered is 1.23 years.

Slight differences in the solution spaces produced for the variable thrust and constant thrust engine configurations also exist. Compared to the policies trained for a variable thrust engine in the CR3BP, for a constant thrust engine, the policies P_1 and P_6 produce transfers with higher propellant mass usages and similar flight times; policies P_2, P_4, P_5 develop transfers with lower propellant mass usages and similar flight times; policy P_7 constructs transfers with lower propellant mass usages and longer flight times; and policies P_3 and P_8 generate transfers with similar propellant mass usages and flight times. These small differences may be attributed to both the distinct engine configurations the policies are trained on and the stochastic nature of the training process. These policies demonstrate that MRPPO may be used to recover low-thrust transfers from an L_2



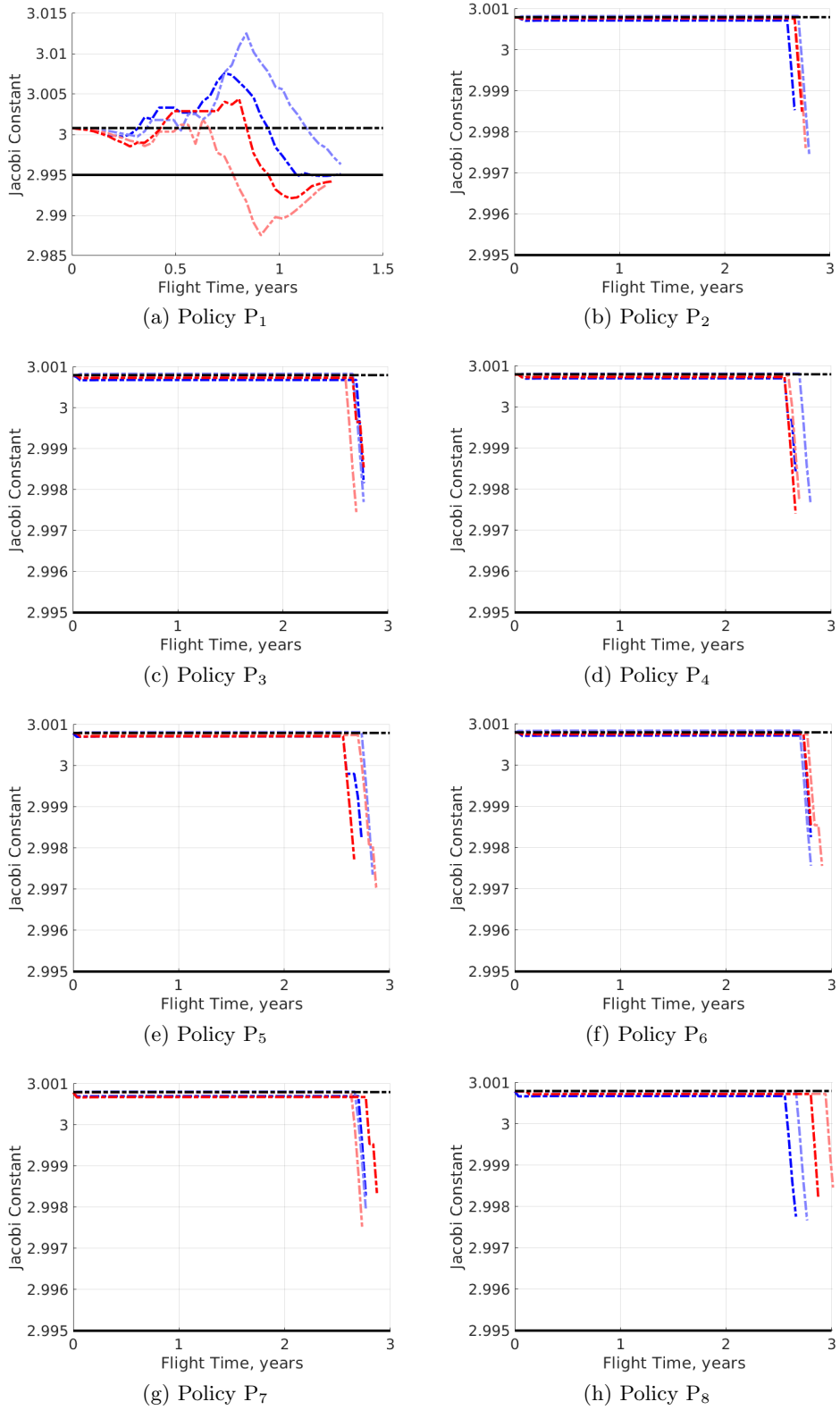
(a) Policy P_1 (b) Policy P_2

(c) Policy P_3 (d) Policy P_4

(e) Policy P_5 (f) Policy P_6

(g) Policy P_7 (h) Policy P_8

Figure 7.68: Thrust magnitude profiles of the reference trajectories developed for policies P_1 - P_8 using MRPPO for a constant thrust configuration, Sun-Earth CR3BP transfer.



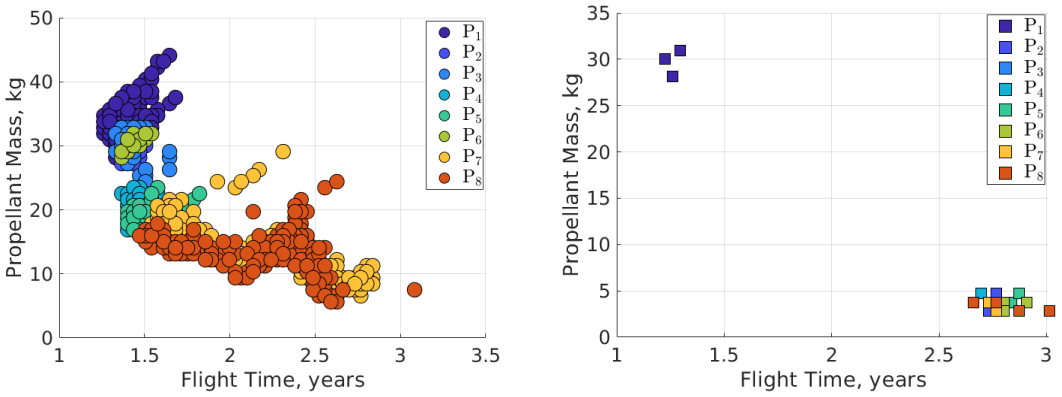
(a) Policy P_1 (b) Policy P_2

(c) Policy P_3 (d) Policy P_4

(e) Policy P_5 (f) Policy P_6

(g) Policy P_7 (h) Policy P_8

Figure 7.69: Jacobi constants of the reference trajectories developed for policies P_1 - P_8 using MRPPO for a constant thrust configuration, Sun-Earth CR3BP transfer.



(a) Evaluation solution space

(b) Reference solution space

Figure 7.70: Evaluation and reference solution spaces developed using MRPPO for a constant thrust configuration, Sun-Earth CR3BP transfer.

northern halo orbit to an L_5 short period orbit for both variable thrust and constant thrust engine configurations in the Sun-Earth CR3BP.

7.3.6 Constant Thrust Transfers in the Sun-Earth-Moon Point Mass Ephemeris Model

Policies trained to produce locally optimal low-thrust transfers for a constant thrust engine configuration in the Sun-Earth CR3BP are used to initialize neural networks for training in the Sun-Earth-Moon point mass ephemeris model. This engine and dynamical model combination is most similar to the exploration performed by Elliott et al. which used a dynamical systems theory approach to design trajectories for SmallSats from secondary deployment conditions near the Earth to a Sun-Earth L_5 short period orbit in the Sun-Earth-Moon point mass ephemeris model [48]. Although that analysis did not perform optimziation or explore the flight time and propellant mass usage trade space, it offers a useful comparison to these results. However, in this investigation, MRPPO is used to recover locally optimal transfers using a higher maximum thrust magnitude and different initial conditions. Figure 7.71 displays the total reward through each update to the policies to demonstrate that the policies have improved during training. In the Sun-Earth CR3BP,

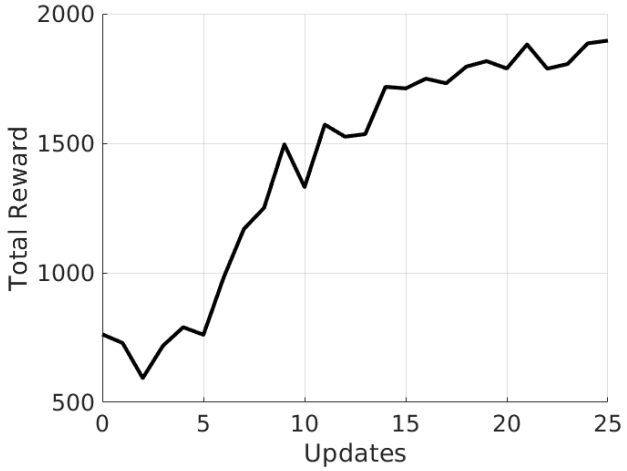


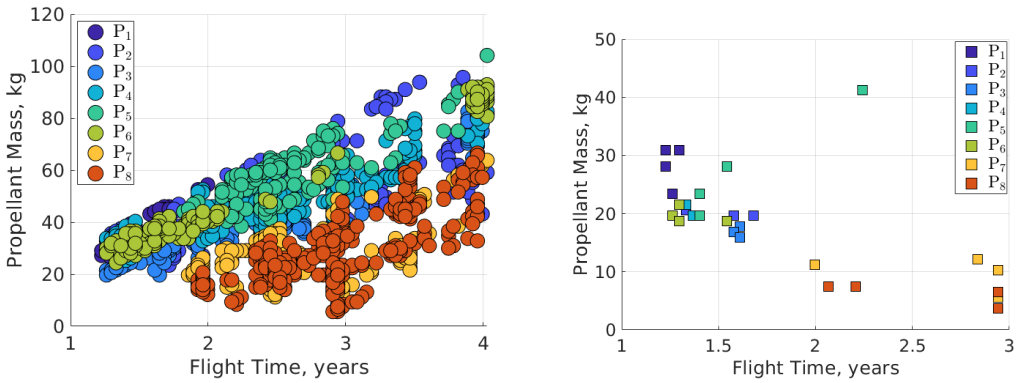
Figure 7.71: Total reward developed using transfer learning for a constant thrust, Sun-Earth-Moon point mass ephemeris model transfer.

the policies return a cumulative reward of 5850 once training is concluded. However, in the Sun-Earth-Moon point mass ephemeris model, the initial guess for the policies returns a total reward of 762 prior to training. After training, the policies are significantly improved and converge to return a total reward of 1896. Additionally, without the initial guess, policies initialized with random weights and biases return a total reward of approximately -8000. While transfers to the Sun-Earth L_5 short period orbit are feasible for each initial condition drawn, a number of challenges prevent the policies from being retrained to consistently target the final periodic orbit. These challenges include: 1) the small convergence hyperspheres at the final periodic orbit, 2) the ephemeris trajectories used to approximate the Sun-Earth L_5 short period orbit possessing discontinuities between epochs in the reinforcement learning environment, 3) the pulsating rotating frame in the Sun-Earth-Moon point mass ephemeris model, and 4) the neural network parameters of the trained policies are locally optimal in the Sun-Earth CR3BP, but may not easily be modified for the Sun-Earth-Moon point mass ephemeris model. Due to these challenges, the policies struggle to return higher total rewards without increasing the size of the convergence hyperspheres.

Additional insights are generated from examining the propellant mass usage and flight time trade space for this trajectory design scenario. Figures 7.72a and 7.72b depict the evaluation

trajectory and reference trajectory solution spaces, respectively, for this engine and dynamical model combination for all trajectories. However, if a trajectory does not converge on the L_5 short period orbit, the characteristics of that trajectory are omitted from the evaluation trajectory solution space. Compared to the transfers produced in the Sun-Earth CR3BP, the policies trained in the Sun-Earth-Moon point mass ephemeris model possess a larger variation in flight time and propellant mass usage across evaluation trajectories. Yet, the reference trajectories are more evenly distributed across the propellant mass usage and flight time trade space. This may be attributed to many of the evaluation trajectories requiring an additional revolution about the Sun-Earth L_5 short period orbit prior to converging within the specified hyperspheres. Additionally, a significant number of the evaluation trajectories for policies P_2, P_4, P_5, and P_8 fail to converge. This also may be attributed to the differences between modeling transfers in the Sun-Earth and the local optimally training parameters that these policies recovered not being as transferable to a higher-fidelity dynamical model. However, of the transfers that do converge, the maximum final mass transfer requires 3.75 kg of propellant mass usage while the minimum flight time transfer recovered is 1.23 years. While all of the evaluation trajectories reach the vicinity of the Sun-Earth L_5 short period orbit, approximately 5% of the transfers for policies P_3 and P_6 - P_8 and approximately 50% of the transfers for policies P_2, P_4, and P_5 do not enter the convergence hyperspheres defined for the final orbit. These are some of the policies that have significant changes in transfer geometry suggesting that through this transfer learning approach, the policies are significantly disrupted and are unable to stay close to the behavior that they developed in the Sun-Earth CR3BP.

Then, the evaluation and reference trajectories also provide insights into the characteristics of the recovered multi-objective solution space. Figure 7.73 illustrates the evaluation trajectories that converge on the final orbit for policies P_1-P_8, demonstrating that a number of the policies struggle to produce transfers that converge on the final orbit within the specified convergence hyperspheres without additional revolutions near the Sun-Earth L_5 short period orbit. This additional revolution drives the increase in flight time exhibited in Fig. 7.72a for all of the policies. Further, policies P_2-P_6 possess the most significant change in transfer geometries, which corresponds to decreased rates



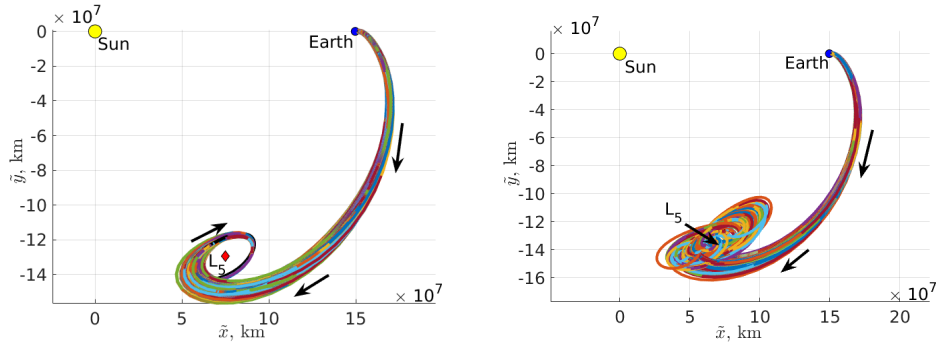
(a) Evaluation solution space

(b) Reference solution space

Figure 7.72: Evaluation and reference solution spaces developed using transfer learning for a constant thrust configuration, Sun-Earth-Moon point mass ephemeris model transfer.

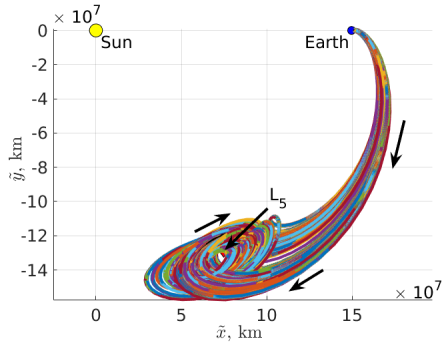
of convergence to the final orbit. This again may be attributed to these policies being unable to recover the transfer geometries used in the Sun-Earth CR3BP in this higher-fidelity model and being required to relearn a new transfer geometry to target the final periodic orbit. Additional training or modifications to the methodology may improve the convergence rate of policies in this dynamical and engine model combination. Further, Fig. 7.74 depicts the reference trajectories generated by each policy in the Sun-Earth-Moon point mass ephemeris model. Similar to the variable thrust configuration, the transfers in the ephemeris model produced by policies with smaller penalties on propellant mass usage tend to possess more similar geometries across distinct departure locations. However, as the penalty on propellant mass usage increases, the policies recover a wider variety of transfer geometries. Additionally, the transfer geometries of the trajectories produced in the Sun-Earth CR3BP for policies P_1, P_7, ad P_8 are approximately retained in the Sun-Earth-Moon point mass ephemeris model.

Finally, Fig. 7.75 displays the thrust magnitude profiles of the reference trajectories generated in the Sun-Earth-Moon point mass ephemeris model. The policies produce reference trajectories with thrust magnitude profiles that gradually increase the amount of coasting time in conjunction with the increase in propellant mass usage penalty for the constant thrust configuration in the Sun-Earth-Moon point mass ephemeris model. Additionally, the reference trajectories produced by

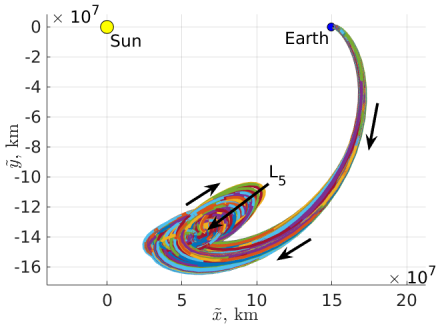


(a) Policy P_1

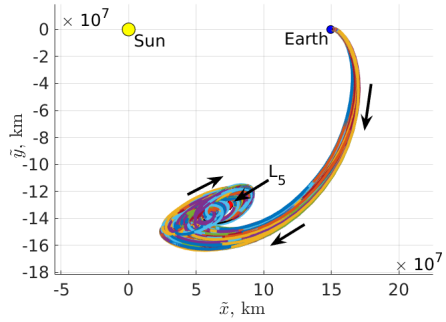
(b) Policy P_2



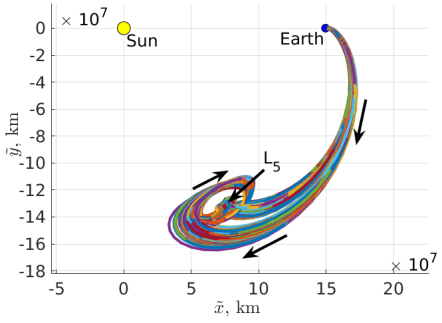
(c) Policy P_3



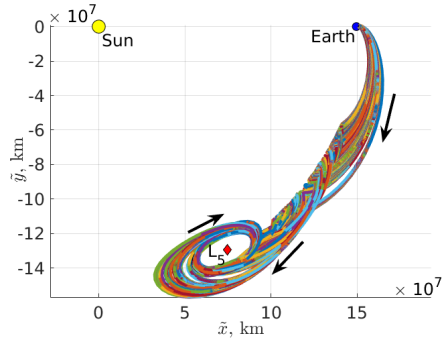
(d) Policy P_4



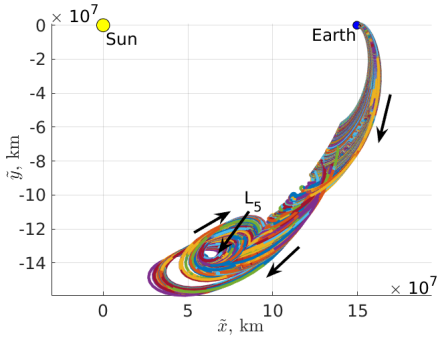
(e) Policy P_5



(f) Policy P_6

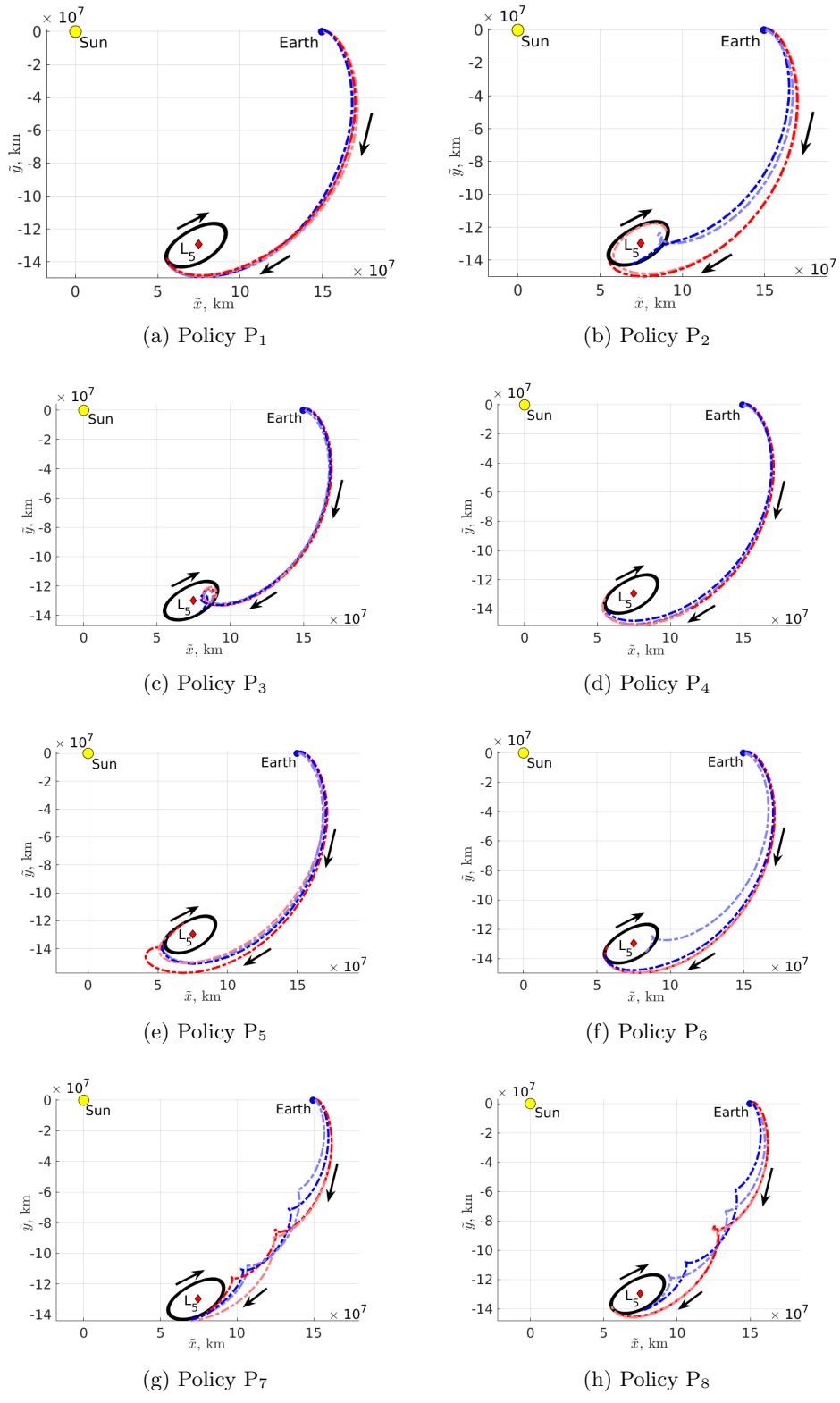


(g) Policy P_7



(h) Policy P_8

Figure 7.73: Evaluation trajectories developed by policies P_1 - P_8 using MRPPO for a constant thrust configuration, Sun-Earth-Moon point mass ephemeris model transfer (primaries not to scale).



(a) Policy P_1 (b) Policy P_2

(c) Policy P_3 (d) Policy P_4

(e) Policy P_5 (f) Policy P_6

(g) Policy P_7 (h) Policy P_8

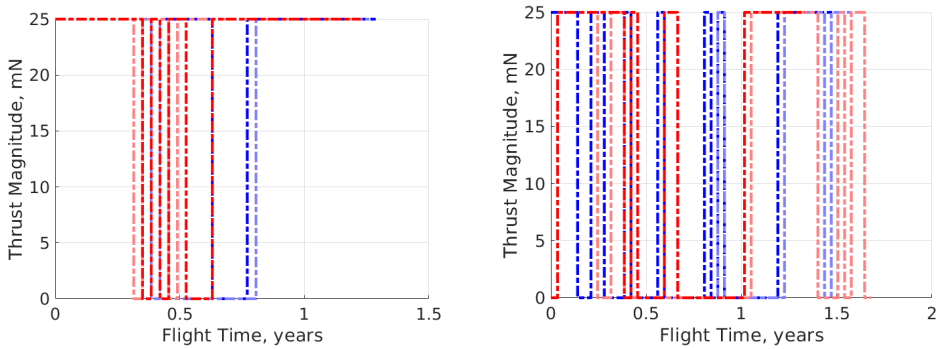
Figure 7.74: Reference trajectories developed by policies P_1 - P_8 using MRPPO for a constant thrust configuration, Sun-Earth-Moon point mass ephemeris model transfer (primaries not to scale).

policies P_1 and P_8 in both the Sun-Earth CR3BP and Sun-Earth-Moon point mass ephemeris model retain similar thrust magnitude profiles. However, policies P_2 - P_7 develop reference trajectories that possess more thrust time in the Sun-Earth-Moon point mass ephemeris model than reference trajectories in the Sun-Earth CR3BP. Further, while the investigation performed by Elliott et al. assumed that the spacecraft deployed as a secondary payload with distinct initial conditions with a smaller maximum available thrust and recovered transfers requiring 2.578 years of flight time and 30.31 kg of propellant mass, the transfers recovered by policy P_8 in this investigation possess a similar geometry, thrust profile, and a smaller flight time and required propellant mass usage [48].

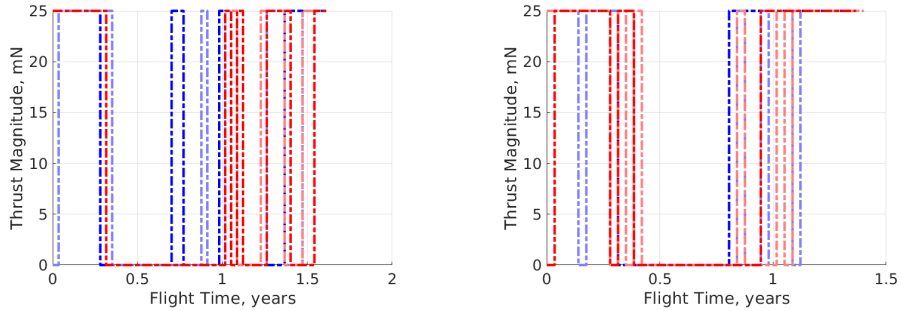
7.4 Summary of Key Findings

MRPPO is used to train policies that generate low-thrust transfers that span the propellant mass usage and flight time solution space in three trajectory design scenarios. In each scenario, insights into the behavior of the policies is generated via analysis of the evaluation and reference trajectories developed by MRPPO and comparison to other methodologies. The key findings of this dissertation include:

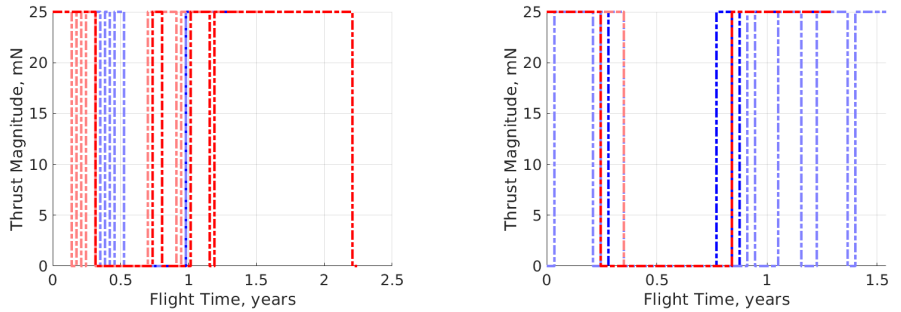
- In the first trajectory design scenario, policies recover transfers with continuous thrust profiles. However, in the second and third trajectory design scenarios, the reference and evaluation trajectories recovered by MRPPO possess a bang-bang-like control profile resembling invariant manifold arcs.
- MRPPO recovers better solutions than PPO for the first trajectory design scenario for a lower computational time.
- The low-thrust transfers for both a variable thrust and constant thrust engine configuration in the third trajectory design scenario are compared and demonstrate similar geometries and characteristics.
- In the first trajectory design scenario, the optimization algorithm produces transfers with similar total rewards, but lower propellant mass usages by transforming the thrust magni-



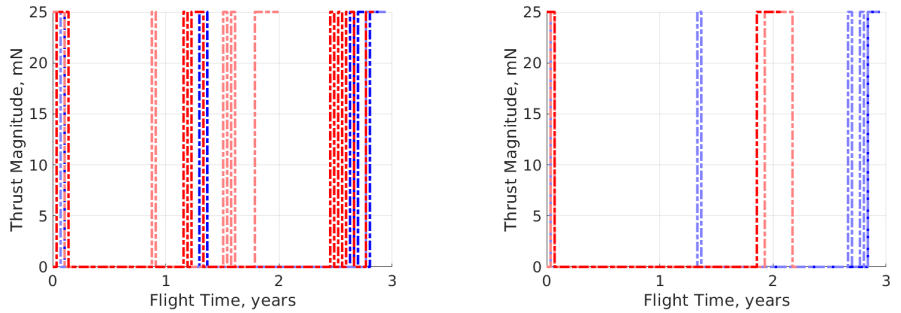
(a) Policy P_1 (b) Policy P_2



(c) Policy P_3 (d) Policy P_4



(e) Policy P_5 (f) Policy P_6



(g) Policy P_7 (h) Policy P_8

Figure 7.75: Thrust magnitude profiles of the reference trajectories developed by policies P_1 - P_8 using MRPPO for a constant thrust configuration, Sun-Earth-Moon point mass ephemeris model transfer.

tude profiles of the transfers to a bang-bang-like control scheme. However, in the second and third trajectory design scenarios, the reference trajectories developed by the policies closely resemble the solutions produced by the optimization scheme. In all three trajectory design scenarios, MRPPO does produce meaningful results. However, how close the results are to locally optimal solutions developed by an optimization scheme does vary.

- An exploration into the number of reference trajectories per policy reveals that larger numbers of reference trajectories per policy enables policies to recover a wider variety of transfer geometries in the third trajectory design scenario, but has a detrimental influence on the policies in the second trajectory design scenario.

- A transfer learning approach enables the policies to rapidly develop transfers in the higher-fidelity model. However, due to a number of factors, a significant percentage of the evaluation trajectories fail to converge on the final orbit in the Sun-Earth-Moon point mass ephemeris model. Transfer learning was not as successful as required, and significant future development is needed for updating policies for higher-fidelity dynamical models.

Chapter 8

Concluding Remarks

8.1 Summary

Developing new techniques that autonomously generate optimal low-thrust transfers in multi-body gravitational environments provides insights into a trajectory design scenario while limiting the time and effort required of a human trajectory designer. Reinforcement learning is one approach to autonomously recover optimal solutions in an unknown dynamical environment while balancing exploration of new regions of the solution space with the exploitation of current knowledge. Specifically, reinforcement learning is an approach that updates a controls scheme that may be modeled using neural networks to develop optimal behavior that maximizes a single set of objectives. However, in multi-body trajectory design scenarios, low-thrust transfers typically exist within multi-objective solution spaces. Further, these solution spaces are often examined through the nondominating solutions that distinctly balance multiple objectives to generate insights into the trade space. To efficiently and autonomously uncover multiple low-thrust transfers spanning a multi-objective solution space in a multi-body trajectory design scenario, multi-objective reinforcement learning is employed.

A multi-objective reinforcement learning algorithm, designated Multi-Reward Proximal Policy Optimization (MRPPO), is designed using Proximal Policy Optimization (PPO) as a foundation to enable multiple policies to be trained simultaneously in a shared environment with a multi-objective solution space. By sharing propagation data from the dynamical model, each policy may learn from the potentially beneficial or disadvantageous actions undertaken by other policies

thereby introducing additional exploration into the training; this produces a more stable, efficient training phase for MRPPO compared to PPO. Additionally, the structure of MRPPO reduces the computational resources and time required to train policies compared to sequentially training the policies using PPO.

To generate these solutions, the transfer design problem is translated into a reinforcement learning problem that also supports a successful and efficient training process. Inspired by the concept of a waypoint that is often used in path planning, this reinforcement learning problem is formulated using information about the best transfers generated during training. This approach balances exploitation by incorporating the best transfers into the state and reward definitions with exploration by updating these best transfers using training data. By incorporating information about the best transfers generated throughout training, policies recover feasible solutions for low-thrust transfers between periodic orbits. Then, MRPPO is used to simultaneously train multiple policies, each with a scalar reward function that reflects a distinct relative weighting between the two competing objectives of minimizing flight time and propellant mass usage.

In this investigation, MRPPO is applied to train policies that construct low-thrust transfers balancing multiple objectives in three trajectory design scenarios: 1) between L_2 southern halo orbits in the Earth-Moon CR3BP, 2) from an L_1 northern halo orbit to an L_2 southern halo orbit in the Earth-Moon CR3BP, and 3) from an L_2 northern halo orbit to an L_5 short period orbit in both the Sun-Earth CR3BP and Sun-Earth-Moon point mass ephemeris model. The trained policies for each trajectory design scenario are evaluated using the reference trajectories and evaluation trajectories generated from a common set of initial conditions near the initial periodic orbit. This enables the recovery of a subset of solutions along the propellant mass usage and flight time trade space and insights into the transfer geometries recovered by the policies. Additionally, for the first trajectory design scenario, a preliminary investigation into the effect of hyperparameter selection is performed to generate insights into the influence of each hyperparameter on the behavior of the policies throughout the training process. Further, for the third trajectory design scenario, low-thrust transfers are designed for both a variable thrust and constant thrust engine configuration

and transfer learning is employed to update policies for the Sun-Earth-Moon point mass ephemeris model. Exploring distinct engine configurations facilitates additional insights into how the spacecraft parameters influence the resulting multi-objective solution space. Finally, for each trajectory design scenario, the reference trajectories generated by MRPPO are leveraged as initial guesses in an optimization algorithm to approximately validate the solution space produced by MRPPO. The results generated by the optimization scheme are sufficiently close to the solution space produced by MRPPO. These results demonstrate the applicability and potential of MRPPO to autonomously construct low-thrust transfers spanning the multi-objective solution space in multi-body gravitational environments without an initial guess thereby reducing the time and effort required of a human trajectory designer.

8.2 Future Work

The multi-objective reinforcement learning framework developed in this research and applied for low-thrust trajectory design scenarios in multi-body gravitational environments may benefit from incorporating new research areas within the machine learning and astrodynamics communities. For instance, within the multi-objective reinforcement learning framework, improvements in the convergence behavior may be generated via incorporating:

- Physics-informed neural networks developed by Raissi, Perdikaris, and Karniadakis and used by Schiass et al. and Martin and Schaub for trajectory design in two-body gravitational environments, which ensure the neural networks conform to defined dynamics [93, 116, 122].

- Constrained policy optimization techniques that naturally enforce constraints via the policy updates rather than within the reward function similar to the proposal by Achiam et al. [2]. This may simplify the learning process and enable policies to recover low-thrust transfers that solely maximize the final mass.

- Robust reinforcement learning techniques that enable policies to contend with state or

action uncertainty within a reinforcement learning environment and guarantee local optimality similar to the technique proposed by Tessler, Efroni, and Mannor [153].

Similarly, within the trajectory design scenarios, a number of improvements that may be incorporated to improve the results returned by MRPPO including:

- Trajectory design techniques that enable the recovery of transfers with distinct numbers of revolutions about primary bodies to enable the recovery of distinct segments of the multi-objective solution space.
- Improving the description of target motions in higher-fidelity ephemeris models to improve the performance for the neural networks targeting small convergence hyperspheres.
- Currently, MRPPO can only design solutions that target a dynamical arc to within significant hyperspheres. Reducing the size of the hyperspheres would produce solutions with a higher accuracy, facilitate more direct comparisons to solutions produced by an optimization scheme, and generate insights into transfers targeting tight convergence to a dynamical arc.
- Exploring alternative reward function formulations that are only composed of flight time, propellant mass usage, or other significant goals of the trajectory design scenario.

Once incorporated, MRPPO could be used to robustly and autonomously generate insights into complex trajectory design scenarios with limited time and effort required of a trajectory designer.

Bibliography

[1] Andrew J. Abraham, David B. Spencer, and Terry J. Hart. Early mission design of transfers to halo orbits via particle swarm optimization. The Journal of the Astronautical Sciences, 63(2):81–102, 2016. https://doi.org/10.1007/s40295-016-0084-2.

[2] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In 34th International Conference on Machine Learning, Sydney, AUS, pages 22–31. Proceedings of Machine Learning Research, 2017.

[3] Charles H. Acton. Ancillary data services of nasa's navigation and ancillary information facility. Planetary and Space Science, 44:65–70, 1996. https://doi.org/10.1016/0032-0633(95)00107-7.

[4] M. Akioka, T. Nagatsuma, W. Miyake, K. Ohtaka, and K. Marubashi. The l5 mission for space weather forecasting. Advances in Space Research, (1):65–69, 2005. https://doi.org/10.1016/j.asr.2004.09.014.

[5] Rodney L. Anderson and Martin W. Lo. Role of invariant manifolds in low-thrust trajectory design. Journal of Guidance, Control, and Dynamics, 32(6):1921–1930, 2009. https://doi.org/10.2514/1.37516.

[6] Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphael Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What matters in on-policy reinforcement learning? a large-scale empirical study. arXiv preprint arXiv:2006.05990, 2020.

[7] Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. In 31st Conference on Neural Information Processing Systems, Long Beach, CA, page 5366–5376. Curran Associates Inc., 2017. https://dl.acm.org/doi/10.5555/3295222.3295288.

[8] Jonathan D. Aziz, Daniel Scheeres, and Gregory Lantoine. Differential Dynamic Programming in the Three-Body Problem. 2018. https://doi.org/10.2514/6.2018-2223.

[9] Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. Neural networks, 2(1):53–58, 1989. https://doi.org/10.1016/0893-6080(89)90014-2.

[10] Leon Barrett and Srini Narayanan. Learning all optimal policies with multiple criteria. In 25th International Conference on Machine Learning, Helsinki, FIN, page 41–47. Association for Computing Machinery, 2008. https://doi.org/10.1145/1390156.1390162.

[11] June Barrow-Green. Poincaré and the Three Body Problem. Number 11. American Mathematical Society, 1997.

[12] Andrew G. Barto. Reinforcement learning and dynamic programming. In 6th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design and Evaluation of Man-Machine Systems, Cambridge, MA, volume 28, pages 407–412, 1995. https://doi.org/10.1016/S1474-6670(17)45266-9.

[13] Richard Bellman. The theory of dynamic programming. Bulletin of the American Mathematical Society, 60(6):503–515, 1954.

[14] Richard Bellman. A markovian decision process. Journal of Mathematics and Mechanics, 6(5):679–684, 1957.

[15] John T. Betts. Survey of numerical methods for trajectory optimization. Journal of Guidance, Control, and Dynamics, 21(2):193–207, 1998. https://doi.org/10.2514/2.4231.

[16] John T. Betts. Very low-thrust trajectory optimization using a direct sqp method. Journal of Computational and Applied Mathematics, 120(1):27–40, 2000. https://doi.org/10.1016/S0377-0427(00)00301-0.

[17] Paul T. Boggs and Jon W. Tolle. Sequential quadratic programming. Acta Numerica, 4:1–51, 1995. https://doi.org/10.1017/S0962492900002518.

[18] Stefano Bonasera, Natasha Bosanac, Christopher J. Sullivan, Ian Elliott, Nisar Ahmed, and Jay McMahon. Designing station-keeping maneuvers near a sun-earth l2 halo orbit via reinforcement learning (under review). Journal of Guidance, Control, and Dynamics, 2022.

[19] Stefano Bonasera, Ian Elliott, Christopher J. Sullivan, Natasha Bosanac, Nisar Ahmed, and Jay McMahon. Designing impulsive station-keeping maneuvers near a sun-earth l2 halo orbit via reinforcement learning. In 31st AAS/AIAA Space Flight Mechanics Meeting, Virtual, 02 2021.

[20] Spencer Boone, Stefano Bonasera, Jay W. McMahon, Natasha Bosanac, and Nisar R. Ahmed. Incorporating observation uncertainty into reinforcement learning-based spacecraft guidance schemes. In 32nd AIAA/AAS Space Flight Mechanics Meeting, San Diego, CA, 2022. https://doi.org/10.2514/6.2022-1765.

[21] Spencer Boone and Jay McMahon. Orbital guidance using higher-order state transition tensors. Journal of Guidance, Control, and Dynamics, 44(3):493–504, 2021. https://doi.org/10.2514/1.G005493.

[22] Spencer Boone and Jay McMahon. Variable time-of-flight spacecraft maneuver targeting using state transition tensors. Journal of Guidance, Control, and Dynamics, 44(11):2072–2080, 2021. https://doi.org/10.2514/1.G005890.

[23] Natasha Bosanac. Leveraging Natural Dynamical Structures to Explore Multi-Body Systems. PhD thesis, Purdue University, West Lafayette, IN, 2016.

[24] Natasha Bosanac, Farah Alibay, and Jeffery R. Stuart. A low-thrust enabled smallsat heliophysics mission to sun-earth l5. IEEE Aerospace Conference, Big Sky, MT, 2018. https://doi.org/10.1109/AERO.2018.8396375.

[25] Natasha Bosanac, Stefano Bonasera, Christopher J. Sullivan, Jay McMahon, and Nisar Ahmed. Reinforcement learning for reconfiguration maneuver design in multi-body systems. In AAS/AIAA Astrodynamics Specialist Conference, Virtual, 2021.

[26] Natasha Bosanac, Andrew D. Cox, Kathleen C. Howell, and David C. Folta. Trajectory design for a cislunar cubesat leveraging dynamical systems techniques: The lunar icecube mission. Acta Astronautica, pages 283–296, 2018. https://doi.org/10.1016/j.actaastro.2017.12.025.

[27] Matthew Botvinick, Sam Ritter, Jane X Wang, Zeb Kurth-Nelson, Charles Blundell, and Demis Hassabis. Reinforcement learning, fast and slow. Trends in Cognitive Sciences, 23(5):408–422, 2019. https://doi.org/10.1016/j.tics.2019.02.006.

[28] P. Cage, I. Kroo, and R. Braun. Interplanetary trajectory optimization using a genetic algorithm. In Astrodynamics Conference, Scottsdale, AZ, 1994. https://doi.org/10.2514/6.1994-3773.

[29] Ian Carnelli, Bernd Dachwald, and Massimiliano Vasile. Evolutionary neurocontrol: A novel method for low-thrust gravity-assist trajectory optimization. Journal of Guidance, Control, and Dynamics, 32(2):616–625, 2009. http://dx.doi.org/10.2514/1.32633.

[30] Andrea Castelletti, Giorgio Corani, Andrea E. Rizzolli, Rodolfo Soncinie-Sessa, and Enrico Weber. Reinforcement learning in the operational management of a water system. In IFAC Workshop on Modeling and Control in Environmental Issues, pages 325–330, 2002.

[31] Runqi Chai, Al Savvaris, Antonios Tsourdos, Senchun Chai, and Yuanqing Xia. A review of optimization techniques in spacecraft flight trajectory design. Progress in Aerospace Sciences, 109, 2019. https://doi.org/10.1016/j.paerosci.2019.05.003.

[32] Po-Wei Chou, Daniel Maturana, and Sebastian Scherer. Improving stochastic policy gradients in continuous control with deep reinforcement learning using the beta distribution. In 34th International Conference on Machine Learning, Sydney, AUS, pages 834–843. Proceedings of Machine Learning Research, 2017.

[33] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). arXiv preprint arXiv:1511.07289, 2015.

[34] Guido Colasurdo and Dario Pastrone. Indirect optimization method for impulsive transfers. In Astrodynamics Conference, Scottsdale, AZ, pages 441–448, 1994. https://doi.org/10.2514/6.1994-3762.

[35] V. Coverstone-Carroll, J. W. Hartmann, and W. J. Mason. Optimal multi-objective low-thrust spacecraft trajectories. Computer Methods in Applied Mechanics and Engineering, 186(2-4):387–402, 2000. https://doi.org/10.1016/S0045-7825(99)00393-X.

[36] Andrew D. Cox, Kathleen C. Howell, and David C. Folta. Dynamical structures in a low-thrust, multi-body model with applications to trajectory design. Celestial Mechanics and Dynamical Astronomy, 131(3):1–34, 2019. https://doi.org/10.1007/s10569-019-9891-7.

[37] Bernd Dachwald. Low-thrust trajectory optimization and interplanetary mission analysis using evolutionary neurocontrol. Doktorarbeit, Institut für Raumfahrttechnik, Universität der Bundeswehr, München, 2004.

[38] Bernd Dachwald. Optimization of interplanetary solar sailcraft trajectories using evolutionary neurocontrol. Journal of Guidance, Control, and Dynamics, 27(1):66–72, 2004. https://doi.org/10.2514/1.9286.

[39] Ashwati Das-Stuart. Artificial Intelligence Aided Rapid Trajectory Design in Complex Dynamical Environments. PhD thesis, Purdue University, West Lafayette, IN, 2019.

[40] Ashwati Das-Stuart, Kathleen C. Howell, and David C. Folta. Rapid trajectory design in complex environments enabled by reinforcement learning and graph search strategies. Acta Astronautica, 171:172–195, 2020. https://doi.org/10.1016/j.actaastro.2019.04.037.

[41] Diane C. Davis, Sean M. Phillips, Kathleen C. Howell, Srianish Vutukuri, and Brian P. McCarthy. Stationkeeping and transfer trajectory design for spacecraft in cislunar space. In AAS/AIAA Astrodynamics Specialist Conference, Stevenson, WA, 2017.

[42] Kalyanmoy Deb, Nikhil Padhye, and Ganesh Neema. Interplanetary trajectory optimization with swing-bys using evolutionary multi-objective optimization. In International Symposium on Intelligence Computation and Applications, pages 26–35. Springer, 2007.

[43] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Transactions on Evolutionary Computation, 6(2):182–197, 2002.

[44] Marc Deisenroth and Carl E. Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In 28th International Conference on Machine Learning, Bellevue, WA, pages 465–472. Citeseer, 2011.

[45] Michael Dellnitz, Sina Ober-Blöbaum, Marcus Post, Oliver Schütze, and Bianca Thiere. A multi-objective approach to the design of low thrust space trajectories using optimal control. Celestial Mechanics and Dynamical Astronomy, 105(1):33–59, 2009. https://doi.org/10.1007/s10569-009-9229-y.

[46] Edsger W. Dijkstra. A note on two problems in connexion with graphs. Numerische Mathematik, 1(1):269–271, 1959. https://doi.org/10.1007/BF01386390.

[47] Dariush Ebrahimi, Sanaa Sharafeddine, Pin-Han Ho, and Chadi Assi. Autonomous uav trajectory for localizing ground objects: A reinforcement learning approach. IEEE Transactions on Mobile Computing, 20(4):1312–1324, 2020. https://doi.org/10.1109/TMC.2020.2966989.

[48] Ian Elliott, Christopher Sullivan, Natasha Bosanac, Jeffrey R. Stuart, and Farah Alibay. Designing low-thrust trajectories for a smallsat mission to sun–earth l5. Journal of Guidance, Control, and Dynamics, 43(10):1854–1864, 2020. https://doi.org/10.2514/1.G004993.

[49] Jacob A. Englander, Matthew Vavrina, and Alexander R. Ghosh. Multi-objective hybrid optimal control for multiple-flyby low-thrust mission design. In 25th AAS/AIAA Space Flight Mechanics Meeting, Williamsburg, VA, 2015.

[50] Robert Falck and John Dankanich. Optimization of low-thrust spiral trajectories by collocation. In AIAA/AAS Astrodynamics Specialist Conference, Minneapolis, MN, 2012. https://doi.org/10.2514/6.2012-4423.

[51] Lorenzo Federici, Andrea Scorsoglio, Alessandro Zavoli, and Roberto Furfaro. Autonomous guidance for cislunar orbit transfers via reinforcement learning. In AAS/AIAA Astrodynamics Specialist Conference, Virtual, 2021.

[52] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In 25th International Conference on Machine Learning, Stockholm, SWE, pages 1587–1596. Proceedings of Machine Learning Research, 2018.

[53] Roberto Furfaro and Richard Linares. Waypoint-based generalized zem/zev feedback guidance for planetary landing via a reinforcement learning approach. In 3rd International Academy of Astronautics Conference on Dynamics and Control of Space Systems, pages 401–416, 2017.

[54] Brian Gaudet and Roberto Furfaro. Robust spacecraft hovering near small bodies in environments with unknown dynamics using reinforcement learning. In AIAA/AAS Astrodynamics Specialist Conference, Minneapolis, MN, 2012. https://doi.org/10.2514/6.2012-5072.

[55] John Gittins, Kevin Glazebrook, and Richard Weber. Multi-Armed Bandit Allocation Indices. John Wiley & Sons, 2nd edition, 2011. https://doi.org/10.1002/9780470980033.

[56] Kathryn F. Graham and Anil V. Rao. Minimum-time trajectory optimization of multiple revolution low-thrust earth-orbit transfers. Journal of Spacecraft and Rockets, 52(3):711–727, 2015. https://doi.org/10.2514/1.A33187.

[57] Daniel J. Grebow. Generating periodic orbits in the circular restricted three-body problem with applications to lunar south pole coverage. Master's thesis, Purdue University, West Lafayette, IN, 2016.

[58] Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E. Turner, Bernhard Schölkopf, and Sergey Levine. Interpolated policy gradient: Merging on-policy and off-policy gradient estimation for deep reinforcement learning. In 31st Conference on Neural Information Processing Systems, Long Beach, CA, 2017. https://dl.acm.org/doi/10.5555/3294996.3295141.

[59] Siyu Guo, Xiuguo Zhang, Yisong Zheng, and Yiquan Du. An autonomous path planning model for unmanned ships based on deep reinforcement learning. Sensors, 20(2):426, 2020. https://doi.org/10.3390/s20020426.

[60] Pini Gurfil and N. Jeremy Kasdin. Characterization and design of out-of-ecliptic trajectories using deterministic crowding genetic algorithms. Computer Methods in Applied Mechanics and Engineering, 191(19-20):2169–2186, 2002. https://doi.org/10.1016/S0045-7825(01)00380-2.

[61] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In 35th International Conference on Machine Learning, Stockholm, SWE, volume 80, pages 1861–1870. Proceedings of Machine Learning Research, 2018.

[62] C.R. Hargraves and S.W. Paris. Direct trajectory optimization using nonlinear programming and collocation. Journal of Guidance, Control, and Dynamics, 10(4):338–342, 1987. https://doi.org/10.2514/3.20223.

[63] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics, 4(2):100–107, 1968. https://doi.org/10.1109/TSSC.1968.300136.

[64] John W. Hartmann, Victoria L. Coverstone-Carroll, and Steven N. Williams. Optimal interplanetary spacecraft trajectories via a pareto genetic algorithm. The Journal of the Astronautical Sciences, 46(3):267–282, 1998.

[65] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In Neural Networks for Perception, pages 65–93. Elsevier, 1992. https://doi.org/10.1016/B978-0-12-741252-8.50010-8.

[66] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In 32nd AAAI Conference on Artificial Intelligence, New Orleans, LA, 2018.

[67] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415, 2016.

[68] Mario S. Holubar and Marco A. Wiering. Continuous-action reinforcement learning for playing racing games: Comparing spg to ppo. arXiv preprint arXiv:2001.05270, 2020.

[69] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. Neural Networks, 2(5):359–366, 1989. https://doi.org/10.1016/0893-6080(89)90020-8.

[70] Kathleen C. Howell, Brian T. Barden, and Martin W. Lo. Application of dynamical systems theory to trajectory design for a libration point mission. The Journal of the Astronautical Sciences, 45(2):161–178, 1997. https://doi.org/10.1007/BF03546374.

[71] Bryce Huber. Designing high thrust, interplanetary trajectories using the neuroevolution of augmenting topologies (neat) algorithm. Master's thesis, University of Colorado Boulder, 2019.

[72] Jemin Hwangbo, Inkyu Sa, Roland Siegwart, and Marco Hutter. Control of a quadrotor with reinforcement learning. IEEE Robotics and Automation Letters, 2(4):2096–2103, 2017. https://doi.org/10.1109/LRA.2017.2720851.

[73] Dario Izzo, Christopher Iliffe Sprague, and Dharmesh Vijay Tailor. Machine learning and evolutionary techniques in interplanetary trajectory design. In Modeling and Optimization in Space Engineering, pages 191–210. Springer, 2019. https://doi.org/10.1007/978-3-030-10501-3_8.

[74] M. Jacobi. Sur l'élimination des noeuds dans le problème des trois corps. Astronomische Nachrichten, 20:81, 1842. https://doi.org/10.1002/asna.18430200602.

[75] Erica L. Jenson and Daniel J. Scheeres. Multi-objective optimization of covariance and energy for asteroid transfers. Journal of Guidance, Control, and Dynamics, 44(7):1253–1265, 2021. https://doi.org/10.2514/1.G005609.

[76] Jet Propulsion Laboratory. Distributed Spacecraft Technology Website, 2019. Accessed: December 10, 2019.

[77] Xingji Jin, Timo Pukkala, and Fengri Li. Fine-tuning heuristic methods for combinatorial optimization in forest planning. European Journal of Forest Research, 135(4):765–779, 2016. https://doi.org/10.1007/s10342-016-0971-x.

[78] Brett A. King, Tyler Hammond, and Jake Harrington. Disruptive technology: Economic consequences of artificial intelligence and the robotics revolution. Journal of Strategic Innovation and Sustainability, 12(2):53–67, 2017. https://doi.org/10.33423/jsis.v12i2.801.

[79] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In 3rd International Conference on Learning Representations, San Diego, CA, 2015.

[80] Vijay R. Konda and John N. Tsitsiklis. On actor-critic algorithms. SIAM Journal on Control and Optimization, 42(4):1143–1166, April 2003. https://doi.org/10.1137/S0363012901385691.

[81] Wang Sang Koon, Martin W. Lo, Jerrold E. Marsden, and Shane D. Ross. Dynamical Systems, The Three-Body Problem and Space Mission Design. Marsden Books, 2006.

[82] Kristián Kovalskỳ and George Palamas. Neuroevolution vs reinforcement learning for training non player characters in games: The case of a self driving car. In International Conference on Intelligent Technologies for Interactive Entertainment, Virtual, pages 191–206. Springer, 2020.

[83] Nicholas B. LaFarge, Kathleen C Howell, and David C. Folta. An autonomous stationkeeping strategy for multi-body orbits leveraging reinforcement learning. In 32nd AIAA/AAS Space Flight Mechanics Meeting, San Diego, CA, 2022. http://dx.doi.org/10.2514/6.2022-1764.

[84] Nicholas B. LaFarge, Daniel Miller, Kathleen C. Howell, and Richard Linares. Autonomous closed-loop guidance using reinforcement learning in a low-thrust, multi-body dynamical environment. Acta Astronautica, 186:1–23, 2021. https://doi.org/10.1016/j.actaastro.2021.05.014.

[85] Seungwon Lee, Paul von Allmen, Wolfgang Fink, A. E. Petropoulos, and R. J. Terrile. Multi-objective evolutionary algorithms for low-thrust orbit transfer optimization. In Genetic and Evolutionary Computation Conference, Washington, DC, 2005.

[86] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. arXiv preprint arXiv:2005.01643, 2020.

[87] Kai Li, Wei Ni, Eduardo Tovar, and Mohsen Guizani. Joint flight cruise control and data collection in uav-aided internet of things: An onboard deep reinforcement learning approach. IEEE Internet of Things Journal, 2020. https://doi.org/10.1109/JIOT.2020.3019186.

[88] Paul A. Lightsey, Charles B. Atkinson, Mark C. Clampin, and Lee D. Feinberg. James webb space telescope: Large deployable cryogenic telescope in space. Optical Engineering, 51(1):011003, 2012. https://doi.org/10.1117/1.OE.51.1.011003.

[89] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971, 2015.

[90] Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad E. Alsaadi. A survey of deep neural network architectures and their applications. Neurocomputing, 234:11–26, 2017. https://doi.org/10.1016/j.neucom.2016.12.038.

[91] Daniel J. Lizotte, Michael H. Bowling, and Susan A. Murphy. Efficient reinforcement learning with multiple reward functions for randomized controlled trial analysis. In 27th International Conference of Machine Learning, Haifa, ISR, page 695–702. Omnipress, 2010.

[92] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In International Conference on Learning Representations, New Orleans, LA, 2019.

[93] John Martin and Hanspeter Schaub. Reinforcement learning and orbit-discovery enhanced by small-body physics-informed neural network gravity models. In 32nd AIAA/AAS Space Flight Mechanics Meeting, San Diego, CA, 2022.

[94] MATLAB Optimization Toolbox Version 8.2, 2018. The MathWorks, Natick, MA, USA.

[95] Leslie McNutt, Les Johnson, Pater Kahn, Julie Castillo-Rogez, and Andreas Frick. Near earth asteroid (nea) scout. In AIAA Space 2014 Conference and Exposition, San Diego, CA, 2014. https://doi.org/10.2514/6.2014-4435.

[96] Daniel Miller and Richard Linares. Low-thrust optimal control via reinforcement learning. In 29th AAS/AIAA Space Flight Mechanics Meeting, Ka'anapali, HI, 2019.

[97] Marvin Minsky. Steps toward artificial intelligence. in Proceedings of the IRE, 49(1):8–30, 1961. https://doi.org/10.1109/JRPROC.1961.287775.

[98] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei Rusu, Joel Veness, Marc Bellemare, Alex Graves, Martin Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. Nature, 518:529–533, 02 2015. https://doi.org/10.1038/nature14236.

[99] Thomas M. Moerland, Joost Broekens, and Catholijn M. Jonker. Model-based reinforcement learning: A survey. arXiv preprint arXiv:2006.16712, 2020.

[100] Hossam Mossalam, Yannis M. Assael, Diederik M. Roijers, and Shimon Whiteson. Multi-objective deep reinforcement learning. arXiv preprint arXiv:1610.02707, 2016.

[101] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In 27th International Conference on Machine Learning, Haifa, ISR, page 807–814. Omnipress, 2010.

[102] NASA. Evolutionary Mission Trajectory Generator (EMTG), May 2020. https://github.com/nasa/EMTG.

[103] Sriraam Natarajan and Prasad Tadepalli. Dynamic preferences in multi-criteria reinforcement learning. In 22nd International Conference on Machine Learning, Bonn, DEU, page 601–608. Association for Computing Machinery, 2005. https://doi.org/10.1145/1102351.1102427.

[104] Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In 16th International Conference on Machine Learning, Bled, SVN, page 278–287, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[105] Thanh Thi Nguyen, Ngoc Duy Nguyen, Peter Vamplew, Saeid Nahavandi, Richard Dazeley, and Chee Peng Lim. A multi-objective deep reinforcement learning framework. Engineering Applications of Artificial Intelligence, 96:1–12, 2020. https://doi.org/10.1016/j.engappai.2020.103915.

[106] Chigozie Enyinna Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning. arXiv preprint arXiv:1811.03378, 2018.

[107] Mike R. Osborne. On shooting methods for boundary value problems. Journal of Mathematical Analysis and Applications, 27(2):417–433, 1969. https://doi.org/10.1016/0022-247X(69)90059-6.

[108] M. T. Ozimek and K. C. Howell. Low-thrust transfers in the earth-moon system, including applications to libration point orbits. Journal of Guidance, Control, and Dynamics, 33(2):533–549, 2010. https://doi.org/10.2514/1.43179.

[109] Aleksandr I. Panov, Konstantin S. Yakovlev, and Roman Suvorov. Grid path planning with deep reinforcement learning: Preliminary results. Procedia Computer Science, 123:347–353, 2018. https://doi.org/10.1016/j.procs.2018.01.054.

[110] Jung-Jun Park, Ji-Hun Kim, and Jae-Bok Song. Path planning for a robot manipulator based on probabilistic roadmap and reinforcement learning. International Journal of Control, Automation, and Systems, 5(6):674–680, 2007.

[111] Nathan Luis Olin Parrish. Low Thrust Trajectory Optimization in Cislunar and Translunar Space. PhD thesis, University of Colorado Boulder, 2018.

[112] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In 33rd Conference on Neural Information Processing Systems, Vancouver, CAN, pages 8024–8035. Curran Associates Inc., 2019.

[113] Daniel Pérez-Palau and Richard Epenoy. Fuel optimization for low-thrust earth–moon transfer via indirect optimal control. Celestial Mechanics and Dynamical Astronomy, 130(2):1–29, 2018. https://doi.org/10.1007/s10569-017-9808-2.

[114] Bonnie J. Prado Pino. Energy-Informed Strategies For Low-Thrust Trajectory Design in Cislunar Space. PhD thesis, Purdue University Graduate School, 2020.

[115] Robert E. Pritchett. Numerical Methods for Low-Thrust Trajectory Optimization. PhD thesis, Purdue University, 2016.

[116] Maziar Raissi, Paris Perdikaris, and George E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics, 378:686–707, 2019. https://doi.org/10.1016/j.jcp.2018.10.045.

[117] David L. Richardson. Halo orbit formulation for the isee-3 mission. Journal of Guidance and Control, 3(6):543–548, 1980. https://doi.org/10.2514/3.56033.

[118] Craig Roberts, Sara Case, John Reagoso, and Cassandra Webster. Early mission maneuver operations for the deep space climate observatory sun-earth l1 libration point mission. In AIAA/AAS Astrodynamics Specialist Conference, Vail, CO, 2015.

[119] Gavin A Rummery and Mahesan Niranjan. On-line Q-learning Using Connectionist Systems, volume 37. University of Cambridge, Department of Engineering, 1994.

[120] Ryan P. Russell. Primer vector theory applied to global low-thrust trade studies. Journal of Guidance, Control, and Dynamics, 30(2):460–472, 2007. https://doi.org/10.2514/1.22984.

[121] Wayne A. Scheel and Bruce A. Conway. Optimization of very-low-thrust, many-revolution spacecraft trajectories. Journal of Guidance, Control, and Dynamics, 17(6):1185–1192, 1994. https://doi.org/10.2514/3.21331.

[122] Enrico Schiassi, Andrea D'Ambrosio, Kristofer Drozd, Fabio Curti, and Roberto Furfaro. Physics-informed neural networks for optimal planar orbit transfers. Journal of Spacecraft and Rockets, pages 1–16, 2022. https://doi.org/10.2514/1.A35138.

[123] W.K.H. Schmidt and V. Bothmer. Stereoscopic viewing of solar coronal and interplanetary activity. Advances in Space Research, (4):369–376, 1996. https://doi.org/10.1016/0273-1177(95)00602-B.

[124] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In 32nd International Conference on Machine Learning, Lille, FRA, 2015.

[125] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. arXiv preprint arXiv:1506.02438, 2015.

[126] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.

[127] Andrea Scorsoglio, Roberto Furfaro, Richard Linares, and Mauro Massari. Actor-critic reinforcement learning approach to relative motion guidance in near-rectilinear orbit. In 29th AAS/AIAA Space Flight Mechanics Meeting, Ka'anapali, HI, 2019.

[128] Vishwa Shah, Ryne Beeson, and Victoria L. Coverstone. Automated global optimization of impulsive trajectories in three-body problems. In AIAA/AAS Astrodynamics Specialist Conference, Long Beach, CA, 2016. https://doi.org/10.2514/6.2016-5438.

[129] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe1, John Nham, Nal Kalchbrenner, Timothy Lillicrap,

Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. Nature, 529(7587):484–489, 2016. https://doi.org/10.1038/nature16961.

[130] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. Nature, 550(7676):354–359, 2017. https://doi.org/10.1038/nature24270.

[131] Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications, pages 369 – 386. International Society for Optics and Photonics, SPIE, 2019. https://doi.org/10.1117/12.2520589.

[132] Marshall Smith, Douglas Craig, Nicole Herrmann, Erin Mahoney, Jonathan Krezel, Nate McIntyre, and Kandyce Goodliff. The artemis program: An overview of nasa's activities to return humans to the moon. In 2020 IEEE Aerospace Conference, Big Sky, MT, pages 1–10. IEEE, 2020. https://doi.org/10.1109/AERO47225.2020.9172323.

[133] Thomas R. Smith and Natasha Bosanac. Using motion primitives to design libration point orbit transfers in the earth-moon system. In AAS/AIAA Astrodynamics Specialist Conference, Virtual, 2021.

[134] Yunlong Song, Mats Steinweg, Elia Kaufmann, and Davide Scaramuzza. Autonomous drone racing with deep reinforcement learning. In IEEE/RSJ International Conference on Intelligent Robots and Systems, Prague, CZE, 2021.

[135] V. Spandan, P. Bharadwaj, M. Bassenne, and L. Jofre. Scalar source tracking in turbulent environments using deep reinforcement learning. In Proceedings of the 2018 Summer Program, pages 155–164. Center for Turbulence Research, 2018.

[136] Nidamarthi Srinivas and Kalyanmoy Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. Evolutionary Computation, 2(3):221–248, 1994. https://doi.org/10.1162/evco.1994.2.3.221.

[137] Jeffrey R. Stuart. A Hybrid Systems Strategy for Automated Spacecraft Tour Design and Optimization. PhD thesis, Purdue University, 2014.

[138] Jeffrey R. Stuart, Kathleen C. Howell, and Roby Wilson. Automated design of propellant-optimal, end-to-end, low-thrust trajectories for trojan asteroid tours. Journal of Spacecraft and Rockets, (5):1631–1647, 2014. http://dx.doi.org/10.2514/1.A32748.

[139] Jeffrey R. Stuart, Martin Ozimek, and Kathleen C. Howell. Optimal, low-thrust, path-constrained transfers between libration point orbits using invariant manifolds. In AIAA/AAS Astrodynamics Specialist Conference, Toronto, CAN, 2010.

[140] Christopher J. Sullivan and Natasha Bosanac. Using multi-objective deep reinforcement learning to uncover a pareto front in multi-body trajectory design. In AAS/AIAA Astrodynamics Specialist Conference, Virtual, 2020.

[141] Christopher J. Sullivan, Natasha Bosanac, Rodney L. Anderson, and Alinda Mashiku. Exploring the low-thrust transfer design space in an ephemeris model via multi-objective reinforcement learning. In 32nd AIAA/AAS Space Flight Mechanics Meeting, San Diego, CA, 2022. https://doi.org/10.2514/6.2022-1887.

[142] Christopher J. Sullivan, Natasha Bosanac, Rodney L. Anderson, Alinda K. Mashiku, and Jeffrey R. Stuart. Exploring transfers between earth-moon halo orbits via multi-objective reinforcement learning. In IEEE Aerospace Conference, Virtual, 2020. https://doi.org/10.1109/AERO50100.2021.9438267.

[143] Christopher J. Sullivan, Natasha Bosanac, Alinda K. Mashiku, and Rodney L. Anderson. Multi-objective reinforcement learning for low-thrust transfer design between libration point orbits. In AAS/AIAA Astrodynamics Specialist Conference, Virtual, 08 2021.

[144] Christopher J. Sullivan, Jeffrey Stuart, Rodney L. Anderson, and Natasha Bosanac. Designing low-thrust transfers to high-inclination science orbits via hybrid optimization. Journal of Spacecraft and Rockets, 58(5):1339–1351, 2021. https://doi.org/10.2514/1.A34980.

[145] Richard S. Sutton. Integrated modeling and control based on reinforcement learning and dynamic programming. In Advances in Neural Information Processing Systems 3, Denver, CO, 1990.

[146] Richard S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction. MIT press, Cambridge, MA, 2nd edition, 2018.

[147] Grzegorz Swirszcz, Wojciech Marian Czarnecki, and Razvan Pascanu. Local minima in training of deep networks. arXiv preprint arXiv:1611.06310, 2016.

[148] Victor Szebehely. Theory of Orbits: The Restricted Problem of Three Bodies. Academic Press, New Haven, CT, 1967. https://doi.org/10.1016/B978-0-12-395732-0.X5001-6.

[149] Ehsan Taheri, Ilya Kolmanovsky, and Ella Atkins. Enhanced smoothing technique for indirect optimization of minimum-fuel low-thrust trajectories. Journal of Guidance, Control, and Dynamics, 39(11):2500–2511, 2016. http://dx.doi.org/10.2514/1.G000379.

[150] Gao Tang and Fanghua Jiang. Capture of near-earth objects with low-thrust propulsion and invariant manifolds. Astrophysics and Space Science, 361(1):1–14, 2016. https://doi.org/10.1007/s10509-015-2592-0.

[151] Matthew E. Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. Journal of Machine Learning Research, 10(56):1633–1685, 2009.

[152] Gerald Tesauro, Rajarshi Das, Hoi Chan, Jeffrey Kephart, David Levine, Freeman Rawson, and Charles Lefurgy. Managing power consumption and performance of computing systems using reinforcement learning. In 21st Conference on Neural Information Processing Systems, Vancouver, CAN, pages 1497–1504, 2007.

[153] Chen Tessler, Yonathan Efroni, and Shie Mannor. Action robust reinforcement learning and applications in continuous control. In 36th International Conference on Machine Learning, Long Beach, CA, pages 6215–6224. Proceedings of Machine Learning Research, 2019.

[154] Thanh-Trung Trinh, Dinh-Minh Vu, and Masaomi Kimura. Point-of-conflict prediction for pedestrian path-planning. In 12th International Conference on Computer Modeling and Simulation, Brisbane, AUS, page 88–92, 2020. https://doi.org/10.1145/3408066.3408079.

[155] Alexander Trott, Stephan Zheng, Caiming Xiong, and Richard Socher. Keeping your distance: Solving sparse reward tasks using self-balancing shaped rewards. In 33rd Conference on Neural Information Processing Systems, Vancouver, CAN. Curran Associates Inc., 2019.

[156] David A. Vallado. Fundamentals of Astrodynamics and Applications. Microcosm Press, El Segundo, CA, 4th edition, 2013.

[157] Peter Vamplew, Richard Dazeley, Adam Berry, Rustam Issabekov, and Evan Dekker. Empirical evaluation methods for multiobjective reinforcement learning algorithms. Machine learning, 84(1):51–80, 2011. https://doi.org/10.1007/s10994-010-5232-5.

[158] Kristof Van Moffaert and Ann Nowé. Multi-objective reinforcement learning using sets of pareto dominating policies. Journal of Machine Learning Research, 15(1):3483–3512, 2014.

[159] Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. arXiv preprint arXiv:1707.08817, 2017.

[160] Angelos Vourlidas. Mission to the sun-earth l5 lagrangian point: An optimal platform for space weather research. Space Weather, pages 197–201, 2015. https://doi.org/10.1002/2015SW001173.

[161] Christopher J. C. H. Watkins. Learning from Delayed Rewards. PhD thesis, 1989.

[162] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning, 8(3):229–256, 1992. https://doi.org/10.1007/BF00992696.

[163] Chao Yan, Xiaojia Xiang, and Chang Wang. Towards real-time path planning through deep reinforcement learning for a uav in dynamic environments. Journal of Intelligent & Robotic Systems, 98(2):297–309, 2020. https://doi.org/10.1007/s10846-019-01073-3.

[164] Collin E. York, Kathleen C. Howell, and Belinda Marchand. Convergence basin analysis for spacecraft trajectory targeting. In AIAA/AAS Astrodynamics Specialist Conference, Virtual, 2021.

[165] Baochang Zhang, Zhili Mao, Wanquan Liu, and Jianzhuang Liu. Geometric reinforcement learning for path planning of uavs. Journal of Intelligent & Robotic Systems, 77(2):391–409, 2015. https://doi.org/10.1007/s10846-013-9901-z.

[166] Zhenyu Zhang, Xiangfeng Luo, Tong Liu, Shaorong Xie, Jianshu Wang, Wei Wang, Yang Li, and Yan Peng. Proximal policy optimization with mixed distributed training. In IEEE 31st International Conference on Tools with Artificial Intelligence, Portland, OR, 2019. https://doi.org/10.1109/ICTAI.2019.00206.

[167] Zhenpeng Zhou, Steven Kearnes, Li Li, Richard N. Zare, and Patrick Riley. Optimization of molecules via deep reinforcement learning. Scientific Reports, 9(1):1–10, 2019. https://doi.org/10.1038/s41598-019-47148-x.

[168] Zhuangdi Zhu, Kaixiang Lin, and Jiayu Zhou. Transfer learning in deep reinforcement learning: A survey. arXiv preprint arXiv:2009.07888, 2020.